Fall 2014

Selection of Step Size for Total Variation Minimization in CT

Anna N. Yeboah

SELECTION OF STEP SIZE FOR TOTAL VARIATION MINIMIZATION IN CT

by

ANNA NYARKO YEBOAH

(Under the Direction of Xiezhang Li)

ABSTRACT

Medical image reconstruction by total variation minimization is a newly developed area in computed tomography (CT). In compressed sensing literature, it has been shown that signals with sparse representations in an orthonormal basis may be reconstructed via l_1-minimization. Furthermore, if an image can be approximately modeled to be piecewise constant, then its gradient is sparse. The application of l_1-minimization to a sparse gradient, known as total variation minimization, may then be used to recover the image. In this paper, the steepest descent method is employed to update the approximation of the image. We propose a way to estimate an optimal step size so that the total variation is minimized. A new minimization problem is also proposed. Numerical tests are included to illustrate the improvement.

Key Words: Total Variation Minimization; Compressed Sensing; L1 Minimization; Reweighted Minimization; Computed Tomography; Compressive Sensing

2009 *Mathematics Subject Classification*: 65K05

SELECTION OF STEP SIZE FOR TOTAL VARIATION MINIMIZATION IN CT

by

ANNA NYARKO YEBOAH

B.S. in Mathematics

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment of the Requirement for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

2014

SELECTION OF STEP SIZE FOR TOTAL VARIATION MINIMIZATION IN CT

by

ANNA NYARKO YEBOAH

Major Professor:	Xiezhang Li
Committee:	Jiehua Zhu
	Yan Wu

Electronic Version Approved:

July 2014

ACKNOWLEDGMENTS

I give my first and utmost thanks to my Lord and Savior. I would never have made it this far without Him. I would also like to thank my thesis advisor, Dr. Xiezhang Li, for his wisdom and guidance throughout this entire process. I would like to thank my committe members, Dr. Yan Wu and Dr. Jiehua Zhu, for their time and additional instruction. Lastly, and certainly not least, I thank my mother, father, sister, and my family and friends for their love, encouragement and support during this time. I love you all dearly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Medical imaging is a process that provides visual images of the interior of the human body. These visual images allow for a noninvasive way to examine what goes on inside of the body. With this procedure, doctors are able to observe, diagnose, and treat various medical conditions. An accurate interpretation of what is happening on the inside of the body could mean the retardation or complete circumvention of certain ailments.

There are many different techniques, such as nuclear and optical imaging, that are used to obtain medical images. We focus on radiological methods, more specifically computerized tomography (CT). CT scanning combines x-ray images and computer processing to generate cross-sectional views, and sometimes three dimensional views, of the body. The mathematical process of converting x-ray projection data into a digital image is called image reconstruction.

Image quality is a growing concern in medicine. Image reconstruction techniques obtain the projection data, perform extensive data analysis and transformation, and produce a medical image. Traditionally, data is measured, or recorded, in full length and then compressed [2]. Direct methods are highly accurate due to the high number of measurements taken. But what happens when radiation, or other potentially harmful methods, are used to measure the data? We now have other factors to consider, such as safety and radioactive exposure. For this reason, we are interested in the conditions under which an image may be reconstructed using a significantly reduced number of measurements. The latter results in compressed data, where the interpretation, analysis and reconstruction of this data is called compressive sensing

(CS).

We consider the system $Ax = b$, where $A \in \mathbb{R}^{m \times N}$ is the projection matrix, $b \in \mathbb{R}^m$ is the projection data, and $x \in \mathbb{R}^N$ is the reconstructed signal. For the reasons explained earlier, this is usually an underdetermined system, where typically $m \ll N$ (for simplicity, N is a perfect square). Assume that x has a sparse representation in some orthonormal basis. CS theory states that for A satisfying the restricted isometry property (RIP), this system can be recovered at a high level of accuracy from fewer measurements m [4, 5, 6]. A highly underdetermined system usually has infinitely many solutions. However, it has been proven with high probability that when these conditions are satisfied the solution is unique and is the sparsest solution (see Theorem A.2.4).

Sparsity is defined as the number of nonzero elements of a vector. Inexact sparisty occurs when a vector has entries that are nonzero but are within a very small neighborhood of 0. A vector is called k sparse if it contains k nonzero entries. (For an inexact k-sparse vector, we consider the k largest entries, while very small entries are considered to be zero.) Finding the sparsest solution means identifying the zero and nonzero components. The sparsest solution problem is:

$$\min_{x \in \mathbb{R}^N} \|x\|_0 \text{ subject to } Ax = b, \tag{P_0}$$

where $\| \cdot \|_0$ represents the number of nonzeros of x.

(P_0) is a nonconvex problem and is not easily solved. Consequently, the l_1-minimization problem was proposed as follows:

$$min\|x\|_1 \text{ subject to } Ax = b. \tag{P_1}$$

where $\|x\|_1 = \sum\limits_i |x_i|$. The l_1-minimization problem solves for the x with the least total magnitude in hopes that the approximation coincides with actual solution. For a sufficiently sparse x and for A satisfying certain conditions, (P_0) and (P_1) have the same solution [2]. However, the l_1-minimization problem measures magnitude and not sparsity. In general, it tells us nothing about the locations of the nonzero entries. Therefore the results are not always the most accurate. In fact this method is inadequate for any x with entries that have relatively large, or small, magnitudes.

This restriction led to the formulation of the reweighted l_1-minimization problem. Instead of solving (P_1), we solve the weighted problem

$$min\|W_R x\|_1 \text{ subject to } Ax = b, \tag{P_2}$$

where $W_R = diag\{w_1, w_2, ..., w_N\} \in \mathbb{R}^{N\times N}$with $w_i = \frac{1}{|x_i|+\varepsilon}$, $i = 1, \ldots, N$ (ε being used to avoid division by 0) are weights. The idea is that W_R is nearly inversely proportional to x so that the affects of larger magnitudes are minimized [1]. By proportionality, the smallest x_i will have the largest weight while the largest x_i will have the smallest weight. This weight function also reduces the affect of individual magnitudes on the l_1 norm.

Suppose the system $Ax = b$ has a k-sparse solution x^*. Consider the case of noisy measurements $u = Ax + e$, where e is a noise vector. Then the error for the l_1 minimization has an upperboud of $\frac{2\alpha}{1-\rho} \cdot \varepsilon$, with $\rho = \frac{\sqrt{2}\delta}{1-\delta}$, $\delta \in (0, \sqrt{2} - 1)$, and $\alpha > 0$ (these values depend on the properties of A) [3]. For the reweighted minimization problem, the error has an upperbound of $\frac{2\alpha}{1+\rho} \cdot \varepsilon$ [11]. As ρ approaches its limit of 1, the upperbound for the l_1-minimization problem increases to ∞ while the reweighted problem remains bounded. Thus, the reweighted problem provides a more accurate approximation than the unweighted l_1-minimization problem because its error has a smaller upperbound.

Though the reweighted l_1-minimization problem outperforms the unweighted l_1-minimization problem, it idoes not necessarily provide the best choice of weights for very small or very large entries of x. Further improvement of the weight function is proposed by Zhu and Li [1], leading to what is known as the generalized l_1 greedy method. Let $M = \|x_0\|_\infty$ and initialize $0 \leq \alpha \leq \beta \leq 1$, $\gamma \geq 1000$, $\delta \in (0, \frac{1}{1000}]$, $s \in (0, 1]$ and $\epsilon > 0$. The generalized l_1 greedy algorithm is:

$$min\|W_G x\|_1 \text{ subject to } Ax = b, \tag{P_3}$$

where W_G is a diagonal matrix whose diagonal entries are given by

$$w_i = \begin{cases} \delta, & \text{for } |x_i| \geq \beta M s^{k-1} \\ \gamma, & \text{for } |x_i| < \alpha M s^{k-1} \\ \frac{1}{|x_i|+\epsilon}, & \text{otherwise.} \end{cases}$$

This is an improvement upon the reweighted l_1 minimization problem in that it further exaggerates the weights of smaller and larger entries of x by making those weights even larger or smaller, respectively.

Unfortunately, not all signals have sparse representations in an orthonormal basis. Is it possible to reconstruct a signal given that it does not satisfy sparsity conditions? Recoverability depends on the properties of the subspace Ω containing x. Exact recovery was originally guaranteed for any x satisfying the condition

$$\|x\|_0 \leq \frac{m}{log(\frac{N}{m})},$$

where m and N are the dimensions of projection matrix A [8, 10]. Recoverability results were extended such that exact recovery is also ensured for x satisyfing

$$\|x\|_0 \leq \frac{c_1^2}{4} \cdot \frac{m}{log(\frac{N}{m})},$$

where $c_1 > 0$) [2]. This extension implies that the measurement matrix need only to be in the null space of some matrix B, where the columns of B are Gaussian vectors with independent standard normal entries [8]. Consider an image x that can be approximately modeled to be piecewise constant (here, we let the image represent a signal with some additional properties). Then the gradient μ of x is sparse. In this context, the application of l_1-minimization to the gradient is known as total variation minimization.

For a projection matrix A and corresponding projection data b, the total variation minimization problem is

$$min\|x\|_{TV} \text{ subject to } Ax = b. \tag{P_4}$$

We enhance the recoverability of x^* by improving the selection of step size. A new problem is also presented as follows:

$$min\|x\|_{TV} + \lambda\|\mu\|_0 \text{ subject to } Ax = b, \tag{P_5}$$

where $\lambda \in (0, \frac{1}{2}]$. Unlike (P_4), (P_5) also takes into account the sparsity level of the gradient.

