# Problem 1

(a) Given the Matrix A

$$A = \begin{bmatrix} 4 & -1 \\ 1 & 0 \end{bmatrix}, \tag{1}$$

in this exercise I find four points $b \in^2$ such that $\|b\|_2 = 1$ and $\|Ab\|_2 = 1$. Conceptually, each point b is a vector with an x and y coordinate:

$$b = \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2}$$

To do this, I consider the formula and definition of the 2-norm and create a system of 2 equations for which I solve for x and y. Given that

$$||b|| = \left\| \begin{matrix} x \\ y \end{matrix} \right\| = 1 = \sqrt{x^2 + y^2} = 1 \tag{3}$$

and given that

$$||Ab|| = \begin{bmatrix} 4 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \left\| \begin{matrix} 4x & -y \\ x & 0 \end{matrix} \right\| = 1 = \sqrt{(4x-y)^2 + x^2} = 1 \tag{4}$$

we can solve for this system of equations:

$$\begin{cases} \sqrt{x^2 + y^2} = 1 \\ \sqrt{(4x-y)^2 + x^2} = 1 \end{cases} \tag{5}$$

$$\begin{cases} x^2 + y^2 = 1 \\ (4x-y)^2 + x^2 = 1 \end{cases}$$

The second equation will solve as:

$$\begin{aligned} (4x-y)^2 + x^2 &= 1 \\ 16x^2 - 8xy + y^2 + x^2 &= 1 \end{aligned}$$

We know from above that $y^2 + x^2 = 1$, hence $16x^2 - 8xy = 0$ and $8x(2x - y) = 0$, where x will be either $x = 0$ or x will have to be $x = y/2$ given that $2x - y = 0$. By plugging in $x = y/2$ in the formula $x^2 + y^2 = 1$ we get that $y = -(\sqrt{4/5}$ or $y = +(\sqrt{4/5})$, which subsequently leads to $x = y/2$ hence $x = (\sqrt{4/5})/2$ and $x = -(\sqrt{4/5})/2$. Given this, the four points b are:

$$b_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, b_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, b_3 = \begin{bmatrix} 0.447 \\ 0.894 \end{bmatrix}, b_4 = \begin{bmatrix} -0.447 \\ -0.894 \end{bmatrix} \tag{6}$$

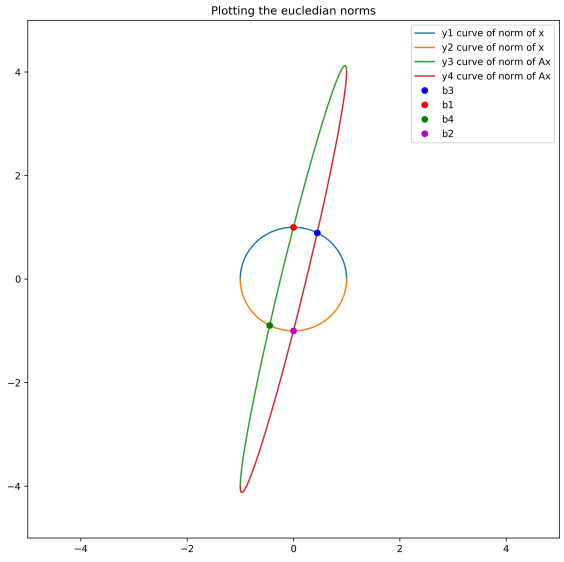
The curves $y_1$ and $y_2$ of $\|x\| = 1$ are coming from:

$$\begin{aligned}\sqrt{x^2 + y^2} &= 1\\ 1 &= x^2 + y^2\\ 1 - x^2 &= y^2\\ y_1 &= +\sqrt{1 - x^2}\\ y_2 &= -\sqrt{1 - x^2}\end{aligned}$$

The curves $y_3$ and $y_4$ of $\|Ab\| = 1$ come from:

$$\begin{aligned}\sqrt{(4x - y)^2 + x^2} &= 1\\ 1 &= (4x - y)^2 + x^2\\ 1 - x^2 &= (4x - y)^2\\ \sqrt{1 - x^2} &= (4x - y)\\ \sqrt{1 - x^2} - 4x &= -y\\ y_3 &= 4x + \sqrt{1 - x^2}\\ y_4 &= 4x - \sqrt{1 - x^2}\end{aligned}$$

These plotted are:




(b) Here I find four points $c \in^2$ such that $\|c\|_\infty = 1$ and $\|Ac\|_\infty = 1$. Given that conceptually each point c is a vector with an x and y coordinate,

$$c = \begin{bmatrix} x \\ y \end{bmatrix},$$

I consider the infinity norm of c to be $max(|x|, |y|) = 1$. Subsequently, from

$$||Ac|| = \left\| \begin{matrix} 4x & -y \\ x & 0 \end{matrix} \right\| = 1$$

it follows that the infinity norm of $||Ac||$ is stemming from $max(|4x - y|, |x|) = 1$.

