MATH 200C Linear Algebra

Project 2: Age – Class Population Models

DUE: Monday, May 11

The *age distribution* of the human population in many countries is an important current social issue. Will there be enough individuals of working age paying taxes, for example, to support a burgeoning number of aging retired folk dependent on Social Security? How can we effectively plan for new schools at the elementary, middle and high school levels? Will adequate housing stocks be available for younger families?

In Part I, you will build in MATLAB a simple model which forecasts an age structured population many years into the future. The assumptions are based on the premise that certain demographic parameters remain constant over an extended period of the time. You will be initially employing the **Leslie Matrix model** but eventually examining similar models and creating your own model.

The Leslie Matrix is mathematical tool widely used in ecological and demographic studies. It was initially created by Patrick Holt Leslie (1900 – 1972), a Scottish physiologist best known for his contributions to population dynamics. [Leslie, P.H. (1945) "The use of matrices in certain population mathematics," *Biometrika*, 33(3), 183–212.]

The main idea is to split a population into several distinct *age classes* and model how individuals are created and subsequently move from one age class to the next. In the Leslie model every member of one age class either dies or moves into the next age class at every iteration of the model. In the model you will eventually be constructing, individuals will experience one of three possibilities: death, stasis (remaining in the same class) and maturation (moving on to the next class).

Part I: The Leslie Matrix

Focusing initially on the Leslie model, there are two parameters associated with each age class. The *percent survival rate* for the age class represents the proportion of the individuals in an age class which survive to move into the next age class during the next iteration of the model. The second parameter is the *fecundity parameter* which represents how many offspring (age class zero) are produced, on average, by each member of the age class. Fecundity is most closely related to the number of *females* in the population and thus Leslie models typically count only the females in a population.

Let's introduce some mathematical notation. Suppose $P(i)$ represents the population of age class i. If $s(i)$ represents the survival rate of age class i, then the population in age class $i+1$ in the next iteration can be computed using the current population in age class i through the formula

next iteration's $P(i+1) = s(i)*P(i)$.

If we let N be the last age class in our model (all individuals in age class N die before the next iteration) then the formula above works for values of i from 0 up to $N-1$, inclusive. This still leaves us without a population for the next iteration's youngest age class, age class 0.

We use the fecundity parameters to determine how many newborns are created during the next iteration of the model. If we let $f(i)$ be the fecundity (average birth rate) per individual in age class i, then the total number of newborns will be:

(1) $f(0)*P(0) + f(1)*P(1) + f(2)*P(2) + \ldots + f(N)*P(N)$

Task A:

Explain, in your own words, why formula (1) is valid.

If P is a *column* vector of population by age class, then it is possible to compute the P vector for the next iteration through a simple matrix multiplication. In particular, if A is the $(N+1) \times (N+1)$ matrix whose first row is the fecundity rates $f(i)$ and whose diagonal below the main diagonal is made up of the survival rates $s(i)$, then the next iteration's population column vector can be found by performing the matrix multiplication AP. [in MATLAB A * P]

Task B:

In MATLAB, create the matrix *A* and vector *P* described below and determine *AP*

As an example, suppose we have a population of critters who never grow past five years old. Suppose, further, that 80% of the critters survive each year to move onto the next age class, with the exception of the five year old critters who always die. If newborns and yearlings have no offspring, and if middle aged critters (2, 3 and 4 year olds) have, on average, 0.35 newborns each year, while the oldest critters (5 year olds) have only 0.1 newborns on average, how can we use current populations to forecast future populations and the age structure of that population?