CHAPTER 2

TOTAL VARIATION

In this chapter we will introduce notations, definitions, and the properties of the total variation function. We will also discuss the necessity of using both forward and backward differences to calculate the gradient of an image.

2.1 Properties

Total variation (TV) is real-valued function that measures the difference, or variation, between the entries of a matrix. The TV functional is convex and is not differentiable. Convexity guarantees that the minimizer is unique. TV minimization is the application of l_1 minimization to the gradient of an image x. TV minimization is an edge preserving method for denoising images was introduced by Rudin, Osher and Fatemi in a praised 1992 paper[12].

Throughout this paper, for any vector $a \in \mathbb{R}^N$, where $N = n^2$, we let $\hat{a} = reshape(a, n, n)$. In the discrete setting, the total variation of a square image x is defined as

$$TV(x) = \|x\|_{TV} = \sum_t |\mu_t| = \sum_{i,j} \sqrt{((Dx)^1_{i,j})^2 + ((Dx)^2_{i,j})^2}, \tag{1}$$

where $(Dx)^1_{i,j} = \hat{x}_{i+1,j} - \hat{x}_{i,j}$ and $(Dx)^2_{i,j} = \hat{x}_{i,j+1} - \hat{x}_{i,j}$ and μ is the gradient of x. Here, we assume that $\hat{x}$ is piecewise constant so that the gradient is sparse. With very little changes, we can apply the iterative methods mentioned earlier to the gradient in order to solve for x.

2.2 Derivation

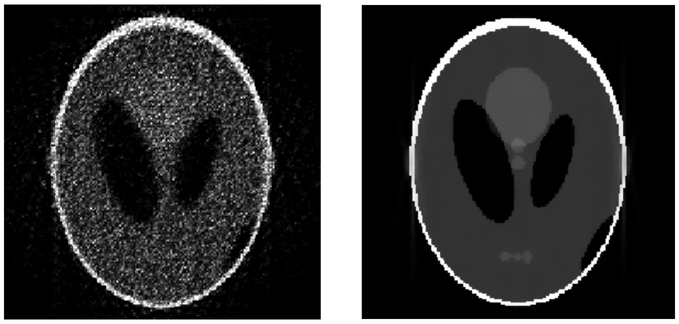
In our implementation of each algorithm, we let $(Dx)_{i,j}$ be a 4 dimensional vector of forward and backward differences (an idea proposed in [9]). That is,

$$\|x\|_{TV} = \sum_{i,j} \sqrt{\frac{1}{2}[((Dx)^1_{i,j})^2 + ((Dx)^2_{i,j})^2 + ((Dx)^3_{i,j})^2 + ((Dx)^4_{i,j})^2]}, \tag{2}$$

where $(Dx)^3_{i,j} = \hat{x}_{i-1,j} - \hat{x}_{i,j}$, $(Dx)^4_{i,j} = \hat{x}_{i,j-1} - \hat{x}_{i,j}$, and $(Dx)^1_{i,j}$, $(Dx)^2_{i,j}$ as defined before. (It is important to note that defining (Dx) in this way results in a gradient that is less sparse.)

Consider an element $\hat{\mu}^{(p)}_{i,j}$ of a gradient matrix $\hat{\mu}^{(p)}$, with p being the number of directions used to calculate the gradient (total variation). If either $\hat{x}_{i+1,j} \neq \hat{x}_{i,j}$ or $\hat{x}_{i,j+1} \neq \hat{x}_{i,j}$, then $\hat{\mu}^{(2)}_{i,j}$ is a nonzero entry. That is, $\hat{\mu}^{(2)}_{i,j} = \sqrt{((Dx)^1_{i,j})^2 + ((Dx)^2_{i,j})^2} > 0 \Rightarrow \sqrt{\frac{1}{2}[((Dx)^1_{i,j})^2 + ((Dx)^2_{i,j})^2 + ((Dx)^3_{i,j})^2 + ((Dx)^4_{i,j})^2]} = \hat{\mu}^{(4)}_{i,j} > 0$. Vice versa, suppose that $\hat{x}_{i+1,j} = \hat{x}_{i,j+1} = \hat{x}_{i,j}$ and, without loss of generality, $\hat{x}_{i-1,j} \neq \hat{x}_{i,j}$. Then $\hat{\mu}^{(4)}_{i,j} > 0$ while $\hat{\mu}^{(2)}_{i,j} = 0$. (Note that $\hat{\mu}^{(4)}_{i,j} = 0 \Rightarrow \hat{\mu}^{(2)}_{i,j} = 0$.) The calculation of total variation at a given point in two directions results in the loss of information. Any information obtained in $\hat{\mu}^{(2)}$ is also included in $\hat{\mu}^{(4)}$. That is, the nonzero entries that are located from variation measured at two directions are still recorded as nonzero entries in $\hat{\mu}^{(4)}$.

We see that information is excluded at certain boundary points of the image. Let $F_{i,j} = \{\hat{x}_{i+1,j}, \hat{x}_{i-1,j}, \hat{x}_{i,j+1}, \hat{x}_{i,j-1}\}$ be the set of entries corresponding to $\hat{x}_{i,j}$, where these entries are the surrounding elements. Any component $\hat{x}_{i,j}$ is considered a boundary point if there exists an $\hat{x}_b \in F_{i,j}$ such that $\hat{x}_b \neq \hat{x}_{i,j}$. For certain boundary points, the corresponding gradient in the 2 direction case is recorded as 0 so that there is no variation. If there is no variation at this point, then it is not necessarily a boundary.



(a) *(b)*

Figure 2.1: Visual comparision of the reconstructed Shepp Logan phantom at (a) 2 directions and (b) 4 directions via TV minimization.

Rather, the variation is recorded at the entries that surround the actual boundary point. The resulting image is less sharp and the algorithm cannot efficiently reconstruct the boundaries, nor can it distinguish which entries are causing the variation at the surrounding points. (See Figure 2.1)

Calculation of total variation at any given component that uses partial information creates an incomplete image. There is no way to explicitly connect the entries that surround the component. Information pertaining to the set of backwards differences at this component may only be obtained through the calculation of forward differences at two other seperate components.

Additionally, there is an extra emphasis placed on the variation between any consecutive entries. Therefore, if there is zero variation between a given pair of entries, then the variation is reflected as truly being zero. Likewise, if variation exist between a pair, then that variation is reflected at both entries and is twice minimized with respect to their surrounding entries.

Though using 4 directions causes an increase in the sparsity level, the total variation at each $\hat{x}_{i,j}$ is more accurately reflected. This option quickly distinguishes the boundary locations of the constant pieces in the image. In doing so, the algorithm does not waste time in determining whether or not the variation at a given point is truly 0. The quick identification of nonzero entries immediately reflects in the weight function, leading to a more accurate descent direction.

CHAPTER 3
STEP SIZE

In this chapter we will discuss the importance of step size, as well as the advantages and disadvantages of using a geometric sequence as the sequence of step sizes. We will also provide background information for strip based projection and block cyclic projection for CS (BCPCS).

3.1 Block Cyclic Projection

In the code, we use strip based projection to generate the projection matrix A with the assumption that an image will be scanned using equidistant parallel X-rays. For an n by n image, the strip based model will intersect, at most, n entries at one time. Suppose that at a certain scan direction we have a resulting matrix $A \in \mathbb{R}^{m \times N}$. Since each row of A has at most n nonzeros entries, it follows that A is highly sparse, and thus compressible. We take advantage of this by only including the nonzero entries of A and update A such that $A \in \mathbb{R}^{m \times n}$, where the entries of A record the locations of the non-zero entries of x and are zero otherwise. We then update of the approximation x by applying a cyclic orthogonal projection at each block (BCPCS).

Iterative methods are progressions that begin with an initial guess x_0 and generate a sequence of iterates such that converges to x^*. The $(k+1)$-th iterate is defined as $x_{k+1} = x_k + \tau_k d_k$, where $k \geq 0$, $x_k, d_k \in \mathbb{R}^N$ and $\tau_k \in (0, 1)$. Here, x_k is the previous iterate, d_k is the steepest descent direction of the total variation of x_k, and τ_k represents the step size. In this paper, d_k is a unit vector.

Consider a projection matrix $A \in \mathbb{R}^{m \times N}$ and the corresponding projection data

$b \in \mathbb{R}^m$, where $A = [A_1, \ldots, A_r]^T$ contains r blocks corresponding to the scanning directions of an image and $b = [b_1, \ldots, b_r]^T$. Let x^* be the solution to (1). We solve (1) for x^* using the steepest descent method. We use Algorithm 1 to solve the total variation minimization problem. Algorithm 2 solves the same problem using weights.