Considering the infinity norm of c leads us to four possible cases:

- Case 1: $|x| = 1$ with $x = 1$. In this case, the absolute value of $y$ has to be smaller than one. Hence the conditions of case ones are: $|x| = 1$, $|y| < 1$, $|y| < |x|$.

  From $max(|4x - y|, |x|) = 1$ and by following the conditions of Case 1 we have $max(|4x - y|, 1) = 1$, hence $|4x - 1| < 1$. From here we get $4 > y > 3$ and $4 < y < 5$. From this case we see that $x = 1$ breaks our conditions, so we cannot consider it for any point c.

- Case 2: $|x| = 1$ with $x = -1$. In this case the conditions are: $|y| = 1$, $|x| < 1$, $|x| < |y|$.

  From $max(|4x - y|, |x|) = 1$ and by following the conditions of Case 2 we have $max(|4x - y|, 1) = 1$, hence $|4x - 1| < 1$. From here we get $-4 > y > -5$ and $-4 < y < -3$. From this case we see that also $x = -11$ breaks our conditions, so we cannot consider it for any point c.

- Case 3: $|y| = 1$ with $y = 1$. In this case, the absolute value of $x$ has to be smaller than one. Hence the conditions of case ones are: $|y| = 1$, $|x| < 1$, $|x| < |y|$.

  From $max(|4x-y|, |x|) = 1$ by following the conditions of Case 3 we get $|4x-1| = 1$, where $4x - y = 1$ and $4x - y = -1$. From solving these last 2 equations, we get $x = 1/2$ and $x = 0$ respectively. Hence the first 2 points are:

$$c_1 = \begin{bmatrix} x = 0.5 \\ y = 1 \end{bmatrix}, c_2 = \begin{bmatrix} x = 0 \\ y = 1 \end{bmatrix},$$

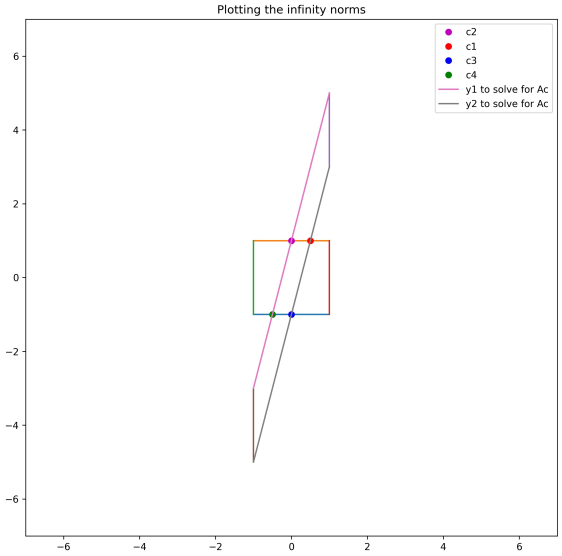
  .

- Case 4: $|y| = 1$ with $y = -1$. In this case the conditions are: $|y| = 1$, $|x| < 1$, $|x| < |y|$.

  From $max(|4x-y|, |x|) = 1$ following the conditions of Case 2 we get $|4x-1| = 1$, where $4x - y = 1$ and $4x - y = -1$. From solving these last 2 equations with $y = -1$, we get $x = 0$ and $x = -1/2$ respectively. Hence the first 2 points are:

$$c_3 = \begin{bmatrix} x = 0 \\ y = -1 \end{bmatrix}, c_4 = \begin{bmatrix} x = -0.5 \\ y = -1 \end{bmatrix}$$

These points can be seen at the intersection of the plotted curves of the norms $\|x\|_\infty = 1$ and $\|Ac\|_\infty = 1$.

