Summer 8-13-2013

Iteratively Regularized Methods for Inverse Problems

Leslie J. Meadows

ITERATIVELY REGULARIZED METHODS FOR INVERSE PROBLEMS

by

LESLIE J. MEADOWS

Under the Direction of Dr. Alexandra Smirnova

ABSTRACT

We are examining iteratively regularized methods for solving nonlinear inverse problems. Of particular interest for these types of methods are application problems which are unstable. For these application problems, special methods of numerical analysis are necessary since classical algorithms tend to be divergent.

INDEX WORDS: Inverse problems, Iteratively regularized methods, Iteratively Regularized Newton, Iteratively Regularized Gauss-Newton

ITERATIVELY REGULARIZED METHODS FOR INVERSE PROBLEMS

by

LESLIE J. MEADOWS

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2013

ITERATIVELY REGULARIZED METHODS FOR INVERSE PROBLEMS

by

LESLIE J. MEADOWS

DEDICATION

To my mother, Yvonne Harris Meadows: Mama, I tried to pinpoint when life blurred the lines between mother and daughter and made you my best friend. But, I realize now, that you have *always* been my best friend. I truly admire you. Thank you for your never-ending support and for always being there to listen. You have “seen me through” from kindergarten to doctorate. I love you.

To my brother, Terrence Neal Harris: Terry, life could not have provided me with a better person as a brother. Your sense of duty to family is an inspiration to everyone. Thank you for “stepping in” on some of my inevitable responsibilities, thus granting me the space to pursue my education. Thank you, as well, for leading the way for me mathematically and for sacrificing sleep to help me prepare. I am even more proud of you, than you of me. I love you.

To my husband, Kevin Michael Dalrymple, I could not have done this without you. Thank you for providing both emotional and financial support. As I pursued my degree, you helped me to stand when I thought I would fall and you stepped in, from time to time, as both father and mother to our children. You ran late night simulations and listened to countless math stories; your love is truly *unconditional*. My degree belongs partially to you. I love you.

To my children, Brenston Neal Dalrymple and Julianna Beverly Yvonne Dalrymple: thank you for your patience and understanding and for keeping friends away when “Mommy had to study”. Brenston, thank you, as well, for lending me your amazing brain and for helping to run some of those late night simulations. Julianna, your talent and love for mathematics *warms my heart*. I am extremely proud of both of you. I love you.

To my grandmothers, Lillie Mae Harris (95 years old) and Anna Elizabeth Byrd Meadows Peterson (97 years old). Thank you for being proud of me, always encouraging me to continue and for helping me to keep my faith. You will see more of me now. I love you.

To my father, Winston Rudolph Meadows: Daddy, you were my biggest cheerleader. I thought your illness and death would kill me, instead, your spirit gave me strength and determination. I can feel the positive energy of your love and pride reaching out from Heaven. Justice is ours; we won. Thank you for staying close and for helping to keep Jesus Christ near me at all times. You kept your promise; your heart is safe with me. You can go now. I love you. *Godspeed*.

ACKNOWLEDGEMENTS

I will be forever grateful to my advisor, Dr. Alexandra Smirnova; you are brilliant, kind and patient. Thank you for investing so much of your time and energy towards helping me earn a doctorate in mathematics. You made this dissertation possible by introducing me to the field of applied inverse problems and working, tirelessly, with me to see it through. I could not have done this without you. Thank you for your enthusiasm, imbuing me with some of your knowledge, helping me to gain financial support and for continuing to be a champion in my corner.

Many thanks, as well, to the other members of my dissertation committee, Dr. Michael Stewart and Dr. Changyong Zhong. I appreciate your guidance and suggestions for improvement; I have included them all. I would be remiss in not thanking the rest of my professsors: Dr. Mariana Montiel (you have a gentle kindness and a talent for teaching), Dr. Lifeng Ding, Dr. Imre Patyi, Dr. Draga Vidakovic (your advice and encouragement mean the world to me), Dr. Frank Hall (I love your stories), Dr. Vladimir Bondarenko (who served on the committee for my candidacy exam and also serves as a source of encouragement), Dr. Zhongshan Li, Dr. Valerie Miller (you saved me; I will never forget) and Dr. Yichuan Zhao. All of you have been part of an incredible journey, helping me to gain the knowledge and ability to conduct research in the fields of matrix analysis, collegiate mathematics education and applied mathematics.

To the chair of the Mathematics and Statistics Department, Dr. Guantao Chen: from the minute that I started the graduate program, I felt your support. Thank you for standing up for me.

To the staff of the mathematics department: Sandra Ahuama-Jonas (thank you for going out of your way to be there for me), Yvonne Pierce, Earnestine Collier-Jones and Matthew Reed. Each of you helped me to sustain a level of sanity; I thank you for that.

I would also like to thank some of my fellow graduate students for their help and support: Tingli Xing (one of my favorite people of all time and a great LaTeX troubleshooter), Hui Lui (my Ph.D. sister), Jeffrey McCammon (we had a simultaneous journey and many hours of study and thoughtful discussion), Malcolm DeVoe (you are wonderful) Mary Hudachek-Buswell (a former colleague and current friend), Douglas Carter, Jr. (wow, we did it), Dirk Gilmore (thanks for taking an interest in my stories), Sajiya Jalil, MaryGeorge Whitney, Nana Li, Linda DeCamp, Benham Aryafar, Annie Banks, Paula Mullins, Brent Woolridge, Heather King, Mikhail Stroeve, Songling Shan, Rachid Marsli, Kelvin Rozier and Sara Malec.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In general, ill-posed and inverse problems have a wide range of applicability. They can be used, for example, to detect geological anomalies given a set of measurements collected at the surface, to identify the shape of a scattering object from electromagnetic wave scattering, to design non-invasive biomedical imaging technologies, etc. From a mathematical standpoint, they can be formulated as ill-conditioned linear systems, inverse Fourier and/ or Laplace transforms, linear and nonlinear integral equations of the first kind and so on.

The problems mentioned above, as well as others, can be represented as solutions to an operator equation of the form: $F(x) = 0$, where the operator F may be linear or nonlinear between two metric spaces. Hadamard has imposed certain conditions on the above operator equation for it to be well-posed [1], namely:

1. there exists a solution of the problem (existence),
2. there is at most one solution of the problem (uniqueness), and
3. the solution depends continuously on the data (stability).

In Chapter 2, our goal is to study numerical methods for solving ill-posed (due to the lack of stability), non-linear operator equations in a real Hilbert space $\mathcal{H}$:

$$F\left(x\right) = 0,\ F : \mathcal{H} \to \mathcal{H}. \tag{1.1}$$

We require that our operator F is Fréchet differentiable [2] and we assume that the following inequality holds:

$$\left\langle F'\left(h\right)g, g\right\rangle \geq 0, \quad \forall h, g \in \mathcal{H}. \tag{1.2}$$

Assumption (1.2) guarantees that the operator F is monotone (which will be very important as we carry out our convergence analysis), but it does not imply that the problem is stable [3]. We consider a practically interesting 1D inverse gravimetry problem [4, 5] and show that, due to its ill-posedness, the problem cannot be solved by classical Newton's Method. Hence, we introduce Iteratively Regularized Newton's [3] procedure (IRN) and discuss theoretical and numerical aspects of its application.

In Chapter 3, we focus our attention on a general class of nonlinear operator equations (1.1), for which we can no longer demonstrate monotonocity; assumption (1.2) is not satisfied for many important operators. Hence, the IRN method is no longer justified. As opposed to the IRN algorithm, convergence of Iteratively Regularized Gauss-Newton [6–8] procedure (IRGN) can be proven for the case where F is a general nonlinear operator under some special assumption on the level of ill-posedness (the so-called source-type condition) [6, 9]. Note, that another important advantage of utilizing the IRGN algorithm is that in this algorithm, one may consider the nonlinear operator F acting on two different real Hilbert spaces, say $\mathcal{H}_1$ and $\mathcal{H}_2$. In this case, the discretization would generally produce the Jacobian that is $m \times n$ ($m \neq n$), i.e., no longer a square matrix and so Newton's method (IRN in our case) is not applicable.

CHAPTER 2

THEORETICAL AND NUMERICAL STUDY OF NONLINEAR INVERSE PROBLEMS WITH MONOTONE OPERATORS

To show what the lack of stability means from a numerical standpoint, we consider the nonlinear operator F in the following form:

$$F(x) := \int_a^b \mathcal{K}(t,s,x(s))ds - f(t), \quad F: L_2[a,b] \to L_2[a,b]. \tag{2.1}$$

One of the applications of (1.1) with F defined by (2.1) is in gravitational sounding theory [5]. In this application, the kernel $\mathcal{K}$ takes the form:

$$\mathcal{K}(t,s,x(s)) := \frac{\rho}{4\pi} \ln\left[\frac{(t-s)^2+H^2}{(t-s)^2+(H-x(s))^2}\right]. \tag{2.2}$$

In the resulting operator equation, $f(t)$ represents the measured data (the vertical component of the gravitational field), and $x(s)$ is to be calculated (the interface between two media of different densities). Here, ρ is the density of the sources of the gravitational field in the domain $D := \{a \le t \le b,\ -H \le t \le -H + x(t)\}$, where a, b and H are parameters of the domain; this is known as the 1D Gravimetry problem:

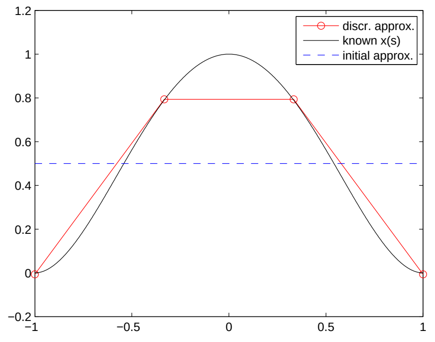
$$F(x) := \frac{\rho}{4\pi} \int_a^b \ln\left[\frac{(t-s)^2+H^2}{(t-s)^2+(H-x(s))^2}\right] ds - f(t) = 0. \tag{2.3}$$

2.1 Failure of Classical Newton's Method

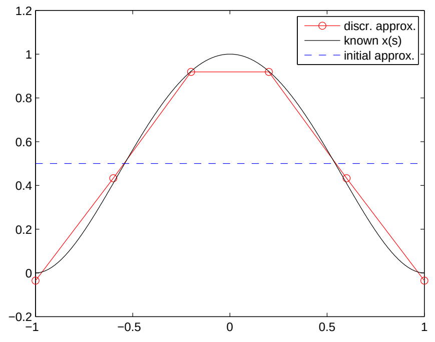
Initially we discretize and attempt to solve by the classical Newton method:

$$x_{k+1} = x_k - [F'(x_k)]^{-1} F(x_k), \quad x_0 \in \mathcal{H}. \tag{2.4}$$

The graphical demonstrations presented in Figure 2.1 illustrate the stability of iterations (2.4), utilizing 4 and 6 grid points, respectively, and how seemingly nice the classical Newton Method appears to be for finding a stable solution.



(a) only 4 grid points

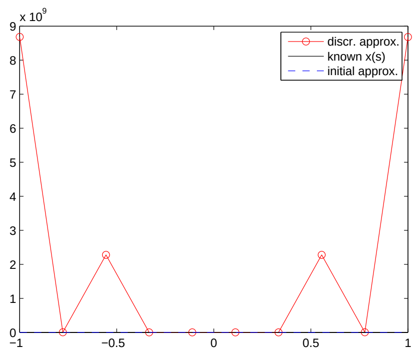


(b) only 6 grid points

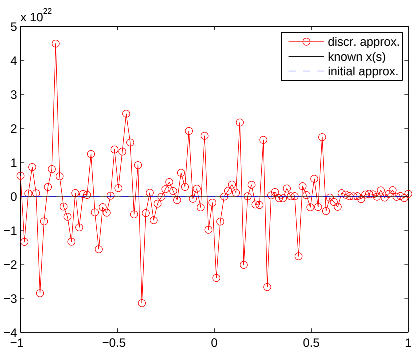
Figure 2.1. Newton's Method, 1D Gravimetry, 4 and 6 Grid Points

Note: Using only a few grid points, we seem to have stability with the classical Newton Method

But, since we are approximating an infinite dimensional problem on such a coarse grid, our accuracy suffers. We would expect that increasing the number of grid points will produce stable solutions with better accuracy, so we proceed with a modest increase to 10 points.



(a) 10 grid points



(b) 100 grid points

Figure 2.2. Newton's Method, 1D Gravimetry, 10 and 100 Grid Points

Note: Divergence with the classic Newton Method

The graph in Figure 2.1(a) illustrates the instability of (2.4) with this modest increase. We may be tempted to believe that the instability with 10 points is an anomaly, so we test this theory with 100 grid points as well. The accuracy and stability are much worse with an increase to 100 grid points and the disastrous results are demonstrated in Figure 2.1(b).

Note, that at each step of the Newton method we have to solve a linear system with the Jacobian matrix $F'(x_k)$ so the accuracy of this step will depend on the conditioning of this matrix.

Table 2.1. Condition Number of Jacobian Using Classical Newton Method

Dimension of F'	Cond$(F'(x_4))$
4	$2.615441 \cdot 10^{2}$
6	$8.664735 \cdot 10^{3}$
8	$2.782761 \cdot 10^{5}$
10	$8.579651 \cdot 10^{6}$
100	$8.685722 \cdot 10^{18}$
1000	$6.445224 \cdot 10^{20}$

The condition numbers for $F'(x_k)$, shown in Table 2.1, indicate that for the original, infinite dimensional equation, $F'(x_k)$ is not invertible. But, by discretizing with only a few grid points (originally, only 4 and 6), we actually make our problem artificially stable, i.e., we provide regularization by discretization. However, we can never expect to reasonably approximate an infinite dimensional problem based on so few grid points.

2.2 Introduction of Auxiliary Problem

As a means of improving the conditioning of the subsequent Jacobian matrix for our equation, we introduce an auxiliary problem:

$$F_\tau(x) := F(x) + \tau(x - \xi) = 0, \quad \tau > 0, \ \xi \in \mathcal{H}. \tag{2.5}$$

The Newton scheme for equation (2.5) is as follows:

$$x_{k+1} = x_k - \left[F'(x_k) + \tau I\right]^{-1} \left(F(x_k) + \tau \left(x_k - \xi\right)\right), \;\; x_0 \in \mathcal{H}. \tag{2.6}$$

Why do we have hope for a stable solution of the auxiliary problem $F_\tau(x) = 0$? The answer lies in the condition number of the resulting Jacobians, $F'(x_k) + \tau I$, at each iteration of this regularized Newton method. Table 2.2 displays the condition numbers as a result of allowing the parameter τ to be 0 (no regularization), 0.01 and 0.1, respectively. For consistency in our comparisons, we examine the results after four iterations of (2.6).

Table 2.2. Condition Number of Jacobian Using Newton Method with Regularization

Dimension of F'	Cond($F'(x_4)$)	Cond($F'(x_4) + 0.01I$)	Cond($F'(x_4) + 0.1I$)
4	$2.615441 \cdot 10^{2}$	$1.140308 \cdot 10^{2}$	$2.502910 \cdot 10^{1}$
6	$8.664735 \cdot 10^{3}$	$2.977130 \cdot 10^{2}$	$2.566069 \cdot 10^{1}$
8	$2.782761 \cdot 10^{5}$	$2.857222 \cdot 10^{2}$	$2.555857 \cdot 10^{1}$
10	$8.579651 \cdot 10^{6}$	$2.804867 \cdot 10^{2}$	$2.543360 \cdot 10^{1}$
100	$8.685722 \cdot 10^{18}$	$2.575277 \cdot 10^{2}$	$2.450715 \cdot 10^{1}$
1000	$6.445224 \cdot 10^{20}$	$2.548297 \cdot 10^{2}$	$2.439423 \cdot 10^{1}$

As one can see, the condition numbers are improving as the values of τ go up. Theoretically, what makes the auxiliary operator equation (2.5) better than the original operator equation (1.1)? It is clear from Table 2.2, that for the original infinite dimensional problem (1.1), the Jacobian $F'(x)$ is not boundedly invertible at all. We granted a sort of "artificial stability" when we discretized with only a few grid points; but it cannot provide reasonable accuracy for an infinite problem. At the same time, the derivative $F'_\tau(x) = F'(x) + \tau I$, under assumption (1.2) is boundedly invertible and the following estimate holds:

$$\left\| \left[F'(x) + \tau I\right]^{-1} \right\| \leq \frac{1}{\tau}. \tag{2.7}$$

However, when we apply the Newton Method to the auxiliary/ regularized equation (2.5), the process will converge to the solution of (2.5); which is different from the solution of the original 1D Gravimetry problem (1.1). Clearly, for very small values of the parameter τ, the solution to the auxiliary problem will be close to the solution of the original problem, but as τ gets larger (to make our process stable), the solution to this problem will deviate from the original problem. If, however, τ is "too small", we fail to gain stability as we seek a solution. With the following method we are able to resolve this dilemma, thus finding a solution to the original operator equation.

2.3 Iteratively Regularized Newton Method (IRN)

In general, for any ill-posed operator equation of the first kind, we may try to utilize the corresponding auxiliary equation (2.5), thus borrowing its stability, while simultaneously finding a solution to the original operator equation (1.1). To overcome the problem of the Newton scheme applied to the auxiliary problem (2.5) converging to a solution that is different from the one of the original problem, we drive the regularization parameter τ to zero as we iterate. To that end, we create a sequence $\{\tau_k\}$, where $\tau_k \to 0$ as $k \to \infty$. Hence we arrive at a new scheme, which we refer to as the Iteratively Regularized Newton Method (IRN) (see [3] and references therein):

$$x_{k+1} = x_k - \left[F'(x_k) + \tau_k I\right]^{-1} \left[F(x_k) + \tau_k \left(x_k - \xi\right)\right], \;\; x_0, \xi \in \mathcal{H}. \tag{2.8}$$

Now, we expect x_k to converge to the solution of the original operator equation which, for the remainder of the paper, will be referred to as y. To implement the IRN algorithm, we begin by rewriting (2.8) as:

$$\left[F'\left(x_k\right) + \tau_k I\right] p_k = -\left\{F\left(x_k\right) + \tau_k \left(x_k - \xi\right)\right\}, \qquad x_0, \xi \in \mathcal{H}, \tag{2.9}$$

where $p_k = x_{k+1} - x_k$. Consider the term $F'(x_k)\,p_k$ on the left-hand side of (2.9):

$$F'(x)\,p = \int_b^a \mathcal{K}'_x(t,s,x(s))\,p(s)\,ds \approx \sum_{j=1}^{N} \mathcal{K}'_x(t_i,s_j,x_j)\,p_j w_j, \qquad i = 1,\ldots,N,$$

where $x(s_j)$ is denoted as x_j, and w_j, $j = 1,\ldots,N$ are the weights associated with the choice of a quadrature method. Denoting $\mathcal{K}'_x(t_i,s_j,x_j)\,w_j := \mathcal{W}_{ij}$, we can write the linear system representing the approximation of $F'(x)\,p$ in matrix form as:

$$\sum_{j=1}^{N} \mathcal{K}'_x(t_i,s_j,x_j)\,p_j w_j \equiv \begin{bmatrix} \mathcal{W}_{11} & \mathcal{W}_{12} & \cdots & \mathcal{W}_{1N} \\ \mathcal{W}_{21} & \mathcal{W}_{22} & \cdots & \mathcal{W}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{W}_{N1} & \mathcal{W}_{M2} & \cdots & \mathcal{W}_{NN} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}.$$

So, for each iteration, k, we can utilize IRN (2.9) to get:

$$\begin{bmatrix} \mathcal{W}_{11}^{(k)} + \tau^{(k)} & \mathcal{W}_{12}^{(k)} & \cdots & \mathcal{W}_{1N}^{(k)} \\ \mathcal{W}_{21}^{(k)} & \mathcal{W}_{22}^{(k)} + \tau^{(k)} & \cdots & \mathcal{W}_{2N}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{W}_{N1}^{(k)} & \mathcal{W}_{N2}^{(k)} & \cdots & \mathcal{W}_{NN}^{(k)} + \tau^{(k)} \end{bmatrix} \begin{bmatrix} p_1^{(k)} \\ p_2^{(k)} \\ \vdots \\ p_N^{(k)} \end{bmatrix} = \begin{bmatrix} g_1^{(k)} \\ g_2^{(k)} \\ \vdots \\ g_N^{(k)} \end{bmatrix},$$

where we choose $\xi = x_0$ and allow the notation: $g_i^{(k)} := -\left[F\left(x_j^{(k)}\right) + \tau^{(k)}\left(x_i^{(k)} - x_i^{(0)}\right)\right]$, where $F\left(x_j^{(k)}\right) := \sum_{j=1}^{N} \mathcal{K}\left(t_i,s_j,x_j^{(k)}\right) p_j^{(k)} w_j - f(t_i)$ for $i = 1,\ldots,N$. Let us write:

$$\left(\mathbf{W}_\tau^{(k)}\right)\left(\mathbf{p}^{(k)}\right) = \mathbf{g}^{(k)}, \tag{2.10}$$

where $\mathbf{W}_\tau^{(k)} := \mathbf{W}^{(k)} + \tau^{(k)}\mathbf{I}$. Therefore, our scheme becomes:

$$\mathbf{p}^{(k)} = \left[\mathbf{W}_\tau^{(k)}\right]^{-1} \mathbf{g}^{(k)}, \qquad \mathbf{x}^{(k+1)} = \mathbf{p}^{(k)} - \mathbf{x}^{(k+1)}.$$

We will present the convergence analysis for the IRN scheme, but before we do so, let us examine some of the numerical results from applying the IRN algorithm to the 1D Gravimetry problem (2.3) with a model solution of $x(s) = (1 - s^2)^2$. Generally, $x(s)$ will be unknown (since this is the solution we seek). However, we first solve our problem with simulated (rather than real) data to test the algorithm.

