8-11-2015

On Regularized Newton-type Algorithms and A Posteriori Error Estimates for Solving Ill-posed Inverse Problems

Hui Liu

ON REGULARIZED NEWTON-TYPE ALGORITHMS AND A POSTERIORI ERROR ESTIMATES FOR SOLVING ILL-POSED INVERSE PROBLEMS

by

HUI LIU

Under the Direction of Professor Alexandra Smirnova

ABSTRACT

Ill-posed inverse problems have wide applications in many fields such as oceanography, signal processing, machine learning, biomedical imaging, remote sensing, geophysics, and others. In this dissertation, we address the problem of solving unstable operator equations with iteratively regularized Newton-type algorithms. Important practical questions such as selection of regularization parameters, construction of generating (filtering) functions based on a priori information available for different models, algorithms for stopping rules and error estimates are investigated with equal attention given to theoretical study and numerical experiments.

INDEX WORDS: irregular operator equation, nonlinear ill-posed problem, iterative regularization, stopping rule, image restoration, inverse scattering problem, a posteriori error estimate, inverse magnetometry problem.

ON REGULARIZED NEWTON-TYPE ALGORITHMS AND A POSTERIORI ERROR ESTIMATES FOR SOLVING ILL-POSED INVERSE PROBLEMS

by

HUI LIU

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

2015

ON REGULARIZED NEWTON-TYPE ALGORITHMS AND A POSTERIORI ERROR ESTIMATES FOR SOLVING ILL-POSED INVERSE PROBLEMS

by

HUI LIU

To my Mother and Father

谨献给：刘如明，陈少梦

ACKNOWLEDGEMENTS

I have the deepest gratitude to my advisor, **Professor Alexandra Smirnova**. I still vividly remember our first seminar about a magnetometric problem in her research group. I was very impressed by the passion for her work and her vision. It has been a great pleasure to work with her, and I have definitely benefited from her expertise and patience over the past years. I thank her for giving me the problems investigated in this dissertation, for her continuous and insightful guidance during my studies, and for her understanding and encouragements. Her demands for high quality and attention to details have set an exceptional standard. I would not have written this dissertation without discussion with her. It is my great fortune to be one of her graduate students and to work under her supervision.

I am particularly grateful to the members of my committee, Professor Vladimir Bondarenko, Professor Zhongshan Li, and Professor Michael Stewart for their help on my dissertation and support of my defense.

Professor Vladimir Bondarenko: Professor Bondarenko is a very nice professor who is willing to help all the time. I very much enjoyed his Applied Mathematics and Biostatistics classes. I was fortunate to work with him on a bioinformatic problem. Although it was a short experience, I leant a lot from him.

Professor Zhongshan Li: I am especially thankful to Professor Li. He has a great heart. He treats his students as part of his family. I still remember his help on purchasing my first cell phone in the US and the little translation book he has lent to me. He hosts Thanksgiving parties, New Year parties, Spring Festival parties so his foreign graduate students would not feel home sick and he always makes sure all his students have enough financial support. Without his help, my life at Georgia State would not be this easy. Of course, he is also a great professor. I am still using the knowledge learnt from his Advanced Matrix Analysis class.

Professor Michael Stewart: Professor Stewart inspires my interest in numerical analysis, and I am a super fan of his numerical analysis series. Over the past years I was in his Numerical Analysis I class, Numerical Analysis II class, Numerical Linear Algebra class, and Advanced Numerical Analysis class. I am very impressed by his logical mind. He is well organized and never lose the big picture. Moreover, I appreciate his sense of humor and very much enjoy talking to him.

Among many other people that helped me at Georgia State, I am particularly thankful to **Professor Guantao Chen**, **Professor Frank Hall**, and **Professor Gengsheng Qin** for their continuous support and concern from time to time. I am grateful to **Dr. Changyong Zhong** who is my teaching coordinator and the instructor of my Real Analysis classes, for his great help on my teaching questions and studies.

Besides the above people who have made our department like a big family, I am also thankful to **Ms. Sandra Ahuama-Jonas**, **Ms. Earnestine Collier**, and **Ms. Yvonne Pierce** for their prompt help and continuous support on multifarious paperwork.

I have many people to thank for making my fantastic experience at Georgia State, and I always lack words with which to express my thanks. I thank you all!

The research led to this dissertation was partially supported by NSF grant DMS-1112897.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- SVD - Singular Value Decomposition
- TSVD - Truncated SVD
- MTSVD - Modified Truncated SVD
- ITN - Iteratively Truncated Newton
- IRGN - Iteratively Regularized Gauss-Newton
- GCV - Generalized Cross Validation

Chapter 1

ILL-POSED INVERSE PROBLEMS

According to [1], an inverse problem is the process of calculating the causal factors from a set of observations. For example, image restoration in computer tomography, source estimation in acoustics, or computation of the Earth density from measurements of its gravitational field. Inverse problems have wide applications in optics, radar design, acoustics, communication theory, signal processing, medical imaging, oceanography, computer vision, geophysics, astronomy, remote sensing, natural language processing, machine learning, nondestructive testing, and many other fields. They can provide information about important system parameters that we cannot directly observe. As opposed to the corresponding forward problems, inverse problems deduce causes from results, and they are often unstable and/or not uniquely solvable. Such inverse problems are called *ill-posed or improperly-posed.*

1.1 Regularization of Ill-posed Inverse Problems

If an inverse problem is ill-posed, then even small noise in the data may cause a substantial error in the computed solution. Therefore, techniques known as regularization need to be incorporated in computational algorithms for ill-posed problems. In many cases, the reason for instability is the lack of information in the original model. This can be illustrated by the following simple example.

Consider a linear system of two equations with two unknowns, $Ax = b$, where A is a 2×2 matrix, and b is the data, which is measured with some level of noise. That is, a different system, $Ax = b_\delta$, is solved instead.

In Figure 1.1, the red dot represents the exact solution of the original linear system, and the navy dot represents the computed solution of the noisy system. As one can see, the graph on the left in Figure 1.1 corresponds to a well-posed problem, and the graph on the right corresponds to an ill-posed problem, where small noise in the data results in

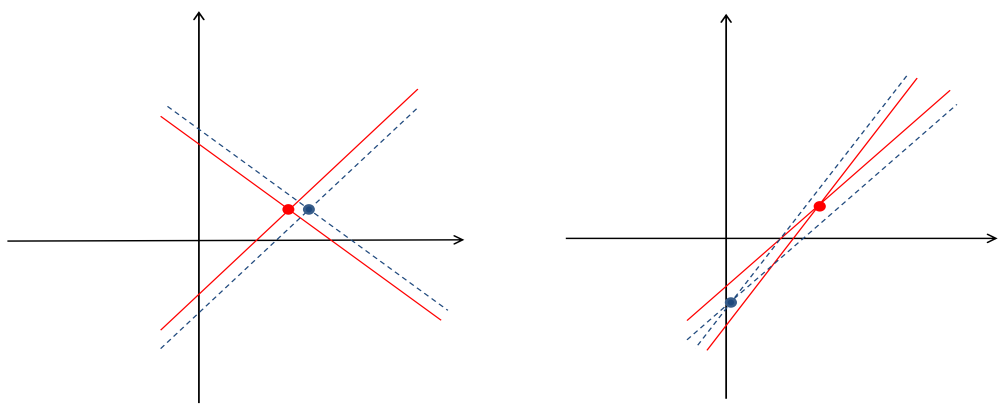

Figure 1.1: Well-posed and ill-posed problems

a large error in the computed solution. Intuitively, one can understand why the second problem is unstable: the two equations in this system are not “entirely different”, and the information they provide is almost the same. On the contrary, the two equations in the first system are “very different”. Therefore, small noise in the data does not cause any substantial damage to the computed solution.

This trivial observation brings us to the main aspect of the regularization theory. Its goal is to incorporate some extra (*a priori*) information into the model in order to make it more stable. Since *a priori* information may not be completely reliable, it is usually weighted by a relatively small regularization parameter. By choosing a “near optimal” value of this parameter, one is trying to strike the best possible balance between accuracy and stability.

1.2 Nonlinear Ill-posed Models

Nonlinearity of some ill-posed inverse problems adds an extra layer of difficulty to the construction of a regularization procedure, since for a nonlinear model regularization often needs to be incorporated into some iterative numerical solver. One of the most used computational methods, which combines both regularization and iterative approximation of the solution is the iteratively regularized Gauss-Newton scheme.

In Chapter 2 of this dissertation, we look at different ways of regularizing Gauss-Newton steps based on *a priori* information available for particular models. We also study an iterative approach to the selection of a regularization parameter and propose a new *a posteriori* stopping rule to terminate Gauss-Newton iterations "just in time" before the noise propagation can potentially destroy an approximate solution. Numerical experiments for both linear and nonlinear models are conducted to illustrate this technique.

In Chapter 3, we continue our analysis of iteratively regularized algorithms for solving (non)linear irregular operator equations. We focus on iteratively truncated Newton's scheme, and illustrate practical aspects of this algorithm with a 2D nonlinear inverse problem in magnetometry. Specifically, we observe the behavior of truncated singular values as iterations progress.

In Chapter 4, the possibility of *a posteriori* error estimates for linear and nonlinear inverse problems is investigated. This is a critical aspect in the analysis of regularized computational methods, since discrepancy alone cannot guarantee the accuracy of an approximate solution when the model is ill-posed. *A posteriori* estimates for the solution of a 2D nonlinear magnetometry equation are studied numerically in order to illustrate the theoretical findings.

Chapter 2

ITERATIVE METHODS FOR ILL-POSED OPERATOR EQUATIONS

In this chapter, we address the problem of solving a nonlinear unstable operator equation on a pair of Hilbert spaces using iteratively regularized Gauss-Newton (IRGN) scheme. We begin by considering three basic groups of generating functions and by looking into the possibility of nonstandard approximation of the pseudoinverse through a "gentle" iterative truncation. Its optimality on the class of generating functions with the same correctness coefficient is proven. In the second part of Chapter 2, we introduce and justify a novel *a posteriori* stopping rule, designed to accommodate noise in both the data and the source condition. In conclusion, we illustrate practical aspects of the regularized algorithm with numerical simulations for a large-scale linear image de-blurring system as well as a nonlinear inverse scattering model. Presentation in this chapter follows [2].

Consider the following inverse problem

$$F(x) = y, \quad F : \mathcal{X} \to \mathcal{Y}, \tag{2.1}$$

where a nonlinear operator F is mapping between two Hilbert spaces $\mathcal{X}$, $\mathcal{Y}$, and the exact data y is contaminated by noise

$$||y - y^{(\delta)}|| \leq \delta. \tag{2.2}$$

Assume, for now, that F is Fréchet differentiable with a Lipschitz continuous derivative, i.e., there exist $N, L \geq 0$ for any $\tilde{x}^{(1)}, \tilde{x}^{(2)} \in \mathcal{X}$ such that

$$||F'(\tilde{x}^{(1)})|| \leq N, \quad ||F'(\tilde{x}^{(1)}) - F'(\tilde{x}^{(2)})|| \leq L||\tilde{x}^{(1)} - \tilde{x}^{(2)}||. \tag{2.3}$$

One of the best known numerical algorithms for solving minimization problem (2.1) is the Gauss-Newton scheme [3]:

$$x^{(n+1)} = \xi - [F'^{*}(x^{(n)})F'(x^{(n)})]^{-1}F'^{*}(x^{(n)})\{F(x^{(n)}) - y^{(\delta)} - F'(x^{(n)})(x^{(n)} - \xi)\},$$

$$x^{(0)}, \xi \in \mathcal{X}. \tag{2.4}$$

It can be viewed as a simplified version of the full Newton method applied to the normal equation, which avoids evaluation of the second derivative operator to get a linear equation with a self-adjoint operator at every step of the iterative process.

Suppose that $\hat{x}$ is a solution to $F(x) = y$, maybe nonunique. In the case when $F'^{*}(\hat{x})F'(\hat{x})$ is not boundedly invertible, A. Bakushinsky [4] suggested to regularize (2.4) iteratively

$$x^{(n+1)} = \xi - \theta[F'^{*}(x^{(n)})F'(x^{(n)}), \alpha^{(n)}]F'^{*}(x^{(n)})\{F(x^{(n)}) - y^{(\delta)} - F'(x^{(n)})(x^{(n)} - \xi)\},$$

$$x^{(0)}, \xi \in \mathcal{X}, \quad \alpha^{(n)} > 0, \quad \lim_{n\to\infty} \alpha^{(n)} = 0, \tag{2.5}$$

with $\theta = \theta(\sigma, \alpha)$ being a filtering function of a spectral variable $\sigma \in [0, N^2]$ and a regularization parameter $\alpha > 0$. In [4] and later in [5], the following conditions on $\theta = \theta(\sigma, \alpha)$ have been used for the convergence analysis of (2.5):

$$\sup_{\sigma\in[0,N^2]} |\theta(\sigma, \alpha)\sqrt{\sigma}| \le C_1\alpha^{-1/2}, \tag{2.6}$$

$$\sup_{\sigma\in[0,N^2]} |\theta(\sigma, \alpha)\sigma - 1|\sigma^p \le C_2(p)\alpha^p, \quad p \ge \frac{1}{2}, \tag{2.7}$$

$$\sup_{\sigma\in[0,N^2]} |\theta(\sigma, \alpha)\sigma - 1| \le C_3, \tag{2.8}$$

where $C_1, C_2(p)$, and C_3 are nonnegative constants.

2.1 Mathematical Preliminaries and Optimality of the Modified Truncation

Considering the most used types of generating functions, one can divide IRGN algorithms (2.5) in three basic groups.

1. The first group includes algorithms formed through Tikhonov's regularization [1].

For example, $\mathcal{M}$-times iterated Tikhonov method [6], where

$$\theta(\sigma,\alpha) = \sum_{k=0}^{\mathcal{M}-1} (\sigma+\alpha)^{-(k+1)}\alpha^k, \quad \mathcal{M} \in \mathbb{N}, \tag{2.9}$$

and the iteratively regularized scheme is a combination of inner and outer iterations for the corresponding discrete problem. For every n, $x^{(n)}$ is changed over the course of $\mathcal{M}$ inner steps.

The qualification order of this algorithm is $\mathcal{M}$, which means that the best convergence rate that can be achieved here is $O([\alpha^{(n)}]^{\mathcal{M}})$ provided we have the same or higher order of the Hölder-type source condition:

$$\hat{x} - \xi = (F'^*(\hat{x})F'(\hat{x}))^{\nu}\omega, \quad \omega \in \mathcal{X}, \quad \nu \geq \mathcal{M}. \tag{2.10}$$

At $p = \mathcal{M}$ (see (2.7)), the so-called saturation occurs, and after that the convergence rate is no longer improving.

2. The second group consists of procedures where the operator $F'^*(\cdot)F'(\cdot)$ is viewed as an infinite series and the regularization is carried out by truncating its tail. The generating function below corresponds to the Newton-Landweber method [4]

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

In practice, it is executed as a sequence of outer Newton steps with a growing number of inner Landweber iterations to solve each subproblem. The qualification order of this method is infinity, i.e., it is saturation free. Apart from Landweber's algorithm, the conjugate gradient scheme has also been used for inner iterations [7]. For methods in this group, the regularization parameter can be viewed as the reciprocal of the number of inner iterations.

3. Last but not least are the methods constructed through iterative truncation. The

best known approach here is to fully truncate the small elements of the spectrum:

$$\theta(\sigma,\alpha) := \begin{cases} \frac{1}{\sigma}, & \sigma \geq \alpha, \\ 0, & 0 \leq \sigma < \alpha. \end{cases} \tag{2.11}$$

The function $\theta = \theta(\sigma,\alpha)$ in (2.11) regularizes $[F'^*(\cdot)F'(\cdot)]^{-1}$, the inverse to $F'^*(\cdot)F'(\cdot)$. The less aggressive approach is to truncate continuously

$$\theta(\sigma,\alpha)\sqrt{\sigma} := \begin{cases} \frac{1}{\sqrt{\sigma}}, & \sigma \geq \alpha, \\ \frac{1}{\sqrt{\alpha}}, & 0 \leq \sigma < \alpha. \end{cases} \tag{2.12}$$

In case of a linear operator, it has been studied in [8]. As one can see, the function $\theta = \theta(\sigma,\alpha)$ in (2.12) regularizes $[F'^*(\cdot)F'(\cdot)]^{-1}F'^*(\cdot)$, the pseudo-inverse to $F'(\cdot)$.

The modified truncation has a very nice optimal property, which we illustrate here for the linear case. Consider a linear operator equation $Ax = y$, and define an approximate solution as follows:

$$x_{\alpha,\delta} = \theta(A^*A,\alpha)A^*y^{(\delta)} := R_\alpha y^{(\delta)}.$$

Here $\theta = \theta(\sigma,\alpha)$ with $\sigma \in [0, ||A||^2]$ and $\alpha > 0$. The well-known estimate for the relative error shows the trade-off between accuracy and stability for the regularizing strategy

$$\frac{||\hat{x} - x_{\alpha,\delta}||}{||\hat{x}||} \leq \frac{||\hat{x} - x_\alpha||}{||\hat{x}||} + \underbrace{||A||\,||\mathcal{R}_\alpha||}_{\text{cond}_\alpha(A)} \frac{||y - y^{(\delta)}||}{||y||}, \quad x_\alpha = R_\alpha y. \tag{2.13}$$

The first term $\frac{||\hat{x}-x_\alpha||}{||\hat{x}||}$ in (2.13) gives the accuracy of the regularization algorithm, and $||A||\,||\mathcal{R}_\alpha||$ can be viewed as the regularized condition number. Basically, inequality (2.13) generalizes the well-known formula

$$\frac{||\hat{x} - x_\delta||}{||\hat{x}||} \leq \underbrace{||A||\,||A^{-1}||}_{\text{cond}(A)} \frac{||y - y^{(\delta)}||}{||y||}, \quad x_\delta = A^{-1}y^{(\delta)}. \tag{2.14}$$

Estimate (2.14) can be obtained from (2.13) if one passes to the limit as $\alpha \to 0$. We consider the following problem: among all regularizing strategies with the same regularized condition number, find the one whose accuracy is the best. In other words, among regularizing strategies, where α is selected to provide the same regularized condition number, find the strategy that minimizes $||\hat{x} - x_\alpha||/||\hat{x}||$. One can easily verify that an answer to the above question is continuous truncation defined in (2.12) with $\alpha = \frac{1}{\mathcal{K}^2}$ and $\mathcal{K} := \text{cond}_\alpha(A)/||A||$:

$$\theta\left(\sigma, \frac{1}{\mathcal{K}^2}\right)\sqrt{\sigma} := \begin{cases} \frac{1}{\sqrt{\sigma}}, & \sigma \geq \frac{1}{\mathcal{K}^2}, \\ \mathcal{K}, & 0 \leq \sigma < \frac{1}{\mathcal{K}^2}. \end{cases} \tag{2.15}$$

Indeed, for any $\mathcal{R}_\alpha(\theta)$, one has

$$\frac{||\hat{x} - x_\alpha||}{||\hat{x}||} = \frac{||\hat{x} - \theta(A^*A, \alpha)A^*A\hat{x}||}{||\hat{x}||} \leq \sup_{\sigma \in S(A^*A)} |1 - \theta(\sigma, \alpha)\sigma|,$$

where $S(B)$ denotes the spectrum of an operator B. Suppose there is some other strategy $\bar{\theta} = \bar{\theta}(\sigma, \bar{\alpha})$ with some choice of the parameter $\bar{\alpha}$, whose accuracy is better:

$$\sup_{\sigma \in S(A^*A)} |1 - \bar{\theta}(\sigma, \bar{\alpha})\sigma| < \sup_{\sigma \in S(A^*A)} \left|1 - \theta\left(\sigma, \frac{1}{\mathcal{K}^2}\right)\sigma\right|.$$

Then at least for some $\bar{\sigma} \in S(A^*A)$,

$$|1 - \bar{\theta}(\bar{\sigma}, \bar{\alpha})\bar{\sigma}| < \left|1 - \theta\left(\bar{\sigma}, \frac{1}{\mathcal{K}^2}\right)\bar{\sigma}\right| = \begin{cases} 0, & \bar{\sigma} \geq \frac{1}{\mathcal{K}^2}, \\ 1 - \mathcal{K}\sqrt{\bar{\sigma}}, & 0 \leq \bar{\sigma} < \frac{1}{\mathcal{K}^2}. \end{cases}$$

Thus $\mathcal{K} < \bar{\theta}(\bar{\sigma}, \bar{\alpha})\sqrt{\bar{\sigma}}$, which means $\mathcal{K} < ||\mathcal{R}_{\bar{\alpha}}(\bar{\theta})||$, and we arrive at a contradiction with the fact that the two strategies have the same condition number.

2.2 Prior Convergence Results for the Iteratively Regularization Gauss-Newton Algorithm

Since after A. Bakushinsky proposed the original IRGN scheme in 1991 and then the generalized scheme in 1995, the algorithm has been studied by many authors ([4], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]) both theoretically and numerically. As opposed to the linear case, where the convergence analysis can be carried out without any source conditions and the source conditions are used to derive the convergence rates, in the nonlinear case the source conditions are generally needed for both, convergence analysis and justification of the convergence rates, and the higher the nonlinearity the stronger the source condition has to be. In general, the *source conditions* take the form

$$\hat{x} - \xi = \varphi(F'^{*}(\hat{x})F'(\hat{x}))\omega, \quad \omega \in \mathcal{X}, \quad ||\omega|| \leq \rho. \tag{2.16}$$

They can be viewed as structural assumptions on the solution that reflect the level of ill-posedness of a particular problem. These conditions are, in fact, necessary for justifying convergence rates as was shown in [23]. The function $\varphi : [0, ||F'(\hat{x})||^2] \to [0, \infty)$ in (2.16) is assumed to be continuous and increasing with $\varphi(0) = 0$. Initially, method (2.5) was investigated under Hölder source condition for $\nu \geq 0$ ([4], [9]):

$$\varphi(t) := t^{\nu}. \tag{2.17}$$

For $\nu \geq \frac{1}{2}$, the convergence analysis has been carried out first, for an *a priori* stopping rule [4], and later for an *a posteriori* one [24]. When F is moderately nonlinear ([9], [10], [12]), such that for some linear operators $\mathcal{R}$ and $\mathcal{Q}$

$$F'(\tilde{x}^{(1)}) = \mathcal{R}(\tilde{x}^{(1)}, \tilde{x}^{(2)})F'(\tilde{x}^{(2)}) + \mathcal{Q}(\tilde{x}^{(1)}, \tilde{x}^{(2)}), \quad ||I - \mathcal{R}(\tilde{x}^{(1)}, \tilde{x}^{(2)})|| \leq C_{\mathcal{R}},$$

$$||\mathcal{Q}(\tilde{x}^{(1)}, \tilde{x}^{(2)})|| \leq C_{\mathcal{Q}}||F'(\hat{x})(\tilde{x}^{(1)} - \tilde{x}^{(2)})||, \quad \tilde{x}^{(1)}, \tilde{x}^{(2)} \in \mathcal{B}_{\sigma}(\hat{x}), \tag{2.18}$$

with σ, $C_{\mathcal{R}}$, $C_{\mathcal{Q}}$ being sufficiently small, ν can be reduced to $0 \leq \nu < \frac{1}{2}$ [9] and, at expense of lower convergence rates, the source condition can be changed to a logarithm [10]

$$\varphi(t) := \begin{cases} (-\ln t)^{-\nu}, & \text{if} \quad 0 < t < \mathrm{e}^{-1}, \quad \nu > 0 \\ 0, & \text{if} \quad t = 0, \end{cases} \tag{2.19}$$

or to a general monotonically increasing function φ, for which the following function

$$\Phi(t) := t(\varphi \cdot \varphi)^{-1}(t) \tag{2.20}$$

is convex and twice differentiable. In (2.20), the expression $(\varphi\cdot\varphi)^{-1}$ stands for the inverse of $\varphi\cdot\varphi$ and $(\varphi\cdot\varphi)(t) := \varphi^2(t)$ (see [12], p.320). As an alternative to the source conditions based on the classical spectral theory, approximate source conditions have been investigated recently by Hofmann and co-authors both in Banach and Hilbert spaces ([18], [25], [26], [27]). Some of the most recent developments in the theory of IRGN have been motivated by a number of specific large-scale applications, which required a more sophisticated implementation of the original IRGN procedure and a more careful convergence analysis. For example, in [19], it has been shown how to construct and update a spectral preconditioner for exponentially ill-posed parameter identification problems.

