AM205 HW2 Writeup

Jiawen Tong

Problem 1
See plot in Figure 1.

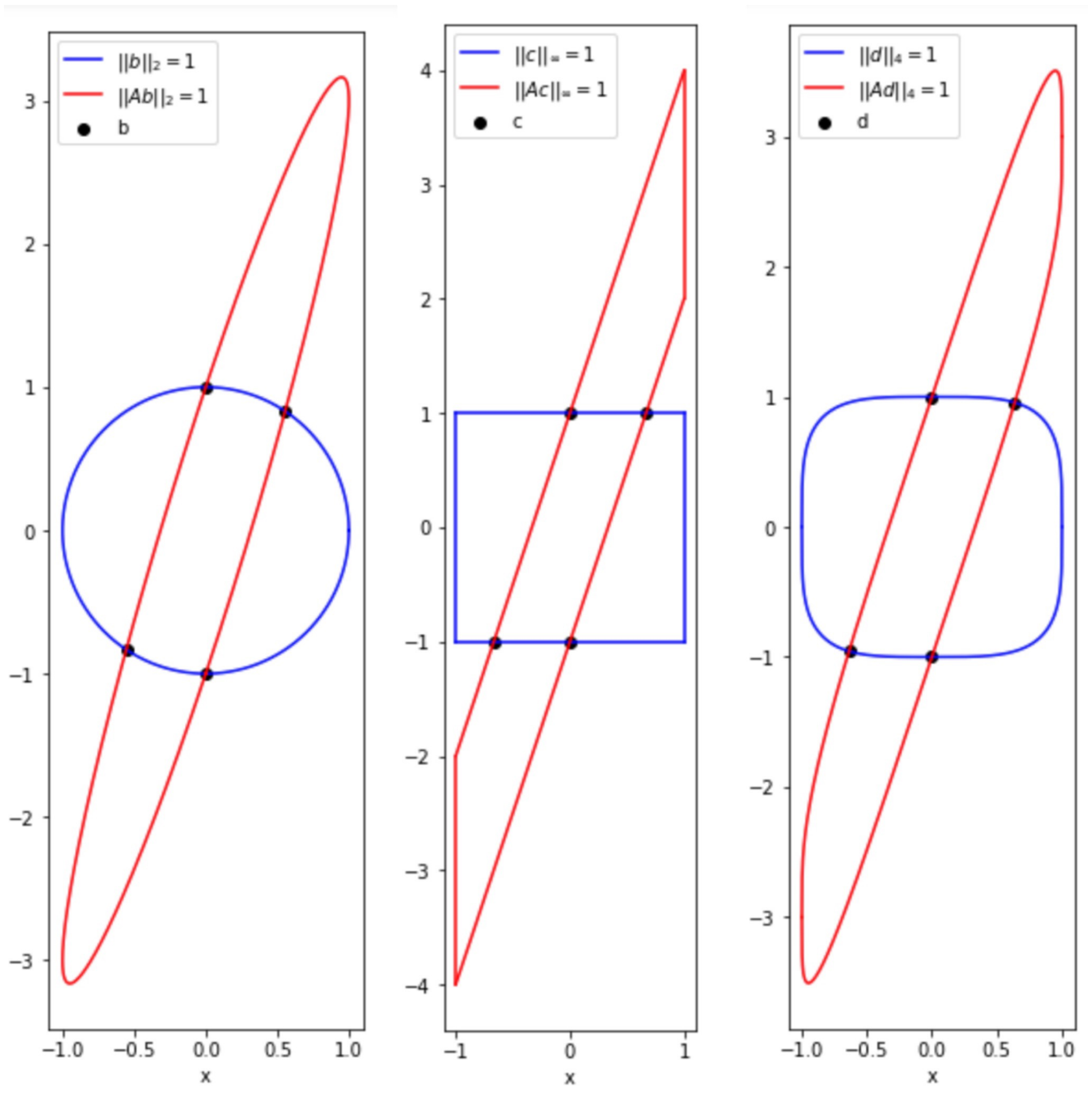

Figure 1: Problem 1: (a) (b) (c)

(a)

Let $b = \begin{bmatrix} x \\ y \end{bmatrix}$. By $||b||_2 = 1$, we get $\sqrt{x^2 + y^2} = 1$.

