Due: Wednesday, March 14 at 4 PM

- For this assignment you will create (or fix) 5 files; `hw5Ober.m`, `circle2Ober.m`, `fern1Ober.m`, and `fern2Ober.m`, and `fern3Ober.m`, where "Ober" is replaced by the first four letters of your last name. You will also be using `circleOber.m` from your previous assignment. BECAUSE YOU MAY NEED TO FIX THIS FUNCTION TO GET IT WORKING PROPERLY, RENAME THE (FIXED) FUNCTION TO `circle2Ober.m` TO USE FOR THIS ASSIGNMENT.
- This SCRIPT FILE `hw5Ober.m` should be formatted in such a way to be "publishable" as a webpage. Your webpage should appear on the G-drive in the same folders as your previous assignments. Within a browser, copy and paste the URL for your page into Moodle by the due date/time. NEW: YOU WILL PUBLISH `hw5Ober` WITH THE INCLUDE CODE OPTION SET TO "false". This can be changed under "Edit Publishing Options" on the toolbar menu or by typing `publish('hw5Ober', 'showCode',false)` in the command window.
- All other directions are the same as the previous assignment.

TIP 1: running the fern functions will take some time, so **do not publish until you know you have them working**, and when you do publish BE PATIENT!

TIP 2: for the last few problems, remember that the output of commands aren't shown until you have a `%%`, so you may want to have lines of just `%%` between some of the commands so it publishes a set of commands or comments, then creates output, publishes the next set of commands or comments, then output, etc.

To turn in:

1. The URL of the HTML file created by publishing your m-file submitted via Moodle. Have the URL appear in the notes part of turning in the assignment and make it a clickable link.
2. The m-files (both `hw5Ober.m` and all of the function files) uploaded to Moodle by the due date/time.

---

1. **Set up**
   (a) According to the Office of Marketing and Communications, the RGB code for Loyola colors are as follows:

Name	RGB Code	variable name
Loyola Green	0-104-87	loygreen
Loyola Gray	200-200-200	loygray
Loyola Gold	250-227-125	loygold
Loyola Red	179-8-56	loyred

   Save the `LoyolaColors.mat` file in the H-drive to your folder/directory and use the `load('LoyolaColors.mat')` command to load the vectors that can be used as colors.

   (b) One of the shapes we will use in this assignment will be an ellipse made with 100 points using the parametric equations $x = 2\cos(t)$ and $y = 3\sin(t)$ for an appropriate domain for t. Set up the vectors of these values (variable names of your choice!) to plot the ellipse in this problem an in subsequent problems. To test, use the `plot` command to plot the ellipse in Loyola Green with a Line Width of 2.

2. **Visualizing Linear Transformations.**

For this problem we will be visualizing some examples of linear transformations. The process will be the same or similar for all of them, so copy/paste will be your friend. Remember that using the `plot(x,y)` command, Matlab "connects the dots" of the points given with `x` and `y` as vectors of the x and y-coordinates of the points. We will use this idea to draw shapes and the transformed shapes. One of the shapes will be the ellipse discussed in #1b. Note that once the x and y vectors for the ellipse are created in #1b, as long as those vector names are unique you don't need to recreate them in subsequent problems. The other shapes will be made using your function `circleOber2` that is a (fixed) copy of the function created in Assignment #4.

Our shapes will plotted by forming a vector comprised of the x-values and and another comprised of the y-values of the points that make up our shapes. Each of our transformations will be associated with a 2×2 matrix A. What the matrices should be for reflections, rotations, scaling, and compositions are in the lecture/class notes. To visualize the transformation associated with a matrix A, we want to calculate $A\mathbf{v}$ for each of the points $\mathbf{v}$ (which is actually a column vector) that make up our shapes and store the answer in a new set of points. But this would be tedious and/or inefficient to do this for each point. Instead, we can do this all at once to all of the points by using matrix calculations. This is where it is easiest to have the original points in a $2 \times n$ matrix V (n is determined by the number of points that make up our shape). If we set up the matrix V so that the first row contains the x-values of the points, and the second row contains the y-values, then each *column* of V is one point. The beauty of matrix multiplication is that $T = AV$ will be all of the points that make up the transformed shape. For each of the linear transformations we will do the following.

- Formulate the vectors `x`, `y`, and matrix `V` that are composed of the vertices or points of the specified original shape. Plot the original shape in Loyola Green with Line Width thicker than the default.
- Figure out what the appropriate matrix A would be for the transformation, and display that matrix. In other words, when defining the matrix for each transformation, do not suppress the output so we can see the values in the matrix when it is published. It may be better to have a unique name for each transformation matrix, instead of A for each problem.
- Calculate $T = AV$ and plot the transformed shape in the stated color with Line Width thicker than the default.
- Plot the two stated points (markers) of the original shape in Loyola Green, using the "$\star$" marker and square marker, respectively. Plot the two transformed markers using the same shapes, but in the same color as the the transformed shape. Thus you should be able to see where the markers went under the transformation.
- Use the command `axis equal` and set the axes appropriately so the edges and points of the shapes don't touch the edge of the figure. **You may have to experiment with the order of `axis equal` and the command(s) to set the axes.**
- Use a legend with the titles "Original" and "Transformed" for the 2 shapes, and "1st marker" and "2nd marker" for the specified points. Note that the order of your plot commands will affect the order the titles should be in the legend! You may want to change the location of the legend with `'Location', 'Southeast'` (or some other location specification) so you can see all of the figures and legend. (This location may need to change for different transformations below.) For example,
  `legend('Sample title1', 'Sample title2', 'Location', 'Southeast').`