2.4 Numerical Results for 1D Gravimetry Problem

The numerical results are given for three regularization sequences with different rates of convergence:

$$\tau_k := \frac{\tau_0}{\ln(e+k)},\ \ \tau_k := \frac{\tau_0}{k+1}\ \text{ and } \tau_k := \frac{\tau_0}{e^k}.$$

The sequences are listed from slowest to fastest convergence rates, respectively. Later, we discuss the importance of the rates of convergence of the regularization sequences as related to the properties of the IRN algorithm.

The results demonstrated in Tables 2.3 and 2.4 are obtained when we define the regularization sequence as $\tau_k := \frac{\tau_0}{\ln(e+k)}$, where τ_0 is the initial regularization parameter and $k := 0, 1, \ldots, n$, (for our demonstrations, we allow the number of iterations, n, to be 8). We pair a particular initial regularization parameter, τ_0, with an initial approximation, x_0. Results include: the relative error after 8 iterations of the IRN algorithm and the condition numbers of the Jacobian for both $k = 0$ and $k = 8$, i.e., $\text{cond}(F'(x_0) + \tau_0 I)$ and $\text{cond}(F'(x_8) + \tau_8 I)$. For comparison, we also include the condition numbers of the Jacobian for the classic Newton Method, $\text{cond}(F'(x_4))$. The figures which accompany Tables 2.3 and 2.4, are graphs which demonstrate the successive approximations after each iteration; this allows a visual, per say, of the convergence.

Table 2.3 shows the results of utilizing the regularization sequence given above and pairing the initial regularization parameter $\tau_0 = 1$ with the initial approximation $x_0 = -0.2$ and of pairing $\tau_0 = 10^{-1}$ with $x_0 = 0$, while Table 2.4 utilizes the same regularization sequence but pairs the initial regularization parameter $\tau_0 = 10^{-2}$ with the initial approximation $x_0 = 0.5$ and pairs $\tau_0 = 10^{-1}$ with $x_0 = 1.5$. The different pairings demonstrate the balance between the initial approximation of x_0 and the initial regularization parameter τ_0.

An initial approximation which is "further away" from our model solution will require larger values of the initial regularization parameter to achieve stabilty as we iterate. However, if x_0 is "closer" to the model solution, then we are able to obtain stability with smaller values of τ_0.

Table 2.3. Condition Numbers and Relative Error for IRN and Log Regularization Parameter

Results for particular pairs of initial regularization parameter, τ_0, and initial solution, x_0.

τ_k	τ_0	x_0	Rel. error	cnd$(F'(x_4))$	cnd$(F'(x_0)+\tau_0 I)$	cnd$(F'(x_8)+\tau_8 I)$
$\frac{\tau_0}{\ln(e+k)}$	1	-0.2	$3.971 \cdot 10^{-1}$	$2.482 \cdot 10^{19}$	$2.634 \cdot 10^{0}$	$5.932 \cdot 10^{0}$
	10^{-1}	0	$1.178 \cdot 10^{-1}$	$2.285 \cdot 10^{19}$	$1.868 \cdot 10^{1}$	$5.554 \cdot 10^{1}$

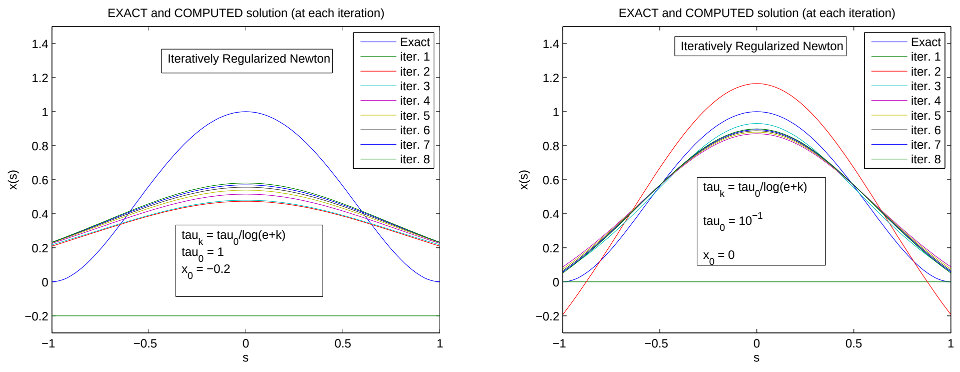


Figure 2.3. Iterative Approximations which Accompany Table 2.3, $\tau_k = \frac{\tau_0}{\ln(e+k)}$

Table 2.4. Condition Numbers and Relative Error for IRN and Logarithmic Regularization Parameter

Results for particular pairs of initial regularization parameter, τ_0, and initial solution, x_0.

τ_k	τ_0	x_0	Rel. error	$\text{cnd}(F'(x_4))$	$\text{cnd}(F'(x_0)+\tau_0 I)$	$\text{cnd}(F'(x_8)+\tau_8 I)$
$\frac{\tau_0}{\ln(e+k)}$	10^{-2}	0.5	$2.845 \cdot 10^{-2}$	$9.252 \cdot 10^{18}$	$2.200 \cdot 10^{2}$	$6.038 \cdot 10^{2}$
	10^{-1}	1.5	$1.929 \cdot 10^{-1}$	$4.167 \cdot 10^{19}$	$4.117 \cdot 10^{1}$	$5.572 \cdot 10^{1}$

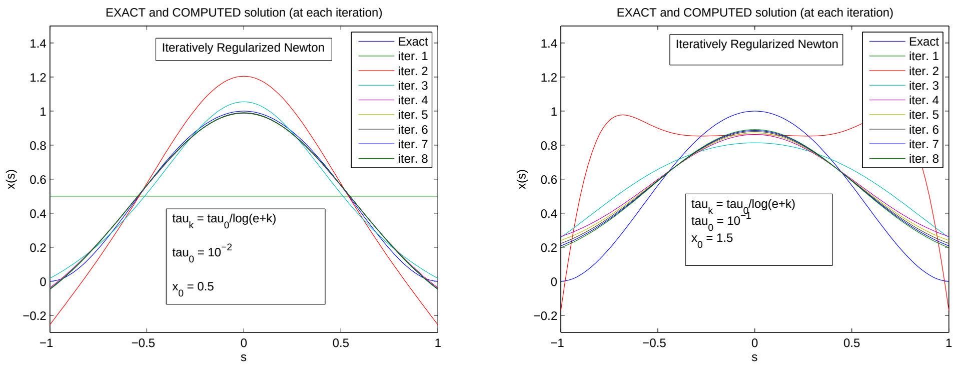


Figure 2.4. Iterative Approximations which Accompany Table 2.4, $\tau_k = \frac{\tau_0}{\ln(e+k)}$

Similar results are shown allowing the regularization sequence, $\{\tau_k\}$, to be defined as $\tau_k := \frac{\tau_0}{k+1}$. Table 2.5 shows the results of utilizing this regularization sequence and pairing the initial regularization parameter $\tau_0 = 1$ with the initial approximation $x_0 = -0.2$ and of pairing $\tau_0 = 10^{-1}$ with $x_0 = 0$:

Table 2.5. Condition Numbers and Relative Error for IRN and Power Regularization Parameter

Results for particular pairs of initial regularization parameter, τ_0, and initial solution, x_0.

τ_k	τ_0	x_0	Rel. error	cond$(F'(x_4))$	cond$(F'(x_0)+\tau_0 I)$	cond$(F'(x_8)+\tau_8 I)$
$\frac{\tau_0}{k+1}$	1	-0.2	$2.259 \cdot 10^{-1}$	$1.761 \cdot 10^{19}$	$2.634 \cdot 10^{0}$	$1.939 \cdot 10^{1}$
	10^{-1}	0	$4.475 \cdot 10^{-1}$	$2.268 \cdot 10^{19}$	$1.868 \cdot 10^{1}$	$2.028 \cdot 10^{2}$

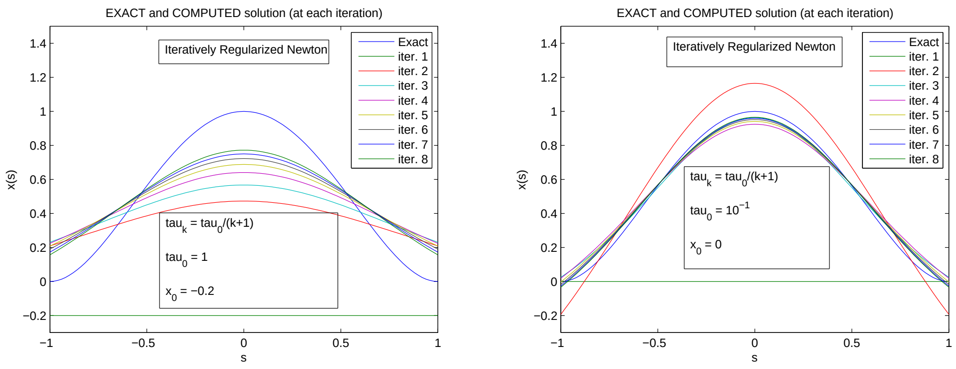


Figure 2.5. Iterative Approximations which Accompany Table 2.5, $\tau_k = \frac{\tau_0}{k+1}$

Table 2.6 shows the results of utilizing the regularization sequence above and pairing the initial regularization parameter $\tau_0 = 10^{-2}$ with the initial approximation $x_0 = 0.5$ and of pairing $\tau_0 = 10^{-1}$ with $x_0 = 1.5$.

Table 2.6. Condition Numbers and Relative Error for IRN and Power Regularization Parameter

Results for particular pairs of initial regularization parameter, τ_0*, and initial solution,* x_0*.*

τ_k	τ_0	x_0	Rel. error	cond$(F'(x_4))$	cond$(F'(x_0)+\tau_0 I)$	cond$(F'(x_8)+\tau_8 I)$
$\frac{\tau_0}{k+1}$	10^{-2}	0.5	$2.289 \cdot 10^{-2}$	$6.811 \cdot 10^{18}$	$2.200 \cdot 10^{2}$	$2.318 \cdot 10^{3}$
	10^{-1}	1.5	$6.716 \cdot 10^{-2}$	$7.661 \cdot 10^{18}$	$4.117 \cdot 10^{1}$	$2.028 \cdot 10^{2}$

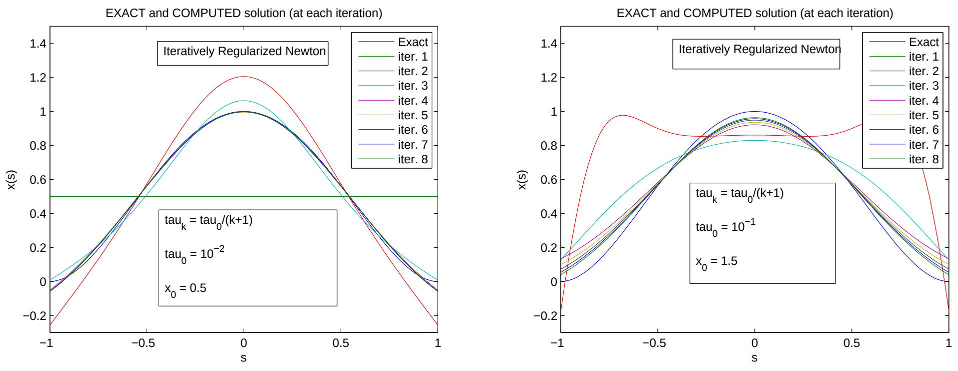


Figure 2.6. Iterative Approximations which Accompany Table 2.6, $\tau_k = \frac{\tau_0}{k+1}$

For our final regularization sequence, $\{\tau_k\}$, defined as $\tau_k := \frac{\tau_0}{e^k}$ we obtain the results shown in Tables 2.7 and 2.8. Table 2.7 shows the results of utilizing this regularization sequence and pairing $\tau_0 = 1$ with $x_0 = -0.2$ and of pairing $\tau_0 = 10^{-1}$ with $x_0 = 0$.

Table 2.7. Condition Numbers and Relative Error for IRN and Exponential Regularization Parameter

Results for particular pairs of initial regularization parameter, τ_0, and initial solution, x_0.

τ_k	τ_0	x_0	Rel. error	$\text{cond}(F'(x_4))$	$\text{cond}(F'(x_0)+\tau_0 I)$	$\text{cond}(F'(x_8)+\tau_8 I)$
$\frac{\tau_0}{e^k}$	1	-0.2	$2.374 \cdot 10^{-2}$	$1.188 \cdot 10^{19}$	$2.634 \cdot 10^{0}$	$3.234 \cdot 10^{3}$
	10^{-1}	0	$1.172 \cdot 10^{-2}$	$6.239 \cdot 10^{19}$	$1.868 \cdot 10^{1}$	$3.845 \cdot 10^{4}$

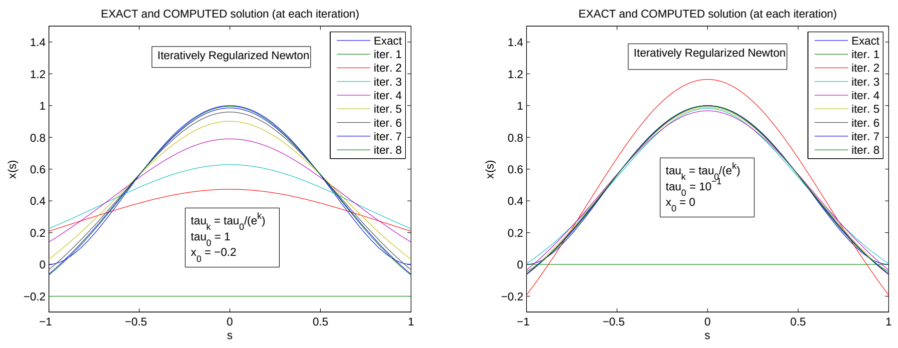


Figure 2.7. Iterative Approximations which Accompany Table 2.7, $\tau_k = \frac{\tau_0}{e^k}$

Table 2.8. Condition Numbers and Relative Error for IRN and Exponential Regularization Parameter

Results for particular pairs of initial regularization parameter, τ_0, and initial solution, x_0.

τ_k	τ_0	x_0	Rel. error	$\text{cond}(F'(x_4))$	$\text{cond}(F'(x_0)+\tau_0 I)$	$\text{cond}(F'(x_8)+\tau_8 I)$
$\frac{\tau_0}{e^k}$	10^{-2}	0.5	$2.374 \cdot 10^{-3}$	$1.188 \cdot 10^{19}$	$2.634 \cdot 10^{2}$	$3.234 \cdot 10^{5}$
	10^{-1}	1.5	$1.058 \cdot 10^{-2}$	$7.594 \cdot 10^{19}$	$4.117 \cdot 10^{1}$	$3.859 \cdot 10^{4}$

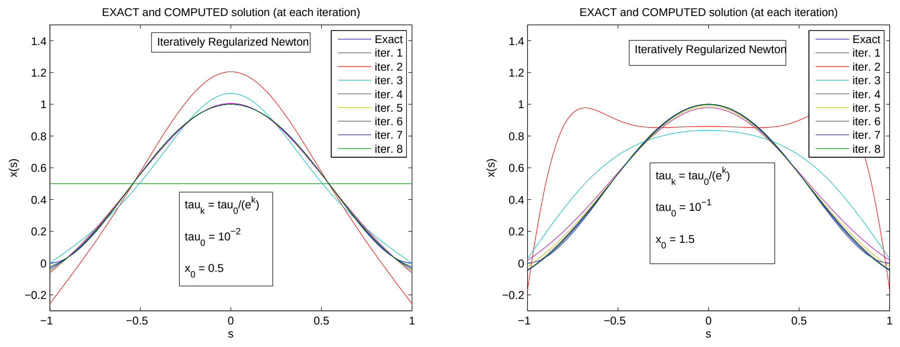


Figure 2.8. Iterative Approximations which Accompany Table 2.8, $\tau_k = \frac{\tau_0}{e^k}$

Table 2.8 also shows the results of utilizing the regularization sequence $\tau_k := \frac{\tau_0}{e^k}$ but pairs the initial regularization parameter $\tau_0 = 10^{-2}$ with the initial approximation $x_0 = 0.5$ and pairs $\tau_0 = 10^{-1}$ with $x_0 = 1.5$.

2.5 Convergence Analysis for IRN

We have offered demonstrations of the numerical results, now to study the theoretical convergence of our IRN method (2.8), i.e., to see under what assumptions $\lim_{k\to\infty} \|x_k - y\| = 0$, we begin by estimating $\|x_k - y\|$ as follows:

$$\|x_k - y\| \le \|x_k - z_k\| + \|z_k - y\|\,, \tag{2.11}$$

where z_k solves the auxiliary problem (2.5) with $\tau = \tau_k$. We wish to show that both norms on the right-hand side of our inequality converge to zero as $k \to \infty$ (see [3] and references therein). In the continuous form, this approach was analyzed in [10].

2.6 Convergence Result for the Auxiliary Problem

In order to show that $\lim_{k\to\infty} \|x_k - y\| = 0$, we must first show that a solution z_k to our auxiliary problem exists. We know that the solution y to our original problem exists by the very nature of the problem (it is the interface between two media of different densities).

However, we can make no such assumptions about the existence of a solution to our auxiliary equation since it is an equation which we have introduced. Next, we show that our sequence $\{z_k\}$ has a weakly convergent subsequence and that the element to which it converges must be the solution y to our original operator equation. Finally we will show that our subsequence is, in fact, strongly convergent to y.