Plot the curve $||x||_2 = 1$ with parametric representation as

$$\begin{cases} x = cos\theta \\ y = sin\theta \end{cases}$$

Let $Ab = \begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 3x' - y' \\ x' \end{bmatrix}$.

Similarly, by $||Ab||_2 = 1$, we get $\sqrt{(3x' - y')^2 + x'^2} = 1$

which gives $Ab = \begin{bmatrix} 3x' - y' \\ x' \end{bmatrix} = \begin{bmatrix} cos\alpha \\ sin\alpha \end{bmatrix}$.

Plot the curve $||Ax||_2 = 1$ with parametric representation as

$$\begin{cases} x' = sin\alpha \\ y' = 3sin\alpha - cos\alpha \end{cases}$$

The intersection points of $||b||_2 = 1$ and $||Ab||_2 = 1$ could be solved as

$\begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} cos\theta \\ sin\theta \end{bmatrix} = \begin{bmatrix} 3cos\theta - sin\theta \\ cos\theta \end{bmatrix}$

$(3cos\theta - sin\theta)^2 + cos\theta^2 = 1$

$cos\theta = 0$ or $cos\theta = \frac{2}{3}sin\theta$

Therefore, the 4 points b are:

$b_1 = (0, 1) \quad b_2 = (0, -1) \quad b_3 = (\frac{2}{\sqrt{13}}, \frac{3}{\sqrt{13}}) \quad b_4 = (\frac{-2}{\sqrt{13}}, \frac{-3}{\sqrt{13}})$

(b)

Let $c = \begin{bmatrix} x \\ y \end{bmatrix}$. By $||c||_\infty = max(|x|, |y|) = 1$, we get

$$\begin{cases} |x| \le |y| = 1 \quad or \\ |y| < |x| = 1 \end{cases}$$

Equivalently, we can plot curve for $||c||_\infty = 1$ as

$$\begin{cases} x = 1, \quad -1 \le y \le 1 \\ x = -1, \quad -1 \le y \le 1 \\ y = 1, \quad -1 \le x \le 1 \\ y = -1, \quad -1 \le x \le 1 \end{cases}$$

Let $Ac = \begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 3x' - y' \\ x' \end{bmatrix}$.

By $||Ac||_\infty = max(|3x' - y'|, |x'|) = 1$, we get

$$\begin{cases} |x'| \le |3x' - y'| = 1 \quad or \\ |3x' - y'| < |x'| = 1 \end{cases}$$

Equivalently, we can plot the curve $||Ac||_\infty = 1$ as

$$\begin{cases} x' = 1, & 2 \le y' \le 4 \\ x' = -1, & -4 \le y' \le -2 \\ y' = 3x' + 1, & -1 \le x' \le 1 \\ y' = 3x' - 1, & -1 \le x' \le 1 \end{cases}$$

The intersection points of $||c||_\infty = 1$ and $||Ac||_\infty = 1$ could be solved as

$$\begin{cases} y = 1 \\ y = 3x + 1 \end{cases}$$

or

$$\begin{cases} y = 1 \\ y = 3x - 1 \end{cases}$$

or

$$\begin{cases} y = -1 \\ y = 3x + 1 \end{cases}$$

or

$$\begin{cases} y = -1 \\ y = 3x - 1 \end{cases}$$

Therefore, the 4 points c are:
$c_1 = (0, 1) \qquad c_2 = (\frac{2}{3}, 1) \qquad c_3 = (-\frac{2}{3}, -1) \qquad c_4 = (0, -1)$

(c)

Let $d = \begin{bmatrix} x \\ y \end{bmatrix}$. By $||d||_4 = (x^4 + y^4)^{-\frac{1}{4}} = 1$, we can plot the curve $||d||_4 = 1$ as

$$\begin{cases} y = (1 - x^4)^{-\frac{1}{4}} \qquad or \\ y = -(1 - x^4)^{-\frac{1}{4}} \end{cases}$$

Let $Ad = \begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 3x' - y' \\ x' \end{bmatrix}$.