In this model the parameters are given in the table below:

Age Class	Survival Rate	Birthrate
i	*s(i)*	*f(i)*
0	0.8	0.00
1	0.8	0.00
2	0.8	0.35
3	0.8	0.35
4	0.8	0.35
5	–	0.10

To create the matrix A mentioned above for this particular set of parameters, the MATLAB command below would do the trick. (Note the ellipses at the end of some lines. This is how we enter a MATLAB command which takes more than one line to complete.)

```
A = [ 0.00 0.00 0.35 0.35 0.35 0.10; 0.8 0 0 0 0 0; ...
0 0.8 0 0 0 0; 0 0 0.8 0 0 0; 0 0 0 0.8 0 0; ...
0 0 0 0 0.8 0]
```

which creates the matrix *A* shown below:

```
A    =

    0.0000    0.0000    0.3500    0.3500    0.3500    0.1000
    0.8000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.8000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.8000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.8000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.8000    0.0000
```

To forecast the future age structured population of our colony of critters we need only determine the current age structured population, represent this population as a column vector *P* and then move forward one year at a time in our model by performing the multiplication *A*P*.

For example, if our initial population consists of 60 critters evenly spread throughout the six age classes, our initial *P* vector would be:

```
P = [10;10;10;10;10;10]
```

Note that the semicolons cause *P* to be a column vector. We could have achieved the same end by separating the 10's with spaces or commas, creating a row vector, and then using the matrix transform operator (an apostrophe).

To determine the population distribution next year we multiply *A* and *P*, obtaining the vector

```
   11.5000
    8.0000
    8.0000
    8.0000
    8.0000
    8.0000
```

Task C:

Using the data above, find the population 10 years from now using both methods described below.

If we want the population two years in the future there are a number of ways to proceed. First, we can take the vector above and multiply it by *A* to get the population for the next iteration. Proceeding in this fashion we get a series of vectors, each of which gives the population for the succeeding iteration.

If we only want populations for specific future years a second way may make more sense. This method is based on noting that for each year into the future we go, we arrive at the population by multiplying by the matrix *A*. Thus, if we want to go ten years into the future we get there by

multiplying by *A* ten times. We can simplify this process by using the matrix exponentiation operator carat (^).

To move ten years into the future we start with the current population (held in the column vector *P*) and multiply it by *A* ten times. This could be accomplished with the MATLAB command

```
A*A*A*A*A*A*A*A*A*A*P
```

There is an easier way however. Note that in the computation above, we multiply the matrix *A* by itself 10 times. We can use the exponentiation operator (^) as a way to indicate repeated matrix multiplications without writing them all out. Using matrix exponentiation, the MATLAB command given above can be recast as:

```
A^10*P
```

Task D:

Using the for loop concept discussed below, determine the population distribution after each of the first 10 years.

If we want to display the population distribution for each year in the next decade, we could type in the commands

```
A*P
A^2* P
…
A^9*P
A^10*P
```

Note that this would be an impractical way to display the distribution for each of the next 100 years as we would have retype essentially the same command 100 times with the only difference being the value of the exponent *k* in the A^*k*. This approach is also computationally inefficient. If there is a separate command to MATLAB to compute A^10, it has to multiply the matrix A by itself 10 times. If MATLAB already knows A^9 and has stored this value somewhere, then it can compute A^10 with a single multiplication A^9 * A.

All computer programming languages provide for **LOOP** structures which give quick ways to instruct a program to repeat essentially the same command many times with only a minor change at each step. Here we'll illustrate the **for** loop in MATLAB.

```
NumberOfYears = 10
Q = P
For Year = 1 : NumberOfYears
    Year
    Q = A * Q
end
```

All three approaches should give you the same population distribution vector after 10 years:

3.0288
2.7583
2.4885
2.3087
2.1601
1.9545

Note that the total population after a decade has shrunk from 60 to less than 15, a result which leads us to be very concerned about the long term viability of this population.

Task E:

Complete the 6 exercises listed below:

1. Why is the following MATLAB statement relevant to the above model? What value does it compute for us?

   ```
   ones(1,6)*(A^10*P)
   ```

2. What MATLAB expression would give us the total number of individuals, regardless of age, in the critter population ten years hence?

