Alternating Minimization for Computed Tomography with Unknown Geometry Parameters.

Phuong Mai Huynh Pham
Emory University
phuongmai.huynhpham@gmail.com

Manuel Santana
Utah State University
manuelarturosantana@gmail.com

Advisors:

Ana Castillo
Proximity Learning
ana.t.castillo.rivas@gmail.com

James Nagy
Emory University
jnagy@emory.edu

Abstract

Due to the COVID-19 pandemic, there is an increasing demand for portable CT machines worldwide in order to diagnose patients in a variety of settings. This has led to a need for CT image reconstruction algorithms that can produce high quality images in the case when multiple types of geometry parameters have been perturbed. In this paper we present an alternating minimization algorithm to address this issue, where one step minimizes a regularized linear least squares problem, and the other step minimizes a bounded non-linear least squares problem. Additionally, we survey existing methods to accelerate convergence of the algorithm and discuss implementation details. Finally, numerical experiments are conducted to illustrate the effectiveness of the algorithm.

1 Introduction

In medical imaging, computed tomography (CT) techniques are becoming increasingly popular for their ability to produce high quality images of the human body. These help doctors diagnose several types of cancers and recently handle COVID-19 cases. A CT scanner is a device that is composed of a scanning gantry, X-ray generator, computer system, console panel and a physician's viewing console. The scanning gantry is the part that produces and detects X-rays. In a typical CT scan, a patient will lay on a bed that will move through the gantry. An X-ray tube rotates around the patient and projects X-ray beams through the human body at different angles. These X-ray measurements are then processed on a computer using mathematical algorithms to create tomographic (cross-sectional) images of the tissues inside the body. Limitations arise when using CT scanners for these medical procedures since these devices require extensive care, such as regular maintenance. Additionally, transporting these to remote locations is not an easy task. Point-of-care imaging addresses these challenges by allowing radiologists to add portable CT scanners to their departments to increase patient satisfaction and improve medical outcomes. However, the parameters that are associated with the geometry of these devices cannot always be precisely calibrated in point-of-care situations. These parameters may change over time, when the scanner is adjusted during the image acquisition process or when transported to a new location.

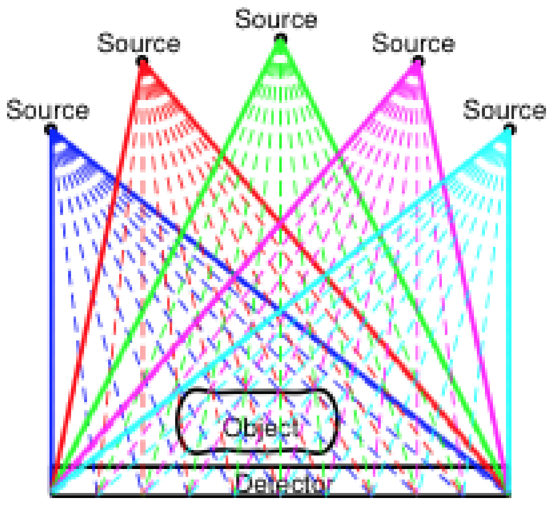


Figure 1: Object-Source-Detector

Figure 1 illustrates how an X-ray source rotates around an object during a typical CT acquisition process. The location of the source is measured by a view angle, i.e, location of the source on a circle around the object, and the distance from the source to the center of the object. If the location of the X-ray source has been perturbed, the reconstructed image will be of very poor quality. The case where only the view angles have been perturbed is an active area of research [16] [17]. In [6] Ding devises the numerical scheme we study, and uses it to investigate when the angles and the distance from the source to the object are unknown. The purpose of this paper is to study reconstruction methods in the case when both view angles and distance from the source to the object have been perturbed. Ding performed one small experiment of this type, and we build upon his work by testing larger images, larger perturbations, more acceleration techniques, more linear solvers, parallelizing the code, and adding to the `IRtools` [8] package to include this algorithm.

The rest of the paper will proceed as follows. In section 2 we give a brief background to the CT problem, present the algorithm, and discuss some theory behind the algorithm. Next in section 3 we provide an alternate form of the algorithm as a fixed point iteration and discuss acceleration techniques to improve convergence. Section 4 outlines considerations for implementation including parallelization. Numerical experiments and results are outlined in section 5. Future directions are presented in section 6. An example on how to set up and run an experiment is given in the appendix.

2 Block Coordinate Descent

In this section, we discuss the mathematical concepts and modeling of the CT image reconstruction problem as well as different techniques involved to effectively solve the inverse, ill-posed problem. We begin by setting up the computed tomography problem which boils down to finding attenuation coefficients of an object made up of multiple materials. An image can then be constructed by a color mapping based on the attenuation coefficients. For an object made up of a single material, Beer's Law describes the amount of radiation that can pass through it [7]:

$$I = I_0 e^{-\mu d}. \tag{2.1}$$

In the above equation I_0 is the initial energy of the X-ray that goes into an object, and I is the energy of the X-ray that leaves it. d is the distance that the X-ray beam travels through the object and μ is the attenuation coefficient. Figure 3 illustrates these parameters.

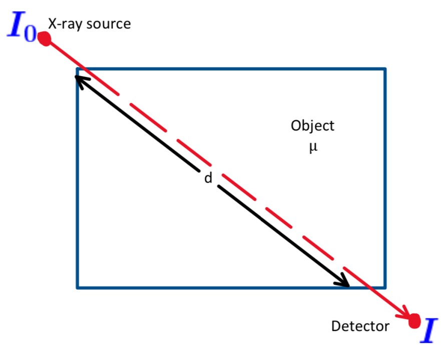


Figure 2: Beer's Law for a Single Material Object

Beer's law implies that

$$-\log\left(\frac{I}{I_0}\right) = \mu d.$$

Thus, if the initial and final radiation amounts are known, as well as the distance d, finding the attenuation coefficient is as simple as solving a linear equation. In practice objects of interest are made up of several materials, which each have their own attenuation coefficients as illustrated in figure 3.