Algorithm 1: TV Minimization for CT

```
1  Generate $x_0$ by initial guess
2  for $k = 1$ to $k_{max}$ do
3      for $j = 1$ to $r$ do
4          update $x_k$ via BCPCS for the system $A_j x_k = b_j$
5          calculate the gradient $\mu$ of $x_k$
6          calculate the steepest descent direction $d_k$ of $\mu$
7          solve $\tau_k = \arg\min_{\tau \in (0,1)} \|x_k + \tau_k d_k\|_{TV}$ by numerical method
8          update $x_k = x_k + \tau_k d_k$
9      end
10     update $x_{k+1} = x_k$
11     exit if a stop criterion is satisfied
12 end
```

Algorithm 2: Generalized TV Minimization for CT

```
1  Generate $x_0$ by initial guess or reweighted $l_1$ minimization
2  Set $M = \|x_0\|_\infty$ and initialize $0 \le \alpha \le \beta \le 1$, $\gamma \ge 1000$, $\delta \in (0, \frac{1}{1000}]$,
   $s \in (0, 1]$ and $\epsilon > 0$
3  for $k = 1$ to $k_{max}$ do
4      for $j = 1$ to $r$ do
5          update $x_k$ via BCPCS for the system $A_j x_k = b_j$
6          calculate the gradient $\mu$ of $x_k$
7          update the diagonal matrix $W^k$ by $w_i^k = \begin{cases} \delta, & \text{for } |x_i| \ge \beta M s^k \\ \gamma, & \text{for } |x_i| < \alpha M s^k \\ \frac{1}{|x_i|+\epsilon}, & \text{otherwise.} \end{cases}$
8          calculate the steepest descent direction $d_k$ of $\mu$
9          set a reweighted direction $d_k = W^k d_k$
10         solve $\tau_k = \arg \min_{\tau \in (0,1)} \|x_k + \tau d_k\|_{TV}$ by numerical method
11         update $x_k = x_k + \tau_k d_k$
12     end
13     update $x_{k+1} = x_k$
14     exit if a stop criterion is satisfied
15 end
```

3.2 The Influence of the Step Size

We need to determine the amount of influence that the step size has on the convergence rate of this sequence of iterates. Is the task of "finding an optimal τ" one worth pursuing? For a simple test, we will reconstruct the 2D Shepp Logan phantom, a 256 $\times$ 256 image, using reweighted l_1 minimization and provide results at small perturbations in the parameter τ. Let $\{\tau_k\}$ be a geometric sequence of step sizes corresponding to the sequence of iterates $\{x_k\}$ with $\tau_0 = 0.7$ and a common ratio r. We test this sequence of step sizes as being unbounded below, bounded below by 0.001, and with a change in ratio, respectively. The algorithm performs a maximum of 85 iterations and stops if the relative error is less than *tol*.

From Table 3.1, it is obvious that step size has a very dramatic affect on the performance of the algorithm. For this simple example, and at only one value of τ_0, we have very different results that are perhaps most obvious in sparsity level and error. (Recall that we are actually seeking out the sparsest solution.) We need a general procedure that will yield acceptable results without having to worry ourselves with whether or not to bound τ or what is the best common ratio. Later in this paper, we will use numerical methods to find a suitable τ.

τ	Ratio	Iterations	Total Variation	Relative Error	Sparsity
0.7	0.9	78	1919.53	0.000963	3852
0.7 (bounded)	0.9	85	1942.17	0.001368	16128
0.7	0.7	85	2217.49	0.153821	49449

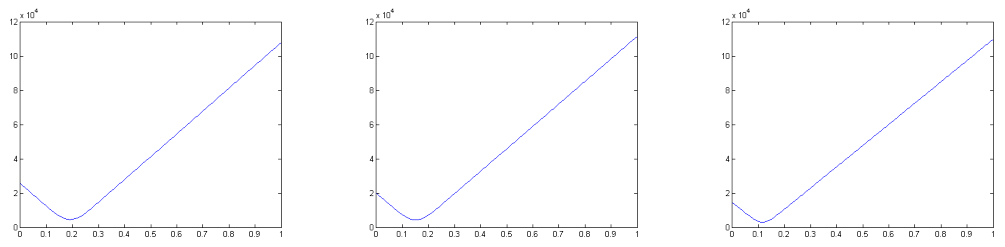
Table 3.1: We reconstruct the Shepp Logan 256 phantom with tol $= \varepsilon = 0.001$ *and a maximum of 100 iterations. (1)* τ *unbounded and* $r = 0.9$*, (2)* $\tau > \varepsilon$ *and* $r = 0.9$*, and (3)* τ *unbounded and* $r = 0.7$*. The algorithm stops once the relative error is less than our tolerance or if the algorithm reaches the maximum number of iterations.*

3.3 Disadvantages of a Geometric Sequence

We now examine the case where the sequence of step sizes is chosen to be geometric. Iterative algorithms first generate x_0 and the corresponding descent direction d_0. The geometric series then assigns a given number τ_0 as the initial step size. The idea of the geometric series is this: At the first iteration, a generous amount of descent should be applied to x_0. That is, we begin with a step size τ_0 close to 1. Successive step sizes are defined as $\tau_k = \tau_0 r^k$. Thus, the sequence $\{\tau_k\}$ is monotonically decreasing.

A geometric sequence is typically used because it guarantees the convergence of the BCPCS algorithm, which requires that $\sum\limits_k \tau_k < \infty$. While the choice of a geometric sequence has performed well, it is not necessarily conducive for quick convergence. A major flaw of a geometric sequence of step sizes is that it does not take into account the descent direction. Another issue is that this type of sequence is independent of the image itself. The geometric sequence is always fixed and has no correction capabilities.

Though it is generally the case that $\tau_{k+1} < \tau_k$, this is not necessarily true for all



(a) $\tau_k^* \approx 0.2$ *(b)* $\tau_k = 0.35$ *(c)* $\tau_k = 0.05$

Figure 3.1: Total Variation: (a) Observe the location of the minimum at τ_k^* at the current iteration, (b) TV at the next iteration if τ_k is too large, (c) TV at the next iteration if τ_k were too small

k. How do we determine the amount of descent to apply to x_k? Though the amount of descent needed may be small, it is possible that the geometric step size is not small enough. Hence, the image is still not updated to the best possible approximation. Furthermore, what happens if the need for descent slightly increases at some x_k?

Suppose we want to minimize the total variation of a certain image. Suppose we are at the $(k+1)$-th iteration and want to determine a good step size τ_k. Figure 3.1 shows the differences in potential total variation at the next iteration when $\tau_k \neq \arg\min_{\tau\in(0,1)} \|x_k + \tau_k d_k\|_{TV}$ in the absence of BCPCS. Figure 3.1a shows the τ_k^* at which the lowest possible total variation is achieved for the $(k+1)$-th iteration. Figure 3.1b shows the total variation graph at the next iteration if the step size is chosen such that $\tau_k > \tau_k^*$ while Figure 3.1c shows what happens if $\tau_k < \tau_k^*$. From this example we see that in overestimating τ_k, the total variation for x_{k+1} (given at $\tau = 0$), may not decrease by much, if at all. In fact, the graphs in Figures 3.1a and 3.1b are practically the same. In other words, $\|x_{k+1}\|_{TV} \approx \|x_k\|_{TV}$. We also observe that in the case where $\tau_k < \tau_k^*$, we have $\|x_{k+1}\|_{TV} < \|x_k\|_{TV}$. Obviously this is because d_k is a descent direction. Therefore $\|x_k + \tau_k d_k\|_{TV} < \|x_k\|_{TV}$ for all τ_k $(0, \tau_k^*]$.

It is better to underestimate τ_k than to overestimate it. Since the right side of τ_k^* represent values at which there is too much descent is applied, this side shoots up very quickly. By continuity, $\exists \varepsilon > 0$ such that $\|x_k + \tau_k d_k\|_{TV} < \|x_k\|_{TV}$ for all $\tau_k \in (\tau_k^*, \tau_k^* + \varepsilon)$. Though we acknowledge that it is possible to choose a $\tau_k \in (\tau_k^*, 1)$ such that $\|x_{k+1}\|_{TV} < \|x_k\|_{TV}$, on average this does not happen. Choosing a step size that is larger than the optimal step size does not guarantee that the total variation of the approximation will be reduced. It is by chance that Figure 3.1b does not display horrible results. On the other hand, choosing τ_k such that $0 < \tau_k < \tau_k^*$ is much safer because it guarantees that the total variation of x_{k+1} is at least smaller than that of x_k; however, it does not yield in the lowest possible total variation of x_{k+1}. Figure B.1 shows the total variation of x_{k+1} when $\tau_k \approx \tau_k^*$.

Now consider the Shepp Logan and Cardiac phantoms (see Figure B.2) and their respective gradients. These images have their own unique properties. The Shepp Logan phantom is much more piecewisely constant than the Cardiac phantom. For that reason, the gradient of the Shepp Logan phantom has a lower sparsity level than the gradient of the cardiac phantom. Therefore the approximations for the cardiac phantom will converge more slowly than that of the Shepp Logan phantom, and yet (given that τ_0 and r are the same in both methods) the same exact sequence of step sizes will be used to update both images. It is clear that this could present a problem.

We need a sequence of step sizes that is customized for each image. Beginning with some random τ_0 could definitely sabotauge the convergence rate of the algorithm since the subsequent step sizes are all dependent on the first. If τ_0 is a horrible choice for x_0, and even if it is a good choice, we have no control over the behavior of the algorithm. That is, if at some τ_k, the algorithm took a turn for the worst, we would have no way of correcting it. We could only hope that the behvior of the image eventually rebound to align itself with the step size. We need step sizes that are indepent of themselves and dependent on the behavior of the objective function.

3.4 Determining the Optimal Step Size

The total variation function (1) is not a differentiable function of x. However, when we replace x_{k+1} with $x_k + \tau d_k$, the total variation becomes a function of τ. This simple substitution results in a differentiable function of τ, which we denote as $\phi(\tau) = \|x_k + \tau d_k\|_{TV}$ for a given x and d. Since we know the values x_k and d_k, minimizing this function at the $(k+1)$-th iteration is the same as solving for the $(k+1)$-th optimal step size, where $\tau^*_{k+1} = \arg\min\limits_{\tau \in (0,1)} \phi(\tau)$.

