Optimization Models in Engineering—Homework 4

1. Answer the following questions about SVD (note: all calculations should be done by hand):

   i) Consider the matrix
$$A = \begin{bmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ 2 & 2 & -1 \end{bmatrix} \tag{1}$$
   Show that the columns of A are orthogonal to each other. By using this fact, find a singular value decomposition of A.

   ii) Find a singular value decomposition of the matrix
$$C = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ 2 & 2 & -1 \end{bmatrix} \tag{2}$$

   iii) Consider the optimization problem
$$\min_{B \in \mathbb{R}^{3\times 3}} \|C - B\|_F \qquad \text{s.t.} \qquad \text{rank}(B) \le 2 \tag{3}$$
   Find the optimal solution B^* and compute the error $\frac{\|C-B^*\|_F^2}{\|C\|_F^2}$.

Solution.

i) We first consider the first two columns of A and show that they are orthogonal (i.e., zero inner product):
$$\begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}^\top \begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix} = -1 \times 2 + 2 \times -1 + 2 \times 2 = -2 - 2 + 4 = 0.$$
We can then do the same for the first and third columns:
$$\begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}^\top \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} = -1 \times 2 + 2 \times 2 + 2 \times -1 = -2 + 4 - 2 = 0.$$
Finally, we repeat for the second and third columns:
$$\begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix}^\top \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} = 2 \times 2 + -1 \times 2 + 2 \times -1 = 4 - 2 - 2 = 0.$$
Hence, all columns of A are orthogonal to each other.
Next, we use this property to find a singular value decomposition (SVD) for A. Note that while all columns are orthogonal to each other, for SVD we need to find orthogonal matrices, which has orthonormal columns. Observing that
$$\|(-1,2,2)\|_2 = \|(2,-1,2)\|_2 = \|(2,2,-1)\|_2 = \sqrt{4+4+1} = 3,$$
we know that $\frac{1}{3}A$ is a orthogonal matrix.
As a result, we can decompose A into
$$A = U_A \Sigma_A V_A^\top = I_3 \cdot (3I_3) \cdot (\tfrac{1}{3}A).$$
Since all identity matrices are orthogonal, we can verify that $U_A = I_3$ is orthogonal. Moreover, $\Sigma_A = 3I_3$ implies that A has three singular values all equal to 3. Recalling that $\frac{1}{3}A$ is orthogonal, we conclude that the above is a valid SVD of A.

ii) For the purpose of singular value decomposition (SVD), we need to decompose C into a multiplication of three matrices $U\Sigma V^\top$, where Σ is diagonal and $U, V^\top$ are orthogonal.

Consider the pre-multiplier of C, which we denote as M. We want to find its singular value decomposition $M = U_M \Sigma_M V_M^\top$. Note that M is anti-diagonal, and we can decompose an anti-diagonal matrix into an orthogonal anti-diagonal matrix multiplied by a diagonal matrix. I.e.,

$$M = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{U_M} \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\Sigma_M} \underbrace{I_3}_{V_M^\top}, \tag{4}$$

which completes the SVD of M.

Alternatively, we can also do the usual decomposition steps covered in discussions to find the SVD of M. To do so, we start by finding the eigenvalues of $M^\top M$. In this case,

$$M^\top M = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix}^\top \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

which is already diagonal.

Hence, the eigenvalues of $M^\top M$ are 9, 4, and 1, and the singular values of M are simply the square root of these eigenvalues, i.e., 3, 2, and 1. Furthermore, the corresponding normalized eigenvectors of $M^\top M$, which are the right singular vectors of M, are the standard basis vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. Hence, as in (4), we again find that

$$\Sigma_M = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad V_M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_3.$$

Now, we only need to find the matrix U_M, which consists of the left singular vectors of M, or the eigenvectors of $MM^\top$. Note that

$$MM^\top = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix}^\top = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 3 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 9 \end{bmatrix},$$

which is again diagonal. Hence, the eigenvectors are again standard basis vectors. The eigenvector corresponds to the eigenvalues of 9, 4, 1 are $(0, 0, 1)$, $(0, 1, 0)$, and $(1, 0, 0)$ correspondingly, and therefore we again arrive at the decomposition in (4):

$$U_M = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Recall that in part i), we have shown that $\frac{1}{3}A$ is an orthogonal matrix. Putting everything together, we have

$$C = U_M \Sigma_M V_M^\top A = U_M (3\Sigma_M) V_M^\top (\tfrac{1}{3}A) = U_M (3\Sigma_M)(\tfrac{1}{3} V_M^\top A) = U_M (3\Sigma_M)(\tfrac{1}{3}A),$$

where the last step holds because $V_M = I_3$.

Because $3\Sigma_M$ is diagonal and both U_M and $\frac{1}{3}A$ are orthogonal, we have completed our SVD. The final result is

$$C = U_M (3\Sigma_M)(\tfrac{1}{3}A) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 9 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix}.$$

iii) By the Young-Mirsky theorem, one solution to this problem is simply the rank-2 SVD approximation of C, which is originally rank-3 (minimizing the square of the Frobenius norm is equivalent to minimizing the norm itself).

Since we already have the SVD of C, we can conveniently calculate B^* as

$$B^* = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 9 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 4 & -2 & 4 \\ -3 & 6 & 6 \end{bmatrix}.$$

To compute the relative error, we first calculate the matrix C:

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ 2 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & -1 \\ 4 & -2 & 4 \\ -3 & 6 & 6 \end{bmatrix}$$

Next, we calculate the relative error $\frac{||C-B^*||_F^2}{||C||_F^2}$.