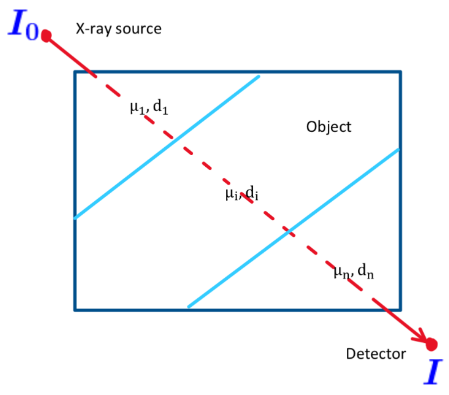


Figure 3: Beer's Law for an Object of Multiple Materials

Applying Beer's law to each different material, assuming there are n materials, the X-ray energy leaving the object can be written as

$$I = I_0 e^{-\sum_{j=1}^{n} \mu_j d_j}$$

which implies

$$-\log\left(\frac{I}{I_0}\right) = \sum_{j=1}^{n} \mu_j d_j.$$

This gives one linear equation in multiple variables, and so additional X-ray beams, at different angles, need to be transmitted through the object to obtain n linearly independent equations for the n variables.

Most often in practice it is not known precisely where each material begins and ends. To overcome this a pixel grid is overlaid on the object, and the attenuation of each pixel is sought. This involves taking measurements using many X-ray projections from different angles. Often in practice the X-ray source projects multiple X-rays spread out like a fan as illustrated in figure 4. In addition, the source and detector rotate around the object to obtain additional measurements.

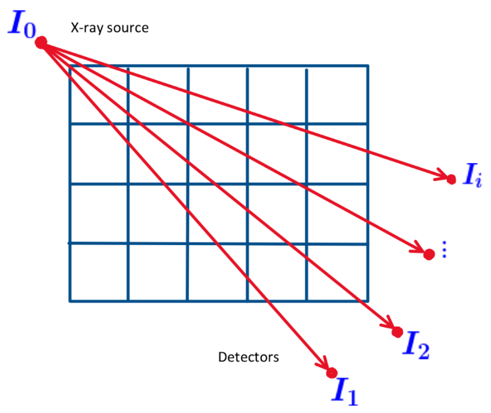


Figure 4: Fan-beam X-rays

Now suppose that there are n pixels and m X-ray beams are projected through the object. Let d_{ij} be the distance that the ith X-ray beam travels through pixel j. I_0 is still the initial energy, and let I_i be the energy of the ith X-ray beam as it leaves the object and hits the detector. Then the relationship between the unknown attenuation coefficients and the known distances and energies can be written as a system of equations

$$\underbrace{\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} -\log{(I_1/I_0)} \\ -\log(I_2/I_0) \\ \vdots \\ -\log{(I_m/I_0)} \end{bmatrix}}_{\mathbf{b}}. \tag{2.2}$$

In this system of equations the vector $\mathbf{b}$ is known as the *sinogram*. Note that the matrix $\mathbf{A}$ in the above equation will be sparse because each X-ray will only pass through a few of the pixels.

Now, notice that each distance d_{ij} is a function of the X-ray source location. Let R denote the distance of the X-ray source to the center of the object, and θ denote the angle of rotation of the source relative to the center of the object. Then the matrix $\mathbf{A}$ is a function of the vector $\mathbf{p}$, denoted as $\mathbf{A}(\mathbf{p})$, where $\mathbf{p}$ contains the R_i, and θ_i for each source location (that is each time a fan of X-ray beams is emitted).

In a standard computed tomography problem with known geometry parameters, solving equation (2.2) requires regularization due to the ill-conditioning of $\mathbf{A}$ [9]. In this paper, the problem we seek to study has unknown geometry parameters, and can be stated as

$$\mathbf{x} = \arg\min_{x,p} ||\mathbf{A}(\mathbf{p})\mathbf{x} - \mathbf{b}||_2^2 + \alpha^2 ||\mathbf{x}||_2^2. \tag{2.3}$$

Here α is a regularization parameter, which allows us to get a more accurate solution when the matrix $\mathbf{A}$ is ill-conditioned. The regularization parameter recasts the ill-conditioned problem to a nearby well conditioned one. We refer the reader to [10],[13] for information on the role of regularization in inverse and least squares problems. The solution to (2.3) can be approximated by using an alternating minimization scheme known as block coordinate descent or BCD, which we describe in algorithm 1.

Algorithm 1: Tomography Block Coordinate Descent

```
1 Input $\mathbf{p_0} \in \mathbb{R}^m$
2 Output $\mathbf{x_k}$
3 for $k = 0, 1, 2, ...$ do
4     $\mathbf{x_{k+1}} = \arg\min ||\mathbf{A}(\mathbf{p_k})\mathbf{x} - \mathbf{b}||_2^2 + \alpha^2||\mathbf{x}||_2^2$
5     $\mathbf{p_{k+1}} = \arg\min ||\mathbf{A}(\mathbf{p})\mathbf{x_{k+1}} - \mathbf{b}||_2^2$
```

Thus, equation (2.3) can be approximated by an alternating minimization scheme where one iteration involves solving a large linear least squares problem, and a large non-linear least squares problem. To solve the large regularized linear least squares problem in step 4, we mainly use a hybrid conjugate gradient method (LSQR) that can automatically choose regularization parameters. Mathematical details describing this method can be found in [12]. The implementation we use is described in [8] (specifically, we use the method called IRhybrid_lsqr). A comparision of this to other linear least squares solvers is given in section 5.3. To solve the non-linear least squares problem in step 5 we use a projected quasi-Newton method in Kelley's implicit filtering package [14].

3 Acceleration Techniques

In this section we discuss techniques to accelerate the convergence of the BCD algorithm. The key is to recast the problem in terms of a fixed point problem.

