

# Homework 3

Math 5424

Numerical Linear Algebra

Alexander Novotny

Zuriah Quinton

October 21, 2023

1. Let  $A$  be  $n \times m$  with  $n > m$ . Show that  $\|A^\top A\|_2 = \|A\|_2^2$  and  $\kappa_2(A^\top A) = \kappa_2(A)^2$ .

Let  $M$  be  $n \times n$  and positive definite and  $L$  be its Cholesky factor so that  $M = LL^\top$ . Show that  $\|M\|_2 = \|L\|_2^2$  and  $\kappa_2(M) = \kappa_2(L)^2$ .

*Proof.*



2. In this question we will ask how to solve  $B\vec{y} = \vec{c}$  given a fast way to solve  $A\vec{x} = \vec{b}$ , where  $A - B$  is “small” in some sense.

- (a) Prove the *Sherman-Morrison formula*: Let  $A$  be nonsingular,  $\vec{u}$  and  $\vec{v}$  be column vectors, and  $A + \vec{u}\vec{v}^\top$  be nonsingular. Then  $(A + \vec{u}\vec{v}^\top)^{-1} = A^{-1} - (A^{-1}\vec{u}\vec{v}^\top A^{-1})/(1 + \vec{v}^\top A^{-1}\vec{u})$ .

More generally, prove the *Sherman-Morrison-Woodbury formula*: Let  $U$  and  $V$  be  $n \times k$  rectangular matrices, where  $k < n$  and  $A$  is  $n \times n$ . Then  $T = I + V^\top A^{-1}U$  is nonsingular if and only if  $A + UV^\top$  is nonsingular, in which case  $(A + UV^\top)^{-1} = A^{-1} - A^{-1}UT^{-1}V^\top A^{-1}$ .

*Proof.* First, we have


$$\begin{aligned} (A + \vec{u}\vec{v}^\top) \left( A^{-1} - \frac{A^{-1}\vec{u}\vec{v}^\top A^{-1}}{1 + \vec{v}^\top A^{-1}\vec{u}} \right) &= \cancel{AA^{-1}} - \frac{\cancel{AA^{-1}}\vec{u}\vec{v}^\top A^{-1}}{1 + \vec{v}^\top A^{-1}\vec{u}} + \vec{u}\vec{v}^\top A^{-1} - \frac{\vec{u}\vec{v}^\top A^{-1}\vec{u}\vec{v}^\top A^{-1}}{1 + \vec{v}^\top A^{-1}\vec{u}} \\ &= I - \frac{\vec{u}\vec{v}^\top A^{-1} - \vec{u}\vec{v}^\top A^{-1}(1 + \vec{v}^\top A^{-1}\vec{u}) + \vec{u}\vec{v}^\top A^{-1}\vec{u}\vec{v}^\top A^{-1}}{1 + \vec{v}^\top A^{-1}\vec{u}} \\ &= I - \frac{\cancel{\vec{u}\vec{v}^\top A^{-1}} - \cancel{\vec{u}\vec{v}^\top A^{-1}} - \vec{u}\vec{v}^\top A^{-1}\vec{v}^\top A^{-1}\vec{u} + \vec{u}\vec{v}^\top A^{-1}\vec{u}\vec{v}^\top A^{-1}}{1 + \vec{v}^\top A^{-1}\vec{u}} \\ &= I - \frac{\vec{v}^\top A^{-1}\vec{u}(-\cancel{\vec{u}\vec{v}^\top A^{-1}} + \cancel{\vec{u}\vec{v}^\top A^{-1}})}{1 + \vec{v}^\top A^{-1}\vec{u}} \\ &= I, \end{aligned}$$

so  $(A + \vec{u}\vec{v}^\top)^{-1} = A^{-1} - (A^{-1}\vec{u}\vec{v}^\top A^{-1})/(1 + \vec{v}^\top A^{-1}\vec{u})$ . Then, assume  $T = I + V^\top A^{-1}U$  is nonsingular. We have

$$\begin{aligned} (A + UV^\top)(A^{-1} - A^{-1}UT^{-1}V^\top A^{-1}) &= \cancel{AA^{-1}} - \cancel{AA^{-1}}UT^{-1}V^\top A^{-1} \\ &\quad + UV^\top A^{-1} - UV^\top A^{-1}UT^{-1}V^\top A^{-1} \\ &= I - U(T^{-1} - I + \underbrace{V^\top A^{-1}U}_{T-I})V^\top A^{-1} \\ &= I - U(\cancel{T^{-1}} - I + \cancel{TT^{-1}} - \cancel{T^{-1}})V^\top A^{-1} \\ &= I, \end{aligned}$$

