CENG 371 - Scientific Computing
2023-1
Homework 3

Anıl Eren Göçer
e2448397@ceng.metu.edu.tr

December 30, 2023

Question 1

a) The dimensionality of the solution space for a homogeneous system $Ax = 0$ (where 0 is the zero vector) is given by the nullity of A. The nullity is the dimension of the null space (also known as the kernel) of A.

In this case, since rank(A) = n and n is the number of columns, the nullity of A is m - rank(A). This means the null space of A has dimension m - n.

Now, let's consider the set B. Each vector b' in B corresponds to a solution of the homogeneous system Ax = 0, and the dimensionality of B is equal to the nullity of A. There dim(B) = m - rank(A) = m - n.

b) My code is in q1.ipynb file under Q1 folder.

In order to calculate a basis for B, we need to calculate the null space of A. I did this in my code and the results were consistent with the part a). As you can see in my code, there is no basis vector printed in the code and this means that dim(null(A)) = 0. Note that, the matrix A has rank 5 since all rows are linearly independent. This results in the fact that set B has dimensionality 5 - rank(A) = 5 - 5 = 0. So, results I found in part a) and part b) are consistent.

Question 2

My code is in the q2.m file under Q2 folder of my submittion and other related materials (images etc.) can be found under Q2 folder.

a)

Figure 1: I (Original Image)

Figure 2: $I_{r/2}$

Figure 3: I_{r-10}

My image is of the size 1024 X 682. So, $r = min(1024, 682) = 682$, $r/2 = 341$ and $r - 10 = 672$.

The original image (I) represents the full-resolution grayscale. In contrast, I_{r-10}, discards the smallest 10 singular values which has a bit degradation, resulting in a compressed representation that captures essential visual features with reduced detail. Similarly, $I_{r/2}$ further emphasizes compression by retaining fewer singular values, leading to a more significant loss of fine details but providing an even more compact representation.

The difference between them may not be so clear to you, so I investigated $I_{r/4}$, $I_{r/8}$, $I_{r/16}$, $I_{r/32}$, and $I_{r/64}$ since the difference between them is sharper.

Figure 4: $I_{r/4}$

Figure 5: $I_{r/8}$

Figure 6: $I_{r/16}$

Figure 7: $I_{r/32}$

Figure 8: $I_{r/64}$

As you can see in the Figures 4-9, the higher rank it is, the more high resolution the approximation preserve. That is, the images with higher rank will include more details and the images with lower rank will be more compact.

b)