Algorithm 2: Fixed Point Tomography Reconstruction

```
1 Input $\mathbf{x}_0 \in \mathbb{R}^n$
2 Output $\mathbf{x}_k$
3 for $k = 0, 1, 2, ...$ do
4     $\mathbf{p}_{k+1} = \arg\min ||\mathbf{A}(\mathbf{p})\mathbf{x}_k - \mathbf{b}||_2^2$
5     $\mathbf{x}_{k+1} = \arg\min ||\mathbf{A}(\mathbf{p}_{k+1})\mathbf{x} - \mathbf{b}||_2^2 + \alpha^2||\mathbf{x}||_2^2$
```

Now we define a function g that is one iteration of algorithm 2. That is assuming the $\mathbf{b}$, α, and the function $\mathbf{A}(\mathbf{p})$ are known define $g(\mathbf{x}_k)$ as

$$g(\mathbf{x}_k) := \arg\min ||\mathbf{A}(\mathbf{p}_{k+1})\mathbf{x} - \mathbf{b}||_2^2 + \alpha^2||\mathbf{x}||_2^2$$

where $\mathbf{p}_{k+1}$ is defined as

$$\mathbf{p}_{k+1} := \arg\min ||\mathbf{A}(\mathbf{p})\mathbf{x}_k - \mathbf{b}||_2^2.$$

Thus, the problem of finding an approximate solution of (2.3) becomes that of finding a fixed point of g. This is useful as acceleration techniques for fixed point problems have been widely studied [19]. It should be noted that the problem can also be stated in terms of finding a fixed point of $g(\omega_k)$ where

$$g(\omega_k) := \arg\min ||\mathbf{A}(\mathbf{p})\mathbf{x} - \mathbf{b}||_2^2 + \alpha^2||\mathbf{x}||_2^2$$

and $\omega_k := (\mathbf{x}_k, \mathbf{p}_k)$. However numerically this performed worse as shown in subsection 5.2.

In our numerical experiments we used three fixed point acceleration schemes, and the remainder of the section will be devoted to discussing them. For the remainder of the section we will denote a function as $f(x)$ if it is a function is a single variable, and $F(\mathbf{x})$ if it is a function of several variables.

We motivate the first two acceleration methods starting with the single variable fixed point problem of finding a point x_k such that $f(x_k) = x_{k+1}$. To begin, we let Δ denote the difference operator, that is let $\Delta x_k := f(x_k) - x_k$, $\Delta f(x_k) := f(f(x_k)) - f(x_k)$, and $\Delta^2 x_k := \Delta f(x_k) - \Delta x_k$. Aitken's Δ^2 method [1] along with its recursive application and the second order Steffensen method are popular choices for solving single variable fixed point problems [18]. The Aitken Δ^2 method is derived from approximating the multiplying constant for a linearly converging sequence, and can be stated as

$$x_{k+1} = x_k - \frac{(f(x_n) - x_n)^2}{f(f(x_k)) - 2f(x_k) + x_k} = x_k - \frac{(\Delta x_k)^2}{\Delta^2 x_k}.$$

Or equivalently

$$x_{k+1} = f(f(x_k)) - \frac{(\Delta f(x_k))^2}{\Delta^2 x_k}. \tag{3.1}$$

Vector generalizations of this algorithm have been extensively studied, see [19] for several references. Most come from defining the inverse of a vector $\mathbf{x}$ as

$$\mathbf{x}^{-1} := \mathbf{x}\frac{1}{||\mathbf{x}||_2^2}.$$

Application of this definition to (3.1) leads to the *Irons-Tuck* method [11]

$$\mathbf{x}_{k+1} = F(F(\mathbf{x}_k)) - \frac{\Delta F(\mathbf{x}_k)\cdot\Delta^2\mathbf{x}_k}{||\Delta^2\mathbf{x}_k||^2}\Delta F(\mathbf{x}_k).$$

Where $\cdot$ denotes the dot product, $F(\mathbf{x}_k)$ is the vector valued function we seek to find a fixed point of, and the Δ operator is defined as in the scalar case. Numerically the Irons-Tuck method has been shown to outperform many other vector generalizations of the Δ^2 process [15], but is computationally expensive when $F(\mathbf{x}_k)$ is expensive to evaluate. One alternative is to use the *crossed secant* method

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k) - \frac{(F(\mathbf{x}_k) - F(\mathbf{x}_{k-1})\cdot(\Delta\mathbf{x}_k - \Delta\mathbf{x}_{k-1})}{||(\Delta\mathbf{x}_k - \Delta\mathbf{x}_{k-1})||_2^2}\Delta\mathbf{x}_k.$$

It has the advantage of only needing one evaluation of F per iteration, and in many numerical tests performs similar to the Irons-Tuck method [19]. In fact, if each iteration of the crossed secant method is alternated with a standard fixed point iteration the resulting method is the Irons-Tuck method.

Another alternative is *Anderson Acceleration*, sometimes called *Anderson Mixing*. The general idea is to use information from a predetermined number of residuals to increase convergence. Here we only state a more easily implementable version of the algorithm. For a more general algorithm see [22]. Let r be the number of residuals to use, and denote

$$\mathcal{X}_k = (\Delta\mathbf{x}_{k-r}, \cdots, \Delta\mathbf{x}_{k-1}).$$

Then the Anderson Acceleration has the following form:

Algorithm 3: Anderson Acceleration

```
1 Input $\mathbf{x}_0 \in \mathbb{R}^n$, $r \geq 1$
2 Output $\mathbf{x}_k$
3 Compute $F(\mathbf{x}_0) = \mathbf{x}_0$ for $k = 1, 2, ...$ do
4     $r_k = \min\{r, k\}$
5     $\gamma^{(k)} = \arg\min ||\Delta\mathbf{x}_k - \mathcal{X}_k\gamma||_2$
6     $\mathbf{x}_{k+1} = F(\mathbf{x}_k) - \sum_{i=1}^{r_k}\gamma_i^{(k)}[(F(\mathbf{x}_{k-r_k+i+1}) - F(\mathbf{x}_{k-r_k+i})]$
```

Implementation details can be found in [21]. In short, the linear least squares problem can be efficiently solved by updating a QR factorization of $\mathcal{X}$ after every iteration. Additionally, we implemented a hyperparameter which removes a column of $\mathcal{X}$ if the condition number of the upper triangular matrix in the QR factorization becomes too large. In section 5.2 we compare the use of these three algorithms in our implementation.