Note that when calculating $||C||_F^2$, we can use the sum of squares of the singular values:

$$||C||_F^2 = 3^2 + 6^2 + 9^2 = 9 + 36 + 81 = 126.$$

For $||C - B^*||_F^2$, we can use the sum of squares of the omitted singular values:

$$||C - B^*||_F^2 = 3^2 = 9.$$

Hence, the relative error is

$$\frac{||C - B^*||_F^2}{||C||_F^2} = \frac{9}{126} = \frac{1}{14}.$$

2. An exam with m questions is given to n students. The instructor collects all the grades in an $n \times m$ matrix G where G_{ij} shows the grade of student $i \in \{1, 2, ..., n\}$ for question $j \in \{1, 2...., m\}$. By analyzing the matrix G, the goal is to design a difficulty score for each question that shows the difficulty level of that question. As a naive approach, one may consider the average grade $\frac{\sum_{i=1}^{n} G_{ij}}{n}$ as the difficulty score of question j. To understand the issue with this difficulty score, assume that $n = m = 2$ and G is equal to

$$G = \begin{bmatrix} 50 & 100 \\ 50 & 0 \end{bmatrix} \tag{5}$$

where the minimum and maximum grades for each question are 0 and 100. In this example, both questions have the same average grade of 50. Both students have done poorly on question 1. For question 2, one student got the highest grade possible while the other student got 0 (which may imply that the student was not prepared for that question rather than the question being hard). Question 1 seems to be much harder than question 2 due to the distribution of the grades while the average grades cannot provide any useful information. To address this issue for arbitrary values of n and m, we propose an optimization model for the design of difficulty scores.

i) Consider the optimization problem

$$\min_{B \in \mathbb{R}^{n \times m}} \|G - B\|_F \qquad \text{s.t.} \qquad \text{rank}(B) \leq 1 \tag{6}$$

We decompose the optimal solution B^* as xy^T, where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$. Assume that $x, y \geq 0$ (note: if no student receives a zero score on any question, then it can be proven that x and y are automatically non-negative vectors). Assume that the error $\frac{\|G-B^*\|_F^2}{\|G\|_F^2}$ is small. Explain how a difficulty score can be designed for each question in terms of x and y.

ii) Consider the case with $n = 3$ and $m = 5$, together with the grade matrix

$$G = \begin{bmatrix} 100 & 90 & 100 & 80 & 70 \\ 80 & 70 & 60 & 70 & 80 \\ 60 & 50 & 40 & 50 & 60 \end{bmatrix} \tag{7}$$

Using Part (i), design a difficulty score for each question, and rank the questions from the hardest to the easiest based on their scores (note: you can use a computer code for SVD calculations).

Solution.

1) The terms x and y can be interpreted as follows:
   - Student Ability (x): Each element x_i represents an inherent ability or performance level of student i.
   - Question Easiness (y): Each element y_j represents the easiness level of question j.

   Note that conceptually y is correlated with the scores that students receive, and hence a smaller value indicates a more difficult problem.

   Moreover, consider the case where G is rank-1 and in the form of $xy^\top$, then the grades for questions i and j are xy_i and xy_j respectively, which means that although different students receive different grades, they are all scaled by the factor $\frac{y_j}{y_i}$ when we go from question i to question j.

2) By the Young-Mirsky theorem, we can solve the problem (6) with a rank-1 SVD approximation, where we find the largest singular value of G and the corresponding left and right singular vectors. I.e., we want to find $G = usv^\top$, where $u \in \mathbb{R}^3$, $s \in \mathbb{R}$, and $v \in \mathbb{R}^5$ are the SVD results. Then, we can simply set x as su and y as v. Here we are only interested in the easiness y, specifically its ranking, and therefore y's scaling does not matter.

   We can find u, s, and v via "SVDS" in MATLAB or Python. SVDS computes a subset of singular values/vectors, which is more efficient than computing the full SVD. Note that there are two solutions for u

and v, one with all entries non-negative and the other non-positive. Here we want to take the non-negative solution so that y still has the real-world meaning of "easiness". The result is the following:

$$u = \begin{bmatrix} 0.7044 \\ 0.5751 \\ 0.4159 \end{bmatrix}, \quad s = 280.2, \quad v = \begin{bmatrix} 0.5046 \\ 0.4441 \\ 0.4339 \\ 0.4190 \\ 0.4292 \end{bmatrix} = y. \tag{8}$$

Recall that a smaller value in y indicates a more difficult problem. Hence, the ranking from the hardest to the easiest is 4, 5, 3, 2, 1.