2.6.1 Existence of z_k

To prove the existence of a solution to our auxiliary equation, z_k, we first introduce the following definitions:

Definition 2.6.1. A sequence $\{v_k\}$ in a Hilbert space $\mathcal{H}$ is said to ***converge weakly*** to $v \in \mathcal{H}$ if $v_k \to v$ in the scalar product as $k \to \infty$ (i.e., if $\langle v_k, w\rangle \to \langle v, w\rangle$ for any $w \in \mathcal{H}$). To denote weak convergence, we use the notation: $v_k \rightharpoonup v$. [11]

Definition 2.6.2. Let X, Y be Hilbert spaces amd let $T : X \to Y$. The mapping T is said to be ***hemicontinuous*** at $x \in X$ if and only if $T(x + ty) \rightharpoonup T(x)$, as $t \to 0$, i.e., in terms of the inner product: $\langle T(x + ty), z\rangle \to \langle T(x), z\rangle$as $t \to 0$ for any $z \in X$ [12].

Definition 2.6.3. $F : D \subset H \to H$ is said to be ***monotone*** if $\langle F(x) - F(y), x - y\rangle \geq 0$ for all $x, y \in D$ [3, 12].

We will need these definitions as we formulate and prove the following existence theorem:

Theorem 2.6.4. *[10, 12] If $\langle F'(h) g, g\rangle \geq 0$ for all $h, g \in \mathcal{H}$ and F is hemicontinuous, then our auxiliary equation (2.5),*

$$F_\tau(x) := F(x) + \tau(x - \xi) = 0, \quad \tau > 0, \quad \xi \in \mathcal{H}$$

is uniquely solvable.

Proof. According to [12], our auxiliary equation (2.5) is solvable if F_τ is monotone, hemicontinuos and $\|F_\tau(x)\| \to \infty$ as $\|x\| \to \infty$. Note, since we assume hemicontinuity, it is enough to show monotonicity and that $\|F_\tau(x)\| \to \infty$ as $\|x\| \to \infty$. Monotonicity is an immediate consequence of the variational inequality (1.2). Indeed,

$$\langle F(u) - F(v), u - v\rangle = \langle F'(\eta)(u-v), u-v\rangle \geq 0.$$

To estimate $\|F_\tau(x)\|^2$ from below, we use the fact that this norm has been introduced through the scalar product. In our estimation, we utilize zero in the scalar product, monotonicity of the operator (a direct result of (1.2)) and take advantage of basic "algebra tricks":

$$\begin{aligned}
\|F_\tau(x)\|^2 &= \langle F(x) + \tau(x-\xi), F(x) + \tau(x-\xi)\rangle \\
&= \|F(x)\|^2 + \tau^2\|x-\xi\|^2 + 2\tau\langle F(x) - F(\xi), x-\xi\rangle + 2\tau\langle F(\xi), x-\xi\rangle \\
&\geq \|F(x)\|^2 + \tau^2\|x-\xi\|^2 + 2\tau\langle F(\xi), x-\xi\rangle \\
&\geq \|F(x)\|^2 + \tau^2\|x-\xi\|^2 - 2\tau\|F(\xi)\|\,\|x-\xi\|\,\delta\cdot\frac{1}{\delta} \\
&\geq \|F(x)\|^2 + \tau^2\|x-\xi\|^2 - \left(\tau\|F(\xi)\|\cdot\frac{1}{\delta}\right)^2 - (\delta\|x-\xi\|)^2 \\
&\geq \left(\tau^2-\delta^2\right)\|x-\xi\|^2 - \frac{\tau^2}{\delta^2}\|F(\xi)\|^2 = \left(\tau^2-\delta^2\right)\|x-\xi\|^2 - C,
\end{aligned}$$

where $C = \frac{\tau^2}{\delta^2}\|F(\xi)\|^2$. So for $\delta = \frac{\tau}{2}$,

$$\|F_\tau(x)\|^2 \geq \tfrac{\tau^2}{2}\|x-\xi\|^2 - C \to \infty \text{ as } \|x\| \to \infty.$$

Thus we have shown that (2.5) is solvable, which guarantees the existence of z_τ. □

2.6.2 Weak Convergence of z_k to y

Now that we have proven the existence of z_τ, we present another theorem to prove that $z_\tau \to y$ as $\tau \to 0$. However, before introducing this theorem, we present the following definition and lemma which will be utilized in the proof of the theorem. Note that $x_n \rightharpoonup \xi$ denotes the weak convergence of x_n to ξ as $n \to \infty$.

Definition 2.6.5. The operator F is called ***weakly closed*** if $x_n \rightharpoonup \xi$ and $F(x_n) \to \eta$ imply $\eta = F(\xi)$ [12].

Lemma 2.6.6. A monotone, hemicontinuous operator is weakly closed [12].

Theorem 2.6.7. *Suppose the assumptions of Theorem 2.6.4 are satisfied and there exists a unique solution, y, to $F(x) = 0$. Then*

$$\lim_{\tau \to 0} \|z_\tau - y\| = 0,$$

where z_τ solves (2.5).

Proof. Let us show that $\{z_\tau\}$ is bounded. Indeed,

$$F(z_\tau) - F(y) + \tau (z_\tau - \xi) = 0 \implies F(z_\tau) - F(y) + \tau (z_\tau - y + y - \xi) = 0.$$

Therefore one concludes:

$$F(z_\tau) - F(y) + \tau (z_\tau - y) = \tau (\xi - y).$$

Multiplication in the inner product by $z_\tau - y$ produces:

$$\langle F(z_\tau) - F(y), z_\tau - y \rangle + \tau \langle z_\tau - y, z_\tau - y \rangle = \tau \langle \xi - y, z_\tau - y \rangle.$$

By the monotonicity of F, it means

$$\|z_\tau - y\|^2 \le \langle \xi - y, z_\tau - y \rangle, \tag{2.12}$$

and, hence, it follows from the Cauchy-Schwartz inequality that:

$$\|z_\tau - y\|^2 \le \|\xi - y\| \|z_\tau - y\| \Rightarrow \|z_\tau - y\| \le \|\xi - y\|.$$

Thus, there exists a subsequence $\{z_{\tau_k}\}$ such that: $z_{\tau_k} \rightharpoonup \widetilde{y} \in \mathcal{H}$ as $k \to \infty$. We have found that our subsequence $\{z_{\tau_k}\}$ was weakly convergent to $\widetilde{y} \in \mathcal{H}$. Now we show that $\widetilde{y}$ is, in fact, the solution to our original operator equation $F(x) = 0$, i.e., $\widetilde{y} = y$. Indeed, from our auxiliary equation (2.5) it follows:

$$\|F(z_{\tau_k})\| = \tau_k \|z_{\tau_k} - \xi\| \leq 2\tau_k \|\xi - y\| \text{ as } k \to \infty.$$

Since F is weakly closed, $z_{\tau_k} \rightharpoonup \widetilde{y}$ and $F(z_{\tau_k}) \to 0$ as $k \to \infty$, yield $F(\widetilde{y}) = 0$, by the uniqueness of the solution to our original operator equation (1.1), we may conclude: $\widetilde{y} = y$. Now, as we show strong convergence of z_k to y, note that by (2.12):

$$\|z_{\tau_k} - y\|^2 \leq \langle \xi - y, z_{\tau_k} - y\rangle \to \langle \xi - y, 0\rangle = 0 \text{ as } k \to \infty,$$

since z_{τ_k} converges weakly to y. Hence, $\lim\limits_{k\to\infty} \|z_{\tau_k} - y\| = 0$ as was to be shown. $\square$

2.7 Asymptotic Behavior of Iterations

As a reminder, by the triangle inequality presented in (2.11), to prove that $\lim\limits_{k\to\infty} \|x_k - y\| = 0$, we need to show that $\|x_k - z_k\| \to 0$ and $\|z_k - y\| \to 0$. So far, we have proven that $\lim\limits_{k\to\infty} \|z_k - y\| = 0$, which is a standard functional analysis result that does not depend on our specific numerical method. The remainder of our analysis will involve the iterative sequence $\{x_k\}$ generated through our IRN algorithm. We will concentrate on proving that:

$$\lim_{k\to\infty} \|x_k - z_k\| = 0, \tag{2.13}$$

and with this in mind, we introduce the following Lemma:

Lemma 2.7.1. If F is differentiable and its derivative is Lipschitz continuous, i.e.,

$$\|F'(y) - F'(x)\| \leq L\|y - x\| \text{ for any } x, y \in \mathcal{H}.$$

then,

$$\|F(y) - F(x) - F'(x)(y - x)\| \leq \frac{L}{2}\|y - x\|^2. \tag{2.14}$$

Lemma 2.7.1 allows us to continue without having to assume that our operator F is twice differentiable. The proof is simply a result of the Fundamental Theorem of Calculus. Indeed,

Proof. Fix $x, y \in \mathcal{H}$ and introduce

$$f(t) := F(x + t(y - x)).$$

Clearly, $f(0) = F(x)$ and $f(1) = F(y)$. One has

$$f'(t) = F'(x + t(y - x))(y - x).$$

The Fundamental Theorem of Calculus gives,

$$f(1) - f(0) = \int_0^1 f'(t)\,dt \implies F(y) - F(x) = \int_0^1 F'(x + t(y - x))(y - x)\,dt. \tag{2.15}$$

Now one can estimate as follows:

$$\begin{aligned}\|F(y) - F(x) - F'(x)(y - x)\| &= \left\| \int_0^1 F'(x + t(y - x))(y - x)\,dt - F'(x)(y - x) \right\| \\ &= \left\| \int_0^1 \{F'(x + t(y - x)) - F'(x)\}\,dt \cdot (y - x) \right\| \\ &\leq \int_0^1 \|F'(x + t(y - x)) - F'(x)\|\,dt \cdot \|y - x\|.\end{aligned}$$

By the Lipschitz continuity of F', one concludes

$$\begin{aligned}\|F(y) - F(x) - F'(x)(y - x)\| &\leq \int_0^1 L\|x + t(y - x) - x\|\,dt \cdot \|y - x\| \\ &= L\|y - x\|^2 \int_0^1 (t)\,dt = \frac{L}{2}\|y - x\|^2,\end{aligned}$$

so as a final result $\|F(x) - F(y) - F'(y)(x - y)\| \leq \frac{L}{2}\|x - y\|^2$, as was to be shown. $\square$

2.7.1 The Main Convergence Theorem

Now, we proceed with the presention and proof of the Convergence Theorem for our IRN method.

Theorem 2.7.2. *Let the following conditions be fulfilled:*

1. *The sequence $\{x_k\}$ is generated by Iteratively Regularized Newton's Method (IRN) [3]:*

$$x_{k+1} = x_k - \left[F'(x_k) + \tau_k I\right]^{-1}(F(x_k) + \tau_k(x_k - \xi)),$$

2. *The exact nonlinear equation $F(x) = 0$ is uniquely solvable in a real Hilbert space $\mathcal{H}$,*

and all conditions of Theorem 7 hold,

3. *The operator F is Fréchet differentiable and the conditions of Lemma 2.7.1 hold,*

4. *The regularization sequence, $\{\tau_k\}$, satisfies the assumptions:*

$$\tau_k > 0, \;\; \tau_k \searrow 0, \;\; \frac{\tau_k - \tau_{k+1}}{\tau_k \tau_{k+1}} \leq \lambda,$$

5. *Assume that τ_0 is chosen such that : $\|x_0 - z_0\| \leq l\tau_0$, where $l := \frac{1}{L(\lambda\tau_0+1)}$,*

6. *Let λ satisfy: $2L(\lambda\tau_0 + 1) C\lambda \leq 1$, where $C = 2\|\xi - y\|$,*

Then $\|z_k - x_k\| \leq l\tau_k$, where z_k is a solution to:

$$F_{\tau_k}(x) := F(x) + \tau_k (x - \xi) = 0, \; \xi \in \mathcal{H}.$$

and $\lim\limits_{k\to\infty} \|x_k - y\| = 0$, with y being a unique solution to $F(x) = 0$.

Remark 2.7.3. Before proceeding with a proof of the IRN Convergence Theorem, we will determine whether Assumption 4 can be realistically satisfied. We will examine the three types of regularization sequences utilized when we demonstrated our numerical results, namely: $\tau_k := \frac{\tau_0}{\ln(e+k)}$, $\tau_k := \frac{\tau_0}{(1+k)^p}$, and $\tau_k := \frac{\tau_0}{e^{\alpha k}}$, where $\tau_0 > 0$, $\alpha > 0$ and $0 < p < \infty$. These regularization sequences were specifically chosen based on their rates of convergence (from slowest to fastest, respectively). Later we will offer a discussion and demonstration on the importance of the speed of convergence of the regularization sequences as it relates to the stability of our approximations.

For each of our sequences it is obvious that $\tau_k > 0$ and $\tau_k \searrow 0$ so for Assumption 4 to be satisfied we need to determine whether $\frac{\tau_k - \tau_{k+1}}{\tau_k \tau_{k+1}}$ is bounded.

2.7.2 IRN Assumption 4 and Logarithmic Sequence

For the first sequence under examination: $\tau_k := \frac{\tau_0}{log(e+k)}$, one has

$$\begin{aligned}\lim_{k\to\infty}\frac{\tau_k-\tau_{k+1}}{\tau_k\tau_{k+1}} &= \lim_{k\to\infty}\left[\frac{\tau_0}{\log(e+k)}-\frac{\tau_0}{\log(e+k+1)}\right]\cdot\left[\frac{\log(e+k)\log(e+k+1)}{\tau_0^2}\right]\\ &= \lim_{k\to\infty}\left[\frac{\log(e+k+1)-\log(e+k)}{\log(e+k)\log(e+k+1)}\right]\cdot\left[\frac{\log(e+k)\log(e+k+1)}{\tau_0}\right]\\ &= \frac{1}{\tau_0}\lim_{k\to\infty}\left[\log\left(\frac{e+k+1}{e+k}\right)\right] = \frac{1}{\tau_0}\lim_{k\to\infty}\left[\log\left(1+\frac{1}{e+k}\right)\right] = 0.\end{aligned}$$

Thus, for $\tau_k := \frac{\tau_0}{\ln(e+k)}$, the sequence $\frac{\tau_k-\tau_{k+1}}{\tau_k\tau_{k+1}}$ is convergent and thus bounded and, therefore, *Assumption 4* can be realistically attained.

2.7.3 IRN Assumption 4 and Power Sequence

For our next sequence, $\tau_k := \frac{\tau_0}{(1+k)^p}$:

$$\begin{aligned}\frac{\tau_k-\tau_{k+1}}{\tau_k\tau_{k+1}} &= \left(\frac{\tau_0}{(1+k)^p}-\frac{\tau_0}{(2+k)^p}\right)\cdot\left(\frac{(1+k)^p(2+k)^p}{\tau_0^2}\right)\\ &= \frac{(2+k)^p-(1+k)^p}{(1+k)^p(1+k)^p}\cdot\frac{(1+k)^p(1+k)^p}{\tau_0} = \frac{(2+k)^p-(1+k)^p}{\tau_0}.\end{aligned}$$

To determine boundedness, we restrict our attention to the convergence of the numerator given above:

$$\begin{aligned}\lim_{k\to\infty}\left[(2+k)^p-(1+k)^p\right] &= \lim_{k\to\infty}\left[\left(\frac{2+k}{1+k}\right)^p-1\right](1+k)^p\\ &= \lim_{k\to\infty}\left[\left(1+\frac{1}{1+k}\right)^p-1\right](1+k)^p = \lim_{k\to\infty}\frac{\left(1+\frac{1}{1+k}\right)^p-1}{\left(\frac{1}{1+k}\right)^p}.\end{aligned}$$

Let $t := \frac{1}{1+k}$, which implies,

$$\lim_{k\to\infty}\frac{\left(1+\frac{1}{1+k}\right)^p-1}{\left(\frac{1}{1+k}\right)^p} = \lim_{t\to 0^+}\frac{(1+t)^p-1}{(t)^p}.$$

Now, for $p=1$:

$$\lim_{t\to 0^+}\frac{(1+t)^p-1}{(t)^p} = \lim_{t\to 0}\frac{1+t-1}{t} = 1.$$

While for $p \neq 1$: we have the indeterminate form: $\frac{0}{0}$, so, by L'Hospital's:

$$\lim_{t\to 0^+} \tfrac{(1+t)^p - 1}{(t)^p} = \lim_{t\to 0^+} \tfrac{p(1+t)^{p-1}}{p(t)^{p-1}}.$$

So, the limit takes the form:

$$\lim_{t\to 0^+} \tfrac{t^{1-p}}{(1+t)^{1-p}} = \begin{cases} 0, & 0 < p < 1 \\ \infty, & p > 1 \end{cases}.$$

Hence, for $\tau_k = \frac{\tau_0}{(1+k)^p}$, $\frac{\tau_k - \tau_{k+1}}{\tau_k \tau_{k+1}}$ is convergent for $0 < p \leq 1$, and therefore bounded. At the same time, it is unbounded for $p > 1$.

2.7.4 IRN Assumption 4 and Exponential Sequence

The last sequence under examination, $\tau_k := \frac{\tau_0}{e^{\alpha k}}$, has the fastest rate of convergence; we consider:

$$\begin{aligned} \lim_{k\to\infty} \frac{\tau_k - \tau_{k+1}}{\tau_k \tau_{k+1}} &= \lim_{k\to\infty} \left[\left(\frac{\tau_0}{e^{\alpha k}} - \frac{\tau_0}{e^{\alpha(k+1)}} \right) \cdot \left(\frac{e^{\alpha k} \cdot e^{\alpha(k+1)}}{\tau_0^2} \right) \right] \\ &= \lim_{k\to\infty} \frac{1}{e^{\alpha k}} \left(1 - \frac{1}{e^{\alpha}} \right) \cdot \frac{e^{\alpha k} e^{\alpha(k+1)}}{\tau_0} = \lim_{k\to\infty} \frac{\left(1 - \frac{1}{e^{\alpha}}\right) e^{\alpha(k+1)}}{\tau_0} \longrightarrow \infty. \end{aligned}$$

Thus, we see that $\tau_k := \frac{\tau_0}{e^{\alpha k}}$ does not satisfy Assumption 4 (which emphasizes that Assumption 4 is a sufficient but not necessary condition for convergence).