2.3 The Undetermined Reverse Connection and Basic Estimates

The accurate convergence analysis of IRGN scheme (2.5) ([23], [28]) indicates that its convergence rates are, in general, in agreement with the type of the source condition. Numerical experiments, for the most part, confirm the theoretical findings (see ([10], [22]) for details), which means that the special structure, imposed by a source-type condition, is essential for the behavior of the sequence $\{x^{(n)}\}$. At the same time, practical implementation of the iterative process (2.5) with various choices of ξ suggests that the requirement for $||\omega||$ in (2.16) to be small is not that critical and can possibly be relaxed. This observation motivated further research on Gauss-Newton-type iterations. In [29], [30], and [31], a regularized version of (2.4) has been investigated in the following form:

$$x^{(n+1)} = \xi^{(n)} - \theta(F'^*(x^{(n)})F'(x^{(n)}), \alpha^{(n)})F'^*(x^{(n)})\{F(x^{(n)}) - y^{(\delta)} - F'(x^{(n)})(x^{(n)} - \xi^{(n)})\},$$

$$x^{(0)}, \xi^{(n)} \in \mathcal{X}. \tag{2.21}$$

The noise-free case has been analyzed in [29], an *a priori* and *a posteriori* stopping rules for algorithm (2.21) have been justified in [30] and [31], respectively. In [29], [30], and [31], the modified source condition

$$\hat{x} - \xi^{(n)} = (F'^*(x^{(n)})F'(x^{(n)}))^p\omega^{(n)}, \quad p \geq \frac{1}{2}, \quad \omega^{(n)} \in \mathcal{X}, \tag{2.22}$$

which depends on the current iteration point $x^{(n)}$, plays the part of assumption (2.16) and (2.17). We call condition (2.22) *the undetermined reverse connection.* It has been shown in [29], [30], and [31] that even though (2.22) still contains the unknown solution $\hat{x}$, the norm of $\omega^{(n)}$ in (2.22) is greater than the norm of ω in (2.16) and (2.17). Moreover, the norm of $\omega^{(n)}$ can even tend to infinity as $n \to \infty$. Specifically, at every step of iterative process (2.21), the element $\xi^{(n)}$ may be such that

$$||\omega^{(n)}|| \le \frac{\varepsilon}{[\alpha^{(n)}]^k}, \quad \frac{1}{2} \le p - k, \quad \varepsilon \ge 0. \tag{2.23}$$

The main disadvantage of undetermined reverse connection (2.22) is the need to find $\xi^{(n)}$ satisfying (2.22) in each step of the iteration. How can such a $\xi^{(n)}$ be selected in practice? Well, the problem is similar to the one with single ξ in (2.16): no general recipe is known and we just hope to get lucky after trying different ξ's. In case of (2.22), one can argue that the set of potential candidates for the test function is larger due to (2.23). Still, with n source conditions in place of one, it is unlikely that (2.22) will hold at every step "by chance". Therefore, we look into the convergence analysis of algorithm (2.21) under a more realistic "noisy" source condition:

$$\hat{x} - \xi^{(n)} = (F'^*(x^{(n)})F'(x^{(n)}))^p \omega^{(n)} + \zeta^{(n)}, \quad p \ge \frac{1}{2}, \quad ||\omega^{(n)}|| \le \frac{\varepsilon}{[\alpha^{(n)}]^k}, \quad \frac{1}{2} \le p - k,$$

$$\varepsilon \ge 0, \quad ||\zeta^{(n)}|| \le \Delta. \tag{2.24}$$

Let F be Fréchet differentiable with a Lipschitz continuous derivative satisfying (2.3). Like in [4] and [5], suppose that estimates (2.6)-(2.8) hold for the generating function $\theta = \theta(\sigma, \alpha)$. Choose the regularizing sequence $\{\alpha^{(n)}\}$ in (2.21) so that it approaches zero monotonically and

$$1 \le \frac{[\alpha^{(n)}]^{p-k}}{[\alpha^{(n+1)}]^{p-k}} \le R, \quad n = 0, 1, 2, ... \tag{2.25}$$

Let $x^{(0)} \in \mathcal{X}$ satisfy the condition

$$||x^{(0)} - \hat{x}|| \le \eta := l[\alpha^{(0)}]^{p-k} \le 1 \tag{2.26}$$

with l to be specified below. By (2.21), it follows that

$$x^{(n+1)} - \hat{x} = -\theta(F'^*(x^{(n)})F'(x^{(n)}), \alpha^{(n)})F'^*(x^{(n)})\{F(x^{(n)}) - y^{(\delta)} - F'(x^{(n)})(x^{(n)} - \hat{x})\}$$

$$- \theta(F'^*(x^{(n)})F'(x^{(n)}), \alpha^{(n)})F'^*(x^{(n)})F'(x^{(n)})(\xi^{(n)} - \hat{x}) - (\hat{x} - \xi^{(n)}). \qquad (2.27)$$

From (2.27) and modified source condition (2.24), one derives

$$x^{(n+1)} - \hat{x} = -\theta(F'^*(x^{(n)})F'(x^{(n)}), \alpha^{(n)})F'^*(x^{(n)})\{(y-y^{(\delta)}) + F(x^{(n)}) - y - F'(x^{(n)})(x^{(n)} - \hat{x})\}$$

$$-[I - \theta(F'^*(x^{(n)})F'(x^{(n)}), \alpha^{(n)})F'^*(x^{(n)})F'(x^{(n)})]\{(F'^*(x^{(n)})F'(x^{(n)}))^p\omega^{(n)} + \zeta^{(n)}\}. \quad (2.28)$$

The second inequality in (2.3) yields

$$||F(x^{(n)}) - y - F'(x^{(n)})(x^{(n)} - \hat{x})|| \le \frac{L}{2}||x^{(n)} - \hat{x}||^2. \qquad (2.29)$$

Taking into consideration (2.24), we can now find an upper bound for $||x^{(n+1)} - \hat{x}||$. Assumptions (2.6)-(2.8) on the generating function along with polar decomposition for the linear operator $F'(x^{(n)})$ and spectral theorem for the self-adjoint operator $F'^*(x^{(n)})F'(x^{(n)})$ imply

$$||x^{(n+1)} - \hat{x}|| \le \frac{C_1}{\sqrt{\alpha^{(n)}}}\left\{\delta + \frac{L||x^{(n)} - \hat{x}||^2}{2}\right\} + C_2(p)\varepsilon[\alpha^{(n)}]^{p-k} + C_3\Delta$$

$$= \frac{C_1}{\sqrt{\alpha^{(n)}}}\left(\delta + \frac{C_3\Delta\sqrt{\alpha^{(n)}}}{C_1}\right) + \frac{C_1 L}{2\sqrt{\alpha^{(n)}}}||x^{(n)} - \hat{x}||^2 + C_2(p)\varepsilon[\alpha^{(n)}]^{p-k}. \qquad (2.30)$$

2.4 A Novel A Posteriori Stopping Rule

Introduce the following stopping rule for iterations (2.21). Let $\mathcal{N} = \mathcal{N}(\delta, \Delta, y^{(\delta)})$ be the number of the first transition of $||F(x^{(n)}) - y^{(\delta)}||$ through the level σ_n^μ, $\frac{1}{2} \le \mu < 1$, i.e.,

$$||F(x^{(\mathcal{N}(\delta,\Delta,y^{(\delta)}))}) - y^{(\delta)}|| \le \sigma_{\mathcal{N}}^\mu \quad \text{and} \quad \sigma_n^\mu < ||F(x^{(n)}) - y^{(\delta)}||, \quad 0 \le n < \mathcal{N}(\delta, \Delta, y^{(\delta)}), \qquad (2.31)$$

where

$$\sigma_n := \delta + \frac{C_3 \Delta \sqrt{\alpha^{(n)}}}{C_1}. \tag{2.32}$$

The constant Δ in (2.32) is usually harder to estimate than the noise level δ. However for the asymptotic behavior of the approximate solution $x = x^{(\mathcal{N}(\delta,\Delta,y^{(\delta)}))}$ as δ and Δ tend to zero, this is not relevant. It follows from (2.32) that due to the factor $\sqrt{\alpha^{(n)}}$, the contribution of Δ to the total error of the model approaches zero as $n \to \infty$. In other words, error in the source condition disappears in the overall noise as we iterate. Notice that if $F'^*(\cdot)F'(\cdot)$ is compact and the null space of $F'^*(\cdot)F'(\cdot)$ is $\{0\}$, then the range of $F'^*(\cdot)F'(\cdot)$ is dense in $\mathcal{X}$, so in any neighborhood of $x^{(n)}$ there are points $\xi^{(n)}$ for which (2.24) holds with $\Delta = 0$. On the other hand, since the range of $F'^*(\cdot)F'(\cdot)$ is not closed, in the same neighborhood there are also points $\xi^{(n)}$ for which (2.24) holds with $\Delta \neq 0$. In practice, one can try different $\xi^{(n)}$'s and choose those for which the iterative scheme works better, that is convergence is more rapid and the algorithm is more stable. Suppose $n < \mathcal{N}(\delta, \Delta, y^{(\delta)})$. One has

$$\sigma_n^\mu < ||F(x^{(n)}) - y^{(\delta)}|| \le ||F(x^{(n)}) - y|| + ||y - y^{(\delta)}|| \le N||x^{(n)} - \hat{x}|| + \delta \le N||x^{(n)} - \hat{x}|| + \sigma_n. \tag{2.33}$$

Hence

$$\sigma_n^\mu - \sigma_n \le N||x^{(n)} - \hat{x}||.$$

Without loss of generality, assume that $\sigma_0 < 1$, which yields $\sigma_n < 1$ for any $n = 1, 2, ...$ due to our assumptions on $\{\alpha^{(n)}\}$. Then

$$\sigma_n^\mu (1 - \sigma_n^{1-\mu}) \le N||x^{(n)} - \hat{x}|| \quad \text{and} \quad \sigma_n^\mu \le \frac{N}{1 - \sigma_n^{1-\mu}} ||x^{(n)} - \hat{x}||.$$

The last inequality implies that combined noise level σ_n can be estimated as follows

$$\sigma_n \le \left(\frac{N}{1 - \sigma_0^{1-\mu}} \right)^{\frac{1}{\mu}} ||x^{(n)} - \hat{x}||^{\frac{1}{\mu}} := C||x^{(n)} - \hat{x}||^{\frac{1}{\mu}}.$$

One derives from the above that

$$||x^{(n+1)} - \hat{x}|| \leq \frac{C_1 C}{\sqrt{\alpha^{(n)}}}||x^{(n)} - \hat{x}||^{\frac{1}{\mu}} + \frac{C_1 L}{2\sqrt{\alpha^{(n)}}}||x^{(n)} - \hat{x}||^2 + C_2(p)\varepsilon[\alpha^{(n)}]^{p-k}. \tag{2.34}$$

By (2.26), $||x^{(0)} - \hat{x}|| \leq 1$. If one assumes that for any m, $0 \leq m \leq n < \mathcal{N}(\delta, \Delta, y^{(\delta)})$, $||x^{(m)} - \hat{x}|| \leq 1$, then since $\frac{1}{2} \leq \mu$

$$||x^{(n)} - \hat{x}||^2 \leq ||x^{(n)} - \hat{x}||^{\frac{1}{\mu}}. \tag{2.35}$$

Estimate (2.34) combined with (2.35) yield

$$||x^{(n+1)} - \hat{x}|| \leq \left\{ \frac{C_1 C}{\sqrt{\alpha^{(n)}}} + \frac{C_1 L}{2\sqrt{\alpha^{(n)}}} \right\} ||x^{(n)} - \hat{x}||^{\frac{1}{\mu}} + C_2(p)\varepsilon[\alpha^{(n)}]^{p-k}. \tag{2.36}$$

Consider a new variable

$$\gamma_n := \frac{||x^{(n)} - \hat{x}||}{[\alpha^{(n)}]^{p-k}}. \tag{2.37}$$

Inequalities (2.25) for $\{\alpha^{(n)}\}$ imply

$$\gamma_{n+1} \leq C_1 R \left\{ C + \frac{L}{2} \right\} \gamma_n^{\frac{1}{\mu}} [\alpha^{(n)}]^{(p-k)\left(\frac{1}{\mu}-1\right)-\frac{1}{2}} + C_2(p)R\varepsilon.$$

Let parameters in the source condition be restricted as follows:

$$(p-k)\left(\frac{1}{\mu} - 1\right) \geq \frac{1}{2}. \tag{2.38}$$

Take

$$l := \frac{C_2(p)R\varepsilon}{1 - C_1 R\left\{C + \frac{L}{2}\right\}[\alpha^{(0)}]^{(p-k)\left(\frac{1}{\mu}-1\right)-\frac{1}{2}}} := \frac{b}{1-a}, \quad \text{with} \quad a + b \leq 1.$$

Under these assumptions

$$l = \frac{b}{1-a} \leq 1.$$

According to (2.26), the initial guess x_0 is chosen in such a way that $\gamma_0 \leq l$. Suppose by induction that $\gamma_m \leq l$ for any m: $0 \leq m \leq n < \mathcal{N}(\delta, \Delta, y^{(\delta)})$. Then by monotonicity of the sequence $\{\alpha^{(n)}\}$

$$\gamma_{n+1} \le a l^{\frac{1}{\mu}} + b \le al + b = l, \quad \text{and} \quad ||x^{(n+1)} - \hat{x}|| \le l[\alpha^{(n+1)}]^{p-k} \le l[\alpha^{(0)}]^{p-k} \le 1. \quad (2.39)$$

From (2.39) it follows that $||x^{(n)} - \hat{x}|| \le l[\alpha^{(n)}]^{p-k}$ for any $n \le \mathcal{N}(\delta, \Delta, y^{(\delta)})$.

Now let us show that for any noise levels δ and Δ with $0 \le \sigma_0 < 1$, there exists a value $\mathcal{N} = \mathcal{N}(\delta, \Delta, y^{(\delta)})$ such that condition (2.31) holds. Indeed, if this were not the case, then for some $\tilde{\sigma}_n := \tilde{\delta} + \frac{C_3 \tilde{\Delta}\sqrt{\alpha^{(n)}}}{C_1}$,

$$\tilde{\sigma}_n^{\mu} < ||F(x^{(n)}) - y^{(\tilde{\delta})}|| \quad \text{for any} \quad n = 0, 1, 2.... \quad (2.40)$$

Provided that $x^{(0)}$ is chosen according to (2.26), estimates (2.30) and (3.11) (along with the above choice of l) imply

$$||x^{(n)} - \hat{x}|| \le l[\alpha^{(n)}]^{p-k} \quad \text{for all} \quad n \ge 0 \quad \text{and} \quad \lim_{n\to\infty} ||x^{(n)} - \hat{x}|| = 0. \quad (2.41)$$

If one passes to the limit in (3.11) as n approaches infinity, one gets by (2.2) and (2.41)

$$\tilde{\sigma}_n \ge \tilde{\delta} \ge \tilde{\sigma}_n^{\mu}, \quad \frac{1}{2} \le \mu < 1,$$

which means $\tilde{\sigma}_n^{1-\mu} \ge 1$, and $\tilde{\sigma}_n \ge 1$. We arrive at a contradiction. Hence $\mathcal{N} = \mathcal{N}(\delta, \Delta, y^{(\delta)})$ exists.

Suppose $\hat{\mathcal{X}} := \{x \in \mathcal{X} : F(x) = y\}$. We now verify that if the control sequence $\{\xi^{(n)}\}$ satisfies source condition (2.24), then $x^{(\mathcal{N}(\delta,\Delta,y^{(\delta)}))}$ converges to $\hat{\mathcal{X}}$ as $\sqrt{\delta^2 + \Delta^2} \to 0$, and the function $\mathcal{N}(\delta, \Delta, y^{(\delta)})$ is, therefore, admissible. Indeed, assume the converse: there exists $\epsilon > 0$ and $\{\delta_m\}$, $\{\Delta_m\}$, $\lim_{m\to\infty} \sqrt{\delta_m^2 + \Delta_m^2} = 0$, such that

$$\operatorname{dist}\left(x^{(\mathcal{N}(\delta_m,\Delta_m,y^{(\delta_m)}))}, \hat{\mathcal{X}}\right) > \epsilon. \quad (2.42)$$

Two cases are possible.

1. The sequence $\mathcal{N} = \mathcal{N}(\delta_m, \Delta_m, y^{(\delta_m)})$ is bounded, i.e., $\mathcal{N}(\delta_m, \Delta_m, y^{(\delta_m)}) \le \mathcal{N}_0$ for any

$m \geq 0$. Then there is a subsequence $\{m_z\}$, $\lim_{z\to\infty} m_z = \infty$, such that

$$\lim_{z\to\infty} \mathcal{N}(\delta_{m_z}, \Delta_{m_z}, y^{(\delta_{m_z})}) = \tilde{\mathcal{N}} \leq \mathcal{N}_0.$$

By stopping rule (2.31),

$$||F(x^{(\mathcal{N}(\delta_{m_z},\Delta_{m_z},y^{(\delta_{m_z})}))}) - y^{(\delta_{m_z})}|| \leq \sigma^{\mu}_{\mathcal{N}(\delta_{m_z},\Delta_{m_z},y^{(\delta_{m_z})})}. \tag{2.43}$$

Taking the limit in both sides of (2.43) as $z \to \infty$, one concludes that

$$||F(x^{(\tilde{\mathcal{N}})}) - y|| = 0.$$

Thus, $x^{(\tilde{\mathcal{N}})}$ is a solution to $F(x) = y$, i.e., $x^{(\tilde{\mathcal{N}})} \in \hat{\mathcal{X}}$, which indicates that inequality (2.42) is not fulfilled.

2. For some $\{\delta_{m_j}\}$, $\{\Delta_{m_j}\}$, $\lim_{j\to\infty} m_j = \infty$,

$$\mathcal{N}(\delta_{m_j}, \Delta_{m_j}, y^{(\delta_{m_j})}) \longrightarrow \infty \quad \text{as} \quad j \to \infty.$$

Then by the above argument,

$$||x^{(\mathcal{N}(\delta_{m_j},\Delta_{m_j},y^{(\delta_{m_j})}))} - \hat{x}|| \leq l\, \tau^{p-k}_{\mathcal{N}(\delta_{m_j},\Delta_{m_j},y^{(\delta_{m_j})})} \longrightarrow 0 \quad \text{as} \quad j \to \infty.$$

Once again, we arrive at a contradiction. Hence $\lim_{\sqrt{\delta^2+\Delta^2}\to 0} \operatorname{dist}\left(x_{\mathcal{N}(\delta,\Delta,y^{(\delta)})}, \hat{\mathcal{X}}\right) = 0$.

One can see from the above analysis that by extending the *a posteriori* stopping rule for the iteratively regularized numerical process (2.21) with an undetermined reverse connection, we considerably weaken the source condition (through letting it hold with a certain level of noise) without imposing any additional restrictions on the nonlinearity of the operator F. We now formulate this result as a theorem.

Theorem 2.1.

Let the following conditions hold:

1. Suppose that a nonlinear operator F is acting between two Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$, i.e., $F : \mathcal{X} \to \mathcal{Y}$, and $\hat{x} \in \mathcal{X}$ is a solution (not necessarily unique) to the equation

with exact data $F(x) = y$. The right-hand side y is given by its δ-approximation such that $||y - y^{(\delta)}|| \leq \delta$.

2. Assume that F is Fréchet differentiable and its derivative F' is bounded and Lipschitz continuous in the following region

$$\bar{\mathcal{B}}_\eta(\hat{x}) = \{x \in \mathcal{X} : ||x - \hat{x}|| \leq \eta\}, \quad \eta := l[\alpha^{(0)}]^{p-k}, \tag{2.44}$$

so that conditions (2.3) hold. In (2.44), p is defined in (2.24) and (2.7). The two remaining parameters k and l are defined in (2.24) and (2.45), respectively.

3. For every $n \in \mathbb{N}$, the sequence $\{x^{(n)}\}$ is generated according to (2.21) and the control elements $\xi^{(n)} \in \mathcal{X}$ are chosen by means of undetermined reverse connection (2.24), which is fulfilled with the level of noise Δ.

4. The generating function $\theta = \theta(\sigma, \alpha)$, $\sigma \in [0, N^2]$ and $\alpha \in (0, \infty)$, satisfies inequalities (2.6)-(2.8), while the regularizing sequence $\{\alpha^{(n)}\}$ satisfies (2.25).

5. For the initial value of the regularization parameter $\alpha^{(0)}$, condition (2.26) is fulfilled with

$$l := \frac{C_2(p)R\varepsilon}{1 - C_1 R\left\{C + \frac{L}{2}\right\}[\alpha^{(0)}]^{(p-k)\left(\frac{1}{\mu}-1\right)-\frac{1}{2}}} := \frac{b}{1-a}, \quad \text{and} \quad a + b \leq 1. \tag{2.45}$$

Here C_1 and $C_2(p)$ are defined in (2.6) and (2.7), ε is defined in (2.24) and

$$C := \left(\frac{N}{1 - \sigma_0^{1-\mu}}\right)^{\frac{1}{\mu}}, \quad \frac{1}{2} \leq \mu < 1, \quad (p-k)\left(\frac{1}{\mu} - 1\right) \geq \frac{1}{2}.$$

Then

1) the extended discrepancy principle is well-defined, i.e., there exists $\mathcal{N} = \mathcal{N}(\delta, \Delta, y_\delta)$ such that

$$||F(x^{(\mathcal{N}(\delta,\Delta,y^{(\delta)}))}) - y^{(\delta)}|| \leq \sigma_{\mathcal{N}}^\mu \quad \text{and} \quad \sigma_n^\mu < ||F(x^{(n)}) - y^{(\delta)}||, \quad 0 \leq n < \mathcal{N}(\delta, \Delta, y^{(\delta)});$$

2) the function $\mathcal{N}(\delta, \Delta, y^{(\delta)})$ is admissible, that is, $\lim_{\sqrt{\delta^2+\Delta^2}\to 0} \operatorname{dist}\left(x^{(\mathcal{N}(\delta,\Delta,y^{(\delta)}))}, \hat{\mathcal{X}}\right) = 0$ for $\hat{\mathcal{X}} := \{x \in \mathcal{X} : F(x) = y\}$;

3) the following estimate holds

$$||x^{(n)} - \hat{x}|| \leq l[\alpha^{(n)}]^{p-k}, \quad n = 0, 1, 2..., \mathcal{N}(\delta, \Delta, y^{(\delta)}). \tag{2.46}$$

2.5 Numerical Aspect: Linear Image Reconstruction Problem

To illustrate various aspects of numerical implementation of algorithm (2.21), we examine a highly important image restoration problem, where the purpose is to obtain the image of the original scene from an output that is blurred and noise contaminated. This problem is often modeled as a large-scale linear system, $Ax = b$, with A and b representing the blurring matrix and the blurred output, respectively. In most cases, the right-hand side b is given by its δ-approximation $b^{(\delta)}$, $||b - b^{(\delta)}|| \leq \delta$. As it has been pointed out in [32], the blur is generally more significant than the additive noise, $b^{(\delta)} - b$. Thus the emphasis of image restoration is on removing the blur, which can occur for a variety of reasons, such as camera shake and/or misfocus, atmospheric turbulence, and other sources.

To simulate this linear inverse problem, we utilize RestoreTools Matlab software developed by J. Nagy and his group [33], which provides a subfunction generating A from a given point spread function PSF, i.e., a function that specifies how points in the image are distorted.

With RestoreTools [33], we load the point spread image, which is a 256-by-256 matrix called PSF in our program, and use the function $psfMatrix(\ )$ to create the blurring matrix A. After that, we read in a true image, x_true_large, and resize it to a 256-by-256 matrix. The blurring matrix and the true image are used to generate the blurred output, b. Then we add random, normally distributed, noise to the result in order to simulate the data, $b^{(\delta)}$, for the underlying inverse problem. Since our image is colorful, we actually have an 256-by-256-by-3 array for x_true_3. The goal is to reconstruct the exact image, x, from a blurred/noisy image, $b^{(\delta)}$, by solving the equation $Ax = b^{(\delta)}$. For colorful images, we reconstruct the red, green, and blue images separately.

The linear system under consideration is large-scale and severely ill-posed. Therefore, a regularized algorithm needs to be implemented. Since this problem is linear, given

Table 2.1: Noise-free reconstruction

$\|\|b - b^{(\delta)}\|\|/\|\|b^{(\delta)}\|\| = 0, \quad x\_initial = b^{(\delta)}, \quad \alpha^{(n)} = 0.75^n$			
	Mountain Produce		
Algorithm	$\alpha^{(\mathcal{N})}$	$\mathcal{N}$	$\|\|Ax^{(\mathcal{N})} - b^{(\delta)}\|\|/\|\|b^{(\delta)}\|\|$
Tikhonov	4.247×10^{-7}	51	8.249×10^{-5}
TSVD	2.386×10^{-6}	45	1.440×10^{-4}
MTSVD	3.182×10^{-6}	44	8.755×10^{-5}
	Tea Leaves		
Algorithm	$\alpha^{(\mathcal{N})}$	$\mathcal{N}$	$\|\|Ax^{(\mathcal{N})} - b^{(\delta)}\|\|/\|\|b^{(\delta)}\|\|$
Tikhonov	4.247×10^{-7}	51	9.140×10^{-5}
TSVD	2.386×10^{-6}	45	1.826×10^{-4}
MTSVD	2.386×10^{-6}	45	9.929×10^{-5}

a proper value of α one can solve the regularized equation directly and obtain an approximate solution in the form $x_{\alpha,\delta} = \theta(A^*A, \alpha)A^*y^{(\delta)} := R_\alpha y^{(\delta)}$. However, the large size of matrix A makes this approach very difficult. Moreover, to implement a direct method, one has to find a "nearly optimal" regularization parameter, which may result in solving an extra (sometimes nonlinear) problem. Additionally, in cases like L-curve, the parameter selection procedure would only provide an insight into the choice of α rather than a justified algorithm. So, instead of using a direct solver, we apply iteratively regularized algorithm (2.21) with three different generating functions

- $\theta_1(\sigma, \alpha) = \frac{1}{\sigma+\alpha}$, the original iteratively regularized Tikhonov scheme;
- $\theta_2(\sigma, \alpha) := \begin{cases} \frac{1}{\sigma}, & \sigma \geq \alpha, \\ 0, & 0 \leq \sigma < \alpha \end{cases}$, the classical iteratively truncated procedure;
- $\theta_3(\sigma, \alpha)\sqrt{\sigma} := \begin{cases} \frac{1}{\sqrt{\sigma}}, & \sigma \geq \alpha, \\ \frac{1}{\sqrt{\alpha}}, & 0 \leq \sigma < \alpha \end{cases}$, the modified iterative truncation.

For (2.21), there is no question of "optimal parameter". In place of "optimal parameter", one needs a reliable stopping rule to get an accurate result in case of noisy data. The application of stopping rule (2.31)-(2.32) for the image restoration problem is simplified by the fact that, as we have already mentioned in section 2.2, for a linear problem the convergence analysis can be carried out without any source conditions. Therefore we are

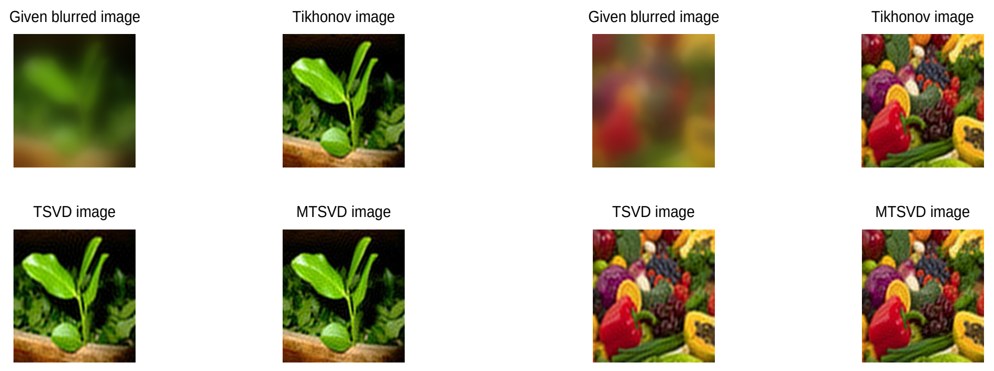

Figure 2.1: Blurred image and three reconstructed images. Noise-free case

free to use constants C_1, C_3, μ, and Δ in (2.31)-(2.32) as control parameters and select the values for which the stopping time is most accurate. In fact, the same approach can also be used for nonlinear problems, since source condition (2.24) is not algorithmically verifiable, in general.