Expanding $\phi(\tau)$ yields

$$\begin{aligned}\phi(\tau) &= \|x_{k+1}\|_{TV} = \|x_k + \tau d_k\|_{TV} \\ &= \sum_{i,j} \sqrt{[(\hat{x}_k + \tau\hat{d}_k)_{i+1,j} - (\hat{x}_k + \tau\hat{d}_k)_{i,j}]^2 + [(\hat{x}_k + \tau\hat{d}_k)_{i,j+1} - (\hat{x}_k + \tau\hat{d}_k)_{i,j}]^2}.\end{aligned}$$

Setting $(Dx)^1_{i,j} = (\hat{x}_k + \tau\hat{d}_k)_{i+1,j} - (\hat{x}_k + \tau\hat{d}_k)_{i,j}$ and $(Dx)^2_{i,j} = (\hat{x}_k + \tau\hat{d}_k)_{i,j+1} - (\hat{x}_k + \tau\hat{d}_k)_{i,j}$, we get

$$\begin{aligned}\phi'(\tau) = \frac{d\phi}{d\tau} &= \sum_{i,j} \frac{[{(Dx)^1_{i,j}}^2 + {(Dx)^2_{i,j}}^2]'}{2 \cdot \sqrt{{(Dx)^1_{i,j}}^2 + {(Dx)^2_{i,j}}^2}} \\ &= \sum_{i,j} \frac{(Dx)^1_{i,j} \cdot ((Dx)^1_{i,j})' + (Dx)^2_{i,j} \cdot ((Dx)^2_{i,j})'}{\sqrt{{(Dx)^1_{i,j}}^2 + {(Dx)^2_{i,j}}^2}} \\ &= \sum_{i,j} \frac{(Dx)^1_{i,j} \cdot (\Delta d_k)^1_{i,j} + (Dx)^2_{i,j} \cdot (\Delta d_k)^2_{i,j}}{\sqrt{{(Dx)^1_{i,j}}^2 + {(Dx)^2_{i,j}}^2}},\end{aligned}$$

where $(\Delta d_k)^1_{i,j} = (\hat{d}_k)_{i,j} - (\hat{d}_k)_{i+1,j}$ and $(\Delta d_k)^2_{i,j} = (\hat{d}_k)_{i,j} - (\hat{d}_k)_{i,j+1}$. For simplicity, we may rewrite $\phi'(\tau)$ as

$$\sum_{i,j} \frac{g(\tau)_{i,j}}{h(\tau)_{i,j}}$$

where

$$g(\tau)_{i,j} = (Dx)^1_{i,j} \cdot (\Delta d_k)^1_{i,j} + (Dx)^2_{i,j} \cdot (\Delta d_k)^2_{i,j} \tag{2}$$

and

$$h(\tau)_{i,j} = \sqrt{((Dx)^1_{i,j})^2 + ((Dx)^2_{i,j})^2}, \tag{3}$$

with $h(\tau)_{i,j} > 0$ for all τ. It is important to note that when $h(\tau)_{i,j} = 0$ this means that the corresponding gradient element $\mu_{i,j}$ is zero and this element is not included in the calculation of total variation.

Then the second derivative is calculated to be

$$\begin{aligned}
\phi''(\tau) &= \frac{d^2\phi}{d\tau^2} \\
&= \sum_{i,j} \frac{h(\tau)_{i,j} \cdot g'(\tau)_{i,j} - h'(\tau)_{i,j} \cdot g(\tau)_{i,j}}{(h(\tau)_{i,j})^2} \\
&= \sum_{i,j} \frac{h(\tau)_{i,j} \cdot [((\Delta d_k)^1_{i,j})^2 + ((\Delta d_k)^2_{i,j})^2] - \frac{g(\tau)_{i,j}}{h(\tau)_{i,j}} \cdot g(\tau)_{i,j}}{(h(\tau)_{i,j})^2} \\
&= \sum_{i,j} \frac{(h(\tau)_{i,j})^2 \cdot [((\Delta d_k)^1_{i,j})^2 + ((\Delta d_k)^2_{i,j})^2] - (g(\tau)_{i,j})^2}{(h(\tau)_{i,j})^3}
\end{aligned}$$

with $g(\tau)_{i,j}$ and $h(\tau)_{i,j}$ previously defined in (2) and (3) respectively.

Theorem 3.4.1. *Let $x_{k+1} = x_k + \tau d_k$ be the kth approximation to x^*. Then $\phi(\tau)$ concaves upward.*

Proof. Suppose that $\mu_{i,j}$ is nonzero. Fix i, j, and $\tau \in (0,1)$.

$$\begin{aligned}
&(h(\tau)_{i,j})^2 \cdot [((\Delta d_k)^1_{i,j})^2 + ((\Delta d_k)^2_{i,j})^2] - (g(\tau)_{i,j})^2 \\
&= (((Dx)^1_{i,j})^2 + ((Dx)^2_{i,j})^2) \cdot [((\Delta d_k)^1_{i,j})^2 + ((\Delta d_k)^2_{i,j})^2] - ((Dx)^1_{i,j} \cdot (\Delta d_k)^1_{i,j} + (Dx)^2_{i,j} \cdot (\Delta d_k)^2_{i,j})^2 \\
&= ((Dx)^1_{i,j}(\Delta d_k)^1_{i,j})^2 + ((Dx)^1_{i,j}(\Delta d_k)^2_{i,j})^2 + ((Dx)^2_{i,j}(\Delta d_k)^1_{i,j})^2 + ((Dx)^2_{i,j}(\Delta d_k)^2_{i,j})^2 \\
&\qquad - ((Dx)^1_{i,j}(\Delta d_k)^1_{i,j})^2 - 2(Dx)^1_{i,j}(\Delta d_k)^2_{i,j}(Dx)^2_{i,j}(\Delta d_k)^1_{i,j} - ((Dx)^2_{i,j}(\Delta d_k)^2_{i,j})^2 \\
&= ((Dx)^1_{i,j}(\Delta d_k)^2_{i,j})^2 - 2(Dx)^1_{i,j}(\Delta d_k)^2_{i,j}(Dx)^2_{i,j}(\Delta d_k)^1_{i,j} + ((Dx)^2_{i,j}(\Delta d_k)^1_{i,j})^2 \\
&= [((Dx)^1_{i,j}(\Delta d_k)^2_{i,j})^2 - ((Dx)^2_{i,j}(\Delta d_k)^1_{i,j})^2]^2
\end{aligned}$$

Therefore, $(h(\tau)_{i,j})^2 \cdot [((\Delta d_k)^1_{i,j})^2 + ((\Delta d_k)^2_{i,j})^2] - (g(\tau)_{i,j})^2 \geq 0$ for all i, j and

$$\phi''(\tau) = \sum_{i,j} \frac{(h(\tau)_{i,j})^2 \cdot [((\Delta d_k)^1_{i,j})^2 + ((\Delta d_k)^2_{i,j})^2] - (g(\tau)_{i,j})^2}{(h(\tau)_{i,j})^3} > 0$$

□

Since the total variation function is concave upward, we may now approximate the root of $\phi'(\tau)$ through the utilization of certain numerical methods.

CHAPTER 4
NUMERICAL METHODS FOR STEP SIZE

In this chapter, we use the bisection method and newton's method to approximate the optimal step size. Since the general trend in iterative methods is that $\tau_{k+1} \in N^*(\tau_k, \epsilon)$, we assume this to be true and use τ_k to calculate τ_{k+1}. We discuss the advantages and disadvantages of each method. (Please note that our step size τ is only updated at the first block at each iteration.)

4.1 Newton's Method

First, consider using Newton's method in solving for the optimal step size τ_k. Newton's method is known to converge very quickly given that the initial guess is within a reasonable "distance" of the actual solution. We use Newton's method to solve the problem $\phi'(\tau) = 0$.

Newton's method requires the computation of the second derivative. In applying Newton's Method, we update τ_k as follows:

$$\tau_k = \tau_{k-1} - \frac{\phi'(\tau_{k-1})}{\phi''(\tau_{k-1})}$$

and use Algorithm 3 to implement it.

The disadvantage of Newton's method is that the choice of initial step size τ_0 is crucial. Consider a very small $\varepsilon > 0$ and $\delta > \varepsilon$. During simulations, the graph $\phi'(\tau)$ is extremely steep for most $\tau \in N^*(\tau^*, \varepsilon)$ (see Figure 4.1). Furthermore, for τ not in $N^*(\tau^*, \delta)$, the graph is more shallow. Since Newton's method uses the tangent lines to solve for a root, it follows that a slight overestimation (or underestimation) of τ^* would cause the approximation τ_k to be negative.