4 Implementation

In this section we discuss practical considerations for the implementation of algorithm 1. Recall that each time the X-ray source fires it is associated to one angle and one R parameter. Therefore $\mathbf{A}$ is created as

$$\mathbf{A}(\mathbf{p}) = \begin{bmatrix}\mathbf{A}(\theta_1, R_1)\\ \vdots \\ \mathbf{A}(\theta_n, R_n)\end{bmatrix}.$$

The non-linear least squares problem can then be solved by solving subproblems of the form

$$\min_{\theta_i, R_i}||\mathbf{A}(\theta_i, R_i)\mathbf{x}_k - \mathbf{b}_i||_2^2$$

where each $\mathbf{b}_i$ is the corresponding part of the $\mathbf{b}$ vector. This is advantageous as it allows for easy parallelization of algorithm 1 leading to significant speedup, and a comparison is made in subsection 5.1.

To simulate the problem we will use the IRTools package [8]. IRTools provides several state of the art regularized linear least squares solvers which we will use in our implementation of algorithm 1. We will compare three linear least squares solvers available in the IRtools package which use different forms of regularization. The first is the hybrid-LSQR algorithm [12], which regularizes by adding a two-norm penalty term. The next is the iteratively re-weighted norm approach or IRN [20], which uses a one-norm regularization. Finally, the fast iterative shrinkage threshold algorithm or FISTA [2] is used which has no regularization term, but constrains the solution vector $\mathbf{x}$ such that $\mathbf{x} \geq 0$. A comparison of these methods is given in subsection 5.3.

To approximate the solution of the non-linear least squares problem we compared two methods. The first is a Trust-Region-Reflective algorithm studied in [4],[5] and implemented in the MATLAB function `lsqnonlin`. The second is the method of implicit filtering [14] which uses a projected quasi-Newton iteration using difference gradients. Both methods are for bounded problems with unknown derivative information, making them suitable for our problem, as the geometry parameters can be realistically bounded. In our numerical tests we found that implicit filtering led to a smaller image error and therefore it was used for all numerical experiments in the next section.

5 Numerical Results

In this section we highlight several experiments for the tomography problem. In each problem an initial guess for the R parameters and angle parameters were given. Then a perturbation generated uniformly was added to each angle and R value, and the BCD was run to produce a better image. For all tests Gaussian noise was added to the sinogram $\mathbf{b}$ creating a new vector $\mathbf{b}_{noise} = \mathbf{b} + \eta$, where $||\eta||_2/||\mathbf{b}||_2$ is equal to a specified noise level, in this case 0.01. The initial guess for the view angles was always 0 : 2 : 358, and the initial guess for the R parameters was 2. We will use the abbreviation $U(a, b)$ for a uniform distribution when describing perturbations to R and angle parameters. In the graphs below the angle and R parameter error represent the relative error between the true perturbations and the current approximation of those perturbations. The ℓ_2-norm was used for relative errors in each case and an initial guess of 0 was given for all perturbations.

The images used were the Shepp-Logan phantom (Figure 10a), and the spine image from the MATLAB image-processing toolbox (Figure 6: True Image), with each image size being 256×256 unless otherwise stated. All tests were run on a Microway system that has four Intel Xeon E5-4627 CPUs with 40 cores and 1 TB of memory running Ubuntu Linux. An example of setting up and running a test is given in appendix A. In the rest of this section we will first show a scaling study demonstrating the effectiveness of parallelizing the code. We will then show that the fixed-point acceleration techniques produce a smaller image error and converge faster than not using an acceleration technique. Finally, we compare several solvers for the linear least squares solver.

5.1 Parallel Speedup

In this section we demonstrate the effectiveness of the parallelization discussed in section 4. Table 1 shows a 20 iteration run of the same simulation using different image sizes with the image size being $N \times N$ where $n = N^2$. In all the timing tests the Shepp-Logan phantom was used with perturbations as realizations of the distribution $U(-0.25, 0.25)$ added to each angle and R value. The hybrid-LSQR algorithm was used as the linear least squares solver. The aforementioned computer was used with twelve workers.

N	32	64	128	256	512
Serial	257	714	1308	4238	13771
Parallel	35	166	289	1026	3810

Table 1: Run Times in Seconds

From the results in table 1 we see drastic speedup in all cases with the parallelization. For larger problems solving the linear least squares problem appears to take most of the time, since $\mathbf{A}$ is a large matrix. For example when N is 256 the size of $\mathbf{A}$ is 65160×65536 and when N is 512 the size of $\mathbf{A}$ is 131400×266256. Regardless, the parallel implementation made possible by the problem model provides significant speedup.

5.2 Acceleration Test

The next test we ran was to compare the different acceleration techniques using the spine image. Perturbations were added as realizations of $U(-0.5, 0.5)$. The hybrid-LSQR algorithm was used as the linear least squares solver, and the accelerated BCD algorithms were run for twenty iterations. In the legends in figure 5, BCD is for no acceleration, CS for crossed secant, AA for Anderson Acceleration, and IT for the Irons-Tuck method. To give reference to table 2 the tests were run on 12 local workers.

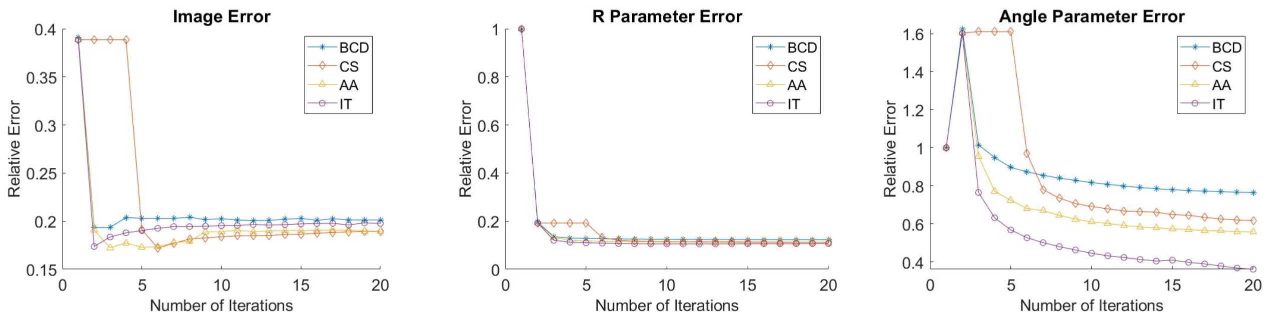


Figure 5: Graphs Comparing Acceleration Techniques

BCD	CS	AA	IT
1004	1042	1095	2375

Table 2: Run Times in Seconds

From these tests we see that the acceleration techniques converge to slightly smaller error norms, with the crossed secant method and Anderson Acceleration performing the best. The Irons-Tuck method converged much better in the angle parameters, but took much longer. This was expected as the function evaluation is quite expensive, and as previously noted Irons-Tuck requires an extra function evaluation at each iteration. Figure 6 shows the true image, and the image after the parameter optimization. Despite having slightly larger image error, the Irons-Tuck image seems to have the least background noise.

