Numerical Computing Final Project
Spring 2020

Jia Zhao

November 15, 2020

Abstract

Computed Tomography (CT) has been widely used to produce a more detailed, cross-sectional image of our body. CT image is a combination of X-ray beams from different angles. So CT can detect some information that may be missed by pure X-rays, for instance, a tumor behind a bone. How to efficiently and accurately reconstruct a CT image has become a popular mathematical problem. In this final project report, we first present a theoretical analysis of the mathematical background for Kaczmarz Method, a famous reconstruction algorithm. We then analyze how the number of iterations and the number of different angles of X-ray beams can affect Kaczmarz's performance.

1 Introduction

Computed Tomography (CT) is an imaging tool that combines X-ray projections from many different angles to produce a more detailed, cross-sectional image of our body. Different from pure X-ray test which is a 2D image, CT image is 3D. When X-rays from different angles stack together, the third dimension is created because we can see the "depth" information. These stacked X-ray beams are received as a sinogram from the CT gantry, and represent the absorption amount of the tested subject. we want to reconstruct the original 2D image of the tested subject. This reconstruction problem is a mathematical problem that aims to generate the tomographic image from X-ray projection data from different angles.

The aim of this final project is to show how linear algebraic methods, especially Kaczmarz method, can be applied to reconstruct CT images.

2 Definitions and Notations

2.1 Simplified models

We first model the relationship between X-ray beams and a medium. The model is shown in Fig.1. We define the intensity, I, as the number of photons travelling along a line, l. I_{in} is incident intensity and I_{out} is the transmitted intensity. Suppose the medium has width Δs. Suppose the medium absorbs X-ray photons depending on μ, the absorption coefficient. Since I_{out} can be detected and I_{in} is known, we can calculate $I_{absorbed} = I_{out} - I_{in}$.

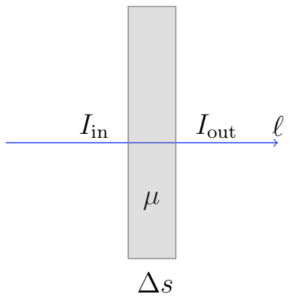
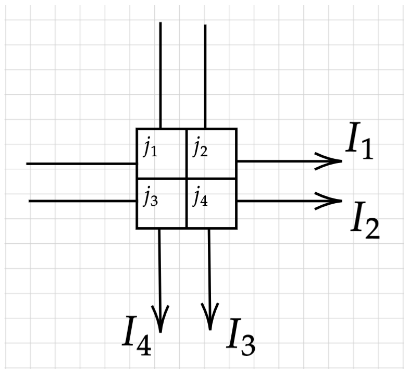


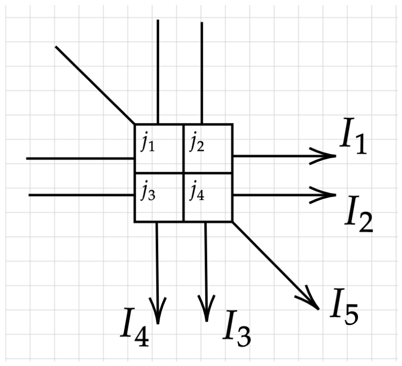
Figure 1: model the relationship between X-ray beams and a tissue

We now present a simplified model for CT reconstruction. we treat each CT image as a vector. Then each pixel is just a number in the vector. Pixels show different levels of colors because of different levels of absorption. For each pixel j, we have some X-ray beams passing through it. Each X-ray beam is denoted as I_i.

We present two examples in Fig.2:



(a) sub-model 1



(b) sub-model 2

Figure 2: two simplified models

We now build two linear systems in the form of $A\mathbf{x} = \mathbf{b}$ to represent the above two models. Let b_i be the total absorbed intensity of X-ray beam I_i, so $b_i = I_{i_{out}} - I_{i_{in}}$. Let each pixel have a unit side length. When X-ray beams pass through one pixel, we mark its travelling length at entry $A_{i,j}$ of matrix A. So $A \in \mathbb{R}^{m\times n}$, where m is the number of X-ray beams and n is the number of pixels. Let $x_i = \mu_i$, so $\mathbf{x} \in \mathbb{R}^n$ contains all absorption coefficients.

Then, the above two models can be represented in two linear systems as follows:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ \sqrt{2} & 0 & 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} \tag{2}$$

We generalize the above idea and present notations in the next section 2.2.

2.2 Notations

We illustrate the above two linear systems (1) and (2) as follows:

$$A\mathbf{x} = \mathbf{b}$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \tag{3}$$

we can write the system as:

$$\begin{cases} a_{1,1} \cdot \mu_1 + a_{1,2} \cdot \mu_2 + \cdots + a_{1,n} \cdot \mu_m = b_1 \\ a_{2,1} \cdot \mu_1 + a_{2,2} \cdot \mu_2 + \cdots + a_{2,n} \cdot \mu_m = b_2 \\ \vdots \\ a_{m,1} \cdot \mu_1 + a_{m,2} \cdot \mu_2 + \cdots + a_{m,n} \cdot \mu_m = b_m \end{cases} . \tag{4}$$

Note, $a_{i,j}$ is the travelling length of X-ray beam I_i on pixel j, μ_i is the absorption coefficient of pixel i, b_i is the total absorbed intensity of X-ray beam I_i. Thus, A is the matrix that maps absorption coefficients (vector $\mathbf{x}$) to the data in the detector pixels (vector $\mathbf{b}$).

In the linear system (3), A is an $m \times n$ matrix; $\mathbf{x}$ is the n-vector we want to find out; and $\mathbf{b}$ is the tomographic data we have. Our goal is to solve the above lienar system using Kaczmarz Method. Each element in the solution $\mathbf{x}$ represents a pixel in the corresponding reconstructed image, and indeed represents how one medium absorbs X-ray beams. To get the reconstructed image, we reshape $\mathbf{x}$ to an $N \times N$ matrix ($N \times N$ is the number of total elements in vector $\mathbf{x}$).

2.3 Kaczmarz Method

Kaczmarz method is an iterative algorithm for solving $Ax = b$.

We first present some geometrical intuitions. From Eq.(4), each equation defines an affine hyperplane in R^n. Assuming the solution to $Ax = b$ is unique, then the solution x^* is the point where all affine hyperplane intersects.

Now, we present an idea how Kaczmarz converge to a solution:

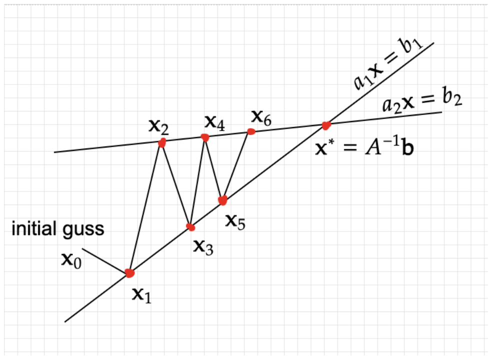


Figure 3: Kaczmarz Convergence Illustration

The basic idea is: at each iteration, Kaczmarz method computes the next vector such that precisely only one of the equations ($a_1x = b_1$ and $a_2x = b_2$) is satisfied. After the initial guess x_0 is selected, we project x_0 onto $a_1x = b_1$ and get x_1. Then we project x_1 onto $a_2x = b_2$ and get x_2. We keep projecting x_k until we get x^* that is the intersection of two hyperplanes ($a_1x^* = b_1$ and $a_2x^* = b_2$).

