1. **Plotting in matlab.** Create a matlab figure window using `figure`. Plot vertical and horizontal axis lines, extending from $-2$ to $+2$. Now generate a random 2-dimensional vector $\vec{v}$ using `randn`, and plot it as a red line emanating from the origin. Execute `axis equal` to tell matlab to make the units of the two axes the same, and then `hold off` to tell matlab you’re done adding things to this figure. Make sure the vector looks like what you expect! Re-execute your code several times (generating a new random vector each time), and verify that it’s working.

Create a vector containing 31 equi-spaced angles from 0 to $2\pi$. Now use this to plot a unit circle (technically, it’s not a circle, but a 30-sided polygon!) in green, on top of vertical and horizontal axes. What happens if you forget to execute `axis equal`? Plot a random 2-vector on top of this (as above), and verify visually and numerically whether its length is greater or less than 1. Do this several times!

2. **Inner product with a unit vector.** Given unit vector $\hat{u}$, and an arbitrary vector $\vec{x}$, write (MATLAB) expressions for computing: (a) the distance from $\vec{x}$ to the N-1 dimensional subspace (a “hyperplane”) perpendicular to $\hat{u}$, (b) the component of $\vec{x}$ lying along the direction $\hat{u}$, (c) the component of $\vec{x}$ that lies in the hyperplane perpendicular to $\hat{u}$.

Convince yourself your code is working by testing it on random vectors $\hat{u}$ and $\vec{x}$ (hint: generate these using `randn`, and don’t forget to re-scale $\hat{u}$ so that it has unit length). First, do this visually with 2-dimensional vectors, by plotting $\hat{u}$, $\vec{x}$, and the two components described in (b) and (c). Then test it numerically in higher dimensions (e.g., 4) by writing (and running) expressions to verify each of the following:

- the vector in (b) is in the same direction as $\hat{u}$
- the length of the vector in (b) is equal to the value computed in (a)
- the vector in (b) is perpendicular to the vector in (c)
- the sum of the vectors in (b) and (c) is equal to $\vec{x}$.

3. **A simple linear system.** Suppose you have a retinal neuron whose response is a weighted sum of the intensities of light that land on 6 photoreceptors (note that these intensities are positive values). The weight vector is $[1, 3, 5, 4, 2, 1]$. (a) What unit-length stimulus vector elicits the largest response in the neuron? Explain how you arrived at your answer. (b) Now generate a unit-length stimulus vector that elicits a zero response in the neuron (and verify that this is true). Is this a physically realizable stimulus? Is there any realizable stimulus (not necessarily unit length) that would elicit a zero response in the neuron? If so, give an example.
4. **Testing for (non)linearity.** Suppose, for each of the systems below, you observe the indicated input/output pairs of vectors (or scalars). Determine whether each system could possibly be a linear system. If so, provide an example of a matrix that is consistent with the observed input/output pairs. If not, explain why.

System 1: \[ 1 \rightarrow [4, 6] \]
\[ 2.5 \rightarrow [10, 14] \]

System 2: \[ [6, 3] \rightarrow [12, 12] \]
\[ [-2, -1] \rightarrow [-6, -6] \]

System 3: \[ [1, 2] \rightarrow [5, -1] \]
\[ [1, -1] \rightarrow [1, 4] \]
\[ [3, 0] \rightarrow [7, 8] \]

System 4: \[ [2, 4] \rightarrow 0 \]
\[ [-2, 1] \rightarrow 3 \]

System 5: \[ 0 \rightarrow [1, 2] \]

5. **Geometry of linear transformations**

   (a) Write a function `plotVec2` that takes as an argument a matrix of height 2, and plots each column vector from this matrix on 2-dimensional axes. It should check that the matrix argument has height two, signaling an error if not. Vectors should be plotted as a line from the origin to the vector position, using circle or other symbol to denote the “head” (see help for function `plot`). It should also draw the x and y axes, extending from -1 to 1. The two axes should be equal size, so that horizontal units are equal to vertical units (read the help for the function `axis`).

   (b) Write a second function `vecLenAngle` that takes two vectors as arguments and returns the length of each vector, as well as the angle between them.

   (c) Generate a random 2x2 matrix, and decompose it using the SVD. Now examine the action of this sequence of transformations \((USVT)\) on the two “standard basis” vectors, \(\{\hat{e}_1, \hat{e}_2\}\). Specifically, use `vecLenAngle` to examine the lengths and angle between two basis vectors \(\hat{e}_n\), the two vectors \(V^T\hat{e}_n\), the two vectors \(SV^T\hat{e}_n\), and the two vectors \(USV^T\hat{e}_n\). Do these values change, and if so, after which transformation? Verify this is consistent with their visual appearance by plotting each pair using `plotVec2`.

   (d) Generate a matrix \(P\) with 65 columns containing 2-dimensional unit-vectors \(\hat{u}_n = [\cos(\theta_n); \sin(\theta_n)]\), and \(\theta_n = 2\pi n/64, n = 0, 1, \ldots, 64\). [Note: Don’t use a for loop! Create a vector containing the values of \(\theta_n\).] Plot a single blue curve through these points, and a red star (asterisk) at the location of the first point. As in the previous problem, apply the SVD transformations one at a time to full set of points (again, don’t use a for loop!), plot them, and describe what geometric changes you see (and why).

6. Load the file `mtxExamples.mat` into your MATLAB world. You’ll find a set of matrices named `mtxN`, with \(N = 1, 2, \ldots\). For each matrix, use the SVD to (a) determine if there are non-trivial (i.e., not the zero vector!) vectors in the input space that the matrix maps to zero
(i.e., determine if there’s a nullspace), and if so, write a MATLAB expression that generates
a random example of such a vector, and verify that the matrix maps it to the zero vector, and
(b) generate a random vector $y$ that lies in the range space of the matrix, and then verify that
it’s in the range space by solving for an input vector, $x$, such that $Mx = y$. 