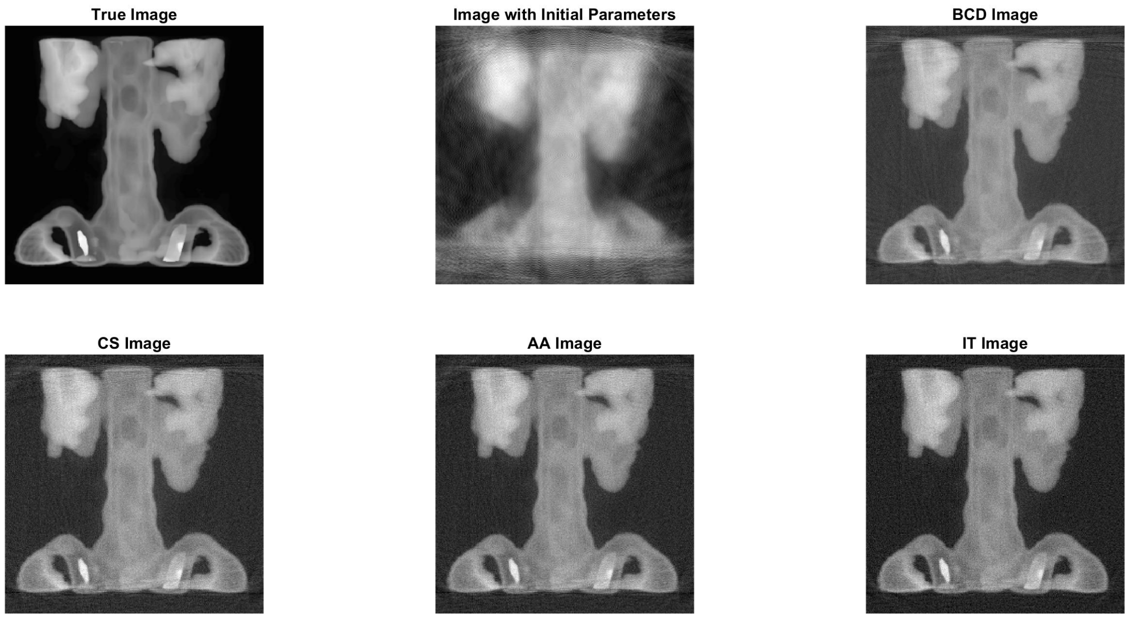


Figure 6: Spine Images after Parameter Optimization

As previously mentioned, in these tests we solved for the fixed point where the fixed point is the image vector. The table below compares the case when the fixed point vector is defined to be $\omega := (\mathbf{x}, \mathbf{p})$, with the same parameters as before.

Acceleration Scheme	Fixed Point Type	Angle Error	R Error	Image Error
Anderson	$\mathbf{x}$	0.557	0.111	0.189
	ω	14.642	14.020	0.190
Crossed Secant	$\mathbf{x}$	0.614	0.112	0.189
	ω	15.251	14.079	0.1774
Irons-Tuck	$\mathbf{x}$	0.359	0.108	0.199
	ω	14.009	14.009	0.195

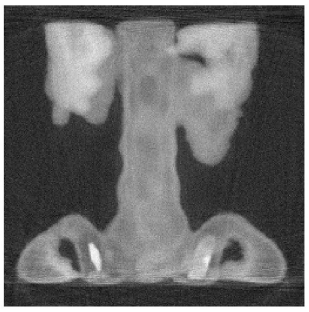
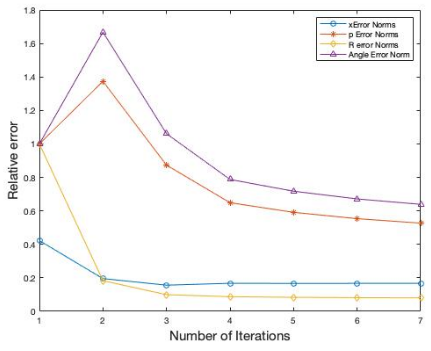
Table 3: Comparison of Fixed Point Problems

As can be seen from table 3 when the ω fixed point scheme is used the image error remains approximately the same, while the error in the angle and R parameters grows large. Thus, in our scheme we choose to only look for a fixed point of $\mathbf{x}$. The conclusion of this experiment overall is that when an accelerated fixed point scheme on $\mathbf{x}$ is used it may produce a smaller image error norm.

Having shown acceleration techniques looking for a fixed point of $\mathbf{x}$ to be effective in reducing error, we now demonstrate that giving a stopping tolerance they can effectively reduce the number of iterations needed to find a fixed point. Different from the other experiments the perturbations used here are sampled from $U(-0.25, 0.25)$ with other factors such as R and view angle initial guesses, and image size held the same. We use a relative error stopping criterion of

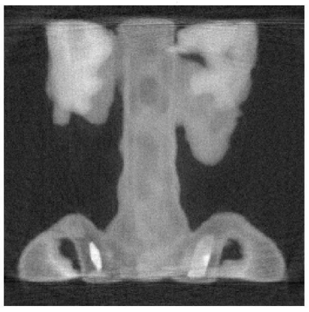
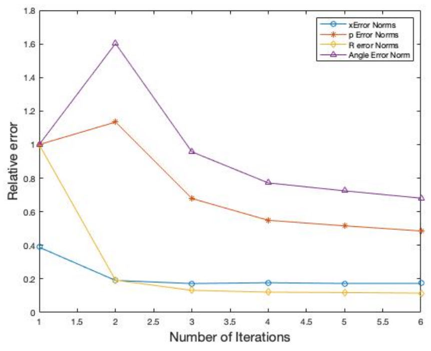
$$\frac{||\mathbf{x}_k - \mathbf{x}_{k-1}||_2}{||\mathbf{x}_{k-1}||_2} \leq 0.03.$$

Below we show the results using BCD without acceleration and with Anderson Acceleration.

