Lab 1 Worksheet

Getting to Know MATLAB

MATLAB is a scientific programming language with an easy-to-use interface. The name stands for “matrix laboratory,” because the program is designed to handle matrices quickly and efficiently with many built-in functions for performing common linear-algebra operations. MATLAB is frequently used as a tool in many areas of biological research, and it is a great starter language if you are new to programming.

When you open MATLAB for the first time, you are presented with the MATLAB environment, which has three re-sizable windows, the most important of which is the Command Window. This is where you define variables, call functions, or anything else you would like to do. Variables in a programming language are just like mathematical variables—in MATLAB the variables are, conveniently, vectors or matrices. Let’s define a variable by entering the following command in the Command Window:

\[
>> a = 4
\]

You might notice that variable \( a \) now shows up in the Workspace window. Let’s also define a row vector:

\[
>> v = [1 5 3]
\]
(Hint: If we add a semicolon to the end of the command, MATLAB won’t print the variable.)
This is three-element vector, which we can check that with the command

```
>> length(v)
```

A deceptively similar-sounding command is `size(v)`. However, `size()` tells us something different—it gives us information about what kind of data type `v` is. If you try it, you will find that `v` is a matrix with one row and three columns. In fact, all MATLAB variables, even `a`, which you defined before, are matrices!

Another way of checking this, which provides more detail, is to use the commands `whos` and `whos('a')`. Try both. In simple terms, what’s the difference between the outputs of these two commands?

These commands give us information which includes the data-structure size ("size"), the amount of memory it occupies ("bytes"), and the type of data contained in the variable ("class") for all or one of your variables. A “double” is a double-precision floating-point number, which is a way of storing a noninteger number that can be very large or have many decimal places. This is the default, although there are several other kinds, which include types of integers.

Now, let’s define two new variables via some arithmetic on our existing variables:

```
>> b = sqrt(a)
```

```
>> c = a / v(2)
```

What do you expect `c` to be?

When we identify the second element in vector `v`, we are indexing. (If you are familiar with other programming languages, keep in mind that MATLAB is 1-indexed, meaning that `v(1)` calls the first element.) We can also scale `c`, i.e. multiply each element by the same scalar:

```
>> c = a * v
```

Before moving on to Problem 1, use `clear all` to delete all of your variable assignments.
Problem 1: Exploring Geometry with MATLAB

We’re going to explore what the concept of a vector’s length means using MATLAB. Let’s start by defining a 2-D vector (here a $1 \times 2$ matrix):

\[
\begin{bmatrix}
7.5 \\
3
\end{bmatrix}
\]

The first time that we learn about vectors, we are usually told that they are objects with magnitude and direction. In this case, our vector points to a location in the $xy$-plane, which we can see by plotting the vector using the following commands:

\[
\text{quiver}(0, 0, w(1), w(2), 0)
\]

The first two arguments specify the $x$- and $y$-coordinates of the base of the vector, in this case the origin. Next come the components of the vector, and the final zero prevents the plot from scaling improperly.

Now, we are going to visualize vector addition. Redefine

\[
\begin{bmatrix}
2 \\
3.5
\end{bmatrix}
\]

and let

\[
\text{v} = \text{w} + \text{v}
\]

Using \text{quiver()} plot \text{w} and \text{v} as a tip-to-tail sum (i.e. first plot \text{w} at the origin, then plot \text{v} starting at the tip of \text{w}). Finally, plot \text{s} on the same figure starting at the origin, and show that its tip coincides with the sum \text{w} + \text{v}. You will need to call \text{quiver()} once for each vector (three times in total), and, in order to show all three on the same plot, you should enter

\[
\text{hold on}
\]

in between each \text{quiver()} command. \text{quiver()} is also useful for visualizing vector fields, which are functions $F(x)$ for which every point has a corresponding vector assigned to it. A vector field is one technique for representing the 3-D structure of chromatin in the nucleus. Chromatin can be thought of as a string, where at each point we place a unit vector that specifies the direction of the tangent to the curve.

In class you learned that the length of a vector $x$ is given by

\[
|x| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} = \sqrt{x \cdot x},
\]

where $x_i$ is the $i$-th component of the vector. Use this formula (with \text{dot()} to plot (in a new figure) a new vector, $u$, which has the same length as $w$ and is along the $x$-axis. If
you would like to save the graph, use the drop-down menu from “File” in the graph’s window.

In class we also discussed the multiplication of a matrix and a vector—the result, \(\mathbf{a} = M\mathbf{b}\), is another vector. A \(2 \times 2\) matrix is defined in MATLAB via

\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}
\]

Call \texttt{size()} to confirm this. Which two elements are in this matrix’s first row?

