A COMPARISON OF THE LEVENBERG-MARQUARDT METHOD WITH STANDARD OPTIMIZATION ALGORITHMS, IN MINIMIZING THE TIKHONOV-TOTAL VARIATIONAL FUNCTIONAL

KEVIN LATOURETTE
PROGRAM IN APPLIED MATHEMATICS, UNIVERSITY OF ARIZONA
TUCSON, AZ USA 85721
MAY 16, 2008

Abstract. The Levenberg-Marquardt method is a very powerful iterative method for solving minimization problems, and is used widely in many high level optimization packages, including Matlab and Python as the solver for medium sized problems. Unfortunately, few of those who use this method actually have an understanding of how or why it works. In this paper I will introduce a variant of the Levenberg-Marquardt, which can be thought of as an adaptive combination of the method of Steepest Descent and Newton's method, and confirm that the method converges super-linearly. This paper will explore the Levenberg-Marquardt method through minimization of the Tikhonov-Total Variational functional, a regularization scheme used frequently in image processing, and will also detail the algorithm. A proof that the Levenberg-Marquardt method has a locally quadratic convergence rate is presented as well.

Contents

1. Introduction

The Levenberg-Marquardt method which I present in this paper establishes acceptable qualitative and quantitative results in minimizing the Tikhonov-Total Variational functional, and improves upon other well-known optimization techniques. This method is widely used in high level optimization packages, but is equally unknown by many who make use of the algorithm's power [5]. I will present here a comparison of the Levenberg-Marquardt method with more standard and well-known algorithms, including Steepest Descent, Newton's Method and the method of Conjugate Gradients. The purpose for conducting such an analysis arises as the magnitude of problems that are asked in modern mathematics and it's applications grow larger and larger, and tolerable differences on smaller scale problems between methods may grow to large differences, both in error and computational time.

This need is precisely why it is important to conduct experiments to highlight the advantages of one method over others, and the basic purpose of this paper is to clearly illustrate that the Levenberg-Marquardt method is superior to more common methods. Further, this paper introduces the reader to Tikhonov Regularization, implementing a Total Variational penalty functional with applications to a one-dimensional signal reconstruction, and two-dimensional image reconstruction.

1.1. **Motivation.**

My interest in this problem stems from Medical Imaging applications, specifically Magnetic Resonance Imaging (*MRI*). In medical image reconstruction, it is often difficult to accurately display defined boundaries between objects, while clearly it is beneficial to be able to differentiate between the tissue of a liver and the surrounding tissue, for example. We take interest in the Tikhonov-Total Variational functional then, because it has been shown to restore discontinuities in an image [1], and preserves said boundaries.

In minimizing the Tikhonov-Total Variational functional, one wants to find an algorithm which does so quickly, efficiently, and consistently due to the size of the problem being posed (*equivalently, the size and quality of the image being reconstructed*). Many standard algorithms provide limitations[1] which I propose that the Levenberg-Marquardt method can overcome.

In what follows, I present first a discussion of Generalized Tikhonov Regularization in Section 1.2.1 and introduce the Total Variational penalty functional in Section 1.2.2. After, I will comment on several common optimization techniques, Section 2, and describe a variation of the Levenberg-Marquardt method in Section 2.3. Finally, a presentation of the 1- & 2-dimensional numerical results between the various algorithms is given in Section 3.

[1]Newton's Method, for example, will converge quadratically to the correct solution given that it starts sufficiently close, but may diverge otherwise.

1.2. **Background.** Before one fully realizes the need for a regularization of any sort, first we must understand the basic set-up. The general 'forward' problem to be solved in image processing is $d = Kf$, where d represents a prediction of observable parameters corresponding to a known physical model f, and the operator K is dependent on the physics of our problem (*often non-linear*) [8]. Solving this sort of system is essentially what is done in many introductory level classes... if I toss a ball in the air at an angle with some initial velocity, I can predict where it will land. In image processing, we would view the forward problem as taking a clean image, and adding noise to obtain d.

What takes a different sort of thinking now, is to solve the *inverse* problem. Lets say that now I know where the ball landed, the angle of impact and velocity at impact, and I know how a ball interacts as it moves through its medium (*air*), but I wish to determine where it was thrown from. This problem would then be posed as the solution f of the linear system $Kf = d$. Naïvely one might simply try the basic approach from an introductory Linear Algebra class, that $f = K^{-1}d$. Often this sort of a solution is not possible due to the tendency of the operator K to be ill-conditioned in most image processing applications[2]. Hence in solving these systems, the necessity of a regularization scheme or other methods becomes evident.

1.2.1. *Generalized Tikhonov Regularization.* For the problem $Kf = d$, where $K : \mathcal{H}_1 \to \mathcal{H}_2$, $f \in \mathcal{H}_1$ and $d \in \mathcal{H}_2$, where $\mathcal{H}_1, \mathcal{H}_2$ are real, finite dimensional Hilbert spaces, we then define the generalized Tikhonov functional by

$$T_\alpha(f, d) = \rho(K(f), d) + \alpha J(f). \tag{1}$$

Here $\alpha > 0$ is the regularization parameter, $J : \mathcal{H}_1 \to \mathbb{R}$ is known as the penalty functional, and $\rho : \mathcal{H}_2 \times \mathcal{H}_2 \to \mathbb{R}$ is the data discrepancy functional [9]. The data discrepancy functional ρ has the purpose of quantifying how well the prediction Kf fits the observed data, d. One familiar and often used functional[3] comes from the 2-norm $\rho(g_1, g_2) = \frac{1}{2}\|g_1 - g_2\|^2$, for $g_1, g_2 \in \mathcal{H}_2$. I note that with this functional, and from the representation of Eqn. (1) that $\rho(f, d)$ is in fact the residual

$$\rho(f^*, d) = \frac{1}{2}\|Kf^* - d\|^2 \tag{2}$$

where f^* is an approximation to the true solution, f. Minimizing this functional explicitly, we would then obtain our true solution (*or best approximation*).

The reason for the penalty functional J in Eqn. (1) is to introduce stability into the system by incorporating a priori information about the desired solution f. The reason we include this term is that it allows for various characteristics of our original solution to be highlighted,

[2]A discrete example of a smoothing kernel often used is $[K]_{ij} \propto e^{((i-j)h)^2}$, where h is the discrete stepsize. For large systems then the eigenvalues of K clearly tend to zero.

[3]Other commonly used data discrepancy functionals include the Kullback-Leibler information divergence, and the Negative Poisson log likelihood functional, see Vogel, [9]. This paper will focus only on the functional in Eqn. (2).