Final Image Error Plot

Figure 7: Convergence of BCD without acceleration

Final Image Error Plot

Figure 8: Convergence of BCD with Anderson Acceleration

For this small example, using Anderson Acceleration caused the BCD to reach the stopping tolerance one iteration faster than without it. Since the fixed point function g is expensive to evaluate saving one iteration reduces computation time non-trivially. For example, in the test performed in figure 5 one evaluation of g averaged 50.2 seconds to evaluate.

5.3 Comparison of Linear Least Squares Solvers

In this section we compare the performance of three linear least squares solvers available in the IRtools package using hybrid LSQR, FISTA, IRN. We did two tests using the Shepp-Logan phantom. Perturbations were chosen from realizations of $U(-0.5, 0.5)$. No acceleration techniques were used in these tests.

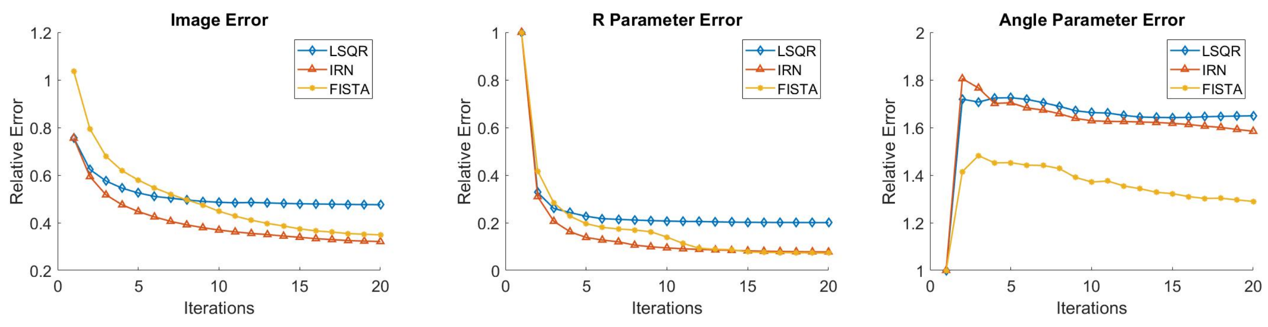
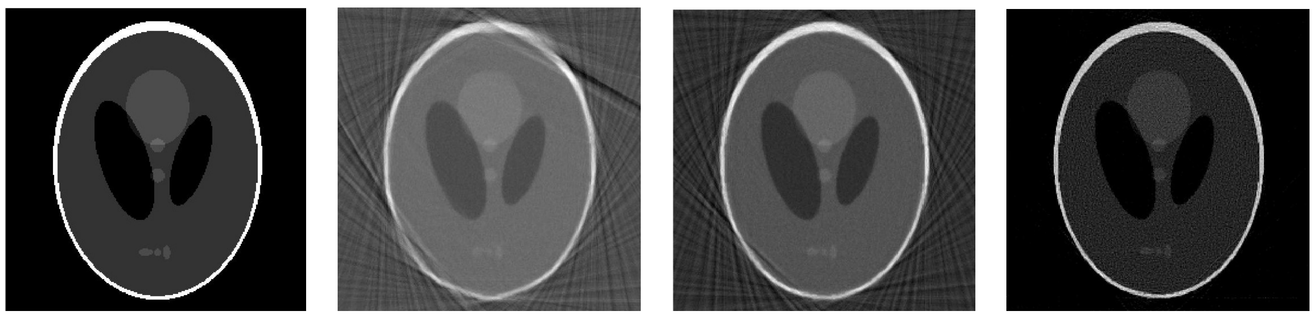


Figure 9: Graphs Comparing Linear Least Squares Solvers

In figure 9 we see that IRN and FISTA perform similarly, and drastically outperform the hyrbid-LSQR method. Interestingly all three methods perform poorly on the angle parameter estimation. Despite having similar image error norms, figure 10 illustrates that visually FISTA appears better than IRN.



(a) True Solution (b) LSQR (c) IRN (d) FISTA

Figure 10: Images for Linear Least Squares Comparison

The second test comparing the linear least squares solvers uses the same parameters as the previous test, except the perturbations are samples of $U(-1, 1)$. This is a significant amount of perturbation in this case as it is up to half the original value of R and half the distance between each angle. Figure 11 shows the relative errors.