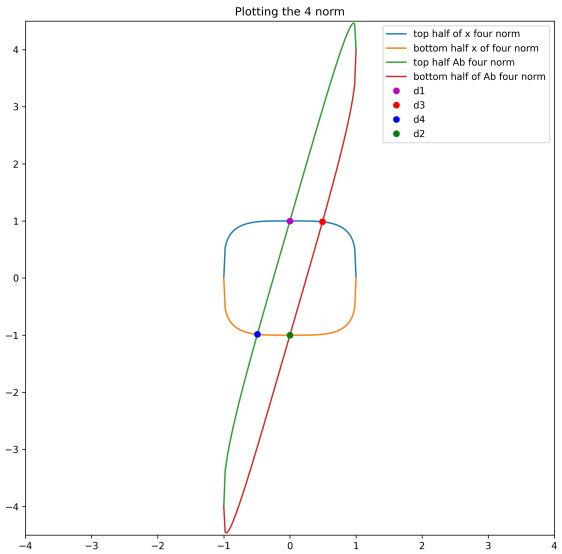



(c) Now I consider a vector function $f : R^2 \to R^2$ defined as

$$f(d) = (\|d\|_4 - 1, \|Ad\|_4 - 1), \tag{7}$$

where $d$ is a vector holding an x and a y coordinate, i.e. $d \in R^2$.

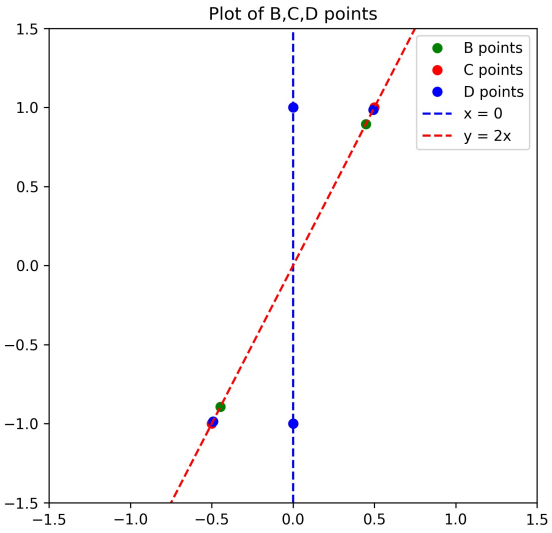
To plot the 4-norm curves, I take into consideration $\sqrt[4]{x^4 + y^4} = 1$ and $\sqrt[4]{(4x - y)^4 + x^4} = 1$. From the former, we get the curves $y = \pm(1 - x^4)^{1/4}$ and from the latter we get the curves $y = \pm(1 - x^4)^{1/4} + 4x$. These can be visualised as:



From here we can intuitively see that one point is likely going to solve as (0, 1) and another as (0, -1). Hence if we use the points $b_1$ and $b_2$ from 1.a as starting points for our newton method coded in 1c (this has been coded by taking into consideration the Jacobian, given that we are in a 2-dimensional space), we end up converging in the same place. However, by picking point $b_3$ as starting point, the function converges to 0 at the point $d_3 = (0.492479, 0.98495812)$. By picking point $b_4$ as starting point, the function converges to 0 at the point $d_4 = (-0.49247906, 0.98495812)$.

(d) Intuition from the Office Hour of Yue Sun.

The points $b$, $c$, and $d$ (12 points in total) lie on two straight lines. This can be visualised as:




This is the case because, in the general case, we are solving for equations of p-norms. These are:

$$\begin{cases} \sqrt[p]{x^p + y^p} = 1 \\ \sqrt[p]{(4x - y)^p + x^p} = 1 \end{cases} \tag{8}$$

If we consider the vector function of a point $b$, or $c$ or $d$, i.e. $f(b)$, $f(c)$ or $f(d)$, we are then also considering the $f(x)$ and $f(y)$ of each point, as each point has a x and a y coordinate, where f is the function indicating the $p_{th}$ norm. Subsequently:

$$\begin{cases} f(x) + f(y) = 1 \\ f(4x - y) + f(x) = 1 \end{cases} \tag{9}$$

By equating these:

$$\begin{aligned} f(x) + f(y) &= f(4x - y) + f(x) \\ f(y) &= f(4x - y) \end{aligned}$$

Additionally, we know that $f(x)$ is an even function, because $|x|^p = |-x|^p$ and because $|y|^p = |-y|^p$, meaning $f(x) = f(-x)$. Consequently, the roots will lie on the two lines:

$$\begin{cases} f(4x-y) = f(y) \\ f(4x-y) = f(-y) \end{cases} \tag{10}$$

which leads to a line to be:

$$\begin{aligned} 4x - y &= y \\ 4x &= 2y \\ y &= 2x \end{aligned}$$

and the latter line to be:

$$\begin{aligned} 4x - y &= -y \\ 4x &= 0 \\ x &= 0 \end{aligned}$$

# Problem 2

(a) Understood binary operations in the Office Hours of Chris.