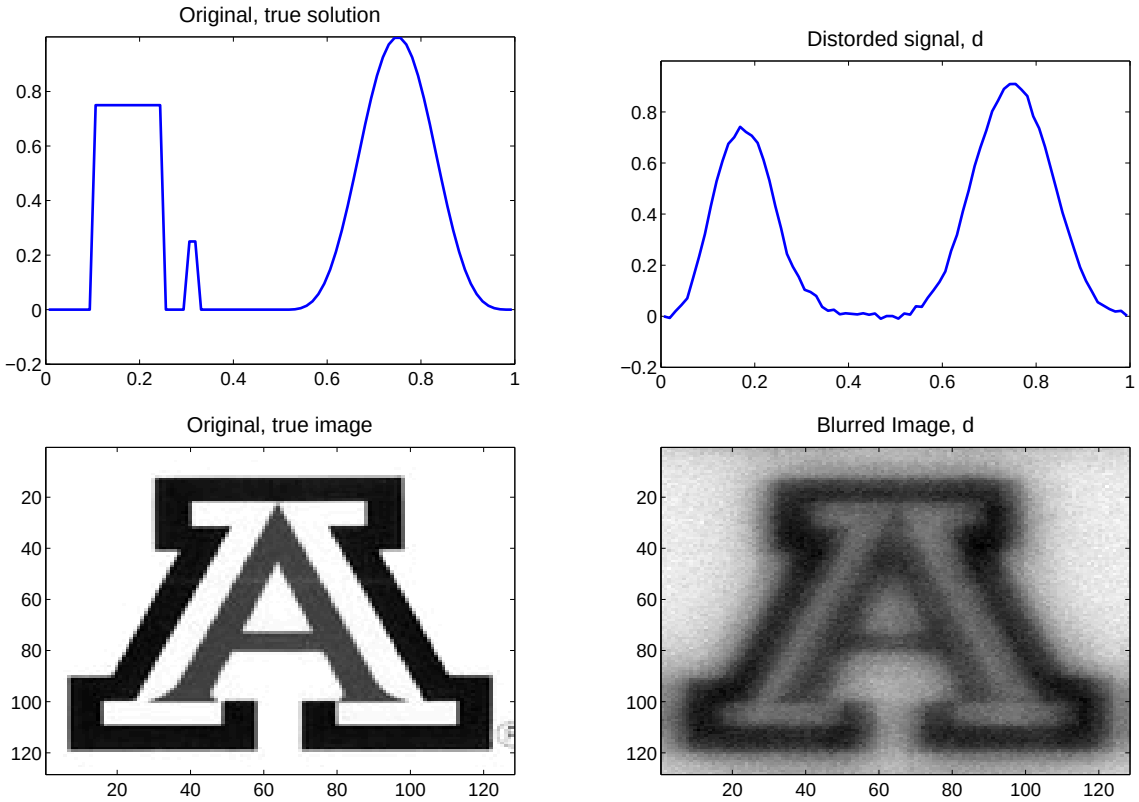

FIGURE 1. On the left we have an example of the true solutions, f, and the right is our observed data d, used in Sections 3.1 & 3.2.

as we will see in Section 1.2.2. Similar to the data discrepancy functional ρ, the standard Tikhonov penalty functional is related to the 2-norm, $J(f) = \frac{1}{2}\|f\|^2$. Other commonly used penalty functionals are the negative entropy functional $J(f) = \langle f, \log f \rangle$, and the Sobolev functional

$$J(f) = \frac{1}{2} \int \sum_i \left(\frac{\partial f}{\partial x_i} \right)^2 \tag{3}$$

which penalizes any non-smooth solutions [9]. While each of the aforementioned functionals serve unique purposes, as you may have guessed from the title of this paper we will be looking at the Total Variational functional, which we present in the following section.

1.2.2. *Total Variational Regularization.* From our favorite real analysis text [2], [6] we find the definition for the Total Variation (TV) of a function f as

$$TV(f) \stackrel{\text{def}}{=} \sum_i |f_{i+1} - f_i| \tag{4}$$

Claim 1. *For $f \in C^1$, let $\|\cdot\|_1$ denote the L_1 norm, then*

$$TV(f) = \|\nabla f\|_1 \tag{5}$$

on the interval $[0, 1]$.

Proof. Given $f \in C^1$, consider the one-dimensional function $f : \mathbb{R} \to \mathbb{R}$, let $f(x_i) = f_i$, $\Delta x_i = x_i - x_{i-1}$, $x_i \in \mathbb{R}$, such that $x_i \neq x_{i-1}$ and $x_i > x_{i-1}$, $i = 1, 2, \ldots n$. Recall the finite difference quotient

$$f_i' = \frac{f_i - f_{i-1}}{\Delta x_i}$$

and the discretized form of the integral

$$\int_0^1 f dx = \sum_{i=1}^n f_i \Delta x_i$$

then

$$\begin{aligned} TV(f) &= \sum_{i=1}^n |f_{i+1} - f_i| \\ &= \sum_{i=1}^n \left| \frac{f_{i+1} - f_i}{\Delta x_i} \right| \Delta x_i \\ &= \sum_{i=1}^n |f_i'| \, \Delta x_i \\ &= \int_0^1 |f'| \, dx \\ &= \|f'\|_1 \end{aligned}$$

Therefore we have that, for the one dimensional case, $TV(f) = \|f'\|_1$. The generalization to n-dimensional space should be clear, hence we have shown that $TV(f) = \|\nabla f\|_1$.

♣

When we view the TV functional as in Eqn (5), we see that this choice penalizes oscillatory solutions, while allowing for jump discontinuities in the original solution to be preserved. Typically, functions of bounded (*total*) variation have a set of discontinuities of measure zero [1], and therefore are continuous almost everywhere. These discontinuities may be

identified with the 'edges' or boundaries of objects in our image we are trying to reconstruct, and this ability is one of the primary reasons it is used in image processing[4]. Further, it has been shown by Caselles, Chambolle & Novaga [1] that set of discontinuities in our solution f^* is contained in the set of discontinuities in the observed data, f.

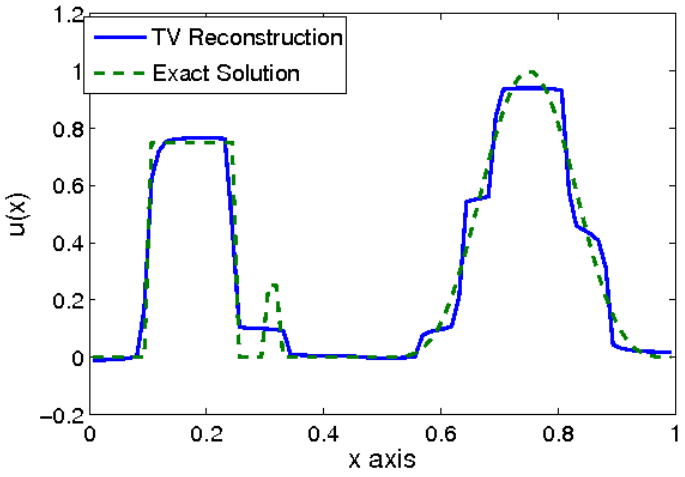

FIGURE 2. Stair-casing effect caused by TV functional.

