Reminder: ***You MUST show your work to get credit.***

Project 4:

Linear and nonlinear transformations in $\mathbb{R}^2$

Goal

Illustrate a connection between transformations of geometric shapes in the plane and matrices.

General requirements

- You may work alone or with *one* other person. If you work with someone else, hand in *one* answer sheet with both of your names on it.
- No groups bigger than two. No collaboration between groups. **Please read "My policies on Projects" posted on the course website.**
- Write your answers on the answer sheet provided in the last few pages of this document. Staple[1] all the paper showing your *neatly presented* [2] work to the answer sheet.

Introduction

Using computer graphics has become part of everyday life for many people. One of the mathematical foundations of computer graphics is the fact that to change a geometric shape on the screen (i.e., in R^2), one needs to apply some transformation to it. While there are infinitely many different transformations, some of them share one underlying mathematical fact: they can be implemented by multiplying some matrix by the collection of vectors representing the geometric shape in question. Thus, any 2×2 matrix represents some transformation in the plane.

As you have learned, all such transformations are called *linear* transformations. Their hallmark is that they transform straight line segments into (different) straight line segments[3]. The reason why this occurs is that multiplication by a matrix changes — i.e., rotates, reflects, or stretches/shrinks — *all vectors in the same manner.*

In this Project, you will first create a geometric shape and then will apply transformations to it that take straight lines into straight lines. You will explore how a linear transformation can be implemented in a computer in two different ways.

In one of the Bonus problem, you will be given a chance to explore truly nonlinear transformations, which take straight lines into curved ones. This occurs because, unlike linear transformations, truly nonlinear ones change different vectors (i.e. line segments) by different rules.

This project makes heavy use of Matlab. *If you have any questions about the required Matlab commands*, you should use Matlab resources posted on the course website, especially the first of them, as well as Appendix A in the textbook. You can also look up the meaning and usage of any command by typing in the command window `help commandname`; for example: `help repmat`.

NOTE: All codes found in this Project description are posted online so that you could use them as templates for your own codes. See instructions on the Answer sheet about naming your codes.

[1] Your grade will be reduced by 5% if you hand in a pile of non-stapled sheets.

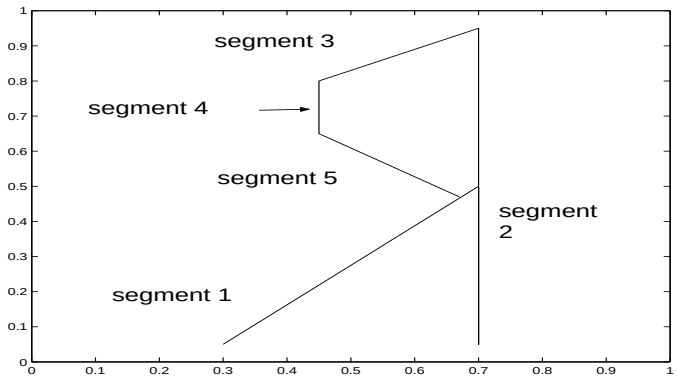
[2] I will reduce your grade by an amount left to my discretion in each particular case if your work is presented in a messy way and I have to waste time deciphering it.

[3] The converse of this is not true: as you saw in Section 3.7 and will see in this Project, there are transformations taking straight lines into straight lines, and yet those transformations are not linear according to the definition given by Section 3.7. There is a trick, mentioned later in this Project, which allows one to treat those transformations as linear, but we will not consider this in any depth here.

