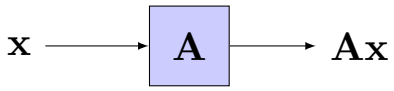
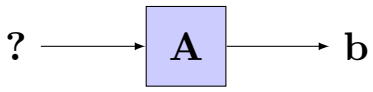
Math 551 Fall 2018
Lab 8

Goal: In this assignment, you will use Matlab to visualize the action of a 2×2 matrix $\mathbf{A}$. You will practice thinking of $\mathbf{A}$ as a type of abstract system that takes as input a 2-vector $\mathbf{x}$ and produces as output another 2-vector $\mathbf{Ax}$ (the matrix-vector product). Using a common engineering style for representing abstract systems, $\mathbf{A}$ can be represented as follows.



With this way of thinking about $\mathbf{A}$, the task of solving a linear system $\mathbf{Ax} = \mathbf{b}$ can be rephrased as a design problem: Find the input $\mathbf{x}$ to give to $\mathbf{A}$ in order to achieve the desired output $\mathbf{b}$.



Learning this way of thinking of matrices can be amazingly valuable in the remainder of Math 551, as well as in a large number of other courses in math, engineering and the sciences.

Getting started: You will need to download three files and place them into your working directory:

- `m551lab08.m`: the M-file you will edit during the lab
- `draw_coords.m`: a function for drawing coordinate systems
- `draw_vec.m`: a function for drawing vectors

What you have to submit: The file `m551lab08.m`, which you will modify during the lab session.

Reminders About the Grading Software

Remember, the labs are graded with the help of software tools. If you do not follow the instructions, you will lose points. If the instructions ask you to assign a value to a variable, you must **use the variable name given in the instructions**. If the instructions ask you to make a variable named `x1` and instead you make a variable named `x` or `X` or `X1` or any name other than `x1`, the grading software will not find your variable and you *will* lose points. Required variables are listed in the lab instructions in a gray box like this:

Variables: `x1`, `A`, `q`

At times you will also be asked to answer questions in a comment in your M-file. You must **format your text answers correctly**. Questions that you must answer are shown in gray boxes in the lab. For example, if you see the instruction

Q7: What does the Matlab command `lu` do?

your file must contain a line similar to the following somewhere

```
% Q7: It computes the LU decomposition of a matrix.
```

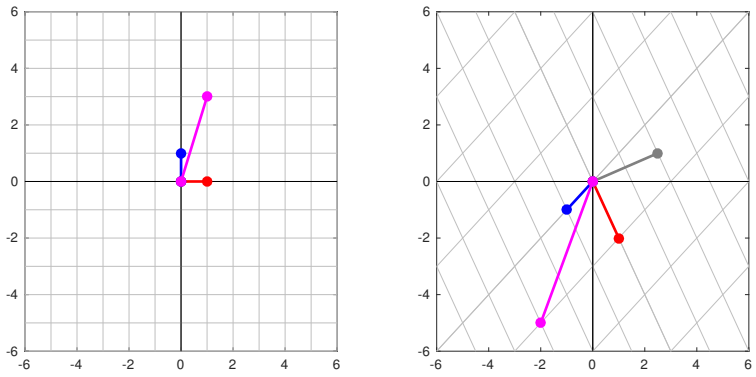
The important points about the formatting are

- The answer is on a single line.
- The first characters on the line is the comment character `%`.
- The answer is labeled in the form `Q`n`:`, where n stands for the question number from the gray box.

If you do not format your answers correctly, the grading software will not find them and you *will* lose points.

Tasks

1. Open the file `m551lab08.m` in the Matlab editor. Move to the cell titled "Part 1" and run it. Matlab should open a figure window that looks like the following.



Spend some time getting to know this figure. On the left, you see a representation of some vectors in $\mathbb{R}^2$. The red and blue vectors are the vectors

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \text{and} \qquad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

respectively. These two vectors define a very natural grid in the plane made up of two types of lines (shown in gray). First, there are lines that run parallel to the blue vector and are spaced apart by a length determined by the red vector. Second, there are lines that run parallel to the red vector and are spaced apart by a length determined by the blue vector. These lines intersect at points of the form $k\mathbf{e}_1 + \ell\mathbf{e}_2$ where k and ℓ are both integers. By looking at the grid lines, you can see that the magenta vector is $\mathbf{x} = \mathbf{e}_1 + 3\mathbf{e}_2$.