Let a_i be the ith row of matrix A. Let m be the number of rows in A. The computation at each iteration is: for $k = 0, 1, ...,$

$$x^{k+1} = x^k + \frac{b_i - < a_i, x^k >}{||a_i||^2} \bar{a}_i, \tag{5}$$

where $i = k \bmod m$, $i = 1, 2, ..., m$; $\bar{a}_i$ denotes the complex conjugate of row vector a_i.

3 Results

We now present results using Kaczmarz method for CT image reconstruction.

The publicly available MATLAB software AIR ToolsII (see [2]) is used for data simulation and Kaczmarz method.

3.1 Data and Goal

We use the famous Shepp-Logan Phantom to test our methods, which is simulated using MATLAB AIR ToolsII. We take our first test data as an example.

Let `[A1,b1,x1] = paralleltomo(N,theta1)`. `paralleltomo(N,theta1)` creates a 2D parallel-beam tomography test problem, where N is the dimension (here $N = 64$) of the phantom, `theta1` is a vector containing projection angles in degrees (here `theta1 = 0:2:180`). Output `A1` is a coefficient matrix with `N^2` columns and `len(theta1)*p` rows, where `p` is the number of used X-rays for each angle (`p` represent the number of X-rays. By default, `p = round(sqrt(2)*N)`). Output `b1` is the RHS of the linear system with `length(theta)*p` elements (as we described in 2.1 and 2.2, `b1` is a vector containing total absorbed intensity of each X-ray beam). Output `x1` is the exact solution of the linear system, i.e. the original phantom (as we described in 2.1 and 2.2, `x1` is a vector containing absorption coefficients for each pixel).

we use `X = kaczmarz(A1,b1,k)` to perform Kaczmarz method. By default, the initial guess `x0=0`. Output `X` is a matrix containing result of each iteration in columns.

Our goal is to reconstruct the phantom from simulated parallel beam tomographic data with different conditions, i.e. we want to solve the linear system $Ax = b$.

3.2 Kaczmarz results

We now present results using Kaczmarz method.

3.2.1 Iterations Analysis

We first test how the number of iterations would affect Kaczmarz results. Our test data is `[A1,b1,x1] = paralleltomo(N,theta1)`, where `N = 64` and `theta1 = 0:2:180`. So there are 91 different angles in total. `A1` is a 8281×4096 matrix, and `b1` is a 8281×1 vector. `x1` is a 4096×1 vector. To show `x1` as the original phantom, we reshape `x1` to a 64×64 matrix.

We want to see errors before looking into individual reconstructed image. We first run Kaczmarz Method with 200 iterations, errors are presented in Fig.4. Three observations are worth noting:

1) between $k = 8$ and $k = 21$, the error at each iteration is not always getting smaller. In fact, it goes up and down between two consecutive iterations. We can see that the error goes down a little bit more than it goes up. So the error goes down overall. Also, errors between $k = 21$ and $k = 43$ are not always going down either. But the oscillation between iterations seems much smaller than between $k = 8$ and $k = 21$. So the curve between $k = 21$ and $k = 43$ looks smoother than the curve between $k = 8$ and $k = 21$. To see this performance more clearly, we present errors between $k = 1$ and $k = 40$ in Fig.5.

2) we can see that the error drops fast before the 8th iteration, roughly from 0.7 to 0.2; however, after going up and down until the 40th iteration, it stays around 0.04 for all the

rest iterations. We propose two hypotheses here: i) Kaczmarz has the characteristic that converging fast at the very beginning and then converging very slowly after some point; ii) between these two stages (converge fast and do not converge), errors always go up and down. We will test our hypotheses in later examples.

3) recall the convergence model we illustrated in Fig.3. Theoretically, Kaczmarz should go beyond the optimal point and produce worse results. So it's reasonable for us to expect a turning point of the error, before which the error is getting smaller and after which the error is getting larger. We first try 800 iterations and the error is shown in Fig.6. Note, we only plot error for $k = 50, 100, 200, 400, 800$, so the "going up and down" phenomena is not present here. From Fig.6, we can see that errors keep getting smaller. So we want to try more iterations. We then try 4000 iterations and the error is shown in Fig.7. Sadly, we still don't get the turning point. But running 4000 iterations already takes several minutes and the error only ends around 0.027. Since Kaczmarz converges much more slowly as iterations grow, we can doubt it may take more than 20K, even 30K, iterations to *possibly* find the turning point, which would take maybe half an hour to run. At the same time, it seems that Kaczmarz can only find an approximation but not an exact solution. Since it takes thousands of iterations to approach that exact solution, we can claim that it's rare, at least not very possible, for Kaczmarz to go beyond the exact solution. The good news is, from this figure, it's obvious that we get a pretty satisfying result in the first 50 iterations.

Now, let's look at some reconstructed images. From our above error analysis, 0.04 is a good (small enough) error, which appears around $k = 200$. So we present some results between $k = 1$ and $k = 200$ in Fig.9 and Fig.10, as well as the original phantom in Fig.8. Obviously, when $k = 1$, the reconstructed image is not clear and looks like pixelated. This gets much better when $k = 10$. After $k = 10$, images gets clearer but in a much slower rate. This echos our above analysis regarding the error. We can still see some pixelation-like mark when $k = 20, 30, 50$. But $k = 50$ is clear enough. Then we look at $k = 100$, we find less pixelation-like mark. If we compare $k = 150$ and $k = 200$ with $k = 100$, it seems that they have less pixelation-like mark (but this is not very obvious). If we compare $k = 150$ with $k = 200$, our eyes cannot tell any more difference.