so  $\mathbf{A} + \mathbf{UV}^\top$  is nonsingular and  $(\mathbf{A} + \mathbf{UV}^\top)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\mathbf{T}^{-1}\mathbf{V}^\top\mathbf{A}^{-1}$ . Finally, assume  $\mathbf{A} + \mathbf{UV}^\top$  is nonsingular and  $(\mathbf{A} + \mathbf{UV}^\top)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{UBV}^\top\mathbf{A}^{-1}$  for some  $\mathbf{B}$ ,  $k \times k$ . Then we have

$$\begin{aligned} \mathbf{I} &= (\mathbf{A} + \mathbf{UV}^\top)(\mathbf{A} + \mathbf{UV}^\top)^{-1} \\ &= (\mathbf{A} + \mathbf{UV}^\top)(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{UBV}^\top\mathbf{A}^{-1}) \\ &= \cancel{\mathbf{AA}^{-1}} - \cancel{\mathbf{AA}^{-1}}\mathbf{UBV}^\top\mathbf{A}^{-1} + \mathbf{UV}^\top\mathbf{A}^{-1} - \mathbf{UV}^\top\mathbf{A}^{-1}\mathbf{UBV}^\top\mathbf{A}^{-1} \\ \Rightarrow 0 &= \mathbf{U}(-\mathbf{B} + \mathbf{I} - \mathbf{V}^\top\mathbf{A}^{-1}\mathbf{UB})\mathbf{V}^\top\mathbf{A}^{-1} \\ \Rightarrow 0 &= \mathbf{I} - \underbrace{(\mathbf{I} + \mathbf{V}^\top\mathbf{A}^{-1}\mathbf{U})}_{\mathbf{T}}\mathbf{B} \\ \Rightarrow \mathbf{I} &= \mathbf{TB}, \end{aligned}$$

therefore  $\mathbf{T}$  is invertible and non-singular, and  $\mathbf{B} = \mathbf{T}^{-1}$ . 

- (b) If you have a fast algorithm to solve  $\mathbf{A}\vec{x} = \vec{b}$ , show how to build a fast solver for  $\mathbf{B}\vec{y} = \vec{c}$ , where  $\mathbf{B} = \mathbf{A} + \vec{u}\vec{v}^\top$ .

**Answer.** We have

$$\begin{aligned} \vec{y} &= \mathbf{B}^{-1}\vec{c} \\ &= (\mathbf{A} + \vec{u}\vec{v}^\top)^{-1}\vec{c} \\ &\stackrel{2a}{=} (\mathbf{A}^{-1} - (\mathbf{A}^{-1}\vec{u}\vec{v}^\top\mathbf{A}^{-1})/(1 + \vec{v}^\top\mathbf{A}^{-1}\vec{u}))\vec{c} \\ &= \mathbf{A}^{-1}\vec{c} - (\mathbf{A}^{-1}\vec{u})\vec{v}^\top(\mathbf{A}^{-1}\vec{c})/(1 + \vec{v}^\top(\mathbf{A}^{-1}\vec{u})). \end{aligned}$$

Then, an algorithm to quickly solve this problem can be found in algorithm 1.

---

**Algorithm 1:** An algorithm to quickly solve  $(\mathbf{A} + \vec{u}\vec{v}^\top)\vec{y} = \vec{c}$  given an algorithm to quickly solve  $\mathbf{A}\vec{x} = \vec{b}$ .