In exercise 2 the matrices at hand are binary. Consequently, to code the bitwise addition, the symbol "hat" has been used. To code the bitwise multiplications, the symbol & has been used. To code forward substitution, the algorithm of slide 38 of Unit 2 has been used. This follows:

$$\begin{aligned} x_1 &= b_1/l_{11} \\ x_2 &= (b_2 - l_{21}x_1)/l_{22} \\ &\dots \\ x_j &= (b_j - \sum_{k=1}^{j-1} l_{jk}x_k)/l_{jj} \end{aligned}$$

In the code, an error is raised if the matrix is singular. If a matrix if singular, it means that it does not have full rank. Meaning, there are some columns that are a linear combination of each other. To check how many columns are a linear combination of each other, I can look at the amount of 0s in the diagonal of L. However, the exercise simply asks whether the matrix is singular or not. Hence, to code it I check if the product of the diagonal goes to 0. In python I do this with this: L.diagonal().prod() == 0. If this does go to 0, there is at least 1 column that is a linear combination of another, hence not a full rank matrix, hence a singular matrix.

(b) To code backward substitution, the algorithm of slide 37 of Unit 2 has been used. This follows:

$$x_n = b_n/u_{nn}$$
$$x_{n-1} = (b_{n-1} - u_{n-1,n}x_n)/u_{n-1,n-1}$$
$$\dots$$
$$x_j = (b_j - \sum_{k=j+1}^{n} u_{jk}x_k)/u_{jj}$$

Also here I raise the error of singular matrix in the same fashion as before.

(c) Exercise 2c has been coded following the algorithm and pseudocode provided in slide 70 of Unit2. To create the LU factorization, that pseudo code uses partial pivoting.

The pivoting process loops through each column, and is looking for the biggest value in that column so that to put that element in another row so that that biggest element ends up being on the diagonal.

The partial pivoting process in U means that each loop looks in a column and looks for the biggest value (here the biggest value happens to be 1, but it can be any highest absolute value in a normal case) and switches the right part of that row from j onwards to another row so that that value ends up being on the diagonal. It is called partial pivoting because only a part of the row is switched with another, not the entire row. In the case of U, from the j_th column to the end, i.e. what is to the right of the j, is switched. In the partial pivoting in L, what is up to column j, i.e. to the left of the matrix, is switched with another row so that the row that has the biggest element on column j ends up being on the diagonal.

In the partial pivoting also the matrix P gets updated. It will keep track of what has been changed by switching (pivoting) the pair-rows but entirely, not just partially.

(d) LU-factorization can help us solve more easily a system of equations. Given an LU-factorization $PA = LU$, it is possible to solve $Ax = b$.

$$Ax = b$$
$$PAx = Pb$$
$$PAx = LUx = Pb$$

Where $y$ can be backsolved from $Ly = Pb$ using forward substitution. Later, $x$ can be backsolved from $Ux = y$, by using backward substitution.

In exercise 2d I read the text files provided, and get an A matrix and a b vector. The goal is to back solve for x using the LU factorization method explained above. First I backsolve for $y$ with forward substitution, then then for $x$ with backward substitution.

As sanity check, I use the code provided by the professor to check if the results are right. I utilize the code he provided with bin_mull, to check if the binary multiplication of $Ly.$ matches the binary multiplication of $Pb$. Same sanity check to see if the binary multiplication of $Ax$ later will match with $b$.

The result of the small txt files is a column vector x with dtype=int8:

$$x = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0].T$$

The result of the large txt files is a column vector x with dtype=int8 can be seen in the code. The result is a column vector, it's a numpy array with 71 rows and 1 column.

# Problem 3

(a) Understood the rule behind A in the Chris' Office Hour.

The game lights out provided in the exercise has a 7x7 grid. However the rule behind the grid is a 49x49 matrix. The rule behind this is that, where $k$ has been pressed, its neighbours will light up and will get the value of one, as the button that has been pressed, hence: $k = 1$, $k + 1 = 1$, $k - 1 = 1$, $k - n = 1$, $k + n = 1$ will light up. The rule is $k = n * i + j$, where $n = 7$.

(b) Once the matrix A with the rules of the game has been coded, now we can back-solve for x. The vector b is the vector of what has been light up. Consequently, I take each box given in the exercise and I reshape it to a column vector with shape (49,1), i.e. 49 rows and 1 column. Given the previously coded matrix A, with shape 49x49, I can now back solve for x with first forward substitution ( I find $y$ ), then I find $x$ with backward substitution, as explained in exercise 2d.

The vector x shows what buttons have been pressed. The buttons pressed have 1 as value, and the buttons not pressed are 0. After this, I reshape (in python I do .reshape()) the results, i.e. these x vectors, back to a 7x7 box. These plotted are shown in Figure 1 and Figure 2, plotted with the command spy in python. Black boxes are the ones with 1 values, the whites have 0. The additional box picked by me is the last one, box6.