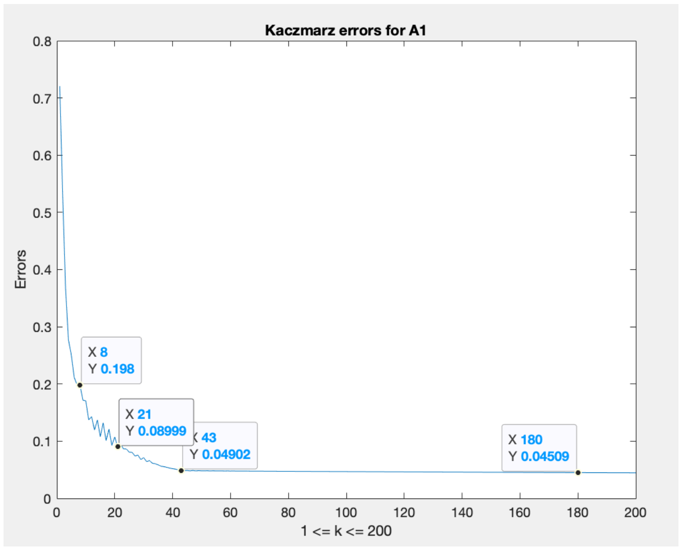


Figure 4: Errors using Kaczmarz method for example 1, kmax = 200

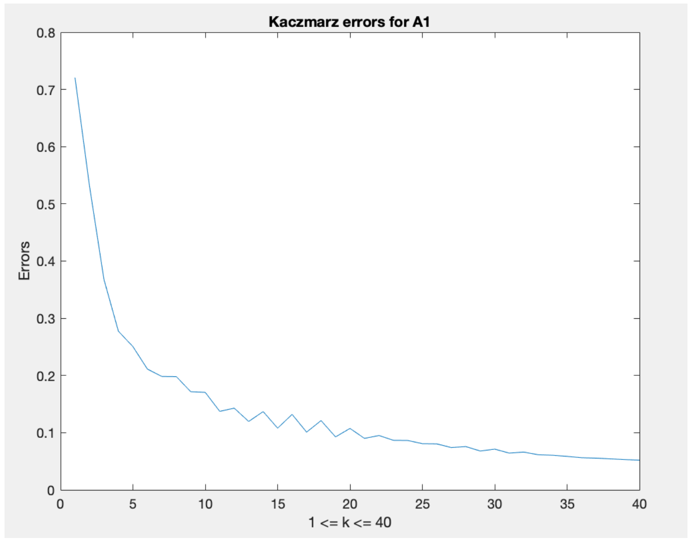


Figure 5: Errors using Kaczmarz method for example 1, kmax = 40

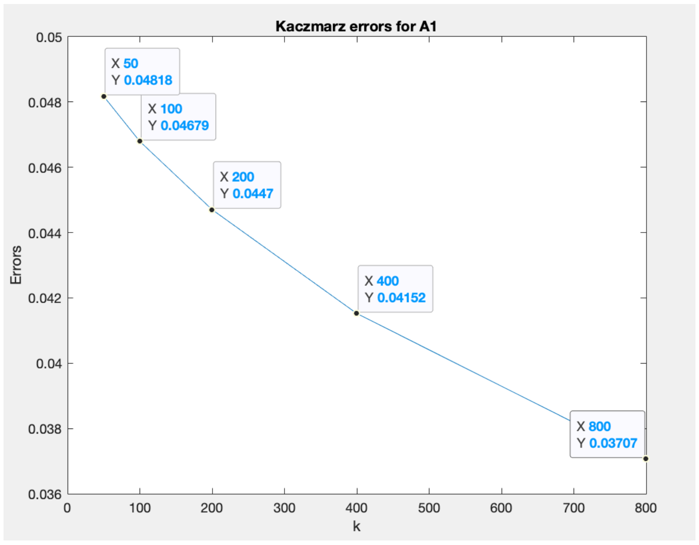


Figure 6: Errors using Kaczmarz method for example 1, kmax = 800

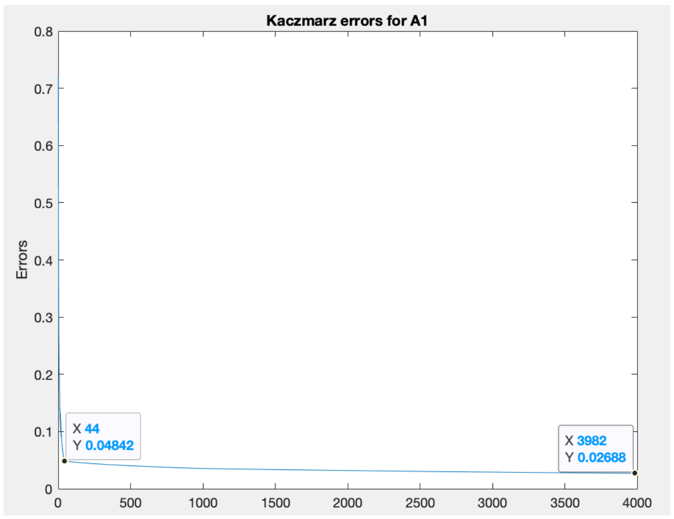


Figure 7: Errors using Kaczmarz method for example 1, kmax = 4000

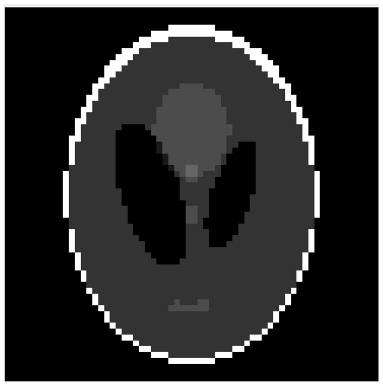
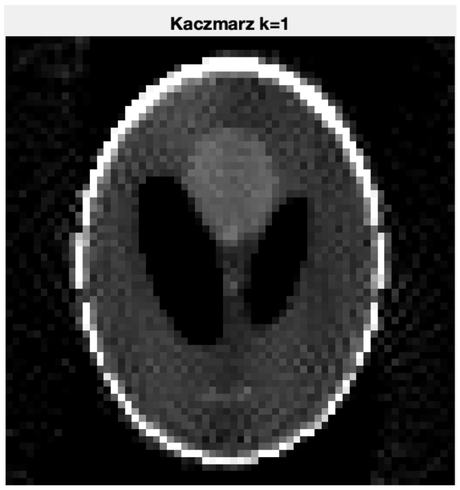
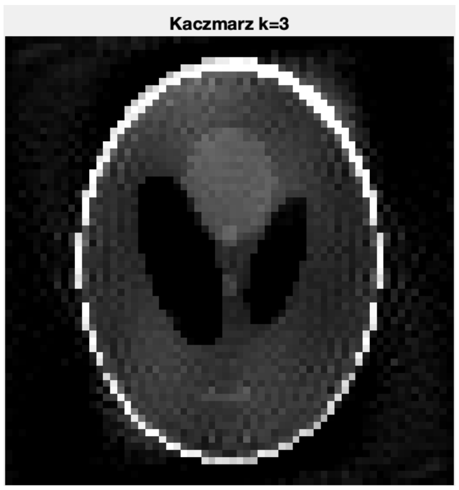