MATLAB code:

```matlab
% Define the matrix G
G = [100, 90, 100, 80, 70;
     80, 70, 60, 70, 80;
     60, 50, 40, 50, 60];

% Compute the largest singular value and corresponding singular vectors
k = 1; % Number of singular values/vectors to compute
[x, s, y] = svds(G, k);

% Flip signs if needed to make sure the singular vectors are non-negative
if sum(x) < 0
    x = -x; y = -y;
end

% Sort the hardness of problems
[~, p] = sort(y, 'ascend'); % lower to higher is hardest to easiest
disp(y'); disp(p')
```

Python code:

```python
import numpy as np
from scipy.sparse.linalg import svds

# Define the matrix G
G = np.array([[100, 90, 100, 80, 70],
              [80, 70, 60, 70, 80],
              [60, 50, 40, 50, 60]], dtype=float)

# Compute the largest singular value and corresponding singular vectors
# k is number of singular values/vectors to compute
x, s, yt = svds(G, k=1, which='LM')

# Flip signs if needed to make sure the singular vectors are non-negative
if x.sum() < 0:
    x, yt = -x, -yt

# Sort the hardness of problems
p = yt.argsort() # lower to higher (default) is hardest to easiest
print(f"Easiness score: {yt.squeeze()}")
print(f"Ranking by hardness: {p.squeeze()}")
```

3. Consider a picture of the Berkeley logo named Berkeley1.png that you can download from bCourses $\rightarrow$ Files $\rightarrow$ HW.

   i) Convert the image to a grayscale image, and store its data into a matrix A.

   ii) Plot the singular values of A. To do so, draw the point (k, σ_k) in $\mathbb{R}^2$ for $k = 1, ..., \min(m, n)$, where m and n denote the dimensions of A and σ_k is the k^{th} largest singular value of A.

   iii) Consider the optimization problem

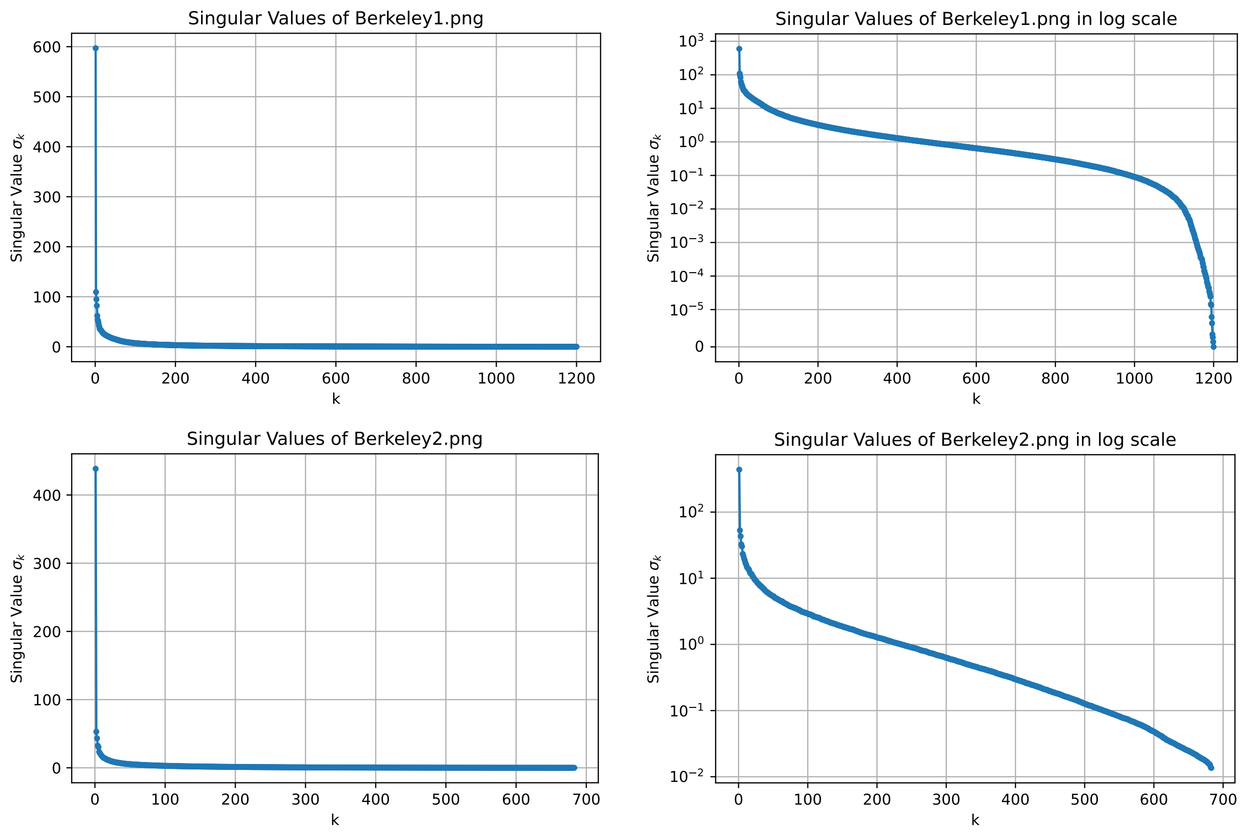
$$\min_{B \in \mathbb{R}^{m \times n}} \|A - B\|_F, \qquad \text{s.t.} \qquad \text{rank}(B) \leq k \tag{9}$$

   and consider the error $e_k = \frac{\|A - B^*\|_F^2}{\|A\|_F^2}$, where B^* is an optimal solution. Solve the above optimization problem in three scenarios of $k = 30$, $k = 80$ and $k = 100$. For each case, report the error e_k and the percentage $\frac{k}{\min(m,n)} \times 100$ (which shows what percentage of the singular values of A is used), and also draw the grayscale image corresponding to B^* (note: you do not need to report the matrix B^* in your homework submission and only the image together with the code is enough).

   iv) Redo Parts (i), (ii), and (iii) for the Berkeley campus picture Berkeley2.png that you can download from bCourses $\rightarrow$ Files $\rightarrow$ HW.