Our method for choosing C_1, C_3, μ, and Δ is as follows. Let $||b - b^{(\delta)}|| \leq \delta$. Assume that $\Delta \approx \delta$ in (2.32). Then

$$\sigma_n = \left(1 + \frac{C_3\sqrt{\alpha^{(n)}}}{C_1}\right)\delta \leq \left(1 + \frac{C_3\sqrt{\alpha^{(0)}}}{C_1}\right)\delta := c\,\delta.$$

The constant c is used as a control parameter, i.e., the experiment for a test image and for a test value of $\delta/||b^{(\delta)}||$ is conducted with $c = 1, 2, ..., 10$. The best result is achieved for $c = 8$. The constant μ in (2.31) is viewed as another control parameter. Even though the stopping rule is justified for $0.5 \leq \mu < 1$, for the test experiment we try $\mu = 0.5, 0.75, 1, 1.25, 1.5$. The best accuracy is attained for $\mu = 1$. With $c = 8$ and $\mu = 1$, we perform simulations for all other images and values of δ, and stop iterations when

$$\left| \frac{||Ax^{(\mathcal{N})} - b^{(\delta)}||}{||b^{(\delta)}||} - \frac{c\delta}{||b^{(\delta)}||} \right| < 10^{-4}. \tag{2.47}$$

Since A is a $256^2 \times 256^2$ matrix and both exact image, x, and blurred image, b, are 256×256 matrices, x must be compressed into a $256^2 \times 1$ vector for the multiplication of A by x to be carried out. Still, considering the size of A, it is too large to be stored.

Table 2.2: Reconstruction with relative noise 0.1 %

$\lvert\lvert b - b^{(\delta)}\rvert\rvert/\lvert\lvert b^{(\delta)}\rvert\rvert = 0.001,\quad x\_initial = b^{(\delta)},\quad \alpha^{(n)} = 0.75^n,\quad c = 8,\quad \mu = 1$			
	Mountain Produce		
Algorithm	$\alpha^{(\mathcal{N})}$	$\mathcal{N}$	$\lvert\lvert Ax^{(\mathcal{N})} - b^{(\delta)}\rvert\rvert/\lvert\lvert b^{(\delta)}\rvert\rvert$
Tikhonov	2.381×10^{-4}	29	8.050×10^{-3}
TSVD	3.175×10^{-4}	28	8.035×10^{-3}
MTSVD	1.338×10^{-3}	23	8.025×10^{-3}
	Tea Leaves		
Algorithm	$\alpha^{(\mathcal{N})}$	$\mathcal{N}$	$\lvert\lvert Ax^{(\mathcal{N})} - b^{(\delta)}\rvert\rvert/\lvert\lvert b^{(\delta)}\rvert\rvert$
Tikhonov	1.786×10^{-4}	30	8.067×10^{-3}
TSVD	2.381×10^{-4}	29	8.015×10^{-3}
MTSVD	1.003×10^{-3}	24	8.019×10^{-3}

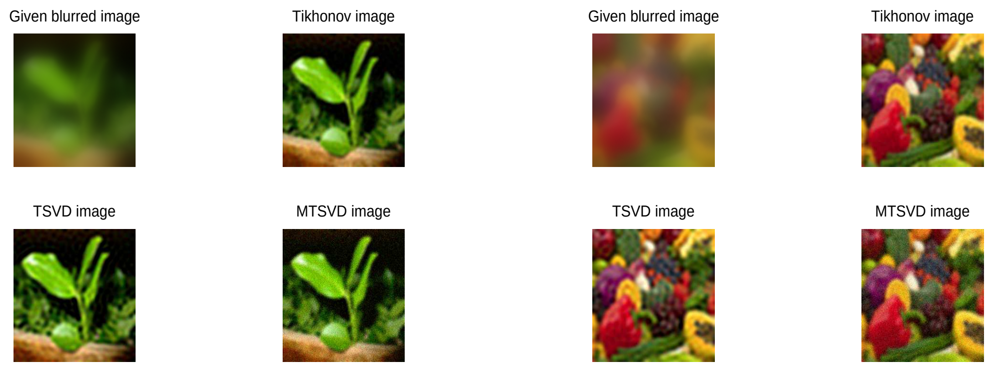

Figure 2.2: Blurred image and three reconstructed images. Relative noise 0.1%

In the object oriented Matlab package RestoreTools [33], matrix vector multiplication is done using the two dimensional discrete Fourier transform provided that the blur is spatially invariant and the boundary conditions are periodic. Thus, the right hand side, b, is generated as $b = A * x\_true$ by overloading the $*$ operator. Then algorithm (2.21) is implemented with the use of the singular value decomposition.

For all three generating functions, the blurred/noisy image, $b^{(\delta)}$ is used as an initial approximation, $x\_initial$, and the regularization sequence is set to be $\alpha^{(n)} = 0.75^n$. While $x\_initial = b^{(\delta)}$ is by far the best choice of the initial guess we have tried, the choice of $\alpha^{(n)}$ does not seem to be that important as long as assumption of Theorem 2.1 are fulfilled, i.e., the change of $\alpha^{(n)}$ would only change the stopping time but not the

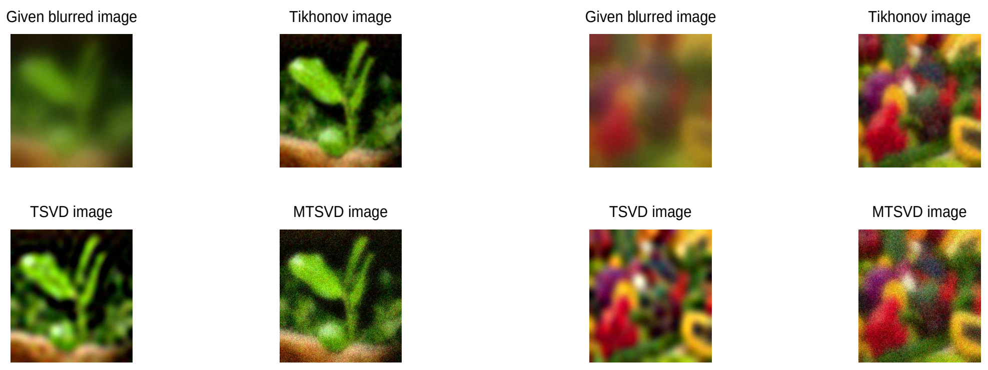

Figure 2.3: Blurred image and three reconstructed images. Relative noise 0.5%

accuracy of reconstruction.

The experiment has been conducted for multiple images and different values of δ in order to make sure that the choice of control parameters depends neither on the image to be recovered nor on the noise level. Figures 2.1, 2.2, and 2.3 along with Tables 2.1, 2.2, and 2.3 illustrate numerical results for two images, "Mountain Produce" and "Tea Leaves". The original pictures, used to simulated blurred/noisy data, have been taken from [34] and [35], respectively.

Figure 2.1 shows a nearly perfect reconstruction in the noise-free case. In the absence of noise, iterations were not stopped until the discrepancy started to get worse due to rounding errors. The values of the regularization parameters and the relative discrepancies at the stopping time are displayed in Table 2.1.

In the presence of 0.1% relative noise in the right-hand side, the reconstruction is still rather accurate (see Figure 2.2 and Table 2.2 for details). For both images, the iterations were stopped by the practical method (2.47). As the noise level goes up to 0.5%, one can observe a reduction in the approximate image quality, but even in that case it is acceptable.

2.6 Computational Study of Nonlinear Inverse Scattering Model

In this section, we consider an inverse problem of identifying the shape of a 2D obstacle from far-field scattering data [36]. By parameterizing the boundary with polar

Table 2.3: Reconstruction with relative noise 0.5 %

$\|\|b-b^{(\delta)}\|\|/\|\|b^{(\delta)}\|\| = 0.005$, $x\_initial = b^{(\delta)}$, $\alpha^{(n)} = 0.75^n$, $c = 8$, $\mu = 1$			
	Mountain Produce		
Algorithm	$\alpha^{(\mathcal{N})}$	$\mathcal{N}$	$\|\|Ax^{(\mathcal{N})}-b^{(\delta)}\|\|/\|\|b^{(\delta)}\|\|$
Tikhonov	2.378×10^{-3}	21	3.994×10^{-2}
TSVD	1.784×10^{-3}	22	3.993×10^{-2}
MTSVD	1.782×10^{-2}	14	4.008×10^{-2}
	Tea Leaves		
Algorithm	$\alpha^{(\mathcal{N})}$	$\mathcal{N}$	$\|\|Ax^{(\mathcal{N})}-b^{(\delta)}\|\|/\|\|b^{(\delta)}\|\|$
Tikhonov	1.338×10^{-3}	23	3.990×10^{-2}
TSVD	1.003×10^{-3}	24	3.995×10^{-2}
MTSVD	1.002×10^{-2}	16	4.009×10^{-2}

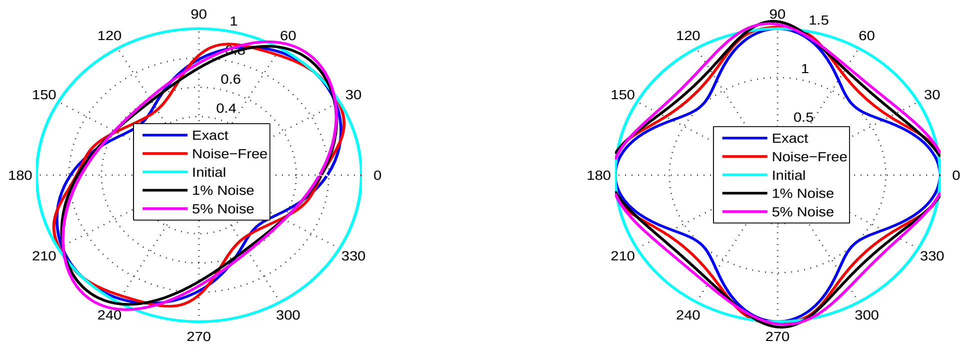

Figure 2.4: Reconstruction with Tikhonov-type algorithm

coordinates, this problem may be reduced to nonlinear integral equation of the first kind, $F(r) = y$, where ([37], [38], [39])

$$F(r) := \int_0^{2\pi} \mathcal{Q}(\psi, \phi, r(\psi))\, d\psi, \quad F : \mathcal{X} \to \mathcal{Y}, \tag{2.48}$$

and

$$\mathcal{Q}(\psi, \phi, r) = [\exp(\beta r)(\beta r - 1) + 1]/\beta^2, \quad \beta = -i2k_0 \cos(\phi - \psi). \tag{2.49}$$

In the above, k_0 is a single fixed wave-number, where data is available. The noisy data, $y^{(\delta)}$, is such that $||y - y^{(\delta)}|| \le \delta$, and the noise is due to Born approximation as well as due to imperfect measurements. As in [37], it is assumed that $y = y(\phi)$ has a 120^o

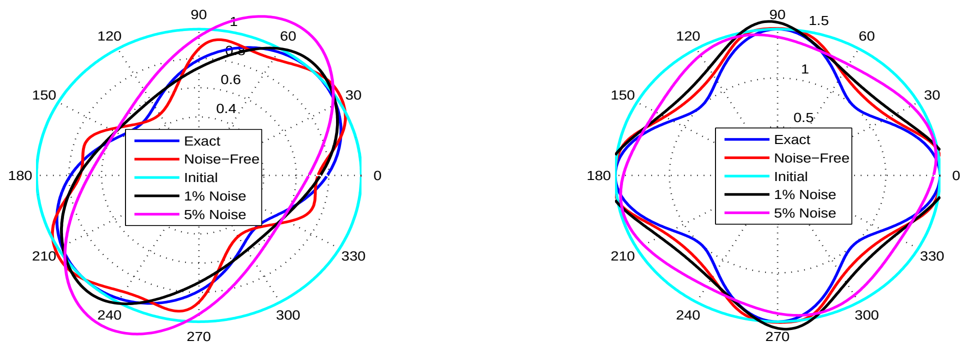

Figure 2.5: Reconstruction with TSVD algorithm

Table 2.4: Noise-free reconstruction of a peanut-shaped object

$r_0(\psi) = 1, \quad \alpha^{(n)} = 10^{-5}/(n+1), \quad \xi^{(n)}(\psi) = r_n(\psi)$					
Algorithm	$\mathcal{N}$	$\alpha^{(\mathcal{N})}$	cond(F')	$\frac{\|\|F(r_\mathcal{N})-y^{(\delta)}\|\|}{\|\|y^{(\delta)}\|\|}$	$\frac{\|\|r_\mathcal{N}-r_{mod}\|\|}{\|\|r_{mod}\|\|}$
Tikhonov	10	1.0×10^{-6}	6.46×10^{17}	7.460×10^{-4}	3.775×10^{-2}
TSVD	30	3.3×10^{-7}	4.44×10^{17}	7.128×10^{-3}	7.071×10^{-2}
MTSVD	32	3.1×10^{-7}	4.84×10^{17}	$1.220 \times 10-4$	1.902×10^{-2}

aperture, i.e., $\phi \in [0, 2\pi/3]$. Set $\mathcal{X} = L_2[0, 2\pi]$ and $\mathcal{Y} = L_2[0, 2\pi/3]$. In order to compare strengths and weaknesses of generating functions θ_j, $j = 1, 2, 3$, and to investigate the efficiency of stopping rule (2.31)-(2.32), we simulate exact scattering data for two model solutions:

- "Peanut": $r_{mod}(\psi) := \left(\cos^2\left(\psi - \frac{\pi}{4}\right) + 0.25\sin^2\left(\psi - \frac{\pi}{4}\right)\right)^{1/2}$,

- "Pillow": $r_{mod}(\psi) := 1.25 + 0.25\cos(4\psi)$, $\psi \in [0, 2\pi]$,

using a high-accuracy built-in Matlab integration subfunction. Notice that because the kernel of the equation is complex valued and the exact solution is real, the simulated right-hand side, y, (the exact data) is complex valued. Hence, when certain percentage of noise is added to the data, the noise is randomly distributed between real and imaginary parts of $y^{(\delta)}$. The reconstructed solution, $r_\mathcal{N}$, is, in general, complex valued. This is the case even when the error in the right-hand side is due to discretization only. However, since for this particular application it is known *a priori* that the true solution is real, we

Table 2.5: Noise-free reconstruction of a pillow-shaped object

$r_0(\psi) = 1.5, \quad \alpha^{(n)} = 10^{-5}/(n+1), \quad \xi^{(n)}(\psi) = r_n(\psi)$					
Algorithm	$\mathcal{N}$	$\alpha^{(\mathcal{N})}$	cond(F')	$\frac{\lVert F(r_{\mathcal{N}})-y^{(\delta)}\rVert}{\lVert y^{(\delta)}\rVert}$	$\frac{\lVert r_{\mathcal{N}}-r_{mod}\rVert}{\lVert r_{mod}\rVert}$
Tikhonov	200	5.0×10^{-8}	3.56×10^{17}	4.922×10^{-4}	5.376×10^{-2}
TSVD	99	1.0×10^{-7}	2.45×10^{17}	2.276×10^{-3}	5.888×10^{-2}
MTSVD	200	5.0×10^{-8}	2.20×10^{17}	7.271×10^{-4}	5.541×10^{-2}

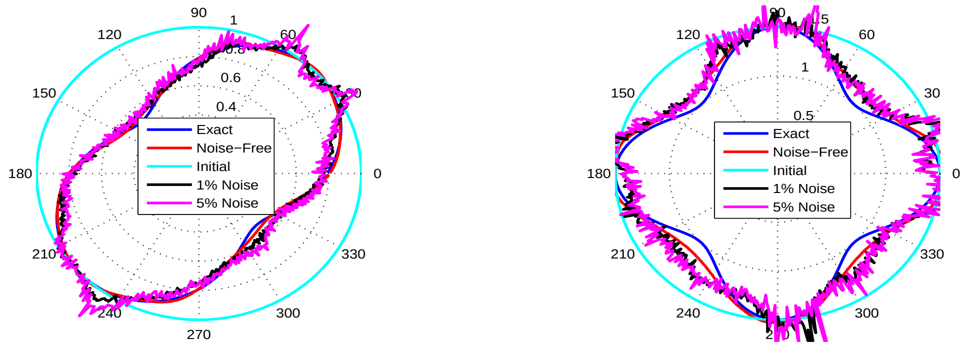

Figure 2.6: Reconstruction with MTSVD algorithm

view the real part of $r_{\mathcal{N}}$ as the approximate shape of the object once the iterations have been terminated. In Figures 2.4-2.6, it is the real part of $r_{\mathcal{N}}$ that is compared to r_{mod}. As in the previous section, the first set of experiments has been conducted for the noise-free case, i.e., in the presence of discretization error of order 10^{-6} only. Due to instability and the lack of data, even for $\delta = 0$, the scheme still needs to be regularized. In the noise-free case, we use $\alpha^{(n)} = 10^{-5}/(n+1)$ and stop iterations when the discrepancy starts increasing.

Table 2.6: Reconstruction of a peanut-shaped object in the presence of 1% noise

$r_0(\psi) = 1, \quad \alpha^{(n)} = 10^{2}/(n+1), \quad \xi^{(n)}(\psi) = r_n(\psi)$					
Algorithm	$\mathcal{N}$	$\alpha^{(\mathcal{N})}$	cond(F')	$\frac{\lVert F(r_{\mathcal{N}})-y^{(\delta)}\rVert}{\lVert y^{(\delta)}\rVert}$	$\frac{\lVert r_{\mathcal{N}}-r_{mod}\rVert}{\lVert r_{mod}\rVert}$
Tikhonov	26	3.85	5.737×10^{17}	1.004×10^{-2}	5.381×10^{-2}
TSVD	100	1.00	6.340×10^{17}	1.005×10^{-2}	5.289×10^{-2}
MTSVD	16	6.25	4.758×10^{17}	1.013×10^{-2}	4.503×10^{-2}

Table 2.7: Reconstruction of a pillow-shaped object in the presence of 1% noise

$r_0(\psi) = 1.5, \quad \alpha^{(n)} = 10^2/(n+1), \quad \xi^{(n)}(\psi) = r_n(\psi)$					
Algorithm	$\mathcal{N}$	$\alpha^{(\mathcal{N})}$	cond(F')	$\frac{\lVert F(r_{\mathcal{N}})-y^{(\delta)}\rVert}{\lVert y^{(\delta)}\rVert}$	$\frac{\lVert r_{\mathcal{N}}-r_{mod}\rVert}{\lVert r_{mod}\rVert}$
Tikhonov	100	1.000	2.715×10^{17}	1.013×10^{-2}	7.371×10^{-2}
TSVD	100	1.000	2.567×10^{17}	1.187×10^{-2}	8.151×10^{-2}
MTSVD	100	1.000	2.566×10^{17}	2.268×10^{-2}	1.792×10^{-1}

For all three generating functions, we set $\xi^{(n)} = r_n$. For θ_1, this results in a well-known Iteratively Regularized Levenberg-Marquardt (IRLM) algorithm. For this particular nonlinear inverse problem, that choice of $\xi^{(n)}$ was by far the best among those we tried. For both model solutions $r_{mod}(\psi)$, no *a priori* information on the shape of the object was assumed to be available. In each case, an outer circle was used as initial guess (see Figures 2.4-2.6).

The next two experiments were carried out in the presence of random noise. When the data is corrupted, the initial value of the regularization parameter $\alpha^{(0)}$ needs to be substantially increased (even more so due to the fact that $\xi^{(n)} = r_n$). For the levels of noise 1% and 5%, $\alpha^{(0)} = 10^2$ and $\alpha^{(0)} = 10^3$ are used, respectively. Thus, the term $\frac{C_3\sqrt{\alpha^{(n)}}}{C_1}$ in stopping rule (2.31)-(2.32) becomes very important. Taking that into account, we modify our strategy for choosing C_1, C_3, μ, and Δ as follows. Once again, we assume $\Delta \approx \delta$ and $\mu = 1$. Define

$$\sigma_n = \left(1 + \frac{C_3\sqrt{\alpha^{(n)}}}{C_1}\right)\delta := (1 + \tilde{c}\sqrt{\alpha^{(n)}})\,\delta.$$

The constant $\tilde{c}$ is viewed as a control parameter, and tests are conducted for $\tilde{c} \in [10^{-3}, 10^{-1}]$. Based on the tests, $\tilde{c} = 0.002$ is taken. With $\tilde{c} = 0.002$ and $\mu = 1$, simulations are performed for all other data sets and values of δ, and iterations are terminated when the condition

$$\frac{||F(r_n) - y^{(\delta)}||}{||y^{(\delta)}||} < \frac{(1 + \tilde{c}\sqrt{\alpha^{(n)}})\delta}{||y^{(\delta)}||} \tag{2.50}$$

Table 2.8: Reconstruction of a peanut-shaped object in the presence of 5% noise

$r_0(\psi) = 1, \quad \alpha^{(n)} = 10^3/(n+1), \quad \xi^{(n)}(\psi) = r_n(\psi)$					
Algorithm	$\mathcal{N}$	$\alpha^{(\mathcal{N})}$	cond(F')	$\frac{\|\|F(r_{\mathcal{N}})-y^{(\delta)}\|\|}{\|\|y^{(\delta)}\|\|}$	$\frac{\|\|r_{\mathcal{N}}-r_{mod}\|\|}{\|\|r_{mod}\|\|}$
Tikhonov	36	2.8×10^1	6.64×10^{17}	5.083×10^{-2}	6.377×10^{-2}
TSVD	100	1.0×10^1	5.35×10^{17}	9.210×10^{-2}	1.404×10^{-1}
MTSVD	17	5.9×10^1	5.34×10^{17}	5.089×10^{-2}	6.894×10^{-2}

Table 2.9: Reconstruction of a pillow-shaped object in the presence of 5% noise

$r_0(\psi) = 1.5, \quad \alpha^{(n)} = 10^3/(n+1), \quad \xi^{(n)}(\psi) = r_n(\psi)$					
Algorithm	$\mathcal{N}$	$\alpha^{(\mathcal{N})}$	cond(F')	$\frac{\|\|F(r_{\mathcal{N}})-y^{(\delta)}\|\|}{\|\|y^{(\delta)}\|\|}$	$\frac{\|\|r_{\mathcal{N}}-r_{mod}\|\|}{\|\|r_{mod}\|\|}$
Tikhonov	62	1.6×10^1	2.53×10^{17}	5.043×10^{-2}	9.976×10^{-2}
TSVD	100	1.0×10^1	4.23×10^{17}	6.782×10^{-2}	1.552×10^{-1}
MTSVD	58	1.7×10^1	2.20×10^{17}	5.055×10^{-2}	1.618×10^{-1}

is fulfilled for the first time, or when the number of iterations reaches 100.

In the noise-free case, IRML and MTSVD algorithms were equally accurate with the second one being a bit more reliable, as Tables 2.4 and 2.5 illustrate. For noise contaminated data, both IRLM and MTSVD have their advantages and disadvantages. While IRLM method definitely does a great job of producing stable and smooth solutions, it is not capable of actually approximating a peanut for the first $r_{mod}(\psi)$ and returns an ellipse instead. MTSVD regularizes too "gently", but the approximate curves follow the shape of a peanut more closely, even when the noise is 5%. The same happens with the second $r_{mod}(\psi)$: its shape is reproduced by MTSVD much better though the IRLM curve is more smooth and stable.

2.7 Conclusion

Numerical simulations conducted for a large-scale image restoration system, indicate that our practical stopping rule (2.47) is efficient. For this particular problem, it allows selection of the control parameters using a test image and a test value of δ. Afterwards, the selected parameters can be used on multiple other images and noise levels. From our experience, it has been easier to adjust the control parameters in the stopping rule rather than to find an “optimal” value of α to be used with a direct solver. Besides, the direct solver is hard to implement in case of the above example considering the matrix size.

For the nonlinear inverse scattering problem, the stopping rule also proved to work well. Its successful implementation confirms that one does not need to derive a very accurate estimate for the constants $C_3\Delta/C_1$ and μ in order to terminate generalized Gauss-Newton iterations with our proposed stopping rule. What is more important, is the knowledge that the coefficient with Δ in (2.32) is $O(\sqrt{\alpha^{(n)}})$. Hence, as $\alpha^{(n)} \to 0$, the contribution of the noise due to the source condition is dwindling. For mildly unstable ill-posed problems, when $\alpha^{(0)}$ can be small and $\alpha^{(n)}$ can be driven to zero pretty fast, the value of Δ is practically irrelevant. At the same time, for severely ill-posed problems, Δ becomes more “dangerous”, and therefore the choice of initial guess gets more complicated.

To summarize, the main difference between the new stopping rule and the classical discrepancy principle ([28], [24]) is that in the classical discrepancy principle the transition moment is $c\delta$ with $c > 1$. In our stopping rule, the transition time is $(\delta + C\sqrt{\alpha^{(n)}})^{\mu}$, where C can (and should) be viewed as a control parameter. So, asymptotically, for $\mu \approx 1$, the new rule is less aggressive while it allows for extra noise: noise due to the violation of the source condition.

Chapter 3

THEORETICAL AND NUMERICAL STUDY OF ITERATIVELY TRUNCATED NEWTON'S ALGORITHM

As shown in the previous chapter, the modified truncation method (MTSVD) tends to stabilize too "gently", and the solution turns out to be under-regularized. While this solution follows the shape of the model curves more closely compared to Tikhonov's and classical truncated method (TSVD), it appears less stable and less noise resistant.