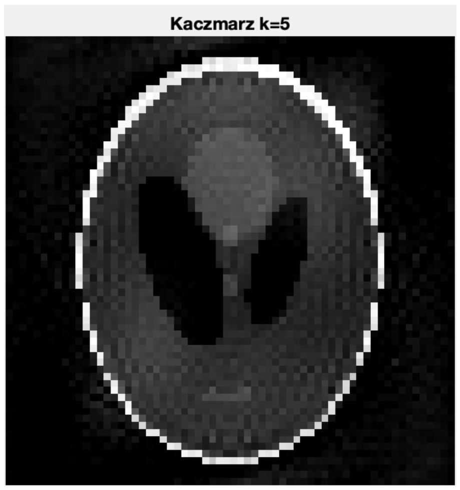
Figure 8: Original Phantom



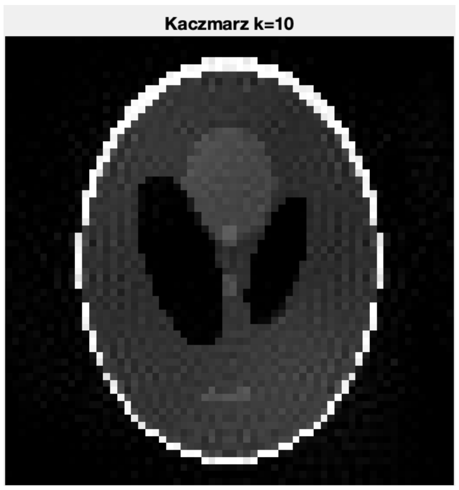
(a) k=1



(b) k=3

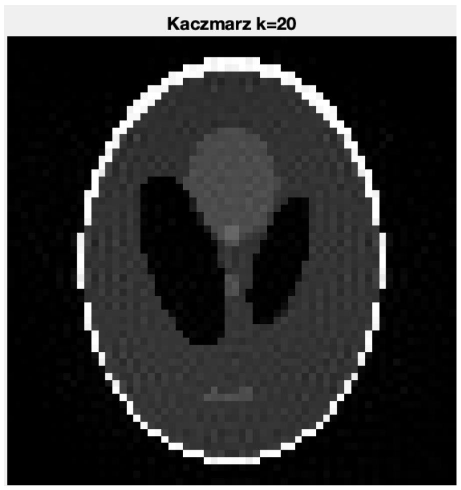


(c) k=5

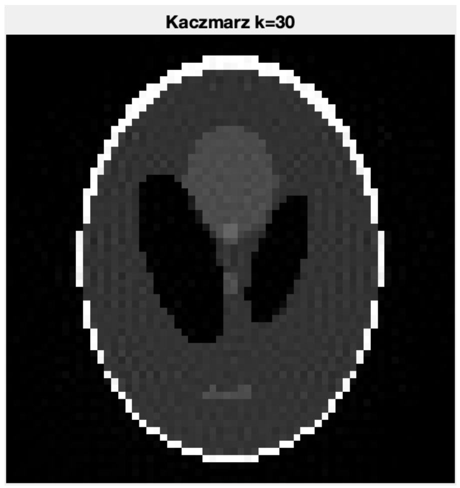


(d) k=10

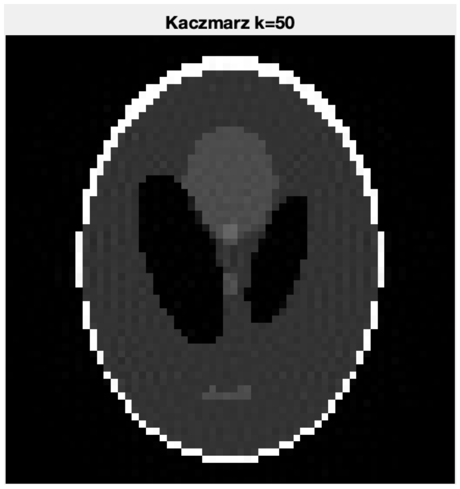
Figure 9: Results using Kaczmarz for example 1 Part I



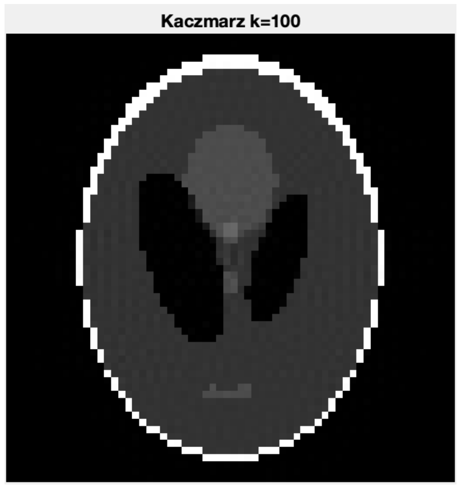
(a) k=20



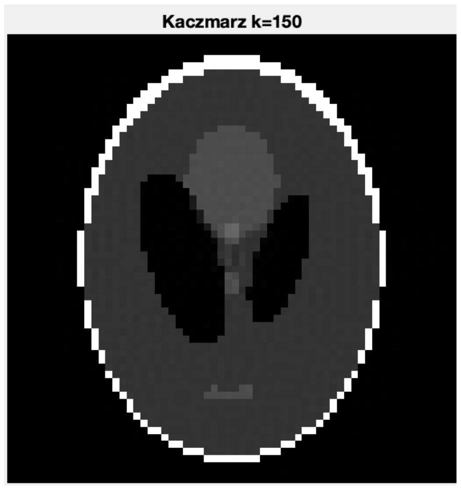
(b) k=30



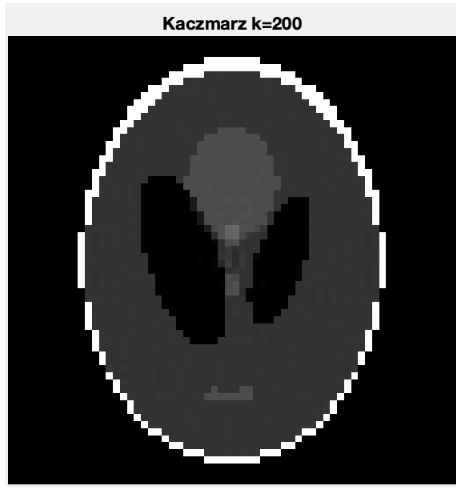
(c) k=50



(d) k=100



(e) k=150



(f) k=200

Figure 10: Results using Kaczmarz for example 1 Part II

3.2.2 Theta Analysis

We want to test how the choice of different `theta` can affect Kaczmarz's results. Our example 2 is `theta2 = 0:10:180`, `[A2, b2, x2] = paralleltomo(N,theta2)`. We have only 19 angles now (recall, we had 91 different angles in example 1).

We still use `X = kaczmarz(A2,b2,k)` to perform Kaczmarz method.

Similar as Section-3.2.1, we analyze errors first. See Fig.11. Surprisingly, we find a turning point of the error (which is at $k = 21$), but this is **not** the turning point we look for. Here, the error does not even reach 0 and suddenly turned to grow larger. The average error is above 0.6, which is pretty high. So given other conditions the same comparing to example 1, we can conclude that less angles produce worse reconstruction results.

Let's evaluate the two hypotheses we proposed in Section 3.2.1. Recall the two hypotheses are i) Kaczmarz method converges fast at the beginning and very slowly after some point; ii) between these two stages (converge fast and do not converge), errors always go ups and downs.

If we view $k = 2$ as the first iteration, errors drop fast within two iterations, then errors go ups and downs for a while (between $k = 5$ and $k = 15$), then errors almost stay at the same value between $k = 15$ and $k = 20$. So our hypotheses still hold between $[0, 20]$. However, after $k = 20$, our hypotheses do not hold anymore because errors become larger and larger. Kaczmarz method is not converging to the solution, but to the opposite direction. So we can add one condition onto our hypotheses: should only by applied when Kaczmarz method is converging.

Different from example 1, in which the error started to drop at $k = 1$. Here, the error goes up a bit and then started to drop. One possible assumption is that the initial guess is not a good one.

