VISUALIZING LINEAR ALGEBRA APPLICATIONS WITH MATLAB

David Szurley
Francis Marion University
Department of Mathematics
Florence, SC 29501

1 Introduction

There are many applications that could be covered during a first semester linear algebra course. Some examples are predator-prey models, designing an electrical circuit with certain properties, rotating computer graphics, and least-squares lines (see [2], [3]). Due to simplifying assumptions, these applications are not always realistic, but they serve two purposes: they provide an illustration to the student of the usefulness of linear algebra and a groundwork so that the student may explore more difficult models.

Many students have the requisite knowledge to be able to carry out the calculations necessary for these applications. For example, most students would be able to rotate the vector $\mathbf{x} = [1\ 0]^T$ counterclockwise about the origin through an angle of $\varphi = \frac{\pi}{2}$ and obtain the vector $[0\ 1]^T$. The calculations are generally straightforward. The question then becomes: how much do students actually learn about the application by simply multiplying matrices and looking at numbers? Using a scientific program to illustrate these applications would lead to better insight into an application, and would be more beneficial and interesting to the student. One such scientific program that could be used is Matlab, which is published by The MathWorks.

Matlab is a high-level language that provides an interactive graphical user interface. As such, it can be used by both advanced and intermediate students in two different manners. Advanced students may use Matlab as their primary programming language, while intermediate students may take advantage of the graphical user interface to explore concepts learned in class. Matlab is featured in hundreds of textbooks in engineering and the sciences, and is integrated in the curriculum at many universities [1]. Using Matlab to illustrate concepts in a linear algebra course will not only help the student with the concept, but also expose them to a scientific program that is widely used in science and engineering disciplines.

The concepts of population modeling and genetics will be considered. It will be shown how to take advantage of Matlab's graphing capabilities to enhance the students' experience with these applications.

2 Population Modeling

2.1 Specifics

Consider the Leslie model of a population with $n = 6$ classes. This discrete-time model describes the evolution of the population of the 6 classes taking into account mortality and reproduction. Evolution of the classes is governed by powers of the matrix

$$L = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ s_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_5 & 0 \end{bmatrix},$$

where the constant birthrate of people per member of class i is given by b_i, and the constant survival rate giving the fraction of class i to survive to the next time period is given by s_i. Let $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_6]^T$ represent the population of the classes at any time period. Notice that the population at the next time period is given by

$$L\mathbf{x} = \begin{bmatrix} b_1x_1 + b_2x_2 + \cdots + b_6x_6 \\ s_1x_1 \\ s_2x_2 \\ s_3x_3 \\ s_4x_4 \\ s_5x_5 \end{bmatrix}.$$

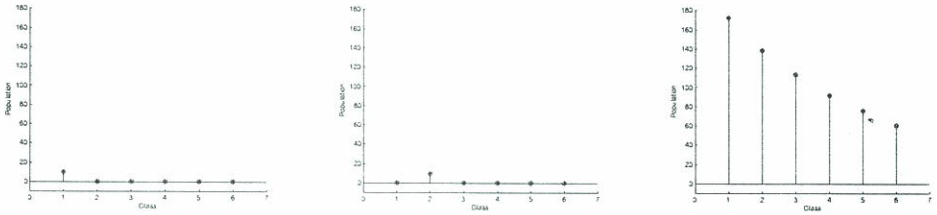
2.2 Example #1

The following set of commands in Figure 1 will compute, plot, and generate a movie for the evolution of the population of the six classes.

```
 9 -  L = [0 0.88 1.06 0.08 0 0 ; ...
10 -       0.98 0 0 0 0 0 ; ...
11 -       0 0.98 0 0 0 0 ; ...
12 -       0 0 0.98 0 0 0 ; ...
13 -       0 0 0 0.98 0 0 ; ...
14 -       0 0 0 0 0.96 0] ;
15
16 -  x = [10 ; 0 ; 0 ; 0 ; 0 ; 0] ;
17
18 -  FirstEx = avifile('Pop6.avi','fps',1) ;
19
20 -  for i = 0 : 6,
21 -      x = L^i * x ;
22 -      figure ;
23 -      hold on ;
24 -      xlabel('Class') ;
25 -      ylabel('Population') ;
26 -      plot([1 : 2 : 3 : 4 : 5 : 6], x, 'o', ...
27 -           [1 : 2 : 3 : 4 : 5 : 6], x, '-') ;
28 -      line([0 7], [0 0]) ;
29 -      for j = 1 : 6,
30 -          line([j j], [0 x(j)]) ;
31 -      end
32 -      axis([0 7 -10 180]) ;
33 -      frame = getframe(gcf) ;
34 -      for k = 1 : 6,
35 -          FirstEx = addframe(FirstEx, frame) ;
36 -      end
37 -  end
38
39 -  FirstEx = close(FirstEx) ;
```

Figure 1: Matlab evaluation, plotting, and movie generation commands.

Figure 2 contains three frames of the movie. The numbers used to plot these three frames (as well as the other frames) are easily found; for the ith time frame, we compute $L^i\mathbf{x}$. However, some characteristics of the situation may be overlooked when one only looks at the numbers. Notice that there are no members of the first class at the first time step since no-one is born to replace these members as they age into the next class. If students only compute certain powers of L, this fact might be lost. It is when these computations are plotted and then viewed side-by-side, or viewed in the context of a movie, that observations such as this are easier to discover.