Perform the above process for each of the following transformations and shapes. Have the text that is in bold be the title of your figure (first example, `title('Reflection Example')`)

(a) **Reflection Example** with $\theta = \pi/6$. Original shape: the ellipse. Transformed shape in Loyola Red. First marker is the first point, the second marker is the 25th point. Also draw the reflection line of the angle θ (easiest to use trig!) as a black, dashed line.

(b) **Scaling Examples** with $c = 1.5$. and $c = 1/2$. Original shape: a hexagon using your `circle2Ober` function. Transformed shapes are in Loyola Grey and Loyola Gold, respectively. First marker is the first point, the second marker is the second point. Plot all three shapes: the original and two transformed, on the same figure.

(c) **Rotation Example** with $\theta =?$. Original shape: a pentagon. Notice that using your `circle2Ober` function to create the pentagon makes the shape "off kilter" or tipped. We want to orient the pentagon so the bottom side is perfectly horizontal, so we need to rotate it by an appropriate angle θ. Figure out what θ should equal and use this angle for your rotation example. Transformed shape should be in Loyola Red. First marker is the first point, the second marker is the second point.

(d) **Two Composition Examples** Original shape: a triangle using your `circle2Ober`. Transformed shapes are in Loyola Red and Loyola Gold, respectively. First marker is the first point, the second marker is the second point. The first transformation is a reflection about the y-axis (what is θ?) followed by a rotation with $\theta = \pi/4$. The second transformation is the rotation and then reflection. Plot all three shapes on the same figure.

(e) **Two random transformations** Original shape: the ellipse. Transformed shapes are in Loyola Gray and Loyola Gold. First marker is the first point, the second marker is the tenth point. The matrix for these random linear transformations are 2×2 matrices with random integer entries from -5 to 5.

(f) **Two random transformations** Original shape: the square. Transformed shapes are in Loyola Gray and Loyola Gold. First marker is the first point, the second marker is the second point. The matrix for these random linear transformations are 2×2 matrices with random integer entries from -5 to 5.

3. **Visualizing affine transformations** For this problem we will be doing the following:

- Save the `affineS18.mat` file in the H-drive to your folder/directory and use the `load('affineS18.mat')` command to load the matrices and vectors for the affine transformations (listed below).
- Draw the hexagon using your `circle2Ober` in black.
- Calculate the transformed hexagon using the affine transformations below (T_1 for first transformation, T_2 for second, etc.).
- Plot the transformed hexagons in Loyola Green, Loyola Red, and Loyola Gold, respectively.
- Use the command `axis equal` and define the axes to an appropriate view so the hexagons don't touch the edge of the figure.
- Have a legend identifying the original, first transformation, etc.
- No axis labels are needed for these graphs.
- The title of the graph should be "Affine Transformations."

Affine transformations $T_k(\mathbf{x}) = M_k\mathbf{x} + \mathbf{v}_k$, $(k = 1, 2, 3)$ where

$$M_1 = \begin{bmatrix} 0 & 1 \\ 3 & 1 \end{bmatrix},\ \mathbf{v}_1 = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix},\ \mathbf{v}_2 = \begin{bmatrix} -1.25 \\ -1.25 \end{bmatrix},\ M_3 = \begin{bmatrix} -3 & 0 \\ 2 & 3 \end{bmatrix},\ \mathbf{v}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

(NOTE: your points/vectors $\mathbf{x}$ should be a column vectors for this notation).

4. **Fern fractals** We will create functions `fern1Ober.m`, `fern2Ober.m`, and `fern3Ober.m` that have as input a positive integer n and optional input variables. The function should check for the validity of n, if not, return an error. You may use your previous functions to help with this error check. Optional input variables will be plot specifiers (color, marker size, marker type, etc.). The functions then will use the affine transformations defined by the *iterated function system* (IFS) below using n iterations to generate a figure of a fern fractal. The output of the function will be the value from the `toc` command. Each affine transformation T_k $(k = 1, 2, 3, 4)$ involves six parameters a, b, c, d, e, f:

$$T_k(\mathbf{x}) = M_k\mathbf{x} + \mathbf{v}_k = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \mathbf{x} + \begin{bmatrix} e \\ f \end{bmatrix}$$

where each transformation has probability p_k of being performed. The IFS for the fern we will be drawing is:

T	a	b	c	d	e	f	p
1	0	0	0	$\frac{4}{25}$	0	0	$\frac{1}{100}$
2	$\frac{17}{20}$	$\frac{1}{25}$	$-\frac{1}{25}$	$\frac{17}{20}$	0	$\frac{8}{5}$	$\frac{17}{20}$
3	$\frac{1}{5}$	$-\frac{13}{50}$	$\frac{23}{100}$	$\frac{11}{50}$	0	$\frac{8}{5}$	$\frac{7}{100}$
4	$-\frac{3}{20}$	$\frac{7}{25}$	$\frac{13}{50}$	$\frac{6}{25}$	0	$\frac{11}{25}$	$\frac{7}{100}$

The three functions will use slightly different algorithms. The first one is a basic one in which you may find the algorithm in a basic linear algebra text or even basic programming text.

