CS1132 Fall 2012 Assignment 1b

> *Adhere to the Code of Academic Integrity.* You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 3 of this document.

Do not use the `break` or `continue` statement in any homework or test in CS1132.

1 Population Dynamics

[The following description is taken from "Mathematical Models in Population Biology and Epidemiology" by Brauer and Castillo-Chavez, 2001.]

Consider a population that is divided into a finite number of age classes labeled from 0 to m. One method for describing the number of members in each age class as a function of time is the *linear discrete-time model* for population growth. In such a model, we let $\alpha_{j,n}$ denote the number of members in the jth class at the nth time. We assume that the length of time spent in each age class is the same. Then $\alpha_{j,n+1}$, the number of members in the j'th age class at the $(n+1)$st time, is equal to $\alpha_{j+1,n}$ minus the number of members of this age cohort who die before entering the next age class. We assume that the probability of survival from one age class to the next depends only on age. Let p_j be the probability that a member of the jth age class survives until the $(j+1)$st age class. All new members recruited into the population are assumed to come from a birth process, with fecundity depending only on age. Assume that there are constants $\beta_0, \beta_1, \ldots, \beta_m$ such that

$$\alpha_{0,n+1} = \beta_0\alpha_{0,n} + \beta_1\alpha_{1,n} + \ldots + \beta_m\alpha_{m,n}.$$

If we define

$$\vec{\alpha}_n = \begin{pmatrix} \alpha_{0,n} \\ \alpha_{1,n} \\ \vdots \\ \alpha_{m,n} \end{pmatrix}$$

and define the *Leslie matrix* to be

$$A = \begin{pmatrix} \beta_0 & \beta_1 & \beta_2 & \ldots & \beta_{m-1} & \beta_m \\ p_0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & p_1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & p_2 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & p_{m-1} & 0 \end{pmatrix}$$

then the change in the population through time can be described by the vector difference equation

$$\vec{\alpha}_{n+1} = A\vec{\alpha}_n.$$

In the above equation, $A\vec{\alpha}_n$ is the multiplication of a matrix and a vector, which results in a vector. This operation will be explained below. Let Q_n be the total population at time n. Then the vector

$$\frac{\vec{\alpha}_n}{Q_n}$$

gives the fraction of the population in each age class at time n.

To do: Write a script `simPopulation` that simulates a population according to a given Leslie matrix and an initial population.

1. Implement the following function to generate the Leslie matrix:

```
function A = getLeslieMatrix(m)
% A is an (m+1)-by-(m+1) Leslie matrix with m survival probabilities on the
% first subdiagonal linearly spaced between [0.9, 0.1] and uniform random
% integer fecundities between [1,5] on the first row. All other elements are zeros.
```

   For example, `A = getLeslieMatrix(5)` might return

$$\texttt{A} = \begin{bmatrix} 1 & 5 & 4 & 2 & 1 & 1 \\ 0.9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 \end{bmatrix}$$

   Use built-in function `rand`—do not use `randi`. Another useful function is `linspace`.

2. Implement the following function to perform matrix multiplication:

```
function  w = matVecMult(L,v)
%L is an m-by-m matrix of numbers; v is a length m column vector of numbers.
% w is a column vector such that w(i) is
%    L(i,1)*v(1) + L(i,2)*v(2) + ... + L(i,m)*v(m)
% Use SCALAR multiplication operations only, i.e., do not use MATLAB's
% matrix (vector) multiplication facility.
```

3. Write a script `simPopulation` that simulates a population with seven age-classes for 50 time steps, where the initial population size is five individuals in age-class zero. Use your `getLeslieMatrix` function to generate your Leslie matrix for this population. At each time step, use your `matVecMult` function to compute the new population vector and record the fractions of the population in each age-class, i.e., record the vector $\frac{\vec{\alpha}_n}{Q_n}$ (hint: matrix concatenation may be useful). Plot the fractions of all seven age-classes through time on the same figure and display a legend. Add a descriptive title to your plot and label the x-axis as "time" and the y-axis as "fraction in each age class."

   Graphics help ... The skeleton code below shows one way of producing several graphs in one figure and displaying a legend:

```
textArray= cell(numAgeClasses,1); % Initialize a cell array to store legend text
hold all                          % Hold subsequent plot commands on current axes and keep
                                  %   cycling through the default plot colors
for i = 1:numAgeClasses
    plot( _______ , _______ )                  % Make ith plot
    textArray{i}= sprintf('Age Class %d', i);  % Make and store ith label
end
legend(textArray)  % Display legend with the text stored in textArray
hold off           % Subsequent plot commands refresh axes
```

   Variable `numAgeClasses` in the skeleton above is the number of age-classes. Cell arrays will be discussed in detail in Module 2.

2 Self-check list

The following is a list of the minimum *necessary* criteria that your assignment must meet in order to be considered *satisfactory*. Failure to satisfy any of these conditions will result in an immediate request to

resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time. Although all these criteria are necessary, meeting all of them might still not be *sufficient* to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- Δ Comment your code! Make sure your functions are properly commented, regarding function purpose and input/output parameters.
- Δ Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- Δ Make sure your functions names are *exactly* the ones we have specified, *including* case.
- Δ Check that the number and order of input and output parameters for each of the functions match exactly the specifications we have given.
- Δ Test each one of your functions independently, whenever possible, or write short scripts to test them.
- Δ Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite loops. Check your script several times in a row. Before each test run, type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

3 Submission instructions

1. Upload files `getLeslieMatrix.m`, `matVecMult.m`, and `simPopulation.m` to CMS in the submission area corresponding to Assignment 1b in CMS before the deadline. Late submission is accepted up to 24 hours after the deadline with a 10% penalty.
2. *After grading:* If you resubmit the assignment, upload your corrected files and *be sure to select the* ***Regrade Request*** *option.*