One special kind of matrix is the rotation matrix, which takes some vector and rotates it around the origin by an angle \(\theta\). For a \(2\)-element vector, we define

\[
R_\theta = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}.
\]

Let’s start by rotating \(\mathbf{u}\) by 90 degrees (you will need to make \(\mathbf{u}\) a column vector, by taking its transpose):

\[
\begin{bmatrix}
\text{y} = R \cdot \text{transpose}(\mathbf{u})
\end{bmatrix}
\]

(Note: You need to convert to radians in MATLAB.) When you plot \(\mathbf{y}\), you will notice that it is perpendicular to \(\mathbf{u}\), which is parallel to the \(x\)-axis. Did the length of the vector change when you rotated it?

Now, produce a plot in which \(\mathbf{w}\) is rotated so that it is along the \(x\)-axis; in other words, the rotated vector, \(\mathbf{w}'\), should be on top of \(\mathbf{u}\). You will need to figure out what \(\theta\) is. Use the dot product relation

\[
\frac{\mathbf{w} \cdot \mathbf{b}}{|\mathbf{w}||\mathbf{b}|} = \cos \theta,
\]

where \(\theta\) is the angle between vectors \(\mathbf{w}\) and \(\mathbf{b}\) (“\texttt{acos()}” is the inverse cosine function in MATLAB). What should \(\mathbf{b}\) be?

MATLAB also includes a lot of options for a nice-looking and professional plot. You’re going to finish this problem by exploring a few of them. Reproduce the plot that you just made so that \(\mathbf{w}'\) is almost, but not quite, on top of \(\mathbf{u}\) (so you can see both of them). Add a title to the plot, label the axes, and add a legend. All three of these things can be done easily from the dropdown menu “Insert” within the figure. Also, make the window size different from its automatic size (zoom in or out). Do this—again, in the figure—via “Edit” \(\rightarrow\) “Axis Properties...” then change the “X limits,” for instance.
Further Problems

Problem 2: Colorblindness

In this problem you are going to explore a nice physical example of linear superposition (or linear combination) and relate it to colorblindness. A common test of certain types of colorblindness is an image like the one you see below\(^1\)—can you distinguish a number inside the circle?

You are likely familiar—perhaps from a grade school art class—with the red, green, and blue (rgb) color palette. The idea is that with paints of these three colors you can make any possible color. There is a direct connection between this observation made by artists and the way in which our eyes perceive color! And this connection can be explained mathematically.

Recall that a vector can be expressed as a linear combination of other vectors. For example,

\[
\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \end{bmatrix}.
\]

Similarly,

\[
\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.
\]

Now, let’s decide that the first vector on the left-hand side stands for the color red and the second for green. What color does the third vector thus represent?

We can test this in MATLAB:

\[
\begin{align*}
\texttt{>> x} & \texttt{ = 0:10} \\
\texttt{>> plot(x, x, 'o', 'MarkerEdgeColor', [1 1 0])}
\end{align*}
\]

As expected, the plot you produce is yellow (naturally, \([0 0 1]\) produces blue). Although we could theoretically decide to have \([0 0 1]\) represent red, MATLAB uses the standard color-assigning convention. What you just graphed was the function \(y = x\). What is the role of the first two arguments in the \texttt{plot()} function (i.e. \(x, x\)), and why is the range of the graph what it is?

Check what “\texttt{x = 0:0.5:10}” does and compare it to the command above.

Note: The third argument of \texttt{plot()} specifies the type of marker MATLAB uses for each point, and \texttt{MarkerEdgeColor} tells MATLAB to only give color to the edge of each marker, as opposed to the inside.

Now, make two new plots. In the first, show \(\sin x\) in gold, and, in the second, show \(e^x\) in magenta. In both cases, you can look up the rgb-color code online and you should plot the functions over an appropriate interval. In order to display both plots at once (but in different windows), keep the first plot open, enter the command \texttt{figure}, and then execute the second plotting command. Once you have finished trying out a few different colors, figure out how to make the line white.

If you are familiar with the basics of vision, then you know that the first step in color perception involves the retina, which is the lining at the back of your eye. The retina is comprised of millions of cone cells, which fall into three categories: long-wavelength (reddish), medium-wavelength (greenish), and short-wavelength (bluish). Each of these cells has a high sensitivity to light of its “favorite” wavelength, and, when stimulated appropriately, it sends a signal to the cognitive machinery of the brain, which (through a more complicated process) produces our impression of color.