---

**Data:**  $\mathbf{A}$ ,  $\vec{c}$ ,  $\vec{u}$ ,  $\vec{v}$

**Result:**  $\vec{y} = (\mathbf{A} + \vec{u}\vec{v}^\top)^{-1}\vec{c}$

- 1 Solve  $\mathbf{A}\vec{w} = \vec{c}$ ;
  - 2 Solve  $\mathbf{A}\vec{x} = \vec{u}$ ;
  - 3  $\vec{y} \leftarrow \vec{w} - \vec{x}\vec{v}^\top\vec{w}/(1 + \vec{v}^\top\vec{x})$ ;
- 

3. Consider the linear system  $\mathbf{A}\vec{x} = \vec{b}$  where

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} \quad \text{and} \quad \vec{b} = \begin{bmatrix} 4 \\ 11 \\ 29 \\ 30 \end{bmatrix}.$$

- (a) Calculate the appropriate determinants to show that  $\mathbf{A}$  can be written as  $\mathbf{A} = \mathbf{LU}$  where  $\mathbf{L}$  is a unit lower triangular matrix and  $\mathbf{U}$  is a non-singular upper triangular matrix. You may use MATLAB to compute the determinants. But the remaining parts of this problem should be done by paper-and-pencil.

**Answer.**

- (b) Compute the LU decomposition  $\mathbf{A} = \mathbf{LU}$  and convert  $\mathbf{A}\mathbf{x} = \mathbf{b}$  into  $\mathbf{U}\mathbf{x} = \mathbf{y}$  where  $\mathbf{U}$  is upper triangular and solve for  $\mathbf{x}$ .

**Answer.** The LU decomposition is please

- (c) Using  $\mathbf{L}$  and  $\mathbf{U}$  from the earlier steps, solve the new system  $\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$  where

$$\tilde{\mathbf{b}} = \begin{bmatrix} 5 \\ 15 \\ 41 \\ 45 \end{bmatrix}.$$

Do NOT perform the Gaussian elimination from scratch. You already have  $\mathbf{U}$  and  $\mathbf{L}$ .

**Answer.**

4. Let  $\mathbf{A}$  have the following economy (thin) SVD:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \\ 0 & -0.8 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 0.6 & -0.8 \\ -0.8 & -0.6 \end{bmatrix}^\top.$$

Answer the following questions without forming  $\mathbf{A}$  and without using Matlab. None of these questions require using numerical software. At most, you need to be able to perform simple matrix arithmetic using paper-and-pencil.

(a) Find  $\text{rank}(\mathbf{A})$ ,  $\|\mathbf{A}\|_2$  and  $\|\mathbf{A}\|_F$ .

**Answer.** We have

$$\begin{aligned} \text{rank}(\mathbf{A}) &= 2, \\ \|\mathbf{A}\|_2 &= \max_i \sqrt{\lambda_i(\mathbf{A}^\top \mathbf{A})} \\ &= \sqrt{\sigma_1^2(\mathbf{A})} \\ &= |\sigma_1(\mathbf{A})| = 4, \\ \|\mathbf{A}\|_F &= \sqrt{\text{tr}((\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top)^\top (\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top))} \\ &= \sqrt{\text{tr}(\mathbf{V}^\top \mathbf{V} \boldsymbol{\Sigma}^\top \boldsymbol{\Sigma} \mathbf{U}^\top \mathbf{U} \boldsymbol{\Sigma})} \\ &= \sqrt{\text{tr}^2(\boldsymbol{\Sigma})} \\ &= |\text{tr}(\boldsymbol{\Sigma})| \\ &= 4 + 3 = 7. \end{aligned}$$

(b) Find an orthonormal basis for  $\text{Range}(\mathbf{A})$ ,  $\text{Null}(\mathbf{A})$ ,  $\text{Range}(\mathbf{A}^\top)$ , and  $\text{Null}(\mathbf{A}^\top)$ .

**Answer.** We have

$$\begin{aligned} \text{Range}(\mathbf{A}) &= \text{span} \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.6 \\ -0.8 \end{bmatrix} \right\}, \\ \text{Null}(\mathbf{A}) &= \text{span } \emptyset, \\ \text{Range}(\mathbf{A}^\top) &= \text{span} \left\{ \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix}, \begin{bmatrix} -0.8 \\ -0.6 \end{bmatrix} \right\}, \\ \text{Null}(\mathbf{A}^\top) &= \mathbb{R}^3 \setminus \text{span} \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.6 \\ -0.8 \end{bmatrix} \right\} \\ &= \text{span} \left\{ \begin{bmatrix} 0 \\ 0.8 \\ 0.6 \end{bmatrix} \right\}. \end{aligned}$$

Note that the given sets consist of orthogonal unit vectors, so they are orthonormal bases for the sets which they span.