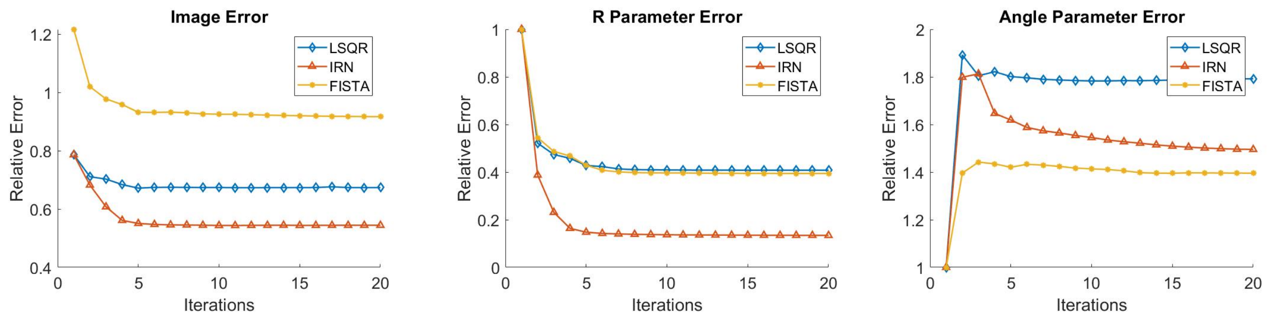
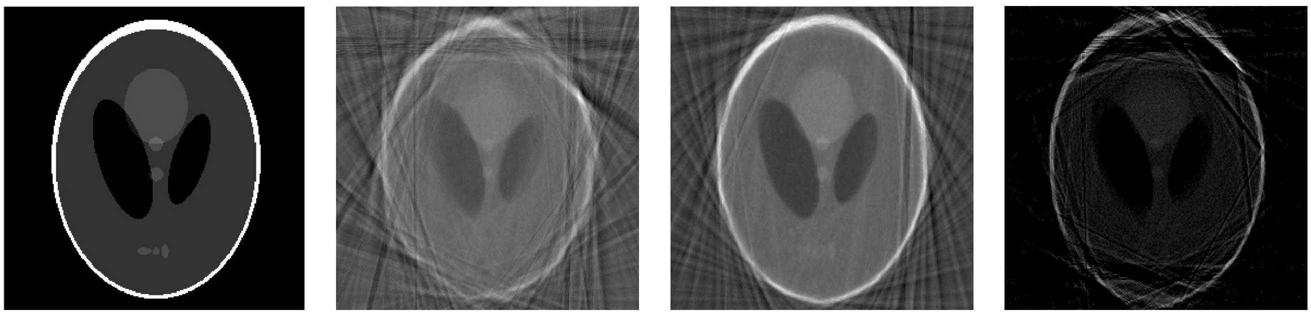


Figure 11: Graphs Comparing Linear Least Squares Solvers For Second Test Problem

Interestingly with a high perturbation level FISTA does the worst, while IRN continues to significantly outperform the LSQR algorithm. Unsurprisingly all three algorithms maintain a high relative error for the angles when larger perturbations are added. Figure 12 shows the reconstructed image, and while no image is a high quality reconstruction, it is clear IRN is the cleanest. The two tests show though that there is certainly an image and perturbation level dependence as to which method has the best reconstructions.



(a) True Solution (b) LSQR (c) IRN (d) FISTA

Figure 12: Images for Linear Least Squares Comparison for the Second Test Problem

6 Conclusion and Outlook

We have devised an algorithm to effectively estimate the unknown geometry parameters of an uncalibrated portable CT machine. We exploited the problem structure to allow for parallelization and have shown significant speedup. Also, we have used fixed-point acceleration techniques to both reduce the image error and the number of iterations required for convergence. Additionally, we have surveyed and tested up-to-date methods to solve regularized linear least squares problems, and have demonstrated that the best choice of solver depends on the size of the perturbations to the geometry parameters.

In the future, we hope to investigate more about why the acceleration techniques work better only with the image vector. Additionally, our stopping criterion was chosen naively, and there may be better a method for choosing a stopping criterion and error tolerance, and we intend to investigate those. Finally, we will look into other acceleration techniques for the optimization problem.

We also hope to apply our algorithm to other medical imaging applications where geometry parameters may be unknown such as bedside tomosynthesis [3].

7 Acknowledgements

We would like to express our appreciation to our mentor Dr. James Nagy. Additionally, we would like to thank the faculty at the Department of Mathematics, Emory University for dedicating their time to answering our questions, facilitating seminars and discussions related to this project. We also acknowledge the referees for their helpful comments in revising this manuscript. Lastly, we would like to acknowledge the National Science Foundation for giving us this opportunity to be part of NSF 2021 REU/RET Computational Mathematics for Data Science. This work was funded under NSF grant number DMS-2051019.

References

[1] A. Aitken. On Bernoulli's numerical solution of algebraic equations. *Proceedings of the Royal Society of Edinburgh*, 49:289–305, 1926.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences*, 2(1):183–202, 2009.

[3] J. Cant, A. Snoeckx, G. Behiels, P. M. Parizel, and J. Sijbers. Can portable tomosynthesis improve the diagnostic value of bedside chest X-ray in the intensive care unit? A proof of concept study. *European Radiology Experimental*, 2017.

[4] T. F. Coleman and Y. Li. On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds. *Mathematical Programming*, 67(2):189–224, 1994.

[5] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):418–445, 1996.

[6] P. Ding. Accelerated alternating minimization for X-ray 2 tomographic reconstruction. arXiv:2108.01017.

[7] C. L. Epstein. *Introduction to the Mathematics of Medical Imaging, Second Edition*. SIAM, Philadelphia, PA, 2007.

[8] S. Gazzola, P. C. Hansen, and J. G. Nagy. IR Tools: A MATLAB package of iterative regularization methods and large-scale test problems. *Numerical Algorithms*, 81:773–811, 2019.

[9] P. C. Hansen, J. S. Jørgensen, and W. R. B. Lionheart. *Computed Tomography: Algorithms, Insight and Just Enough Theory*. SIAM, Philadelphia, PA, 2021.

[10] P.C. Hansen. *Discrete Inverse Problems: Insights and Algorithms*. SIAM, Philadelphia, PA, 2010.

[11] B. M. Irons and R. C. Tuck. A version of the Aitken accelerator for computer iteration. *International Journal for Numerical Methods in Engineering*, 1:275–277, 1969.

[12] D. P. O'Leary J. Chung, J. G. Nagy. A weighted-GCV method for Lanczos-hybrid regularization. *Electronic Transmissions on Numerical Analysis*, 28:149–167, 2008.

[13] Åke Björck. *Numerical Methods For Least Squares Problems*. SIAM, Philadelphia, PA, 1996.

[14] C. T. Kelley. *Implicit Filtering*. SIAM, Philadelphia, PA, 2011.

[15] A. J. Macleod. Acceleration of vector sequences by multi-dimensional δ^2 methods. *Communications in Applied Numerical Methods*, 2(4), 1986.

[16] Y. Dong N. A. B. Riis and P. C. Hansen. Computed tomography reconstruction with uncertain view angles by iteratively updated model discrepancy. *Journal of Mathematical Imaging and Vision*, 63:133–143, 2021.