Colorblindness, in one of its many forms, affects 1 in 12 American men and 1 in 200 women. Two common forms are \textit{anomalous trichromacy}, wherein all three cones are functioning, but not properly, and \textit{dichromacy}, wherein only two of the cones are functioning. As a UC Davis student, you would have particularly bad luck if you had \textit{tritanomaly}, referred to as “blue/yellow” colorblindness.

A nice website for exploring colorblindness is www.vischeck.com. Go to this page, and use the tool for simulating colorblindness. Find a blue and gold UCD logo online, and start
by selecting tritanopic colorblindness. You’ll notice that the name is a bit of a misnomer—someone with this condition can distinguish the two colors from one another, but perceives them differently. Produce images with the two other simulators, as well.

When you plotted the functions earlier in this problem in gold and magenta, you were using linear superposition to produce these nonstandard colors. This website does the opposite: it decomposes the images into their rgb-form and then demonstrates what the images would look like if you had difficulty seeing one of those colors.

Problem 3: Stoichiometry of a Biochemical Reaction

One of the most important biochemical metabolic pathways is glycolysis, in which the human body accesses energy stored in glucose. The first step of this process is given by the following reaction equation:

\[ \text{glucose (GLC)} + \text{ATP} \Rightarrow \text{glucose 6-phosphate (G 6-P)} + \text{ADP} + \text{H}^+. \]

We can write this out in all its chemical detail as follows:

\[ \text{C}_6\text{H}_{12}\text{O}_6 + \text{C}_{10}\text{H}_{16}\text{N}_5\text{O}_{13}\text{P}_3 \Rightarrow \text{C}_6\text{H}_{13}\text{O}_9\text{P} + \text{C}_{10}\text{H}_{15}\text{N}_5\text{O}_{10}\text{P}_2 + \text{H}^+ \]

In chemistry a common problem is to determine the correct stoichiometry of a reaction. Stoichiometry means the relative number of each element (e.g. C, H) involved in the reaction, and it expresses conservation of mass. This is also called “elemental balancing.”

For a quick example, consider the following equation for the reaction by which water is made:

\[ \text{H} + \text{O}_2 \Rightarrow \text{H}_2\text{O} \]

Add integer coefficients to each term, as needed, to balance this equation.

One way to check if your equation is balanced is to set up an “elemental matrix,” \( D \), in which the the rows are the chemical elements and the columns are the reaction compounds. The other thing you’ll need is the stoichiometric column matrix, \( S \) (a column matrix is the same thing as a column vector—after all, why have one name for something when you can have two?). If a reactant is consumed (left side of the equation), a value of \(-1\) is listed; if a compound is produced as a product (right side of the equation), the value is \(+1\); and if something does not participate at all, then a 0 is given.

---

For this reaction:

\[
D = \begin{bmatrix}
6 & 10 & 6 & 10 \\
12 & 13 & 11 & 13 \\
6 & 13 & 9 & 10 \\
0 & 3 & 1 & 2 \\
0 & 5 & 0 & 5
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
-1 \\
-1 \\
1 \\
1
\end{bmatrix}
\]

What does all of this mean? Take a look at the first column. Glucose is a molecule that has 6 carbon atoms, 12 hydrogens, and 6 oxygens (as we saw earlier in the detailed chemical equation). Therefore, the elemental matrix \( D \) gives us the relevant chemical information needed to make each of the reaction compounds. How many nitrogen atoms are in ADP?

If the elemental balance expressed in these two matrices is correct, then each of the row vectors of \( D \) must be orthogonal to the column vector \( S \). Two vectors are orthogonal to one another if their dot product is equal to zero:

\[
v \cdot u = v_x u_x + v_y u_y + v_z u_z = 0
\]

Check if this is the case by hand.

Now, let’s confirm this result via MATLAB. Recall that the dot product of two vectors \( v \) and \( u \) is related to the angle between them (i.e. \( v \cdot u = \cos 90 = 0 \)). As you may recall, the first rotation that you performed in Problem 1 was by 90 degrees, and the result was a vector that was perpendicular to the original vector. Use MATLAB and the function `dot()` to check again that \( S \) is orthogonal to each row of \( D \). (Note: `dot()` does not care whether the vectors are rows or columns.)

You will notice that \( S \) is not orthogonal to all of the rows of \( D \)—which row(s) is not orthogonal to \( S \)?

This means that our equation is not properly balanced. Use MATLAB to determine the
product $DS$. In descending order, the rows of this vector correspond to C, H, O, P, and N. What conclusion can you draw about the stoichiometry error?