By $||Ad||_4 = ((3x - y)^4 + x^4)^{-\frac{1}{4}} = 1$, we can plot the curve $||Ad||_4 = 1$ as

$$\begin{cases} y' = 3x' - (1 - x'^4)^{-\frac{1}{4}} \quad or \\ y' = 3x' + (1 - x'^4)^{-\frac{1}{4}} \end{cases}$$

The intersection points of $||d||_4 = 1$ and $||Ad||_4 = 1$ could be solved as

$$\begin{cases} F_1 = x^4 + y^4 - 1 = 0 \\ F_2 = (3x - y)^4 + x^4 - 1 = 0 \end{cases}$$

Using Newton's root finding method, the Jacob Matrix of (F_1, F_2) is calculated to iteratively approach the roots from our initial guess by observartion:

$inits = (2, 2), (-2, -2), (1, 10), (-1, -10)$

Therefore, the 4 points d are:
$d_1 = (0.6373, 0.9559) \quad d_2 = (-0.6373, -0.9559) \quad d_3 = (0, 1) \quad d_4 = (0, -1)$
Implementation details see code.

(d)

I got hints for this question from Chris Office Hour :D

We are solving the equations

$$\begin{cases} |3x - y|^p + |x|^p = 1 \\ |x|^p + |y|^p = 1 \end{cases}$$

Let $f(x) = |x|^p$, then we get

$$\begin{cases} f(3x - y) + f(x) = 1 \\ f(x) + f(y) = 1 \end{cases}$$

Therefore, we have $f(3x - y) = f(y)$. Since $f(x) = |x|^p = | - x|^p = f(-x)$ is an even function, the roots of this function (the family of points b, c, d) lie on two lines:

$$\begin{cases} line1: 3x - y = y \\ line2: 3x - y = -y \end{cases}$$

which could be equivalently written as

$$\begin{cases} line1: y = \frac{3}{2}x \\ line2: x = 0 \end{cases}$$

See Figure 2.

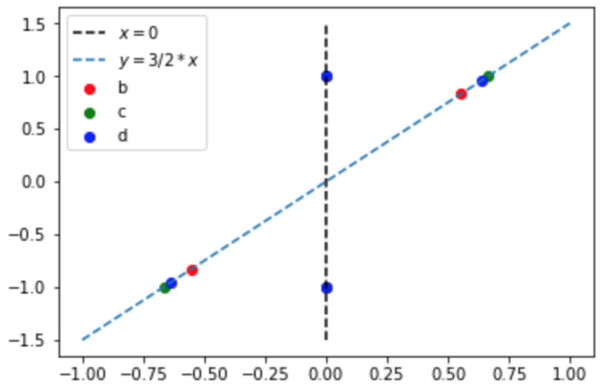

Figure 2: Problem 1: (d)

Problem 2

(a) Referenced lecture notes for forward substitution. See implementation details in code. My fsolve(L, b) raises an Exception reporting L being a singular matrix when L has 0 on its diagonal.

(b) Referenced lecture notes for backward substitution. See implementation details in code. My rsolve(U, b) raises an Exception reporting U being a singular matrix when U has 0 on its diagonal.

(c) Referenced lecture notes for normal LU factorization, and then implemented the binary version. See implementation details in code. My binary LU factorization can find U, L, P such that PA = LU, regardless of A being singular or non-singular.