Algorithm 3: Newton's Method for Step Size at kth step

```
1  Generate $\tau_{k_0}$ by initial guess or numerical method using $\tau_{k-1}$..
2  Set $t_k = i = j = l = 0$, $\tau_k = t = \tau_{k_0}$ and $t_1 = \tau_{k-1}$. Initialize $\varepsilon = 10^{-(5+\lceil g_1 * k \rceil)}$,
   $tol_1 = 0.001$ and $tol_2 = 10^{\lfloor g_2 * k \rfloor}$, with $g_1 \in [\frac{1}{20}, \frac{1}{10}]$ and $g_2 \in [\frac{1}{20}, \frac{1}{5}]$
3  If $\phi'(1) \leq 0$, return
4  while $i < 5$
5      Calculate $\frac{\phi'(\tau_k)}{\phi''(\tau_k)}$.
6      Set $t_2 = \tau_k$
7      while $\frac{\phi'(\tau_k)}{\phi''(\tau_k)} - j \geq \tau_k$ or $\frac{\phi'(\tau_k)}{\phi''(\tau_k)} + j \leq \tau_k - 1$
8          If $\frac{\phi'(\tau_k)}{\phi''(\tau_k)} \leq \tau_k$, update $t_2 = 0.8 \cdot t_2$, else update $t_2 = 1.2 \cdot t_2$.
9          Generate $\tau_k$ via bisection method on $[0, t_2]$.
10         If $\tau_k - \frac{\phi'(\tau_k)}{\phi''(\tau_k)} \in (0, 1)$, update $j = 1$ and break.
11         If $l > 3$, break, else update $l = l + 1$.
12     end
13     Update $\tau_k = \tau_k - \frac{\phi'(\tau_k)}{\phi''(\tau_k)}$.
14     if $\tau_k \in (0, 1)$
15         If $(|\phi'(\tau_k)| \leq tol_1)$ or $(|\phi'(\tau_k)| \leq tol_2$ and $i == 4)$, return.
16     else
17         If $|\phi'(t_1)| \leq tol_2$, update $\tau_k = t_1$ and return.
18         Otherwise, update $\tau_k = t_1$ and return.
19     end
20     Store $t_1 = \tau_k$ and update $i = i + 1$.
21 end
22 output $\tau_k$
```

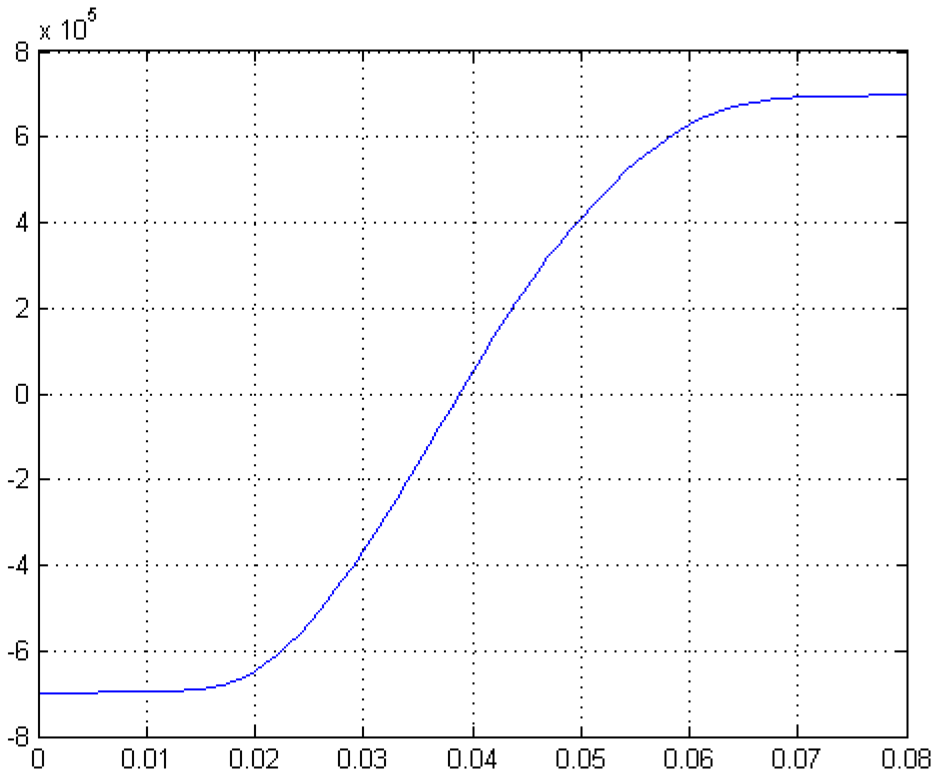


Figure 4.1: Graph of the first derivative, $\phi'(\tau)$.

In this algorithm, the tolerance tol_1 is nearly 0 while the tolerance tol_2 increases as the iterations progress. This increase in the second tolerance is to accommodate for the drastic increase in the slope at some step sizes that are within a reasonable distance of τ^*. Therefore, the slope of the current approximation for step size, τ, is checked at each iteration of Newton's method to see if it is nearly 0. Should this not be the case, then the slope at τ is checked at the last iteration of Newton's method to see if it is within a reasonable distance of τ^*. In either case, choose $\tau_k = \tau$. Otherwise, either the previous approximation is used or the initial guess (which, in our case, is calculated using the bisection method).

Observe Figure 4.2. For a fixed τ such that $\phi'''(\tau) = 0$, $|\tau^* - \tau|$ is arbitrarily

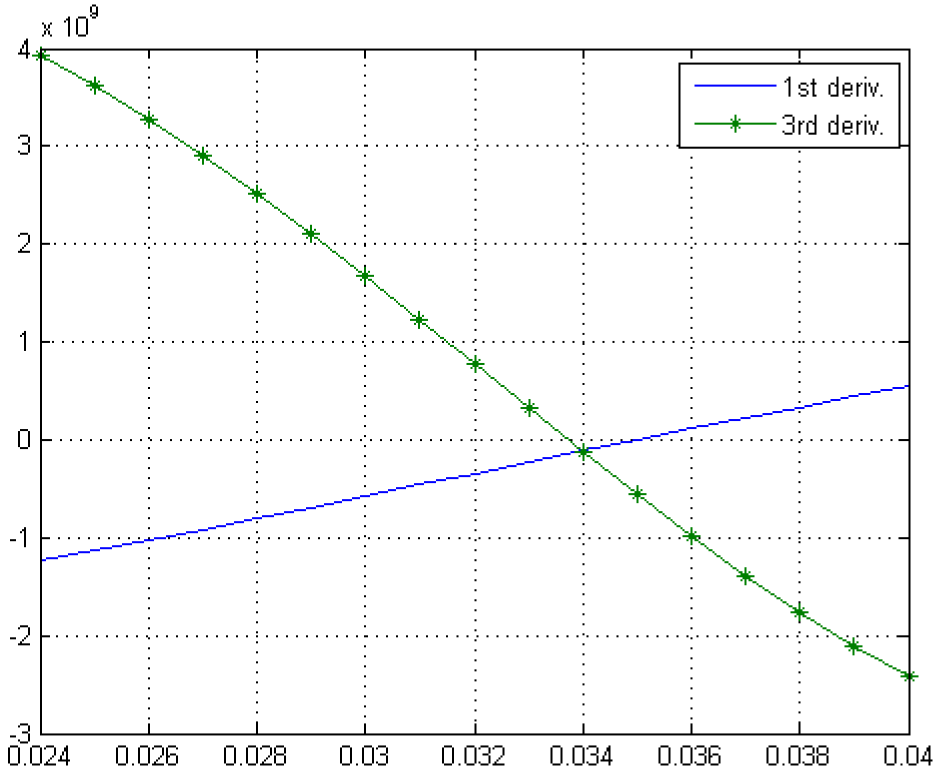


Figure 4.2: (a) Observe the location of the root, $\tau^* \approx 0.0338$*, of the first derivative and the root,* $\tau_3 \approx 0.0349$*, third derivative. (Note: the first derivative graph is stretched for better observations.)*

small. (The central difference formula $\phi'''(\tau) \approx \frac{\phi'(\tau+h)-2\phi'(\tau)+\phi'(\tau-h)}{h^2}, h > 0$, along with observations, are used to confirm this statement.) This causes the method to diverge if τ_0 is not already a highly accurate approximation of τ^*. Therefore, it is almost useless to put in the extra effort to generate a suitable initial step size. Thus, we see that a general application of Newton's method yields unfavorable results.

4.2 Bisection Method

The bisection method is ideal for finding an optimal step size. Since step size τ is restricted to values between 0 and 1, the bisection method is able to find τ relatively quickly. The bisection method only requires the computation of the first derivative. For $\phi(\tau)$ defined on any interval $[a, b] \subset [0, 1]$, let $\{[a_k, b_k]\}$ be the sequence of intervals determined by the bisection method. If a minimum exists, then continuity ensures that $\forall \varepsilon > 0, \exists \delta > 0$ such that $|\phi'(\tau)| < \varepsilon$ whenever $b_k - a_k < \delta$ for some k. The bisection method solves the problem for optimal step size with high accuracy, where the choice of $\delta \in (0, 10^{-k/M}]$ (for some $M \in \mathbb{Z}$) directly influences the accuracy of selection.

In general, the step size in bisection method is updated as

$$\tau_k = a_k$$

where $[a_k, b_k]$ is the interval containing τ_k^* and satisfies the condition $b_k - a_k < \delta$. By defining τ_k as the left endpoint, we have that$\|x_{k+1}\|_{TV} < \|x_k\|_{TV}$. This is a reasonably accurate approximation of τ_k^*. It is obvious that τ_k most likely underestimates τ_k^*; however, the narrowness of the interval, along with the guarantee that the minimizer is contained within this interval, ensures that we choose a step size that is within an acceptable distance of the optimal step size.

The bisection method uses two instances in solving for step size. The first is used for an approximation of the initial step size τ_0, and the second for all subsequent τ_k. We call this method as follows:

$$\tau_k = \begin{cases} bisection(\hat{x_k}, \hat{d}_k, \hat{W}^k, 1), & \text{for } k = 0 \\ bisection(\hat{x_k}, \hat{d}_k, \hat{W}^k, \tau_{k-1}), & \text{otherwise} \end{cases}$$

and it is executed as shown in Algorithm 4 on a given interval $[a, b]$.