Step-by-step instructions:

Part 1 Use MATLAB to draw your first or last name's initial. The sample code shown below[4] produces such a drawing of the last letter of the Russian alphabet. The mathematical concept used in this code is the fact that equation $x = a + (b - a)t$, where $0 \le t \le 1$, defines a segment $[a, b]$.

```
% Draw the letter
t=[0:0.01:1];
            % parameter used to construct all segments
xmin_1=0.3; ymin_1=0.05; xmax_1=0.7; ymax_1=0.5;
            % coordinates of end points of the 1st segment
x_1=xmin_1+(xmax_1-xmin_1)*t; y_1=ymin_1+(ymax_1-ymin_1)*t;
            % 1st segment of the letter
xmin_2=0.7; ymin_2=0.05; xmax_2=0.7; ymax_2=0.95;
x_2=xmin_2+(xmax_2-xmin_2)*t; y_2=ymin_2+(ymax_2-ymin_2)*t;
            % 2n segment of the letter
xmin_3=0.7; ymin_3=0.95; xmax_3=0.45; ymax_3=0.8;
x_3=xmin_3+(xmax_3-xmin_3)*t; y_3=ymin_3+(ymax_3-ymin_3)*t;
            % 3rd segment of the letter
xmin_4=0.45; ymin_4=0.8; xmax_4=0.45; ymax_4=0.65;
x_4=xmin_4+(xmax_4-xmin_4)*t; y_4=ymin_4+(ymax_4-ymin_4)*t;
            % 4th segment of the letter
xmin_5=0.45; ymin_5=0.65; xmax_5=0.67; ymax_5=0.47;
x_5=xmin_5+(xmax_5-xmin_5)*t; y_5=ymin_5+(ymax_5-ymin_5)*t;
            % 5th segment of the letter
x_letter=[x_1 x_2 x_3 x_4 x_5]; y_letter=[y_1 y_2 y_3 y_4 y_5];
            % put all segments together into a letter
figure(401);  % open a new figure and name it "figure 401"
plot(x_letter,y_letter,'k')
axis([0 1 0 1])
            % plot the letter inside the rectangle [0 1] x [0 1];
            %  the option 'k' specifies the color (black)
```



Your task: Write a code to draw *your* initial, and produce a plot of the letter. See the answer sheet for the format of submission of your work.

[4]Note that each clarifying comment is added *after* the line that it clarifies.

To help you do Exercises 2 and 3, I will first present an example. You will need to mimic and extend it to do the remainder of this Project.

Example Let us first shift the initial you have created by 0.8 units horizontally and by -0.3 units vertically. This means that we need to add 0.8 to the x-coordinate of each point of the letter and also add -0.3 to (i.e., *subtract* 0.3 from) the y-coordinate of each point. This, as well as plotting of the shifted letter, can be done by the following code:

```
%  Version 1  (presented only to illustrate the basic concept;
%              its steps should NOT be mimicked)
%  Before running this code,
%  execute the code for part 1 and do NOT clear the workspace.
shift_horiz = 0.8;
shift_vert = -0.3;   % define the shift's components
x_letter_shifted = x_letter + shift_horiz;
y_letter_shifted = y_letter + shift_vert;
                     % shift the coordinates of the letter
figure(4021);
plot(x_letter,y_letter,x_letter_shifted,y_letter_shifted,'k')
axis([0 2 -0.5 1])
            % plot the original (default blue) and the shifted (black) letters
            % inside the rectangle  [0 2] x [-0.5 1]
```

A note is in order about Matlab's syntax used here. Above, `x_letter` and `y_letter` are row vectors of length 505. (Why so? [5]) On the other hand, `shift_horiz` and `shift_vert` are scalars. By rules of Linear Algebra, you cannot add a scalar to a vector. Yet, Matlab *allows* one to do so, interpreting this operation as the addition of the same scalar to *each entry* of the vector. However, this is where Matlab's loose interpretation of the matrix addition ends. Namely, it will *not* allow you to add two vectors of different lengths, or add a matrix to a vector. In particular (see below), it *will not allow* you to add a 2×505 matrix and a 2×1 column vector.