(a) For `fern1Ober.m`, here is the algorithm:

i. Use the command `tic`

ii. Use the `load('fernIFSs18.mat')` that you've copied from the H-drive to load the above data with the matrices, vectors, and probabilities that are used to do the following tasks.

iii. Let $\mathbf{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

iv. If n is the only input, plot the point $\mathbf{x}$ as a point using "*" marker. Otherwise, plot the point with the additional inputs using `varargin`.

v. Use the random number generator `rand` to select one of the affine transformations T_k according to the given probabilities.

vi. Redefine $\mathbf{x}$ to be the new $\mathbf{x} = T_k(\mathbf{x}) = M_k\mathbf{x} + \mathbf{v}_k$.

vii. Plot the new point $\mathbf{x}$ as you did with the original $\mathbf{x}$.

viii. Repeat steps (vi) through (viii) so a total of n new points are plotted (where should the `hold on` and `hold off` commands be? How many times should your loop iterate?).

ix. The last command of the function should be storing `toc` into your output variable to capture the time it took to generate and plot the fern fractal.

Have a link to `fern1Ober.m` at this point on the webpage.

(b) For `fern2Ober.m`, we will adjust the process of `fern1Ober.m` by **vectorizing the code**.

i. Use the command `tic`

ii. Use the `load('fernIFSs18.mat')` that you've copied from the H-drive to load the above data with the matrices, vectors, and probabilities that are used to do the following tasks.

iii. Let $\mathbf{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and let $X = \mathbf{x}$.

iv. Use the random number generator `rand` to select one of the affine transformations T_k according to the given probabilities.

v. Redefine $\mathbf{x} = T_k(\mathbf{x}) = M_k\mathbf{x} + \mathbf{v}_k$. Concatenate X with the new $\mathbf{x}$. On subsequent iterations concatenate the matrix X with the new vector $\mathbf{x}$ (add another column to X that is the new $\mathbf{x}$).

vi. Repeat steps (v) through (vi) so a total of n new points are in the matrix $\mathbf{x}$

vii. If n is the only input, plot the points in the matrix X as a point using "*" marker. Otherwise, plot the points with the additional inputs using `varargin`. (Note that the first row of X contains the x-coordinates, second row contains the y-coordinates.)

viii. The last command of the function should be storing `toc` into your output variable to capture the time it took to generate and plot the fern fractal.

Have a link to `fern2Ober.m` at this point on the webpage.

(c) For `fern3Ober.m`, we will adjust the process of `fern2Ober.m` by **preallocating**.

i. Use the command `tic`

ii. Use the `load('fernIFSs18.mat')` that you've copied from the H-drive to load the above data and then pull the appropriate entries to define the matrices, vectors, and probabilities that are usable to do the following tasks.

iii. Define $\mathbf{x}$ as a matrix with 2 rows and $n + 1$ columns of zeros.

iv. For columns 2 through $n + 1$ of $\mathbf{x}$, use the random number generator `rand` to select one of the affine transformations T_k according to the given probabilities and let that column of $\mathbf{x}$ equal T_k(previous column of $\mathbf{x}$).

v. If n is the only input, plot the points in the matrix X as a point using "*" marker. Otherwise, plot the points with the additional inputs using `varargin`. (Note that the first row of X contains the x-coordinates, second row contains the y-coordinates.)

vi. The last command of the function should be storing `toc` into your output variable to capture the time it took to generate and plot the fern fractal.

Have a link to `fern3Ober.m` at this point on the webpage.

5. Display the fern using `fern1Ober`, `fern2Ober`, and `fern3Ober` with $n = 2000$. Before each fern command, use the command `clf`, and after each fern command, use the commands `axis equal` and `axis off`. For each of the ferns, use the `sprintf` command to create titles that say which fern function, the value of n, and how much time elapsed to create the fern fractal.

6. Display the fern using `fern1Ober`, `fern2Ober`, and `fern3Ober` with $n = 5000$, and specify the square marker, color Loyola Green, and have the marker size be smaller than the default. Before each fern command, use the command `clf`, and after each fern command, use the commands `axis equal` and `axis off`. For each of the ferns, use the `sprintf` command to create titles that say which fern function, the value of n, and how much time elapsed to create the fern fractal.

7. Display the fern using `fern2Ober` and `fern3Ober` with $n = 50{,}000$ and specify a marker and a color of your choice (do not use the default color). Use other specifiers if you want. Before each fern command, use the command `clf`, and after each fern command, use the commands `axis equal` and `axis off`. For each of the ferns, use the `sprintf` command to create titles that say which fern function, the value of n, and how much time elapsed to create the fern fractal.