(c) Form the optimal rank-1 approximation  $\mathbf{A}_1$  to  $\mathbf{A}$  in the 2-norm? What is the error matrix  $\mathbf{A} - \mathbf{A}_1$ , and what is the norm of the error  $\|\mathbf{A} - \mathbf{A}_1\|_2$ ? The error matrix and the norm of the error follow directly from the SVD; no computation is needed.

**Answer.** We have

$$\begin{aligned}
 \mathbf{A}_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \\ 0 & -0.8 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.6 & -0.8 \\ -0.8 & -0.6 \end{bmatrix}^\top \\
 &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [4] \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix}^\top, \\
 \mathbf{A} - \mathbf{A}_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \\ 0 & -0.8 \end{bmatrix} \left( \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix} - \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 0.6 & -0.8 \\ -0.8 & -0.6 \end{bmatrix}^\top \\
 &= \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \\ 0 & -0.8 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 0.6 & -0.8 \\ -0.8 & -0.6 \end{bmatrix}^\top, \\
 \|\mathbf{A} - \mathbf{A}_1\|_2 &= 3.
 \end{aligned}$$

5. (From Golub and van Loan) Given  $\mathbf{A} \in \mathbb{R}^{m \times n}$ , let  $\sigma_1$  denote the largest singular value of  $\mathbf{A}$ . Prove that

$$\sigma_1(\mathbf{A}) = \max_{\substack{\vec{y} \in \mathbb{R}^m \\ \vec{x} \in \mathbb{R}^n}} \frac{\vec{y}^\top \mathbf{A} \vec{x}}{\|\vec{y}\|_2 \|\vec{x}\|_2}.$$

*Proof.*



6. Let  $\mathbf{A} \in \mathbb{R}^{m \times n}$  have the singular value decomposition

$$\mathbf{A} = \sum_{j=1}^r \sigma_j \vec{u}_j \vec{v}_j^\top.$$

We showed in class that the matrix

$$\mathbf{A}_k = \sum_{j=1}^k \sigma_j \vec{u}_j \vec{v}_j^\top$$

is an optimal rank- $k$  approximation to  $\mathbf{A}$  in the 2-norm and  $\|\mathbf{A} - \mathbf{A}_k\| = \sigma_{k+1}$ . But this minimizer (optimal approximant) is not unique. In other words, using truncated SVD we can find another rank- $k$  matrix  $\tilde{\mathbf{A}}_k \in \mathbb{R}^{m \times n}$  such that  $\|\mathbf{A} - \tilde{\mathbf{A}}_k\| = \sigma_{k+1}$ . In this problem, you will prove this fact.

Define

$$\tilde{\mathbf{A}}_k = \sum_{j=1}^k (\sigma_j - \eta_j) \vec{u}_j \vec{v}_j^\top \quad \text{where} \quad 0 \leq |\eta_j| < \sigma_{k+1}. \quad (1)$$


Show that  $\tilde{\mathbf{A}}_k$  in (1) has rank- $k$  and minimizes the error, i.e.,  $\|\mathbf{A} - \tilde{\mathbf{A}}_k\| = \sigma_{k+1}$ .

*Proof.* Observe that  $\sum_{j=1}^k (\sigma_j - \eta_j) \vec{u}_j \vec{v}_j^\top$  is the SVD of  $\tilde{\mathbf{A}}_k$ , with singular values of  $(\sigma_j - \eta_j)$ . Since  $|\eta_j| < \sigma_{k+1} \leq \sigma_j$  for all  $1 \leq j \leq k$ , then  $\sigma_j - \eta_j > 0$ , and  $\tilde{\mathbf{A}}_k$  has  $k$  non-zero singular values, so it has rank  $k$ . As well, we have

$$\begin{aligned}
 \mathbf{A} - \tilde{\mathbf{A}}_k &= \sum_{j=1}^r \sigma_j \vec{u}_j \vec{v}_j^\top - \sum_{j=1}^k (\sigma_j - \eta_j) \vec{u}_j \vec{v}_j^\top \\
 &= \sum_{j=1}^k [\sigma_j \vec{u}_j \vec{v}_j^\top - (\sigma_j - \eta_j) \vec{u}_j \vec{v}_j^\top] + \sum_{j=k+1}^r \sigma_j \vec{u}_j \vec{v}_j^\top \\
 &= \sum_{j=1}^k \eta_j \vec{u}_j \vec{v}_j^\top + \sum_{j=k+1}^r \sigma_j \vec{u}_j \vec{v}_j^\top.
 \end{aligned}$$