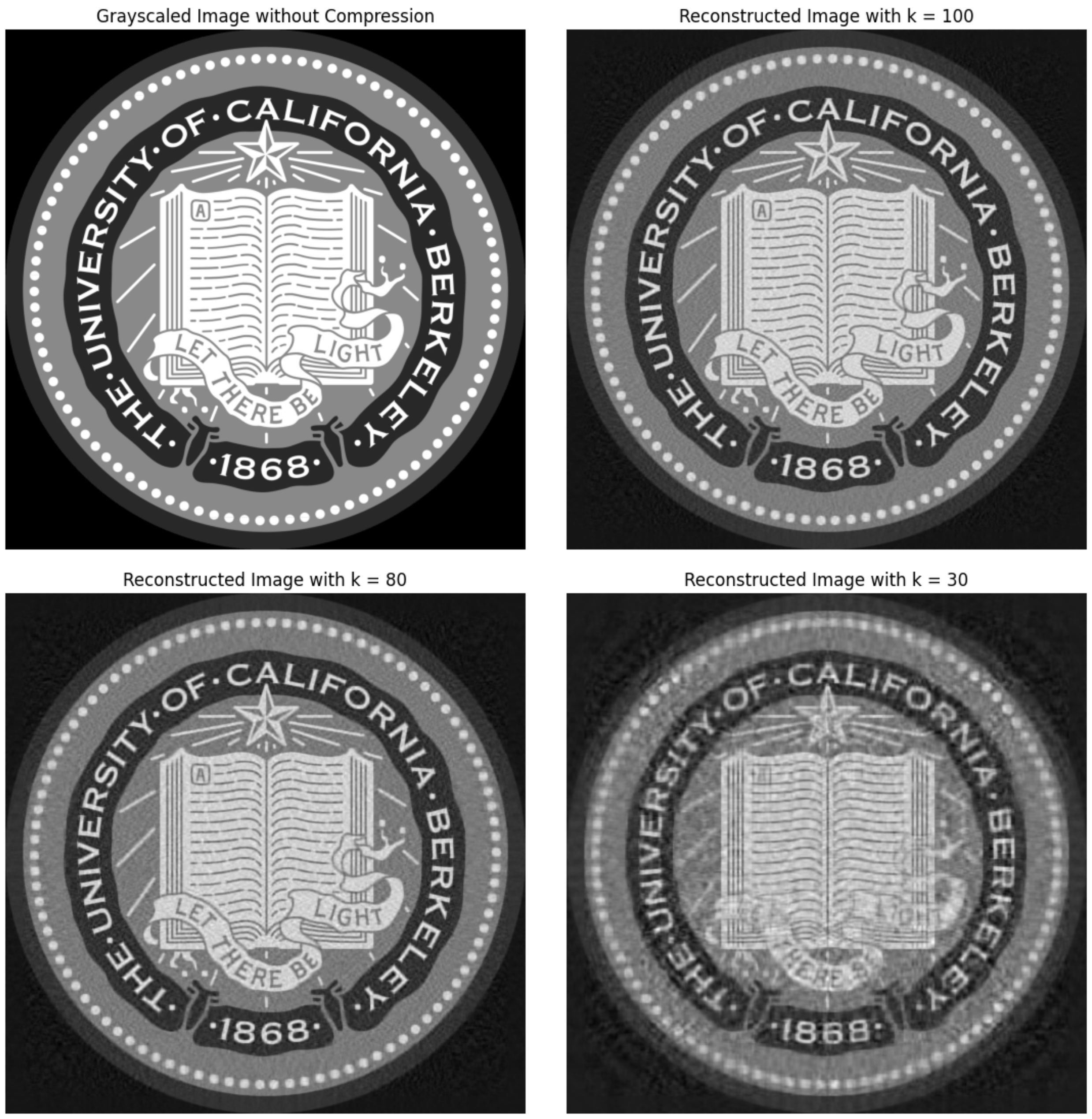
Solution.

For `Berkeley1.png` and `Berkeley2.png`, after converting to grayscale and normalizing pixel values to be within $[0, 1]$, the singular values are plotted as follows:

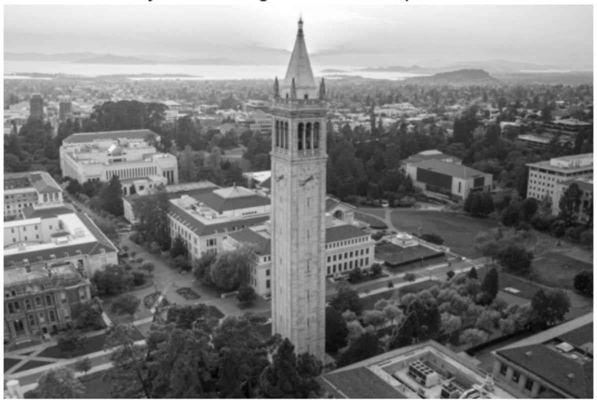


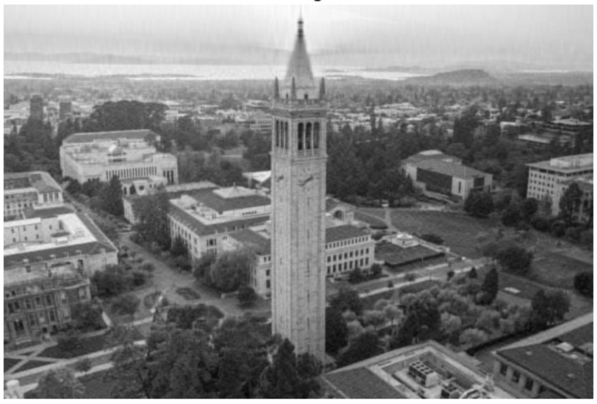
Here, `Berkeley1.png` has dimension 1200×1200, with 1200 singular values. Meanwhile, `Berkeley2.png` has dimension 683×1024, with 683 singular values. Both images have several dominant singular values that are much larger than others, meaning that the prominent information in the image lies in the several largest singular vectors, and removing the rest preserves the main information. By observing the figure in log scale, we see that `Berkeley1.png` has many tiny singular values, meaning that removing those will have minimal effect on image quality. Nonetheless, for this problem we will be only keeping the 30, 80, and 100 largest singular values, and hence some non-trivial singular values will inevitably get removed, causing some quality decrease. We will verify this phenomenon below.

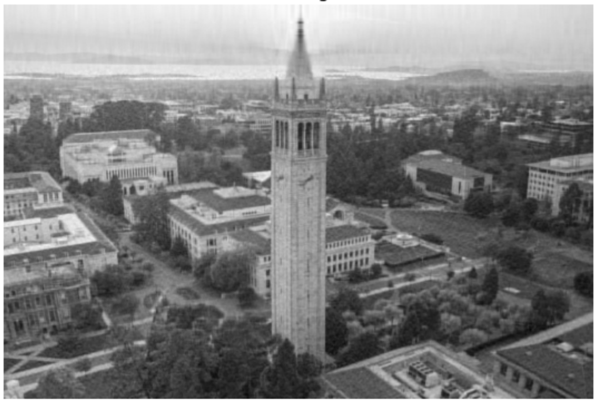
For `Berkeley1.png`, compressing to $k = 30$, $k = 80$, and $k = 100$ corresponds to percentages of retained singular values $\frac{k}{\min(m,n)} \times 100\%$ of 2.50%, 6.67%, and 8.33%. The compressed images are shown below alongside the original grayscale image. We can observe that the smaller k gets, the more "noisy" the image gets and the lower the dynamic range becomes. To quantify the decrease in overall quality, we measure the compression error $e_k = \frac{\|A-B^*\|_F^2}{\|A\|_F^2} \times 100\%$ as 3.53%, 1.09%, and 0.79% for $k = 30$, 80, and 100, respectively. Even though we can see some differences after compression with our eyes, the error is still quite small!

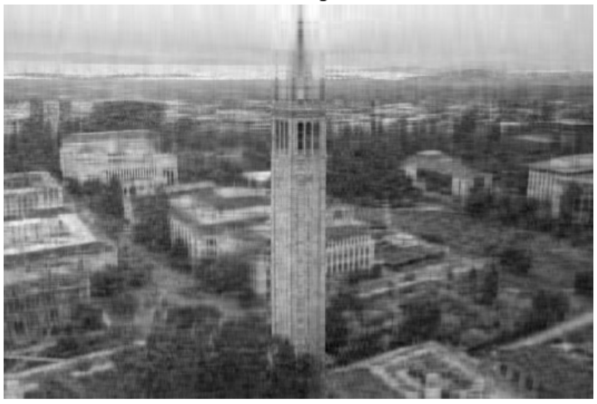

Grayscaled Image without Compression
Reconstructed Image with k = 100
Reconstructed Image with k = 80
Reconstructed Image with k = 30

For `Berkeley2.png`, compressing to $k = 30$, $k = 80$, and $k = 100$ corresponds to percentages of retained singular values $\frac{k}{\min(m,n)} \times 100\%$ of 4.39%, 11.71%, and 14.64%. The compressed images are shown below alongside the original grayscale image. Again, the smaller k gets, the more "noisy" the image gets and the lower the dynamic range becomes. The compression error $e_k = \frac{\|A-B^*\|_F^2}{\|A\|_F^2} \times 100\%$ as 1.02%, 0.35%, and 0.25% for $k = 30$, 80, and 100, respectively. Since the original `Berkeley2.png` is smaller, the same number of singular values corresponds to a smaller relative compression level, and hence a smaller compression error.