The above code does the job of shifting the letter. However, for Parts 2 – 3 of this Project, it will be convenient to represent your letter as a single matrix, rather than working separately with its x- and y-coordinates (i.e., `x_letter` and `y_letter`, respectively). To this end, let us consider the first point of the letter. In the xy-plane, it is represented by the *column* vector $\begin{pmatrix} \text{x\_letter}(1) \\ \text{y\_letter}(1) \end{pmatrix}$. Likewise, matrices

$$\begin{pmatrix} \text{x\_letter}(1) & \text{x\_letter}(2) \\ \text{y\_letter}(1) & \text{y\_letter}(2) \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \text{x\_letter}(1) & \text{x\_letter}(2) & \text{x\_letter}(3) \\ \text{y\_letter}(1) & \text{y\_letter}(2) & \text{y\_letter}(3) \end{pmatrix}$$

represent the first two and three points of the letter, etc. Generalizing, matrix

$$\text{letter\_matrix} = \begin{pmatrix} \text{x\_letter} \\ \text{y\_letter} \end{pmatrix}$$

represents the entire letter. What are its dimensions? Again, **do not proceed without answering this question**: it will come back and bite you. To check your answer, you may type `size(letter_matrix)` after running the code shown below.

[5] **Do not read on until you answer this question; otherwise you will likely run into problems later on.** If you are not sure, review the code in Part 1, paying attention to its first command line. If you do not understand what that line does, follow the suggestion in the paragraph before the Note at the end of the Introduction.

The following code (Version 2) will produce the same result as Version 1 above, but will handle the letter as a matrix rather than two separate rows of x- and y-components:

```
%  Version 2  (the version to be mimicked)
%  Before running this code,
%  execute the code for part 1 and do NOT clear the workspace.
% Step 1:
shift_horiz = 0.8;
shift_vert = -0.3;   % define the shift's components
shift_vector = [shift_horiz; shift_vert];
                     % define the vector of the shift
shift_matrix = repmat(shift_vector, 1, length(x_letter));
              % see the end of the Introduction
% Step 2:
letter_matrix = [x_letter; y_letter];
              % create the matrix containing the coordinates of the letter
% Step 3:
letter_matrix_shifted = letter_matrix + shift_matrix;
              % shift the letter
% Step 4:
x_letter_shifted = letter_matrix_shifted(1,:);
y_letter_shifted = letter_matrix_shifted(2,:);
              % these commands extract the 1st and 2nd rows of the shifted
              %  letter matrix and put them into two separate row-vectors
figure(4022);
plot(x_letter,y_letter,x_letter_shifted,y_letter_shifted,'k--')
axis([0 2 -0.5 1])
            % plot the original (default blue) and the shifted (black, dashed)
            % letters inside the rectangle  [0 2] x [-0.5 1]
```

In the assignments to follow, you should use Step 2 of this code without change and use Steps 3 and 4 after appropriate modification.

To complete this example, let us recall (see Example 4b in the lecture for Sec. 3.7) that translation of a vector is *not* a linear transformation. Above, we have translated not a vector but a matrix. However, this matrix is just a collection of column vectors representing individual points of the letter. Therefore, translation of this matrix is also *not* a linear transformation.[6] You will be asked if the transformation that you need to perform in Part 2 is linear or not.

Let us now return to the assignment of the Project.

Part 2 Write a Matlab code that reflects your letter about the y-axis *using a matrix–matrix multiplication*. One of the matrices is your letter's matrix; the other one *must be* a matrix derived by you in Project 1. Also, your work *must* satisfy the following requirements.

Note 1: You *must base your code on Version 2* of the code in the Example above. (Carefully read the

[6] There is a trick, commonly used in computer graphics, of how to *make* a translation appear as a linear transformation. Google 'homogeneous coordinates' if you are curious. We, however, will not consider this trick in this Project.

sentence immediately after that code. It tells you which Steps of the code you should mimic and which ones you should not. Also, if a Step is not mentioned there, you *should not mimic it*.)

Note 2: You must give your counterpart of the `shift_matrix` from the Example a *different name*, which would be more descriptive of the action of that matrix. Indeed, you are not shifting your letter in this exercise, right? So, the name of the matrix should reflect what it does to your letter. Your *score will be reduced* if the name that you give to the matrix does not reflect its action.