2.7.5 Proof of IRN Convergence Theorem

We shall prove that if the assumptions presented in Theorem 2.7.2 are satisfied, then $\|z_k - x_k\| \leq l\tau_k$, where z_k is a solution to:

$$F_{\tau_k}(x) := F(x) + \tau_k (x - \xi) = 0, \; \xi \in \mathcal{H}.$$

Proof. Consider, $x_{k+1} - z_{k+1}$, where we utilize the fact that z_k solves the auxiliary equation (2.5) to introduce a special form of zero:

$$x_{k+1} - z_{k+1} \;=\; x_k - \left[F'(x_k) + \tau_k I\right]^{-1}\left(F(x_k) + \tau_k(x_k - \xi) - F(z_k) - \tau_k(z_k - \xi)\right) - z_{k+1},$$

next, we add and subtract to introduce more zeros:

$$\begin{aligned} x_{k+1} - z_{k+1} &= x_k - \left[F'(x_k) + \tau_k I\right]^{-1} \\ &\quad \times \left(F'(x_k)(x_k - z_k) + \tau_k(x_k - z_k) - F'(x_k)(x_k - z_k) + F(x_k) - F(z_k)\right) - z_k + z_k - z_{k+1}. \end{aligned}$$

So, factoring and utilizing matrix identity properties, we obtain:

$$x_{k+1} - z_{k+1} = -\left[F'(x_k) + \tau_k I\right]^{-1}\left(F(x_k) - F(z_k) - F'(x_k)(x_k - z_k)\right) + z_k - z_{k+1}.$$

By Lemma 2.7.1,

$$\left\|F(z_k) - F(x_k) - F'(x_k)(x_k - z_k)\right\| \le \frac{L}{2}\left\|x_k - z_k\right\|^2,$$

which implies,

$$\left\|x_{k+1} - z_{k+1}\right\| \le \frac{L}{2\tau_k}\left\|x_k - z_k\right\|^2 + \left\|z_k - z_{k+1}\right\|.$$

As we try to find an upper bound for $\|z_k - z_{k+1}\|$, recall:

$$F(z_k) + \tau_k(z_k - \xi) = 0 \implies F(z_{k+1}) + \tau_{k+1}(z_{k+1} - \xi) = 0.$$

So, we proceed by subtracting the left side of each equation above and adding zero as we demonstrate below:

$$F(z_k) \;-\; F(z_{k+1}) \;+\; \tau_k(z_k - \xi) \;-\; \tau_{k+1}(z_{k+1} - \xi) + \tau_k(z_{k+1} - \xi) - \tau_k(z_{k+1} - \xi) \;=\; 0,$$

and by making the proper substitutions, distributing and then factoring, we obtain:

$$F'(\eta_k)(z_k - z_{k+1}) + \tau_k (z_k - z_{k+1}) + (\tau_k - \tau_{k+1})(z_{k+1} - \xi) = 0.$$

By the result above,

$$[F'(\eta_k) + \tau_k I](z_k - z_{k+1}) = (\tau_k - \tau_{k+1})(\xi - z_{k+1}),$$

which implies,

$$z_k - z_{k+1} = [F'(\eta_k) + \tau_k I]^{-1} (\tau_k - \tau_{k+1})(\xi - z_{k+1}).$$

So,

$$\|z_k - z_{k+1}\| \leq \frac{(\tau_k - \tau_{k+1}) \|\xi - z_{k+1}\|}{\tau_k} \leq \frac{2(\tau_k - \tau_{k+1}) \|\xi - y\|}{\tau_k}.$$

Hence,

$$\|x_{k+1} - z_{k+1}\| \leq \frac{L}{2\tau_k} \|x_k - z_k\|^2 + \frac{2(\tau_k - \tau_{k+1}) \|\xi - y\|}{\tau_k}. \tag{2.16}$$

Proceeding by induction, we'll verify $\|x_{k+1} - z_{k+1}\| \leq l\tau_{k+1}$:

As we indicated in our assumptions, we have chosen τ_0 such that $\|x_0 - z_0\| \leq l\tau_0$, now assume by induction:

$$\|x_n - z_n\| \leq l\tau_n \quad \text{for} \quad n \leq k.$$

Next, we will verify that it is true for $n = k+1$. Allowing $C := 2\|\xi - y\|$, based on our inductive step and (2.16), we have:

$$\|x_{k+1} - z_{k+1}\| \leq \left[\frac{L}{2\tau_k} l^2 \tau_k^2 + \frac{C(\tau_k - \tau_{k+1})}{\tau_k}\right] \frac{\tau_{k+1}}{\tau_{k+1}} = \left[\frac{Ll^2\tau_k}{2\tau_{k+1}} \tau_k^2 + \frac{C(\tau_k - \tau_{k+1})}{\tau_k \tau_{k+1}}\right] \tau_{k+1}.$$

From our assumptions, note that:

$$\tfrac{\tau_k - \tau_{k+1}}{\tau_k \tau_{k+1}} \leq \lambda \quad \text{implies} \quad \tfrac{\tau_k}{\tau_{k+1}} \leq \lambda\tau_0 + 1.$$

So,

$$\|x_{k+1} - z_{k+1}\| \leq \left[\tfrac{Ll^2(\lambda\tau_0 + 1)}{2} + C\lambda\right] \tau_{k+1}.$$

We will show that our choice of l guarantees:

$$\tfrac{Ll^2(\lambda\tau_0+1)}{2} + C\lambda \le l.$$

According to Assumption 5 of Theorem 7,

$$l := \tfrac{1}{L(\lambda\tau_0+1)} \quad\text{and}\quad 2CL\lambda\left(\lambda\tau_0+1\right) \le 1.$$

Hence,

$$\begin{aligned}\frac{L\left(\lambda\tau_0+1\right)}{2}\cdot\frac{1}{L^2\left(\lambda\tau_0+1\right)^2} + C\lambda - \frac{1}{L\left(\lambda\tau_0+1\right)} &= -\frac{1}{2L\left(\lambda\tau_0+1\right)} + C\lambda \\ &\le -\frac{1}{2L\left(\lambda\tau_0+1\right)} + \frac{1}{2L\left(\lambda\tau_0+1\right)} = 0,\end{aligned}$$

and, therefore,

$$\|x_{k+1} - z_{k+1}\| \le l\tau_{k+1},$$

as was to be shown. $\square$

2.7.6 Monotonicity Assumption

As it has been pointed out in [4], the operator:

$$F\left(x\right) := \frac{\rho}{4\pi}\int_a^b \ln\left[\frac{(t-s)^2+H^2}{(t-s)^2+(H-x(s))^2}\right] ds - f(t) \tag{2.17}$$

is not monotone in L_2. Indeed, if one takes $x_1\left(t\right) = 0$ and $x_2\left(t\right) = 3H$, one concludes:

$$\left\langle F\left(x_2\right) - F\left(x_1\right), x_2 - x_1\right\rangle_{L_2} = \frac{3H\rho}{4\pi}\int_a^b\int_a^b \ln\left[\frac{(t-s)^2+H^2}{(t-s)^2+4H^2}\right] ds\, dt < 0.$$

However, as suggested by numerical simulations, it is possible that operator (2.17) is monotone in the domain:

$$D := \left\{x\left(t\right) \in L_2\left[a,b\right] :\ x\left(t\right) \le H - \varepsilon\right\}, \quad \varepsilon > 0,$$

though to the best of our knowledge, theoretical justification of this claim is not available. The above observation explains, at least partly, why the Iteratively Regularized Newton algorithm is efficient for 1D Inverse Gravimetry problem.

CHAPTER 3

THEORETICAL AND NUMERICAL STUDY OF GENERAL NONLINEAR INVERSE PROBLEMS

Now, as was mentioned in Chapter 1, we focus our attention on a general class of nonlinear operators:

$$F : \mathcal{H}_1 \to \mathcal{H}_2, \tag{3.1}$$

where $\mathcal{H}_1$ and $\mathcal{H}_2$ are not necessarily the same Hilbert spaces and monotonicity can no longer be demonstrated. For this class of operators, the IRN method is no longer justified in solving the operator equation $F(x) = 0$ and we will define and investigate the use of a different iteratively regularized procedure, specifically, the Iteratively Regularized Gauss-Newton algorithm. When $\mathcal{H}_1$ and $\mathcal{H}_2$ are different, the equation $F(x) = 0$ is not, in general, solvable. Therefore, it is natural to understand the solution in the sense of least squares when one is trying to minimize the functional $\|F(x)\|^2$. As the result, in place of solving $F(x) = 0$, one ends up dealing with the normal equation:

$$F'^*(x)\,F(x) = 0, \quad F'^*F : \mathcal{H}_1 \to \mathcal{H}_1. \tag{3.2}$$

If we attempt to implement the classical Newton method, then the Fréchet derivative will take the following form:

$$F''^*(x)\,F(x) + F'^*(x)\,F'(x)\,.$$

If the residual is small, the contribution of the first term will decrease as we iterate. Additionally, evaluating $F''(x)$ may not be an easy task for some nonlinear operators. Moreover, the structure of $F''^*F + F'^*F'$ is not as beneficial as the structure of F'^*F', which is always nonnegative (in the sense of variational inequalities). Hence, the idea of the Gauss-Newton method [13] is to remove the first term and to use the following iterative scheme, to approximate the solution to (3.2):

$$x_{k+1} = x_k - \left[F'^*\left(x_k\right) F'\left(x_k\right)\right]^{-1} F'^*\left(x_k\right) F\left(x_k\right), \quad x_0 \in \mathcal{H}_1. \tag{3.3}$$

In the ill-posed case, $F'^*(x)\,F'(x)$ is not boundedly invertible in any neighborhood for the minimizer. Therefore, as in the previous chapter, one has to incorporate iterative regularization into the numerical algorithm. To that end, we consider the iteratively regularized version of the Gauss-Newton procedure (IRGN) [6, 9, 14]:

$$x_{k+1} = x_k - \left[F'^*\left(x_k\right) F'\left(x_k\right) + \tau_k I\right]^{-1} \left\{F'^*\left(x_k\right) F\left(x_k\right) + \tau_k\left(x_k - \xi\right)\right\}, \quad x_0, \xi \in \mathcal{H}_1. \tag{3.4}$$

One can think of (3.4) as the Gauss-Newton method being applied to the regularized minimization problem:

$$\left\|F(x)\right\|^2 + \tau\left\|x - \xi\right\|^2 \underset{x \in \mathcal{H}_1}{\longrightarrow} \min, \quad \tau > 0,$$

with τ being updated at every step. Alternatively, one can write (3.4) in the following form:

$$x_{k+1} = \xi - \left[F'^*\left(x_k\right) F'\left(x_k\right) + \tau_k I\right]^{-1} F'^*\left(x_k\right)\left\{F\left(x_k\right) - F'\left(x_k\right)\left(x_k - \xi\right)\right\}. \tag{3.5}$$

The above form has the advantage that the regularization is only contained in the inverse operator, $\left[F'^*\left(x_k\right) F'\left(x_k\right) + \tau_k I\right]^{-1}$. Hence, one can generalize further and investigate a family of regularized Gauss-Newton algorithms [7]:

$$x_{k+1} = \xi - \theta\left(F'^*\left(x_k\right) F'\left(x_k\right), \tau_k\right) F'^*\left(x_k\right)\left\{F\left(x_k\right) - F'\left(x_k\right)\left(x_k - \xi\right)\right\},$$

with $\theta = \theta(\lambda, \tau)$ being a function of a spectral parameter $\lambda \in [0, N^2]$ and a regularization parameter $\tau \in (0, \infty)$.

3.1 Main Convergence Theorem for IRGN

In this section, we provide convergence analysis for the Iteratively Regularized Gauss-Newton procedure (IRGN):

$$x_{k+1} = \xi - \left[F'^{*}(x_k) F'(x_k) + \tau_k I\right]^{-1} F'^{*}(x_k) \left\{F(x_k) - F'(x_k)(x_k - \xi)\right\}, \qquad (3.6)$$

where the general nonlinear operator, F, is under the level of ill-posedness as stated in Assumption (5) of the following theorem. We will refer to this level of ill-posedness as the "source-type" [6, 9, 14] condition.

Theorem 3.1.1. *Let the following conditions be fulfilled:*

1. *The sequence $\{x_k\}$ is generated by Iteratively Regularized Gauss-Newton Method (IRGN):*

$$x_{k+1} = \xi - \left[F'^{*}(x_k) F'(x_k) + \tau_k I\right]^{-1} F'^{*}(x_k) \left\{F(x_k) - F'(x_k)(x_k - \xi)\right\}.$$

2. *The exact nonlinear equation $F(x) = 0$, $F : \mathcal{H}_1 \to \mathcal{H}_2$, is solvable, not necessarily uniquely.*

3. *F is Fréchet differentiable and its Fréchet derivative F' is Lipschitz continuous:*

$$\|F'(x) - F'(y)\| \le L \|x - y\|, \qquad \forall x, y \in \mathcal{H}_1.$$

4. *The regularization sequence $\{\tau_k\}$ satisfies the assumptions:*

$$\tau_k > 0, \qquad \lim_{k\to\infty} \tau_k = 0, \qquad \sqrt{\frac{\tau_k}{\tau_{k+1}}} \le r,$$

5. *For a solution $\hat{x}$ and some $\xi \in \mathcal{H}_1$ and $w \in \mathcal{H}_2$*

$$(\hat{x} - \xi) \in F'^{*}(\hat{x})\, S, \qquad S := \{w, \|w\| \le \varepsilon\},$$

we assume:

$$L\varepsilon + \sqrt{\frac{L\varepsilon}{2}} \le \frac{1}{r},$$

6. For the initial approximation x_0*, we require:* $\frac{\|x_0-\hat{x}\|}{\sqrt{\tau_0}} \leq l^*$*, where* $l^* := \frac{2(1-Lr\varepsilon)}{Lr}$.

Then the sequence, as defined in Assumption (1), is well-defined and it converges to $\hat{x}$ *at the following rate:*

$$\frac{\|x_k - \hat{x}\|}{\sqrt{\tau_k}} \leq l^*, \qquad k = 0, 1, 2, \ldots .$$

Before we begin the proof of the convergence of the IRGN method, we present the following:

Lemma 3.1.2. Assume that F is a nonlinear, Frechet differentiable operator. Then $F'^*(x)\,F'(x) + \tau I$ is boundedly invertible for any $x \in \mathcal{H}_1$.

Proof. Since $\mathcal{H}_1$ and $\mathcal{H}_2$ are Hilbert spaces, for any $h \in \mathcal{H}_1$ and $\alpha > 0$ one has:

$$\begin{aligned}
\langle (F'^*(x)\,F'(x) + \tau I)\,h, h\rangle &= \langle F'^*(x)\,F'(x)\,h, h\rangle + \tau\,\langle h, h\rangle \\
&= \langle F'(x)\,h, F'(x)\,h\rangle + \tau\,\langle h, h\rangle \\
&= \|F'(x)\,h\|^2 + \tau\,\|h\|^2 \geq \tau\,\|h\|^2 .
\end{aligned}$$

Now, $\langle (F'^*(x)\,F'(x) + \tau I)\,h, h\rangle \geq \tau\,\|h\|^2$ implies:

$$\left\| [F'^*(x)\,F'(x) + \tau I]^{-1} \right\| \leq \frac{1}{\tau}.$$

Hence, iterations (3.6) are well-defined. □

We utilize this result as we prove the convergence of our IRGN method which is applicable for a general nonlinear operator F.

3.1.1 Proof of Theorem 10

Proof. According to Assumption (1),

$$x_{k+1} - \hat{x} = \xi - \hat{x} - [F'^*(x_k)\,F'(x_k) + \tau_k I]^{-1}\,F'^*(x_k)\,\{F(x_k) - F'(x_k)\,(x_k - \xi)\}.$$

Define $G(\hat{x}, x_k) := F(x_k) - F'(x_k)(x_k - \hat{x})$. Since $F(\hat{x}) = 0$, by Lemma 8 in Chapter 2 one obtains:

$$\|G(\hat{x}, x_k)\| = \|F(x_k) - F'(x_k)(x_k - \hat{x})\| \leqslant \frac{L}{2}\|x_k - \hat{x}\|^2, \tag{3.7}$$

and

$$F(x_k) - F'(x_k)(x_k - \xi) = G(\hat{x}, x_k) + F'(x_k)(\xi - \hat{x}).$$

So, according to our scheme and properties of the inner product space,

$$\begin{aligned}\|x_{k+1} - \hat{x}\| &= \left\|\xi - \hat{x} - [F'^*(x_k)F'(x_k) + \tau_k I]^{-1} F'^*(x_k)\{G(\hat{x}, x_k) + F'(x_k)(\xi - \hat{x})\}\right\| \\ &\leq \left\|[F'^*(x_k)F'(x_k) + \tau_k I]^{-1} F'^*(x_k) G(\hat{x}, x_k)\right\| \\ &\quad + \left\|\left\{I - [F'^*(x_k)F'(x_k) + \tau_k I]^{-1} F'^*(x_k)F'(x_k)\right\}(\xi - \hat{x})\right\|.\end{aligned}$$

Clearly, we can state:

$$\begin{aligned}&I - [F'^*(x_k)F'(x_k) + \tau_k I]^{-1} F'^*(x_k)F'(x_k) \\ &\qquad = [F'^*(x_k)F'(x_k) + \tau_k I]^{-1}\{(F'^*(x_k)F'(x_k) + \tau_k I) - F'^*(x_k)F'(x_k)\} \\ &\qquad = \tau_k [F'^*(x_k)F'(x_k) + \tau_k I]^{-1}.\end{aligned}$$

Based on the source condition, $\hat{x} - \xi = F'^*(\hat{x})v$ for some $v \in S$. Therefore, it follows from the above that

$$\begin{aligned}\|x_{k+1} - \hat{x}\| &\leq \left\|[F'^*(x_k)F'(x_k) + \tau_k I]^{-1} F'^*(x_k) G(\hat{x}, x_k)\right\| \\ &\quad + \tau_k \left\|[F'^*(x_k)F'(x_k) + \tau_k I]^{-1} F'^*(\hat{x}) w\right\|.\end{aligned}$$

Introduce the notations:

$$T_1 := \left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} F'^{*}(x_k) G(\hat{x}, x_k) \right\|,$$
$$T_2 := \left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} F'^{*}(\hat{x}) w \right\|.$$

By polar decomposition $F'(x_k) = U\left(F'^{*}(x_k) F'(x_k)\right)^{\frac{1}{2}}$, where U is a partial isometry. Thus one derives:

$$\begin{aligned}
\left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} F'^{*}(x_k) \right\| &= \left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} \left\{ U\left(F'^{*}(x_k) F'(x_k)\right)^{\frac{1}{2}} \right\}^{*} \right\| \\
&= \left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} \left(F'^{*}(x_k) F'(x_k)\right)^{\frac{1}{2}} U^{*} \right\| \\
&\le \left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} \left(F'^{*}(x_k) F'(x_k)\right)^{\frac{1}{2}} \right\| \\
&= \sup_{\lambda \in \sigma(F'^{*}F')} \frac{\sqrt{\lambda}}{\lambda + \tau_k},
\end{aligned}$$

and we obtain:

$$\left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} F'^{*}(x_k) \right\| \le \frac{1}{2\sqrt{\tau_k}}. \tag{3.8}$$

According to (3.7) and (3.8), one estimates T_1 as:

$$T_1 \le \left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} F'^{*}(x_k) \right\| \left\| G(\hat{x}, x_k) \right\| = \frac{L}{4\sqrt{\tau_k}} \left\| x_k - \hat{x} \right\|^2.$$

For T_2 one has:

$$\begin{aligned}
T_2 \le &\left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} \left(F'(\hat{x}) - F'(x_k) \right)^{*} w \right\| \\
&\qquad + \left\| \left[F'^{*}(x_k) F'(x_k) + \tau_k I \right]^{-1} F'^{*}(x_k) w \right\|.
\end{aligned}$$

Since $\|F'(\hat{x}) - F'(x_k)\| \leq L\|x_k - \hat{x}\|$ and $\|w\| \leq \varepsilon$, by Lemma 3.1.2 we can make the following deduction:

$$T_2 \leq \frac{1}{\tau_k} L\|x_k - \hat{x}\|\varepsilon + \frac{\varepsilon}{2\sqrt{\tau_k}}.$$

Now, from the above one concludes,

$$\|x_{k+1} - \hat{x}\| \leq T_1 + \tau_k T_2 \leq \frac{L}{4\sqrt{\tau_k}}\|x_k - \hat{x}\|^2 + L\|x_k - \hat{x}\|\varepsilon + \frac{\sqrt{\tau_k}}{2}\varepsilon.$$

Recall that $\|x_0 - \hat{x}\| \leq l^*\sqrt{\tau_0}$; by induction hypothesis, let $\|x_j - \hat{x}\| \leq l^*\sqrt{\tau_j}$, for $j = 1, 2, \ldots, k$. In what follows, we will verify that:

$$\|x_{j+1} - \hat{x}\| \leq l^*\sqrt{\tau_{j+1}}.$$

For ease of notation, introduce: $\gamma_j := \frac{\|x_j - \hat{x}\|}{\sqrt{\tau_j}}$; induction hypothesis implies: $\gamma_j \leq l$. Now

$$\gamma_{j+1} = \frac{\|x_{j+1} - \hat{x}\|}{\sqrt{\tau_{j+1}}} \leq \frac{L\sqrt{\tau_j}}{4\tau_j\sqrt{\tau_{j+1}}}\|x_j - \hat{x}\|^2 + \frac{L\|x_j - \hat{x}\|\varepsilon}{\sqrt{\tau_{j+1}}} + \frac{\sqrt{\tau_j}}{2\sqrt{\tau_{j+1}}}\varepsilon.$$