Grayscaled Image without Compression


Reconstructed Image with k = 100


Reconstructed Image with k = 80


Reconstructed Image with k = 30

Python Code:

```python
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from tqdm import tqdm

def process_image(image_path, k_values):
    # Part (i): Load the image and convert it to grayscale
    img = Image.open(image_path)
    image_name = image_path.split('.')[0]
    gray_img = img.convert('L')  # Convert to grayscale
    A = np.array(gray_img, dtype=float) / 255.0

    plt.figure(figsize=(5.6, 5.6))
    plt.imshow(A, cmap='gray', aspect='equal')
    plt.title('Grayscaled Image without Compression')
    plt.axis('off')
    plt.tight_layout()
    plt.savefig(f'Grayscale_{image_path}', bbox_inches='tight')
```

```
        # Get the dimensions of the image
        m, n = A.shape
        print(f"Image dimensions: {m} x {n}")

        # Part (ii): Compute SVD and plot the singular values
        U, S, Vt = np.linalg.svd(A, full_matrices=False)
        print(f"Number of singular values: {len(S)}")
        k_range = np.arange(1, len(S) + 1)

        plt.figure(figsize=(5.8, 4))
        plt.plot(k_range, S, '.-')
        plt.title(f'Singular Values of {image_path}')
        plt.xlabel('k')
        plt.ylabel(r'Singular Value $\sigma_k$')
        plt.grid(True)
        plt.tight_layout()
        plt.savefig(f'Singular_Values_{image_name}.pdf', bbox_inches='tight')

        # Part (iii): Solve the optimization problem for given k values
        for k in tqdm(k_values):
            # Truncate the SVD components
            U_k = U[:, :k]
            S_k = np.diag(S[:k])
            Vt_k = Vt[:k, :]

            # Reconstruct the image using rank-k approximation
            B_k = np.dot(U_k, np.dot(S_k, Vt_k))

            # Compute the error e_k
            e_k = ((A - B_k) ** 2).sum() / (A ** 2).sum() * 100

            # Compute the percentage of singular values used
            percentage = (k / len(S)) * 100

            # Display the results
            print(f'\nFor k = {k}:')
            print(f'Error e_k = {e_k:.2f}%')
            print(f'Percentage of singular values used: {percentage:.2f}%\n')

            # Draw the grayscale image corresponding to B_k
            plt.figure(figsize=(5.6, 5.6))
            plt.imshow(B_k, cmap='gray', aspect='equal')
            plt.title(f'Reconstructed Image with k = {k}')
            plt.axis('off')
            plt.tight_layout()
            plt.savefig(f'Reconstructed_Image_{image_name}_k={k}.png', bbox_inches='
        tight')

    # Define the k values to test
    k_values = [30, 80, 100]

    # Process Berkeley1.png
    print('Processing Berkeley1.png\n')
    process_image('Berkeley1.png', k_values)

    # Process Berkeley2.png
    print('Processing Berkeley2.png\n')
    process_image('Berkeley2.png', k_values)
```

MATLAB code:

```
% Define the k values to test
k_values = [30, 80, 100];

% Process Berkeley1.png
fprintf('Processing Berkeley1.png\n');
process_image('Berkeley1.png', k_values);

% Process Berkeley2.png
fprintf('\nProcessing Berkeley2.png\n')
process_image('Berkeley2.png', k_values);

% Local function definition
function process_image(image_path, k_values)
    % Get the image name without extension
    [~, image_name, ~] = fileparts(image_path);

    % Part (i): Load the image and convert it to grayscale
    img = imread(image_path);
    if size(img, 3) == 3  % Check if the image is RGB
        gray_img = rgb2gray(img);
    else
        gray_img = img;  % Image is already grayscale
    end
    A = double(gray_img) / 255.0;  % Normalize pixel values to [0,1]

    % Display and save the grayscale image without compression
    figure('Position', [100, 100, 560, 560]);
    imshow(A);
    title('Grayscaled Image without Compression', 'FontSize', 16);
    axis off;
    axis image;
    % Save the figure
    set(gcf, 'PaperPositionMode', 'auto');
    saveas(gcf, ['Grayscale_', image_name, '.png']);
    close;

    % Get the dimensions of the image
    [m, n] = size(A);
    fprintf('Image dimensions: %d x %d\n', m, n);

    % Part (ii): Compute SVD and plot the singular values
    [U, S, V] = svd(A, 'econ');
    singular_values = diag(S);
    k_range = 1:length(singular_values);
    fprintf('Number of singular values: %d\n', length(singular_values));

    figure('Position', [100, 100, 500, 350]);
    plot(k_range, singular_values, '.-');
    title(['Singular Values of ', image_path], 'FontSize', 16);
    xlabel('k', 'FontSize', 12);
    ylabel('Singular Value \sigma_k', 'FontSize', 12);
    grid on;
    % Adjust layout to tight
    set(gca, 'FontSize', 12);
    set(gca, 'LooseInset', get(gca, 'TightInset'));
    % Save the figure
    set(gcf, 'PaperPositionMode', 'auto');
    saveas(gcf, ['Singular_Values_', image_name, '.eps']);
    close;

    % Part (iii): Solve the optimization problem for given k values
    num_k = length(k_values);
    for idx = 1:num_k
        k = k_values(idx);
        % Truncate the SVD components
        U_k = U(:, 1:k);
        S_k = S(1:k, 1:k);
        V_k = V(:, 1:k);

        % Reconstruct the image using rank-k approximation
        B_k = U_k * S_k * V_k';

        % Compute the error e_k as a percentage
        e_k = (norm(A - B_k, 'fro')^2 / norm(A, 'fro')^2) * 100;

        % Compute the percentage of singular values used
        percentage = (k / length(singular_values)) * 100;

        % Display the results
        fprintf('\nFor k = %d:\n', k);
        fprintf('Error e_k = %.2f%%\n', e_k);
        fprintf('Percentage of singular values used: %.2f%%\n', percentage);

        % Draw the grayscale image corresponding to B_k and save it
        figure('Position', [100, 100, 560, 560]);
        imshow(B_k);
        title(['Reconstructed Image with k = ', num2str(k)], 'FontSize', 16);
        axis off;
        axis image;
        % Save the figure
        set(gcf, 'PaperPositionMode', 'auto');
        saveas(gcf, ['Reconstructed_Image_', image_name, '_k=', num2str(k), '.png'
            ]);
        close;
    end
end
```