When A is singular, we will find all zeros on and below U[j,j] for the column U[:,j]. Then:
1) a warning that 'A is singular' is made,
2) this column is skipped,
3) and LU factorization moves forward to deal with the next column.

(d)
To solve $Ax = b$, firstly get U, L, P = LU_factorize(A)

such that $PA = LU$

Thus, $PAx = LUx = Pb$

Let $Ux = y$, we get $Ly = Pb$.

Solve $y = fsolve(L, Pb)$,

and then solve $x = rsolve(U, y)$

Solved functions for q2_small and q2_large, and used the bin_mul() routine provided in the HW files to validate the solution x_small and x_large are correct.

Problem 3

(a)

The routine 'construct_A_light(m, n)' creates an m by n matrix A_light_game. Plug in $m = n = 7$ will give the binary matrix A asked in this question.

The idea to construct this A matrix is to loop through 0 to 48 to make A's columns. Let x denote some kind of presses, and b denote the corresponding light patterns. We can write out the relationship between A,x,b as:

For two presses, suppose we got $Ax_i = b_i, Ax_j = b_j (i \neq j)$

$Ax_i - Ax_j = A(x_i - x_j) = b_i - b_j$

$A\Delta x = \Delta b$

Let Δx_i denote pressing only the i-th button, and Δb denote the change in light patterns caused by pressing the i-th button, $(i = 0, 1, ..., 48)$ for A $\in R^{7x7}$.

Then each Δx_i has only 1 on its i-th entry and 0 otherwise. That's way we can construct A easily, column by column. Δb could be defined using the orthogonal toggling rules.

For instance, toggling the center button (3,3) on the board is making and lighting on itself (3,3) and its neighbors (2, 3), (4, 3), (3, 2), (3, 4).

$\Delta x[24] = 1$ makes

$$\begin{cases} \Delta b[24] = 1 & self \\ \Delta b[17] = 1 & upper \\ \Delta b[31] = 1 & lower \\ \Delta b[23] = 1 & left \\ \Delta b[25] = 1 & right \end{cases}$$

which creates the column A[:, 24]

See implementation details in code.

(b)

Plugged in the 5 light patterns plus my own pattern into b vector, and solve $Ax = b$ using the same procedure as in Problem 2 (d).

See presses and light patterns pairs in Figure 3.

(c)

For $m, n \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, calculate $f(m, n) = m * n - rank(A)$, where $rank(A)$ is the number of nonzero entries on U's diagonal in the LU factorization of A. The number of nonzero entries on U's diagonal is counted using

np.count_nonzero(np.tril(np.triu(U)))