Now, let's look at some reconstructed results. From our above error analysis, we cannot expect the reconstructed images to be very clear. Also, since all errors are within $[0.6, 0.635]$, we cannot expect obvious differences among the reconstructed images either. As Fig.13 shows, all three reconstruction images are pretty vague and blur. Our eyes cannot really tell the difference among them. In order to convince ourselves that they are different, we calculate their norms using MATLAB:

```
norm(X(:,2)) = 13.4987
norm(X(:,15)) = 13.8717
norm(X(:,40)) = 13.9025
```

Since changing theta is interesting, we want to try a more extreme example. Our example 3 is `theta3 = 0:50:180 = [0,50,100,150]`. So we have only 4 different angles now. Similar as example 2, we first present errors. From Fig.12, we can see that the error is huge! All errors are between 0.861 and 0.8635. The smallest error is even larger than the largest error (≈ 0.7) for example 1 (see Fig.5). Similar as example 2, Kaczmarz method starts to converge to the opposite direction from some point. Different from example 2, now it starts to converge to the opposite direction right after $k = 2$. One potential reason for "converging opposite" is: at some iteration k, x_k is projected in a wrong way and leads all following iterations producing x towards the opposite direction. Since x_k is projected to two hyperplanes (as we discussed in Section-2.3), which are affected by $\mathbf{b}$, changing $\mathbf{b}$ would then affect the overall performance of Kaczmarz method. $\mathbf{b}$ is the total absorbed intensity, whose size is `length(theta)*p` (see Section-3.1). Then, changing `theta` will affect projections within each iterations and finally lead to some unexpected results.

We now test our proposed hypotheses. However, there seems no need to test the hypotheses because Kaczmarz method does not converge in this case. Error only drops when $k = 2$, so there's no room for any observations. Remember we added one condition to our hypotheses when analyzing example 2: should only be applied when Kaczmarz method is converging. Since no convergence is happening here, we cannot apply our hypotheses.

Let's look at the reconstructed images for example 3 in Fig.14. Based on our above error analysis, we cannot expect clear images. Since $k = 2$ produces seems the best approximation and $k = 15$ seems the worst in our range, we only show images at these two iterations. As we expected, they are not clear at all. We can only see two vague black blocks in the middle. Since errors for both images are within $[0.861, 0.8635]$, both images should appear very similar and our eyes definitely cannot tell any difference.

So, in what `theta` range does Kaczmarz methods work well? To figure this out, we need to test some more `theta`. Recall, in example 1, we have `theta1 = 0:2:180`; in example 2, we have `theta2 = 0:10:180`; and in example 3, we have `theta3 =0:50:180`. we found that Kaczmarz method does not converge well *at least* at `theta2`. So we test some all cases between `theta1` and `theta2`.

Let `theta4=0:3:180`, `theta5=0:4:180`, `theta6=0:5:180`, `theta7=0:6:180`, `theta8=0:7:180`, `theta9=0:8:180`, and `theta10=0:9:180`. Let A_i denote the corresponding problem. We plot errors in Fig.15 and Fig.16. The good news is, the two proposed hypotheses hold for all of them! We can find errors going ups and downs after quick converging and before slow converging. However, the bad news is, average errors become larger as the number of angles become smaller. From example 4 to example 10, if we look at the error e_{40} at $k = 40$ for each example, we can find that e_{40} gets larger and larger. Example 10 seems like an exception because, for some unsure reason, e_{40} for example 10 is smaller than e_{40} for example 8 and example 9. Furthermore, most errors directly drop from $k = 1$, however, errors in example 8 and example 9 started to drop from $k = 2$ (similar as example 2 and example 3). It seems that initial guess and the number of different angles may together determine how Kaczmarz behaves. But the relation is not explicit.

It suggests that more angles are needed for a better result. We can be sure that `theta11=0:1:180` would produce a much better result w.r.t. errors. This is shown in Fig.16. Within 40 iterations, the error already drops down to 0.1. Moreover, our hypotheses still hold!

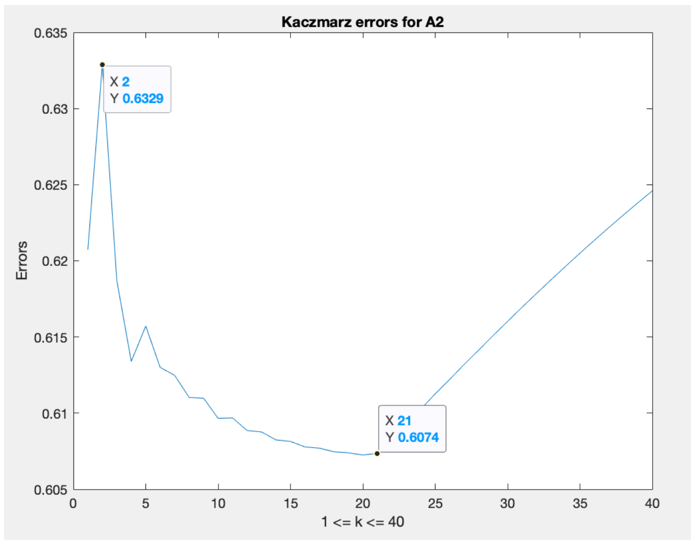


Figure 11: Errors using Kaczmarz method for example 2, kmax = 40

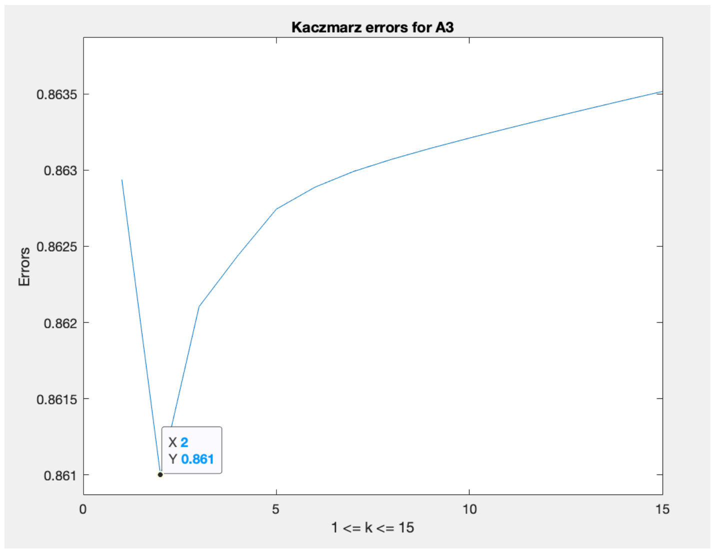
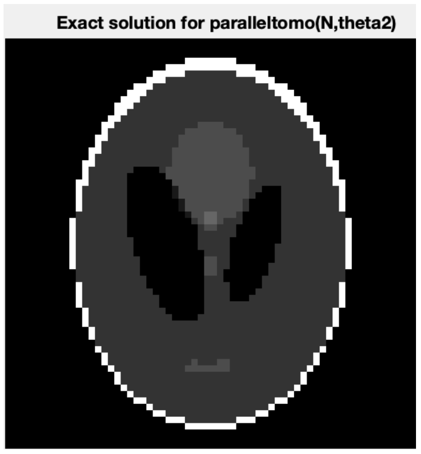
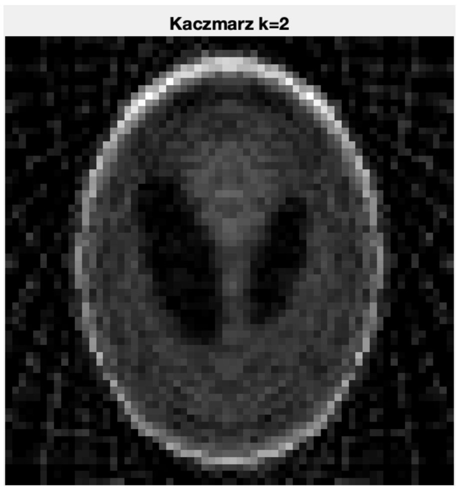