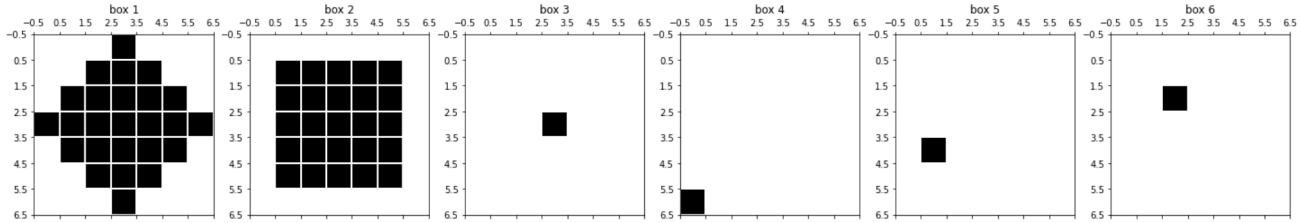



Figure 1: Lighted buttons, lighted pattern given in the exercise. These are the vectors $b$.

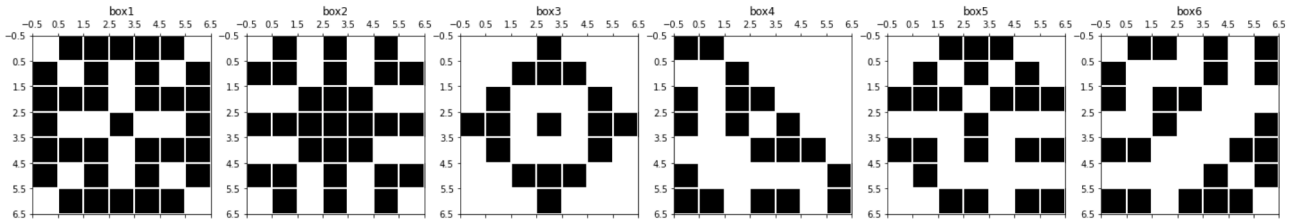



Figure 2: Buttons that have been pressed are the black buttons, with 1 values. These are the vectors $x$.

(c) Understood nullspaces in the Office Hour of Jovana.

In the exercise 3c, we are told from the prompt that the given matrix A, is a $49x49$ matrix (where the game is the $7x7$ grid, where grid is $m * n$, where $m = 7$ and $n = 7$), where this matrix is non-singular, i.e. it has full rank. Given this prompt, I can know from the exercises that the 49 columns of A (the matrix storing the back-rules of the game) are linearly independent from each other.

Additionally, in this particular game, this means that every combination of lights can be created by pressing buttons.

This is not always the case for any squared matrix mn x mn A and for any game with grid n x m. In the exercise we are told to try to find the ranks and nullspaces for grids m x n with m in range 1 to 10 and n in range from 1 to 10.

To check if my code works, I first try to find the nullspaces of the $49x49$ matrix, these should be 0. Consequently, I check that the diagonal of U from the LU factorization of A should have all 1 values. By looking at the diagonal np.diagonal(U) and counting the 0s with np.count_nonzero(d==0), I can see that my assumptions held. Hence my code there should be ok. After this I went ahead and coded the general case of any grid m x n, i.e. any matrix with shape mn x mn. The result is in Figure 3.

```
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]])
```

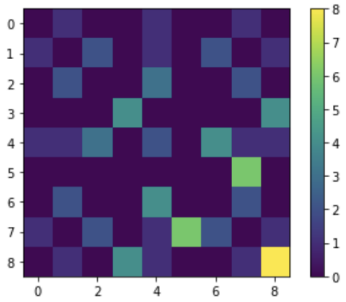


Figure 3: 2 On the left: Nullspaces. On the right: nullspaces shown with command plt.imshow

# Problem 4

(a) In exercise 4a we are given a very rare and particulare case of a matrix $G_n$ that has 1 in the elements of the diagonal, has 1 in the elements of the last column, has $-1$ in the left side of the lower left triangle of the matrix, and has the rest 0s.