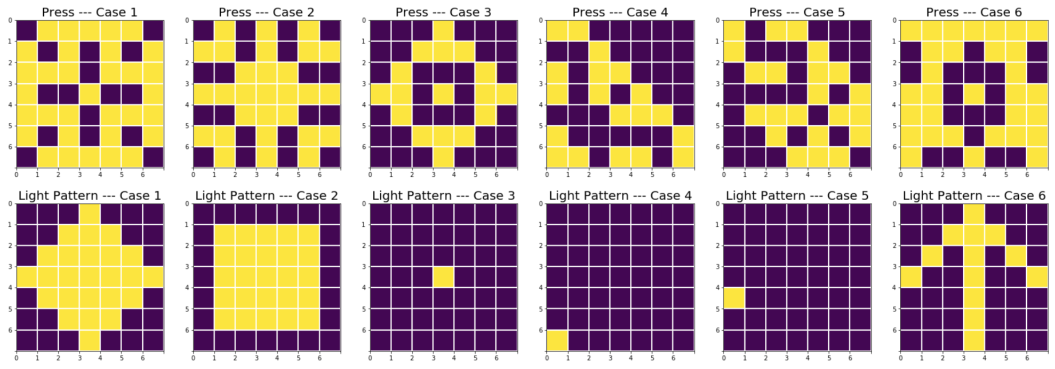

Figure 3: Problem 3 (b): Presses and Light Patterns

See visualization of the f matrix in Figure 4.

```
array([[0, 1, 0, 0, 1, 0, 0, 1, 0],
       [1, 0, 2, 0, 1, 0, 2, 0, 1],
       [0, 2, 0, 0, 3, 0, 0, 2, 0],
       [0, 0, 0, 4, 0, 0, 0, 0, 4],
       [1, 1, 3, 0, 2, 0, 4, 1, 1],
       [0, 0, 0, 0, 0, 0, 0, 6, 0],
       [0, 2, 0, 0, 4, 0, 0, 2, 0],
       [1, 0, 2, 0, 1, 6, 2, 0, 1],
       [0, 1, 0, 4, 1, 0, 0, 1, 8]], dtype=int8)
```

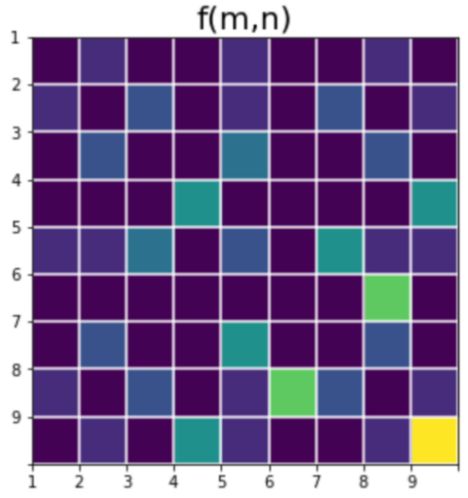

Figure 4: Problem 3 (c): Visualization of f(m, n)

Problem 4

(a)

generate_g(n) gets the lower triangle of n x n 1s, multiply it by (-1), plus on $2I$, and change the last column to be all 1s.

See implementation details in the code.

(b)

I learned to use np.polyfit() function after discussing with Rui Fang :D

$t(n) = \alpha n^{\beta}$

$log_{10}t(n) = log_{10}\alpha + \beta log_{10}n$

The slope in the $log_{10}log_{10}$ plot in Figure 5 could be determine by inspection. That is $\beta = 2$

Then, I used numpy.polyfit(ns, times, deg=2) to get the coefficients of the polynomial fitting function. The first coefficient returned is the value of α = 2.61630168423e-08

Plug in the coefficients to construct the ploy-fit curve and plot it together with the original time(n) curve. See Figure 5.

Explain/Discuss: $\beta = 2$ is very reasonable and natural since we are generating n x n matrix. The constraining operations are the np.ones(n) which has computational complexity $O(n^2)$.

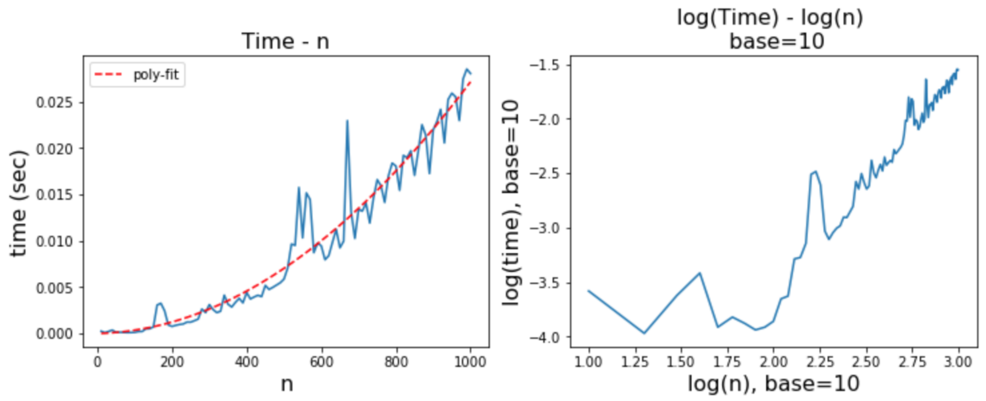

Figure 5: Problem 4 (b): time(n) in original with poly-fit and $log_{10}log_{10}$ scale

(c)
See the plot for 2-norm relative error as a function of n in Figure 6.