Recalling that $\lim\limits_{k\to\infty}\tau_k = 0$, that τ_k is monotonically decreasing, i.e., $\sqrt{\frac{\tau_k}{\tau_{k+1}}} \leq r$ and our induction hypothesis:

$$\gamma_{j+1} \leq \frac{Lr}{4}\gamma_j^2 + L\gamma_j r\varepsilon + \frac{r\varepsilon}{2} \leq \frac{Lr}{4}l^{*^2} + Ll^* r\varepsilon + \frac{r\varepsilon}{2}.$$

Let us show that under Assumption (6) of Theorem 10, it follows that :

$$\frac{Lr}{4}l^{*^2} - (1 - Lr\varepsilon)\, l^* + \frac{r\varepsilon}{2} \leq 0.$$

Indeed, since $l^* = \frac{2(1-Lr\varepsilon)}{Lr}$, one derives:

$$\gamma_{j+1} - l^* \leq \frac{Lr}{4}l^{*^2} - (1 - Lr\varepsilon)\, l^* + \frac{r\varepsilon}{2} = -\frac{(1 - Lr\varepsilon)^2}{Lr} + \frac{r\varepsilon}{2}.$$

According to Assumption (5) of Theorem (10), one has: $L\varepsilon + \sqrt{\frac{L\varepsilon}{2}} \leq \frac{1}{r}$, which yields:

$$\sqrt{\frac{L\varepsilon}{2}} \leq \frac{1 - Lr\varepsilon}{r} \Leftrightarrow \sqrt{\frac{r\varepsilon}{2}} \leq \frac{1 - Lr\varepsilon}{\sqrt{Lr}} \Rightarrow \frac{r\varepsilon}{2} \leq \frac{(1 - Lr\varepsilon)^2}{Lr}.$$

Thus, we may conclude:

$$\gamma_{k+1} - l^* \leq -\frac{(1 - Lr\varepsilon)^2}{Lr} + \frac{r\varepsilon}{2} \leq 0,$$

as was to be shown. $\square$

3.2 Computation Algorithm for 2D Inverse Gravimetry Problem

As an example of the applicability of the IRGN method to a two-dimensional nonlinear integral equation of the first kind, consider the following inverse Gravimetry problem: [4, 15]

$$A(x) := g\triangle\sigma \int_a^b \int_c^d \mathcal{K}(t, v, s, u, x(s,u))\, ds\, du = f(t,v), \quad t \in \left[\tilde{a}, \tilde{b}\right], \; v \in \left[\tilde{c}, \tilde{d}\right], \tag{3.9}$$

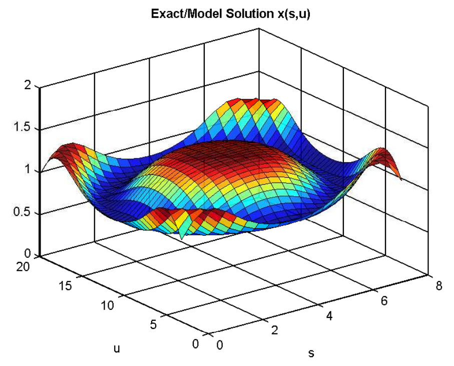
where $F(x) := A(x) - f$ and the kernel, $\mathcal{K}$, is defined as follows:

$$\mathcal{K}(t, v, s, u, x(s,u)) := \left\{ \frac{1}{\left[(s-t)^2 + (u-v)^2 + x^2(s,u)\right]^{\frac{1}{2}}} - \frac{1}{\left[(s-t)^2 + (u-v)^2 + H^2\right]^{\frac{1}{2}}} \right\}.$$

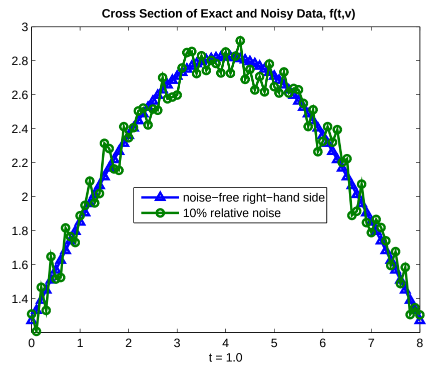
Here, $x(s,u)$ is the interface (which we wish to reconstruct) between two media of different densities, g is the gravitational constant, $\triangle\sigma$ is the density jump on the interface, H is an a priori information about the domain and $f(t,v)$ is measured surface data. As was explained in Chapter 2, in regards to the IRN method, it is necessary to test the accuracy and stability of the IRGN algorithm using simulated data. To this end, we will solve the forward two-dimensional Gravimetry problem (3.9) first by defining :

$$x(s,u) := \frac{1}{4} \cos\left((4s-2)^2 + (4u-2)^2\right) + 1. \tag{3.10}$$

The solution to this forward problem will represent our simulated data for the inverse problem. After using the IRGN method to reconstruct (3.10) with this $f\left(t,v\right)$, we will add some random noise, δ, to the data. We will then attempt to reconstruct (3.10) using this noise-contaminated data, $f_{\delta}\left(t,v\right)$.



(a) exact solution $x\left(s,u\right)$



(b) cross-section of f and f_{δ}

Figure 3.1. Exact/Model Solution and Data

We solve the corresponding forward problem for the 2D Gravimetry equation (3.9) with $x = x\left(s,u\right)$ defined by (3.10) using a very fine grid on $\left[\tilde{a},\tilde{b}\right] \times \left[\tilde{c},\tilde{d}\right]$. We implement the two dimensional analog of the composite trapezoidal quadrature rule and this high accuracy scheme allows us to consider the values that we obtain as the exact measurement data $f\left(t,v\right)$.

3.2.1 IRGN Algorithm and Inverse 2D Gravimetry Problem

To implement the IRGN algorithm, we begin by rewriting (3.4) as:

$$\left[F'^{*}\left(x_k\right)F'\left(x_k\right)+\tau_k I\right]p_k = -\left\{F'^{*}\left(x_k\right)F\left(x_k\right)+\tau_k\left(x_k-\xi\right)\right\}, \qquad x_0,\xi \in \mathcal{H}_1, \tag{3.11}$$

where $p_k = x_{k+1} - x_k$.

At the first step, we partition the intervals $[a,b]$, $[c,d]$, $\left[\tilde{a},\tilde{b}\right]$ and $\left[\tilde{c},\tilde{d}\right]$ with mesh points $\{s_j\}_1^J$, $\{u_l\}_1^L$, $\{t_i\}_1^I$ and $\{v_n\}_1^N$, respectively. We use a convergent quadrature formula such that:

$$F(x) \approx \sum_{j=1}^{J}\sum_{l=1}^{L} \mathcal{K}\left(t_i, v_n, s_j, u_l, x_{j,l}\right) w_{j,l} - f_\delta\left(t_i, v_n\right), \qquad i = 1, 2, \ldots, I, \quad n = 1, 2, \ldots, N, \tag{3.12}$$

where $w_{j,l}$ are the weights of the quadrature rule. As the result, we have a system of $M = I \times N$ nonlinear equations with $K = J \times L$ unknowns. Now, we solve (3.12) using IRGN. Note that, in this case,

$$F'(x)\, p = \int_a^b \int_c^d \mathcal{K}'_x\left(t, v, s, u, x(s,u)\right) p(s,u)\, ds\, du \approx \sum_{j=1}^{J}\sum_{l=1}^{L} \mathcal{K}'_x\left(t_i, v_n, s_j, u_l, x_{j,l}\right) p_{j,l} w_{j,l},$$

where

$$\mathcal{K}'_x\left(t, v, s, u, x(s,u)\right) := -g\triangle\sigma \left\{ \frac{x(s,u)}{\left[(s-t)^2 + (u-v)^2 + x^2(s,u)\right]^{\frac{3}{2}}} \right\}.$$

In the 1D Gravimetry problem as mentioned earlier, this was described as a two-dimensional array multiplied by a vector. However, in the 2D problem, $F'(x)\,p$ is approximated by an array of dimension $I \times N \times J \times L$ multiplied by an array of dimension $J \times L$. Let us denote the 4D array as $\mathbf{W}$ and the 2D array, by $\mathbf{p}$, respectively. Additionally, for (3.11), we have 2D arrays, $\mathbf{F}$ and $(\mathbf{x} - \xi)$. From this point, we choose $\xi = \mathbf{x}^{(0)}$. Before we can solve (3.11) numerically, we need to obtain an IRGN form that is dimensionally equivalent to (2.10). Therefore, we proceed by compressing the 4D array, $\mathbf{W}^{(k)}$, to an equivalent 2D array, $\mathbb{W}^{(k)}$, and also compressing the 2D arrays, $\mathbf{p}^{(k)}$, $\mathbf{F}^{(k)}$ and $\left(\mathbf{x}^{(k)} - \mathbf{x}^{(0)}\right)$ to equivalent vector forms $\mathbb{P}^{(k)}$, $\mathbb{F}^{(k)}$ and $\left(\mathbb{X}^{(k)} - \mathbb{X}^{(0)}\right)$, respectively.

The procedure for compressing a 2D array (matrix) to a 1D array (vector) is organized as follows: consider, for example, $\mathbf{A}_{J\times L}$, a J by L matrix (for clarity, as we discuss compressions, we reference the dimension of the matrix as subnotation), the first row of the matrix, which coordinates with $j = 1$ and $l = 1, \ldots, L$, becomes the first L elements of the vector $\mathbb{A}_K$, the second row of the matrix ($j = 2$ and $l = 1, \ldots, L$) becomes elements $L + 1$ through $2L$ of $\mathbb{A}_K$. In general, the jth row becomes elements $(j-1)\,L + 1$ through jL of $\mathbb{A}_K$. We continue, until finally, the last row of the matrix, $\mathbf{A}_{J\times L}$, becomes elements $(J-1)L+1$ through JL of $\mathbb{A}_K$:

$$\mathbf{A}_{J\times L} := \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,L} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,L} \\ \vdots & \vdots & & \vdots \\ a_{J,1} & a_{J,2} & \cdots & a_{J,L} \end{bmatrix}$$

$$\mathbf{A}_{J\times L} \quad \xrightarrow{\text{compress}} \quad \mathbb{A}_K,$$

where

$$\mathbb{A}_K := \left[\overbrace{a_{1,1} \quad a_{1,2} \quad \cdots \quad a_{1,L}} \; \overbrace{a_{2,1} \quad a_{2,2} \quad \cdots \quad a_{2,L}} \; \cdots \; \overbrace{a_{J,1} \quad a_{J,2} \quad \cdots \quad a_{J,L}} \right]^T.$$

Similarly, we compress a 4D array to a 2D array. Indeed, consider a four-dimensional array $\mathbf{B}_{I\times N\times J\times L} = [b_{i,n,j,l}]$, if we fix $i = 1$ and $n = 1$, then we are reduced to a 2D array, $\mathbf{B}_{1\times 1\times J\times L} = [b_{1,1,j,l}]$ which, when compressed, becomes the first row of our new 2D array, $\mathbb{B}_{M\times K}$. From fixing $i = 1$ and $n = 1, \ldots, N$ we obtain the first N rows of $\mathbb{B}_{M\times K}$ and from fixing $i = 2$ and $n = 1, \ldots, N$ we obtain rows $N+1$ through $2N$ of $\mathbb{B}_{M\times K}$, we continue until, finally, we fix $i = I$ and $n = 1, \ldots, N$ and obtain rows $(I-1)N+1$ through $I \times N = M$ of $\mathbb{B}_{M\times K}$. In this manner, the four dimensional array $\mathbf{B}_{I\times N\times J\times L}$, becomes the two dimensional array $\mathbb{B}_{M\times K}$, i.e.,

$$\mathbf{B}_{I\times N\times J\times L} \quad \xrightarrow{\text{compress}} \quad \mathbb{B}_{M\times K},$$

where

$$\mathbb{B}_{M\times K} := \begin{bmatrix} \overbrace{\begin{matrix} b_{1,1,1,1} & \cdots & b_{1,1,1,L} \\ b_{1,2,1,1} & \cdots & b_{1,2,1,L} \\ \vdots & & \vdots \\ b_{i,n,1,1} & \cdots & b_{i,n,1,L} \\ \vdots & & \vdots \\ b_{I,N,1,1} & \cdots & b_{I,N,1,L} \end{matrix}} & \overbrace{\begin{matrix} b_{1,1,2,1} & \cdots & b_{1,1,2,L} \\ b_{1,2,2,1} & \cdots & b_{1,2,2,L} \\ \vdots & & \vdots \\ b_{i,n,2,1} & \cdots & b_{i,n,2,L} \\ \vdots & & \vdots \\ b_{I,N,2,1} & \cdots & b_{I,N,2,L} \end{matrix}} & \begin{matrix} \cdots \\ \cdots \\ \\ \cdots \\ \\ \cdots \end{matrix} & \overbrace{\begin{matrix} b_{1,1,J,1} & \cdots & b_{1,1,J,L} \\ b_{1,2,J,1} & \cdots & b_{1,2,J,L} \\ \vdots & & \vdots \\ b_{i,n,J,1} & \cdots & b_{i,n,J,L} \\ \vdots & & \vdots \\ b_{I,N,J,1} & \cdots & b_{I,N,J,L} \end{matrix}} \end{bmatrix}.$$

With the above notations for our new 2D and 1D arrays, we now have an IRGN form that is dimensionally equivalent to (2.10) :

$$\left[\left(\mathbb{W}^{(k)}_{M\times K}\right)^* \left(\mathbb{W}^{(k)}_{M\times K}\right) + \tau^{(k)}\mathbb{I}_{K\times K}\right]\mathbb{P}^{(n)}_K = -\left\{\left(\mathbb{W}^{(k)}_{M\times K}\right)^* F^{(k)}_M + \tau^{(k)}\left(\mathbb{X}^{(k)} - \mathbb{X}^{(0)}\right)_K\right\}. \tag{3.13}$$

We may proceed with the implementation of the IRGN algorithm. Since we have compressed our arrays, the approximate solution, $\mathbb{X}$, will be in the form of a vector. Once the iterative process is complete, we uncompress $\mathbb{X}$ to return to the matrix form of $\mathbf{x}$, so that our approximate solution will represent a surface.

3.3 Noise-free Simulation Results

If one takes a very good initial approximation, say $x_0 = 0.5$ or $x_0 = 1.0$ for our particular model, the iterations converge with no regularization; discretization is enough, in this case. However, for a more reasonable initial guess that is further away, say $x_0 = 1.9$, regularization is essential, even for the noise-free experiments. This is rather natural; the less a priori information we have, the more regularization is required. After introducing τ_k, both the relative error and the condition number of the Jacobian improve with increasing values of the initial regularization parameter, τ_0. Hence we gain the stability that we need as demonstrated in Table 3.1 and Figure 3.3 (with $\tau_k = \frac{\tau_0}{e^k}$).

Table 3.1. Condition Numbers and Relative Error for IRGN and Exponential Regularization Parameter

Noise free results ($\delta = 0$) for initial approximation $x_0 = 1.9$ and increasing values of τ_0.

τ_k	x_0	τ_0	k	Rel. error	$\text{cond}(F'^*F'(x_0) + \tau_0 I)$
$\frac{\tau_0}{e^k}$	1.9	0	1	$9.058 \cdot 10^0$	$1.793 \cdot 10^{16}$
		$1 \cdot 10^{-3}$	1	$1.132 \cdot 10^0$	$3.854 \cdot 10^4$
		$1 \cdot 10^{-2}$	2	$8.224 \cdot 10^{-1}$	$3.586 \cdot 10^3$
		$2.5 \cdot 10^{-2}$	3	$6.942 \cdot 10^{-1}$	$1.435 \cdot 10^3$
		$5 \cdot 10^{-2}$	6	$5.161 \cdot 10^{-1}$	$7.180 \cdot 10^2$
		$5 \cdot 10^{-1}$	15	$2.858 \cdot 10^{-2}$	$7.270 \cdot 10^1$
		1	16	$2.866 \cdot 10^{-2}$	$3.685 \cdot 10^1$
		1.6	17	$2.893 \cdot 10^{-2}$	$2.341 \cdot 10^1$
		2	17	$2.880 \cdot 10^{-2}$	$1.892 \cdot 10^1$
		2.5	17	$2.870 \cdot 10^{-2}$	$1.534 \cdot 10^1$

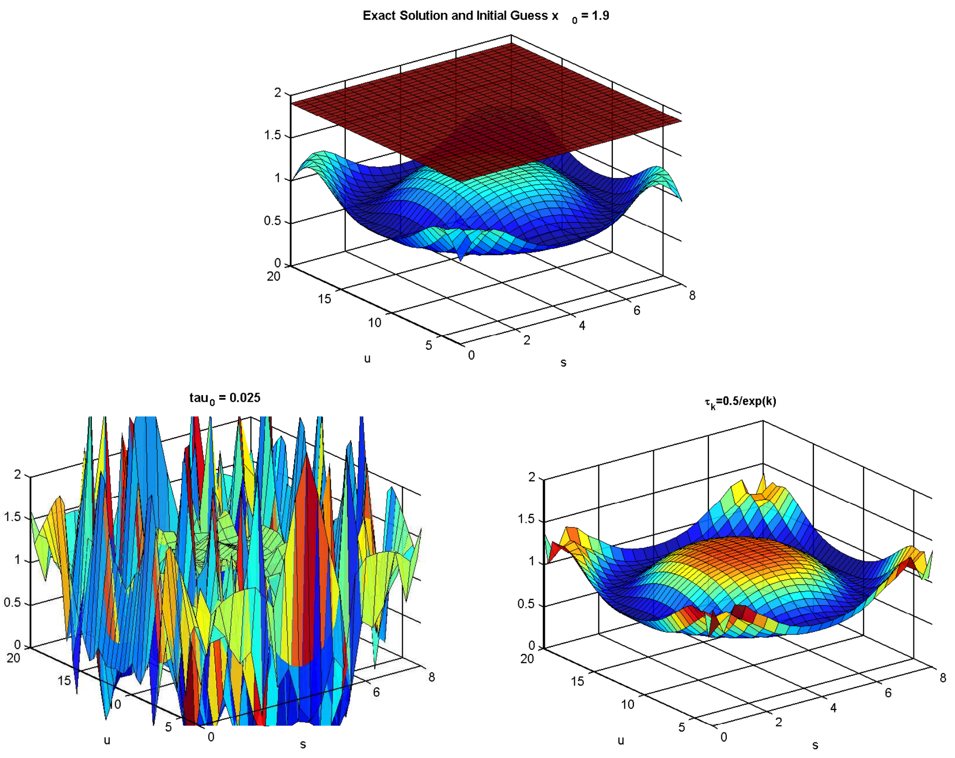


Figure 3.2. Exact $x(s,u)$ and Approximations which Accompany Table 3.1, $\tau_k = \frac{\tau_0}{e^k}$.

For $\tau_0 = 0$; 10^{-3}; 10^{-2}; $2.5 \cdot 10^{-2}$ and $5 \cdot 10^{-2}$, the process is clearly under-regularized and the desired accuracy is not achieved. The value $\tau_0 = 5 \cdot 10^{-1}$ is near optimal, as one can see both from the table and from the accompanying picture. Table 3.1 also illustrates that there is not much danger in over-regularizing since regularization is being done iteratively. In the case where the process is over-regularized, one simply iterates longer before the stopping criteria is met.