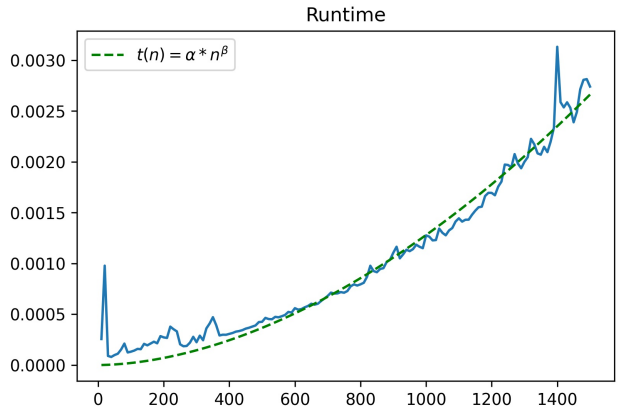
$$G_n = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 1 \\ -1 & 1 & 0 & 0 & \dots & 1 \\ -1 & -1 & 1 & 0 & \dots & 1 \\ -1 & -1 & -1 & 1 & \dots & 1 \\ \vdots & & & & \ddots & \vdots \\ -1 & -1 & -1 & -1 & \dots & 1 \end{bmatrix} \tag{11}$$

The code to recreate $G_n$ is in the jupyter notebook file attached.

(b) The code has 2 for loops in n because I am creating an n x n matrix to generate the G matrix. Given these 2 loops, the code is said to have a big o notation of $O(n^2)$.

To code the runtime and find it from the code, in python I import from time the process time and I find the run time for this code. The run time can be also fitted as $t(n) = \alpha * n^{\beta}$. With the python function curve_fit I can find the parameters $\alpha = 4.86e-09$ and $\beta = 1.806$. Every time I run the code I have slightly different numbers, the last time I ran it I actually got beta to be 2.0081546670375574.

$\beta = 1.8$ follows my intuition of the $O(n^2)$, being $\beta$ very close to 2.

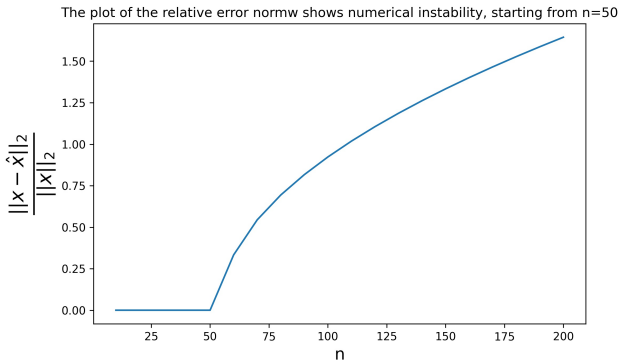



(c) Intuition from Chris Office Hours.

From the graph below we can see that there is numerical instability starting from around $n = 50$ to $n > 50$. This is the case because the relative error $\frac{||x - \hat{x}||_2}{||x||_2}$ starts to grow exponentially from around $n = 50$. G has a good condition number. Hence the problem $Gx = b$ is ok and is a mathematically well defined problem. Consequently, the problem comes from the algorithm, in this case the LU factorization.

This is the case because from the LU factorization of the matrix G, the last column of U has powers of 2. I.e entries of $2^n$ (like $2^1$, $2^2$, $2^3$ up to $2^{n-1}$).

When solving with first forward and then backward substitution, we can incur into rounding errors in the backward subtitution, because we are dividing for bigger powers of 2 (i.e. $2^{50}$, $2^{51}$, and so on), where python will truncate these results to 0 - even though the result isn't actually 0. This is what caused numerical instability in this case.



# Problem 6

(a) In exercise 6a I read all the images with io.imread and get the average of them. To get the RGB version of the average image I do 1 - the average of the CMY image. See Figure 4.

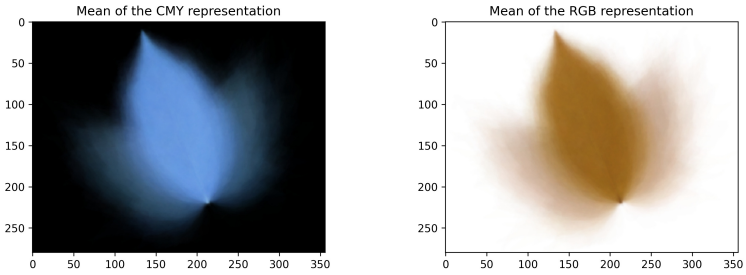



Figure 4: 2 On the left: Average CMY image. On the right: average RGB image

(b) In 6b I perform reduced singular value decomposition with the python command np.linalg.svd(A, full_matrices= False). After this, I find the positive and negative components. These are defined as:

$$u_{i,j}^{\mathrm{P}} = \max\left\{0, \frac{u_{i,j}}{d_j}\right\}, \qquad u_{i,j}^{\mathrm{N}} = \max\left\{0, \frac{u_{i,j}}{c_j}\right\}, \tag{12}$$

Then I plot the negative component and the positive component for images 1, 2 and 3. These plotted are in Figure 5 and 6. As I am confused about indexing, given that in python the first one is at 0, I will add also image 0.