Explain/Discuss: As the error explodes for $n \geq 50$, we say Gaussian elimination with partial pivoting here to be numerically unstable.

The reason that there is no error for $n < 50$ but expanding error for $n > 50$ is that machine has finite precision to store floats. The U matrix in the LU factorization of G_n has 2^{n-1} as the coefficient for the last entry. The very small $\hat{x}$ $(\frac{1}{2^{50}} \sim 10^{-15})$ is below the machine precision limit (underflows) and thus is truncated to zero, causing the numerical instability.

(d)
See plot for the inequality in Figure 7.

Problem 6
(a)
Images from the 356 x 280 set.

Simply read in all 143 images, sum them up and divide by 143. Displayed the mean_leaf in both CMY (white background) and RGB (black background). See Figure 8.
(b)

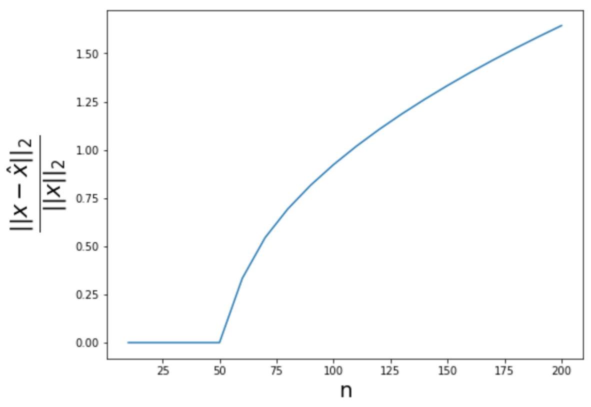

Figure 6: Problem 4 (c): 2-norm relative error

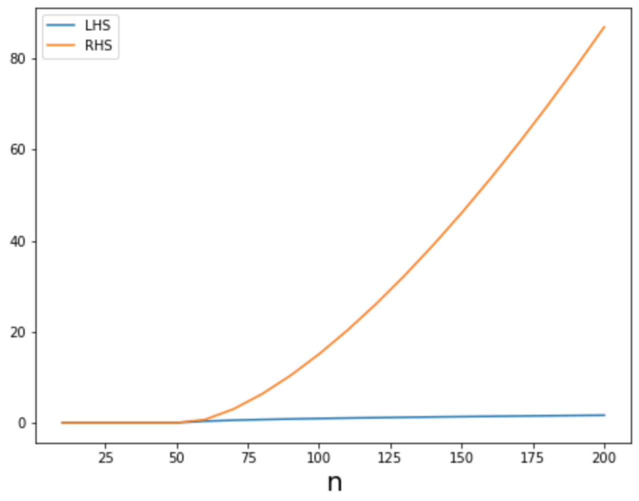

Figure 7: Problem 4 (c): Plot for the inequality

Constructed A_leaf as the $3mnxL$ matrix where each column A_leaf[:,j] is a mean centered leaf image.Did reduced SVD of A_leaf. For all columns of U, separated positive and negative values into uP and uN.

As instructed to plot uP and uN as images for the j=1,2,3 leaf images, I am not sure if this is asked based on **0-indexing or 1-indexing**. Thus, I ploted the first **4** columns of uP and uN as images in Figure 9 and 10.

(c)
I chose leaf007.png and plotted its closest point projection onto the subspace centered on mean_leaf and spanned by its first k left singular vectors (U[:, j], $j \in 0, 1, ..., k-1$). See Figure 11.

Comment: At k=1, the projection is blurred. As k grows, the projections have clearer contours.