Table 3.2. Comparison of Approximations for Different Regularization Parameters

Noise free results ($\delta = 0$) for initial approximation, $x_0 = 1.9$, and $\tau_0 = 5 \cdot 10^{-1}$

τ_0	τ_k	k	Rel. error	$\text{cond}(F'^{*}F'(x_0) + \tau_0 I)$
$5 \cdot 10^{-1}$	$\frac{\tau_0}{\ln(e+k)}$	30	$2.109 \cdot 10^{-1}$	$3.564 \cdot 10^{1}$
	$\frac{\tau_0}{k+1}$	30	$1.662 \cdot 10^{-1}$	$5.375 \cdot 10^{1}$
	$\frac{\tau_0}{e^k}$	15	$2.858 \cdot 10^{-2}$	$7.270 \cdot 10^{1}$
	$\frac{\tau_0}{e^{k^{10}}}$	2	$8.058 \cdot 10^{-1}$	$7.270 \cdot 10^{1}$

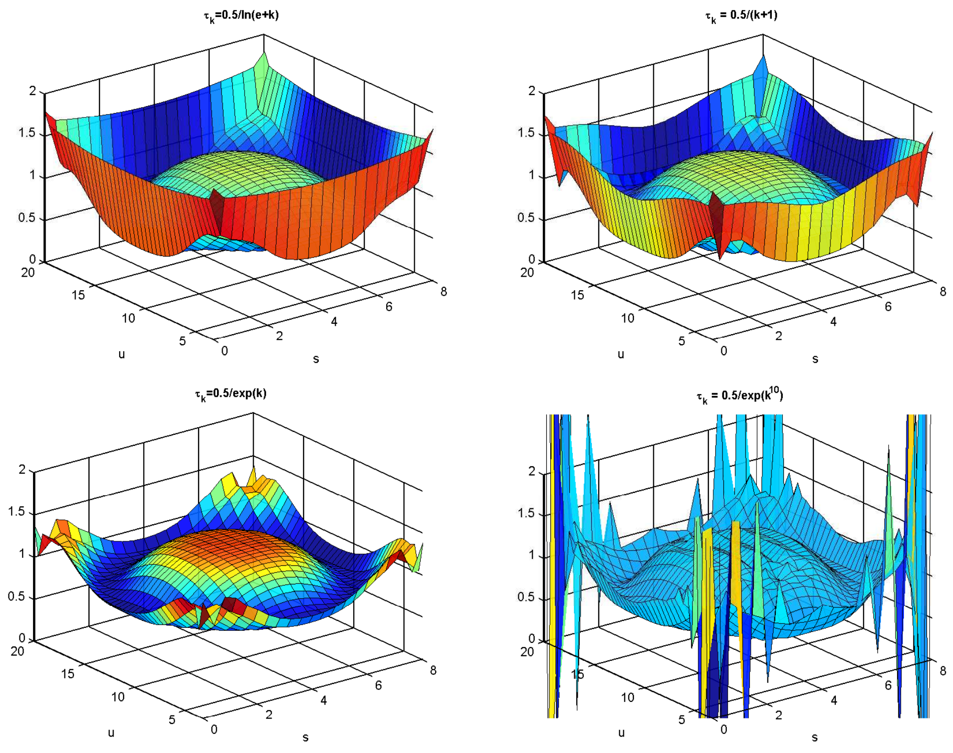


Figure 3.3. Comparison of Approximations for Differentτ_k, Accompanies Table 3.2

In chapter 2, for the 1D Gavimetry problem, we utilized three regularization parameters with different rates of convergence, namely, $\tau_k := \frac{\tau_0}{\ln(e+k)}$, $\tau_k := \frac{\tau_0}{k+1}$ and $\tau_k := \frac{\tau_0}{e^k}$, as we compared the numerical results. In Table 3.2, we compare the effect of the speed of convergence of these and one additional regularization parameter on our iterative results for the 2D Gravimetry problem. For a valid comparison, we hold the initial approximation and initial regularization parameters constant in each simulation; the results are shown in Table 3.2 and the accompanying Figure 3.3. Obviously, our logarithmic parameter, $\tau_k := \frac{\tau_0}{\ln(e+k)}$, has the slowest rate of convergence and, indeed, the associated relative error of the corresponding approximation is the highest with $k = 30$, the maximum number of iterations allowed. With the same number of iterations involved, our power function, $\tau_k := \frac{\tau_0}{k+1}$, has a lower relative error, however, our exponential parameter, $\tau_k := \frac{\tau_0}{e^k}$, has the smallest associated relative error with fewer iterations ($k = 15$). The additional sequence, $\tau_k := \frac{\tau_0}{e^{k^{10}}}$, converges at a faster rate than the exponential. This sequence converges so quickly, that stability is lost before the desired accuracy is attained. This indicates that one cannot drive $\{\tau_k\}$ to zero too fast for the process to remain stable. Note, that condition (4) of Theorem 10 is not satisfied for $\tau_k := \frac{\tau_0}{e^{k^{10}}}$.

CHAPTER 4

A POSTERIORI STOPPING RULE FOR NOISY DATA

In this chapter we consider the family of regularized Gauss-Newton type procedures with $F(x) := A(x) - f$ in the following form:

$$x_{k+1} = \xi - \theta\left(A'^{*}(x_k)\,A'(x_k),\tau_k\right)A'^{*}(x_k)\left\{A(x_k) - f_\delta - A'(x_k)(x_k - \xi)\right\}, \quad x_0, \xi \in \mathcal{H}_1. \tag{4.1}$$

Here, $A : \mathcal{H}_1 \to \mathcal{H}_2$ is a nonlinear operator (of a fairly general structure) between two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, and f_δ is noise contaminated measured data such that:

$$\|f - f_\delta\| \le \delta, \quad \delta > 0.$$

Note that in the original Iteratively Regularized Gauss-Newton scheme [6],

$$\theta(\lambda,\tau) := \frac{1}{\lambda + \tau}.$$

In the general case (4.1), as justified in [7, 16], we impose the following basic conditions on the generating function $\theta = \theta(\lambda,\tau)$:

$$\sup_{\lambda\in[0,N^2]} |\theta(\lambda,\tau)\lambda - 1| \le c, \tag{4.2}$$

$$\sup_{\lambda\in[0,N^2]} |\theta(\lambda,\tau)\lambda - 1|\,\lambda^{\frac{1}{2}} \le c\tau^{\frac{1}{2}}, \tag{4.3}$$

$$\sup_{\lambda\in[0,N^2]} \left|\theta(\lambda,\tau)\sqrt{\lambda}\right| \le c\tau^{-\frac{1}{2}}. \tag{4.4}$$

In (4.2) - (4.4), the constant c is assumed to be the same in order to simplify the presentation. Conditions (4.2) - (4.4) cover three major types of generating functions. Specifically,

1. Functions constructed through Tikhonov regularizations such as, for example, $\mathcal{M}$-times iterated Tikhonov's method:

$$\theta(\lambda,\tau) := \sum_{k=0}^{\mathcal{M}-1} \frac{\tau^k}{(\lambda+\tau)^{k+1}}, \quad \mathcal{M} \in \mathbb{N}.$$

2. Functions performing iterative truncation of the remainder of infinite series representing the inverse operator $(F'^*F')^{-1}$. As an example, one can take the Newton-Landweber algorithm:

$$\theta(\lambda,\tau) := \begin{cases} 1-(1-\mu\lambda)^{\frac{1}{\alpha}}, & \lambda \neq 0 \\ \frac{\mu}{\tau}, & \lambda = 0 \end{cases}, \quad 0 < \mu < \tfrac{2}{N^2}.$$

3. Functions implementing iterative spectral truncation, for example:

$$\theta(\lambda,\tau) := \begin{cases} \frac{1}{\lambda}, & \lambda \geq \tau \\ 0, & 0 \leq \lambda < \tau \end{cases}.$$

In what follows, we introduce a modified Discrepancy Principle stopping rule. Note that the classical Dicrepancy Principle, $\|Ax_\alpha - f_\delta\| = \delta$, was introduced and justified by Morozov [17] for linear, ill-posed problems. A more general Discrepancy Principle, similar to (4.5) below, was proposed in [18], where it was also applied to linear unstable models. For nonlinear problems, the Discrepancy Principle was analyzed in [19]. In this paper, we suggest to terminate the generalized Gauss-Newton type iterations after the first transition of the modified discrepancy through the level $\sigma\delta$:

$$\begin{aligned} &\left\|\sqrt{\tau_{k_\delta}}\theta\left(A'^*(x_{k_\delta})A'(x_{k_\delta}),\tau_{k_\delta}\right)A'^*(x_{k_\delta})\left(A(x_{k_\delta})-f_\delta\right)\right\| \leq \sigma\delta \\ &\qquad < \left\|\sqrt{\tau_k}\theta\left(A'^*(x_k)A'(x_k),\tau_k\right)A'^*(x_k)\left(A(x_k)-f_\delta\right)\right\|, \\ &\qquad\qquad \sigma > 1, \quad 0 \leq k < k_\delta = k_\delta(\delta, f_\delta). \end{aligned} \tag{4.5}$$

We show that for the level of ill-posedness

$$\hat{x} - \xi \in A'^{*}(\hat{x})\, S, \quad S := \{v,\ \|v\| \leq \varepsilon\}, \tag{4.6}$$

the convergence rate $O\left(\sqrt{\tau_k}\right)$, $k = 0, 1, \ldots, k_\delta(\delta, f_\delta)$ is attained under merely the Lipschitz continuity assumption on A', without further restrictions on the nonlinearity of the operator A [20].

To illustrate the convergence rate of $O\left(\sqrt{\tau_k}\right)$ for regularization algorithm (4.1) - (4.5), introduce the notation:

$$\mu_k := \frac{\|x_k - \hat{x}\|}{\sqrt{\tau_k}}, \quad k = 0, 1, \ldots \quad . \tag{4.7}$$

From identities (4.1) and $A(\hat{x}) = f$, one concludes the following: for any $k < k_\delta(\delta, f_\delta)$, one gets:

$$\begin{aligned} x_{k+1} - \hat{x} = -\theta\left(A'^{*}(x_k) A'(x_k), \tau_k\right) A'^{*}(x_k) \left\{A(x_k) - A(\hat{x}) - A'(x_k)(x_k - \hat{x})\right\} \\ - \left[\hat{x} - \xi - \theta\left(A'^{*}(x_k) A'(x_k), \tau_k\right) A'^{*}(x_k) A'(x_k)(\hat{x} - \xi)\right] \\ - \theta\left(A'^{*}(x_k) A'(x_k), \tau_k\right) A'^{*}(x_k)(f - f_\delta). \end{aligned} \tag{4.8}$$

As it has been verified in Lemma 8 in Chapter 2, Lipschitz continuity of the Frchet derivative, A', yields:

$$\|A(x_k) - A(\hat{x}) - A'(x_k)(x_k - \hat{x})\| \leq \frac{L}{2} \|x_k - \hat{x}\|^2.$$

Taking into account source condition 4.6, one derives:

$$\begin{aligned} \hat{x} - \xi - \theta\left(A'^{*}(x_k) A'(x_k), \tau_k\right) A'^{*}(x_k) A'(x_k)(\hat{x} - \xi) \\ = \left[I - \theta\left(A'^{*}(x_k) A'(x_k), \tau_k\right) A'^{*}(x_k) A'(x_k)\right] \left(A'(\hat{x}) - A'(x_k)\right) * v \\ + \left[I - \theta\left(A'^{*}(x_k) A'(x_k), \tau_k\right) A'^{*}(x_k) A'(x_k)\right] A'^{*}(x_k)\, v. \end{aligned} \tag{4.9}$$

Combining (4.8) and (4.9) and using assumptions (4.2) - (4.4), one arrives at the estimate:

$$\|x_{k+1}-\hat{x}\| \le \frac{cL}{2\sqrt{\tau_k}}\|x_k-\hat{x}\|^2 + cL\|x_k-\hat{x}\|\|v\| + c\sqrt{\tau_k}\|v\| + \frac{c\delta}{\sqrt{\tau_k}}. \qquad (4.10)$$

Since $k < k_\delta(\delta, f_\delta)$, according to (4.5) one has

$$\sigma\delta < \|\sqrt{\tau_k}\theta\left(A'^*(x_k)A'(x_k), \tau_k\right)A'^*(x_k)\left(A(x_k)-f_\delta\right)\|, \quad \sigma > 1.$$

The above inequality implies

$$\begin{aligned}\sigma\delta &< \|\sqrt{\tau_k}\theta\left(A'^*(x_k)A'(x_k), \tau_k\right)A'^*(x_k)\left[A(x_k)-A(\hat{x})+f-f_\delta\right]\| \\ &\le \|\sqrt{\tau_k}\theta\left(A'^*(x_k)A'(x_k), \tau_k\right)A'^*(x_k)\left[A'(x_k)(x_k-\hat{x})+A(x_k)-A(\hat{x})-A'(x_k)(x_k-\hat{x})\right]\| \\ &\quad + \|\sqrt{\tau_k}\theta\left(A'^*(x_k)A'(x_k), \tau_k\right)A'^*(x_k)(f-f_\delta)\| \le \sqrt{\tau_k}(c+1)\|x_k-\hat{x}\| \\ &\quad + \sqrt{\tau_k}\frac{c}{\sqrt{\tau_k}}\frac{L\|x_n-\hat{x}\|^2}{2} + \sqrt{\tau_k}\frac{c}{\sqrt{\tau_k}}\delta. \qquad (4.11)\end{aligned}$$

From (4.11), it follows that,

$$(\sigma-c)\delta < \sqrt{\tau_k}(c+1)\|x_k-\hat{x}\| + \frac{L\|x_k-\hat{x}\|^2 c}{2}. \qquad (4.12)$$

Substituting (4.12) into (4.10), one obtains

$$\begin{aligned}\|x_{k+1}-\hat{x}\| &\le \frac{cL}{2\sqrt{\tau_k}}\|x_k-\hat{x}\|^2 + cL\|x_k-\hat{x}\|\|v\| + c\sqrt{\tau_k}\|v\| \\ &\quad + \frac{c\delta}{\sqrt{\tau_k}}\left\{\frac{\sqrt{\tau_k}(c+1)\|x_k-\hat{x}\|}{\sigma-c} + \frac{L\|x_k-\hat{x}\|^2 c}{2(\sigma-c)}\right\} \\ &\le \frac{cL}{2\sqrt{\tau_k}}\left(1+\frac{c}{\sigma-c}\right)\|x_k-\hat{x}\|^2 + c\|x_k-\hat{x}\|\left(L\|v\| + \frac{c+1}{\sigma-c}\right) \\ &\quad + c\sqrt{\tau_k}\|v\|, \qquad k < k_\delta(\delta, f_\delta). \qquad (4.13)\end{aligned}$$

By 4.7, one concludes from 4.13 that

$$\mu_{k+1} \leq \frac{cL}{2}\left(1+\frac{c}{\sigma-c}\right)\left(\frac{\|x_k-\hat{x}\|}{\sqrt{\tau_k}}\right)^2\sqrt{\frac{\tau_k}{\tau_{k+1}}} + \frac{\|x_k-\hat{x}\|}{\sqrt{\tau_k}}\sqrt{\frac{\tau_k}{\tau_{k+1}}}c\left(L\|v\|+\frac{c+1}{\sigma-c}\right)+c\sqrt{\frac{\tau_k}{\tau_{k+1}}}\|v\|. \quad (4.14)$$

If one assumes that $\lim\limits_{k\to\infty}\tau_k=0$ and $\sup\limits_{k\geq 0}\sqrt{\frac{\tau_k}{\tau_{k+1}}}=d<\infty$, then 4.14 yields

$$\mu_{k+1} \leq \mu_k^2\frac{cLd}{2}\left(1+\frac{c}{\sigma-c}\right)+\mu_k cd\left(L\|v\|+\frac{c+1}{\sigma-c}\right)+cd\|v\| := a\mu_k^2+b\mu_k+p. \quad (4.15)$$

Now, using the argument that is similar to the one of Theorem 10, one can show that:

$$\|x_k-\hat{x}\| = O\left(\sqrt{\tau_k}\right), \qquad k=0,1,\ldots,k_\delta\left(\delta,f_\delta\right).$$

4.1 Simulation Results with Noise Added to the Data

To demonstrate the necessity of the stopping rule presented in the previous section, Figure 4.1 shows a graph with a plot of the relative error of our approximations after each iteration, and a plot of the discrepancy. The simulation which produced this figure was run with the exponential regularization parameter, $\tau_k=\frac{\tau_0}{e^k}$, the initial approximation, $x_0=1.9$, the initial regularization parameter, $\tau_0=0.05$ and a relative level of noise, $\delta=15\%$ was added to the data, f_δ. It is desirable for the relative error of the approximations to approach zero. However, since our approximations are generated from noisy data, we run the risk of converging to the solution of the noisy equation if we iterate too long. This would not be an issue in the well-posed case. But when the problem is unstable, even for a small level of noise in the data, the solution of the corresponding "noisy" problem may be very different from the solution we are actually trying to approximate. That is exactly what happened in the course of the experiment illustrated in Figure 4.1. While the discrepancy continues to decrease and to approach zero, at some point during the iterative process, the relative error of the approximations begin to approach the solution of the noisy problem (as demonstrated by the “turn” of the plot for the relative error). This “turn” gives us a visual of $k = k_\delta\left(\delta, f_\delta\right)$.

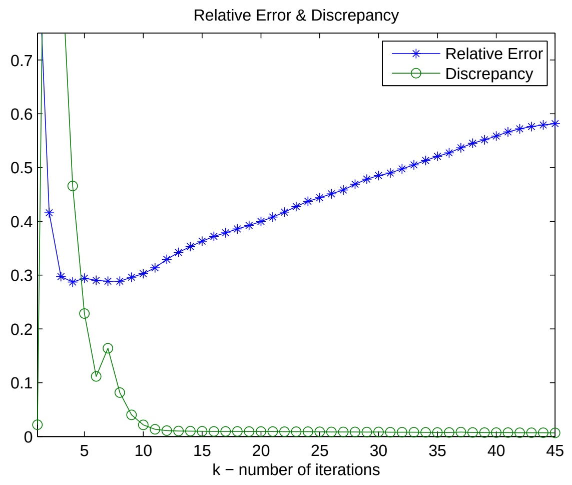


Figure 4.1. Graph of Relative Error and Discrepancy

The plots are produced from simulations with $\tau_k = \frac{\tau_0}{e^k}$, $x_0 = 1.9$, $\tau_0 = 0.05$ *and* $\delta = 15\%$. *(The “turning” point on the plot of relative error indicates* $k_\delta\left(\delta, f_\delta\right)$*.)*

The comparison of Tables 4.1 and 4.2 shows that the simulation results can be improved if the value of the initial regularization parameter and the amount of noise are properly balanced. At the same time, one can observe that the relative error is not overly sensitive to the choice of τ_0. This advantage can also be attributed to the regularization being done iteratively; so that asymptotic behavior of iterations is not considerably affected by the value of τ_0 (as long as it is not too small).

Table 4.1. Increasing Levels of Noise Paired with Increasing Initial Regularization Parameters

Results for $x_0 = 1.9$, level of noise, δ, and initial regularization parameter, τ_0.

τ_k	x_0	δ	τ_0	Rel. error	Discrepancy
$\frac{\tau_0}{e^k}$	1.9	0%	$5 \cdot 10^{-1}$	$3.172 \cdot 10^{-2}$	$1.017 \cdot 10^{-5}$
		5%	$1 \cdot 10^{0}$	$1.513 \cdot 10^{-1}$	$4.613 \cdot 10^{-3}$
		10%	$1.6 \cdot 10^{0}$	$1.879 \cdot 10^{-1}$	$9.044 \cdot 10^{-3}$
		15%	$2 \cdot 10^{0}$	$2.054 \cdot 10^{-1}$	$1.476 \cdot 10^{-2}$
		20%	$2.5 \cdot 10^{0}$	$2.107 \cdot 10^{-1}$	$2.779 \cdot 10^{-2}$

Table 4.2. Increasing Levels of Noise Paired with a Constant Initial Regularization Parameter

Results for $x_0 = 1.9$, level of noise, δ, and initial regularization parameter, $\tau_0 = 5 \cdot 10^{-1}$.