In Chapter 3, we focus on another method of regularizing Quasi-Newton iterations. This method is based on spectral cut off and gives rise to Iteratively Truncated Newton's (ITN) scheme, which can be used for solving nonlinear irregular operator equations and unstable minimization problems. This algorithm is, in fact, a special case of a general procedure developed in [23]. However, convergence and stability analysis conducted in [23] is not applicable here since the generating function is not analytic. Therefore, this chapter presents an independent study of ITN method, which is carried out under the source condition that is the weakest possible if no restrictions on the structure of the operator are imposed. As a practical example, a 2D nonlinear inverse magnetometry problem ([40], [41], [42]) is considered to illustrate advantages and limitations of the proposed algorithm. Presentation in this chapter follows [43].

3.1 Introduction

Consider a nonlinear inverse problem in the form of the operator equation

$$F(x) = y, \qquad F : \mathcal{D}_F \subset \mathcal{X} \to \mathcal{Y}, \tag{3.1}$$

on a pair of two real Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$. Suppose that problem (3.1) is known to be solvable, maybe non-uniquely, and $\hat{x} \in \mathcal{D}_F$ is a solution of interest. Let F be Fréchet differentiable in a neighborhood of $\hat{x}$, and its derivative be a compact operator between

$\mathcal{X}$ and $\mathcal{Y}$. In case when $\mathcal{X}$ is infinite dimensional, this indicates that the problem is necessarily ill-posed, and classical Newton-type iterations are generally undefined. To overcome the lack of stability, we regularize Newton's step as follows:

$$x^{(n+1)} = P^{(n)}x^{(n)} + (I - P^{(n)})\xi - Q^{(n)}\left[F(x^{(n)}) - y\right], \qquad \xi, x^{(0)} \in \mathcal{D}_F \subset \mathcal{X}. \tag{3.2}$$

Here $Q^{(n)}$ is an iteratively regularized pseudo-inverse to $F'(x^{(n)})$ defined by means of the truncation function $\nu(\alpha^{(n)}, \mu_j^{(n)})$

$$Q^{(n)} := \sum_{j=1}^{\infty} \frac{\nu(\alpha^{(n)}, \mu_j^{(n)})}{\mu_j^{(n)}}(\,\cdot\,, v_j^{(n)})u_j^{(n)}, \quad \text{and} \quad \nu(\alpha^{(n)}, \mu_j^{(n)}) := \begin{cases} 1, & \mu_j^{(n)} \geq \alpha^{(n)}, \\ 0, & \mu_j^{(n)} < \alpha^{(n)}, \end{cases} \tag{3.3}$$

with $(\mu_j^{(n)}, u_j^{(n)}, v_j^{(n)})$, $j = 1, 2, ...$, being the singular system of $F'(x^{(n)})$. If one assumes that $J^{(n)}$ is the number of singular values exceeding the threshold $\alpha^{(n)} > 0$, then

$$P^{(n)} := \sum_{j=1}^{J^{(n)}}(\,\cdot\,, u_j^{(n)})u_j^{(n)}$$

is the orthogonal projector into the subspace spanned by the first $J^{(n)}$ eigenvectors of the operator $F'^{*}(x^{(n)})F'(x^{(n)})$.

From a practical standpoint, it has been observed that algorithm (3.2) is very robust and regularization in (3.2) is more accurate in the sense that only "small" singular values that essentially magnify noise get regularized (truncated), while other singular values remain unchanged (unlike the case of iteratively regularized Gauss-Newton scheme, for example). As always, the definition of "small" depends on the amount of noise in the model. In this dissertation, the choice of a threshold level $\alpha^{(n)}$ based on various types of error-perturbation is investigated.

In the next section, the convergence analysis of the iteratively regularized algorithm (3.2) is carried out, and the main convergence result Theorem 3.1 is formulated. The influence of instrumental errors on measured data y, as well as the error in the smoothness assumption on the initial guess, is investigated in section 3.3. To illustrate theoretical findings, Newton-type method (3.2) is used to solve an inverse problem in gravitational sounding, which takes the form of a 2D integral equation of the first kind. Our conclusions based on numerical simulations are presented in section 3.4.

3.2 Convergence Analysis. Noise-Free Data

Suppose that Fréchet derivative F' is Lipschitz continuous in a neighborhood of $\hat{x}$

$$||F'(u) - F'(v)|| \leq \mathcal{M}||u - v|| \quad \text{for any} \quad u, v \in \mathcal{B}(\hat{x}), \tag{3.4}$$

and some $\mathcal{M} \geq 0$. Here $\mathcal{B}(\hat{x})$ is the ball of radius $\hat{l}\alpha^{(0)}$ centered at $\hat{x}$ with $\hat{l}$ and $\alpha^{(0)}$ specified in (3.12) and (3.15) below. From identity (3.2), one concludes

$$x^{(n+1)} - \hat{x} = P^{(n)}x^{(n)} + (I - P^{(n)})\xi - Q^{(n)}F'(x^{(n)})(x^{(n)} - \hat{x})$$

$$-Q^{(n)}\left\{F(x^{(n)}) - y - F'(x^{(n)})(x^{(n)} - \hat{x})\right\} - \hat{x}.$$

Under assumption (3.4),

$$||F(x^{(n)}) - y - F'(x^{(n)})(x^{(n)} - \hat{x})|| \leq \frac{\mathcal{M}}{2}||x^{(n)} - \hat{x}||^2. \tag{3.5}$$

Clearly,

$$Q^{(n)}F'(x^{(n)}) = \sum_{j=1}^{J^{(n)}} \frac{1}{\mu_j^{(n)}}(F'(x^{(n)})\cdot\,, v_j^{(n)})u_j^{(n)} = \sum_{j=1}^{J^{(n)}} \frac{1}{\mu_j^{(n)}}(\,\cdot\,, F'^*(x^{(n)})v_j^{(n)})u_j^{(n)}$$

$$= \sum_{j=1}^{J^{(n)}} \frac{1}{\mu_j^{(n)}}(\,\cdot\,, \mu_j^{(n)}u_j^{(n)})u_j^{(n)} = P^{(n)}, \tag{3.6}$$

and therefore

$$x^{(n+1)} - \hat{x} = (I - P^{(n)})(\xi - \hat{x}) - Q^{(n)}\left\{F(x^{(n)}) - y - F'(x^{(n)})(x^{(n)} - \hat{x})\right\}.$$

Let the source-type condition be satisfied for the test value ξ in the following form

$$F'^*(\hat{x})w = \xi - \hat{x}, \qquad w \in \mathcal{Y}, \tag{3.7}$$

which is equivalent to the Hölder source condition

$$(F'^*(\hat{x})F'(\hat{x}))^p \omega = \xi - \hat{x}, \qquad \omega \in \mathcal{X},$$

with $p = \frac{1}{2}$ and $||w|| = ||\omega||$. This is the least value of the exponent that would guarantee convergence of a generalized Newton-type scheme without further assumptions on the nonlinearity of the operator F. Then

$$(I - P^{(n)})(\xi - \hat{x}) = (I - P^{(n)})[F'(\hat{x}) - F'(x^{(n)})]^* w + (I - P^{(n)})F'^*(x^{(n)})w. \tag{3.8}$$

The last term in (3.8) is, in fact,

$$(I - P^{(n)})F'^*(x^{(n)})w = \sum_{j=J^{(n)}+1}^{\infty} (F'^*(x^{(n)})w, u_j^{(n)})u_j^{(n)} = \sum_{j=J^{(n)}+1}^{\infty} \mu_j^{(n)}(w, v_j^{(n)})u_j^{(n)}.$$

Hence its norm is $o(\alpha^{(n)})$:

$$||(I - P^{(n)})F'^*(x^{(n)})w||^2 \leq (\alpha^{(n)})^2 \sum_{j=J^{(n)}+1}^{\infty} |(w, v_j^{(n)})|^2 \leq (\alpha^{(n)})^2 ||w||^2. \tag{3.9}$$

In a similar manner, one can estimate

$$||Q^{(n)}v||^2 = \left\| \sum_{j=1}^{J^{(n)}} \frac{1}{\mu_j^{(n)}}(v, v_j^{(n)})u_j^{(n)} \right\| \leq \frac{1}{(\alpha^{(n)})^2} \sum_{j=1}^{J^{(n)}} |(v, v_j^{(n)})|^2 \leq \frac{1}{(\alpha^{(n)})^2} ||v||^2, \tag{3.10}$$

where $\alpha^{(n)}$ is the threshold level. Combining (3.4), (3.5), (3.8), (3.9), and (3.10), one derives

$$||x^{(n+1)} - \hat{x}|| \leq \mathcal{M}||x^{(n)} - \hat{x}||\,||w|| + \alpha^{(n)}||w|| + \frac{\mathcal{M}||x^{(n)} - \hat{x}||^2}{2\alpha^{(n)}}. \tag{3.11}$$

Assume that the regularization sequence $\{\alpha^{(n)}\}$ is defined in such a way that for some constant $r > 0$,

$$\alpha^{(n)} > 0, \qquad \lim_{n\to\infty} \alpha^{(n)} = 0, \qquad \frac{\alpha^{(n)}}{\alpha^{(n+1)}} \leq r \quad \text{for any} \quad n = 0, 1, 2, ... \tag{3.12}$$

We now prove that for sufficiently small $||w||$, the sequence $\{\beta^{(n)}\}$,

$$\beta^{(n)} := \frac{||x^{(n)} - \hat{x}||}{\alpha^{(n)}}, \tag{3.13}$$

is bounded by $\hat{l} \geq 0$ (see (3.15) below), if $\beta^{(0)} \leq \hat{l}$. Indeed, if for some $l \geq 0$, one has $\beta^{(k)} \leq l$, $k = 0, 1, 2, ..., n$, then (3.11) yields

$$\beta^{(n+1)} \leq \mathcal{M}\beta^{(n)}r||w|| + r||w|| + \frac{\mathcal{M}}{2}(\beta^{(n)})^2 r \leq \frac{\mathcal{M}r}{2}l^2 + \mathcal{M}r||w||l + r||w||. \tag{3.14}$$

Take

$$\hat{l} := \frac{2r||w||}{1 - \mathcal{M}r||w||}, \tag{3.15}$$

and suppose that

$$1 \geq \mathcal{M}r||w|| + r\sqrt{2\mathcal{M}||w||}. \tag{3.16}$$

Conditions (3.14) and (3.16), and definition (3.15) imply

$$\beta^{(n+1)} - \hat{l} \leq r||w|| \left\{ \frac{2\mathcal{M}r^2||w||}{(1 - \mathcal{M}r||w||)^2} - 1 \right\} \leq 0 \qquad \Longrightarrow \qquad \beta^{(n+1)} \leq \hat{l}. \tag{3.17}$$

Our observations can be summarized in the following

Theorem 3.1.

Let F be a nonlinear operator between two real Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$, that is, $F : \mathcal{D}_F \subset \mathcal{X} \to \mathcal{Y}$. Assume that F is Fréchet differentiable in $\mathcal{B}(\hat{x})$, the ball centered at $\hat{x}$ with radius $\hat{l}\alpha^{(0)}$, and F' is compact and Lipschitz continuous. Let the regularization sequence $\{\alpha^{(n)}\}$ converge to zero at the rate limited by (3.12), and the solution $\hat{x}$ satisfy source condition (3.7), while $||x^{(0)} - \hat{x}|| \leq \hat{l}\alpha^{(0)}$.

Then, if inequality (3.16) is fulfilled, one has

$$||x^{(n)} - \hat{x}|| \leq \hat{l}\alpha^{(n)}, \qquad n = 0, 1, ..., \tag{3.18}$$

where $\{x^{(n)}\}$ is generated by (3.2) and $\hat{l}$ is introduced in (3.15).

3.3 Stability and Stopping Rule

In case of practical implementation, the problem is inevitably contaminated by various types of noise. First of all, there is noise in the measured data y. Secondly, the operator equation $\mathcal{F}(x) = 0, \quad \mathcal{F}(x) := F(x) - y$, is usually the product of approximate modeling under simplifying assumptions. In addition to that, when it comes to numerical simulations, one may have to deal with a discrete analog of the original operator. As a result, instead of problem (3.1), the iterative scheme is actually applied to some different equation $\mathcal{F}_\delta(x) = 0$, where the operator $\mathcal{F}_\delta$ accumulates discretization, modeling, measurement, and other sources of error, and the singular value decomposition is done for the Jacobian of $\mathcal{F}_\delta$:

$$x^{(n+1)} = P^{(n)}x^{(n)} + (I - P^{(n)})\xi - Q^{(n)}\mathcal{F}_\delta(x^{(n)}), \qquad \xi, x^{(0)} \in \mathcal{D}_F \subset \mathcal{X}. \tag{3.19}$$

Finally, source condition (3.7) is not algorithmically verifiable for the majority of inverse problems, and, in general, one would have

$$F'^*(\hat{x})w + \eta = \xi - \hat{x}, \qquad w \in \mathcal{Y}, \qquad \eta \in \mathcal{X}. \tag{3.20}$$

In this section, we assume that $\mathcal{F}_\delta$ approximates $\mathcal{F}$ to the following level of accuracy

$$||\mathcal{F}_\delta(\hat{x})|| \leq \delta_1 \quad ||F'(\hat{x}) - F'_\delta(\hat{x})|| \leq \delta_2, \quad \text{and} \quad ||\eta|| \leq \delta_3, \tag{3.21}$$

where η satisfies (3.20). If iteratively regularized algorithm (3.19) is terminated according to the *a priori* stopping rule

$$\frac{\delta_1}{(\alpha^{(n)})^2} + \frac{\delta_2 + \delta_3}{\alpha^{(n)}} \leq ||w|| < \frac{\delta_1}{(\alpha^{(\mathcal{K})})^2} + \frac{\delta_2 + \delta_3}{\alpha^{(\mathcal{K})}}, \qquad 0 \leq n < \mathcal{K} = \mathcal{K}(\delta_1, \delta_2, \delta_3), \tag{3.22}$$

then for any $n \leq \mathcal{K}(\delta_1, \delta_2, \delta_3)$,

$$||x^{(n+1)} - \hat{x}|| \leq \mathcal{M}||x^{(n)} - \hat{x}||\,||w|| + \alpha^{(n)}||w|| + \frac{\mathcal{M}||x^{(n)} - \hat{x}||^2}{\alpha^{(n)}} + \frac{\delta_1}{\alpha^{(n)}} + \delta_2 + \delta_3. \tag{3.23}$$

From (3.23) and (3.13), one concludes

$$\beta^{(n+1)} \leq \frac{\mathcal{M}r}{2} l^2 + \mathcal{M}r||w||l + 2r||w||, \qquad n \leq \mathcal{K}(\delta_1, \delta_2, \delta_3). \tag{3.24}$$

If we set

$$l^* := \frac{4r||w||}{1 - \mathcal{M}r||w||}, \tag{3.25}$$

and choose $\xi \in \mathcal{X}$ in such a way that

$$1 \geq \mathcal{M}r||w|| + 2r\sqrt{\mathcal{M}||w||}, \tag{3.26}$$

then

$$||x^{(n)} - \hat{x}|| \leq l^* \alpha^{(n)}, \qquad n = 0, 1, ..., \mathcal{K}(\delta_1, \delta_2, \delta_3). \tag{3.27}$$

Moreover, the following convergence rate is guaranteed under (3.22) and (3.26):

$$||x^{(\mathcal{K})} - \hat{x}|| = O(\delta^{1/2}), \qquad \text{where} \qquad \delta = \max(\delta_1, \delta_1, \delta_3). \tag{3.28}$$

Remark 3.1. Even though condition (3.22) may be hard to verify for the majority of inverse problems, the above analysis clearly illustrates that as opposed to iteratively regularized algorithms using different assumptions on the location of the spectrum of $F'(x)$ and/or various restrictions on the nonlinearity of $F'(x)$, iterative methods based on the source-type conditions are stable with respect to small violations in these conditions. Inequalities (3.22) are similar to the assumptions of Theorem 4.2 in [23].

Table 3.1: Convergence rate for $\alpha^{(n)} = \frac{\alpha^{(0)}}{n^{0.5}}$

$\xi = x^{(0)} = 0.5, \quad \alpha^{(n)} = \frac{\alpha^{(0)}}{n^{0.5}}$				
Noise	$\alpha^{(0)}$	Relative error	Relative discrepancy	n
0%	0.5	4.5943E-002	6.3828E-003	5
2.5%	0.8	6.0168E-002	1.9015E-002	5
5%	1.0	7.9092E-002	4.7690E-002	3
7.5%	1.1	8.3718E-002	6.6720E-002	3
10%	1.3	9.1591E-002	7.5625E-002	5

Remark 3.2. In place of (3.22), one can use *a posteriori* stopping rule developed in [5]. The *a posteriori* rule is more practical, but it does not imply (3.28) without additional assumptions on the structure of the nonlinear operator $\mathcal{F}$ [28].

3.4 Numerical Experiments

In order to examine numerical efficiency of ITN method (3.2), we consider a nonlinear inverse magnetometry problem ([40], [41], [42]) in the form of a 2D Fredholm integral equation of the first kind $\mathcal{F}(x) = 0$, where

$$\mathcal{F}(x) := \Delta\mathcal{J} \int_a^b \int_c^d \left\{ \frac{x(s,u)}{\left[(t-s)^2 + (v-u)^2 + x^2(s,u)\right]^{\frac{3}{2}}} \right. \tag{3.29}$$

$$\left. - \frac{H}{\left[(t-s)^2 + (v-u)^2 + H^2\right]^{\frac{3}{2}}} \right\} du\, ds - y(t,v), \quad t \in [\tilde{a}, \tilde{b}], \quad v \in [\tilde{c}, \tilde{d}].$$

The goal here is to reconstruct the interface $x = x(s,u)$ between two media of different densities from the anomalous magnetic data. The function $y = y(t,v)$ is a measured magnetic field caused by the deviation of the unknown surface S from a horizontal plane $x = -H$, which is assumed to be given along with $\Delta\mathcal{J}$, the averaged jump of the vertical component of the magnetization vector.

To simulate $y = y(t,v)$ for inverse magnetometry problem (4.32), one solves the corresponding forward problem for some model solution $x = \hat{x}(s,u)$ using a very fine

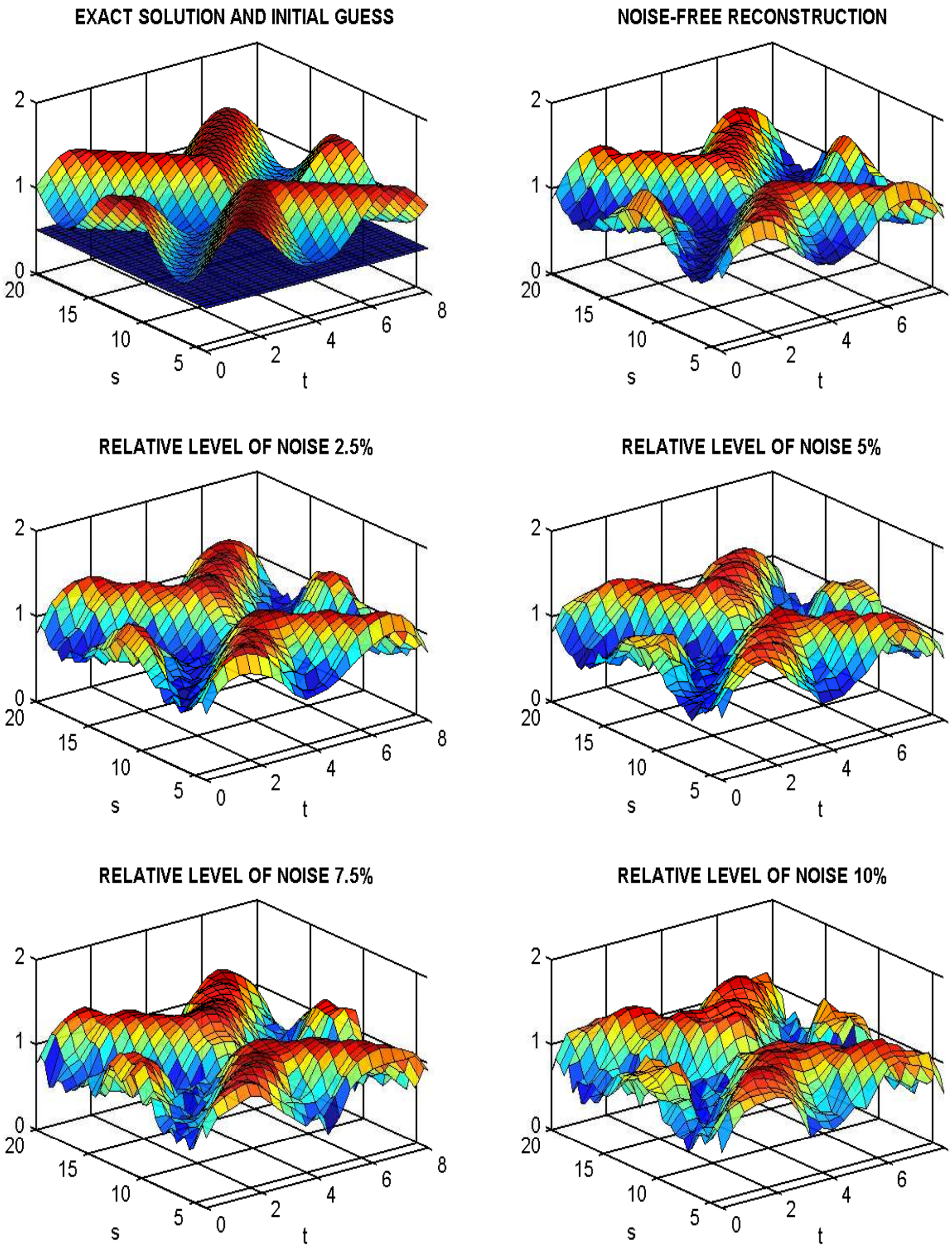

Figure 3.1: Reconstructed Solutions for $\xi = x^{(0)} = 0.5$

Table 3.2: Convergence rate for $\alpha^{(n)} = \frac{\alpha^{(0)}}{n}$

$\xi = x^{(0)} = 0.1, \quad \alpha^{(n)} = \frac{\alpha^{(0)}}{n}$				
Noise	$\alpha^{(0)}$	Relative error	Relative discrepancy	n
0%	0.5	2.1657E-002	3.1968E-004	24
2.5%	1.0	7.7084E-002	1.2272E-002	12
5%	1.5	1.1371E-001	4.2078E-002	6
7.5%	2.0	1.2383E-001	5.1370E-002	9
10%	2.5	1.6765E-001	8.2277E-002	6

Table 3.3: Truncation details for $x^{(0)} = 1$

Relative noise 5%, $\quad x^{(0)} = 1$					
n	1	2	3	4	5
# of trunctd sv	762	659	670	663	654
$\alpha^{(n)}$	9.00E-001	7.57E-001	6.84E-001	6.36E-001	6.02E-001
n	6	7	8	9	10
# of trunctd sv	646	639	634	629	624
$\alpha^{(n)}$	5.75E-001	5.53E-001	5.35E-001	5.20E-001	5.06E-001

grid on $[\tilde{a}, \tilde{b}] \times [\tilde{c}, \tilde{d}]$ and a high accuracy composite integration scheme. This yields exact measurement values of $y = y(t, v)$. Then random noise is added to the solution of forward problem to get the noise-contaminated observables $y = y^{(\delta)}(t, v)$. In order to solve equation (4.32) given $y^{(\delta)}(t, v)$, we discretize independent variables t and v on an $M = I \times K$ grid with fewer grid points, and replace the double integral with a convergent quadrature formula

$$\sum_{j=1}^{J} \sum_{l=1}^{L} \mathcal{K}(t_i, v_k, s_j, u_l, x_{j,l})\mu_{j,l} - y_{i,k}^{(\delta)} = 0, \qquad i = 1, ..., I, \ k = 1, ..., K, \tag{3.30}$$

where

$$\mathcal{K}(t, v, s, u, x) = \Delta\mathcal{J} \left\{ \frac{x}{\left[(t-s)^2 + (v-u)^2 + x^2\right]^{\frac{3}{2}}} - \frac{H}{\left[(t-s)^2 + (v-u)^2 + H^2\right]^{\frac{3}{2}}} \right\}$$

In our experiments, a uniform grid over the rectangular domain $[0.0, 8.0] \times [4.0, 20.0]$