Note 3: When you learned array multiplication in Matlab, you probably learned the command `'.*'` (without quotes). Note that this is *not* the matrix–matrix or matrix–vector multiplication in the Linear Algebra sense! Therefore, you should *not* use it in this Project.

Note 4: While plotting the original and transformed letters in the same figure, you must use different line styles (see `help plot`) for the two letters.

Is this transformation linear? Justify your answer by referring to an appropriate material in the textbook or in the class notes.

Part 3 Here you will explore *the other way* of defining a linear transformation, *when you do not know its matrix but do know how it transforms a basis*. Find a description of this method in the Lecture Notes. Then, follow the steps listed below to reflect your letter about a line L: $y = \big(\tan(\pi/6)\big)\, x$.

(a) A convenient basis is formed by the vector along line L and by a vector perpendicular to it:

```
v1 = [cos(pi/6); sin(pi/6)];
v2 = [-sin(pi/6); cos(pi/6)];
```

Note that $\underline{\mathbf{v}}_1$ and $\underline{\mathbf{v}}_2$ form an orthonormal basis.

On a piece of paper, sketch line L and vectors $\underline{\mathbf{v}}_1$ and $\underline{\mathbf{v}}_2$. Also, sketch, and then record, the result of their reflection about line L:

```
Tv1 = % this should be the result of reflection of v1 about L
Tv2 = % this should be the result of reflection of v2 about L
```

(b) Find coordinates of each column of `letter_matrix` in the basis $\{\underline{\mathbf{v}}_1,\ \underline{\mathbf{v}}_2\}$. Use the fact that this basis is orthonormal and thus use a formula from Sec. 3.6 for finding coordinates in such a basis.

You can avoid doing the work column by column and instead do it in one step using matrix multiplication:

```
c1 = some_matrix_A * some_matrix_B;
c2 = similar
```

Here `c1` and `c2` are 1×505 row vectors containing the coordinates of each column of `letter_matrix` in the basis $\{\underline{\mathbf{v}}_1,\ \underline{\mathbf{v}}_2\}$. One of the "`some_matrices`" is a basis vector and the other is the `letter_matrix`. *Note that one of them must be transposed.* If you answered the questions about dimensions when working through the Example before Part 2, you should be able to easily identify these "`some_matrices`".

(c) Based on the formula

$$T(\mathbf{v}_1 c_1 + \mathbf{v}_2 c_2) \;=\; T(\mathbf{v}_1)c_1 + T(\mathbf{v}_2)c_2,$$

write a formula of the form:

```
letter_matrix_reflected = (Transformed vector 1)*(coordinates 1) + ...
                          (Transformed vector 2)*(coordinates 2);
% "..." at line's end means that current command continues on next line.
```

where the terms on the right-hand side come from your work in previous steps. Now, mimic Step 4 of the code in Example before Part 2 to plot the reflected letter. Show the original and reflected letters *in the same plot*.

Finally, to help you see that your reflected letter is indeed a reflection of the original letter about line L, add that line to your plot (see Note 1 below). *In addition*, Similarly, plot vectors $\{\underline{\mathbf{v}}_1,\ \underline{\mathbf{v}}_2\}$ (without arrows) *in the same figure*.

Note 1: One can plot a line in many ways; for example, plot `yL=xL*tan(pi/6)` versus `xL=[0 1]`. One can similarly plot a vector $\underline{\mathbf{v}} = [v(1),\, v(2)]^T$ by plotting `[0 v(2)]` versus `[0 v(1)]`.

Note 2: In order for your plot to look true to its description above, you need to use one of the two options, `axis('square')` or `axis('equal')` after your plot. Type `help axis` in Matlab for an explanation. Whichever option you use, your vectors $\{\underline{\mathbf{v}}_1,\ \underline{\mathbf{v}}_2\}$ must *look* perpendicular in your plot.

Note 3: Do *not* use different colors for your letters. These are (almost) impossible to tell apart in a black-and-white printout. Instead, use either different line styles (e.g., solid and dashed) or/and line widths; see examples under `help plot`.