τ_k	x_0	δ	τ_0	Rel. error	Discrepancy
$\frac{\tau_0}{e^k}$	1.9	0%	$5 \cdot 10^{-1}$	$3.172 \cdot 10^{-2}$	$1.017 \cdot 10^{-5}$
		5%	$5 \cdot 10^{-1}$	$1.520 \cdot 10^{-1}$	$4.232 \cdot 10^{-3}$
		10%	$5 \cdot 10^{-1}$	$1.900 \cdot 10^{-1}$	$8.480 \cdot 10^{-3}$
		15%	$5 \cdot 10^{-1}$	$2.168 \cdot 10^{-1}$	$2.360 \cdot 10^{-2}$
		20%	$5 \cdot 10^{-1}$	$2.233 \cdot 10^{-1}$	$2.725 \cdot 10^{-2}$

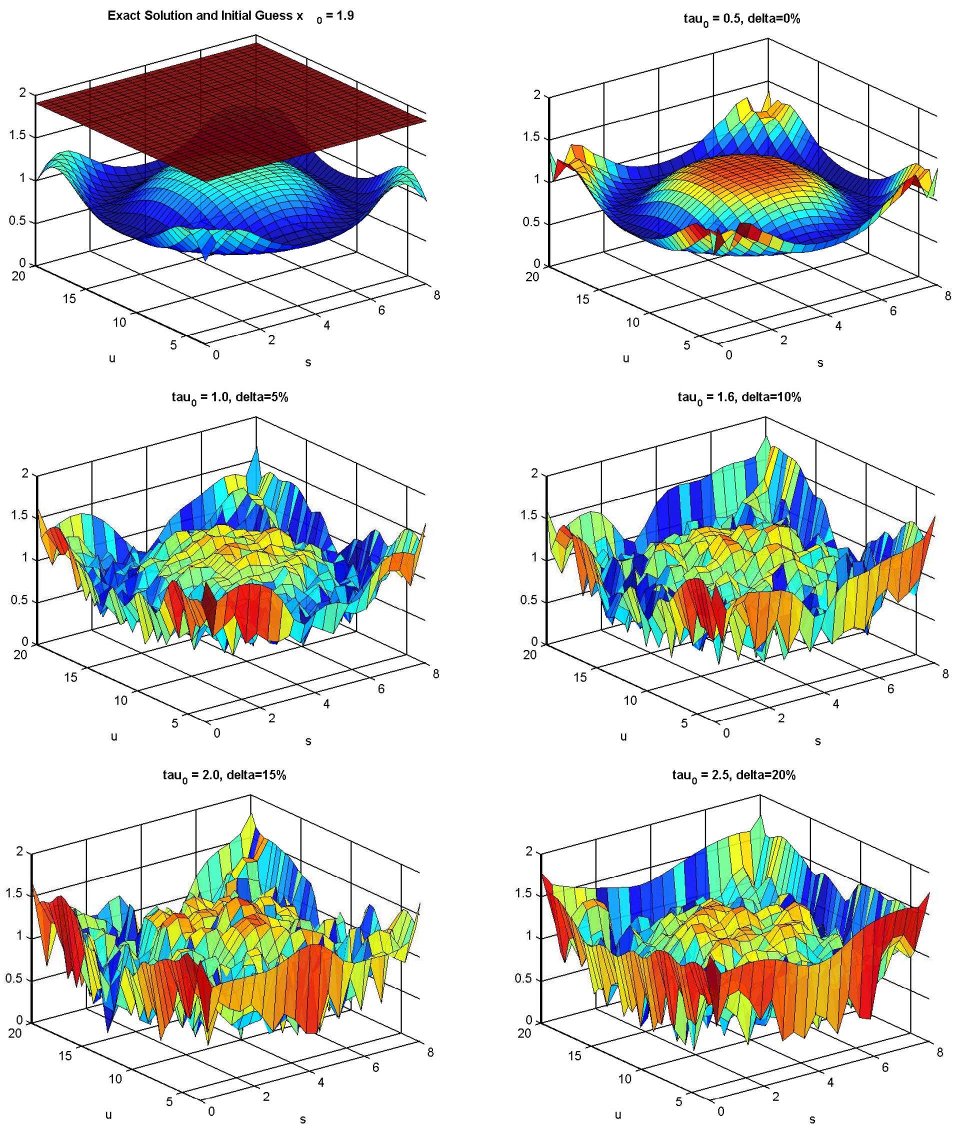
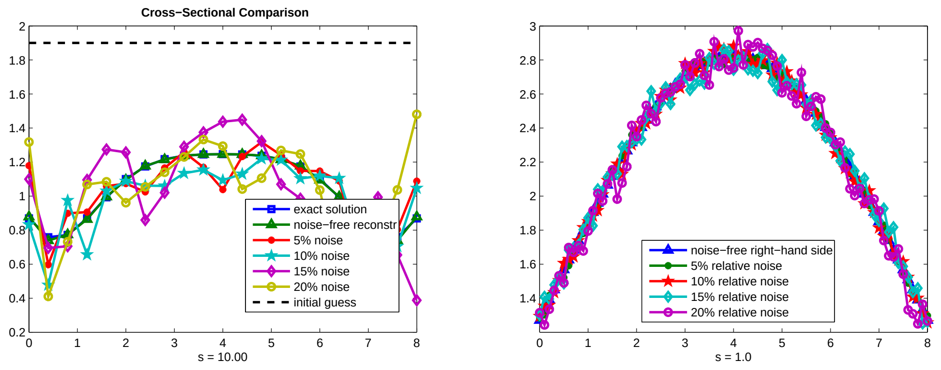


Figure 4.2. Exact $x(s, u)$ and Approximations which Accompany Table 4.1

In Figure 4.1, we see the visual effects of adding relative noise levels of 5%, 10%, 15% and 20% to our simulated data. We are also provided with a visual cross-sectional comparison of the resulting approximations from pairing the levels of noise and initial approximations from Table 4.1.

Figure 4.3. Cross-Sectional Comparison of $x(s, u)$ Approximations and Noisy Data which Accompany Table 4.1



REFERENCES

[1] A. Kirsch, *An Introduction to the Mathematical Theory of Inverse Problems, Applied Mathematical Sciences 120.* Springer-Verlag. New York, 1996.

[2] V. Hutson and J. Pym, *Applications of Functional Analysis and Operator Theory.* Academic Press Inc., Ltd., London, 1980.

[3] A. Bakushinsky and A. Goncharsky, *Ill-posed Problems: Theory and Applications.* Dordrecht Kluwer, 1994.

[4] V. Vasin and A. Ageev, *Ill-posed problems with a priori information.* VNU, Utrecht, 1995.

[5] A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems.* Translated from the Russian. Preface by translation editor Fritz John. Scripts Series in Mathematics. V. H. Winston & Sons, Washington, D.C.: John Wiley & Sons, New York - Toronto, Ont. - London, 1977.

[6] A. Bakushinsky, "On a convergence problem of the iterative-regularized gauss-newton method," *Comput Math. Math phys N9*, vol. 32, pp. 1353–1359, 1992.

[7] K. A. B. and M. Yu, *Iterative Methods for Ill-Posed Operator Equations with Smooth Operators.* Springer Dordrecht, Great Britain, 2004.

[8] B. Kaltenbacher, A. Neubauer, and O. Scherzer, *Iterative regularization methods for nonlinear ill-posed problems.* Radon Series on Computational and Applied Mathematics, Walter de Gruyter, Berlin, 2008.

[9] A. Bakushinsky, "Iterative methods without saturation for solving degenerate nonlinear operator equations," *Dokl. Akad. Nauk N1*, vol. 344, pp. 7–8, 1995.

[10] R. Airapetyan, A. Ramm, and A. Smirnova, "Operator theory and its applications," *Fields Inst. Commun., 25, Amer. Math Soc., Providence, RI.*, vol. 1998, pp. 111–137, 2000.

[11] W. Rudin, *Real and Complex Analysis, Third edition.* McGraw-Hill Book Company, 1987.

[12] K. Deimling, *Nonlinear Functional Analysis.* Springer-Verlag. New York, 1985.

[13] J. Ortega and W. Rheinboldt, *Iterative solutions of nonlinear equations in several variables.* Academic Press, New York - London, 1970.

[14] A. B. Bakushinsky, "Iterative methods for nonlinear operator equations without regularity. new approach," *Dokl. Russian Acad. Sci.*, vol. 330, pp. 282–284, 1993.

[15] E. Akimova and V. Vasin, "Stable parallel algorithms for solving the inverse gravimetry and magnitimetry problems," *CD Proceedings of the 9th International Conference on Numerical Methods in Continuum Mechanics, Zilina, Slovakia, September 9-12*, pp. 9–12, 2003.

[16] A. B. Bakushinsky and A. Smirnova, "On application of generalized discrepancy principle to iterative methods for nonlinear ill-posed problems," *Numerical Functional Analysis and Optimization, N1*, vol. 26, pp. 35–48, 2005.

[17] V. Morozov, "Regularizing families of operators," *Computing Methods and Programming*, vol. VIII, pp. 63–95, 1967.

[18] U. Hamarik, R. Palm, and T. Raus, "A family of rules for the choice of the regularization parameter in the larentiev method in the case of rough estimate of the noise level data," *J. Inverse Ill-Posed Problems*, vol. 20, pp. 831–854, 2012.

[19] Q. Jin and U. Tautenhahn, "On the discrepancy principle for some newton type methods for solving nonlinear inverse problems," *Numer. Math., N4*, vol. 111, pp. 43–62, 2009.

[20] S. Langer and T. Hohage, "Convergence analysis of an inexact iteratively regularized gauss-newton method under general source conditions," *J. Inverse Ill-Posed Probl., N3*, vol. 15, pp. 311–327, 2007.

Appendix A

```
% Numerical Approximation Using Trapezoidal Rule
% Find the integral with respect to s of:
% exp(s)./((x.^2+t.^2).*(x.^2-t.^2)), where x(s)=exp(s)

function output = forward
format long;

% PARAMETERS
M=10; a=3; b=4; step=(b-a)/M; c=a; d=b;

% Compute the quadrature elements
 [s, w, m] = quadrature('trap',a,b,step);
% [s, w, m] = quadrature('simp',a,b,step);
% [s, w, m] = quadrature('midpt',a,b,step);
% [s, w, m] = quadrature('gauss',a,b,step);

t=c:step:d;

% -----------------------------------------------%
% DIRECT PROBLEM
% -----------------------------------------------%

k = length(t);
rmod=model(s,m);     %compute model solution

rhs=F(s,m,rmod,t,w,k);

% -----------------------------------------------%
% ANALYTIC SOLUTION
% -----------------------------------------------%
syms z
fa= inline((1/(4*z^3))*(-2*atan(exp(b)/z)+log(exp(b)-z)-log(exp(b)+z)+2*atan(exp(a)/z)-log(exp(a)-z)+log(exp(a)+z)))
%------------------------------------------------%

% -----------------------------------------------%
% TABLE OF OUTPUT VALUES
% -----------------------------------------------%
disp('_________________________________________________________________________________ ')
disp('         t                  f(t)              fa(t)        |fa(t)-f(t)|/|fa(t)|     ')
disp('                           numerical          analytical          rel. error ')
disp('_________________________________________________________________________________ ')

for t=3:.1:4
    disp([t     F(s,m,rmod,t,w,1)     fa(t)       abs(F(s,m,rmod,t,w,1)-fa(t))/abs(fa(t))])
end

% -----------------------------------------------%
% PLOTTING THE OUTPUT
% -----------------------------------------------%
t=3:.1:4;
n=uint8((t-3)*(1e+001)+1); %return integer value for n to correspond with values of t
plot(t,fa(t),'b-',t,rhs(n),'go')
axis([3 4 .0000390 .0000395])
% title('Integral of K(s,t)= (e^s)/[{e^2}^s+t^2][{e^2}^s-t^2]: Trapezoidal');
% ----------------------------------------------%
% FUNCTION DEFINITIONS
% ----------------------------------------------%
function xmod = model(s,m)
for j=1:m
    xmod(j,1)=exp(s(j));
end
% ----------------------------------------------%
function vect = K(s,x,t)
vect = exp(s)./((x.^2+t.^2).*(x.^2-t.^2));
% ----------------------------------------------%
function matr = Kprime(s,t,x)
matr = (exp(s).*(x.^4-t.^4)-(4*x.^3).*exp(s))./(x.^4-t.^4).^2;
% ----------------------------------------------%
function f = F(s,m,x,t,w,k)
f = zeros(k,1);
for j = 1:m
     f(:,1) = f(:,1) + (K(s(j),x(j),t).*w(j))';
end;
% ----------------------------------------------%
function fp = Fprime(s,t,x,m,w,k)
fp = zeros(k,m);
for j = 1:m
        fp(:,j) = fp(:,j) + (Kprime(s(j),t,x(j)).*w(j))';
end;

function [s,w,m] = quadrature(quadtype,a,b,step)
switch (quadtype)

    case 'trap'
        s = a:step:b;
        m = length(s);
        h = (b-a)/(m-1);
        w = ones(1,m);
        w(1) = 0.5;
        w(m) = 0.5;
        w = w*h;

    case 'simp'
        s = a:step:b;
        m = length(s);

        if (mod(m,2) == 0)
          error('Must have odd number of nodes for Simpson Quadrature');
        end;

        h = (b-a)/(m-1);
        w = 2*ones(1,m);
        for k = 2:2:m-1
            w(k) = 4;
        end;
        w(1) = 1;
        w(m) = 1;
        w = w*h/3;

    case 'midpt'
        s = a+step/2:step:b-step/2;
        m = length(s);
        h = (b-a)/m;
        w = h*ones(1,m);

    case 'gauss'
        s = a:step:b;
        m = length(s);
        u = 1:m-1;
        u = u ./ sqrt(4*u.^2 - 1);
        % Same as A = diag(u,-1) + diag(u,1), but faster (no addition).
        A = zeros(m,m);
        A(2:m+1:m*(m-1)) = u;
        A(m+1:m+1:m^2-1) = u;
        % Find the base points and weight factors for the interval [-1,1].
        [v,s] = eig(A);
        [s,k] = sort(diag(s));
        w = 2 * v(1,k)'.^2;
        % Linearly transform from [-1,1] to [a,b].
        s = (b-a)/2 * s + (a+b)/2;
        w = (b-a)/2 * w;

end;
```

Appendix B

```
% THIS FUNCTION APPLIES IRN METHOD
% TO AN INVERSE PROBLEM
% w/ simulated data
function output = irn
global H
format long;

% PARAMETERS
M=100; a=-1; b=1; step=(b-a)/M; c=a; d=b;
nmax=20;                        % number of iterations
tau0=10^(-2);                    % regularization parameters
%tau0=10^(-1);                    % regularization parameters
%tau0=1;
perc=0.00;                      % percentage of noise
H=2;                          % parameter of the model

% Compute the quadrature elements
 [s, w, m] = quadrature('trap',a,b,step);
% [s, w, m] = quadrature('simp',a,b,step);
% [s, w, m] = quadrature('midpt',a,b,step);
% [s, w, m] = quadrature('gauss',a,b,step);

% -----------------------------------------------%
% INVERSE PROBLEM
% -----------------------------------------------%
% Initial guess
r0 = zeros(m,1);
%r0 =1.1*ones(m,1);
%r0 = ones(m,1);
%r0 = -0.2*ones(m,1);

t=c:step:d;

% -----------------------------------------------%
% DIRECT PROBLEM
% -----------------------------------------------%

k = length(t);
rmod=model(s,m);     %compute model solution
rhs=F(s,m,rmod,t,w,k);

% %%load noise from the file 'ns'
noise_rhs = rhs;
% r0 = rmod;

relerr = norm(r0-rmod)/norm(rmod);
%disp(sprintf('Relative Error=%1.4E', relerr));
output(1,:) = [0 relerr];

rn=r0;
%disp(sprintf('Percentage of noise=%6.2f', perc));

% The beginning of ITERATIVE REGULARIZATION SCHEME
f=F(s,m,rn,t,w,k);
f=f-noise_rhs;
```

```
% The Frechet derivative F'(rn)

% The beginning of FOR - loop
for n=1:nmax
 FP=Fprime(s,t,rn,m,w,k);
 % Parameter for the convergence rate function
 p=1;
 %tau=tau0*(n^(-p));
 tau=tau0*exp(-n*p);
 %tau=tau0*(log(exp(1)+n))^(-p);
 tau_iter(n)=tau;

 condFP(n) = cond(FP);
 condFP_reg(n) = cond(FP+tau*eye(m,m));

 % The Newton step
 sn = -(FP+tau*eye(m,m))\(f+tau*(rn-r0));
       rn = rn + sn;

% The vector f=F(rk)-noise_rhs;
f=F(s,m,rn,t,w,k);
f=f-noise_rhs;

  % COMPUTE THE RELATIVE ERROR (using Frobenius norm)
    relerr = norm(rmod-rn,'fro')/norm(rmod,'fro');
    output(n+1,:) = [n relerr];

    %For output table
    Relerror(n)=relerr;

    % STOP IF CONVERGENCE OR DIVERGENCE DETECTED
    if relerr < 0.01
        disp(sprintf('Convergence Detected!'));
        break;
    else
        if relerr > 2
            disp(sprintf('Divergence Detected!'));
            break;
        end;
    end;

    discrep = norm(f,'fro');
    %For output table
    Discrepancy(n)=discrep;

end; % End of For - loop
discrep = norm(f,'fro');
Discrepancy(n)=discrep;
nlast=n;

str=fprintf('Initial regularization parameter is: tau0=%8.0e\n', tau0);
str=fprintf('Initial approximation is: x0=%8.2e\n', r0(1));

disp('__________________________________________________')
disp('Iter # ... Rel. error   ... Discrepancy  ... Cond(F_prime) ... Cond(F_prime+reg_param) .. iter. tau')
disp('___________________________________________________')
```

```
% Creation of matrix for output of TABLE VALUES
for n=1:nlast
   str=fprintf('%2.1f ...   %8.6e ... %8.6e  ... %8.6e ... %8.6e ...%8.6e\n', n, Relerror(n), Discrepancy(n),
      condFP(n),condFP_reg(n), tau_iter(n));
end

% PLOT THE OUTPUT
%hold on
plot(s,rmod,'k-', s,rn,'ro', s,r0,'b--');
axis([-1 1 -2 4])
xlabel('s');
ylabel('x(s)');
title('EXACT solution and COMPUTED solution');
legend('Exact','Computed','initial approx.');

% -----------------------------------------------%
% FUNCTION DEFINITIONS
% -----------------------------------------------%
function xmod = model(s,m)
for j=1:m
    xmod(j,1)=(1-s(j)*s(j))^2;
end;
% -----------------------------------------------%
function vect = K(s,x,t)
global H
vect = log(((t-s).^2+H^2)./((t-s).^2+(H-x).^2));
% -----------------------------------------------%
function matr = Kprime(s,t,x)
global H
matr = 2.*(H-x)./((H-x).^2+(t-s).^2);
% -----------------------------------------------%
function f = F(s,m,x,t,w,k)
f = zeros(k,1);
for j = 1:m
     f(:,1) = f(:,1) + (K(s(j),x(j),t).*w(j))';
end;
% -----------------------------------------------%
function fp = Fprime(s,t,x,m,w,k)
fp = zeros(k,m);
for j = 1:m
        fp(:,j) = fp(:,j) + (Kprime(s(j),t,x(j)).*w(j))';
end;

function [s,w,m] = quadrature(quadtype,a,b,step)
switch (quadtype)

    case 'trap'
        s = a:step:b;
        m = length(s);
        h = (b-a)/(m-1);
        w = ones(1,m);
        w(1) = 0.5;
        w(m) = 0.5;
        w = w*h;

    case 'simp'
        s = a:step:b;
        m = length(s);
```

```
            if (mod(m,2) == 0)
                error('Must have odd number of nodes for Simpson Quadrature');
            end;
        h = (b-a)/(m-1);
        w = 2*ones(1,m);
            for k = 2:2:m-1
                w(k) = 4;
            end;
        w(1) = 1;
        w(m) = 1;
        w = w*h/3;

    case 'midpt'
        s = a+step/2:step:b-step/2;
        m = length(s);
        h = (b-a)/m;
        w = h*ones(1,m);

    case 'gauss'
        s = a:step:b;
        m = length(s);
        u = 1:m-1;
        u = u ./ sqrt(4*u.^2 - 1);
        % Same as A = diag(u,-1) + diag(u,1), but faster (no addition).
        A = zeros(m,m);
        A(2:m+1:m*(m-1)) = u;
        A(m+1:m+1:m^2-1) = u;
        % Find the base points and weight factors for the interval [-1,1].
        [v,s] = eig(A);
        [s,k] = sort(diag(s));
        w = 2 * v(1,k)'.^2;
        % Linearly transform from [-1,1] to [a,b].
        s = (b-a)/2 * s + (a+b)/2;
        w = (b-a)/2 * w;
end;
```