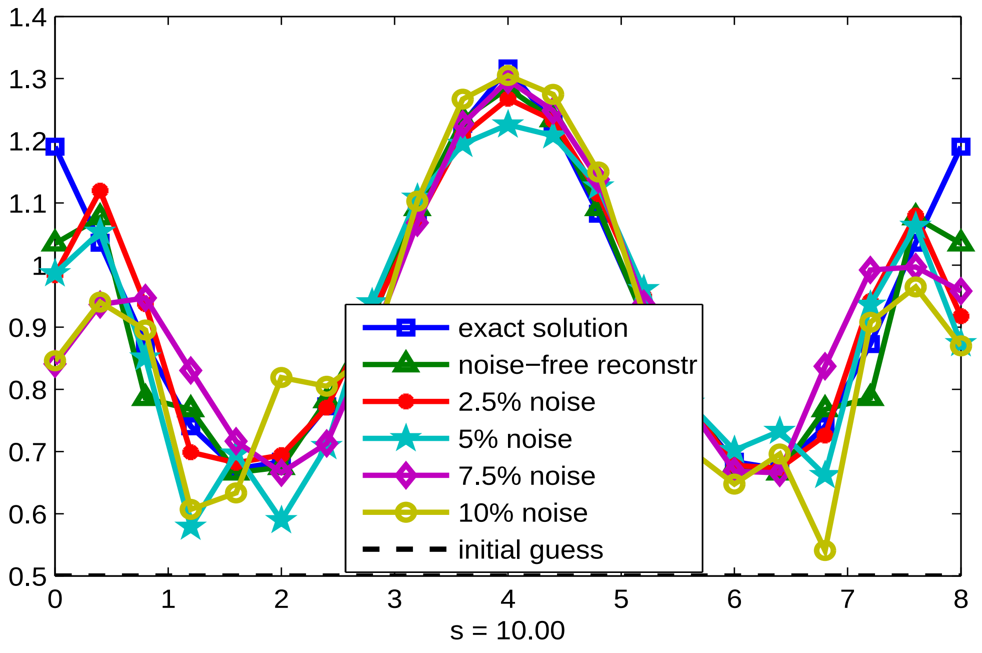

Figure 3.2: Cross-Sectional Comparison 1 for $\xi = x^{(0)} = 0.5$

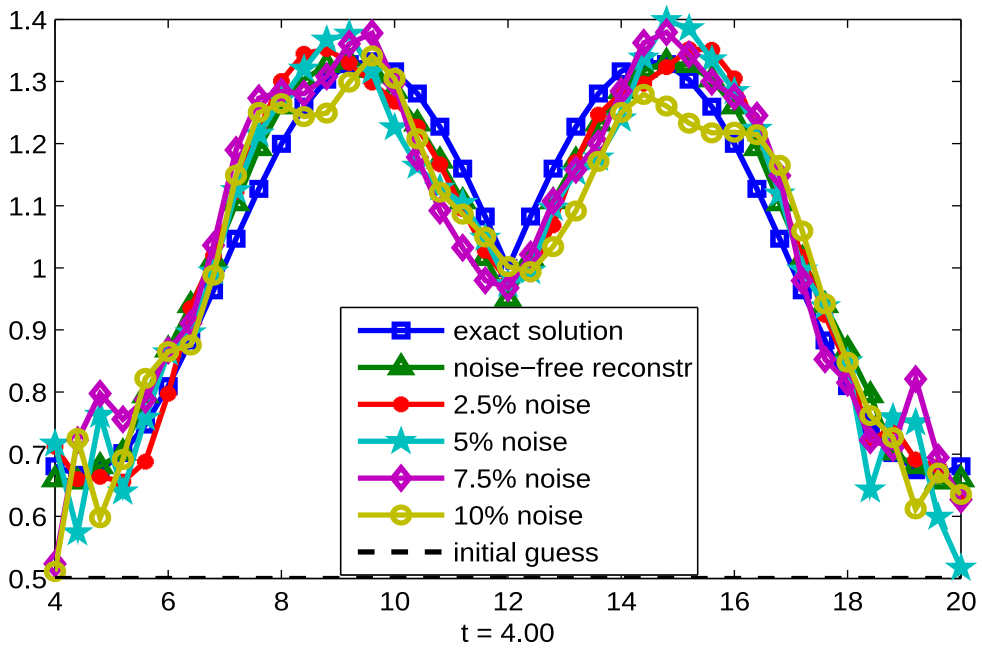

Figure 3.3: Cross-Sectional Comparison 2 for $\xi = x^{(0)} = 0.5$

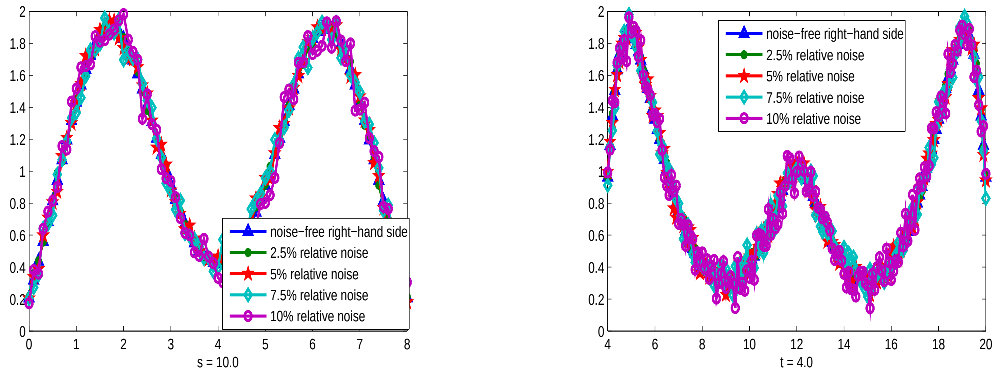

Figure 3.4: Exact and Noisy Data

Table 3.4: Truncation details for $x^{(0)} = 0.1$

Relative noise 5%, $x^{(0)} = 0.1$						
n	1	2	3	4	5	6
trunctd sv	0	0	0	0	24	402
$\alpha^{(n)}$	1.50E+000	7.50E-001	5.0E-001	3.75E-001	3.00E-001	2.50E-001

(km^2) is generated with mesh widths of $h_s = h_u = 0.1$ (km) for data simulation and $h_s = h_u = 0.4$ (km) for solving the inverse problem; $\Delta\mathcal{J} = 1$. The two-dimensional analog of the composite trapezoidal quadrature rule is used to approximate the integral operator. The ground surface height is taken to be $H = 2.0$ (km). The model solution used to simulate the data is of the following form

$$\hat{x}_1(s,u) = -\sin(|10\tilde{s} - 5| - |10\tilde{u} - 5|)/3 + 1; \tag{3.31}$$

where $\tilde{s}$ and $\tilde{u}$ are the re-scaled values of s and u, respectively, i.e.,

$$\tilde{s} = \frac{s-a}{b-a} \in [0,1], \qquad \tilde{u} = \frac{u-c}{d-c} \in [0,1], \qquad s \in [a,b], \qquad u \in [c,d].$$

We use $J = I = 41$ and $L = K = 21$. Thus $M = N = 861$, and the size of the Jacobian is 861×861. The role of the test function ξ in (3.2) is two-fold. On one hand, one makes the process stable by holding $x^{(n)}$ rigid (unchanged) on a subspace that corresponds to singular values truncated at the n-th step. On the other hand, the use of ξ allows to

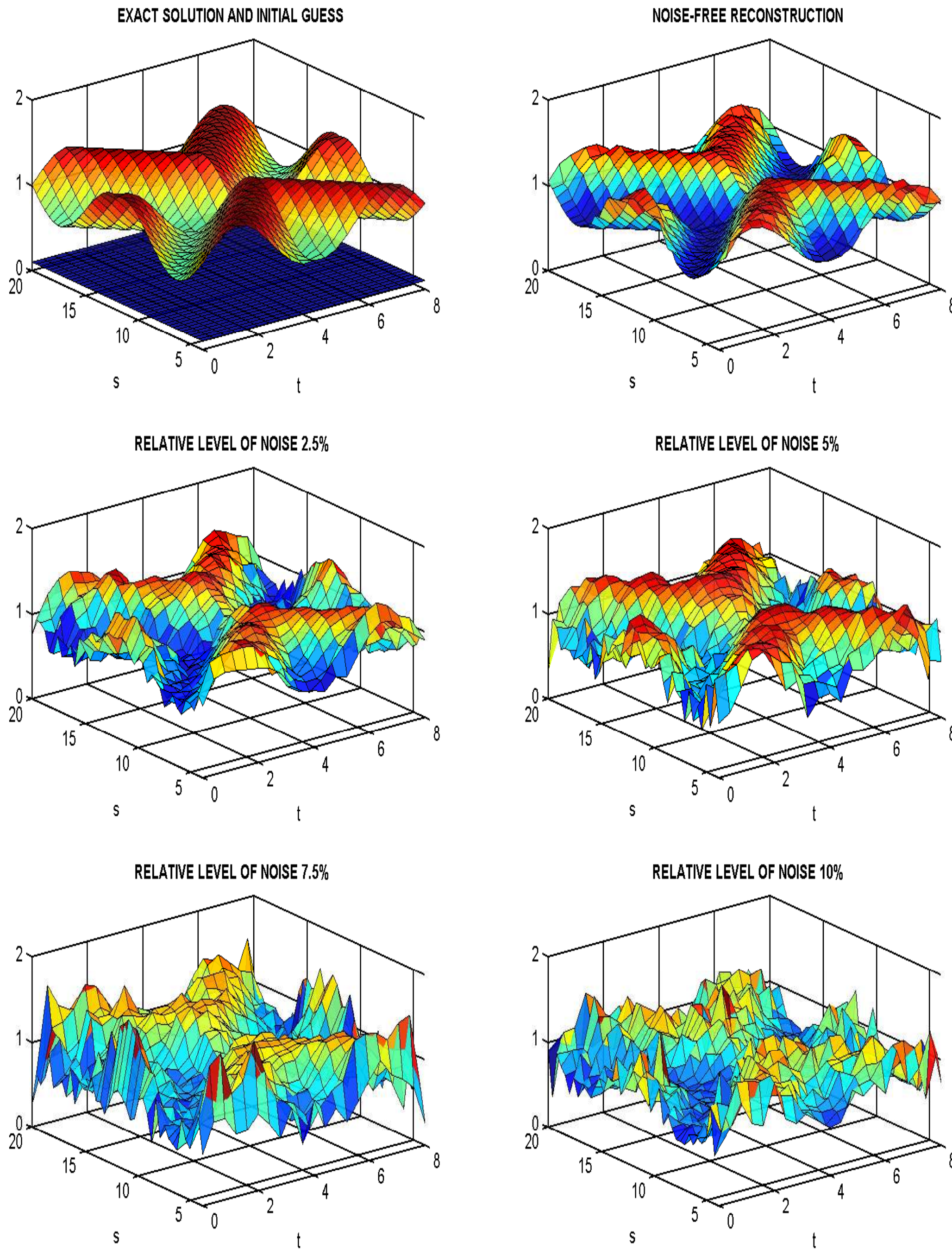

Figure 3.5: Numerical Solutions for $\xi = x^{(0)} = 0.1$

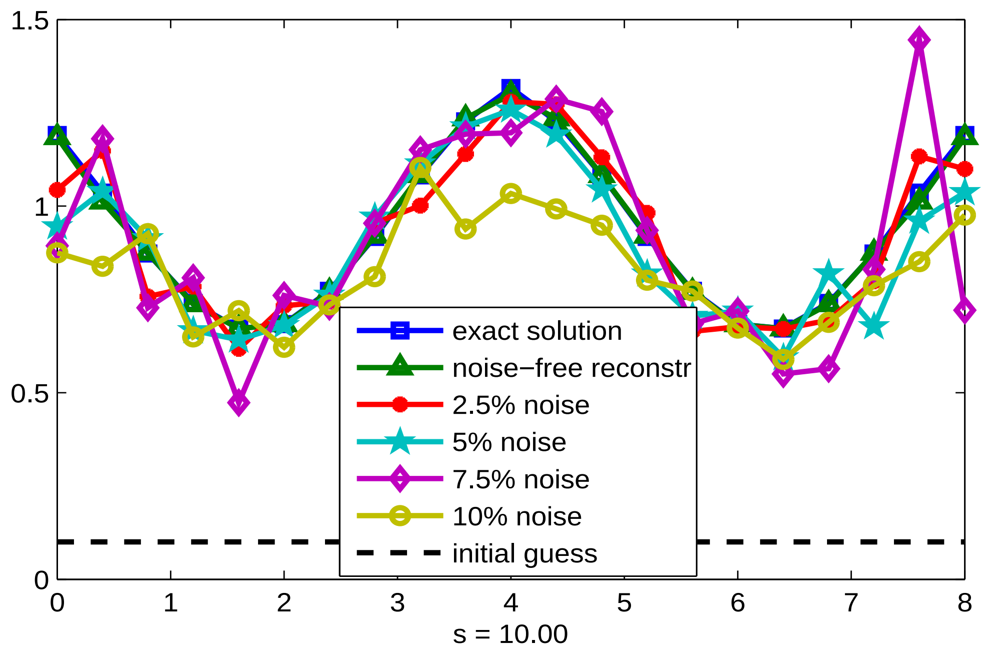

Figure 3.6: Cross-Sectional Comparison 1 for $\xi = x^{(0)} = 0.1$

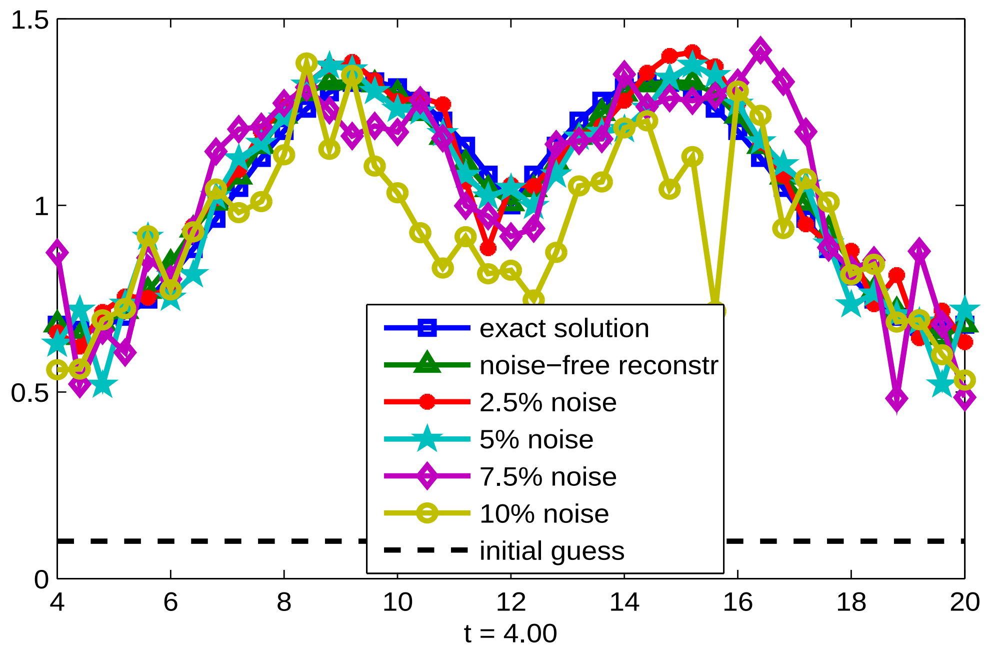

Figure 3.7: Cross-Sectional Comparison 2 for $\xi = x^{(0)} = 0.1$

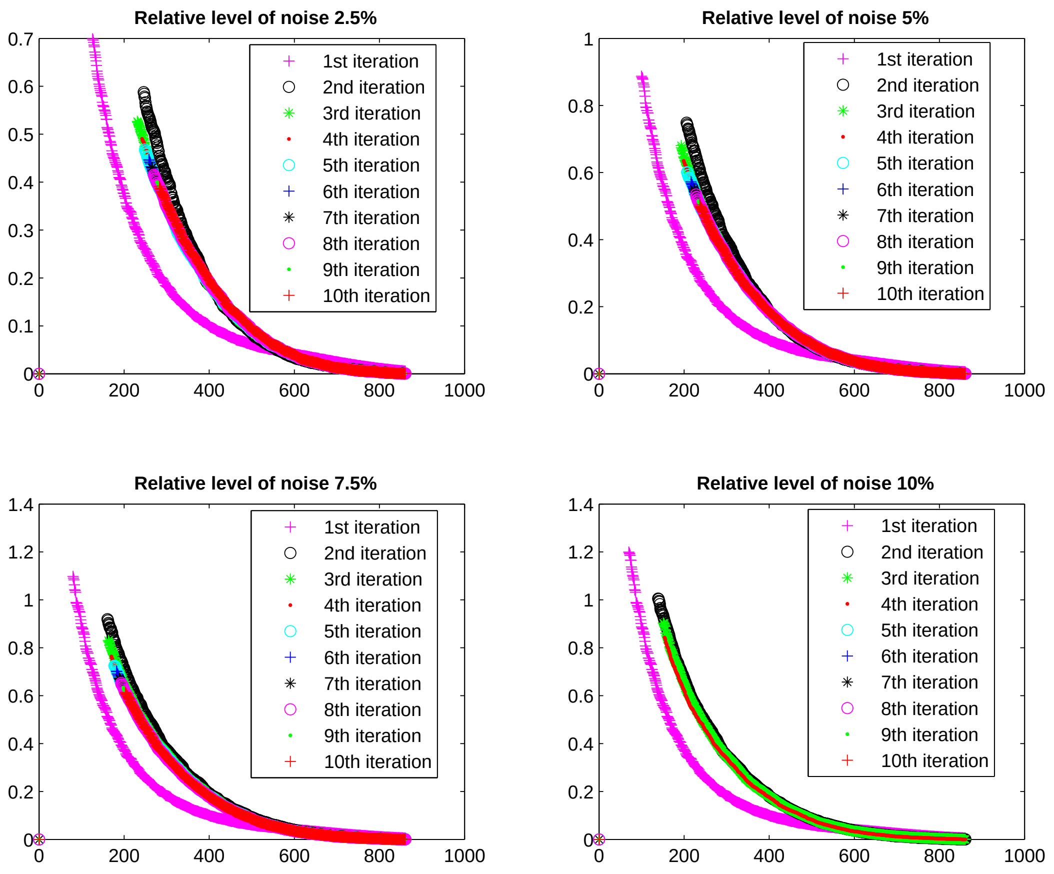

Figure 3.8: Truncated Singular Values for $\xi = x^{(0)} = 1.0$

incorporate an *a priori* information available regarding the true solution. To study how the accuracy of numerical solutions depends on ξ, we conduct our experiments for three different choices of ξ: $\xi = 1.0$, $\xi = 0.5$, $\xi = 0.1$. In each case, ξ also serves as the initial approximation, i.e., $x^{(0)} = \xi$.

For $\xi = 1.0$, the best results are achieved when $\alpha^{(n)}$ is nearly constant: $\alpha^{(n)} = n^{-1/4}$. The value of $\alpha^{(0)}$ is increased from 0.7 to 1.2 as the relative noise level in our data goes up from 2.5% to 10%, see Figure 3.8. Table 3.3 reveals how the number of truncated singular values is changing.

However, as the norm of $\hat{x} - \xi$ is growing, we have to drive $\alpha^{(n)}$ to zero at a faster rate to make sure that stability does not take over accuracy. In general, it is becoming more and more difficult to strike the balance between accuracy and stability as the test function is getting worse. For $\xi = x^{(0)} = 0.5$, the regularization sequence $\alpha^{(n)} = n^{-1/2}$.

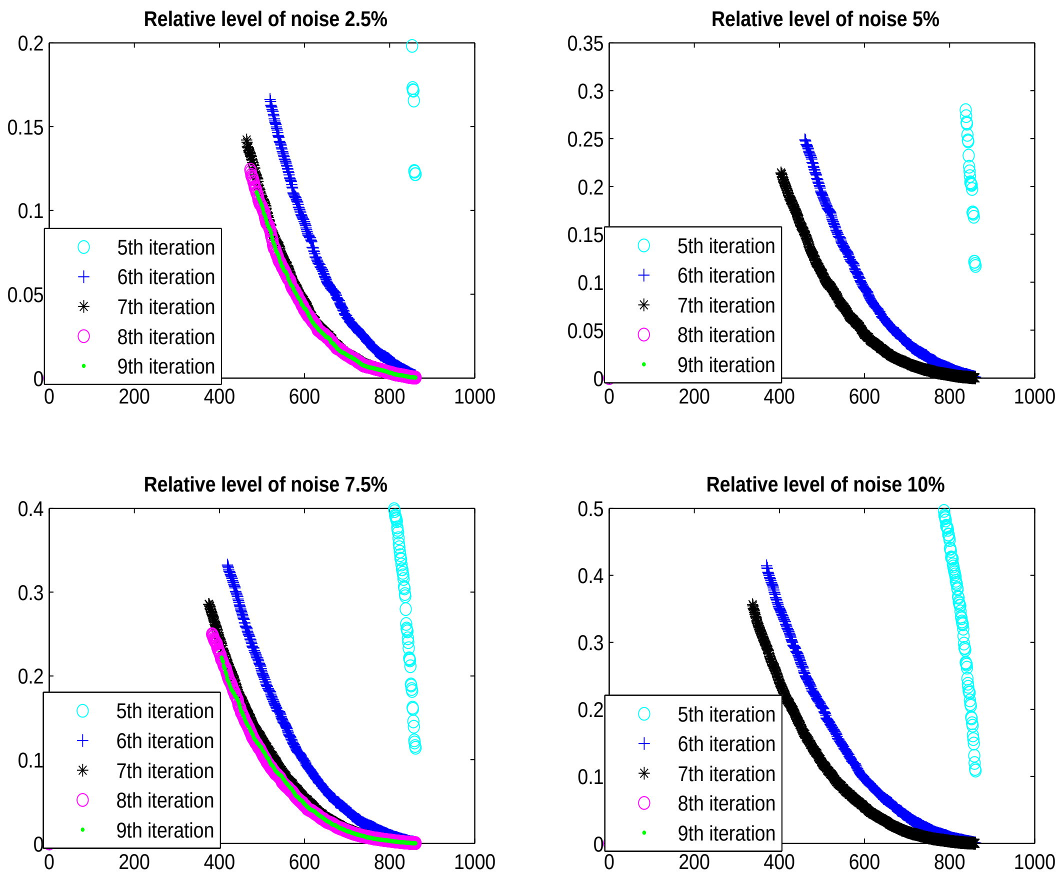

Figure 3.9: Truncated Singular Values for $\xi = x^{(0)} = 0.1$

The accuracy of reconstructions is still very high, see it is shown in Figures 3.1, 3.2, and 3.3, as well as in Table 3.1.

For $\xi = x^{(0)} = 0.1$, one has to take $\alpha^{(n)} = n^{-1}$ and increase $\alpha^{(0)}$. But even that does not prevent the accuracy of the computed solutions from going down as it is evident from Figures 3.5, 3.6, 3.7, and Table 3.2. In all experiments, the iterations are terminated by the discrepancy principle [5].

Figures 3.8 and 3.9 present singular values truncated at every step of the iterative process in order to show what regularization actually does. Figure 3.9 and Table 3.4 illustrate that until $x^{(n)}$ gets close to the solution, singular values do not accumulate at zero, and truncation is not needed.

Chapter 4

A POSTERIORI ERROR ANALYSIS FOR UNSTABLE MODELS

In Chapters 2 and 3, we have investigated iteratively regularized Newton-type algorithms for solving linear and nonlinear operator equations.

In this chapter, we consider the possibility of *a posteriori* error estimates, which is an important aspect in the analysis of numerical solutions to inverse problems. Unlike the well-posed case, discrepancy alone cannot guarantee that the approximate solution of an ill-posed problem is accurate. Given an auxiliary finite-dimensional problem $\Phi(w) = 0$, $\Phi : \mathcal{D}_\Phi \subset \mathcal{E}_N \to \mathcal{E}_M$, that approximates the original infinite model $\mathcal{F}(x) = 0$, $\mathcal{F} : \mathcal{D}_F \subset \mathcal{X} \to \mathcal{Y}$, with a certain level of accuracy, we try to estimate the distance between z, an approximate solution to $\Phi(w) = 0$, and $\hat{x}$, the exact solution to $\mathcal{F}(x) = 0$. The problem $\Phi(w) = 0$ is assumed to accumulate different sources of error (discretization, measurements, etc), and the computed solution z is assumed to satisfy the equation $\Phi(w) = 0$ within a nonzero tolerance β. Both theoretical and numerical study of *a posteriori* error analysis is conducted. Presentation in this chapter follows [44].

4.1 Introduction

Our basic model takes the form of a nonlinear operator equation

$$\mathcal{F}(x) = 0, \quad \mathcal{F} : \mathcal{D}_F \subset \mathcal{X} \to \mathcal{Y}, \tag{4.1}$$

on a pair of two normed spaces $\mathcal{X}$ and $\mathcal{Y}$ over the field $\mathbb{R}$ or $\mathbb{C}$. Suppose that problem (4.1) is known to be solvable, maybe non-uniquely, and $\hat{x} \in \mathcal{D}_F$ is a solution of interest. We do not require F in (4.1) to be differentiable or even continuous, nor do we assume here that the operator is stable with respect to noise in the input data.

In the numerical analysis of equation (4.1), some finite dimensional problem

$$\Phi(w) = 0, \quad \Phi : \mathcal{D}_\Phi \subset \mathcal{E}_N \to \mathcal{E}_M, \tag{4.2}$$

is used, which approximates equation (4.1) in a certain sense. The operator Φ acts between Euclidian spaces $\mathcal{E}_N$ and $\mathcal{E}_M$ and $N \neq M$, in general. Normally, Φ would accumulate different sources of error: due to measurements, approximate modeling, discretization, etc.

Assume that an element $z \in \mathcal{D}_\Phi \subset \mathcal{E}_N$, such that

$$||\Phi(z)|| = \beta, \tag{4.3}$$

with β being rather small, has been computed by means of some numerical procedure. Here $||\cdot||$ denotes the norm of a space, to which the element under the norm belongs. Let p be a connecting operator between $\mathcal{X}$ and $\mathcal{E}_N$ [45] and $\hat{z} := p\hat{x}$. The goal of this chapter is to answer the following question:

How can one estimate $||z - \hat{z}||$ based on condition (4.3)?

In section 4.2, our main theoretical result, *a posteriori* error estimate (4.17) is established. In section 4.3, we consider possible application of estimate (4.17) to a 2D nonlinear integral operator equation of the first kind with noisy data. In section 4.4, the numerical algorithm for solving a nonlinear magnetometry problem is outlined, and the simulation results are presented. We provide comparison of estimated and actual error bounds in section 4.5 followed by the discussion on advantages and limitations of *a posteriori* accuracy assessment by formula (4.17).