Credit for any of the Bonus parts below will be given only if your work for that part is more than 60% correct and clearly presented.
Moreover, in the past, I had instances where students would submit hard copies of extra-credit plots that looked correct, but then when I would try to run their codes, the codes would either not run at all or produce different plots. Therefore, make sure that your extra-credit codes run *with a single click of the "Run" button* in Matlab's Editor. If I find this not to be the case, I will reduce your score by 5 points instead of giving you extra credit.

Bonus for Part 3

In a separate figure, plot `c2` versus `c1`. Similarly to how a plot of `y_letter` versus `x_letter` shows you the letter in the natural basis, the `c2` versus `c1` plot shows you the letter in the basis $\{\underline{\mathbf{v}}_1,\ \underline{\mathbf{v}}_2\}$. I.e., this is how the letter will look like to someone for whom the latter basis is natural. Explain the appearance of this plot *relative to the original plot* of your initial.
Credit will be given only for a correct *explanation with specific quantitative details*, not just for a plot or for answering the prompts in the *Hint* below.
Hint: Consider how the basis $\{\underline{\mathbf{v}}_1,\ \underline{\mathbf{v}}_2\}$ is related to the natural basis $\{\underline{\mathbf{e}}_1,\ \underline{\mathbf{e}}_2\}$. Compare this to how the plot of your transformed initial in Part 3 is related to the original initial.

Bonus–1 Write a Matlab code that rotates your letter about its right-most point by $\phi = \pi/6$. Follow these steps.
(i) Translate the letter so that its right-most point is at the origin.
(ii) Rotate the shifted letter about the origin. The transformation matrix that you will need was derived by you in Project 1; see also p. 236 in the textbook.
(iii) Translate the result of Step (ii) so as to undo the shift made in Step (i).

Note: Plot the original and transformed letters together, using different line styles. Make sure to adjust the scale of your axes (see Part 3) so that the transformed letter looks exactly as it description suggests.

Is this transformation linear? In justifying your answer, follow the lines of Example 5 in the notes for Section 3.7. You may also review the answers in the Example above.

Bonus–2 In the very first lecture of this course, I told you that all computer graphics (in particular, that employed in movies and video games) is based on Linear Algebra. That is, the moving objects that you see on the screen are produced by linear and nonlinear transformations applied to vectors and matrices. In the aforementioned lecture I used a waving flag as a simple example of a moving object. In this Bonus problem, since you already have a code producing your initial letter, I will ask you to make this letter, rather than a flag, wave. To preview what kind of results you may get after doing this Bonus part, type 'waving flag animation' into Google.

We will proceed as follows. First, I will present an example of how to produce a single plot of a letter with wavy contours. This example will consist of two steps. Then, **you will be asked** to modify my code to make these contours wave (i.e., move in a *periodic, oscillatory manner*) *in time*.

Example

To begin, run the code that produced your letter in Part 1 and do *not* clear the workspace.

To get the very basic understanding of the result of a nonlinear transformation, run the following code:

```
x_letter_waveall=1.2*x_letter+0.02*sin(20*x_letter)-0.01*cos(30*y_letter);
y_letter_waveall=0.9*y_letter+0.015*sin(25*x_letter+35*y_letter)+...
                 0.04*cos(15*x_letter-20*y_letter);
figure(4777);
plot(x_letter_waveall,y_letter_waveall,'k')
axis([-0.1 1.2 -0.1 1.2])
```

The lines of the letter are now curved, not straight as before. This is because by adding terms proportional to the nonlinear functions sine and cosine, we changed coordinates of each line in a highly nonlinear way.[7]

The above code changes *all* lines of the letter simultaneously. This is not what you would expect to happen to a flag. Rather, you would expect that its flagpole maintains its straight shape at all times while the banner at any time moment has a wavy shape. To produce such a picture, you would have to selectively modify only those parts of the flag which can move.