Algorithm 4: Bisection Method for Step Size at kth step

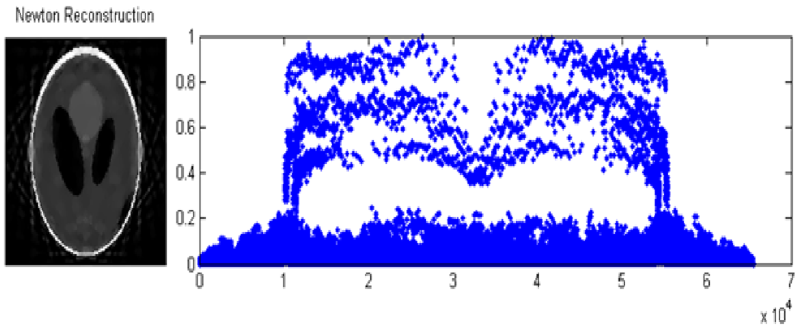
```
1  Set $a = 0$ and $b = \tau_{k-1}$. Initialize $\delta = 10^{\lceil g_1 * k \rceil}$ and $tol = 10^{\lfloor g_2 * k \rfloor}$, with
   $g_1 \in [\frac{1}{20}, \frac{1}{10}]$ and $g_2 \in [\frac{1}{20}, \frac{1}{5}]$.
2  If $|\phi'(b)| \leq tol$, update $\tau_k = b$ and return.
3  If $\phi'(1) < 0$, update $\tau_k = \tau_{k-1}$ and return.
4  while $b - a > \delta$
5      $c = \frac{a+b}{2}$
6      If $|\phi'(c)| \leq tol$, update $\tau_k = c$ and return.
7      If $\phi'(a)$ and $\phi'(c)$ have the same sign, update $a = c$.
8      If $\phi'(b)$ and $\phi'(c)$ have the same sign, update $b = c$.
9      If $a \neq 0$, update $\tau_k = a$. Otherwise, update $\tau_k = b$.
10 end
11 output $\tau_k$
```

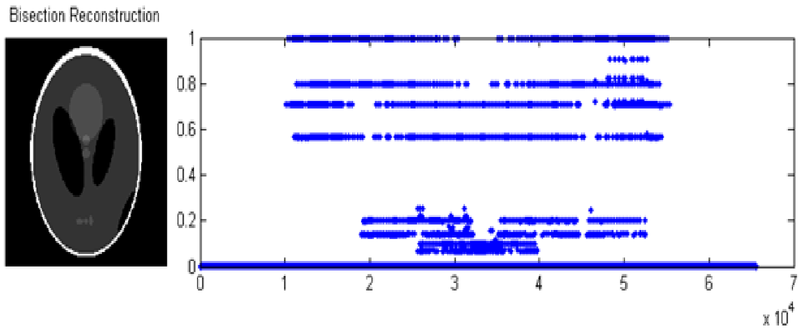
This algorithm searches for τ_0 on the interval $(0, 1)$. In subsequent iterations, the algorithm computes τ_k using the interval $(0, \tau_{k-1}]$, again with the assumption that $\tau_k < \tau_{k-1}$. In the case that the minimum is not contained within this interval (this is not usually the case), the algorithm searches interval $[\tau_{k-1}, 1]$.

If $(0, 1)$ contains no minimum, then $\phi(\tau)$ is monotonically decreasing on the interval and the algorithm returns a default step size according to the behavior of the function. At the *kth* step, $\tau_k = \tau_{k-1}$ is chosen for a default step size. (This number was chosen after many numerical tests and there is currently no formal proof to show that it is a good choice.) If the condition $\phi'(0) < 0 < \phi'(1)$ is satisfied, then the bisection method is executed.

4.3 Comparision Results

This section is used to analyze the performance of both methods. Each method has its own set backs. In general, however, both methods yield better results than a geometric sequence and eliminate the need to determine a τ_0. The results shown are for the 256×256 Shepp Logan Image, whose representation has a sparsity level of 3736 and total variation 1905.6. The geometric sequence of step sizes is generated using $\tau_0 = 0.7$ and ratio $r = 0.9$.





It is obvious that applying a weight function negatively affects the performance of Newton's method. However, we see that the bisection method yields far better results than the geometric method in error, total variation and computational time. Though the results for the geometric sequence are good, these results are for a specific τ_0 and

Comparison						
Algorithm	Method	Total Variation	Rel. Error	Sparsity	Iterations	CPU Time
TV Min.	Geometric	2060.86	0.109178	10582	100	106.546 s
	Bisection	1861.41	0.070389	7628	100	130.743 s
	Newton's	1858.76	0.040523	6004	100	133.153 s
GTV	Geometric	1920.58	0.001092	3736	72/100	80.007 s
	Bisection	1908.36	0.000998	3736	57/85	66.993 s
	Newton's	2723.24	0.261017	30648	72/100	98.523 s

Table 4.1: We compare the performance of the geometric sequence to that of the bisection and newton methods. Conditions:100 max iterations and relative error $\frac{\|x_k - x^*\|}{\|x^*\|} < tol$*, where* $tol = 0.001$.

ratio. The results for the bisection method are, as mentioned before, independent of an initial guess. Table 4.1 gives the results for the two methods. The main disadvantage of employing numerical methods is the increase the number of function calls. The bisection method is heavily dependent on the interval $[a_k, b_k]$. At each iteration, the choice of step size is *at most* accurate to the decimal place of δ. This presents two problems. The first is that accuracy of τ is limited. The second is that the sequence of τ_k is, in a sense, bounded. Though there are some disadvantages, these are insignificant in comparison to the benefits.

CHAPTER 5
A NEWLY PROPOSED PROBLEM

In each of the minimization problems, the primary focus is to minimize the total variation of x, that is, the l_1-norm of the gradient of x. The development of the weight functions of the reweighted and generalized l_1 greedy problems are based upon the magnitudes of the components of gradient. Though these methods are effective, it is important that we keep in mind that our ultimate goal in to minimize the sparsity of the gradient rather than the sum of the magnitudes of its "weighted" components.

It was mentioned earlier that (P_0) is a nonconvex problem. Consider the generated sequence of approximations to the matrix representation x of a piecewise constant image. As stated earlier, we are guaranteed that a unique solution exists. However, the earlier approximations of this sequence are not as piecewisely constant as the actual image. Therefore, the l_0-minimization problem cannot be solved directly. However, if both the total variation and the sparsity level are considered, then we improve the approximation x_k at the kth step. Therefore consider the problem

$$min\|W_1 x\|_{TV} + \lambda\|W_2\mu\|_0 \text{ subject to } Ax = b, \tag{5.1}$$

where $\lambda \in (0, \frac{1}{2}]$, and use this problem to determine an optimal step size. That is, we solve the usual weighted problem and use a step size, τ_k, such that

$$\tau_k = \arg\min_{\tau\in(0,1)} \|W_1(x_k + \tau d_k)\|_{TV} + \lambda\|W_2\mu\|_0$$

where μ is the gradient corresponding to $x_k + \tau d_k$.

A CT scan will obtain a large amount of data from the given object. For this reason, the earlier gradients usually have a very large sparsity level. It would be unwise to use the full measure of $\|\mu\|_0$ for reasons mentioned earlier. Therefore, only a small percentage of the sparsity level of the gradient is considered.

The l_0 norm is implemented approximately. For sufficiently small $\varepsilon > 0$, $\|\mu\|_0 \approx \sum\limits_{i,j} \frac{\phi_{i,j}(\tau)}{\phi_{i,j}(\tau)+\varepsilon}$. Therefore, we implement (5.1) by minimizing

$$f(\tau) = \phi(\tau) + \lambda \cdot \rho(\tau) \tag{5.2}$$

over the interval $(0, 1)$, where $\rho(\tau) = \sum\limits_{i,j} \frac{\phi_{i,j}(\tau)}{\phi_{i,j}(\tau)+\varepsilon}$. It is obvious that the function $f(\tau)$ is both continuous and differentiable so that numerical methods are still employed in finding an optimal step size.

Determining a reasonable weight for λ can be somewhat difficult. After several tests, $\lambda = 0.1$ is chosen to reconstruct the two 256×256 phantoms. It is important to note that the choice of lambda is also dependent on the size of the image. A larger image may require a different choice for λ. Though there may take some time to choose a "good" weight, the efficiency of the algorithm improves noticably.

CHAPTER 6
NUMERICAL RESULTS

We test the new minimization problem and reconstruct two phantoms: Shepp Logan (sparsity level: 3736, total variation: 1905.6) and Cardiac Phantom (sparsity level: 14392, total variation: 2698.2). The Shepp Logan phantom has 5.7 percent sparsity while the Cardiac phantom has 21.9 percent sparsity. (Please note that images that are more piecewisely constant converge at a faster rate and the produce approximations that are more accurate.) The sparsity level is measured differently for the two phantoms due to the extremely high sparsity of the Shepp Logan phantom. The results are given.

In each of the tables below, GTV is used to reconstruct the given image. Newton's method is used to implement the new minimization problem, with a choice of $\lambda = 0.1$. The initial guess x_0 is generated via reweighted minimization. We use $\tau_0 = 0.7$ with ratio $r = 0.9$ to produce the geometric sequence of step sizes. Each initial guess for Newton's method is generated by the bisection method.

GTV for Shepp Logan Phantom				
	Total Variation	Sparsity	Relative Error	Iterations
Geometric	1820.58	3876	0.001092	72
Bisection	1908.36	3838	0.000998	57
Newton's	2723.24	53662	0.261017	72
New Problem	1916.79	3831	0.000937	52

Table 6.1: System Size: 26002×65536*. The results shown are for* $W_1 = W_G$ *and* $W_2 = I_n$*. Sparsity level is measured using* $\varepsilon = 0.001$*.*

It is obvious that our new problem most closely reflects the actual total variation and sparsity level, and also has the smallest error and number of iterations.