Note, then, that this is the SVD of  $\mathbf{A} - \tilde{\mathbf{A}}_k$ , with eigenvalues of  $|\eta_j|$  for  $1 \leq j \leq k$  and  $\sigma_j$  for  $k+1 \leq j \leq r$ . Since  $|\eta_j| < \sigma_{k+1}$  for all  $j$  and  $\sigma_{k+1} \geq \sigma_k$  for all  $j$ , we must have

$$\|\mathbf{A} - \tilde{\mathbf{A}}_k\|_2 = \sigma_{k+1},$$

as required. 

7. Approximation of a Fingerprint Image via SVD: You will complete the Matlab Script `Homework3Problem7.m` to answer this question. You will attach the completed script to the .pdf file.

Before start completing your code, it will help use the commands `help svd`, `help subplot`, `help semilogy` to understand how to use these functions.

Lines 7-10 load the image into Matlab and plot it. The resulting matrix  $\mathbf{A}$  in your workspace is a  $1133 \times 784$  matrix with entries consisting of only ones and zeroes; ones correspond to the white spots in the image and zeroes to the black spots.

- (a) Compute the short (reduced) SVD of  $\mathbf{A}$  using Matlab. Then plot the normalized singular values under `subplot(2,1,2)`. This figure might be hard to read due to the scale of the  $y$ -axis. Then plot only the leading 700 normalized singular values under `subplot(2,1,2)`. What do you observe? What would be your decision for the (numerical) rank of  $\mathbf{A}$  based on these comments? Justify your reasoning. Does the result of the built-in rank command coincide with your decision?

**Answer.** Plots of the singular values of  $\mathbf{A}$  can be found in fig. 1. We notice a significant drop off in singular values after the 654<sup>th</sup> singular value. Because of this, the numerical rank of  $\mathbf{A}$  should be 654. Indeed, this is the computed rank of  $\mathbf{A}$ .