Hence, if our observed data f is smooth, we expect our solution to be smooth as well. At first glance, this should be a surprising result, as we noted before that the TV functional penalizes smooth solutions. This, however, does not exclude transition layer functions [2], and we often see this as a 'stair-casing' effect, nicely displayed in [1] and can also be seen in Figure (2).

Combining the TV functional with the data discrepancy functional in Eqn (2) for our generalized Tikhonov regularization, we then have

$$\mathcal{T}(f) = \frac{1}{2}\|Kf - d\|^2 + \alpha TV(f),$$

which is the form of the functional we wish to minimize in this paper. I do note here that since the minimization algorithms used in the following sections require that the functional $\mathcal{T}$ be differentiable, the non-differentiability of the absolute value at the origin is a cause for concern. To circumvent this issue, we approximate the norm $|\cdot|$ with $\sqrt{(\cdot)^2 + \beta^2}$, for small $\beta > 0$ (*See Figure 3*). Our approximation to the TV functional in Eqn (5) then becomes in one and two spacial dimensions, respectively

[4]The ability of Total Variational Regularization to restore textures of images is less effective, however. This is an obvious result of the relationship in Eqn. (5), and we tend to have nearly piecewise-constant images as a result of this method.

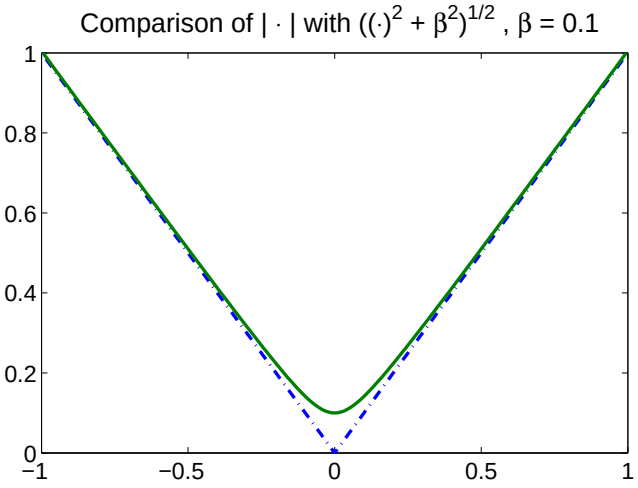

FIGURE 3. Comparison of absolute value function and its approximation.

$$J_\beta(f) = \int_0^1 \sqrt{\left(\frac{df}{dx}\right)^2 + \beta^2} dx, \tag{6}$$

$$J_\beta(f) = \int_0^1 \int_0^1 \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 + \beta^2} dx dy \tag{7}$$

We now focus on minimizing

$$\mathcal{T}(f) = \frac{1}{2}\|Kf - d\|^2 + \alpha J_\beta(f), \tag{8}$$

and move to the discussion of the methods we may use to do so.

2. Optimization Methods

In the following subsections, I will provide a brief introduction to some common minimization algorithms used in this research, and a presentation of the Levenberg-Marquardt method. In what follows, let $L : \mathcal{H} \to \mathbb{R}$, where $\mathcal{H}$ is a finite dimensional Hilbert space, and we wish to determine a minimizer of L over $\mathcal{H}$, which we define to be

$$f^* = \arg\min_{f \in \mathcal{H}} L(f)$$

2.1. **The Steepest Descent.** The method of Steepest Descent is a very familiar and trustworthy optimization technique, guaranteed to make progress so long as the gradient of the functional we wish to minimize is non-zero [4]. Hence, in minimizing $L(f_\nu)$, given that L is differentiable in a neighborhood of f_ν, we compute the minimizer F through the following iterative process [3]

$$f_{\nu+1} = f_\nu - \gamma \nabla L(f_\nu), \tag{9}$$

for a fixed $\gamma > 0$. It is well documented that this method has a linear convergence rate [4], and so I will refrain from proving this explicitly. As I just mentioned, this method will nearly always converge to the minimizer, however if L has any oscillatory or undulating surface behavior, many iterations are required as the algorithm follows the direction of the negative gradient blindly. Having a fixed step-size γ is another limitation of this algorithm.

For example, if our method is iterating in a direction with a very large gradient approaching a local minima, we run the risk of 'overshooting' the minima. Think of this as rolling a ball-bearing down the side of a steep bowl. The bearing picks up great speed, but as we approach the bottom of the bowl the bearing will continue on and roll partially back up the side. From here the bearing slides down the side again, and will continue on until eventually coming to rest at the bottom.

Ideally, in areas where the gradient is very large, we would use a very small step-size, in order to guard against this problem. Conversely, consider the case where we are progressing along our functional L where the gradient is very low. In this case, we would want to take larger steps to avoid having to iterate many times to reach our minima.

The method of Steepest Descent, however, does just the opposite of what we would like it to do. As we see in Eqn. (9) our step-size is $\propto -\nabla L(f_\nu)$, therefore in steep regions where the gradient is large, we take larger steps, while along gentle slopes where the gradient is very small we take very small steps. This deficiency brings about the need for an algorithm which can incorporate some knowledge of the curvature of the functional L as well, which we discuss next with the famous Newton's method.

2.2. **Newton's Method.** We begin by looking at the truncated 2^{nd} order Taylor series of our functional $L(f)$, [4]

$$L(f+s) \approx L(f) + \nabla L(f)^T s + \frac{1}{2} s^T \text{Hess } L(f) s, \quad \text{where}$$
$$\{\text{Hess}\}_{ij} = \frac{\partial^2}{\partial f_i \partial f_j}$$

Minimizing with respect to s, we then have that

$$\begin{aligned} 0 &= \nabla L(f) + \text{Hess } L(f)s \\ &\Downarrow \\ \text{Hess } L(f)s &= -\nabla L(f), \quad \text{hence} \\ s &= -\left[\text{Hess } L(f)\right]^{-1} \nabla L(f) \end{aligned} \tag{10}$$

Substituting this result into the general minimization form $f_{\nu+1} = f_\nu + s$ we have the Newton's method iterations

$$f_{\nu+1} = f_\nu - \left[\text{Hess } L(f_\nu)\right]^{-1} \nabla L(f_\nu). \tag{11}$$

The immediate benefit we gain from this method is that Newton's method has a quadratic convergence rate [9]. I note here that the method presented in Eqn. (11) is no different than the standard Newton root finding method, since we are not necessarily interested when $L(f) = 0$, but we are interested in when $\nabla L(f) = 0$, which guarantees we will arrive at a local minima[5]. Furthermore, this method incorporates information about the curvature of the surface of L, and hence guarantees that the proper step-length and direction will be provided, unlike the method of Steepest Descent [4].

The main problem with Newton's method is that if we do not have an initial guess f_0 which is sufficiently close to the minimizer f^*, Newton's method may exhibit unpredictable behavior. In practice, we are often have some understanding of the basic structure of L which helps us in choosing a suitable guess. In the following method, I introduce the adaptive Levenberg-Marquardt method, which is essentially a mixing between Steepest Descent and Newton's method.