Now, look at the plot on the right. Understanding this plot will help you visualize the behavior of the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ -2 & -1 \end{bmatrix}.$$

First, spend some time convincing yourself that the red and blue vectors on the right are exactly $\mathbf{A}\mathbf{e}_1$ and $\mathbf{A}\mathbf{e}_2$. Next, take a look at the thin gray lines. Although they are no longer horizontal and vertical, these lines still have the interpretation from before. There are lines parallel to red spaced by blue, and lines parallel to blue spaced by red. Think of the lines as city streets. In the left plot, the endpoint of the magenta vector $\mathbf{x}$ can be located by following the directions "Go one block in the red direction and three blocks in the blue direction." And in the right plot, the magenta vector $\mathbf{A}\mathbf{x}$ can be described in exactly the same way! This is a geometric way of visualizing the fact that

$$\mathbf{A}\mathbf{x} = \mathbf{A}(1\mathbf{e}_1 + 3\mathbf{e}_2) = 1(\mathbf{A}\mathbf{e}_1) + 3(\mathbf{A}\mathbf{e}_2).$$

2. Now, you'll use these ideas to solve a linear system. Notice that there is a gray vector drawn in the right plot of the figure above. This is your target vector $\mathbf{b}$ defined in the M-file as

```
% define the target vector
b = [ 2.5; 1 ];
```

Your task is to change the definition of `x_part1` at the top of the cell until the magenta vector is identical to the gray vector. You can do this by trial and error, but you'll be much more efficient if you spend a little time counting the "city blocks". As a hint, notice that you need to go in the opposite direction of the blue vector, but in the same direction as the red. So your blue component should be negative and your red component positive.

When you run the cell, notice that the script produces some output in the command window. With the original value of `x_part1`, the output is

```
Part 1: error in Ax-b = 7.500000e+00
```

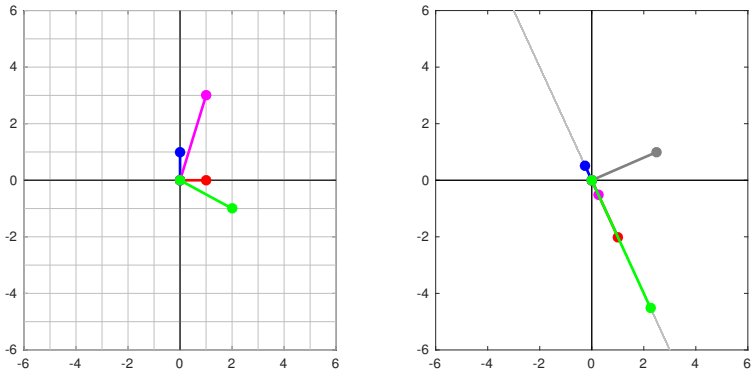
You'll know that you're finished when you get the error to 0. (Yes, in this case, you can actually get 0 error.)

Variables: `x_part1`

3. Now move to the cell labeled "Part 2" and run it. This cell behaves very much like the previous, except for two changes. First, I have changed the definition of $\mathbf{A}$ to

$$\mathbf{A} = \begin{bmatrix} 1 & -0.25 \\ -2 & 0.5 \end{bmatrix}.$$

Second, I have added another vector (green) to the plots. Now, Matlab produces a figure that looks like the following.



To convince yourself that the the plot makes sense, you should verify that the vectors $\mathbf{A}\mathbf{e}_1$ and $\mathbf{A}\mathbf{e}_2$ both lie on the same line through the origin. That means that the "streets" in the blue direction are exactly the same as those in the red direction. In fact, there's only one street in the transformed city.

Note that, again, the target vector $\mathbf{b}$ is drawn in gray. Locate the appropriate place at the very top of the M-file cell and answer the following question.

Q1: Why can't there exist a solution to the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ in this case?

4. Now, let's figure out a system we *can* solve. Near the top of the cell, locate the definition of the variable `b_part2`. Change this definition so that

   (a) a solution does exist, and

   (b) the first component of $\mathbf{b}$ is 2.

   Again, you can probably find the right vector by trial and error, but it might pay to think for a bit about what line $\mathbf{b}$ lies on.