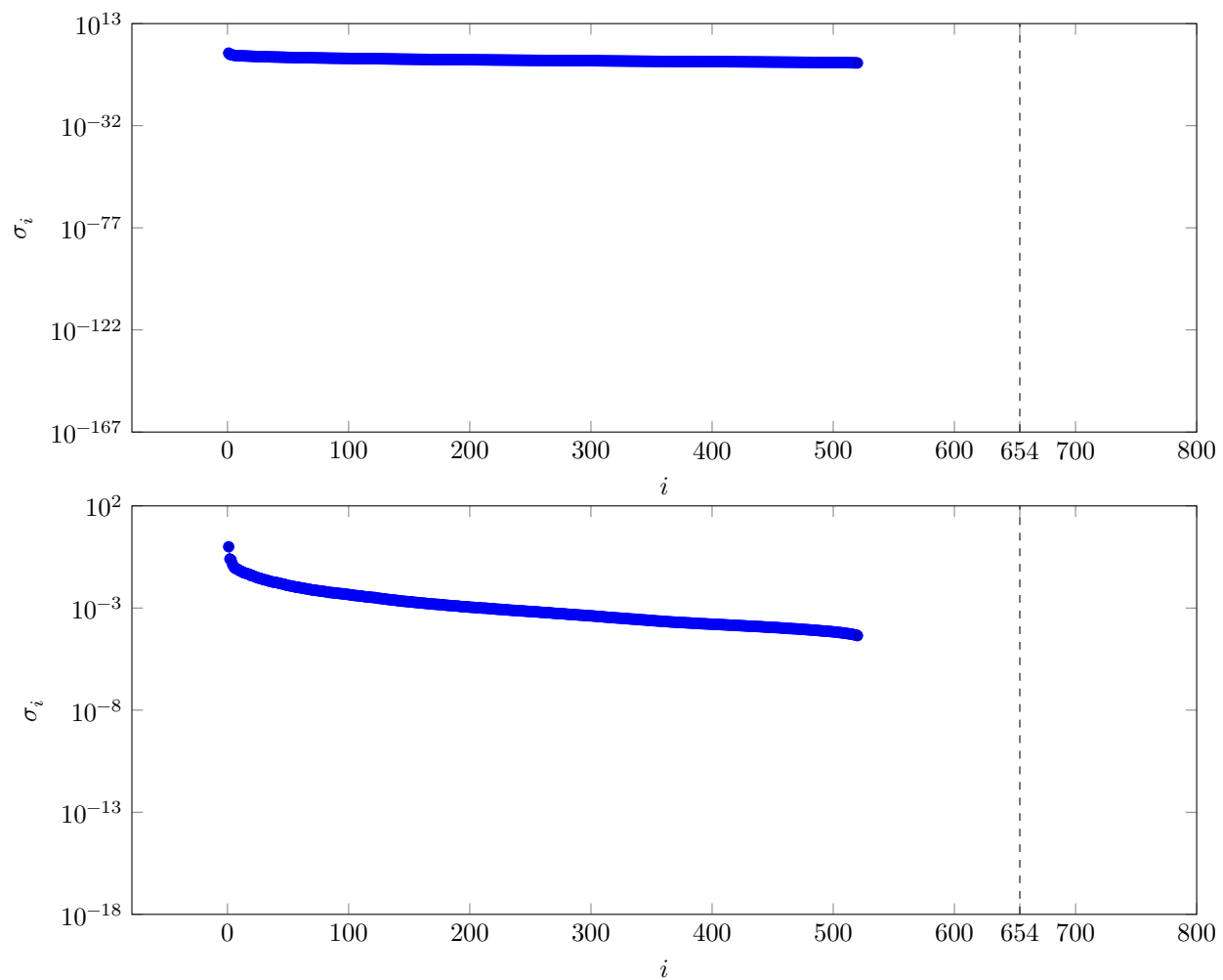


Figure 1: Plots showing all singular values of  $\mathbf{A}$  and just the first 700 singular values of  $\mathbf{A}$ . Note the large gap between values that occurs at the calculated rank of  $\mathbf{A}$  (654).

- (b) Construct the optimal rank- $k$  approximation  $\mathbf{A}_k$  to  $\mathbf{A}$  in the 2-norm for  $k = 1, k = 10$ , and  $k = 50$ . In each case, compute the relative error  $\frac{\|\mathbf{A} - \mathbf{A}_k\|_2}{\|\mathbf{A}\|_2}$ . Note that you do NOT need to form  $\mathbf{A} - \mathbf{A}_k$  to compute these error values; the singular values are all you need. Use `subplot(2,2,1)` to plot the original image in the top left corner. Then, use the `imshow` command on the three low-rank approximations and plot them in the `subplot(2,2,2)`, `subplot(2,2,3)`, and `subplot(2,2,4)` spots. Put appropriate titles on each plot, e.g., “original image”, “rank-1 approximation” etc., using the `title` command. Plots without appropriate labels and explanations will lose points. These plots need to be attached to the .pdf file.

**Answer.** The plots of the rank- $k$  approximations, for  $k = 1, 10, 50$ , can be found in fig. 2, with program output displayed below:

```
Computed Rank of A = 654
Relative error for A_1 = 0.12761236428030379
Relative error for A_10 = 0.08040029616965558
Relative error for A_50 = 0.03400080504768164
```