3. Modify the **for** loop so that it also outputs the total population after each year.

4. How would you quickly create a column vector showing the proportion of the total population in each of the age classes? This kind of information is very interesting in contexts such as the future social security system. Statements such as, "there will be only three workers for every retiree in the year 2040." come from this kind of analysis.

5. Suppose a fertility drug is found which can increase the fecundity of two year olds, but leads to early deaths for older animals. In particular, suppose the drug increases the fecundity of two year old individuals to 2.3 births per year while the survival rates for individuals in age class 3 and 4 decrease from 0.8 to 0.6. An obvious question is whether this will increase the population of critters or not.

   To investigate this question we must update the entries in the *A* matrix to reflect the hypothetical introduction of the new drug. Only three entries in the matrix *A* need be changed, so rather than recreate *A* from scratch we will make the three individual changes.

   The first change is that the birth rate for two year olds must be changed to 2.3. Since the birth rate for two year olds is found in the first row, third column of the matrix *A*, this change can be effected by the MATLAB command:

   ```
   A(1,3)=2.3;
   ```

Make the remaining two changes in *A*, changes which reflected the lowered survival rates for age classes 3 and 4.

Begin with the same initial population (all 10's) as before. Does the population seem healthier (whatever that means) under the drug regime? How does the drug effect the age distribution of the population?

6. For this exercise we go back to the original Leslie setup, ignoring the changes you may have made in the previous exercise.

   If fecundity rates remain unchanged, what survival rates are needed for our critter population to remain stable? What proportion of a stable population falls into each age class?

Task F:

Build a matrix whose columns display the populations in the various age groups in each of 10 successive years. The first column should show the initial population. The second column shows the population after one year; the third column the population after two years, etc.

Call the matrix *TimePop*. Then column 1 is P, column 2 is AP, column 3 is AAP, etc. You can easily build this matrix by adding one line to the body of the **for** loop created earlier by using a clever feature of MATLAB:

Suppose you have a 2 by 3 matrix *M* and you wish to append a third column X to build a 2 by 4 matrix called *N*. The command N = [M X] will do this. For example, if $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ and $X = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$, then $N = [\, M\ X] = \begin{bmatrix} 1 & 2 & 3 & 7 \\ 4 & 5 & 6 & 8 \end{bmatrix}$. What does $N = [M\ X\ X]$ produce?

Task G:

It is fun to see how populations evolve over time. Toward this end it is possible to create a three dimensional graph showing the evolution of a Leslie population over time by creating the *TimePop* matrix whose columns are "snapshots" of the population at various times under study.

The evolution of the critter population over time can be viewed by creating a three dimensional surface plot of the array *TimePop*. To create this surface plot enter the following MATLAB commands and note the plot which is generated. Pay special attention to the creation of labels and the colorbar.

```
surf(TimePop)
view(0,90)
colormap(jet)
```

```
colorbar
xlabel('Year')
ylabel('Age Class')
```

Carry out the instructions to obtain a plot of *TimePop*

Task H:

Plot the populations in a single year and plot the population of a particular age group over the 10 year period.

The colon (:) notation we discussed earlier is a convenient way to create large arrays where the entries are evenly spaced. A second use of the notation is to extract parts of existing matrices. In particular, a single colon is a reference to every row or every column in a matrix, depending on its position.

For example, the notation A(2,:) refers to every column of the second row of the matrix *A*, or, said another way, A(2,:) is the entire second row of *A*.

Similarly, B(:,5) refers to the entire fifth column of the matrix *B*.

1. Continuing with the Leslie population model discussed above, what is the result of this MATLAB command?

```
plot(TimePop(3,:))
```

2. What MATLAB command would produce a plot of the population during the sixth time step? Display the plot.

3. What MATLAB command would produce a plot showing the populations during both the third and sixth time steps? Display the plot.

4. Use MATLAB to plot the population in the youngest age group over a 10 year period.

Using calculators it is difficult to efficiently deal with Leslie models consisting of more than about ten age classes. Using MATLAB, however, it is relatively easy to deal with populations with dozens of age classes. Human populations are generally modeled with 101 age classes, allowing for ages between 0 and 100. It is also much easier to move far into the future, spotting long term trends.