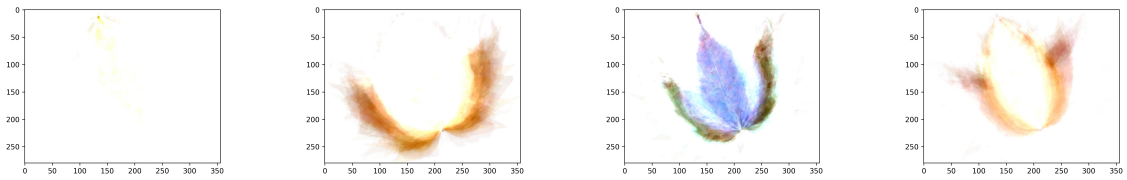


Figure 5: Positive components as images for j=0,1,2,3.

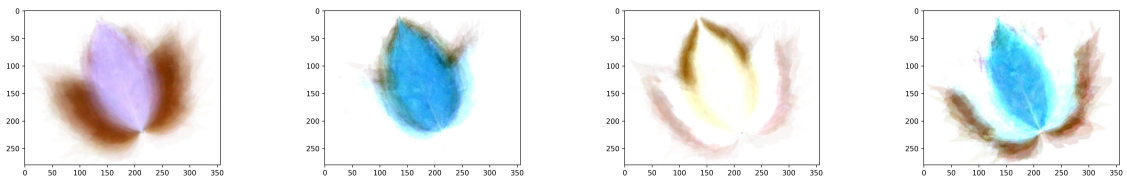


Figure 6: Negative components as images for j=0,1,2,3.

(c) In exercise 6c I perform the reconstruction of an arbitrarily chosen image T for 1 rank, 2 ranks, 4 ranks, 8 ranks and 6 ranks. The reconstruction is mean centered, this can be noticed by how on equation 13 the mean is first substracted from the image and then added again. Further, from this equation:

$$P(T,k) = S_{mean} + \sum_{j=1}^{k} \left[u_{:,j}(T - S_{mean}))\right] u_{:,j}. \tag{13}$$

it follows that each term of that some has a rank of 1. Each singular value holds unique information and has a rank of 1. The more singular values you add up, the higher the rank of the matrix you are using for the reproduction of the image, i.e. more information and more linearly independent columns, i.e. more independent pieces of information, which leads to a more accurate reconstruction. We can see this in Figure 7, where for higher k, for higher ranks, the image became more and more accurate.

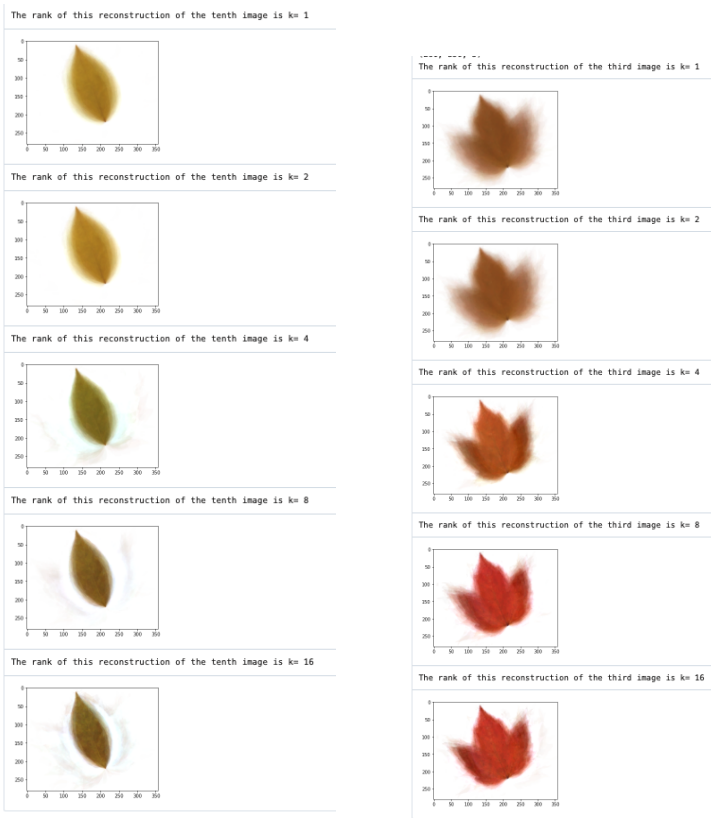



Figure 7: Image reconstruction of two chosen leaves. The higher the rank, the higher the accuracy and clarity, as expected.