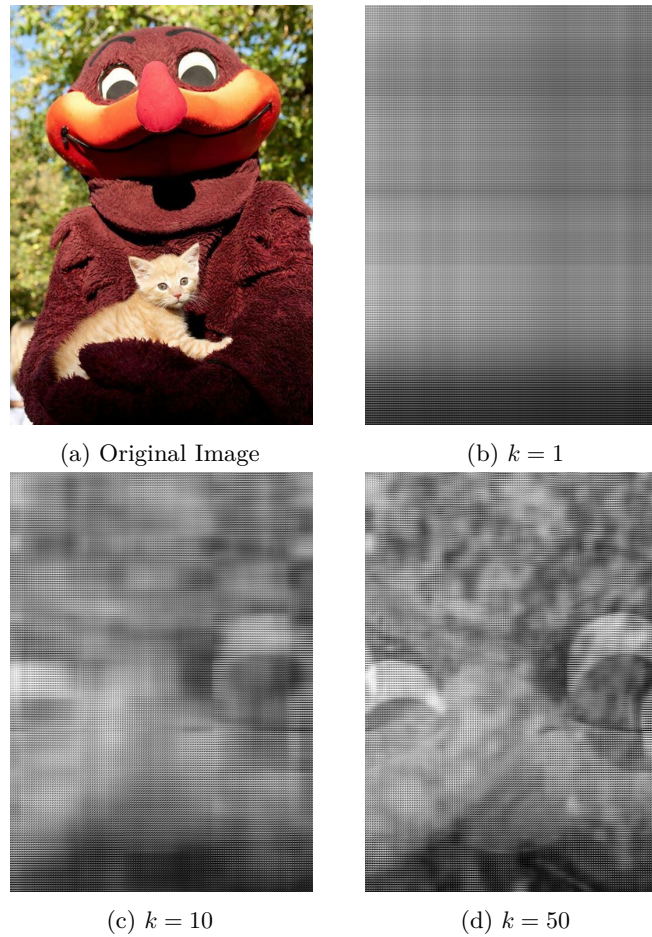


Figure 2: Original fingerprint image compared to its optimal rank- $k$  approximations.

The code is given below:

```

1  #![feature(float_next_up_down)]
2  use image::{DynamicImage::ImageLuma8, GrayImage};
3  use nalgebra::{
4      allocator::Allocator, ComplexField, DMatrix, DefaultAllocator, Dim, DimMin, DimMinimum, Matrix,
5      RawStorage, Storage, SVD,
6  };
7  use num_traits::AsPrimitive;
8  use show_image::create_window;
9  use std::{
10     fs::{self, File},
11     io::Write,
12 };
13
14 #[show_image::main]
15 fn main() {
16     // Open original .pbm image
17     let img = image::open("HW4_Prob6_fingerprint.pbm").unwrap();
18     // Display image
19     let window = create_window("Original Image", Default::default()).unwrap();
20     window.set_image("Figure 1", img.clone()).unwrap();
21
22     // Assert image is 8-bit monochrome
23     let ImageLuma8(img) = img else { unreachable!() };
24     // Convert image to matrix of doubles
25     let a = DMatrix::from_row_iterator(
26         img.height() as usize,
27         img.width() as usize,
28         img.as_raw().iter().map(|x| *x as f64),

```

```

29     );
30
31     // Save size of A for later
32     let (m, n) = a.shape();
33
34     // check matrix is still the same image
35     mat_to_img_show(&a, "Original matrix as image check");
36
37     // Compute SVD of A
38     let svd = SVD::new(a, true, true);
39
40     // Create output file
41     fs::create_dir_all("./out").unwrap();
42     let mut out = File::create("./out/all_singular_values7.dat").unwrap();
43     // Print normalized singular values to output file
44     writeln!(out, "#all singular values of A").unwrap();
45     for s in &svd.singular_values {
46         writeln!(out, "{}", s / svd.singular_values[0]).unwrap();
47     }
48     // notify Terminal of completed SVD printing
49     eprintln!("finished printing");
50
51     // The rank() function in matlab uses a default tolerance value. The nalgebra version doesn't - it needs
52     // ↪ to be provided. So we replicate matlab's default tolerance.
53     // tol from MatLab = max(size(A))*eps(norm(A))
54     // From matlab's documentation, eps(x) is the positive distance between |x| and the next largest float.
55     // Rust's f64::next_up(x) returns that next largest float. As well, we are using the already computed
56     // ↪ largest singular value as norm(A).
57     let tol = (m.max(n) as f64) * (f64::next_up(svd.singular_values[0]) - svd.singular_values[0]);
58
59     // Compute the Rank of A with the computed tolerance
60     println!("Computed Rank of A = {}", svd.rank(tol));
61
62     // Compute optimal rank-k approximations, display them, and save them
63     let mut approx = DMatrix::zeros(m, n);
64     let mut prev_k = 0;
65     for k in [1, 10, 50] {
66         approx = rank_k_approx(&svd, k, &approx, prev_k);
67         prev_k = k;
68         mat_to_img_show(&approx, format!("Rank_{k}_Approximation"));
69         // Print relative errors
70         println!(
71             "Relative error for A_{k} = {}",
72             // Rust is 0 indexed, need to subtract 1
73             svd.singular_values[(k + 1) - 1] / svd.singular_values[1 - 1]
74         );
75     }
76
77     // keep images up until original window is closed
78     for _event in window.event_channel().unwrap() {}
79 }
80
81 /// Calculates the optimal rank `k` approximation of a matrix represented by its singular value decomposition.
82 /// Uses previously-computed optimal approximation `prev`, which is rank `prev_k`, which must be less than
83 /// ↪ `k`.
84 /// For new approximation, pass zero matrix for `prev` and 0 for `prev_k`
85 fn rank_k_approx<T: ComplexField, R: DimMin<C>, C: Dim>(&SVD<T, R, C>,
86     k: usize,
87     prev: &Matrix<T, R, C, impl Storage<T, R, C>>,
88     prev_k: usize,
89 ) -> Matrix<T, R, C, <nalgebra::DefaultAllocator as nalgebra::allocator::Allocator<T, R, C>>::Buffer>
90 where
91     DefaultAllocator: Allocator<T, DimMinimum<R, C>, C>
92     + Allocator<T, R, DimMinimum<R, C>>
93     + Allocator<T::RealField, DimMinimum<R, C>>
94     + Allocator<T, R, C>,
95 {
96     let mut u = svd.u.as_ref().unwrap().clone();
97     // Multiply  $\sigma_i u_i$  for  $i \in [1, k]$ 