2.3. **The Levenberg-Marquardt Method.** Recall from the Method of Steepest Descent and Newton's method, the respective iterative schemes

$$\begin{aligned} f_{\nu+1} &= f_\nu - \gamma \nabla L(f_\nu), \\ f_{\nu+1} &= f_\nu - \left[\text{Hess } L(f_\nu)\right]^{-1} \nabla L(f_\nu). \end{aligned}$$

We now introduce the Levenberg-Marquardt method (LMM), which is an adaptive blending between the two functions shown above: [5]

$$f_{\nu+1} = f_\nu - \left[\text{Hess } L(f_\nu) + \lambda_\nu \mathbb{I}\right]^{-1} \nabla L(f_\nu), \tag{12}$$

where $\mathbb{I}$ represents the identity matrix, and $\lambda_\nu > 0$ an adjustable parameter. Hence we see that if λ_k is large, then our algorithm is behaving like

[5]In our example we are guaranteed that we will have a local/global minimum, since our functional being minimized is symmetric positive definite.

$$f_{\nu+1} = f_\nu - \frac{1}{\lambda_\nu}\nabla L(f_\nu)$$

which we recognize to be the Method of Steepest Descent, with $\lambda_\nu \equiv \frac{1}{\gamma}$. Conversely, if $\lambda_\nu \to 0$, then it should be clear that the LMM performs similarly to Newton's method. The beauty of this algorithm lies in the variability of λ_ν, and there are many variants on just how to adjust this parameter[6]. The method I have chosen to update this scaling factor is outlined in the pseudo-code for the LMM below

```
Given TOL, iteration_max
FLAG = 1, i = 1
lambda_1 = 0.01, nu = 3
f_0 = initial guess

While( FLAG == 1)
  Compute Gradient of L(f_i), Grad_L_i
  Compute Hessian of L(f_i), Hess_L_i

  f_new = f_i - [Hess_L_i + lambda_i*eye]\Grad_L_i
    % eye := Identity matrix with dimensions equal to Hess_L_i

  if( || Grad_L_f_new || < || Grad_L_i || )
    Accept the step,
    f_i+1 = f_new
    lambda_i+1 = lambda_i/nu
    nu = 3
    i = i+1
  else
    Reject the step,
    lambda_i = nu*lambda_i
    nu = 3*nu
  end

  if( || Grad_L_i || < TOL )
    FLAG = 0
  elseif( i > iteration_max )
    FLAG = 0
  end
end
```

Hence we see that if we are approaching a minimia, or equivalently $\|\nabla L(f_{\nu+1})\| < \|\nabla L(f_\nu)\|$, we reduce the size of $\lambda_{\nu+1}$, therefore relying more heavily on the contribution from Newton's Method. Conversely, if a minima is not being approached, we begin to use the power (*albeit slower convergence*) of the Steepest Descent.

I note here that in the presentation of the LMM above, I in no way claim that this is an optimized variant of this algorithm. The choice here to adjust λ_ν be a factor of 3 was chosen simply because it was straightforward to implement, and produced pleasing results, as we see below in Sections 3.1 & 3.2. In general, the more basic update rules for λ_ν, such as the method above, adjust by a factor of 10, or other such relevant value. Other, more elegant methods exist, of which some are designed to solve specific problems, or are well-conditioned for a very large class of functions.

[6]MATLAB uses the LMM in it's Optimization toolbox, and offers a very interesting method of updating λ_ν, using a prediction of future function values $f_{\nu+1}$ using a cubic interpolation. See www.mathworks.com.

In practice, the LMM works very effectively, and is used in many high level optimization software packages, including Python and the MATLAB Optimization toolbox acting as the primary non-linear optimizer for medium sized models [5]. LMM has a super-linear convergence rate, as one might hope from it's relation to the method of Steepest Descent and Newton's method, but has been shown under the proper conditions to converge locally with a quadratic rate, see Appendix A and [10].

Numerical issues do occur in larger models ($\mathcal{O}(10^3)$ *dimensional models*) due to the matrix inversion in finding f_{new} above, as the speed due to the innovation of this model is overtaken by the time to compute said inverse. One possible remedy for this problem, which I have not attempted at this point, would be to use another solver to iteratively determine the value. This subprocess is often carried out by the Conjugate Gradient method, and is in fact exactly the process which occurs in the solver which I compare in the two-dimensional results Section (3.2).

2.4. **Conjugate Gradient.** Finally, we provide a brief overview of the Conjugate Gradient method (CG). Often times it is impractical to compute explicitly the Hessian of the functional L, either due to numerical issues as it is often costly to calculate, or because it is not possible to analytically determine the Hessian. In exceedingly large systems, avoiding the computation of the Hessian at each iteration is highly valuable, as we have a significant decrease in the total number of operations performed. It is for this reason that CG is a very fast and efficient method used regularly in optimization.

This method requires little more information than does the method of Steepest Descent[7], but as you may recall the method of Steepest Descent is prone to a zig-zagging nature as it plods along. CG, however, is constructed to ensure that this cannot occur by modifying the gradient at each iteration to ensure that it is orthogonal to all previous gradient directions, see Eqn. (14). In what follows I will construct the CG method from a linear quadratic function, but nonlinear versions exist as well [9], [4], [7].

Given vectors $x, b \in \mathbb{R}^n$, symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$, and $c \in \mathbb{R}$, we have the quadratic function $g : \mathbb{R}^n \to \mathbb{R}$ defined by

$$g(x) = \frac{1}{2}x^T Ax - x^T b + c.$$

Minimizing with respect to x, we then have that

$$\begin{aligned} f'(x) &= Ax - b, \quad \text{setting } f'(x) = 0 \text{ then gives us} \\ Ax &= b \end{aligned}$$

We define the residual $r = b - Ax$, and note that $f'(x) = -r$. Now, we follow the iterative update rule we used earlier, $x_{\nu+1} = x_\nu + \alpha s_\nu$, where α is a constant, and s_ν is the previous search direction. Minimizing $f(x_{\nu+1}) = f(x_\nu + \alpha s_\nu)$ about α, we see that [4]

$$\begin{aligned} \frac{\partial}{\partial\alpha} f(x_{\nu+1}) &= 0 \\ &= f'(x_{\nu+1})^T \frac{\partial}{\partial\alpha} x_{\nu+1} \\ &= (Ax_{\nu+1} - b)^T \frac{\partial}{\partial\alpha}(x_\nu + \alpha s_\nu) \\ &= -r_{\nu+1}^T s_\nu \end{aligned} \tag{13}$$

Hence we see that our residual, or equivalently the gradient of f, is orthogonal to the previous search direction. At this point, we will begin to see the difference between CG and Steepest Descent. By properly

[7]One of the requirements of the Conjugate Gradient method is that we have a symmetric positive definite function which we wish to minimize.