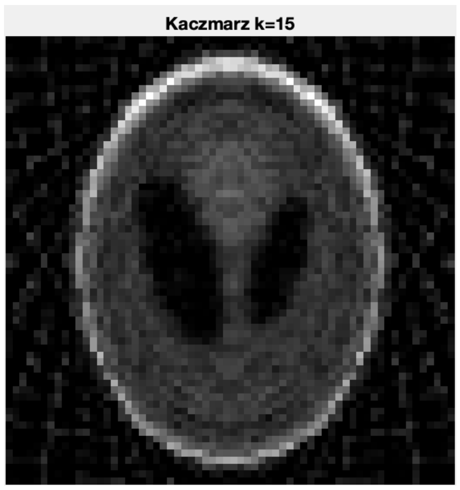
Figure 12: Errors using Kaczmarz method for example 3, kmax = 15



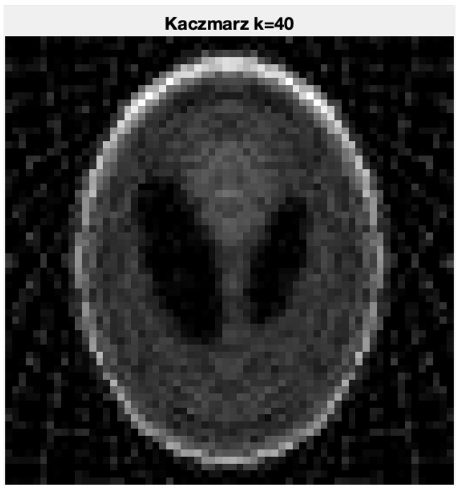
(a) original phantom



(b) k=2

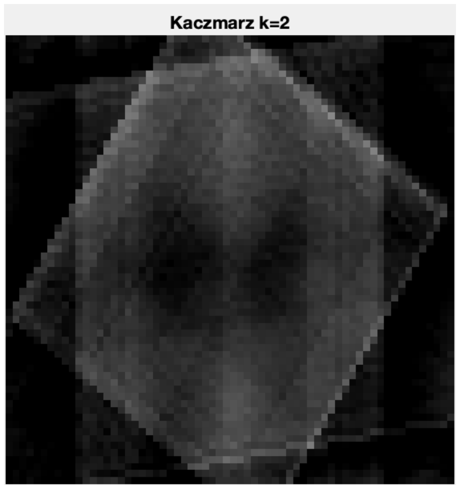


(c) k=15

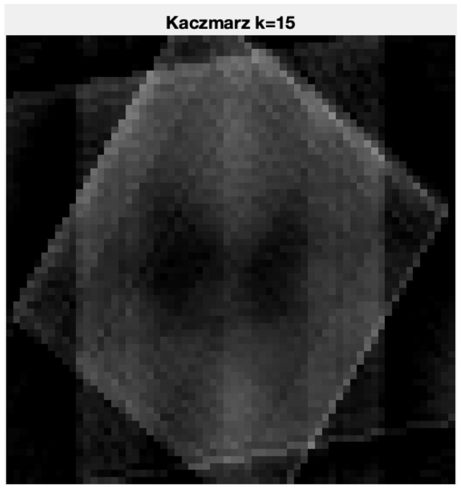


(d) k=40

Figure 13: Results using Kaczmarz for example 2

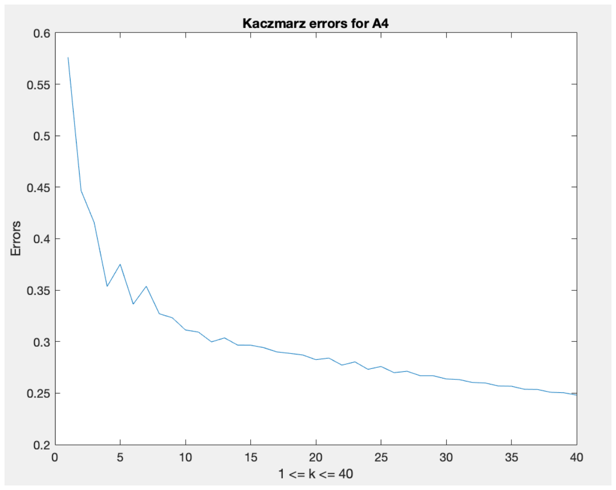


(a) k=2

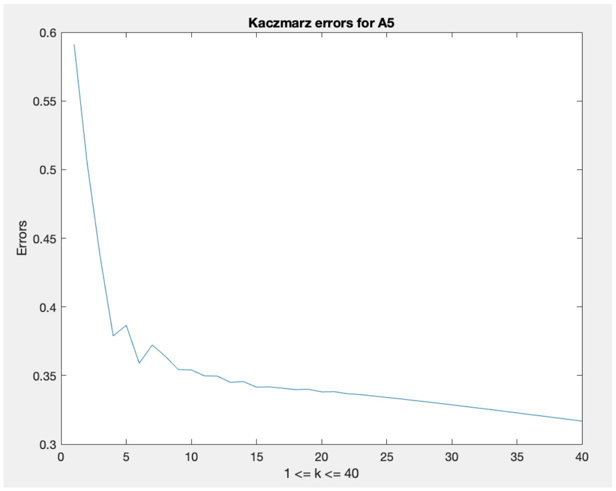


(b) k=15

Figure 14: Results using Kaczmarz for example 3

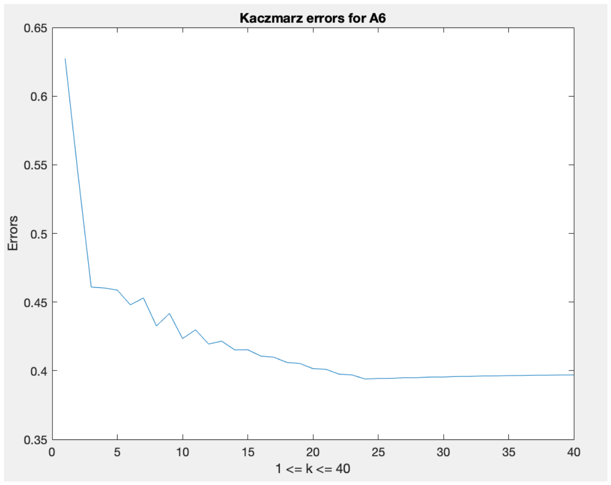


(a) A4