```



```

96     for i in prev_k..k {
97         let val = svd.singular_values[i].clone();
98         u.column_mut(i).scale_mut(val);
99     }
100     // Multiply  $\sum_{i=1}^k (\sigma_i u_i) v_i^T$ 
101     prev + u.columns(prev_k, k - prev_k) * svd.v_t.as_ref().unwrap().rows(prev_k, k - prev_k)
102 }
103
104 /// Displays and saves the image stored in mat to the file "./out/<wind_name>.png"
105 fn mat_to_img_show<T: AsPrimitive<u8>, R: Dim, C: Dim, S: RawStorage<T, R, C>>(<
106     mat: &Matrix<T, R, C, S>,
107     wind_name: impl AsRef<str>,
108 ) {
109     // Convert matrix to 8-bit monochrome image
110     // Annoyingly, image crate and matrix crate use different size types, so converting is required
111     let im2 = GrayImage::from_fn(mat.ncols() as u32, mat.nrows() as u32, |c, r| {
112         image::Luma([mat[(r as usize, c as usize)].as_()])
113     });
114
115     // Create window and display image
116     let window2 = create_window(wind_name.as_ref(), Default::default()).unwrap();
117     window2.set_image("f", im2.clone()).unwrap();
118
119     // Save image as png in output folder
120     im2.save_with_format(
121         format!("./out/{}.png", wind_name.as_ref()),
122         image::ImageFormat::Png,
123     )
124     .unwrap();
125 }

```

8. In the previous problem you used SVD to compress a black-and-white image. In this example, you will use SVD to compute a low-rank approximation to the color image hokiebirdwithacat.jpg. You will complete the Matlab Script Homework3Problem8.m to answer this question. You will attach the completed script to the .pdf file.

Lines 7-20 load the image into Matlab, plot the original image, and extract the images correspond to every color layer, and convert these three layer images to double precision matrices A1, A2, and A3.

- (a) Compute the SVDs of every layer A1, A2, and A3. Then compute the vector of normalized singular values for every layer. Using the subplot comment (and logarithmic y-axis), plot all three vectors in Figure 2.

**Answer.**

- (b) Use the same error tolerance for every layer: Find the smallest rank-k for each layer such that the relative error in each layer is less than

- (i) 10%                      (ii) 5%                      (iii) 1%

So, you will have three approximations. For each case, plot the low-rank image (either as a new figure or all in one plot using subplot). These plots need to be attached to the .pdf file. Make sure to clearly indicate the rank of every layer and which plot corresponds to which error tolerance. Did you obtain the same k value for every layer? Comment on your results.

**Answer.**

- (c) Use different error tolerances for different layers: Now we will choose different tolerances for different values. Pick three different selections:

- 50% error for Layer 1, 1% error for Layer 2, and 1% error for Layer 3,
- 1% error for Layer 1, 50% error for Layer 2, and 1% error for Layer 3,
- 1% error for Layer 1, 1% error for Layer 2, and 50% error for Layer 3,

So, you will have three approximations. For each case, plot the low-rank image (either as a new figure or all in one plot using subplot). These plots need to be attached to the .pdf file. Make sure to clearly indicate the rank of every layer and which plot corresponds to which error tolerance. Comment on your results.

**Answer.**