choosing α, CG ensures that our residual is also orthogonal to *every previous step*, not just the most recent step. Notice

$$\begin{aligned} r_{\nu+1} &= b - Ax_{\nu+1} \\ &= b - A(x_\nu + \alpha s_\nu) \\ &= b - Ax_\nu - \alpha A s_\nu \\ &= r_\nu - \alpha A s_\nu \end{aligned} \tag{14}$$

Using this result, and from Eqn. (13) we have

$$\begin{aligned} 0 &= r_{\nu+1}^T s_\nu \\ &= (r_\nu - \alpha A s_\nu)^T s_\nu \\ &\Downarrow \\ r_\nu^T s_\nu &= \alpha s_\nu^T A^T s_\nu \\ &\Downarrow \\ \alpha &= \frac{r_\nu^T s_\nu}{s_\nu^T A^T s_\nu} \end{aligned} \tag{15}$$

The iterative update rule of s_ν is that of Fletcher & Reeves [4], and is given by

$$s_{\nu+1} = r_{\nu+1} + \frac{r_{\nu+1}^T r_{\nu+1}}{r_\nu^T r_\nu} s_\nu, \quad s_0 = r_0.$$

3. Numerical Results

In this section we consider minimization of the Tikhonov-TV functional, presented in Eqn. (8). Our primary measures for numerical performance are:

- Norm of the relative iterative solution error, $e_\nu = \frac{\|f_\nu^* - f_\nu\|}{\|f_\nu\|}$, again where f represents the true solution, and f^* the approximated solution.
- Clock time (*seconds*), the time necessary to reach some predefined relative iterative solution error tolerance.
- Iteration count.
- Qualitative results[8].

[8]By Qualitative I refer to the famous 'eye-ball norm'... the results should accurately reflect the properties of the true solution.

3.1. **1-Dimensional Numerical Results.** Hence my one-dimensional numerical results. In the table below, the relative iterative solution error stopping tolerance was $E_{tol} = 0.15$, $\alpha = 0.005$:

Method	Steepest Descent	Conjugate Gradient	Newton	Levenberg-Marquardt
Iterations	342	214	84	12
Clock time (s)	7.9212185	5.3332195	2.7865335	0.456148

TABLE 1. 1-Dimensional Results

In this one-dimensional case, we clearly see the fast convergence of the LMM, as well as a relatively low computational time. As expected, Steepest Descent was the slowest method tested, while the Newton iterations (*See Figure 2*) performed the best aside from LMM. While it was noted in Section 2 that the Newton's method has a faster convergence rate than LMM, the slower convergence is attributed to a non-optimized initial guess for the Newton's method. To prevent the Newton solver from 'blowing-up' to an unexpected solution, a line-search has been added to the routine to ensure that this does not occur. For a detailed discussion of how, and why the linesearch works, see Vogel [9]. For the purposes of this paper, I will just say that the line-search keeps the solver moving in the right direction, but does so slowly.

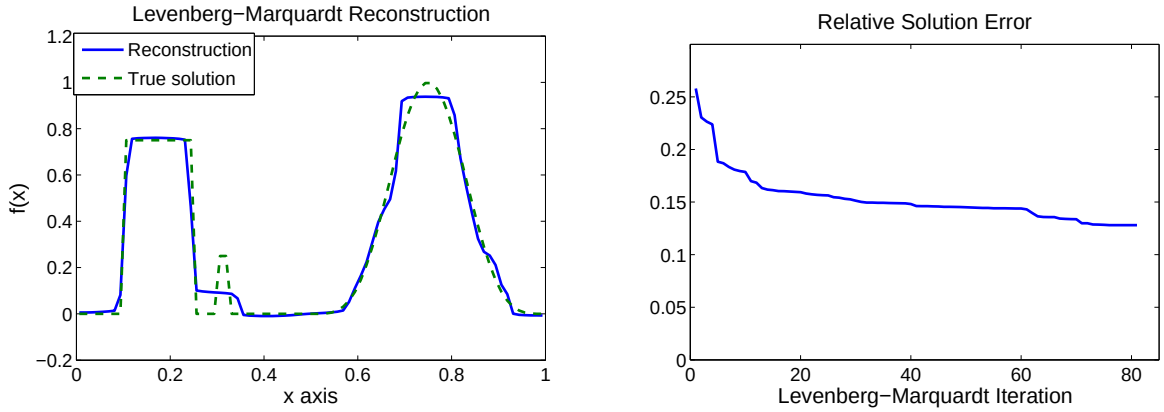

FIGURE 4. 1-D signal reconstruction using LMM, with relative error at each iteration shown in the right hand plot.

In the reconstruction from Figure 4, I note that the step-function on the left-most side is recovered very accurately, as our analysis predicted the Total Variational penalty functional would. On the parabolic structure to the right, we notice that the regularization flattens out the peak, reflecting that the method does lose some of the textures of the original image if it is not piecewise constant. Also, we see that the small peak fails to be recovered by the algorithm. It is in fact possible to accurately recover this peak by reducing the regularization parameter α, but doing so results in an under-regularization of the signal, and our approximated solution becomes very noisy (*oscillatory*), as we must penalize higher-frequencies less to get better resolution of fine structures.

3.2. **2-Dimensional Numerical Results.** Similar to the results for the one-dimensional case, below we have the results for the two-dimensional reconstruction. Here we have a stopping tolerance of $E_{tol} = 0.15$.

I note here that the results from Table (3.2) indicate that the variant of Newton's method used converges faster than LMM, as opposed to the results from Table (3.1) for the one-dimensional results. This is somewhat misleading, however, as I mentioned earlier this variant (2-*D only*) of Newton's method uses the Conjugate Gradient to solve a linear system as a subproblem, consequentially there is an effective 156 Primal Dual Newton iterations (*1 Newton iterations has 2 CG sub-iterations*).

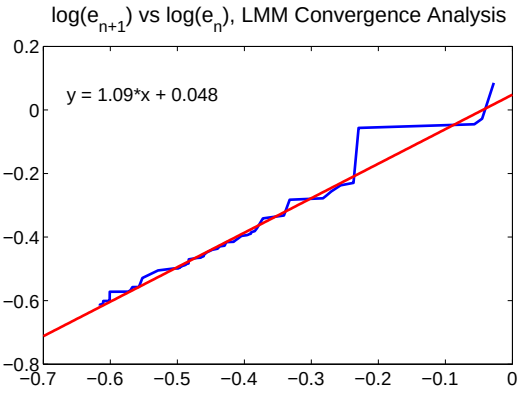

FIGURE 5. Here we see that the convergence rate of LMM is ~ 1.1, hence LMM is super-linear.