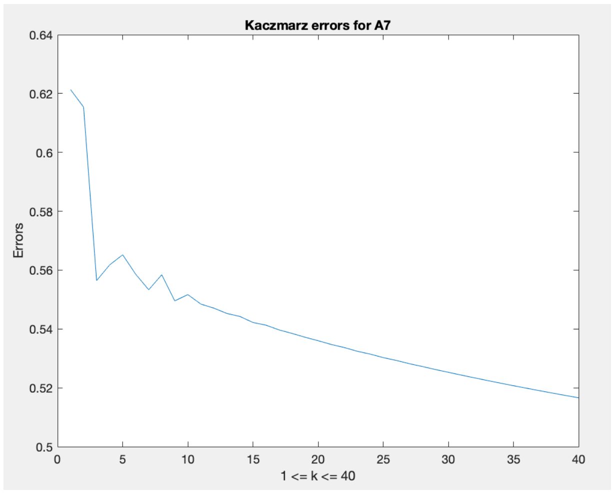


(b) A5

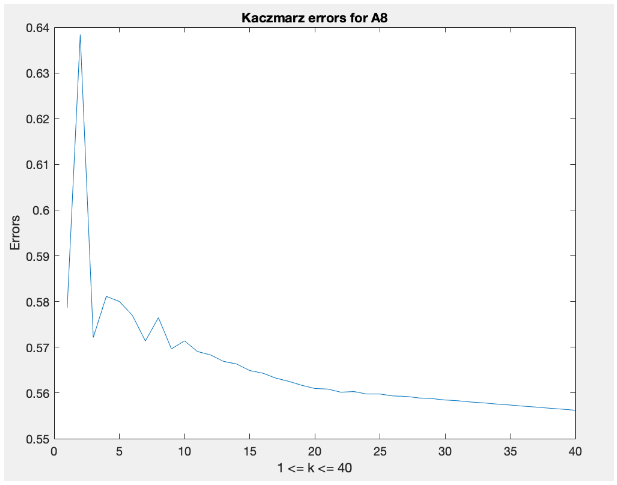
Figure 15: Errors for Example 4,5



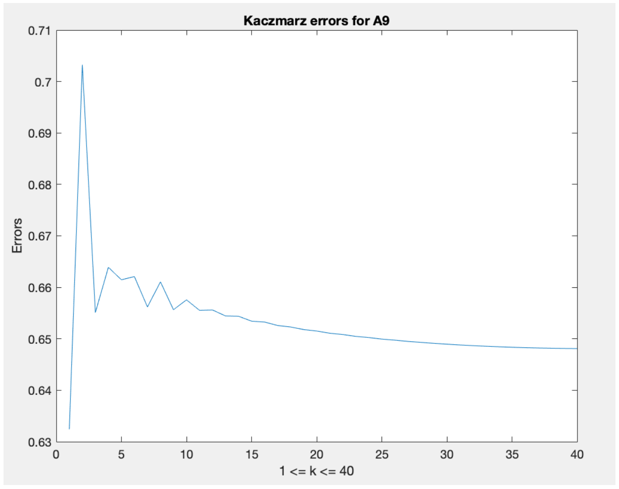
(a) A6



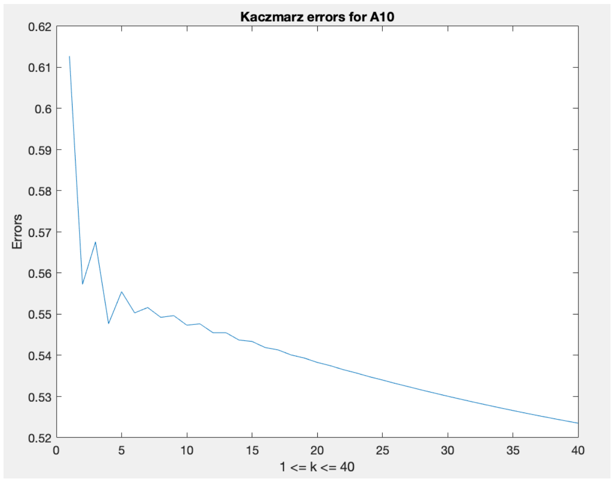
(b) A7



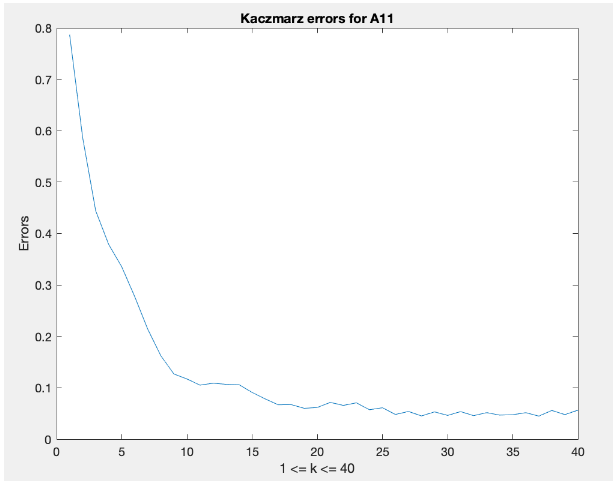
(c) A8



(d) A9



(e) A10



(f) A11

Figure 16: Errors for Example 6,7,8,9,10,11

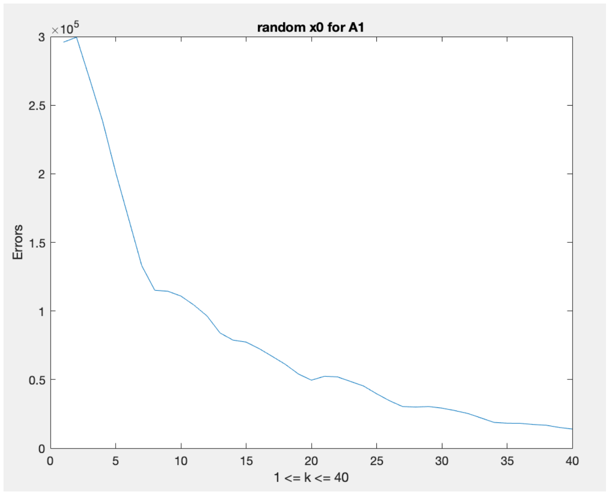
3.3 Initial Guess Analysis

As we realized in Section-3.2.2, initial guess may influence how Kaczmarz behaves. We try four random `x0` for the first example (which behaves well).