(d)
Computed the distance vector dis_S[], and found that

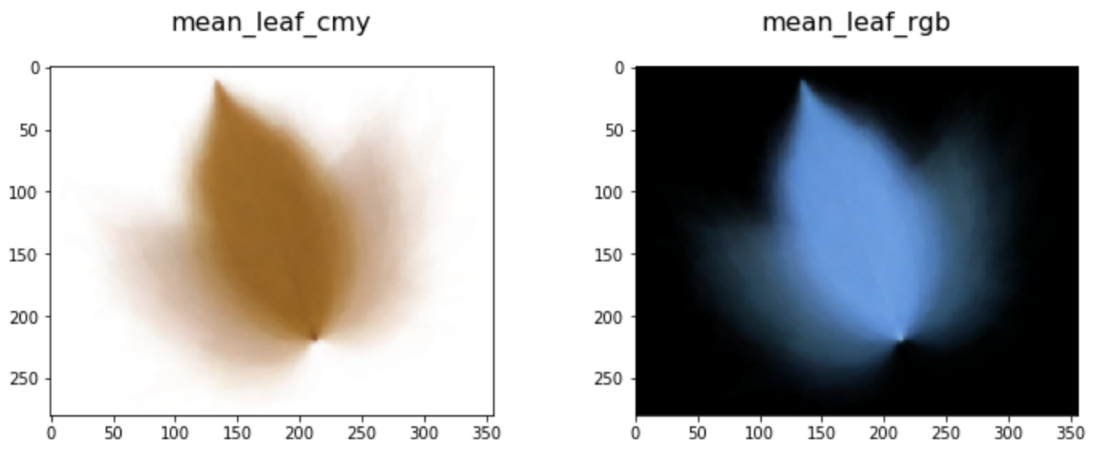

Figure 8: Problem 6 (a): mean leaf in CMY and RGB color mixing

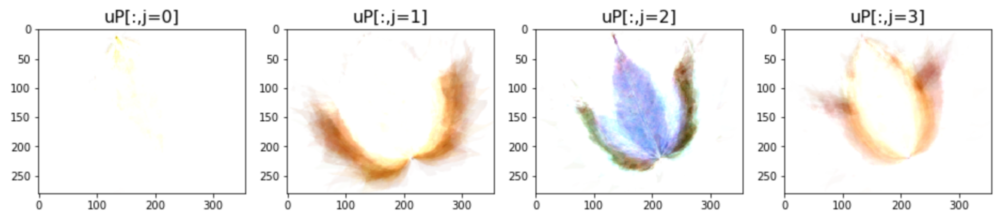

Figure 9: Problem 6 (b): First 4 columns of uP as image

min distance at $index = 57 \quad distance = 0.00936542808813$

max distance at $index = 120 \quad distance = 0.0756077798566$

Comment: The most matching projection is on the yellow eclipse leaf in Figure 12. The least matching projection is on the brown maple leaf in Figure 13. Since maple leaves has more variance in its contours, it matches my expectation that maple leaves are in general more diverse than the eclipse leaves.

(e)
As I computed and printed out the dis_R[], the least 3 values comes from leaf 0, 4, and 5.

$$\begin{cases} leaf0 \quad dis = 0.0190244162532 \\ leaf1 \quad dis = 0.027016464828 \\ leaf2 \quad dis = 0.0357692655408 \\ leaf3 \quad dis = 0.0391542212113 \\ leaf4 \quad dis = 0.0244164627802 \\ leaf5 \quad dis = 0.0263602089712 \\ leaf6 \quad dis = 0.0415619956668 \\ leaf7 \quad dis = 0.0533858591972 \end{cases}$$