Problem 4: Age-Structured Populations

One way of looking at the dynamics of a population in ecology is to study how its age breakdown changes over time. This population could be the deer living in a given area, or it could be the American citizenry. It the latter case, we might be interested in knowing what fraction of the population is of working age (as opposed to say, retirement age, 65+). We call this approach age-structured populations, and it is handled mathematically using Leslie matrices. This is an example of what we call a “discrete-time dynamics,” and it foreshadows some of the things you will discuss in Math 27B. In other words, the Leslie matrix is responsible for taking our population vector at some time $i$ and advancing it one time step, to $t = i + 1$.

$$n_{i+1} = L n_i$$

Here, the Leslie matrix, $L$, is a square matrix of the same dimension as the population vectors. For example, our population could be partitioned as follows:

$$n_i = \begin{bmatrix} 0-1 \text{ yrs} \\ 1-2 \text{ yrs} \\ 2-3 \text{ yrs} \end{bmatrix}.$$ 

Age-structured populations helps us to study organisms whose life cycles are divided into distinct stages, each with particular characteristics. Bison are a good example, because they have three such stages: calf, yearling, and adult.\(^3\)

Mathematical models rely heavily on data from biological experiments and observations. In this example, we start with the data that each adult female has a 95% chance of surviving the year and gives birth with a probability of 0.42; 60% of the calves survive their first year; and 70% of the yearlings reach adulthood. Since we have not discussed matrix multiplication yet, we will simply say that this information is encoded in matrix form as

$$L = \begin{bmatrix} 0 & 0 & 0.42 \\ 0.6 & 0 & 0 \\ 0 & 0.7 & 0.95 \end{bmatrix}.$$ 

For starters, let’s suppose that we introduce a population of 50 adult female bison into a new habitat (we will just consider the female population for ease). Start by writing down $x_0$, a three-dimensional population vector going from first stage to last. Using MATLAB to carry out the multiplication, determine what the population is at $t = 5$ years.

---

\(^3\)Finotti H; “Leslie Matrix Models;” (Presentation) Dept. of Mathematics, University of Tennessee; October 29, 2013.
You have already been introduced—briefly—to eigenvalues and eigenvectors, which capture key information about a matrix. The simplest explanation of these two concepts is that a matrix, $M$, multiplied by one of its eigenvectors, $\lambda$, gives back the same vector multiplied by a number that is called its eigenvalue:

$$Mv = \lambda v.$$

Eigenvalues and eigenvectors also contain subtle information relevant to the physical system being studied. The eigenvectors of a matrix are its “preferred directions.” In this example, you will see that one eigenvector, corresponding to the largest eigenvalue, becomes the longterm state of the population—the $t_\infty$-limit, as we say. This means that the largest eigenvalue (absolute value) becomes the $t_\infty$-growth rate. You’ll come to understand these terms better in a little while.

MATLAB makes it easy to determine the eigenvalues and eigenvectors of a matrix:

```matlab
g >> [V, D] = eig(L)
```

Then, you can call either $V$ or $D$, which, respectively, are the eigenvector matrix and the eigenvalue matrix (these are the diagonal entries). Verify that each of the three columns of $V$ is indeed an eigenvector of $L$ with the correct eigenvalue. Write out which eigenvalue corresponds to each column of the eigenvector matrix.

A discrete-time dynamics is an example of an iterative process—one in which you proceed one-step-at-a-time. When you have any iterative process, which you want to evaluate computationally, you can use a loop structure. The most common such structure is the “for” loop.

Your task is to determine the age structure of the bison population ten years after they are introduced into the new habitat. Your code will include the following:

```matlab
g >> for i = 1:10,...
x_new = L * x_0,...
x_o = x_new,...
end
```

You will notice that the “...” allows you to keep entering code on the next line without executing it. This is useful when you have a simple routine that you want to create quickly.
Let’s finish with a simpler Leslie matrix, which describes an organism with two life stages, child and adult. Given the following matrix and initial condition,

\[
L = \begin{bmatrix}
1 & 4 \\
0.5 & 0 \\
\end{bmatrix} \quad x_0 = \begin{bmatrix}
1 \\
2 \\
\end{bmatrix}
\]

do the following:

1. Determine the eigenvalues and eigenvectors of the matrix, identifying which pair corresponds to the largest eigenvalue.

2. Use quiver() to plot the preferred eigenvector and \(x_0\).

3. Using a “for” loop (you’ll have to figure out how many iterations to go through), determine at what time step the population reaches its \(t_\infty\)-growth rate. Plot the population vectors for each of these intermediate times, along with \(x_0\) and the preferred eigenvector (e.g. your plot might include five vectors in total). You will know that the population has reached its limiting growth rate when \(x_t\) is parallel to the eigenvector. Identify each vector with a different color and choose a window size in which all vectors are visible.