Method	Primal Dual Newton	Levenberg-Marquardt
Iterations	52	113
Clock time (s)	22.171279	53.346740

TABLE 2. 2-Dimensional Results

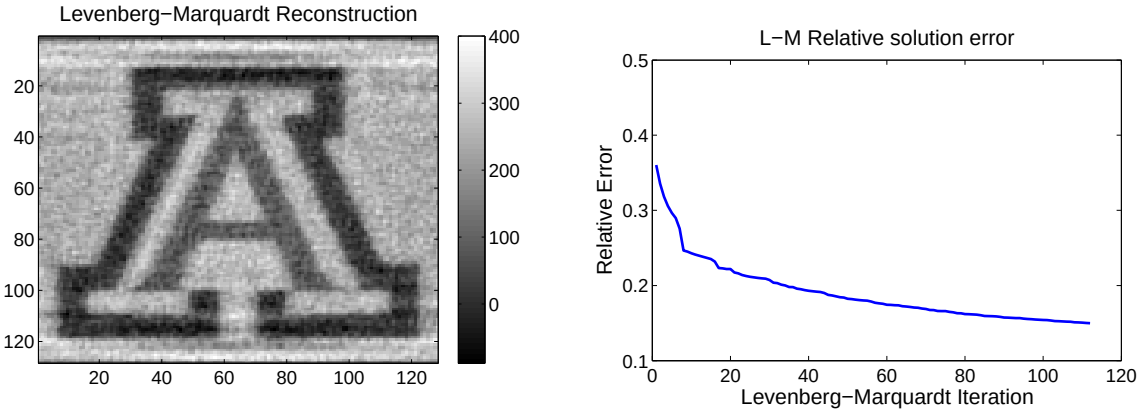

FIGURE 6. 2-D image reconstruction using LMM, with relative error at each iteration shown in the right hand plot.

We also see a longer clock-time for the LMM in the two-dimensional case, due in large part to the inversion of the Hessian matrix, see Eqn (12). I am quite certain that it is possible to speed up the LMM code substantially, with the limitations being my own coding abilities. MATLAB has the ability to solve sytems using linear algebric methods rather quickly, but in the code that I have written each computation is done explicitly. If the proper adjustments were to be made, I am confident that the Clock time would be very close to that of the 2-D Newton code. The Newton code, I will note, was written professionally. Hence, while the results in Table 3.2 seem to suggest otherwise, I claim that the Levenberg-Marquardt method did perform as well as the comparing model, and it remains to be seen if it actually is superior as we saw in the 1-D case.

The qualitative results of the 2-D LMM experiment were, as we see in Figure 6 and Figure 7, quite successful. The images which were reconstructed were ideal for the Total Variational functional, as they are quite clearly piecewise constant with jump discontinuities.

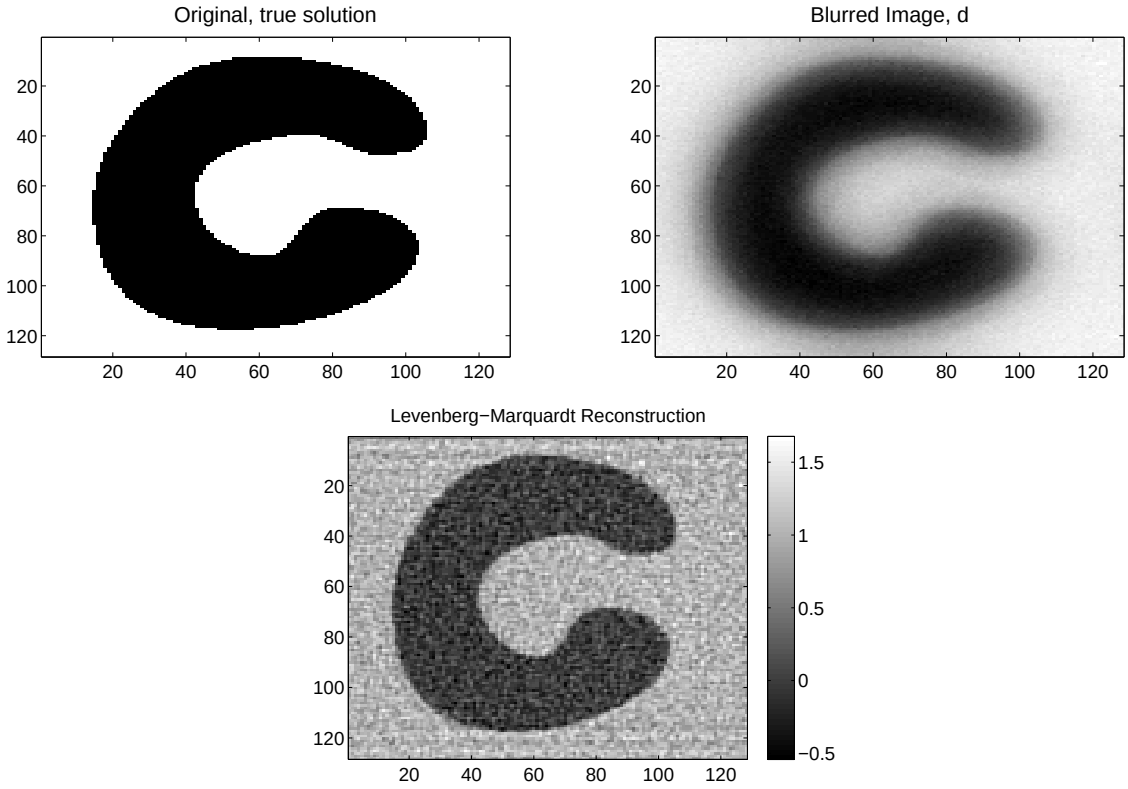

FIGURE 7. Top left plot is the original image, top right plot is the blurred image, and bottom image is the LMM reconstruction.

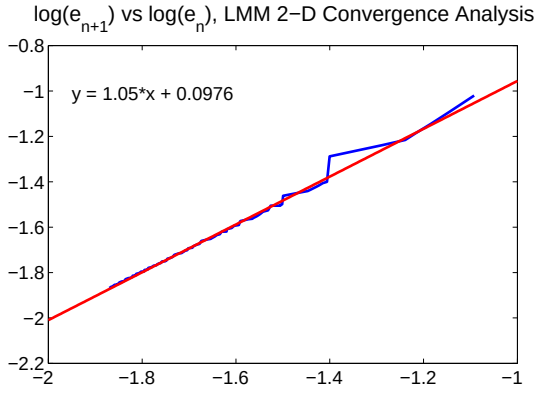

FIGURE 8. Here we see that the convergence again rate of LMM is ~ 1.1 for the 2-d case, hence LMM is super-linear.

In our final example, I attempted to reconstruct an image which was not nearly piece-wise constant, as opposed to those used in the reconstructions of Figures 6 & 7. In Figure 9 we see that the reconstructed image[9] accurately recreates areas of higher variation, but textures are very difficult to see.

[9]Courtesy of Lola, my Lhasa-Poo, who was forced to wear a Bee costume for Halloween one year against her will. I wish to make it clear that this costume was not my choice. . . I wanted it to be Spider Man.