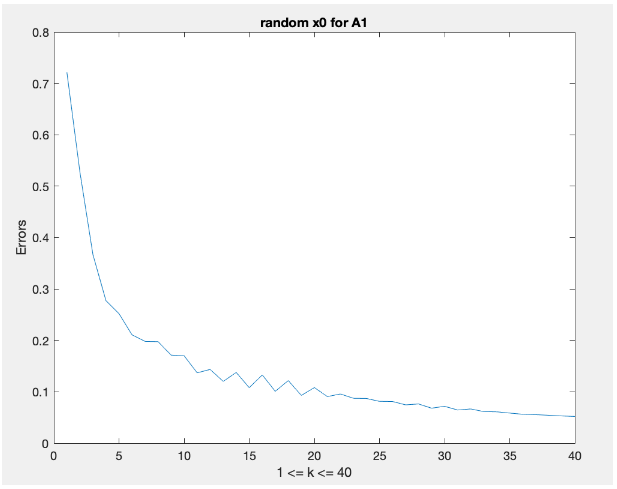
```
n = 1:4096
x01 = n.^2, norm(x01-x1) = 4.8034e+08
x02 = 1./n.^2, norm(x02-x1) = 15.8815
x03 = n, norm(x03-x1) = 1.5137e+05
x04 = n.*10^(-2), norm(x04-x1) = 1.5131e+04
```

Corresponding results are shown in Fig. 17 and Fig,18. From the figures, we can see that initial guess `x02` is closest to the exact solution `x1` and produces a pretty similar result with initial guess `x0=0` (see Fig.5). The other three initial guesses are all far away from the exact solution `x1`, but `x01`'s corresponding errors are huge (in the order of 10^5), while `x03` and `x04` still produce good results. If we look close at `x01`'s result, we can find that the error goes up a bit at the very beginning and then drops, which is similar with example 2 and example 3. Nevertheless, the two hypotheses hold for all four tests.

It is still not clear how initial guess, together with the choice of `theta`, can affect Kaczmarz's behavior. But one point is clear: Kaczmarz method converges super fast at the first several iterations (`theta` should be carefully chosen), even when the initial guess is far away from the exact solution.

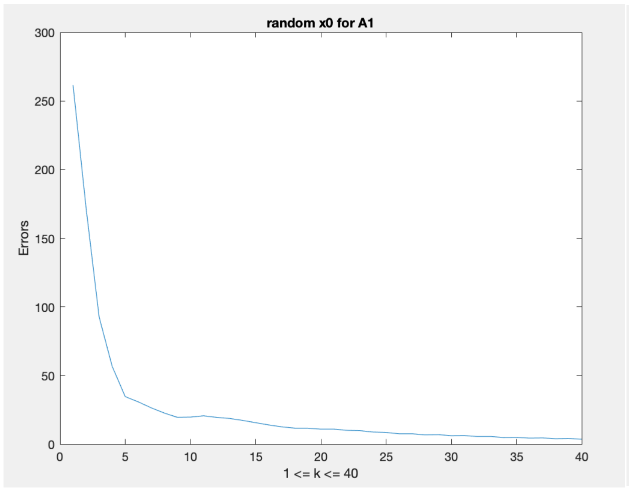


(a) initial guess x01

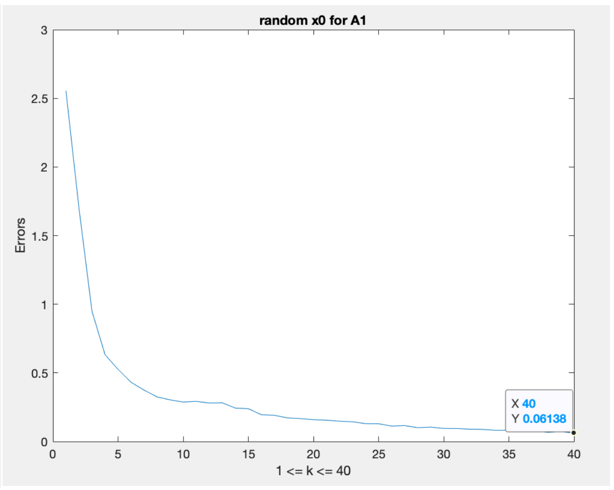


(b) initial guess x02

Figure 17: Test Different Initial Guesses



(a) initial guess x03



(b) initial guess x04

Figure 18: Test Different Initial Guesses

4 Summary

In this final project report, we have introduced the most basic form of Kaczmarz Method and applied it to CT image reconstruction.

We have conducted a detailed analysis on how the number of iterations, the choice of `theta`, and the choice of initial guess `x0` can affect the performance of Kaczmarz Method. As pointed out in Section-3.2.1, the more iteraions we have, the more accurate the result is. However, we do not need to pursue a perfect result because Kaczmarz Method converges much more slowly after some iterations. We should not sacrifice running time for improvements in the order of 10^{-3}.

As pointed out in Section-3.2.2, more different angles bring a better result. When there are around 91 different angles of X-rays, i.e. `theta=0:2:180`, Kaczmarz Method performs good results because the error drops to 0.1 pretty fast (shown in Fig.5). When there are less than 91 different angles of X-rays, the error remains at a high level, i.e. we cannot get good reconstruction images. When there are even less different angles of X-rays, like example 2 and example 3, Kaczmarz Method performs bad and even converges to the opposite direction. As we analyzed, the main part of Kaczmarz Method is projecting current x_k onto one of the hyperplanes and get the next x_{k+1}. Since hyperplanes are determined by $\mathbf{b}$ and $\mathbf{b}$ is a vector containing `length(theta)*p` elements, our choice of `theta` has big impacts on the projection, and thus on the overall performance of Kaczmarz Method.

As pointed out in Section-3.3, even the initial guess is far away from the exact solution, it still converges super fast at the first several iterations. Different initial guesses only result in different first several errors. If we combine our results from Section-3.2.1, we know `theta` should be carefully chosen first. However, it still remains unclear how initial guess and the choice of different angles can affect the Kaczmarz's behavior. Given the Kaczmarz method is indeed finding the point where two hyperplanes intersect, initial guess and the choice of different angles should together have some specific impacts on Kaczmarz's performance.

Along our analysis, we also proposed and tested two hypotheses about Kaczmarz's special behaviors: 1) Kaczmarz Method converges fast at the very beginning and very slowly after some point; 2) between these two stages (converge fast and do not converge), errors always go ups and downs. These hypotheses can only be applied to the converging part of Kaczmarz Method. If Kacmarz Method is not even converging under some conditions, these hypotheses should not be applied.

In the report, we only analyzed three factors of Kaczmarz Method. As we realized in the Section-3.3, it is still not clear how the initial guess, together with the choice of `theta`, can affect Kaczmarz's performance. Due to time limit, we cannot test more. But this can be an interesting future work. Moreover, the number of X-rays is set the same for all of our examples, which is the default `p=sqrt(2)*N` and `N=64`. One future work can focus on the impacts of different number of X-rays. Last but not least, we only test the most basic Kaczmarz Model, more work can be done on the variants of the method, including adding a relaxation parameter and using randomized Kaczmarz Method.

That's all. Thanks for reading!

References

[1] P. C. Hansen (2012), Algebraic Methods for Computed Tomography, http://people.compute.dtu.dk/pcha/HDtomo/Day1algebraic.pdf.

[2] P. C. Hansen and J. S. Jørgensen (2018), AIR Tools II: Algebraic Iterative Reconstruction Methods, Improved Implementation, *Numerical Algorithms* 79, 107-137.