GTV for Cardiac Phantom					
	Total Variation	Sparsity	Relative Error	Iterations	CPU Time
Geometric	2803.87	18226	0.024044	72	112.266 s
Bisection	2730.63	14691	0.012221	72	125.133 s
Newton's	2836.08	23350	0.055021	72	135.539 s
New Problem	2726.15	14551	0.010687	72	155.233 s

Table 6.2: System Size: 39254×65536*. Sparsity level is measured using* $\varepsilon = 0.01$*.*

The results shown are for $W_1 = W_G$ and $W_2 = \begin{cases} \frac{1}{\mu^2+\varepsilon}, & \text{if } \mu < 0.7^k \|\mu\|_0 \\ 0.001, & \text{otherwise.} \end{cases}$. This choice of W_2 is motivated by the idea the components that are closest to 0 are the ones that will likely influence the sparsity level. Therefore, the optimal step size should be chosen so that these components are minimized.

The improvements of the new minimization problem are clear. There is a slight disadvantage of an increase in computational time. The idea, however, is new and may surely be modified so that it is more efficient. The results provided are from implementation via MATLAB with a processing speed of 2.2 GHz.

CHAPTER 7
FUTURE WORK

Future work includes improving the choice of λ for the newly proposed problem, developing a better weight function for the total variation problem, determining the best order of scan directions, testing more phantoms and of course improving the algorithms used to implement each numerical method. Since the total variation graph appears to be roughly symmetric within a small neighboorhood centered at τ^*, it may also be beneficial to prove that this is approximately the case so that we may obtain a more accurate optimal step size without necessarily having to use numerical methods.

Choosing a good scan direction is very important in the implementation of each algorithm. The numerical methods mentioned only select a step size τ at the first block. Since the system of equations obtained at the first block are directly related to the scan direction, it is important that this system yields a step size that is suitable in updating x for the remaining directions. Aside from choosing a "good" first direction, the order of the scan directions is equally important. We seek to optimize the efficiency of the algorithm at each step. It is well know that each pair of directions should be orthogonal to one another. However, we should determine how each of these pairs should be coupled so that more components of x are determined, and at a faster rate.

BIBLIOGRAPHY

[1] Jiehua Zhu and Xiezhang Li, *A generalized L1 greedy algorithm for image reconstruction in CT*, Applied Mathematics and Computation 219 (2013) 5487-5494

[2] Yin Zhang, *On Theory of Compressive Sensing via L1 Minimization: Simple Derivations and Extensions*, CAAM Technical Report TR08-11. July 2008 (2008)

[3] D. Needell *Noisy Signal Recovery via Iterative Reweighted L1-Minimization*, Asilomar SSC, University of California, 2009.

[4] E.J. Candes, J. Romber, T. Tao, *Robust Uncertainty Principles: Exact Signal Recovery from Highly Incomplete Frequency Information*, IEEE Trans. Inf. Theory 52 (2006) 489-509

[5] E.J. Candes, M.B.Wakin, *An Introduction to Compressive Sampling*, IEEE Signal Process. Mag. 25 (2008) 21-30

[6] D. Dnoho *Compressed Sensing*, IEEE Trans. Inf. Theory 52 (2006) 1289-1306

[7] E. Candes, M. Wakin, S. Boyd *Enhancing Sparsity by Reweighted L1 Minimization*, October 2007.

[8] E. Candes, T. Tao *Near Optimal Signal Recovery from Random Projections: Universal Encoding Stategies*, IEEE Transactions of Information Theory, 52 (2006)

[9] X. Li, J. Zhu *Convergence of Block Cyclic Projection and Cimmino algorithms for Compressed Sensing based Tomography*, Journal of X-Ray Science and Technology 18 (2010) 369-379

[10] E. Candes, T. Tao *Decoding by Linear Programming*, IEEE Transactions on Information Theory, 51(2005)

[11] E. J. Candes *The Restricted Isometry Property and It's Implications for Compressed Sensing*, Applied and Computational Mathematics, California Institute of Technology,2008.

[12] L. Rudin, S. Osher, E. Fatemi *Nonlinear total variation based noise removal algorithms* Physica D, 60 (1992) 259-268

Appendix A

APPENDIX

Appendix includes definitions, lemmas, and theorems used to support statements and ideas proposed in this paper.

A.1 Definitions

Definition A.1.1. (Isometry and Sparsity) *For each $s \in \mathbb{Z}^+$, define the isometry constant δ_s of a matrix A as the smallest number such that*

$$(1 - \delta_s)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_s)\|x\|_2^2$$

holds for all s-sparse vectors.

A.2 Lemmas and Theorems

Theorem A.2.1. (Noiseless Recovery) *Assume that $\delta_{2s} < \sqrt{2} - 1$. The the solution x^* to (P_1) satisfies*

$$\|x^* - x\|_1 \leq C_0 \|x - x_s\|_1 \tag{A.1}$$

and

$$\|x^* - x\|_2 \leq C_0 \frac{\|x - x_s\|_1}{\sqrt{s}} \tag{A.2}$$

where $C_0 = \frac{2(1+\rho)}{1-\rho}$ with $\rho = \frac{\sqrt{2}\delta_{2s}}{1-\delta_{2s}}$.

Theorem A.2.2. (Noisy Recovery) *Assume that A satisfies RIP and $\delta_{2s} < \sqrt{2} - 1$ and $\|n\|_2 \leq \epsilon$, where n represents noise. The the solution x^* to (P_1) satisfies*

$$\|x^* - x\|_2 \leq C_0 \frac{\|x - x_s\|_1}{\sqrt{s}} + C_1 \epsilon \tag{A.3}$$

where C_0 is defined as before and $C_1 = \frac{2\alpha}{1-\rho}$ with $\alpha = \frac{2\sqrt{1+\delta_{2s}}}{\sqrt{1-\delta_{2s}}}$ and ρ defined as before.

Theorem A.2.3. (Sparse Recovery) *Assume that A satisfies RIP and $\delta_{2s} < \sqrt{2} - 1$. Let x^* be an s-sparse vector with noisy measurements $u = Ax^* + e$, where noise n satisfies $\|n\|_2 \leq \epsilon$. Assume that the smallest nonzero coordinate x_i satisfies $x_i \geq \frac{4\alpha\epsilon}{1-\rho}$. Then the solution x^* from reweighted l1 minimization satisfies*

$$\|x^* - x\|_2 \leq C_2\epsilon \tag{A.4}$$

where $C_2 = \frac{2\alpha}{1+\rho}$ and ρ and α are defined as before.

Theorem A.2.4. (Extension of CS Results) *Let $m \ll n$, $\Omega \subset \mathbb{R}^n$, and $A \in \mathbb{R}^n$ be standard normal or any rank-m matrix such that $BA^T = 0$ where $B \in \mathbb{R}^{(n-m)\times n}$ is standard normal. Then with probability greater than $1 - e^{-c_0(n-m)}$, then (P_0) and (P_1) have the same solution at $x \in \Omega$ if the sparsity of x satisfies*

$$\|x\|_0 \leq \frac{c_1^2}{4} \cdot \frac{m}{log(\frac{n}{m})} \tag{A.5}$$

where $c_0, c_1 > 0$ are some absolute constants independent of the dimensions m and n.

Appendix B

SECOND APPENDIX

Appendix includes figures, graphs and tables to support numerical results.

B.1 Figures

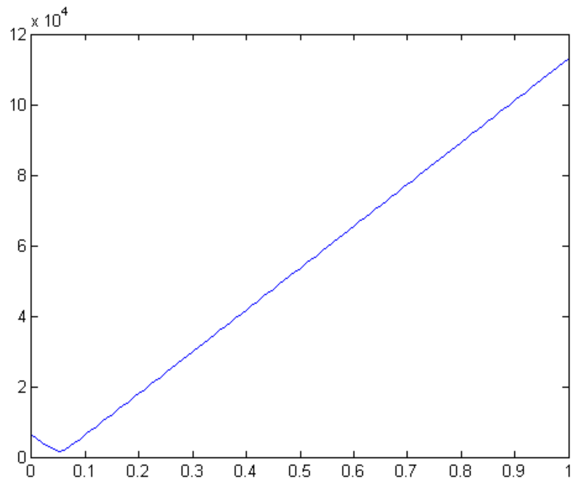


Figure B.1: TV graph at $(k+1)th$ iteration with $\tau_k \in N(\tau_k^*, \epsilon)$

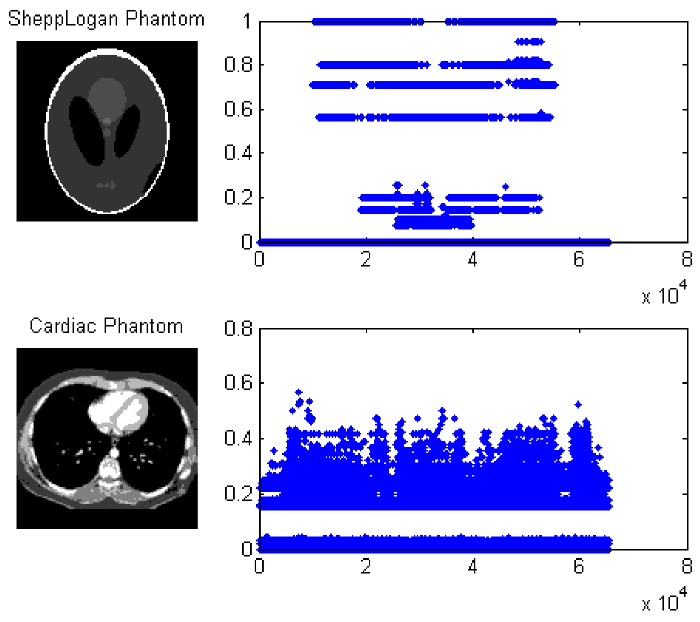


Figure B.2: Images and their gradients. Top - Shepp Logan Phantom, Bottom - Cardiac Phantom

Appendix C
MATLAB CODING

Appendix includes the MATLAB coding used to achieve the results in this paper.