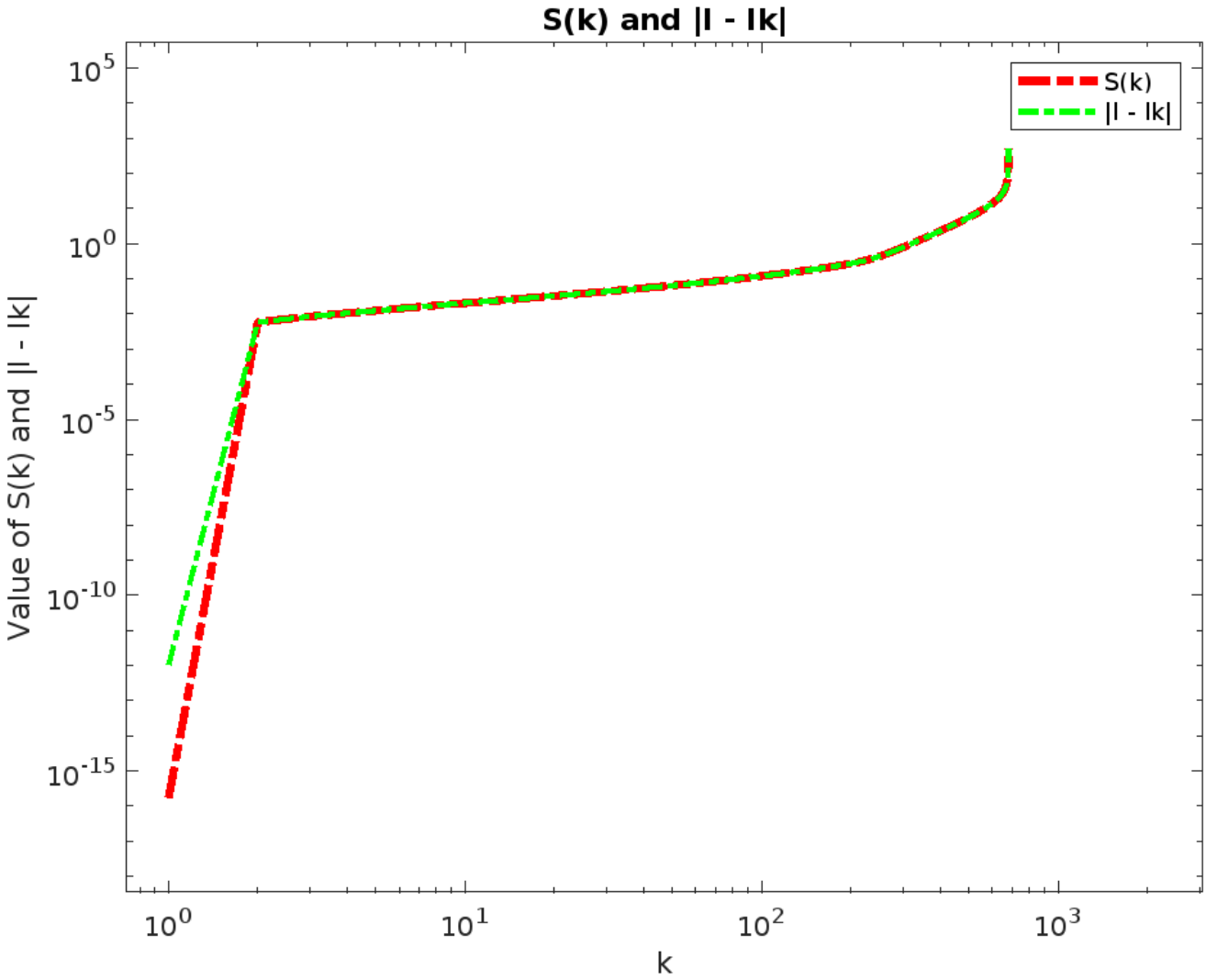

Figure 9: k vs S(k) and k vs Frobenius Norm Errors

I observed that both $S(k)$ and $|I - I_k|_F$ behaves too similarly as k changes. They are even completely equal after a certain point, which is about k = 30 for my image. The underlying reason this observation is that we ignore some of the singular values during calculation of low rank approximations and they are the ones which is summed while calculating $S(k)$ corresponding to approximation I_k. Therefore, $S(k)$ and $|I - I_k|_F$ becomes equal.

c) I think a use case for low-rank approximation can be data compression, especially image compression. For example, imagine that you are doing a video call in an environment in which network band-limit is limited. Since a video is a collection of images, we can send low-rank approximation of original frames. As you can see in part a), half-rank approximation does not result in a too noticeable degradation, so we can save band-limit by sending low-rank approximation of original frames without too much degradation.

Question 3

My code is in the q3.ipynb file under Q3 folder of my submittion.

a) The code regarding this part is in my q3.ipynb file in the section where I specified as "part a)" .
I created sets of observations:

$$O_n = \left\{ (t, N_{obs}(t)) | t \in \left\{ \frac{1}{n+1}, \frac{2}{n+1}, ..., \frac{n-1}{n+1}, \frac{n}{n+1} \right\} \right\}$$

I calculate $N_{obs}(t)$ values by creating a different epsilon for each coefficient in N. That is, for a value of t, $N_{obs}(t)$ is calculated as follows (assuming that actual coefficients are known):

$$N_{obs}(t) = 0.3(t+\epsilon_1)^0 + 2(t+\epsilon_2)^1 - 1.2(t+\epsilon_3)^2 + 0.5(t+\epsilon_4)^3$$

Then, I constructed A_n and b_n as follows:

Let $(t_i, N_{obs}(t_i))$ represents an observartion. For n observation:

$$A_n = \begin{bmatrix} (t_1+\epsilon_{11})^0 & (t_1+\epsilon_{12})^1 & (t_1+\epsilon_{13})^2 & (t_1+\epsilon_{14})^3 \\ (t_2+\epsilon_{21})^0 & (t_2+\epsilon_{22})^1 & (t_2+\epsilon_{23})^2 & (t_2+\epsilon_{24})^3 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ (t_n+\epsilon_{n1})^0 & (t_n+\epsilon_{n2})^1 & (t_n+\epsilon_{n3})^2 & (t_n+\epsilon_{n4})^3 \end{bmatrix} = \begin{bmatrix} 1 & (t_1+\epsilon_{12})^1 & (t_1+\epsilon_{13})^2 & (t_1+\epsilon_{14})^3 \\ 1 & (t_2+\epsilon_{22})^1 & (t_2+\epsilon_{23})^2 & (t_2+\epsilon_{24})^3 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 1 & (t_n+\epsilon_{n2})^1 & (t_n+\epsilon_{n3})^2 & (t_n+\epsilon_{n4})^3 \end{bmatrix}$$

$$b_n = \begin{bmatrix} N_{obs}(t_1) \\ N_{obs}(t_2) \\ . \\ . \\ . \\ N_{obs}(t_n) \end{bmatrix}$$

And these A_n and b_n satisfies the following relation with the coefficients vector, c, of N.

$$A_n c_k^{(n)} = b_n$$

b) The related code is in q3.ipynb file under Q3 folder of my submittion. I marked it as "part b".

I approximated the actual values of c_k using **Augmented System** method as follows: I know the followings:

- $A_n \in R^{nx4}$
- $A_n^T \in R^{4xn}$
- $b_n \in R^n$
- $r \in R^n$
- $c \in R^4$
- $I \in R^{nxn}$
- $0 \in R^{4x4}$

As a result, I obtained the following system.

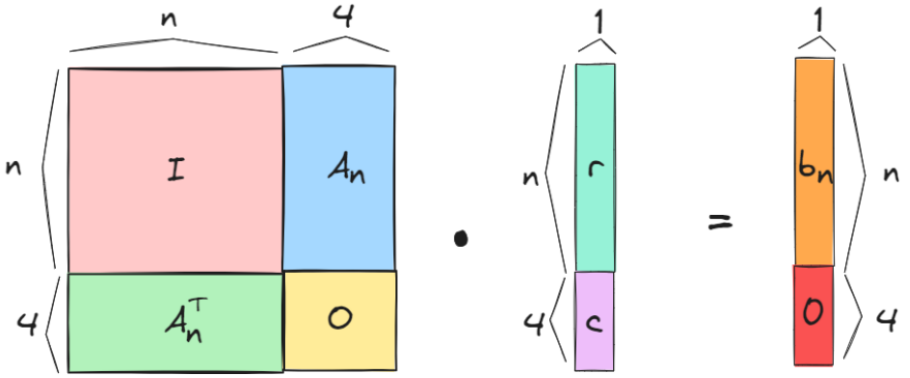

Figure 10: Augmented System

Then, I solved this augmented system. After I obtained the solution, I extracted its last 4 entry since it gives me the approximation $c_k^{(n)}$ to the coefficients c_k

c)

I calculated average errors for n = 5, n = 10, n = 100 and n = 1000 by running approximation process 1000 times and I obtained following results:

- Average error for n = 5 : 63201.05064985479
- Average error for n = 10 : 9947.540204176834
- Average error for n = 100 : 1593.0793040909243
- Average error for n = 1000 : 1029.8845735539326

This shows me that as we collect more samples, we get closer approximation to the actual values. Average error decreases as the number of sample increases. However, rate of decrease also decreases. For example, from n = 10 to n = 100, average error went to 1/10 of its initial value but from n = 100 to n = 1000, it went to 2/3 of its initial value. If we achieved to make all observations in R^n without sampling error, we could perfectly approximate to the actual values.