[17] Y.Doung N. A. B. Riis. A new iterative method for CT reconstruction with uncertain view angles. In *Scale Space and Variational Methods in Computer Vision*, pages 156–167, Cham, 2019. Springer International Publishing.

[18] Y. Nievergelt. Aitken's and Steffensen's accelerations in several variables. *Numerische Mathematik*, 59:295–310, 1991.

[19] I. Ramière and T. Helfer. Iterative residual-based vector methods to accelerate fixed point iterations. *Computers & Mathematics with Applications*, 70(9):2210–2226, 2015.

[20] P. Rodríguez and B. Wohlberg. An efficient algorithm for sparse representations with ℓ^p data fidelity term. In *Proceedings of 4th IEEE Andean Technical Conference (ANDESCON)*, Cusco, Perú, October 2008.

[21] H. F. Walker. Anderson acceleration, algorithms and implementations. `https://users.wpi.edu/~walker/Papers/anderson_accn_algs_imps.pdf`, 2011. Accessed 07/14/2021.

[22] H. F. Walker and P. Ni. Anderson acceleration for fixed point iterations. *SIAM Journal of Numerical Analysis*, 49(4):1715–1735, 2011.

A Running Numerical Experiments

Our implementation of algorithm 1 was built on top of the MATLAB IRtools package and naming conventions were chosen to mirror it. Our code can be found at https://github.com/manuelarturosantana/TomoREU2021. In the base IRtools package the function `PRset` is used to set up options for computed tomography problems as follows

```
options = PRset(options, ‘field_name1’,field_value1, ‘field_name2’,field_value2).
```

We have updated `PRset` to accept values relating to BCD, such as an initial guess for the parameters. When a field name is not passed in default values are selected. Then to simulate a CT problem with unknown geometry parameters `PRtomo_var` is used with the image size being $n \times n$

```
[b,ProbInfo] = PRtomo_var(n, options).
```

Above **b** is the right hand side vector with noise added, and `ProbInfo` contains the initial guess for the parameters. Thus, `PRtomo_var` generates all the data necessary to simulate the inverse problem. Next the `IRset` function is used to set up hyper-parameters for the solvers in IRtools

```
options = IRset(options, ‘field_name1’,field_value1, ‘field_name2’,field_value2).
```

Again `IRset` was updated to include parameters for BCD, such as which acceleration technique to use. Finally a new function `IRbcd` computes the BCD

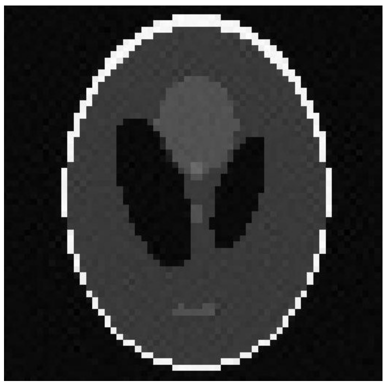
```
[x,iterInfo] = IRbcd(b,iterOptions, probInfo).
```

In the output `x` is the image vector after BCD terminates, and `iterInfo` contains information about errors at each iteration. Now we proceed to demonstrate an example of simulating and solving a CT reconstruction experiment with this code. To begin we generate the data

```
n = 64;
ProbOptions = PRset(‘Rpert’,0.25,‘anglespert’,0.25);
[b,probInfo] = PRtomo_var(n,ProbOptions);
```

This generates a test problem where perturbations are samplings of a uniform distribution on the interval $[-0.125, 0.125]$ are added to the R values and the angles. For ease of computation the same perturbation is added to every fourth of the angles and R values. The image for this problem is the Shepp-Logan phantom. Figure 13 (a) represents the true solution, (b) is the solution with noise in the sinogram **b**, but true parameters, and (c) is the solution with the initial guess parameters.

(a) True Solution


(b) True Parameters Solution

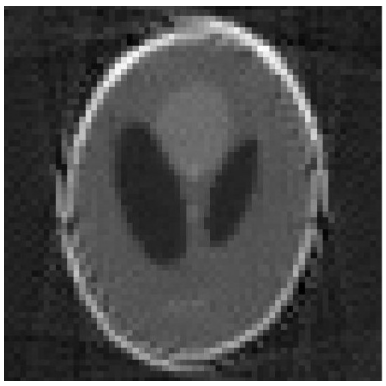

(c) Initial Parameters Solution

Figure 13: Example Problem

The following codes sets up the iteration options and runs the problem

```
iterOptions= IRset('nonlinSolver','imfil','accel','anderson','BCDmaxIter',10,...
'Rbounds',0.1250,'angleBounds',0.1250);
[x,iterInfo] = IRbcd(b,iterOptions,probInfo);
```

After solving the problem we see the solution and the relaltive error graph in figure 14. This illustrates after the BCD algorithm we get a comparable solution to when the true geometry parameters are known.

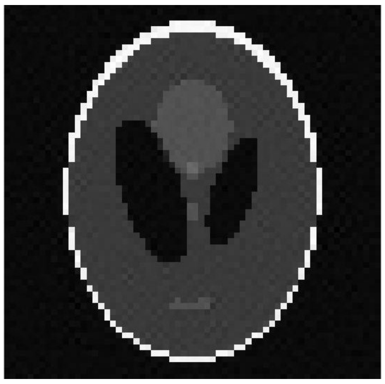


Image after BCD

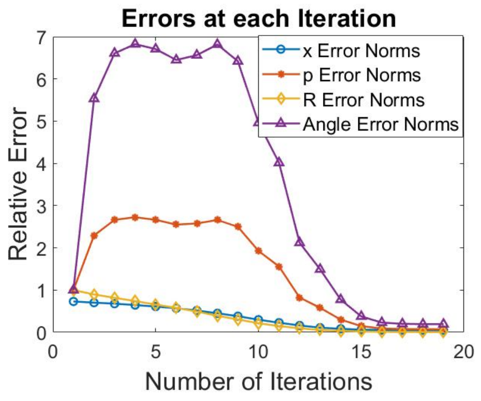


Figure 14: Example Problem Solution

The images in all five figures shown are generated with the following MATLAB code

```
PRshowbcd(iterInfo,probInfo).
```