C.1 Bisection Method

Bisection Method

```
function tau = bisection(f,d,t,w,k,flag, lambda)

a = 0; b = t;

tau = t;

m = ceil(k/17);

i=0;

if m > 5

    m = 5;

end

eps = 10^(-m);

p = ceil(k/20); tol2 = 10^(p);

if tol2 > 10000

    tol2 = 10000;

end

tt = der(f,d,1,w,flag,lambda); as = der(f,d,0,w,flag,lambda);

if tt < 0

    disp('completely decreasing on (0,1)');

    eps = 1; tau = 0.9;

elseif as > 0

    disp('Error: TV has no descent.');
    eps = 1;
end
while b-a > eps
    aa = der(f,d,a,w,flag,lambda);
    bb = der(f,d,b,w,flag,lambda);
    sa = sign(aa);
    sb = sign(bb);
    if sa == sb
        if sb < 0
            a = b; b = b + 0.1;
        else
            disp('Error: TV is constant.');
            break;
        end
    elseif abs(aa)<= tol2 || abs(bb) <= tol2
        if abs(bb) < abs(aa)
            tau = b;
            disp('tau: choose b.');
        else
            tau = a;
            disp('tau: choose a.');
        end
        break;
    else
        c = (a+b)/2;
        cc = der(f,d,c,w,flag,lambda);
        sc = sign(cc);
        if abs(cc) <= tol2
            tau = c;
            disp('tau: choose c.');
            break;
        end
        if sc == sb
            b = c;
        else
            a = c;
        end
        if a~=0
            tau = a;
        else
            tau = b;
        end
    end
    if i > 20
        break;
    end
    i=i+1;
end
end
```

derivative function (used in bisection method)

```
function value = der(f,d,c,w,flag,lambda)
[m, n] = size(f);
```

```
f = f-c*d;

eps=0.001;


% Add edging to F

F= [f(1,1) f(1,:) f(1,n); f(:,1) f f(:,n); f(m,1) f(m,:) f(m,n)];


% Add edging to D

D= [d(1,1) d(1,:) d(1,n); d(:,1) d d(:,n); d(m,1) d(m,:) d(m,n)];


% Calculate Forward and Backward Differences for F

Dxhf = F(2:m+1,3:n+2) - F(2:m+1,2:n+1);

Dxhb = F(2:m+1,1:n) - F(2:m+1,2:n+1);

Dxvf = F(3:m+2,2:n+1) - F(2:m+1,2:n+1);

Dxvb = F(1:m,2:n+1) - F(2:m+1,2:n+1);


% Calculate Forward and Backward Differences for D

Dhf = D(2:m+1,2:n+1) - D(2:m+1,3:n+2);

Dhb = D(2:m+1,2:n+1) - D(2:m+1,1:n);

Dvf = D(2:m+1,2:n+1) - D(3:m+2,2:n+1);

Dvb = D(2:m+1,2:n+1) - D(1:m,2:n+1);


% Perform Newton's Step (Calculate Derivatives)

num = Dxhf .* Dhf + Dxhb .* Dhb + Dxvf .* Dvf + Dxvb .* Dvb;

s = Dxhf.^2 + Dxhb.^2 + Dxvf.^2 + Dxvb.^2;

s1=sqrt(0.5.*s);

w(s==0) = 0;

s(s==0) = 1;
```

```
c = sqrt(2*s);
deriv =(num./c);
if flag == 0
    spars=zeros(n);
else
    a=lambda*eps.*deriv; E=eps.*ones(n);
    b=(s1 + E).^2;
    spars=a./b;
end

dtv = w.*(num./c) + spars;
value = sum(sum(dtv));
end
```

C.2 Newton's Method

Newton's Method

```
function tau = mintv(tau,f,d,w,k,flag,lambda)
t1 = tau; tau_k = tau;
[m,n]=size(f);
m=floor(k/17);
if m > 5
    m=5;
end
eps = 10^(-(5+m)); W=ones(n);
```

```
tau = bisection(f,d,tau,W,k,flag,lambda);
t = tau; % Store step size from bisection
F = f-tau*d; % Previous image
tol1 = 0.001;
p = floor(k/20); tol2 = 10^(p);
i = 0; l=0; j=0; t3 = 0;
tt = der(f,d,1,W,flag,lambda); as = der(f,d,0,w,flag,lambda);
if tt < 0
    disp('completely decreasing on (0,1)');
    i = 5; % tau = 0.9;
elseif as > 0
    disp('Error: TV has no descent.');
    i = 5;
end

while i < 5
    temp = TVD(F,d,W,2,k,flag,lambda);
    t2=tau;
    while temp - j >= tau || temp + j <= (tau - 1) % Negative Fix
        if temp >= tau
            flg=1;
        else
            flg=2;
        end
        if l > 3
            disp('broken');
            break;
        end
        tau=bisection(f,d,t2,W,k,flag,lambda);
        if flg == 1
            t2=0.8*t2;
        else
            t2=1.2*t2;
        end
        F=f-tau*d;
        temp=TVD(F,d,W,2,k,flag,lambda);
        if (temp < tau && flg==1) || (temp > tau-1 && flg == 2)
             j = 1;
             break;
        end
        l=l+1;
    end
    tau = tau-temp;
    F = f-tau*d;
    if tau > 0 && tau < 1
        st = der(f,d,tau,W,flag,lambda);
        if st <=0
            if abs(tau-t1) <= eps || i == 4
                break;
            end
        elseif abs(st)<=tol1 || (abs(st)<=tol2 && i == 4)
            break;
        elseif i == 4
            if abs(der(f,d,t1,W,flag,lambda)) <= tol2
```

```
                tau=t1;
            else
                tau=t;
            end
        end
    else
        if tau < 0
            disp('tau is negative');
        else
            disp('tau too big');
        end
        if abs(der(f,d,t1,W,flag,lambda)) <= tol2
            disp('newton');
            tau=t1;
        else
            disp('bisection');
            tau=t;
        end
        break;
    end
    if tau < tau_k
        t3 = tau;
    end
    t1 = tau;
    i = i + 1;
end
end
```

newton step (used in Newton's Method)

```
function [deriv, fstdrv, scnddrv] = TVD(f,d,w,flag,k,flag1,lambda)

[m,n] = size(f);

eps = 0.0001;


% Add edging to F

F= [f(1,1) f(1,:) f(1,n); f(:,1) f f(:,n); f(m,1) f(m,:) f(m,n)];


% Add edging to D

D= [d(1,1) d(1,:) d(1,n); d(:,1) d d(:,n); d(m,1) d(m,:) d(m,n)];


% Calculate Gradient and Corresponding Weights

if flag == 0

    Mu = (F(3:m+2,2:n+1) - F(2:m+1,2:n+1)).^2 +...

         (F(1:m,2:n+1) - F(2:m+1,2:n+1)).^2 +...

         (F(2:m+1,3:n+2) - F(2:m+1,2:n+1)).^2 +...

         (F(2:m+1,1:n)  - F(2:m+1,2:n+1)).^2;

    Mu = sqrt(Mu/2); mx = 0.7^k*max(max(Mu)); Mu(Mu==0) = 10000;

    W = w./Mu; W(Mu > mx) = eps;

elseif flag == 1

    W=w;

else

    W=ones(m,n);

end
```

```
% Calculate Forward and Backward Differences for F
Dxhf = F(2:m+1,3:n+2) - F(2:m+1,2:n+1);
Dxhb = F(2:m+1,1:n) - F(2:m+1,2:n+1);
Dxvf = F(3:m+2,2:n+1) - F(2:m+1,2:n+1);
Dxvb = F(1:m,2:n+1) - F(2:m+1,2:n+1);

% Calculate Forward and Backward Differences for D
Dhf = D(2:m+1,2:n+1) - D(2:m+1,3:n+2);
Dhb = D(2:m+1,2:n+1) - D(2:m+1,1:n);
Dvf = D(2:m+1,2:n+1) - D(3:m+2,2:n+1);
Dvb = D(2:m+1,2:n+1) - D(1:m,2:n+1);
% display(sum(sum(Dhf))); stop


% Perform Newton's Step (Calculate Derivatives)
s = Dxhf.^2 + Dxhb.^2 + Dxvf.^2 + Dxvb.^2; s1=sqrt(0.5.*s);
W(s==0) = 0;
s(s==0) = 1;
num = Dxhf .* Dhf + Dxhb .* Dhb + Dxvf .* Dvf + Dxvb .* Dvb;
num1 = s.*(Dhf.^2 + Dhb.^2 + Dvf.^2 + Dvb.^2) - num.^2;
c = sqrt(2*s);
deriv =(num./c);
deriv2 = sqrt(1/2).*W.*num1./(s.^(3/2));
if flag1 == 0
   spars=zeros(n); spars2=zeros(n);
else
   E=eps.*ones(n); nmr = s1 + E;
```

```
    a=lambda*eps.*deriv; b=nmr.^2;
    a1=lambda*(eps.*b.*deriv2-2*eps.*(deriv.^2).*nmr);
     b1=nmr.^4;
    spars=a./b;
    spars2 = a1./b1;
end
dtv = W.*deriv + spars;
dtv2 = W.*deriv2 + spars2;
fstdrv = sum(sum(dtv));
scnddrv = sum(sum(deriv2));

if scnddrv > 0
    deriv = fstdrv/scnddrv;
elseif scnddrv < 0
    deriv = fstdrv/scnddrv;
    j=1;
else
    if scnddrv == 0
        j = 2;
    end
    deriv = 0;
    return;
end
end
```