   Once you have found the right-hand side $\mathbf{b}$, move a few lines up in the M-file to where `x1_part2` and `x2_part2` are defined. These lines define vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ (magenta and green, respectively). Your next task is to modify their definitions so that

(a) both are solutions to the linear system with your new $\mathbf{b}$,

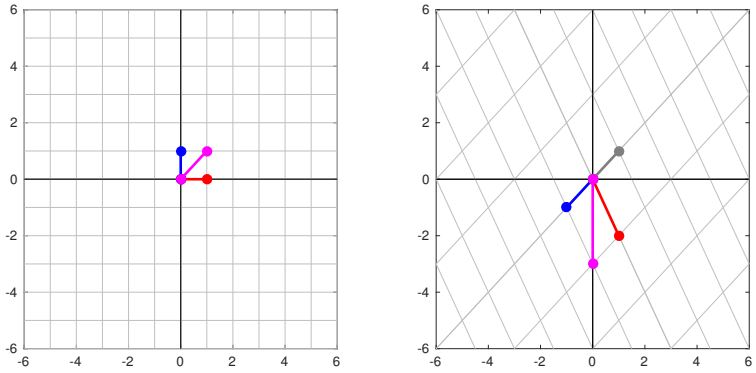
(b) the second component of $\mathbf{x}_1$ is 0, and

(c) the first component of $\mathbf{x}_2$ is 0.

The code in the cell will provide feedback on your progress. You'll know you're finished when no warnings are reported and when both errors are exactly 0.

Variables: `b_part2`, `x1_part2`, `x2_part2`

5. In the final part of this lab, let's return to the original matrix $\mathbf{A}$ and ask a different question. Move to "Part 3" in the M-file and run the cell. Matlab should produce the following figure.



We're basically back to where we started in Part 1, but now, instead of a fixed $\mathbf{b}$, we're going to have a moving target. More specifically, the magenta vector in the left plot is a vector $\mathbf{x}$ defined in the variable `x_part3`, and the magenta vector on the right is $\mathbf{Ax}$. However, this time the gray vector on the right is simply $\mathbf{x}$ again. Your task is to find a vector $\mathbf{x}$ so that, on the right, the gray vector ($\mathbf{x}$) and the magenta vector ($\mathbf{Ax}$) lie on the same line. (Such a vector is called an *eigenvector* of $\mathbf{A}$.)

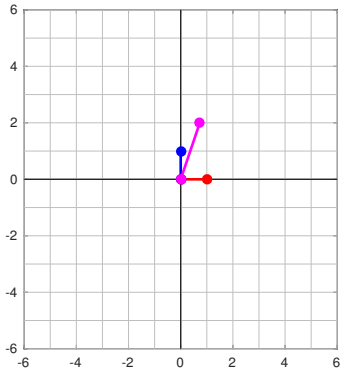
To see that such a thing is possible, find the following code in this cell.

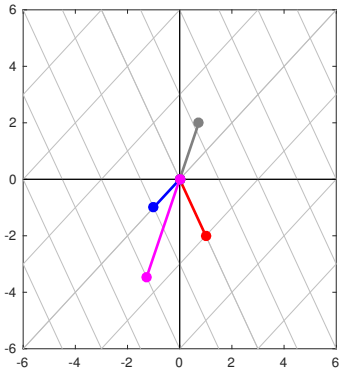
```
% define the vector x
x_part3 = [ 1; 1 ];  % first figure
%x_part3 = [ 0.732050807568877; 2.0 ]; % second figure
```

Switch the comment character (%) from the last line to the one above, so that the code looks like this.

```
% define the vector x
%x_part3 = [ 1; 1 ];  % first figure
x_part3 = [ 0.732050807568877; 2.0 ]; % second figure
```

Run the cell again and Matlab should produce the following picture.





Although it might not be easy to tell for sure just by looking at the plot, the gray and magenta vectors do lie on the same line. In fact, $\mathbf{Ax} = -\sqrt{3}\mathbf{x}$. To check this, enter the following into Matlab's command window.

```
>> A*x_part3 + sqrt(3)*x_part3

ans =

   1.0e-15 *

   -0.8882
    0.4441
```

This shows that the difference between $\mathbf{Ax}$ and $-\sqrt{3}\mathbf{x}$ is $\mathbf{0}$ plus some roundoff errors.

Your final task in this lab is to find another eigenvector of $\mathbf{A}$, this time with $\mathbf{Ax} = +\sqrt{3}\mathbf{x}$. The particular $\mathbf{x}$ you are seeking has 2 as its second component and points into the second quadrant (the first component is negative). If you've learned about eigenvalues already, you may know how to find this vector using Matlab functions. If not, or if you're interested in developing your geometric intuition, use trial and error. Since you know that the second component should be 2 and the first component should be negative, it won't take too long. You'll know you're finished when the code produces no warnings and when your error is less than 10^{-2}. Using only 3 digits for $\mathbf{x}$:

```
x_part3 = [ -?.??; 2 ];
```

I was able to get the output to look like this:

```
************************************************************
* Part 3
************************************************************

  error in current guess = 4.367738e-03


************************************************************
```

(If you want more of a challenge, try to get the error below 10^{-3}. I got an error just below that using 5 digits.)

Variables: x_part3