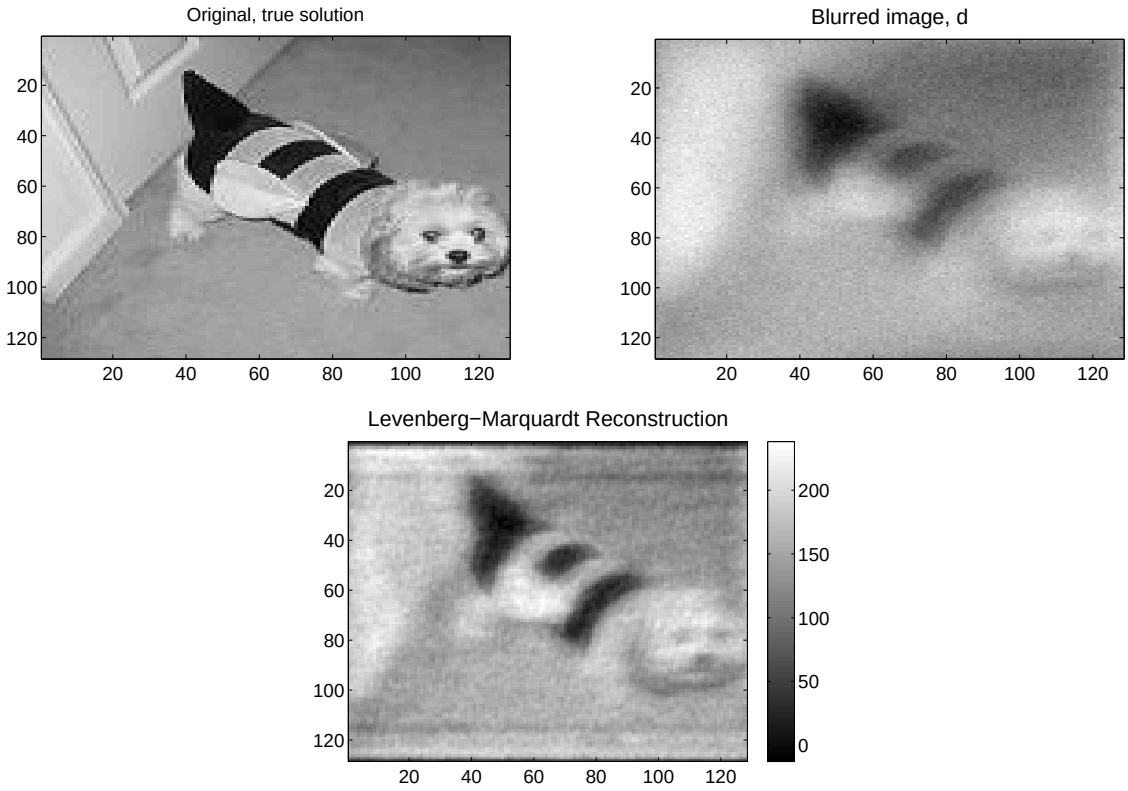

FIGURE 9. Top left plot is the original image, top right plot is the blurred image, and bottom image is the LMM reconstruction.

4. Final Remarks

This paper investigated the Levenberg-Marquardt method in minimizing the Tikhonov-Total Variational functional. When compared to other well know optimization methods, the Levenberg-Marquardt was superior in the one-dimensional case, and also provided sufficient two-dimensional results to justify it's widespread use. Numerical evidence of super-linear convergence of the Levenberg-Marquardt method was also shown.

I would like to acknowledge Professor Juan Restrepo for his assistance and guidance on this research paper, as well as Professor Hermann Flaschka, Professor Shankar Venkataramani and Dr. Ali Bilgin for their helpful discussions during the course of this project. I would also like to recognize Dr. Michael Tabor and the Program in Applied Mathematics at the University of Arizona for support during this research.

Appendix A. Quadratic convergence rate of the Levenberg-Marquardt method

In what follows, I present here a variant of the LMM presented in Eqn. (12), as given by Yamashita & Fukushima [10].

As before, f will be assumed to be a mapping $f : \mathbb{R}^n \to \mathbb{R}^n$, and to be locally differentiable. Given our initial guess $x_0 \in \mathbb{R}^n$, we find our next estimate $x_1 = x_0 + \delta$, where $\delta \in \mathbb{R}^n$ is to be determined by our algorithm below. Evaluating the function f at this point we have by Taylor expansion

$$f(x_0 + \delta) \approx f(x_0) + \delta J_0$$

where J_0 denotes the Jacobian of f evaluated at x_0, assumed to be non-singular. Squaring the right hand side above, and minimizing with respect to δ, we can then determine that

$$\begin{gathered} \frac{d}{d\delta}\left[(f(x_0) + \delta J)^2\right] = 0 \\ \Downarrow \\ J^T J\delta = -J^T f(x_0) \end{gathered} \tag{16}$$

Where LMM differs from other methods is that we now recast (16) as a so-called damped version as follows

$$(J^T J - \lambda I)\delta = -J^T f(x_0). \tag{17}$$

Here I is the $n \times n$ identity matrix, and $\lambda > 0$ is a parameter adjusted at each step, according to some rule. Solving for δ, we then have the update rule

$$x_{\nu+1} = x_\nu - \frac{J^T f(x_0)}{(J^T J - \lambda I)}. \tag{18}$$

Note the similarity to Eqn. (12). The only essential difference here is that we approximate the true Hessian with a product of the Jacobians. As we saw earlier, LMM represents a mixture between Newton iterations and Steepest Descent.

Denote the set of all minimizers of f to be $\mathcal{X} \neq \emptyset$. We define a local error bound for f as

Definition 1. Let $N \subseteq R^n$ such that $\mathcal{X} \cap N \neq \emptyset$. $\|f(x)\|$ denotes a local error bound on N provided there exists $k > 0$ such that

$$kd(x, \mathcal{X}) \leq \|f(x)\|, \quad \forall x \in N \tag{19}$$

Here we have that $\|\cdot\|$ denotes the usual $2-$norm, $d(y, A)$ is the distance between a point y and the set A defined as [6]

$$d(y, A) = \inf\{d(y, \alpha) : \alpha \in A\}. \tag{20}$$

Note that since $(J^T J - \lambda I)$ is positive definite, there exists a unique solution, δ_k to (17).

Recall that we determine successive steps by the update rule

$$x_{k+1} = x_k + \delta_k$$

where δ_k is the unique solution to (17). Define the quadratic function $\phi_k : \mathbb{R}^n \to \mathbb{R}$ by

$$\phi_k(\delta) = \|J_k\delta + f(x_k)\|^2 + \lambda_k\|\delta\|^2$$

where ϕ_k is convex, and it should be clear that the minimization of ϕ_k with respect to δ will be equivalent to (17).