4.2 Theoretical Background

Suppose that the operator Φ of problem (4.2) is differentiable and its Jacobian Φ' is Lipschitz continuous in a neighborhood of z:

$$||\Phi'(u) - \Phi'(v)|| \leq \mathcal{M}||u - v|| \quad \text{for any} \quad u, v \in \mathcal{B}(z), \tag{4.4}$$

and some $\mathcal{M} \geq 0$. Under assumption (4.4), the following identity holds:

$$\Phi'(z)(\hat{z} - z) = \Phi(\hat{z}) - \Phi(z) + G(z, \hat{z}), \quad ||G(z, \hat{z})|| \leq \frac{\mathcal{M}}{2}||z - \hat{z}||^2. \tag{4.5}$$

Consider the operator $\Phi'(z)$. This is a matrix that can be directly computed. A careful analysis of $\Phi'(z)$ results in the construction of various operators, which make it possible to transform equality (4.5) in such a way that the desired estimate is obtained under certain *a priori* assumptions. Specifically, let $\Phi'(z)$ have singular values

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_{\bar{K}-1} \geq \lambda_{\bar{K}} \geq 0, \quad \bar{K} := \min(M, N), \tag{4.6}$$

and (λ_n, u_n, v_n), $n = 1, 2, ..., \bar{K}$, be the singular system. For any $v \in \mathcal{E}_M$, the solution u of the equation $\Phi'(z)u = v$, if exists, is given by Picard's theorem

$$u = \sum_{n=1}^{\bar{K}} \frac{1}{\lambda_n}(v, v_n)u_n \tag{4.7}$$

as long as $v \in \Phi(z)(\mathcal{D}_\Phi)$. This result suggests the way to construct a pseudoinverse [1] to the Jacobian $\Phi'(z)$

$$Q_r := \sum_{n=1}^{\bar{K}} \frac{\nu(\alpha, \lambda_n)}{\lambda_n}(\,\cdot\,, v_n)u_n, \quad \text{where} \quad \frac{\nu(\alpha, 0)}{0} := 0 \quad \text{and} \quad \nu(\alpha, \lambda) := \begin{cases} 1, & \lambda \geq \alpha, \\ 0, & \lambda < \alpha, \end{cases} \tag{4.8}$$

Let r be the number of singular values exceeding the threshold $\alpha > 0$. Then

$$Q_r\Phi'(z) = \sum_{n=1}^{\bar{K}} \frac{\nu(\alpha, \lambda_n)}{\lambda_n}(\Phi'(z)\,\cdot\,, v_n)u_n = \sum_{n=1}^{r} \frac{1}{\lambda_n}(\,\cdot\,, \Phi'^*(z)v_n)u_n = \sum_{n=1}^{r}(\,\cdot\,, u_n)u_n = P_r$$

is the orthogonal projector into the subspace spanned by the first r eigenvectors of the operator $\Phi'^*(z)\Phi'(z)$. In particular for $r = \bar{K}$,

$$Q_r\Phi'(z)u = \sum_{n=1}^{\bar{K}}(u, u_n)u_n = u \quad \text{and} \quad Q_r = (\Phi'^*(z)\Phi'(z))^{-1}\Phi'^*(z).$$

Application of the operator Q_r to both sides of identity (4.5) yields

$$P_r(\hat{z}-z) = Q_r(\Phi(\hat{z}) - \Phi(z)) + Q_r G(z,\hat{z}), \quad ||G(z,\hat{z})|| \leq \frac{\mathcal{M}}{2}||z-\hat{z}||^2 \tag{4.9}$$

From the above, one concludes

$$||P_r(\hat{z}-z)|| \leq ||Q_r|| \left\{ ||\Phi(\hat{z}) - \Phi(z)|| + \frac{\mathcal{M}}{2}\Big(||(I-P_r)(z-\hat{z})||^2 + ||P_r(z-\hat{z})||^2 \Big) \right\}$$

Suppose

$$||Q_r|| = \kappa, \quad ||(I-P_r)(z-\hat{z})|| = \sigma \quad \text{and} \quad ||\Phi(\hat{z})|| = ||\Phi(p\hat{x})|| = \varepsilon. \tag{4.10}$$

Clearly, for $r = \bar{K}$ the value of σ is zero. Inequality (4.3) implies the estimate

$$||P_r(\hat{z}-z)|| \leq \kappa \left\{ \varepsilon + \beta + \frac{\mathcal{M}}{2}\Big(\sigma^2 + ||P_r(z-\hat{z})||^2\Big) \right\}, \tag{4.11}$$

which immediately gives us an upper bound for $||P_r(z-\hat{z})||$ and, by virtue of this, for the norm of $(z-\hat{z})$ itself:

$$||\hat{z}-z|| = ||(I-P_r+P_r)(z-\hat{z})|| = \Big\{\sigma^2 + ||P_r(z-\hat{z})||^2\Big\}^{1/2}. \tag{4.12}$$

Indeed, suppose that combined noise in our problem satisfies the assumption

$$\kappa^2 \mathcal{M}\left[2(\varepsilon+\beta) + \mathcal{M}\sigma^2\right] < 1. \tag{4.13}$$

Then two cases are possible

$$||P_r(z-\hat{z})|| \leq \frac{1 - \Big\{1 - \kappa^2\mathcal{M}\left[2(\varepsilon+\beta)+\mathcal{M}\sigma^2\right]\Big\}^{1/2}}{\kappa\mathcal{M}}, \tag{4.14}$$

or

$$||P_r(z-\hat{z})|| \geq \frac{1 + \Big\{1 - \kappa^2\mathcal{M}\left[2(\varepsilon+\beta)+\mathcal{M}\sigma^2\right]\Big\}^{1/2}}{\kappa\mathcal{M}}. \tag{4.15}$$

To rule out the second option, introduce the following (very reasonable) condition:

$$||P_r(z-\hat{z})|| \le \frac{1}{\kappa \mathcal{M}}. \tag{4.16}$$

Notice that under assumption (4.13), estimate (4.14) is a guaranteed improvement over (4.16). We can summarize our observations in the following

Theorem 4.1. Under assumptions (4.3), (4.4), (4.13), and (4.16), the error in the computed solution z to equation (4.1) satisfies the following *a posteriori* estimate

$$||z-\hat{z}|| \le \sqrt{\sigma^2 + \left(\frac{1-\left\{1-\kappa^2\mathcal{M}\left[2(\varepsilon+\beta)+\mathcal{M}\sigma^2\right]\right\}^{1/2}}{\kappa\mathcal{M}}\right)^2}. \tag{4.17}$$

Here $\hat{z} := p\hat{x}$, $\hat{x} \in \mathcal{D}_F \subset \mathcal{X}$ is the exact solution to (4.1), $p : \mathcal{X} \to \mathcal{E}_N$ is a connecting operator, and the constants κ, σ, and ε are defined in (4.10).

Remark 4.2. Notice that if $\lambda_{\bar{K}} > 0$ in (4.6), and we choose not to cut off, then $\kappa = \frac{1}{\lambda_{\bar{K}}}$. In general, $\kappa \le \frac{1}{\alpha}$ [1]:

$$||Q_r v||^2 = \sum_{n=1}^{\bar{K}} \frac{\nu^2(\alpha,\lambda_n)}{\lambda_n^2}|(v,v_n)|^2 \le \frac{1}{\alpha^2}\sum_{n=1}^{\bar{K}}|(v,v_n)|^2 = \frac{1}{\alpha^2}||v||^2, \tag{4.18}$$

where α is the threshold level in (4.8).

Remark 4.3. In a linear case, the operator $G(z,\hat{z})$ is zero. From (4.11), it follows that

$$||P_r(\hat{z}-z)|| \le \kappa\left(\varepsilon+\beta\right), \tag{4.19}$$

and in place of (4.17), one obtains a very simple estimate

$$||z-\hat{z}|| \le \sqrt{\sigma^2+\kappa^2\left(\varepsilon+\beta\right)^2}. \tag{4.20}$$

The reader may consult ([46], [47], [48]), and references therein for a detailed study of *a posteriori* error estimation in case of linear ill-posed problems under various *a priori* assumptions on the exact solution. An alternative approach aimed at obtaining order optimal *a posteriori* estimates for solutions to linear operator equations solved by variational regularization method is presented in [49].

Remark 4.4. A special case of (4.17), when $r = \bar{K}$ ($\sigma = 0$) has been previously considered by A. B. Bakushinsky in [50]. An *a priori* version of estimate (4.11) for the class of methods with a spectral cut off has been derived in [51].

4.3 Further Discussion

Generally, the operator Φ in (4.2) would accumulate at least two sources of error: due to noise in the measured data and due to discretization. Suppose $\mathcal{F}$ in (4.1) takes the form $\mathcal{F}(x) := A(x) - y$, where A is a nonlinear operator from $\mathcal{D}_F \subset \mathcal{X}$ into $\mathcal{Y}$, and $\mathcal{F}_\delta(x) := A(x) - y^{(\delta)}$ with $y^{(\delta)}$ being the noise contaminated right-hand side, $||y - y^{(\delta)}|| \leq \delta$. Then it is natural to assume that Φ is a discrete analog of $\mathcal{F}_\delta$, i.e.,

$$\Phi(w) := A_{N,M}(w) - y_M^{(\delta)}, \quad A_{N,M} : \mathcal{D}_\Phi \subset \mathcal{E}_N \to \mathcal{E}_M, \tag{4.21}$$

$y_M^{(\delta)} := qy^{(\delta)}$, and q is a connecting operator between $\mathcal{Y}$ and $\mathcal{E}_M$. For example, it can be a discrete approximation of a nonlinear integral operator by a mechanical quadrature method [45]. In that case,

$$||\Phi(\hat{z})|| = ||A_{N,M}(\hat{z}) - y_M^{(\delta)}|| \leq ||A_{N,M}(p\hat{x}) - qA(\hat{x})|| + ||q(y - y^{(\delta)})||. \tag{4.22}$$

The first error term in the right-hand side of (4.22) is solely due to discretization and can be estimated for each particular quadrature formula. The second error is coming from noise contaminated measurements and should be known *a priori.*

To illustrate application of (4.22), consider a 2D nonlinear Fredholm integral equation of the first kind $\mathcal{F}(x) := A(x) - y = 0$,

$$A(x) := \int_a^b \int_c^d \mathcal{K}(t, v, s, u, x(s, u))\, du\, ds, \quad t \in [\tilde{a}, \tilde{b}],\ v \in [\tilde{c}, \tilde{d}], \tag{4.23}$$

$A : \mathcal{X} = W_2^1([a, b] \times [c, d]) \to \mathcal{Y} = L_2([\tilde{a}, \tilde{b}] \times [\tilde{c}, \tilde{d}])$. Let it be known *a priori* from physical considerations that $|\hat{x}| \le \gamma$ for some $\gamma > 0$ and the kernel $\mathcal{K}(t, v, s, u, x)$ is continuous in

$$S := \{a \le s \le b,\ c \le u \le d,\ \tilde{a} \le t \le \tilde{b},\ \tilde{c} \le v \le \tilde{d},\ |x| \le \gamma\}. \tag{4.24}$$

We partition the intervals $[a, b]$, $[c, d]$, $[\tilde{a}, \tilde{b}]$, and $[\tilde{c}, \tilde{d}]$ with mesh points $\{s_j\}_1^J$, $\{u_l\}_1^L$, $\{t_i\}_0^I$, and $\{v_k\}_0^K$, respectively. Take a convergent quadrature formula

$$\int_a^b \int_c^d f(s, u)\, du\, ds = \sum_{j=1}^{J} \sum_{l=1}^{L} \mu_{j,l} f(s_j, u_l) + R(f), \quad \mu_{j,l} \ge 0, \tag{4.25}$$

and use (4.25) to define the approximating operator $\Phi(w) := A_{N,M}(w) - y_M^{(\delta)}$ with $A_{N,M} : \mathcal{D}_\Phi \subset \mathcal{E}_N \to \mathcal{E}_M$,

$$[A_{N,M}(w)]_{i,k} := \sum_{j=1}^{J} \sum_{l=1}^{L} \mu_{j,l} \mathcal{K}(\bar{t}_i, \bar{v}_k, s_j, u_l, w_{j,l}), \quad i = 1, 2, ..., I,\ k = 1, 2, ..., K, \tag{4.26}$$

$N = J \times L$, $M = I \times K$, while $w = [w_{1,1}\, w_{1,2} \ldots w_{1,L}\, w_{2,1}\, w_{2,2} \ldots w_{2,L} \ldots w_{J,1}\, w_{J,2} \ldots w_{J,L}]^T$, $\bar{t}_i = \frac{t_{i-1}+t_i}{2}$, and $\bar{v}_k = \frac{v_{k-1}+v_k}{2}$. Then one has

$$\sum_{j=1}^{J} \sum_{l=1}^{L} \mu_{j,l} \mathcal{K}(t, v, s_j, u_l, \hat{x}(s_j, u_l)) - \int_a^b \int_c^d \mathcal{K}(t, v, s, u, \hat{x}(s, u))\, du\, ds = R(\psi_{t,v}),$$

where the set of functions

$$\Psi := \{\psi_{t,v}(s, u) : \psi_{t,v}(s, u) = \mathcal{K}(t, v, s, u, \hat{x}(s, u)),\ \tilde{a} \le t \le \tilde{b},\ \ \tilde{c} \le v \le \tilde{d}\}$$

is relatively compact in $C([a, b] \times [c, d])$ [45] and, since (4.25) is convergent,

$$\sup_{\psi_{t,v} \in \Psi} |R(\psi_{t,v})| \longrightarrow 0 \quad \text{as} \quad J, L \longrightarrow \infty.$$

The actual value of the supremum will depend on a particular quadrature formula as well as a specific expression for $\mathcal{K}(t, v, s, u, x)$. Let the connecting operators $p : \mathcal{X} \to \mathcal{E}_N$

and $q : \mathcal{Y} \to \mathcal{E}_M$ be introduced as follows

$$p : x(s,u) \to [x(s_1,u_1)\, x(s_1,u_2) \ldots x(s_1,u_L) \ldots x(s_J,u_1)\, x(s_J,u_2) \ldots x(s_J,u_L)]^T,$$

$$q : g(t,v) \to \left[\frac{1}{\triangle\, t_1 \triangle\, v_1} \int_{t_0}^{t_1} \int_{v_0}^{v_1} g(t,v)\, dv\, dt \ldots \frac{1}{\triangle\, t_1 \triangle\, v_K} \int_{t_0}^{t_1} \int_{v_{K-1}}^{v_K} g(t,v)\, dv\, dt \right.$$

$$\left. \ldots \frac{1}{\triangle\, t_I \triangle\, v_1} \int_{t_{I-1}}^{t_I} \int_{v_0}^{v_1} g(t,v)\, dv\, dt \ldots \frac{1}{\triangle\, t_I \triangle\, v_K} \int_{t_{I-1}}^{t_I} \int_{v_{K-1}}^{v_K} g(t,v)\, dv\, dt \right]^T .$$

In order to estimate $||\Phi(\hat{z})|| = ||\Phi(p\hat{x})||$, we first consider

$$[A_{N,M}(p\hat{x}) - qA(\hat{x})]_{i,k} = \frac{1}{\triangle\, t_i \triangle\, v_k} \int_{t_{i-1}}^{t_i} \int_{v_{k-1}}^{v_k} \left(\sum_{j=1}^{J} \sum_{l=1}^{L} \mu_{j,l} \mathcal{K}(\bar{t}_i, \bar{v}_k, s_j, u_l, \hat{x}(s_j,u_l)) \right.$$

$$\left. - \int_a^b \int_c^d \mathcal{K}(\bar{t}_i, \bar{v}_k, s, u, \hat{x}(s,u))\, du\, ds \right) dv\, dt + \frac{1}{\triangle\, t_i \triangle\, v_k} \int_a^b \int_c^d \int_{t_{i-1}}^{t_i} \int_{v_{k-1}}^{v_k}$$

$$\left\{ \mathcal{K}(\bar{t}_i, \bar{v}_k, s, u, \hat{x}(s,u)) - \mathcal{K}(t, v, s, u, \hat{x}(s,u)) \right\} dv\, dt\, du\, ds. \tag{4.27}$$

Assuming that $\mathcal{K}(t,v,s,u,x)$ is Lipschitz continuous in variables t and v, and the corresponding Lipschitz constants, which depend on other variables as on parameters, are bounded with respect to these parameters in set S (4.24) by values $\mathcal{L}_t$ and $\mathcal{L}_v$ respectively, one derives

$$\left| [A_{N,M}(p\hat{x}) - qA(\hat{x})]_{i,k} \right| \le |R(\psi_{\bar{t}_i,\bar{v}_k})| + \frac{\mathcal{C}_{\mathcal{K}}(b-a)(d-c)}{4} [\triangle\, t_i + \triangle\, v_k], \tag{4.28}$$

where $\mathcal{C}_{\mathcal{K}} = \max\{\mathcal{L}_t, \mathcal{L}_v\}$. Let the norm in $\mathcal{E}_M$, the discrete analog of the continuous space $\mathcal{Y} = L_2([\tilde{a}, \tilde{b}] \times [\tilde{c}, \tilde{d}])$, be defined by means of quadrature coefficients $\{\zeta_{i,k}\}$ with $\zeta_{i,k} \ge 0$,

$$||r|| := \left(\sum_{i=1}^{I} \sum_{k=1}^{K} \zeta_{i,k} |r_{i,k}|^2 \right)^{1/2}, \quad M = I \times K. \tag{4.29}$$

If the kernel $\mathcal{K}(t,v,s,u,\hat{x})$ is sufficiently smooth in S, the grid is uniform in both directions, i.e., $\triangle\, t_i = \triangle\, v_k := h_{\mathcal{Y}}$, $i = 1,2,...,I$, $k = 1,2,...,K$, $\triangle\, s_j = \triangle\, u_l := h_{\mathcal{X}}$, $j = 1,2,...,J$, $l = 1,2,...,L$, and $\{\mu_{j,l}\}$ correspond to the composite trapezoidal rule, one obtains

$$||A_{N,M}(p\hat{x}) - qA(\hat{x})||^2 \leq \sum_{i=1}^{I}\sum_{k=1}^{K} \zeta_{i,k} \left[\frac{(b-a)(d-c)}{2}\left(\frac{\mathcal{A}_{\mathcal{K}} h_{\mathcal{X}}^2}{6} + \mathcal{C}_{\mathcal{K}} h_{\mathcal{Y}}\right)\right]^2.$$

Here $\mathcal{A}_{\mathcal{K}} = \sup_S |\mathcal{K}''_{ss}(t,v,s,u,\hat{x})| + \sup_S |\mathcal{K}''_{uu}(t,v,s,u,\hat{x})|$. Provided the last quadrature formula is exact for $g(t,v) := 1$, we conclude

$$||A_{N,M}(p\hat{x}) - qA(\hat{x})|| \leq \frac{(b-a)(d-c)\sqrt{(\tilde{b}-\tilde{a})(\tilde{d}-\tilde{c})}}{2}\left(\frac{\mathcal{A}_{\mathcal{K}} h_{\mathcal{X}}^2}{6} + \mathcal{C}_{\mathcal{K}} h_{\mathcal{Y}}\right). \tag{4.30}$$

We now evaluate the second norm in (4.22)

$$\|q(y - y^{(\delta)})\|^2 = \sum_{i=1}^{I}\sum_{k=1}^{K} \zeta_{i,k}\left(\frac{1}{\triangle\, t_i \triangle\, v_k}\int_{t_{i-1}}^{t_i}\int_{v_{k-1}}^{v_k} (y(t,v) - y^{(\delta)}(t,v))\, dv\, dt\right)^2$$

By Hölder's inequality, one derives

$$\|q(y - y^{(\delta)})\|^2 \leq \sum_{i=1}^{I}\sum_{k=1}^{K} \frac{\zeta_{i,k}}{(\triangle\, t_i \triangle\, v_k)^2}\int_{t_{i-1}}^{t_i}\int_{v_{k-1}}^{v_k} (y(t,v) - y^{(\delta)}(t,v))^2\, dv\, dt \int_{t_{i-1}}^{t_i}\int_{v_{k-1}}^{v_k} dv\, dt.$$

So whenever $\zeta_{i,k} = \triangle\, t_i \triangle\, v_k$, the last expression is equal to $\|y - y^{(\delta)}\|$, that is

$$\|q(y - y^{(\delta)})\| \leq \|y - y^{(\delta)}\|,$$

and

$$\varepsilon := ||\Phi(\hat{z})|| \leq \frac{(b-a)(d-c)\sqrt{(\tilde{b}-\tilde{a})(\tilde{d}-\tilde{c})}}{2}\left(\frac{\mathcal{A}_{\mathcal{K}} h_{\mathcal{X}}^2}{6} + \mathcal{C}_{\mathcal{K}} h_{\mathcal{Y}}\right) + \delta. \tag{4.31}$$

One can see that the main part of estimate (4.31) (the one that cannot be changed) is δ, noise in the measurements. The discretization error can technically be reduced to any tolerance level as long as machine memory and accuracy allow that, and as long as the corresponding derivatives of $\mathcal{K} = \mathcal{K}(t,v,s,u,\hat{x}(s,u))$ are finite.

4.4 Numerical Simulations

To illustrate the above *a posteriori* estimates, we consider the 2D nonlinear Fredholm integral equation in the following form ([40], [41], [42]):

$$A(x) := \triangle\,\mathcal{J} \int_a^b \int_c^d \left\{ \frac{x(s,u)}{\left[(t-s)^2 + (v-u)^2 + x^2(s,u)\right]^{\frac{3}{2}}} \right. \tag{4.32}$$

$$\left. - \frac{H}{\left[(t-s)^2 + (v-u)^2 + H^2\right]^{\frac{3}{2}}}\, du\, ds \right\} = y(t,v),\ t \in [\tilde{a}, \tilde{b}],\ v \in [\tilde{c}, \tilde{d}].$$

Here one is to reconstruct the interface $x = x(s,u)$ between two media from the anomalous magnetic data. The right-hand side $y = y(t,v)$ is a measured magnetic field caused by the deviation of the unknown surface S from a horizontal plane $x = -H$, and $\triangle\,\mathcal{J}$ is a given averaged jump of the vertical component of the magnetization vector. The operator A in (4.32) is a special case of (4.23) with

$$\mathcal{K}(t,v,s,u,x) = \triangle\,\mathcal{J} \left\{ \frac{x}{\left[(t-s)^2 + (v-u)^2 + x^2\right]^{\frac{3}{2}}} - \frac{H}{\left[(t-s)^2 + (v-u)^2 + H^2\right]^{\frac{3}{2}}} \right\}$$

To simulate data for inverse magnetometry problem (4.32), we solve the corresponding forward problem for some model solution $x = \hat{x}(s,u)$ using a very fine grid on $[\tilde{a}, \tilde{b}] \times [\tilde{c}, \tilde{d}]$ and a high accuracy numerical integration scheme. This yields exact measurement values of $y = y(t,v)$. Then random noise is added to the solution of forward problem to get the noise-contaminated observables $y = y^{(\delta)}(t,v)$.

In order to solve equation (4.32) given $y^{(\delta)}(t,v)$, we discretize independent variables t and v on an $M = I \times K$ grid, and replace the double integral with quadrature formula (4.25):

$$[\Phi(x)]_{i,k} := \sum_{j=1}^{J} \sum_{l=1}^{L} \mathcal{K}(t_i, v_k, s_j, u_l, x_{j,l})\mu_{j,l} - y_{i,k}^{(\delta)} = 0, \quad i = 1, ..., I,\ k = 1, ..., K. \tag{4.33}$$

This results in a system of M nonlinear equations with $N = J \times L$ unknowns. Provided $M = N$, nonlinear system (4.33) can be solved by some regularized version of the classical

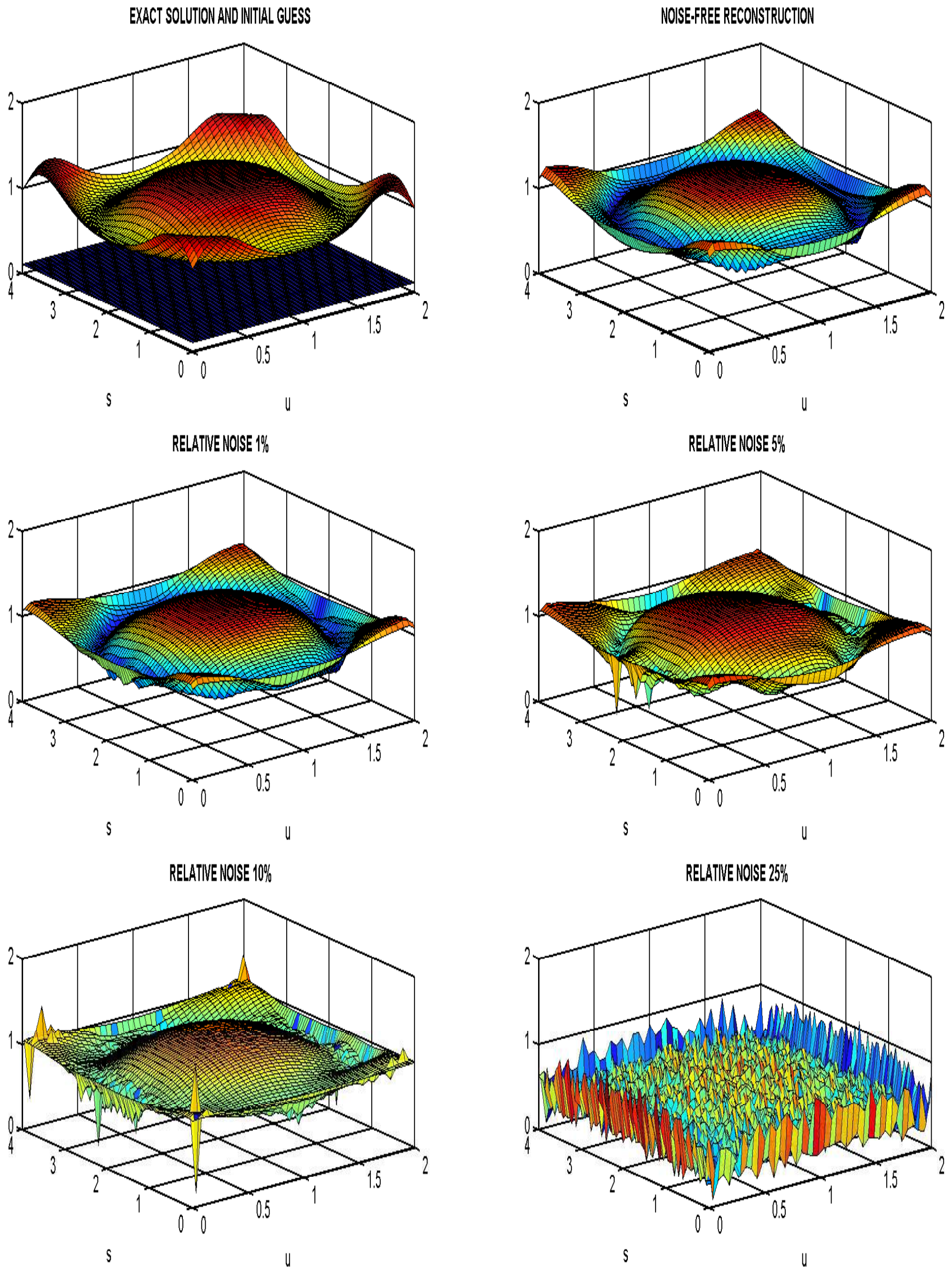

Figure 4.1: Numerical Solutions for the First Data Set

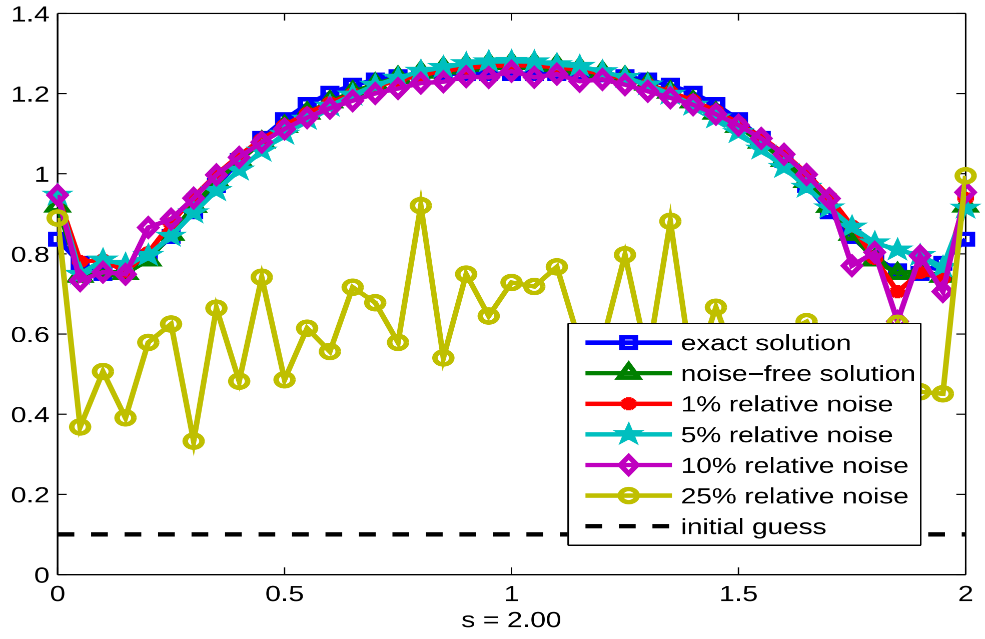

Figure 4.2: Cross-Sectional Comparison for the First Data Set

Newton-Kantorovich method [52], [53]

$$\Phi'(x^{(\nu)})\tau^{(\nu)} = -\Phi(x^{(\nu)}), \quad \tau^{(\nu)} = x^{(\nu+1)} - x^{(\nu)}, \quad x^{(0)} \in \mathcal{D}_\Phi, \tag{4.34}$$

which can be written as

$$\sum_{j=1}^{J}\sum_{l=1}^{L}\mathcal{K}'_x(t_i, v_k, s_j, u_l, x_{j,l}^{(\nu)})\tau_{j,l}^{(\nu)}\mu_{j,l} = -\left[\sum_{j=1}^{J}\sum_{l=1}^{L}\mathcal{K}(t_i, v_k, s_j, u_l, x_{j,l}^{(\nu)})\mu_{j,l} - y_{i,k}^{(\delta)}\right],$$

$$\tau_{j,l}^{(\nu)} = x_{j,l}^{(\nu+1)} - x_{j,l}^{(\nu)}, \quad i = 1, 2, ..., I, \quad k = 1, 2, ..., K, \quad M = I \times K. \tag{4.35}$$