Appendix C

```
% THIS FUNCTION TESTS THE ITERATIVELY REGULARIZED
% GAUSS-NEWTON METHOD ON THE NONLINEAR 2D GRAVIMETRY PROBLEM.

function output = irgn_2D_gravimetry
global H
format long;
warning off;
% PARAMETERS
a = 3.2; b = 20.0;
c = 0; d = 8.0;
m = 21; n = 41; %grid for inverse problem
md = 81; nd = 161; % grid for direct problem
H = 2.0;
kmax =30;
tauvector = [0.5 1.0 1.6 2.0 2.5]; % vect of init reg parameters
delta_set = [0 0.05 0.1 0.15 0.2]; % vect of perc. of noise to add to data
Relerror=zeros(5,kmax+1); %Initializing for speed (for table)
Discrepancy=zeros(5,kmax+1); %Initializing for speed (for table)
cond_adj=zeros(5,kmax+1); %Init condition vect for speed (for table)
cond_reg_adj=zeros(5,kmax+1);%Init reg cond vect for speed (for table)
minSV=zeros(5); %Init vect for min singular value for each delta
maxSV=zeros(5); %Init vect for max singular value for each delta

% -----------------------------------------------%
% COMPUTE THE QUADRATURE ELEMENTS
% -----------------------------------------------%
% 2D quadrature for direct problem
[sd,td,Sd,Td,wnd,wmd] = quadrature2d('trap',a,b,c,d,nd,md);
% 2D quadrature for inverse problem
[s,t,S,T,wn,wm] = quadrature2d('trap',a,b,c,d,n,m);

% -----------------------------------------------%
% DIRECT PROBLEM
% -----------------------------------------------%
% -----------------------------------------------%
% SET Xd TO SOME KNOWN FUNCTION ON A FINE GRID
% Choice of Xd will become model sol., X, for Inverse Problem
% -----------------------------------------------%
TTd = (Td-c)/(d-c);  % domain normalization
SSd = (Sd-a)/(b-a);  % domain normalization
Xd = cos((4*TTd-2).^2 + (4*SSd-2).^2)/4 + 1;

Xk_delta = ones(m,n,5);
Fk_delta = zeros(md,nd,5); %initialization of noisy RHS

kk = 0;
%tau0 = 1e0;
na=1;
fexact = F(Td,Sd,td,sd,wmd,wnd,Xd);
for delta = delta_set
    tau0 = tauvector(na);
    kk = kk+1;
% CALCULATE f DIRECTLY FROM X ON A FINE GRID
% F returns F(x_mod) in  a vector form
```

```
fd = fexact+ delta*(rand(md*nd,1)-rand(md*nd,1));
Fd = VtoM(fd,md,nd);
Fk_delta(:,:,kk) = Fd;
abs_err_rhs = norm(delta*rand(md*nd,1),'fro')*sqrt((b-a)/(n-1));
rel_err_rhs = norm(delta*rand(md*nd,1),'fro')/norm(fexact,'fro');
Discrepancy(na,1)=rel_err_rhs;
% compute f on a loose grid
Fm = zeros(m,n);
for i = 1:m
    for j = 1:n
        Fm(i,j) = Fd(4*i-3,4*j-3);
    end
end

% ----------------------------------------------%
% INVERSE PROBLEM
% ----------------------------------------------%
f = MtoV(Fm,m,n); %data for inverse problem
% ----------------------------------------------%
% X is the model solution (for computation of rel. error)
% X is the same as the Xd chosen for forward problem (from which
%  we generated the data)
% ----------------------------------------------%
TT = (T-c)/(d-c);  % domain normalization
SS = (S-a)/(b-a);  % domain normalization
X = cos((4*TT-2).^2 + (4*SS-2).^2)/4 + 1;
x = MtoV(X,m,n);

% INITIAL SOLUTION
X0 = 0.1*ones(m,n); % for graphing purpose
Xk = X0;
x0 = MtoV(X0,m,n);  % for iterations
X00 = 0.1*ones(m,n); % for graphing purpose
x00 = MtoV(X00,m,n);  % for iterations
xk=x0;

relerr = norm(x0-x,'fro')/norm(x,'fro');
Relerror(na,1)=relerr;

for k = 1:kmax

    Xklast = xk;
    discreplast = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro')/norm(f, 'fro');

    % ITERATIVE REGULARIZATION SCHEME
        beta = 1;
    tau = tau0*((k+1)^-beta);
    % CALCULATE THE MATRIX G := F(Xk)-f
    G = F(T,S,t,s,wm,wn,Xk) - f;

    % APPLY THE LINEAR OPERATOR F'*(Xk) TO G
    FP = Fprime(t,s,T,S,wm,wn,Xk);

           % COMPUTE CONDITION NUMBERS
           cond_adj(na,k)=cond(FP'*FP);

    % FINISH THE ITERATION
    % IRGN steps:
```

```
    %   reminder: pk = x(k+1) - x(k)
    Pk = -(FP'*FP + tau*eye(m*n,m*n))\(FP'*G + tau*(xk - x00));
    xk = xk  + Pk;
    Xk = VtoM(xk,m,n); %VECTOR to MATRIX transformation
            % COMPUTE REG COND NUMBER
            cond_reg_adj(na,k)=cond(FP'*FP + tau*eye(m*n,m*n));
     % COMPUTE THE RELATIVE ERROR
    relerr = norm(x-xk,'fro')/norm(x,'fro');
    Relerror(na,k+1)=relerr; %Leslie added
    output(k+1,:) = [k relerr];

    % STOP IF CONVERGENCE OR DIVERGENCE DETECTED
    if ((norm(xk-Xklast,'fro') < 1E-5) || (relerr < 1E-10))
        fprintf('Convergence Detected!');
        break;
    else
        if ((norm(xk-Xklast,'fro') > 100) || (relerr > 2))
            fprintf('Divergence Detected!');
            xk = Xklast; %vector
            Xk = VtoM(xk,m,n); %VECTOR to MATRIX transformation
            break;
        end;
    end;


    discrep = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro')/norm(f, 'fro');
    %for output table
    Discrepancy(na,k+1)=discrep;
    if (discreplast<discrep)
        xk = Xklast; %vector
        Xk = VtoM(xk,m,n); %VECTOR to MATRIX transformation
        discrep = discreplast;
        break;
    end
end;
Xk_delta(:,:,kk) = Xk;

discrep = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro')/norm(f, 'fro');

na=na+1; %move to next tau0 for higher level of noise.
end %delta

klast=k;
%Creation of Table for output
disp(' ')
str=fprintf('Initial approximation/guess is a flat surface: x0=%8.2e\n',...
    X0(1,1));
disp(' ')
disp('----------------------------------------------------')
fprintf('   | init. unreg cond# = %8.3e | init. unreg cond# = %8.3e | init. unreg cond# = %8.3e | init. unreg
 cond# = %8.3e | init. unreg cond# = %8.3e \n',cond_adj(1,1), cond_adj(2,1),
 cond_adj(3,1),cond_adj(4,1), cond_adj(5,1));
fprintf('   |   init. reg cond# = %8.3e |   init. reg cond# = %8.3e |   init. reg cond# = %8.3e |   init. reg
 cond# = %8.3e |   init. cond# = %8.3e \n',cond_reg_adj(1,1), cond_reg_adj(2,1),
 cond_reg_adj(3,1),cond_reg_adj(4,1), cond_reg_adj(5,1));
fprintf('   |    (delta=%1.2f & tau0=%1.2f)   |    (delta=%1.2f & tau0=%1.2f)   |    (delta=%1.2f &
 tau0=%1.2f)   |    (delta=%1.2f & tau0=%1.2f)   |    (delta=%1.2f & tau0=%1.2f)\n', delta_set(1),
 tauvector(1), delta_set(2), tauvector(2), delta_set(3), tauvector(3), delta_set(4), tauvector(4),
 delta_set(5), tauvector(5));
disp('----------------------------------------------------')
disp('k  |   Rel. error  & Discrepancy   |   Rel. error  & Discrepancy   |   Rel. error  & Discrepancy   |   Rel.
 error  & Discrepancy   |  Rel. error & Discrepancy')
fprintf('   |    (delta=%1.2f & tau0=%1.2f)   |    (delta=%1.2f & tau0=%1.2f)   |    (delta=%1.2f &
tau0=%1.2f)   |    (delta=%1.2f & tau0=%1.2f)   |    (delta=%1.2f & tau0=%1.2f)\n', delta_set(1),
 tauvector(1), delta_set(2), tauvector(2), delta_set(3), tauvector(3), delta_set(4), tauvector(4),
 delta_set(5), tauvector(5));
disp('------------------------------------------------------')
%Creation of matrix for output of Table Values
for n=1:klast+1
    str=fprintf('%2.0d |  %8.5e  &  %8.5e  |  %8.5e  &  %8.5e  |  %8.5e  &  %8.5e  |  %8.5e  &  %8.5e  |
  %8.5e  &  %8.5e\n', n-1, Relerror(1,n),...
        Discrepancy(1,n), Relerror(2,n), Discrepancy(2,n), Relerror(3,n), Discrepancy(3,n),Relerror(4,n),
 Discrepancy(4,n),Relerror(5,n), Discrepancy(5,n));
end

disp('--------------------------------------------------------')
disp(' ')
disp(' ')

% PLOT THE OUTPUT
wantInterp = 0;
transparency = .9;

fig1 = figure;
subplot(3,2,1);
surf(T,S,X,'FaceAlpha',transparency,'FaceLighting','phong');
hold on
surf(T,S,X0,'FaceAlpha',transparency,'FaceLighting','phong');
if (wantInterp)
    shading interp;
end;
axis([c d a b 0.0 H]);
xlabel('t');
ylabel('s');
title('\bf Exact Solution and Initial Guess');
set(gca,'GridLineStyle','-','linewidth',[1])

subplot(3,2,2);
surf(T,S,Xk_delta(:,:,1),'FaceAlpha',transparency,'FaceLighting','phong');
if (wantInterp)
    shading interp;
end;
axis([c d a b 0.0 H]);
xlabel('t');
ylabel('s');
title('\bf Noise-free reconstruction');
set(gca,'GridLineStyle','-','linewidth',[1])

subplot(3,2,3);
surf(T,S,Xk_delta(:,:,2),'FaceAlpha',transparency,'FaceLighting','phong');
if (wantInterp)
    shading interp;
end;
axis([c d a b 0.0 H]);
xlabel('t');
ylabel('s');
```

```
title('\bf Relative level of noise 5%');
set(gca,'GridLineStyle','-','linewidth',[1])

subplot(3,2,4);
surf(T,S,Xk_delta(:,:,3),'FaceAlpha',transparency,'FaceLighting','phong');
if (wantInterp)
    shading interp;
end;
axis([c d a b 0.0 H]);
xlabel('t');
ylabel('s');
title('\bf Relative level of noise 10%');
set(gca,'GridLineStyle','-','linewidth',[1])

subplot(3,2,5);
surf(T,S,Xk_delta(:,:,4),'FaceAlpha',transparency,'FaceLighting','phong');
if (wantInterp)
    shading interp;
end;
axis([c d a b 0.0 H]);
xlabel('t');
ylabel('s');
title('\bf Relative level of noise 15%');
set(gca,'GridLineStyle','-','linewidth',[1])

subplot(3,2,6);
surf(T,S,Xk_delta(:,:,5),'FaceAlpha',transparency,'FaceLighting','phong');
if (wantInterp)
    shading interp;
end;
axis([c d a b 0.0 H]);
xlabel('t');
ylabel('s');
title('\bf Relative level of noise 20%');
set(gca,'GridLineStyle','-','linewidth',[1])

figure(fig1);


fig2 = figure;
axisName = 'sd';
axisValue = 1.0;
switch (axisName)
    case {'sd', 'x'}
        indx = find(abs(sd-axisValue) == min(abs(sd-axisValue)));

        c=plot(td,Fk_delta(:,indx,1),'^-',td,Fk_delta(:,indx,5),'o-');
        set(c,'linewidth',[2]);
    case {'td', 'y'}
        indx = find(abs(t-axisValue) == min(abs(t-axisValue)));

        c=plot(s,X(indx,:),'s-',s,Xk_delta(indx,:,1),'^-',...
            s,Xk_delta(indx,:,2),'*-',s,Xk_delta(indx,:,3),'p-',...
            t,Xk_delta(indx,:,4),'d-',t,Xk_delta(indx,:,5),'o-',...
            s,X0(indx,:),'k--');
        set(c,'linewidth',[2]);
end;
xlabel('t = 1.0');
```

```
title('\bf Cross Section of Exact and Noisy Data, f(t,v)');

legend( 'noise-free right-hand side', '10% relative noise',0);
figure(fig2);

%Additional Figure for exact solution and noisy data
fig3 = figure;

surf(T,S,X,'FaceAlpha',transparency,'FaceLighting','phong');
axis([0 8 0 20 0.0 2.0]);
xlabel('s');
ylabel('u');
title('\bf Exact/Model Solution x(s,u)');
set(gca,'GridLineStyle','-','linewidth',[1])

figure(fig3);
% end Additional figure

load gong;
sound(y,Fs);
fprintf('Done!  Press Any Key to Continue...\n');
pause;
close all;
format;
%clear;

% -----------------------------------------------%
% FUNCTION DEFINITIONS
% -----------------------------------------------%
function vect = K(T,S,xsi,nu,x)
global H
vect = ((T-xsi).^2 + (S-nu).^2 + x^2).^(-0.5) - ...
     ((T-xsi).^2 + (S-nu).^2 + H^2).^(-0.5);
% -----------------------------------------------%
function vect = Kprime(t,s,XSI,NU,X)
vect = -X.*(((t-XSI).^2 + (s-NU).^2 + X.^2).^(-1.5));
% -----------------------------------------------%
function f = F(T,S,xsi,nu,wm,wn,X)
m = length(xsi);
n = length(nu);
f_matr = zeros(m,n);
for i = 1:m
    sum = zeros(m,n);
    for j = 1:n
        sum = sum + K(T,S,xsi(i),nu(j),X(i,j)).*wn(j);
    end;
    f_matr = f_matr + sum.*wm(i);
end;
f=MtoV(f_matr,m,n);
% -----------------------------------------------%
function fp = Fprime(t,s,XSI,NU,wm,wn,X)
m = length(t);
n = length(s);
fp = zeros(m*n,m*n);
w=wm'*wn;
for i = 1:m
    for j = 1:n
        g=Kprime(t(i),s(j),XSI,NU,X).*w;
        fp((i-1)*n+j,:)=(MtoV(g,m,n))';
    end;
end;
% ----------------------------------------------%
function vect = MtoV(A,m,n)
vect = zeros(m*n,1);
for i = 1:m
    for j = 1:n
        vect((i-1)*n+j,1)=A(i,j);
    end
end

function matr = VtoM(x,m,n)
matr = zeros(m,n);
for i = 1:m
    for j = 1:n
        matr(i,j)=x((i-1)*n+j,1);
    end
end
%-------------------------------------------------%
function map = makecolormap(c1, c2, n)
for i = 0:n-1
    for j = 1:3
        map(i+1,j) = c1(j) + i*(c2(j)-c1(j))/(n-1);
    end;
end;
% ----------------------------------------------%
function [x,w] = quadrature1d(quadtype,a,b,n)
switch (quadtype)
    case 'trap'
        h = (b-a)/(n-1);
        x = linspace(a,b,n);
        w = ones(1,n);
        w(1) = 0.5;
        w(n) = 0.5;
        w = w*h;
    case 'simp'
        if (mod(n,2) == 0)
            error('Must have odd number of nodes for Simpson Quadrature');
        end;
        h = (b-a)/(n-1);
        x = linspace(a,b,n);
        w = 2*ones(1,n);
        for i = 2:2:n-1
            w(i) = 4;
        end;
        w(1) = 1;
        w(n) = 1;
        w = w*h/3;
    case 'midpt'
        h = (b-a)/n;
        x = linspace(a+h/2,b-h/2,n);
        w = h*ones(1,n);
    case 'gauss'
        u = 1:n-1;
        u = u ./ sqrt(4*u.^2 - 1);
        % Same as A = diag(u,-1) + diag(u,1), but faster (no addition).
        A = zeros(n,n);
        A(2:n+1:n*(n-1)) = u;
        A(n+1:n+1:n^2-1) = u;
        % Find the base points and weight factors for the interval [-1,1].
        [v,x] = eig(A);
        [x,k] = sort(diag(x));
        w = 2 * v(1,k)'.^2;
        % Linearly transform from [-1,1] to [a,b].
        x = (b-a)/2 * x + (a+b)/2;
        w = (b-a)/2 * w;
end;
% -----------------------------------------------%
function [x,y,X,Y,wx,wy] = quadrature2d(quadtype,a,b,c,d,m,n)
[x,wx] = quadrature1d(quadtype,a,b,m);
[y,wy] = quadrature1d(quadtype,c,d,n);
[X,Y] = meshgrid(x,y);
```

Appendix D

To demonstrate monotonicity, based on the variational inequality, (1.2), simply note:

$$\langle F(u) - F(v), u - v \rangle = \langle F'(\eta)(u - v), u - v \rangle \geq 0.$$

Definition 1. *A vector space H is called an **inner product space** if, for all $x, y \in H$, there is an associated number $\langle x, y \rangle$ such that the following rules hold:*

- $\langle x, y \rangle = \langle y, x \rangle$
- $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
- $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle\,,\ \ \forall x, y \in H, \alpha \in \mathbb{R}$
- $\langle x, x \rangle \geq 0,\ \ \forall x \in H$
- $\langle x, x \rangle = 0 \Leftrightarrow x = 0$

Note: If we define distance in our inner product space H as $\|x - y\|$, such that:
$\|x - y\| \leq \|x - z\| + \|z - y\|$, then
H becomes a metric space.

Definition 2. *[11] A metric space H is a* **complete metric space** *if any Cauchy sequence $\{x_n\}$ converges to some $x \in H$.*

Definition 3. *[11] If H is an inner product space defined on the reals and as a metric space it is complete, then H is called a* **real Hilbert space***.*

Definition 4. *[2] Suppose that $\mathcal{B}$ and $\mathcal{C}$ are Banach spaces. Let $\mathcal{D}$ be an open subset of $\mathcal{B}$, and let A be an operator mapping $\mathcal{D}$ into C. A is said to be **Fréchet differentiable** at the point $g \epsilon \mathcal{D}$ if there exists an operator $L \epsilon \mathcal{L}\,(\mathcal{B}$,$\mathcal{C})$ such that :*

$$\lim_{\|h\|\to 0} \|A\,(g+h) - Ag - Lh\| \,/\, \|h\| = 0,$$

the limit being required to exist as $h \to 0$ in any manner. The operator L, often denoted by $A'(g)$, is called the ***Fréchet derivative*** *of A at g.*