To illustrate the idea, below I show an excerpt of the code that makes only the top and right sides of the letter (i.e., segments 2 and 3) wavy while keeping the rest of it intact.

```
% Setup of segment 1 is the same as before.
xmin_2=0.7; ymin_2=0.05;
xmax_2=0.7; ymax_2=0.95;
x_2=xmin_2+(xmax_2-xmin_2)*t; y_2=ymin_2+(ymax_2-ymin_2)*t;
            % 2nd segment before it is made wavy
```

[7] As mentioned in the Introduction, nonlinear transformations make straight lines into curved ones because they change different line segments differently. Indeed, each segment of your letter is just a set of points on the screen. That is, that segment consists of many *sub*segments joining pairs of its adjacent points. A nonlinear transformation transforms each of these subsegments slightly differently; for example, it rotates the subsegment joining points 1 and 2 slightly differently than the subsegment joining points 2 and 3, etc. This makes the nonlinearly transformed segments of the letter become curved.

```
ax2=0.03;   % amplitude of waves of x-components of segment2
fx2=30;     % frequency of these waves; period = 2*pi/px2
x_2_wavy=x_2 + ax2*sin(fx2*t);
            % x-component of the wavy 2nd segment
            % Its form will be commented on after the code.
y_2_wavy=y_2; % for simplicity we assume the y-coordinates of
              %  segment 2 don't move

xmin_3=0.7; ymin_3=0.95;
xmax_3=0.45; ymax_3=0.8;
x_3=xmin_3+(xmax_3-xmin_3)*t; y_3=ymin_3+(ymax_3-ymin_3)*t;
            % 3rd segment before it is made wavy
ax3=0.02; fx3=5;   ay3=0.03;  fy3=4.5;
            % amplitudes and frequencies of waves on segment3
x_3_wavy=x_3 + ax3*sin(fx3*t);
y_3_wavy=y_3 + ay3*sin(fy3*t); %(x_3-xmin_3));
            % x- and y-components of the wavy 3nd segment

xmin_4=x_3_wavy(end); ymin_4=y_3_wavy(end);
            % The upper endpoint of segment 4 is made to move
            %  according to the motion of segment 3.
xmax_4=0.45; ymax_4=0.65;
x_4=xmin_4+(xmax_4-xmin_4)*t; y_4=ymin_4+(ymax_4-ymin_4)*t;
            % 4th segment
% Setting up the 5th segment, combining the segments, and plotting
% the entire letter are the same as before.
```

The following comments about this code are in order.

(a) Recall that "vector" `t` plays the role of the "coordinate" along the letter. Therefore, when we specify that the wave's shape (given by the function `sin` in the lines that define `x_2_wavy`, `x_3_wavy`, and `y_3_wavy`) depend on `t`, we essentially say that individual points of the segment are transformed according to their own individual rules. This is precisely the difference between a nonlinear transformation (as the one here) and a linear one (as those in Parts 2 and 3), where all points follow one and the same transformation rule.

(b) The wave's shape (i.e., the function `sin` mentioned above) is more or less arbitrary. The only restriction on it is that it must equal zero at the beginning (`t=0`) of each segment; this is needed to ensure continuity between two adjacent segments.

Also note that in setting up segment 3, I used *parametric* equations, {`(x_3_wavy(t), y_3_wavy(t))`}, which you learned in Calculus II and III.

(c) Note how we described each "wave" by two parameters: its amplitude and spatial frequency.

(d) The beginning point of segment 3 was forced to equal the end point of segment 2. This also served to ensure a continuous appearance of the waving letter. The same applies to the junction of segments 3 and 4.

Now, recall that the above code only produces a snapshot of a wavy flag *at just one instance in time*. An animation will need to wrap a loop around such a code in order to *make the picture change over time*. *This is your main task in this assignment*.

Specifically, **your task** is to modify the above code so as to make *at least two, but not all*, segments of *your* initial letter wave, exhibiting *two or three full cycles* (i.e., periods) of oscillations. In at least one of the segments, *both* x- and y-components are to vary nonlinearly in time. *Hint*: What functions of time exhibiting periodic oscillations do you know?

Please make sure that your code satisfies the following requirements.