Denote

$$g_{i,k}^{(\nu)} := \mathcal{K}(t_i, v_k, s_j, u_l, x_{j,l}^{(\nu)})\mu_{j,l} - y_{i,k}^{(\delta)}.$$

To evaluate $\tau_{j,l}^{(\nu)}$ from (4.35), we compress the 4D array $\mathcal{K}'_x(t_i, v_k, s_j, u_l, x_{j,l}^{(\nu)})\mu_{j,l}$ to an M by N matrix $\mathbb{K}_{m,n}^{(\nu)}$, and 2D arrays $\tau_{j,l}^{(\nu)}$ and $g_{i,k}^{(\nu)}$ to N by 1 and M by 1 column-vectors

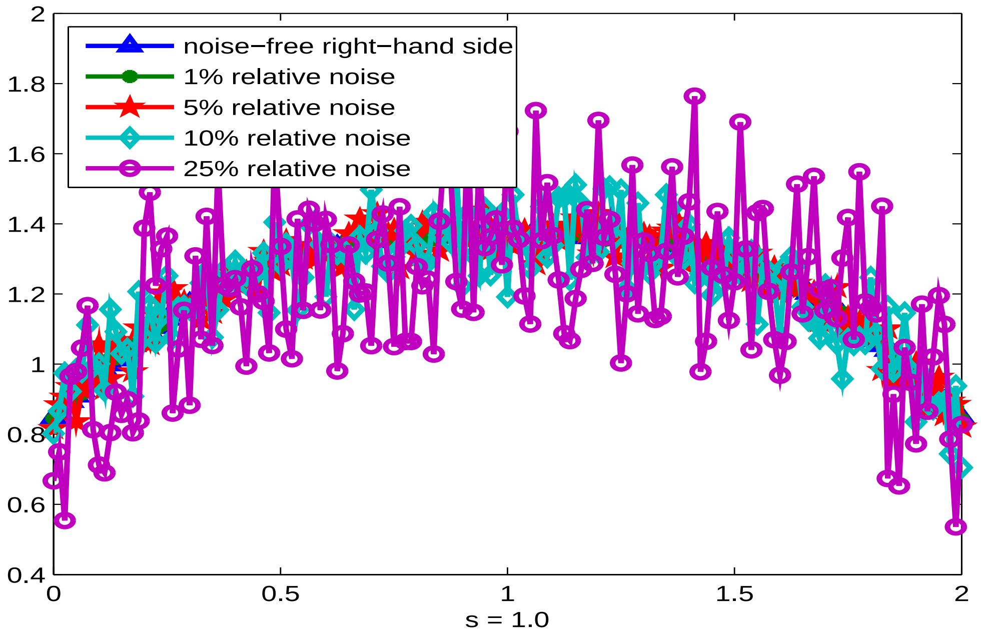

Figure 4.3: Exact and Noisy Data for the First Experiment

$\Gamma_n^{(\nu)}$ and $G_m^{(\nu)}$, respectively. Then (4.35) is equivalent to the linear system

$$\sum_{n=1}^{N} \mathbb{K}_{m,n}^{(\nu)} \Gamma_n^{(\nu)} = -G_m^{(\nu)}, \quad \Gamma_n^{(\nu)} = X_n^{(\nu+1)} - X_n^{(\nu)}. \tag{4.36}$$

Upon convergence of a Newton - type method, we uncompress the solution to get a 2D approximation of $x = \hat{x}(s, u)$.

The 2D to 1D compression is organized as follows: if the 2D array is J by L, then the first row of the matrix becomes the first L elements of the vector, the second row becomes the second L elements, etc. Consequently, to uncompress a vector with $N = J \times L$ coordinates, the first L elements of the vector form the first row of the matrix, the second L elements form the second row, etc.

Using the above rule one can also compress a 4D array to a 2D matrix. Indeed, fix $i = 1$ and $k = 1$. Then $\mathcal{K}'_x(t_1, v_1, s_j, u_l, x_{j,l}^{(\nu)})\mu_{j,l}$ is 2D, and we may apply the 2D to 1D compression rule to convert $\mathcal{K}'_x(t_1, v_1, s_j, u_l, x_{j,l}^{(\nu)})\mu_{j,l}$ to the first row of $\mathbb{K}_{m,n}^{(\nu)}$. Next, fix $i = 1$ and $k = 2$ to convert $\mathcal{K}'_x(t_1, v_2, s_j, u_l, x_{j,l}^{(\nu)})\mu_{j,l}$ to the second row of $\mathbb{K}_{m,n}^{(\nu)}$, etc.

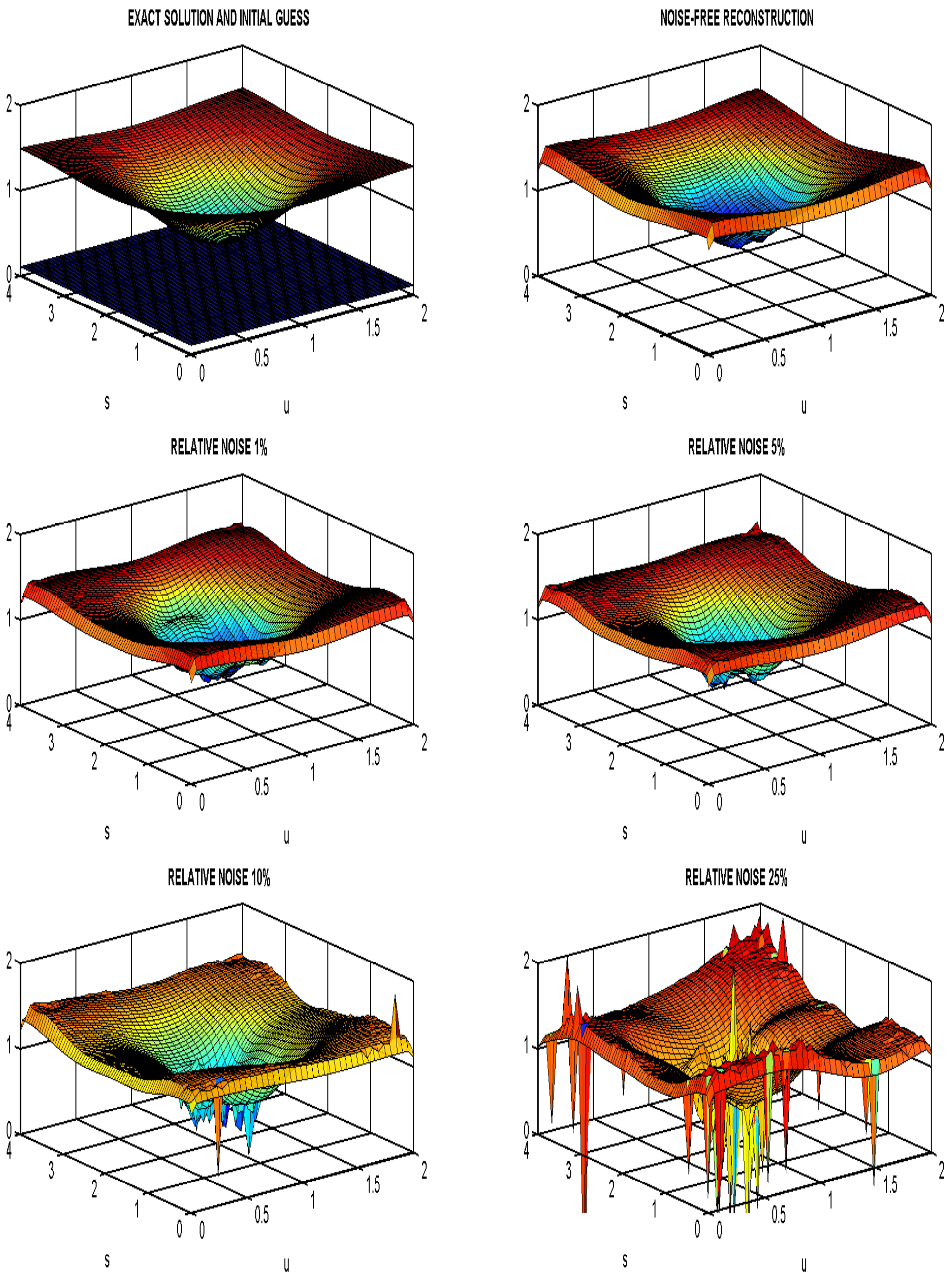

Figure 4.4: Numerical Solution for the Second Data Set

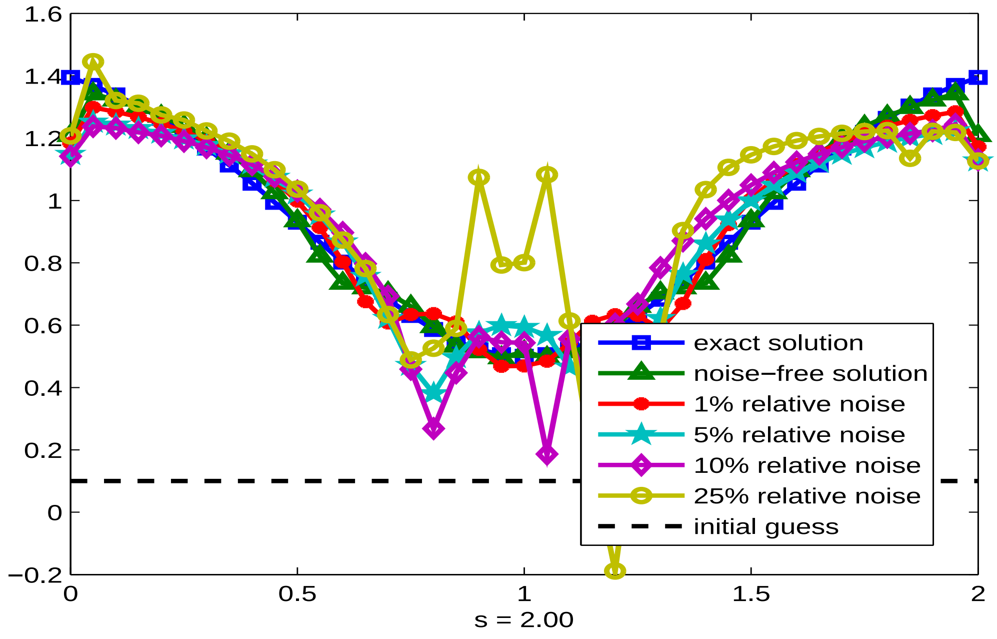

Figure 4.5: Cross-Sectional Comparison for the Second Data Set

Finally, we fix $i = I$ and $k = K$ to convert $\mathcal{K}'_x(t_I, v_K, s_j, u_l, x^{(\nu)}_{j,l})\mu_{j,l}$ to the M^{th} row of $\mathbb{K}^{(\nu)}_{m,n}$. For our numerical simulations we use the following regularized adaptation of method (4.35)

$$\big(\mathbb{K}^{*(\nu)}\mathbb{K}^{(\nu)} + \alpha^{(\nu)}\mathbb{I}\big)\Gamma^{(\nu)} = -\big\{\mathbb{K}^{*(\nu)}G^{(\nu)} + \alpha^{(\nu)}(X^{(\nu)} - X^{(0)})\big\}, \quad \Gamma^{(\nu)} = X^{(\nu+1)} - X^{(\nu)},$$

known as iteratively regularized Gauss-Newton (IRGN) algorithm, which was proposed by A. Bakushinnsky in [54], and further studied in [9], [10], [23], [11], [13], [28], [16] and many other papers. Here the stabilizing sequence $\{\alpha^{(\nu)}\}$ is such that

$$\alpha^{(\nu)} \geq 0, \quad \lim_{\nu\to 0} \alpha^{(\nu)} = 0, \quad 0 \leq \frac{\alpha^{(\nu)}}{\alpha^{(\nu+1)}} \leq \text{const}. \tag{4.37}$$

As opposed to (4.35), IRGN can still be executed in more general case when $M \neq N$, and the compression procedure is the same.

In our experiments, a uniform grid over the rectangular domain $[0.0, 4.0] \times [0.0, 2.0]$ (km^2) is generated with mesh widths of $h_s = h_u = 0.0125$ (km) for data simulation and

Table 4.1: Experiments for different levels of noise

$a = 0.0, \quad b = 4.0, \quad c = 0.0, \quad d = 2.0, \quad L = 41, \quad J = 81$						
Rel Noise	Iter	Rel Error 1	Discrepancy 1	Iter	Rel Error 2	Discrepancy 2
0	13	$3.92 \cdot 10^{-2}$	$4.38 \cdot 10^{-4}$	11	$3.64 \cdot 10^{-2}$	$4.53 \cdot 10^{-4}$
0.01	8	$5.30 \cdot 10^{-2}$	$8.39 \cdot 10^{-3}$	8	$5.31 \cdot 10^{-2}$	$7.06 \cdot 10^{-3}$
0.05	6	$6.71 \cdot 10^{-2}$	$3.94 \cdot 10^{-2}$	7	$8.12 \cdot 10^{-2}$	$3.45 \cdot 10^{-2}$
0.1	6	$8.86 \cdot 10^{-2}$	$1.32 \cdot 10^{-1}$	5	$8.22 \cdot 10^{-2}$	$8.61 \cdot 10^{-2}$
0.25	6	$1.92 \cdot 10^{-1}$	$3.11 \cdot 10^{-1}$	2	$2.27 \cdot 10^{-1}$	$8.93 \cdot 10^{-1}$

$h_s = h_u = 0.05\,(\mathrm{km})$ for solving the inverse problem; $\triangle\ \mathcal{J} = 1$. The two-dimensional analog of the composite trapezoidal quadrature rule is used to approximate the integral operator. The ground surface height is taken to be $H = 2.0\,(\mathrm{km})$. The constant horizontal plane $x^{(0)}(s,u) = 0.1\,(\mathrm{km})$ serves as the initial guess for all the simulations. The model solutions used to simulate the data are of the following form

$$\hat{x}_1(s,u) = \frac{1}{4}\cos((4\tilde{s}-2)^2 + (4\tilde{u}-2)^2) + 1, \tag{4.38}$$

and

$$\hat{x}_2(s,u) = -\exp[-(3\tilde{s}-1.5)^2 - (3\tilde{u}-1.5)^2] + 1.5 \tag{4.39}$$

where $\tilde{s}$ and $\tilde{u}$ are the re-scaled values of s and u, respectively, i.e.,

$$\tilde{s} = \frac{s-a}{b-a} \in [0,1], \quad \tilde{u} = \frac{u-c}{d-c} \in [0,1], \quad s \in [a,b], \quad u \in [c,d].$$

The regularization sequence $\{\alpha^{(\nu)}\}$ is chosen to be $\alpha^{(\nu)} = \alpha^{(0)}\exp(-\nu)$ and $\alpha^{(\nu)} = \alpha^{(0)}\nu^{-1}$ with $\alpha^{(0)} = 0.5$ or 1. The sequence $\alpha^{(\nu)} = \exp(-\nu)$ gives the most aggressive convergence rate for both model solutions and noise levels shown in Table 4.1. The iterations are terminated by the discrepancy principle.

In our calculations, $J = I = 81$ and $L = K = 41$, and therefore $M = N = 3321$. The condition number of 3321 by 3321 Jacobian at the last step of the iterative process is of order 10^{19} - 10^{23}.

Table 4.2: Parameter values and error estimates

$a = 0.0,\quad b = 4.0,\quad c = 0.0,\quad d = 2.0,\quad L = 41,\quad J = 81$								
Subregion	$\mathcal{M}$	$\mathcal{A}_{\mathcal{K}}$	$\mathcal{C}_{\mathcal{K}}$	δ	β	κ	Act Er	Est Err
$[1.75, 2] \times [0.75, 1]$	4.74	216.0	0.248	0.00062	0.000649	6.81	0.00349	0.0151
$[2, 2.25] \times [1, 1.25]$	4.74	216.0	0.152	0.00144	0.000780	6.81	0.00353	0.0281
$[1.75, 2] \times [1, 1.25]$	4.74	216.0	0.200	0.00207	0.001068	5.73	0.00360	0.0359

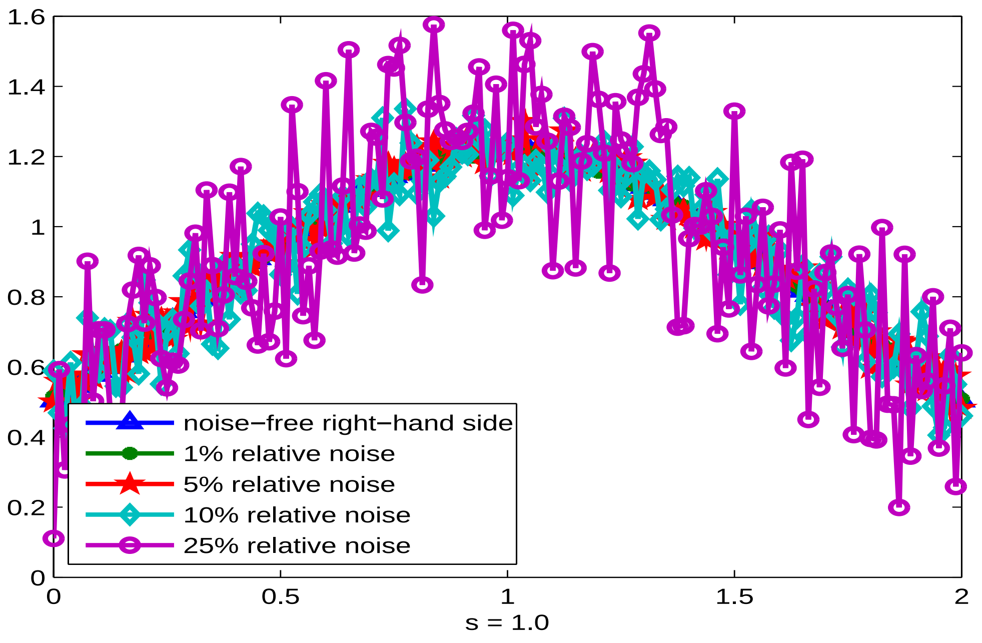

Figure 4.6: Exact and Noisy Data for the Second Experiment

The reader can see the error estimates for numerical solutions obtained from $x = \hat{x}_1(s, u)$ in column 9 compared to the accurate error in column 8 of Table 4.2. The reconstructed surfaces are illustrated in Figures 4.1 and 4.4. Figures 4.2 and 4.5 give cross-sectional comparison between numerical solutions computed for different levels of noise. The corresponding exact and noisy right-hand sides are presented in Figures 4.3 and 4.6.

4.5 Analysis of the Results

Our numerical study revealed both advantages and limitations of *a posteriori* error estimate (4.17). First, for estimate (4.17) to be applicable, the step size for a numerical integration formula needs to be rather small, especially if the reconstruction is done over a large region and $\mathcal{A}_{\mathcal{K}}$ is large. Moreover, since $x = \hat{x}(s, u)$ is unknown, only an upper bound for $\mathcal{A}_{\mathcal{K}}$ is available, which may considerably exceed the actual value. However, from numerical standpoint, decreasing h_s and h_u too much is not beneficial, because it results in a loss of stability that is not compensated by the gain in accuracy. Besides, CPU computer time goes up as the size of Jacobian increases, and eventually computer runs out of memory.

Secondly, the value of σ in estimate (4.17) is unknown for most practically important cases. If one integrates with a relatively large step size ($h_s = h_u = 0.4$ over the region $[2.4, 20.0] \times [0.0, 8.0]$, for example), then the Jacobian is more or less well-conditioned and $||(\Phi'^*(z)\Phi'(z))^{-1}\Phi'^*(z)|| \sim 10^3$. Hence, one can use $r = \bar{K}$ ($\kappa = ||(\Phi'^*(z)\Phi'(z))^{-1}\Phi'^*(z)||$) for the estimate, and then $\sigma = 0$. But inequality (4.13) is not satisfied for the above values of h_s, h_u, a, b, c, and d ($\varepsilon \gg 1$), and (4.17) is not applicable. If one takes a smaller region and uses $h_s = h_u = 0.05$ with the same size of the Jacobian, then $||(\Phi'^*(z)\Phi'(z))^{-1}\Phi'^*(z)|| \sim 10^{20}$ (or even 10^{23}). For this reason, one can no longer use $\kappa = ||(\Phi'^*(z)\Phi'(z))^{-1}\Phi'^*(z)||$. After spectral cut off, the value of κ goes down, but then $\sigma \neq 0$, in general.

Taking into consideration all of the above, we choose three small subregions of the domain $[0.0, 4.0] \times [0.0, 2.0]$, and estimate the accuracy of the computed solution over these subregions only. Specifically, we get the first error bound by restricting the domain of our computed solution from $[0.0, 4.0] \times [0.0, 2.0]$ to $[1.75, 2] \times [0.75, 1]$ and evaluating $\sqrt{\sigma^2 + \left(\left(1 - \left\{1 - \kappa^2\mathcal{M}\left[2(\varepsilon + \beta) + \mathcal{M}\sigma^2\right]\right\}^{1/2}\right)/[\kappa\mathcal{M}]\right)^2}$ for this new domain. The same idea is used for two other estimates. The value of ε is found by (4.31), while β is computed numerically. We use $\alpha = \alpha^{(\tilde{\nu})}$, the value of the regularization parameter at the last iteration of IRGN method, for spectral cut off required to evaluate $||Q_r||$. For the first two subregions, $\tilde{\nu} = 7$ and $\alpha^{(7)} = 1/7$; for the last subregion $\tilde{\nu} = 6$ and

$\alpha^{(6)} = 1/6$. When $\alpha = 1/7$, $\lambda_r = 0.146842...$, and $\kappa = ||Q_r|| = 1/\lambda_r = 6.81....$ If $\alpha = 1/6$, $\lambda_r = 0.174520...$, and $\kappa = ||Q_r|| = 1/\lambda_r = 5.73....$ The value of σ is assumed to be at most 10^{-2} for all three subregions.

To find $\mathcal{M}$, we first notice that in our case $G(z, \hat{z})$ defined in (4.5) is as follows

$$[G(z,\hat{z})]_{i,k} = \frac{1}{2}\sum_{j=1}^{J}\sum_{l=1}^{L}\mathcal{K}''_{xx}(t_i, v_k, s_j, u_l, \tilde{z}_{j,l})(z-\hat{z})^2_{j,l}\,\mu_{j,l}$$

$$= \frac{\triangle\,\mathcal{J}}{2}\sum_{j=1}^{J}\sum_{l=1}^{L}\frac{3\tilde{z}_{j,l}\left(2\tilde{z}^2_{j,l} - 3\left[(t_i-s_j)^2+(v_k-u_l)^2\right]\right)}{\left[(t_i-s_j)^2+(v_k-u_l)^2+\tilde{z}^2_{j,l}\right]^{\frac{7}{2}}}(z-\hat{z})^2_{j,l}\,\mu_{j,l}, \qquad (4.40)$$

$i = 1, ..., I$, $k = 1, ..., K$. From (4.40), one concludes

$$\left|[G(z,\hat{z})]_{i,k}\right| \le \frac{\mathcal{U}}{2}\sum_{j=1}^{J}\sum_{l=1}^{L}(z-\hat{z})^2_{j,l}\,\mu_{j,l} \le \frac{\mathcal{U}}{2}||z-\hat{z}||^2.$$

Here $\mathcal{U} = \sup_S |\mathcal{K}''_{xx}(t, v, s, u, x(s, u))|$. Since by our assumption the quadrature formula in (4.29) is exact for $g(t, v) = 1$, one derives

$$||G(z,\hat{z})|| \le \frac{\mathcal{U}}{2}||z-\hat{z}||^2\left(\sum_{i=1}^{I}\sum_{k=1}^{K}\zeta_{i,k}\right)^{1/2} = \frac{\mathcal{U}}{2}\sqrt{(\tilde{b}-\tilde{a})(\tilde{d}-\tilde{c})}||z-\hat{z}||^2, \qquad (4.41)$$

which means $\mathcal{M} = \mathcal{U}\sqrt{(\tilde{b}-\tilde{a})(\tilde{d}-\tilde{c})}$. To get the value of $\mathcal{M}$ from (4.41), we use an upper bound for $\mathcal{U}$ rather than the actual supremum. For the partial derivatives required to compute $\mathcal{A}_{\mathcal{K}}$ and $\mathcal{C}_{\mathcal{K}}$ we also use upper bounds instead of suprema. The results are summarized in Table 4.2.