(a) Initial Population (b) First time step (c) Sixth time step

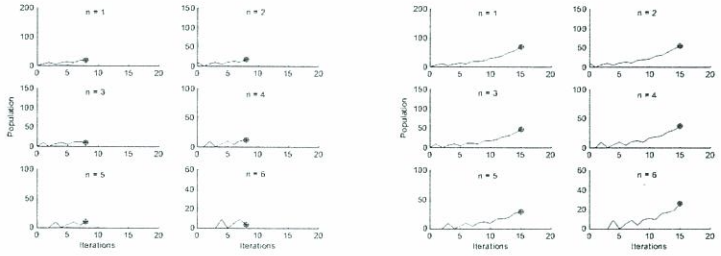
Figure 2: The population at three time periods.

2.3 Example #2

The previous example demonstrated how to make use of Matlab's graphing capability to generate movies. The students are then able to compare the population between classes and view the change. There are other ways to illustrate the same scenario. One drawback of the previous example is that in any given frame, the past evolution of the population is not represented. It is possible to plot the history of the population of each class. Figure 3 displays an abbreviated set of commands that will plot the evolution of the population of one class, and Figure 4 shows two time frames of this movie. In this method, the six classes are plotted separately, time itself is represented by the x-axis, and the population is represented by the y-axis.

```
37 -  Years = linspace(0, n, n + 1) ;
38
39 -  x = zeros(6, n) ;
40
41 -  Population = avifile('PopMovie2.avi') ;
42
43 -  for i = 1 : n,
44 -      x(1 : 6, i) = L^i * x_0 ;
45 -      figure ;
46 -      subplot(3, 2, 1) ;
47 -      hold on ;
48 -      xlabel('Generations') ;
49 -      ylabel('Population (Millions)') ;
50 -      plot(Years(1, 1 : i), x(1, 1 : i), 'r-') ;
51 -      plot(Years(1, i), x(1, i), 'ro', ...
52 -          Years(1, i), x(1, i), 'k*') ;
53 -      axis([0 n 0 200]) ;
54 -      hold off ;
55 -      F(i) = getframe(gcf) ;
56 -      for j = 1 : 6,
57 -          Population = addframe(Population, F(i)) ;
58 -      end
59 -  end
60
61 -  Population = close(Population) ;
```

Figure 3: Matlab commands to generate a movie of the evolution of a class.



(a) Ninth time step (b) Sixteenth time step

Figure 4: The evolution of each class population.

3 Genetics

A certain flower can have one of three colors. The color of the flower is determined genetically by the following key.

$$\begin{aligned} AA &\leftrightarrow \text{red}, \\ aa &\leftrightarrow \text{yellow}, \\ Aa &\leftrightarrow \text{green}. \end{aligned}$$

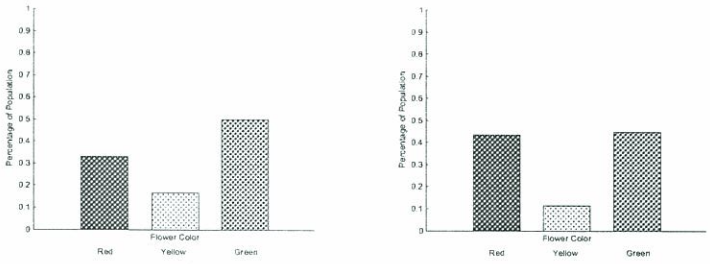
Here we let $\mathbf{x} = [r\ y\ g]^T$ represent the fraction of flowers that have colors of red (r), yellow (y), and green (g), respectively. A breeding program is introduced so that the color distribution at the next time step depends only upon the distribution of the current time step as such.

$$\begin{aligned} r^{(n+1)} &= \frac{2}{3}r^{(n)} + \frac{1}{3}g^{(n)}, \\ y^{(n+1)} &= \frac{1}{3}y^{(n)} + \frac{1}{6}g^{(n)}, \\ g^{(n+1)} &= \frac{1}{3}r^{(n)} + \frac{2}{3}y^{(n)} + \frac{1}{2}g^{(n)}. \end{aligned}$$

There are many different ways to illustrate the evolution of flower color. One method that could attract attention is to plot the fraction of each color with colored rectangles, the height of which represents the fraction of the population. Many of the same commands hold, in particular the commands dealing with the movie generation. Figure 5 shows the portion of the code that is used to plot a rectangle representing the fraction of one color, while Figure 6 gives two time steps of the distrbution of flower color.

```
58 -        X = A^(i-1) * X_0 ;
59
60          %Plot.
61 -        figure1 = figure ;
62 -        axes('Parent', figure1, 'YTickLabel', {}, 'XTick', zeros(1,0)) ;
63 -        hold on ;
64 -        xlabel('Flower Color') ;
65 -        ylabel('Percentage of Population') ;
66 -        plot(linspace(2/3, 4/3, 120), X(1), '-') ;
67 -        fill([2/3 ; 2/3 ; 4/3 ; 4/3], [0 ; X(1) ; X(1) ; 0], 'r') ;
68 -        text(1-0.1, -0.1, 'Red') ;
```

Figure 5: Matlab commands to plot a rectangle for one flower color.



(a) First time step (b) Fifth time step

Figure 6: The evolution of flower color distribution.

4 Conclusions

The movies and code used in the preparation of this paper can be found at `http://acsweb.fmarion.edu/math/szurley/hp_szurley.htm`. There are many applications within a linear algebra course where the students are asked to evaluate matrix multiplications. Population modeling and genetic evolution are two examples of this. The graphing capabilites of Matlab may be used to illustrate this material. In presenting the applications visually, the student may gain insight into their intricacies.

References

[1] Business/Technology Editors. (2001). *The MathWorks Announces Worldwise Availability of Matlab Students Release 12.* Retrieved February 7, 2007, from BusinessWire. Web site: http://www.businesswire.com/webbox/bw.012401/210242193.htm.

[2] Lay, D. (2003). *Linear Algebra and Its Applications.* New York: Addison Wesley.

[3] Wilde, C. (1988). *Linear Algebra,* New York: Addison Wesley.