4. Consider the least-squares problem

$$\min_x \|Ax - y\| \tag{10}$$

where $A \in \mathbb{R}^{2\times 3}$ and $y \in \mathbb{R}^2$. Assume that

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix}, \qquad y = \begin{bmatrix} 3 \\ 5 \end{bmatrix} \tag{11}$$

i) Show that the set of all solutions is equal to

$$S = \left\{ x^* \in \mathbb{R}^3 \;\middle|\; x_1^* + x_2^* + x_3^* = \frac{13}{5} \right\} \tag{12}$$

ii) Assume that y is perturbed to $y + \Delta y$. Find the set of all solutions of the perturbed least-squares problem $\min_x \|Ax - (y + \Delta y)\|$.

iii) Assume that y is perturbed to $y + \Delta y$, where Δy can take any value in the set $\{\Delta y \mid \|\Delta y\| \leq 1\}$. Find the set of all possibilities for the minimum-norm solution of the perturbed least-squares problem $\min_x \|Ax - (y + \Delta y)\|$ (hint: write the ellipsoidal formula for Δx and compute the semi-axes and lengths of the ellipse; you can use a computer code for SVD calculations).

Solution.

i) To prove the desired statement, we can start with the first-order optimality condition. Noticing that the solutions to $\min_x \|Ax - y\|_2$ and $\min_x \|Ax - y\|_2^2$ are equivalent, the optimality condition becomes

$$\nabla_x \|Ax^* - y\|_2^2 = A^\top(Ax^* - y) = A^\top A x^* - A^\top y = 0 \quad \Longrightarrow \quad A^\top A x^* = A^\top y.$$

Plugging in the numbers, we have

$$\begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \end{bmatrix} = \begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix} \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 13 \\ 13 \\ 13 \end{bmatrix} \tag{13}$$

where x_1^*, x_2^*, and x_3^* denote the three elements of the vector x. Notice that (13) can be reduced to a single equation $5x_1^* + 5x_2^* + 5x_3^* = 13$.

Now that we have proven that all solutions to the problem (10) must satisfy $5x_1^* + 5x_2^* + 5x_3^* = 13$ and therefore reside in S (i.e., necessity), all we need to do is to prove the reverse direction (sufficiency). I.e., we want to prove that all vectors in S are solutions to (10). To do so, we need to check the second-order sufficient optimality condition. Note that the Hessian matrix $\nabla^2 \|Ax - y\|_2^2$ is positive semidefinite (PSD) for all $x \in \mathbb{R}^3$, because it is equal to $2A^\top A$ which is independent from x and known to be PSD as proven in lecture. Hence, the second-order sufficient optimality condition is satisfied, completing the proof for sufficiency.

Putting the above two parts together, we have proven that the set of all solutions is equal to

$$S = \left\{ x^* \in \mathbb{R}^3 \;\middle|\; x_1^* + x_2^* + x_3^* = \frac{13}{5} \right\}.$$

ii) With y perturbed to become $y + \Delta y$, the first-order optimality condition becomes

$$A^\top A x^* - A^\top(y + \Delta y) = 0 \quad \Longrightarrow \quad A^\top A x^* = A^\top y + A^\top \Delta y$$

and the second-order condition does not change. Plugging in the numbers, we have

$$\begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix} \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \end{bmatrix} = \begin{bmatrix} 13 \\ 13 \\ 13 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix},$$

which can be simplified into a single equation, which we can use to define the set of solutions S_Δ:

$$S_\Delta = \left\{ x^* \in \mathbb{R}^3 \;\middle|\; 5x_1^* + 5x_2^* + 5x_3^* = 13 + \Delta y_1 + 2\Delta y_2 \right\}.$$

iii) **SVD method**

From lecture, we know that the minimum-norm solution is defined as $x^* = A^\dagger y$ for the original problem, and is $A^\dagger(y+\Delta y) = x^* + A^\dagger \Delta y$ for the perturbed problem, where $A^\dagger$ is the pseudo-inverse of A. Denote $\Delta x := A^\dagger \Delta y$ as the deviation from the nominal minimum-norm solution x^* due to the perturbation Δy.

With SVD, we find

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{15} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \end{bmatrix}$$

$$\approx \begin{bmatrix} 0.4472 & -0.8944 \\ 0.8944 & 0.4472 \end{bmatrix} \begin{bmatrix} 3.8730 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.5774 & 0.5774 & 0.5774 \\ 0.7071 & -0.7071 & 0 \\ 0.4082 & 0.4082 & -0.8165 \end{bmatrix}.$$

Note that the right two columns of the right singular matrix are not unique. As long as they form an orthogonal matrix along with the first singular vector, they are valid.

Since A is rank-1, we can also use the reduced SVD

$$A = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{15} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \approx \begin{bmatrix} 0.4472 \\ 0.8944 \end{bmatrix} \begin{bmatrix} 3.8730 \end{bmatrix} \begin{bmatrix} 0.5774 & 0.5774 & 0.5774 \end{bmatrix},$$

with which we can obtain

$$A^\dagger = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{15}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \approx \begin{bmatrix} 0.5774 \\ 0.5774 \\ 0.5774 \end{bmatrix} \begin{bmatrix} 0.2582 \end{bmatrix} \begin{bmatrix} 0.4472 & 0.8944 \end{bmatrix}.$$