Part II: The Lefkovitch Model

Modifying the Leslie Model

Here we discuss modifications to the Leslie Population Model which describe a slightly different situation than that modeled by Leslie. This section introduces no new MATLAB skills but demonstrates how you might go about refining an existing model to produce a more realistic one.

We'll describe the mathematics of the new model, but will leave it entirely up to you to implement the model in MATLAB.

In the Leslie model only two things can happen to individuals in an age class; they can die or they can move onto the next age class. A similar model which replaces age classes with ontogenetic stages is called a Lefkovitch matrix, whereby individuals can both remain in the same stage class or move on to the next one.

Given that it is possible to remain in the same class for several time steps, it is probably best to leave off the word *age* when describing these classes.

Leonard P. Lefkovitch (1929 – 2010) was a research scientist working for Agriculture Canada. He was a graduate of the University of London, and a past member of the Royal Army Medical Corp. He was also a talented professional musician and composer. Lefkovitch introduced such a stage – structured population matrix model in a 1965 paper "The Study of Population Growth in Organisms Grouped by Stages" (*Biometrics,, Vol. 21,* No. 1 (Mar., 1965), pp. 1-18.). This influential study extended the age-based Leslie matrix to allow for population modeling based on life stages (for example egg, larva, pupa, adult) or sizes.

The Lefkovitch model requires three sets of parameters. Denote the fecundity of class i with the symbol $f(i)$, the proportion of class i moving on to the next class at each iteration with the symbol $s(i)$ and the proportion of each class which survives but remains behind with the symbol $r(i)$.

Proceeding in a fashion reminiscent of the Leslie model development, if we create a model with classes numbered from zero to N, and let $P(i)$ denote the number of individuals in class i, then we can derive the following equations which give the values of the various $P(i)$ for the next time step in terms of the $P(i)$ values for the current time step.

Considering, first, the values of $P(i)$ in the next time step for classes numbered from one to N we see:

$$\textit{Next period's } P(i+1) = s(i)*P(i) + r(i+1)*P(i+1)$$

This equation tells us that next year's class *i+1* is made up of those individuals from this year's class *i* who move up a class as well as those individuals from this year's class *i+1* who survive but do not advance in class. This equation holds for values of *i* from zero to *N-1* inclusive.

A careful look at this situation reveals that we have not yet specified the population of class zero in next year's population. Recall that members of class zero are either true newborns or previous members of class zero who do not advance for some reason.

The mathematical expression which gives the number of class zero residents in next year's population is given by:

*Next period's P(0)= (r(0)+f(0))*P(0) + f(1)*P(1) + f(2)*P(2) + ... + f(N)*P(N)*

Pay special attention to the first term of this equation. It does not follow the pattern of the other terms.

The Lefkovitch model has a structure similar to the Leslie model. In fact, if *P* is a column vector whose *N+1* entries are the populations of each class at some point in time, then there is an *(N+1)x(N+1)* matrix *A* so that the population next year can be determined by calculating the traditional matrix product A*P

Task I:

1. For a population with six classes, use your imagination to determine values for *s(i)*, *r(i)* and *f(i)* for each class. You can use the Leslie model example for inspiration.
2. Using the values found above, design and create in MATLAB the 6 × 6 matrix *A* which can be used to move from one time period to the next in the model.
3. Perform all the **Tasks** from the section on Leslie models using, instead, the Lefkovitch model.
4. Experiment with different values of the parameters to see if you can create Leslie and Lefkovitch models where the entire population grows over time or approaches some stable number.

Task J

Write up a report discussing and explaining what you did at each step and what you discovered as you experimented with these models.

Discuss how you might refine the Lefkovitch model to make it realistic. For extra credit, implement your refinements in MATLAB and discuss the results.

Acknowledgement: This project is a modification of a tutorial created by Steve McKelvey of Saint Olaf College.