Claim 2. *$\exists x^* \in \mathcal{X}$ such that*

(a) *For $\epsilon > 0$, there exists $c_1 > 0$ such that*

$$\|J(y)(x-y) - (f(x) - f(y))\| \leq c_1 \|x-y\|^2, \quad \forall x, y \in N(x^*, \epsilon) \tag{21}$$

provided that f is continuously differentiable, and f and J are Lipschitz continuous, with Lipschitz constants L and K, respectively. $N(a, \epsilon) = \{x : \|x - a\| \leq \epsilon\}$

(b) *$\|f(x)\|$ is a local error bound on $N(x^*, \epsilon)$, that is there exists $c_2 > 0$ such that*

$$c_2 d(x, \mathcal{X}) \leq \|f(x)\|, \quad \forall x \in N(x^*, \epsilon) \tag{22}$$

We have that (21) is true since, for $\xi \in [x, y]$

$$\begin{aligned}
\|J(y)(x-y) - (f(x) - f(y))\| &= \|J(y)(x-y) - J(\xi)(x-y)\| \quad \text{by MVT} \\
&= \|(J(y) - J(\xi))(x-y)\| \\
&\leq \|J(y) - J(\xi)\|\|x-y\| \quad \text{From Holder's inequality} \\
&\leq (K\|x-y\|)\|x-y\| \\
&= K\|x-y\|^2
\end{aligned}$$

where K is the Lipschitz constant for J. Hence we see that (21) is satisfied if $c_1 = K$. Now we also will make the assumption that $\lambda_k = \|f(x_k)\|^2$, and let

$$\|x_k - \bar{x}_k\| = d(x_k, \mathcal{X}).$$

Claim 3. *Given the above assumptions, if $x_k \in N(x^*, \epsilon/2)$, then the solution of (17) δ_k satisfies*

$$\|\delta_k\| \leq c_3 d(x_k, \mathcal{X}) \tag{23}$$

$$\|J(x_k)\delta_k + f(x_k)\| \leq c_4 d(x_k, \mathcal{X}), \quad \textit{where} \tag{24}$$

$$\begin{aligned}
c_3 &= \frac{\sqrt{c_1^2 + c_2^2}}{c_2} \\
c_4 &= \sqrt{c_1^2 + L^2}
\end{aligned}$$

Proof. Given that δ_k is a minimizer of (17), we have that

$$\phi_k(\delta_k) \leq \phi_k(\bar{x}_k - x_k). \tag{25}$$

Also, since $x_k \in N(x^*, \epsilon/2)$ we find

$$\begin{aligned}
\|\bar{x}_k - x^*\| &\leq \|\bar{x}_k - x_k\| + \|x_k - x^*\| \\
&\leq \|x_k - x^*\| + \|x_k - x^*\| \\
&\leq \epsilon
\end{aligned}$$

hence $\bar{x}_k \in N(x^*, \epsilon)$. From the definition of ϕ, (25) and Claim (2) we find

$$\begin{aligned}
\|\delta_k\|^2 &\leq \frac{1}{\lambda_k}\|J(x_k)\delta_k + f(x_k)\|^2 + \|\delta_k\|^2 \\
&= \frac{1}{\lambda_k}\phi_k(\delta_k) \\
&\leq \frac{1}{\lambda_k}\phi_k(\bar{x}_k - x_k) \\
&= \frac{1}{\lambda_k}\|J(x_k)(\bar{x}_k - x_k) + f(x_k)\|^2 + \|\bar{x}_k - x_k\|^2 \\
&\leq \frac{c_1^2}{\lambda_k}\|\bar{x}_k - x_k\|^4 + \|\bar{x}_k - x_k\|^2 \\
&= \left(\frac{c_1^2\|\bar{x}_k - x_k\|^2}{\lambda_k} + 1\right)\|\bar{x}_k - x_k\|^2
\end{aligned}$$

but since $\lambda_k = \|f(x_k)\|^2 \geq c_2^2\|\bar{x}_k - x_k\|^2$ by assumption, we find that our last line gives us

$$\|\delta_k\| \leq \sqrt{\frac{c_1^2 + c_2^2}{c_2^2}}\|\bar{x}_k - x_k\|^2 \tag{26}$$

We then find (24) in similar fashion.

♣

Now we conclude with the proof that LMM converges quadratically, given that its iterations lie sufficiently close to the minimizer x^* [10].

Claim 4. *For $x_k, x_{k-1} \in N(x^*, \epsilon/2)$ we have that*

$$d(x_k, \mathcal{X}) \leq Cd(x_{k-1}, \mathcal{X})^2 \tag{27}$$

is true, where $C = \frac{c_1c_3^2 + c_4}{c_2}$. We then say that the LMM converges quadratically.

Proof. Supposing that $x_k, x_{k-1} \in N(x^*, \epsilon/2)$, and from our update rule $x_k = x_{k-1} + \delta_{k-1}$ it follows from (21)-(24) that

$$\begin{aligned}
c_2d(x_k, \mathcal{X}) &= c_2d(x_{k-1} + \delta_{k-1}, \mathcal{X}) \\
&\leq \|f(x_{k-1} + \delta_{k-1})\| \\
&= \|f(x_{k-1}) + \delta_{k-1}J(x_{k-1})\| \quad \text{by a second order Taylor Expansion} \\
&\leq \|f(x_{k-1}) + \delta_{k-1}J(x_{k-1})\| + c_1\|\delta_{k-1}\|^2 \\
&\leq c_4d(x_{k-1}, \mathcal{X})^2 + c_1c_3^2d(x_{k-1}, \mathcal{X})^2 \\
&= (c_1c_3^2 + c_4)d(x_{k-1}, \mathcal{X})^2 \\
&\Downarrow \\
d(x_k, \mathcal{X}) &\leq \frac{c_1c_3^2 + c_4}{c_2}d(x_{k-1}, \mathcal{X})^2
\end{aligned}$$

Therefore we have that LMM converges quadratically provided that the iterates $x_k \in N(x^*, \epsilon)$ for all k [10].

♠

References

[1] Caselles, V., Chambolle, A. & Novaga, M. *The discontinuity set of solutions of the TV denoising problem and some extensions.* Preprint, www.cmap.polytechnique.fr/∼antonin (2007).
[2] Flaschka, H. Principles of Analysis. The University of Arizona (1991).
[3] Hamming, R. W. Numerical Methods for Scientists and Engineers, 2^{nd} Ed. NY, Dover. (1973).
[4] Heath, M.T. Scientific Computing: An Introductory Survey, 2^{nd} Ed. NY, McGraw Hill (2002).
[5] Roweis, S. Levenberg-Marquardt Optimization. University of Toronto (1996)
[6] Royden, H.L. Real Analysis, 3^{rd} Ed. NY, Macmillan Publishing Company (1988).
[7] Shewchuk, J.R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Ed. $1\frac{1}{4}$. PA, Carnegie Mellon University (1994).
[8] Tarantola, A. Inverse Problem Theory, and Methods for Model Parameter Estimation. PA, Siam (2005).
[9] Vogel, C. Computational Methods for Inverse Problems. Philadelphia, SIAM (2002).
[10] Yamashita, N. & Fukushima, M. On the Convergence of the Levenberg-Marquardt Method. Japan, Kyoto University (2000).