Hence, we can conclude that

$$x^* = A^\dagger y = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{15}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{15}} \end{bmatrix} \begin{bmatrix} \frac{3+2\times 5}{\sqrt{5}} \end{bmatrix} = \frac{13}{5\sqrt{3}} \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} = \frac{13}{15} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Similarly, we know that

$$\Delta x = A^\dagger \Delta y = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{15}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{15}} \end{bmatrix} \begin{bmatrix} \frac{\Delta y_1 + 2\Delta y_2}{\sqrt{5}} \end{bmatrix} = \frac{\Delta y_1 + 2\Delta y_2}{15} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Notice that

$$\Delta y_1 + 2\Delta y_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}^\top \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} \le \left\| \begin{matrix} 1 \\ 2 \end{matrix} \right\|_2 \cdot \left\| \begin{matrix} \Delta y_1 \\ \Delta y_2 \end{matrix} \right\|_2 \le \sqrt{1^2 + 2^2} \times 1 = \sqrt{5},$$

with equality achieved when $\Delta y_1 = \frac{1}{\sqrt{5}}$ and $\Delta y_2 = \frac{2}{\sqrt{5}}$. Here, the first inequality is the Cauchy-Schwarz inequality, and the second is due to the constraint $\|\Delta y\| \le 1$. Due to symmetry, it also holds that $\Delta y_1 + 2\Delta y_2 \ge -\sqrt{5}$, with equality achieved when $\Delta y_1 = -\frac{1}{\sqrt{5}}$ and $\Delta y_2 = -\frac{2}{\sqrt{5}}$.

Putting the above together, we conclude that the set of perturbed minimum-norm solutions, $\mathcal{X}_{MN}$ is

$$\mathcal{X}_{MN} = \left\{ \frac{13+\epsilon}{15} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \middle| -\sqrt{5} \le \epsilon \le \sqrt{5} \right\}.$$

Non-SVD method

Since we want to find the minimum-norm solution, we effectively want to solve the optimization problem

$$\min_{x_1,x_2,x_3} x_1^2 + x_2^2 + x_3^2 \quad \text{subject to} \quad 5x_1 + 5x_2 + 5x_3 = 13 + \Delta y_1 + 2\Delta y_2 \tag{14}$$

where $x_1^2 + x_2^2 + x_3^2$ is the square of $\|x\|_2$.

Note that we can rearrange our constraint as follows:

$$x_1 = \delta - x_2 - x_3, \quad \text{where } \delta = \frac{13 + \Delta y_1 + 2\Delta y_2}{5}.$$

Plugging this back into (14), we arrive at the unconstrained optimization problem

$$\min_{x_2,x_3} (\delta - x_2 - x_3)^2 + x_2^2 + x_3^2,$$

which is equivalent to

$$\min_{x_2,x_3} \delta^2 + x_2^2 + x_3^2 - 2\delta x_2 - 2\delta x_3 + 2x_2x_3 + x_2^2 + x_3^2.$$

For the purpose of solving for x_2 and x_3, the problem is equivalent to

$$\min_{x_2,x_3} x_2^2 + x_3^2 - \delta x_2 - \delta x_3 + x_2x_3.$$

The Hessian of this objective is $\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \succ 0$ (the eigenvalues are 1 and 3) for all x, meaning that we can find the global minima by simply setting the gradient to zero. Hence, at optimum, it holds that

$$\frac{\partial}{\partial x_2}\left(x_2^2 + x_3^2 - \delta x_2 - \delta x_3 + x_2x_3\right) = 2x_2 + x_3 - \delta = 0,$$
$$\frac{\partial}{\partial x_3}\left(x_2^2 + x_3^2 - \delta x_2 - \delta x_3 + x_2x_3\right) = 2x_3 + x_2 - \delta = 0.$$

By symmetry, we know that the optimizer x^* satisfies $x_2^* = x_3^*$, and therefore $2x_2^* + x_2^* - \delta = 3x_2^* - \delta = 0$, implying that $x_2^* = x_3^* = \frac{\delta}{3}$. Hence, $x_1^* = \delta - x_2^* - x_3^* = \frac{\delta}{3}$, and we have

$$x_1^* = x_2^* = x_3^* = \frac{\delta}{3} = \frac{13 + \Delta y_1 + 2\Delta y_2}{15}$$

Thus, the set we are looking for is

$$\mathcal{X}_{MN} = \left\{ \frac{13 + \Delta y_1 + 2\Delta y_2}{15} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \;\middle|\; \Delta y_1^2 + \Delta y_2^2 \le 1 \right\},$$

which is a line segment on the direction of $(1, 1, 1)$.

The only task left to do is to determine the length of this line segment. Notice that

$$\Delta y_1 + 2\Delta y_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}^\top \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} \le \left\| \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\|_2 \cdot \left\| \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} \right\|_2 \le \sqrt{1^2 + 2^2} \times 1 = \sqrt{5},$$

with equality achieved when $\Delta y_1 = \frac{1}{\sqrt{5}}$ and $\Delta y_2 = \frac{2}{\sqrt{5}}$. Here, the first inequality is the Cauchy-Schwarz inequality, and the second is due to the constraint $\|\Delta y\| \le 1$. Due to symmetry, it also holds that $\Delta y_1 + 2\Delta y_2 \ge -\sqrt{5}$, with equality achieved when $\Delta y_1 = -\frac{1}{\sqrt{5}}$ and $\Delta y_2 = -\frac{2}{\sqrt{5}}$.

As a result, the set we are looking for, $\mathcal{X}_{MN}$, is a line segment between $\frac{13+\sqrt{5}}{15}(1, 1, 1)$ and $\frac{13-\sqrt{5}}{15}(1, 1, 1)$.