(d) In this exercise I look at the reconstruction of an image j with rank 8 and how much that is far away from the actual image j. The smallest this distance, the better the leaf matches its projection, i.e. the better the reconstruction of the image matches the real image. This distance is defined as:

$$distance(S_j) = \frac{1}{mn}\|S_j - P(S_j, 8)\|_2^2, \tag{14}$$

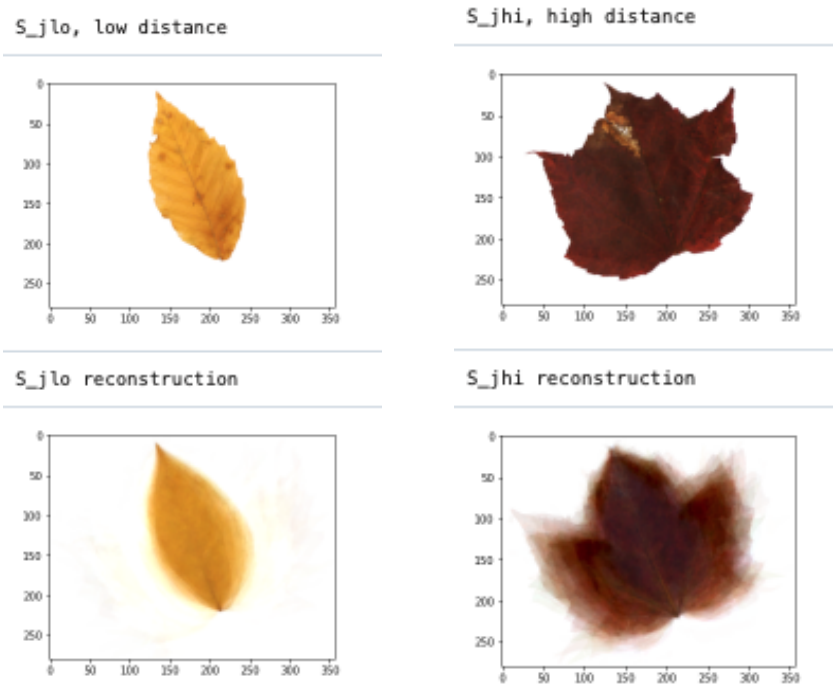



Figure 8: Low distance and high distance image for the same amount of k=8 in the projection.

These two images reconstructions (projections) do not have the same clarity (distance) for the same amount of ranks. For the image with the highest distance, we can notice that the reconstruction with 8 ranks is less clear than the reconstruction of the image with smallest distance, which also uses 8 ranks. In order for the image with highest distance to have the same clarity (same distance) as the image with lowest distance, the image with highest distance needs to add more ranks to its re-construction. See Figure 8.

(e) In exercise 6e we are given another folder with other eight additional leaves, some are from Harvard Yard and some are from New Hampshire - we don't know which ones are from where. The prompt suggests to find the leaves that best match their projection (i.e. with the smallest distance to their projection / reconstruction, here of rank 8). The best 3, i.e. the ones with the smallest distance, will also happen to come from New Hampshire. These are in Figure 9. The distances are smallest for the images 0, 4 and 5. These are the indexes referring to image 143, image, 147 and 148.

```
(8, 280, 356, 3)
This is the image number  0
This is its distance 0.01902441625316513
This is the image number  1
This is its distance 0.027016464827970304
This is the image number  2
This is its distance 0.03576926554078236
This is the image number  3
This is its distance 0.039154221211284305
This is the image number  4
This is its distance 0.024416462780224267
This is the image number  5
This is its distance 0.026360208971227653
This is the image number  6
This is its distance 0.04156199566682986
This is the image number  7
This is its distance 0.05338585919717463
```

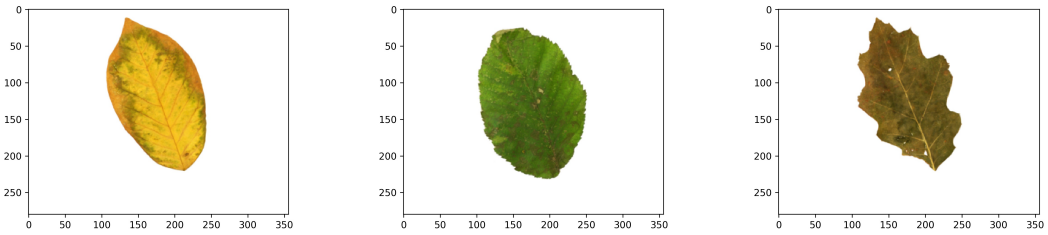


Figure 9: Leaves from New Hampshire