- You will need to use Matlab's `for`-loop, which will allow you to specify the duration of the waving.
- Model the simplest form of waving, whereby the spatial shape of the wave (the '`sin`' in the Example code above) remains the same and *only the amplitude of the wave* changes *periodically in time*.
- When I run your code, I should be able to see a movie of your waving initial letter on my screen, more or less in real time. If you find that it runs too fast, consider placing a pausing command (something like `pause(0.1)`) after the plotting command; see `help` for `pause`.
- You (and I) should be able to change the duration of the waving by changing *just one parameter* of the `for`-loop.

Acknowledgment:
The idea of drawing one's initial is borrowed from Sec. 3.1.1 of R.C. Penny, "Linear Algebra: Ideas and Applications" (Wiley-Interscience, Hoboken, 2004).

Submission instructions:

1. Please submit as a single pdf file only page 2 (i.e., the next page), of this Answer Sheet, as well as all supporting handwritten work. Please do *not* include either this page with submission instructions or/and the Project description itself.

2. There will be a significant number of plots and printouts (of codes) in this project. In order both to organize your work and to save paper/file size, please put any of your printouts and plots into a Word file (see below), convert it to a pdf, append any handwritten work to the same pdf file, and submit only that file. If this is not done, I will **reduce your score by 5 points**.

   To include a figure into your main file, I recommend the following steps. Create a .png, .jpeg, or .tiff figure; Insert it into a Word document; Convert the Word document into a pdf. The syntax in Matlab for creating a .png file is: print -dpng 'foldername/filename'; see help print for more details. It is similar for other picture formats. Append any handwritten work to it as needed.

3. In addition to the printouts of your codes included in the submitted file, you must also e-mail me these codes. These codes should produce the action required in the assignment when I run them. For each code which you submit as *either* a hard or soft copy *but not both*, I will **reduce your score by 5 points**.

4. The subject line of your e-mail must contain the string MATH_122. It is case-insensitive; thus, math_122 or Math_122 will also work. However, the underscore must be present: e.g., math122 or math 122 will *not* work. Moreover, the subject line must indicate the fact that this is a submission of codes for Project 4. I will **reduce your score by up to 3 points** if this is not done.

5. Your codes must be named according to this convention: `p4_YourName(s)_partPart#.m`. E.g., my own code for Part 2 would be named `p4_tlakoba_part2.m`. For each code whose name does not follow this convention, I will **reduce your score by 2 points**.

6. Finally, suggestion and a **strong request**.

   Suggestion: Include the command `clear all` at the beginning of your code for Part 1.

   Strong request: Contrary to what you might have learned in other classes, I ask you **not** to put commands `clc` and `close all` at the beginning of your codes. If I want to clear my command window and close figure windows, I will do so myself, without unsolicited outside "help".

Answer Sheet is continued on next page

Reminder: ***You MUST staple pages with your neatly presented work to get full credit.***

Name(s): _____

Part 1 (27 points)
Include the plot and the code into the pdf document and also e-mail the code to me.

Part 2 (32 points)
Include the plot and the code into the Word document and also e-mail the code to me. Make sure you label (e.g., by hand) the original and the transformed letters in the plots.

Is this transformation linear? Please explain referencing a corresponding statement in the notes or in the first half of Sec. 3.7.

Part 3 (41 points)
Include the plot and the code into the pdf document and also e-mail the code to me.
Also, make sure to attach the hand-drawn sketch of line L and vectors $\underline{\mathbf{v}}_1$, $\underline{\mathbf{v}}_2$, $T(\underline{\mathbf{v}}_1)$, $T(\underline{\mathbf{v}}_2)$.

Bonus for Part 3 (5 points)
Include the plot and the code into the Word document. You may answer the question either here or in that document.

Bonus–1 (21 points)
Include the plot and the code into the pdf document and also e-mail the code to me.
Is this transformation linear? Please explain.

Bonus–2 (27 points)
Include the plot and the code into the pdf document and also e-mail the code to me.