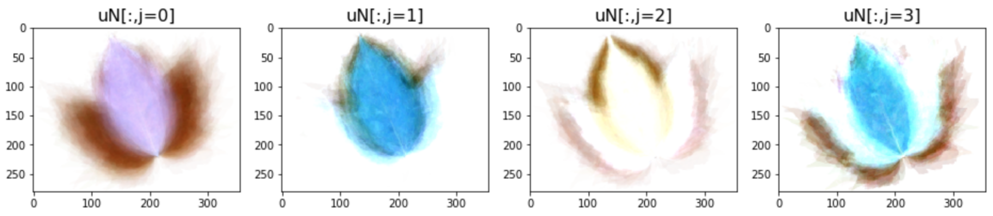

Figure 10: Problem 6 (b): First 4 columns of uN as image

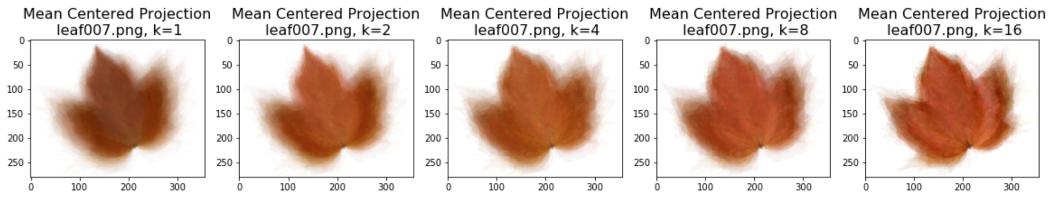
Mean Centered Projection leaf007.png, k=1
Mean Centered Projection leaf007.png, k=2
Mean Centered Projection leaf007.png, k=4
Mean Centered Projection leaf007.png, k=8
Mean Centered Projection leaf007.png, k=16

Figure 11: Problem 6 (c): Mean Centered Projections of leaf007.png (k = 1,2,4,8,16)

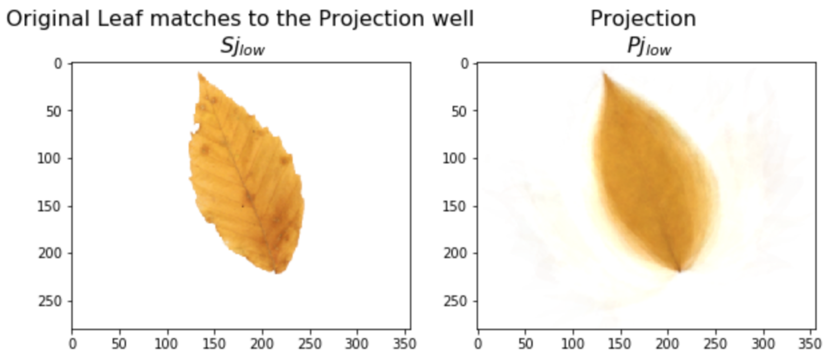

Figure 12: Problem 6 (d): Most matching leaf ($index = 57$)

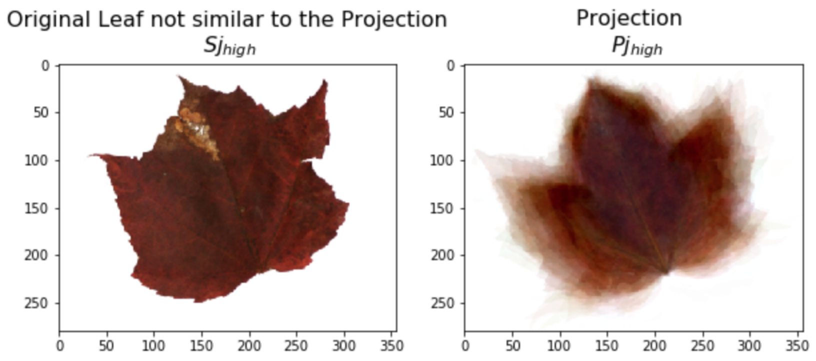

Figure 13: Problem 6 (d): Least matching leaf ($index = 120$)