So at least for small subregions, the estimated errors can be computed by formula (4.17). However, in our opinion the significance of estimate (4.17) is not in its ability to give the precise numerical value of the error bound, but in revealing all sources of this error and their respective weights. Thus it provides important guidance on how to balance accuracy and stability in the construction of numerical algorithms for nonlinear irregular operator equations.

REFERENCES

[1] A. Kirsch, *An introduction to the mathematical theory of inverse problems*, ser. Applied Mathematical Sciences. Springer-Verlag, New York, 1996, vol. 120. [Online]. Available: http://dx.doi.org/10.1007/978-1-4612-5338-9

[2] A. B. Bakushinsky, A. Smirnova, and H. Liu, "A nonstandard approximation of pseudoinverse and a new stopping criterion for iterative regularization," *J. Inverse Ill-Posed Probl.*, 2014. [Online]. Available: http://dx.doi.org/10.1515/jiip-2013-0086

[3] J. M. Ortega and W. C. Rheinboldt, "Local and global convergence of generalized linear iterations," in *Studies in Numerical Analysis, 2: Numerical Solutions of Nonlinear Problems (Symposia, SIAM, Philadelphia, Pa., 1968).* Soc. Indust. Appl. Math., Philadelphia, Pa., 1970, pp. 122–143.

[4] A. B. Bakushinsky, "Iterative methods without saturation for solving degenerate nonlinear operator equations," *Dokl. Akad. Nauk*, vol. 344, no. 1, pp. 7–8, 1995.

[5] A. Bakushinsky and A. Smirnova, "On application of generalized discrepancy principle to iterative methods for nonlinear ill-posed problems," *Numer. Funct. Anal. Optim.*, vol. 26, no. 1, pp. 35–48, 2005. [Online]. Available: http://dx.doi.org/10.1081/NFA-200051631

[6] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*, ser. Mathematics and its Applications. Kluwer Academic Publishers Group, Dordrecht, 1996, vol. 375. [Online]. Available: http://dx.doi.org/10.1007/978-94-009-1740-8

[7] M. Hanke, "Regularizing properties of a truncated Newton-CG algorithm for nonlinear inverse problems," *Numer. Funct. Anal. Optim.*, vol. 18, no. 9-10, pp. 971–993, 1997. [Online]. Available: http://dx.doi.org/10.1080/01630569708816804

[8] A. B. Bakushinsky, "A general method of constructing regularizing algorithms for a linear incorrect equation in Hilbert space," *Ž. Vyčisl. Mat. i Mat. Fiz.*, vol. 7, pp. 672–677, 1967.

[9] B. Blaschke, A. Neubauer, and O. Scherzer, “On convergence rates for the iteratively regularized Gauss-Newton method,” *IMA J. Numer. Anal.*, vol. 17, no. 3, pp. 421–436, 1997. [Online]. Available: http://dx.doi.org/10.1093/imanum/17.3.421

[10] T. Hohage, “Logarithmic convergence rates of the iteratively regularized Gauss-Newton method for an inverse potential and an inverse scattering problem,” *Inverse Problems*, vol. 13, no. 5, pp. 1279–1299, 1997. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/13/5/012

[11] A. Smirnova, R. A. Renaut, and T. Khan, “Convergence and application of a modified iteratively regularized Gauss-Newton algorithm,” *Inverse Problems*, vol. 23, no. 4, pp. 1547–1563, 2007. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/23/4/011

[12] S. Langer and T. Hohage, “Convergence analysis of an inexact iteratively regularized Gauss-Newton method under general source conditions,” *J. Inverse Ill-Posed Probl.*, vol. 15, no. 3, pp. 311–327, 2007. [Online]. Available: http://dx.doi.org/10.1515/jiip.2007.017

[13] Q. Jin, “A convergence analysis of the iteratively regularized Gauss-Newton method under the Lipschitz condition,” *Inverse Problems*, vol. 24, no. 4, pp. 045 002, 16, 2008. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/24/4/045002

[14] D. Langemann and M. Tasche, “Phase reconstruction by a multilevel iteratively regularized Gauss-Newton method,” *Inverse Problems*, vol. 24, no. 3, pp. 035 006, 26, 2008. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/24/16/035006

[15] A. Smirnova and R. A. Renaut, “A family of preconditioned iteratively regularized methods for nonlinear minimization,” *J. Inverse Ill-Posed Probl.*, vol. 17, no. 4, pp. 405–418, 2009. [Online]. Available: http://dx.doi.org/10.1515/JIIP.2009.027

[16] F. Bauer, T. Hohage, and A. Munk, “Iteratively regularized Gauss-Newton method for nonlinear inverse problems with random noise,” *SIAM J. Numer. Anal.*, vol. 47, no. 3, pp. 1827–1846, 2009. [Online]. Available: http://dx.doi.org/10.1137/080721789

[17] P. Mahale and M. T. Nair, "A simplified generalized Gauss-Newton method for nonlinear ill-posed problems," *Math. Comp.*, vol. 78, no. 265, pp. 171–184, 2009. [Online]. Available: http://dx.doi.org/10.1090/S0025-5718-08-02149-2

[18] B. Kaltenbacher and B. Hofmann, "Convergence rates for the iteratively regularized Gauss-Newton method in Banach spaces," *Inverse Problems*, vol. 26, no. 3, pp. 035 007, 21, 2010. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/26/3/035007

[19] T. Hohage and S. Langer, "Acceleration techniques for regularized Newton methods applied to electromagnetic inverse medium scattering problems," *Inverse Problems*, vol. 26, no. 7, pp. 074 011, 15, 2010. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/26/7/074011

[20] A. B. Bakushinsky, M. Y. Kokurin, and A. Smirnova, *Iterative methods for ill-posed problems*, ser. Inverse and Ill-posed Problems Series. Walter de Gruyter GmbH & Co. KG, Berlin, 2011, vol. 54, an introduction.

[21] O. P. Ferreira, M. L. N. Gonçalves, and P. R. Oliveira, "Local convergence analysis of the Gauss-Newton method under a majorant condition," *J. Complexity*, vol. 27, no. 1, pp. 111–125, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jco.2010.09.001

[22] A. Smirnova, "On convergence rates for iteratively regularized procedures with linear penalty terms," *Inverse Problems*, vol. 28, no. 8, pp. 085 005, 19, 2012. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/28/8/085005

[23] A. B. Bakushinsky and M. Y. Kokurin, *Iterative methods for Ill-Posed Operator Equations with Smooth Operators.* Springer, Dordrecht, 2004.

[24] Q. Jin and U. Tautenhahn, "On the discrepancy principle for some Newton type methods for solving nonlinear inverse problems," *Numer. Math.*, vol. 111, no. 4, pp. 509–558, 2009. [Online]. Available: http://dx.doi.org/10.1007/s00211-008-0198-y

[25] J. Flemming and B. Hofmann, “A new approach to source conditions in regularization with general residual term,” *Numer. Funct. Anal. Optim.*, vol. 31, no. 1-3, pp. 254–284, 2010. [Online]. Available: http://dx.doi.org/10.1080/01630561003765721

[26] A. Neubauer, T. Hein, B. Hofmann, S. Kindermann, and U. Tautenhahn, “Improved and extended results for enhanced convergence rates of Tikhonov regularization in Banach spaces,” *Appl. Anal.*, vol. 89, no. 11, pp. 1729–1743, 2010. [Online]. Available: http://dx.doi.org/10.1080/00036810903517597

[27] J. Flemming and B. Hofmann, “Convergence rates in constrained Tikhonov regularization: equivalence of projected source conditions and variational inequalities,” *Inverse Problems*, vol. 27, no. 8, pp. 085 001, 11, 2011. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/27/8/085001

[28] B. Kaltenbacher, A. Neubauer, and O. Scherzer, *Iterative regularization methods for nonlinear ill-posed problems*, ser. Radon Series on Computational and Applied Mathematics. Walter de Gruyter GmbH & Co. KG, Berlin, 2008, vol. 6. [Online]. Available: http://dx.doi.org/10.1515/9783110208276

[29] A. B. Bakushinsky, “Iterative methods with fuzzy feedback for solving irregular operator equations,” *Dokl. Akad. Nauk*, vol. 428, no. 5, pp. 583–585, 2009. [Online]. Available: http://dx.doi.org/10.1134/S1064562409050263

[30] A. Bakushinsky and A. Smirnova, “Irregular operator equations by iterative methods with undetermined reverse connection,” *J. Inverse Ill-Posed Probl.*, vol. 18, no. 2, pp. 147–165, 2010. [Online]. Available: http://dx.doi.org/10.1515/JIIP.2010.005

[31] ——, “Discrepancy principle for generalized GN iterations combined with the reverse connection control,” *J. Inverse Ill-Posed Probl.*, vol. 18, no. 4, pp. 421–431, 2010. [Online]. Available: http://dx.doi.org/10.1515/JIIP.2010.019

[32] J. G. Nagy, K. Palmer, and L. Perrone, “Iterative methods for image deblurring: a Matlab object-oriented approach,” *Numer. Algorithms*, vol. 36, no. 1, pp. 73–93, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:NUMA.0000027762.08431.64

[33] J. Nagy, "Restoretools: An object oriented matlab package for image restoration (2002)," http://www.mathcs.emory.edu/~nagy/RestoreTools.

[34] T. M. P. Market, "http://www.tmbb.com.au/tamborine-mountain-produce-market."

[35] C. tea supplier, "http://www.chinese-tea-supplier.com/organic-tie-guan-yin–wu-long–high-quality-285.html."

[36] D. Colton and R. Kress, *Inverse acoustic and electromagnetic scattering theory*, 3rd ed., ser. Applied Mathematical Sciences. Springer, New York, 2013, vol. 93. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-4942-3

[37] C. R. Vogel, "Numerical solution of a nonlinear ill-posed problem arising in inverse scattering," *Inverse Problems*, vol. 1, no. 4, pp. 393–403, 1985. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/1/4/010

[38] K.-M. Lee, "Inverse scattering via nonlinear integral equations for a Neumann crack," *Inverse Problems*, vol. 22, no. 6, pp. 1989–2000, 2006. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/22/6/005

[39] H.-H. Qin and F. Cakoni, "Nonlinear integral equations for shape reconstruction in the inverse interior scattering problem," *Inverse Problems*, vol. 27, no. 3, pp. 035 005, 17, 2011. [Online]. Available: http://dx.doi.org/10.1088/0266-5611/27/3/035005

[40] E. N. Akimova and V. V. Vasin, "Stable parallel algorithms for solving the inverse gravimetry and magnitimetry problems," in *CD Proceedings of the 9th International Conference on Numerical Methods in Continuum Mechanics, Zilina, Slovakia, September 9-12,*, 2002.

[41] V. V. Vasin, E. N. Akimova, G. Y. Perestoronina, P. S. Martyshko, and V. A. P'yankov, "Methods for solving an inverse magnetometry problem," *Sib. Èlektron. Mat. Izv.*, vol. 5, pp. 620–631, 2008.

[42] V. Vasin and G. Skorik, "Iterative processes of gradient type with applications to gravimetry and magnetometry inverse problems," *J. Inverse*

Ill-Posed Probl., vol. 18, no. 8, pp. 855–876, 2010. [Online]. Available: http://dx.doi.org/10.1515/JIIP.2011.007

[43] A. B. Bakushinsky, A. Smirnova, and H. Liu, "Theoretical and numerical study of iteratively truncated newton's algorithm," *Applied Inverse Problems*, vol. 48, pp. 1–14, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-7816-4_1

[44] ——, "A posteriori error analysis for unstable models," *J. Inverse Ill-Posed Probl.*, vol. 20, no. 4, pp. 411–428, 2012. [Online]. Available: http://dx.doi.org/10.1515/jip-2012-0006

[45] V. V. Vasin and A. L. Ageev, *Ill-posed problems with a priori information*, ser. Inverse and Ill-posed Problems Series. VSP, Utrecht, 1995. [Online]. Available: http://dx.doi.org/10.1515/9783110900118

[46] Y. Gaponenko and V. Vinokurov, "A posteriori solution estimates for ill-posed inverse problems," *Dokl. Russian Acad. Sci.*, vol. 263, no. N2, pp. 277–280, 1982.

[47] A. G. Yagola and V. N. Titarenko, "A posteriori error estimation for ill-posed problems on some sourcewise represented or compact sets," in *Ill-posed and inverse problems*. VSP, Zeist, 2002, pp. 425–442.

[48] K. Y. Dorofeev, V. N. Titarenko, and A. G. Yagola, "Algorithms for constructing a posteriori errors of solutions to ill-posed problems," *Zh. Vychisl. Mat. Mat. Fiz.*, vol. 43, no. 1, pp. 12–25, 2003.

[49] A. Leonov, "On a posteriori accuracy estimates for solutions of linear ill-posed problems and extra-optimal regularizing algorithms," *Internet-journal "Numerical Methods and Programming"*, vol. 11, pp. 14–24, 2010.

[50] A. B. Bakushinsky, "A posteriori error estimates for approximate solutions of irregular operator equations," *Dokl. Akad. Nauk*, vol. 437, no. 4, pp. 439–440, 2011. [Online]. Available: http://dx.doi.org/10.1134/S1064562411020190

[51] M. Y. Kokurin, “Approximations of solutions to general irregular nonlinear operator equations and equations with quadratic operators,” *Comput. Math. Math. Phys.*, vol. 50, pp. 1783–1792, 2010.

[52] L. V. Kantorovich and G. P. Akilov, *Functional analysis*, 2nd ed. Pergamon Press, Oxford-Elmsford, N.Y., 1982, translated from the Russian by Howard L. Silcock.

[53] J. Nocedal and S. J. Wright, *Numerical optimization*, ser. Springer Series in Operations Research. Springer-Verlag, New York, 1999. [Online]. Available: http://dx.doi.org/10.1007/b98874

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

Appendix A

MATLAB CODE

A.1 MATLAB CODE 1

```
% THIS FUNCTION APPLIES GENERALIZED IRGN METHOD
% TO THE INVERSE SCATTERING PROBLEM
function ipi_paper_svd_pictures
global k0
% PARAMETERS
a = 0; b = 2*pi; c = 0; d = 2*pi/3; % limits of integration
step = 0.01; % step of integration
% First experiment
nmax = 100; % number of iterations
%tau0 = 10^(2); % initial regularization parameter with noise 0.005
%tau0 = 10^(2); % initial regularization parameter with noise 0.01 FIRST EXPERIMENT
%tau0 = 10^(3); % initial regularization parameter with noise 0.03
%tau0 = 10^(3); % initial regularization parameter with noise 0.05 SECOND EXPERIMENT
tau0 = 10^(-5); % initial regularization parameter without noise
beta = 1; % auxiliary regularization parameter
perc = 0.0; % percentage of noise
k0 = 1; % parameter of the model
% Compute the quadrture elements
[th, w, m] = quadrature('midpt', a, b, step);
step_phi = 0.001;
phi=c:step_phi:d;
k = length(phi);
```

```
% ——————————————————————%
% DIRECT PROBLEM
% ——————————————————————%
rhs = zeros(k,1);
for j = 1:k
rhs(j,1) = quad(@(theta)kernel(theta, phi(j),'pillow'),a,b);
end;
% Load noise from the file 'ns'
load ns rdm
nrhs = norm(rhs);
nrdm = norm(rdm);
delta = perc*nrhs/nrdm;
noise_rhs = delta*rdm' + rhs;
% ——————————————————————%
% INVERSE PROBLEM
% ——————————————————————%
% Compute model solution
rmod = model(th,'pillow');
% Initial guess
%r0 = 1*ones(m,1); %Peanut
%xi = 1*ones(m,1); %Peanut
r0 = 1.5*ones(m,1); %Pillow
xi = 1.5*ones(m,1); %Pillow
rn = r0;
disp('——————————————————————————————————————————')
disp(' n tau condFP discrepancy relative_error ')
disp('——————————————————————————————————————————')
% The beginning of ITERATIVE REGULARIZATION SCHEME
for n = 1:nmax
tau = tau0*(n^-beta);
```

```
%tau = tau0/exp(beta*n);
f = F(th,m,rn,phi,w,k);
f = f-noise_rhs;
discrep = norm(f)/norm(noise_rhs);
FP=Fprime(th,phi,rn,m,w,k);
condFP = cond(FP);
[U,S,V] = svd(FP);
rt = Tikhonov(U,S,V,tau,rn,xi,f);
%rt = TSVD(U,S,V,tau,rn,xi,f);
%rt = MTSVD(U,S,V,tau,rn,xi,f);
rt = real(rt);
relerr = norm(rt-rmod)/norm(rmod);
f = F(th,m,rt,phi,w,k);
f = f-noise_rhs;
discrept = norm(f)/norm(noise_rhs);
discrep = discrept;
rn = rt;
xi = rn;
fprintf('%6.2f  %8.6e  %8.6e  %8.6e  %8.6e  \n',n,tau,condFP,discrep,relerr);
end % End of For - loop
% PLOT THE OUTPUT
c=polar(th',rmod,'-');
set(c,'linewidth',[2],'color','b');
c=gca;
hold on
c=polar(th',rn,'-');
set(c,'linewidth',[2],'color','r');
c=gca;
hold on
c=polar(th',r0,'-');
```

```
set(c,'linewidth',[2],'color','c');
c=gca;
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nmax = 40; % number of iterations
%tau0 = 10ˆ(2); % initial regularization parameter with noise 0.005
tau0 = 10ˆ(2); % initial regularization parameter with noise 0.01 FIRST EXPERIMENT
%tau0 = 10ˆ(3); % initial regularization parameter with noise 0.03
%tau0 = 10ˆ(3); % initial regularization parameter with noise 0.05 SECOND EXPERI-
MENT
%tau0 = 10ˆ(-5); % initial regularization parameter without noise
beta = 1; % auxiliary regularization parameter
perc = 0.01; % percentage of noise
C = .002; % parameter of the stopping rule
% ——————————————%
% DIRECT PROBLEM
% ——————————————%
delta = perc*nrhs/nrdm;
noise_rhs = delta*rdm' + rhs;
% ——————————————%
% INVERSE PROBLEM
% ——————————————%
% Compute model solution
rmod = model(th,'pillow');
% Initial guess
%r0 = 1*ones(m,1); %Peanut
%xi = 1*ones(m,1); %Peanut
r0 = 1.5*ones(m,1); %Pillow
xi = 1.5*ones(m,1); %Pillow
rn = r0;
```

```
disp('————————————————————————————————————')
disp(' n tau condFP discrepancy relative_error value')
disp('————————————————————————————————————')
% The beginning of ITERATIVE REGULARIZATION SCHEME
for n = 1:nmax
tau = tau0*(n^-beta);
%tau = tau0/exp(beta*n);
f = F(th,m,rn,phi,w,k);
f = f-noise_rhs;
discrep = norm(f)/norm(noise_rhs);
FP=Fprime(th,phi,rn,m,w,k);
condFP = cond(FP);
[U,S,V] = svd(FP);
rt = Tikhonov(U,S,V,tau,rn,xi,f);
%rt = TSVD(U,S,V,tau,rn,xi,f);
%rt = MTSVD(U,S,V,tau,rn,xi,f);
rt = real(rt);
relerr = norm(rt-rmod)/norm(rmod);
f = F(th,m,rt,phi,w,k);
f = f-noise_rhs;
discrept = norm(f)/norm(noise_rhs);
value = (1+C*sqrt(tau))*perc;
check = discrept - value;
% STOP IF DIVERGENCE DETECTED OR DISCREPANCY INCREASED
if check < 0
fprintf('Time to Stop!\n');
break;
else
if relerr > 2
fprintf('Divergence Detected!\n');
```

```
break;
end
end
discrep = discrept;
rn = rt;
xi = rn;
fprintf('%6.2f %8.6e %8.6e %8.6e %8.6e %8.6e\n', n, tau, condFP, discrep, relerr, value);
end % End of For - loop
% PLOT THE OUTPUT
c=polar(th',rn,'-');
set(c,'linewidth',[2],'color','k');
c=gca;
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nmax = 40; % number of iterations
%tau0 = 10^(2); % initial regularization parameter with noise 0.005
%tau0 = 10^(2); % initial regularization parameter with noise 0.01 FIRST EXPERI-
MENT
tau0 = 10^(3); % initial regularization parameter with noise 0.03
%tau0 = 10^(3); % initial regularization parameter with noise 0.05 SECOND EXPERI-
MENT
%tau0 = 10^(-5); % initial regularization parameter without noise
beta = 1; % auxiliary regularization parameter
perc = 0.05; % percentage of noise
C = .002; % parameter of the stopping rule
% Compute the quadrture elements
[th, w, m] = quadrature('midpt', a, b, step);
delta = perc*nrhs/nrdm;
noise_rhs = delta*rdm' + rhs;
% ————————————————%
% INVERSE PROBLEM
% ———————————————————%
% Compute model solution
rmod = model(th,'pillow');
% Initial guess
%r0 = 1*ones(m,1); %Peanut
%xi = 1*ones(m,1); %Peanut
r0 = 1.5*ones(m,1); %Pillow
xi = 1.5*ones(m,1); %Pillow
rn = r0;
disp('————————————————————————————————————')
disp(' n tau condFP discrepancy relative_error value')
disp('————————————————————————————————————')
% The beginning of ITERATIVE REGULARIZATION SCHEME
for n = 1:nmax
tau = tau0*(n^-beta);
%tau = tau0/exp(beta*n);
f = F(th,m,rn,phi,w,k);
f = f-noise_rhs;
discrep = norm(f)/norm(noise_rhs);
FP=Fprime(th,phi,rn,m,w,k);
condFP = cond(FP);
[U,S,V] = svd(FP);
rt = Tikhonov(U,S,V,tau,rn,xi,f);
%rt = TSVD(U,S,V,tau,rn,xi,f);
%rt = MTSVD(U,S,V,tau,rn,xi,f);
rt = real(rt);
relerr = norm(rt-rmod)/norm(rmod);
f = F(th,m,rt,phi,w,k);
f = f-noise_rhs;
```

```
discrept = norm(f)/norm(noise_rhs);
value = (1+C*sqrt(tau))*perc;
check = discrept - value;
% STOP IF DIVERGENCE DETECTED OR DISCREPANCY INCREASED
if check < 0
fprintf('Time to Stop!\n');
break;
else
if relerr > 2
fprintf('Divergence Detected!\n');
break;
end
end
discrep = discrept;
rn = rt;
xi = rn;
fprintf('%6.2f %8.6e %8.6e %8.6e %8.6e %8.6e\n', n, tau, condFP, discrep, relerr, value);
end % End of For - loop
% PLOT THE OUTPUT
c=polar(th',rn,'-');
set(c,'linewidth',[2],'color','m');
c=gca;
legend('Exact','Noise-Free','Initial','1% Noise','5% Noise')
load gong;
sound(y,Fs);
fprintf('Done! Press Any Key to Continue...\n');
pause;
close all;
format;
% ——————————————————————————————%
% FUNCTION DEFINITIONS
% ——————————————————%
function y = kernel(theta,phi,modeltype)
global k0
switch (modeltype)
case 'peanut'
radius = ((cos(theta-pi/4)).^2.+.25.*(sin(theta-pi/4)).^2).^(.5);
case 'peach'
radius = 1.2 - 1/3*sin(theta) - 1/7*sin(3*theta);
case 'pear'
radius = 1.2 + 0.25*cos(3*theta);
case 'pillow'
radius = 1.25 + 0.25*cos(4*theta);
end
b=-2.*1i.*k0.*cos(phi-theta);
y = (exp(b.*radius).*(b.*radius-1)+1)./b.^2;
function radius = model(theta,modeltype)
switch (modeltype)
case 'peanut'
radius = ((cos(theta-pi/4)).^2.+.25.*(sin(theta-pi/4)).^2).^(.5)';
case 'peach'
radius = 1.2 - 1/3*sin(theta) - 1/7*sin(3*theta)';
case 'pear'
radius = 1.2 + 0.25*cos(3*theta)';
case 'pillow'
radius = 1.25 + 0.25*cos(4*theta)';
end
function vect = K(theta,radius,phi)
global k0
b = -2.*1i.*k0.*cos(phi-theta);
```

```
vect = (exp(b*radius).*(b*radius-1)+1)./b.^2;
function matr = Kprime(theta,phi,radius)
global k0
b = -2.*1i.*k0.*cos(phi-theta);
matr = radius.*exp(b*radius);
function f = F(theta,m,radius,phi,w,k)
f = zeros(k,1);
for j = 1:m
f(:,1) = f(:,1) + (K(theta(j),radius(j),phi).*w(j))';
end
function fp = Fprime(theta,phi,radius,m,w,k)
fp = zeros(k,m);
for j = 1:m
fp(:,j) = fp(:,j) + (Kprime(theta(j),phi,radius(j)).*w(j))';
end
function [theta,w,m] = quadrature(quadtype,a,b,step)
switch (quadtype)
case 'trap'
theta = a:step:b;
m = length(theta);
h = (b-a)/(m-1);
w = ones(1,m);
w(1) = 0.5;
w(m) = 0.5;
w = w*h;
case 'simp'
theta = a:step:b;
m = length(theta);
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'
theta = a+step/2:step:b-step/2;
m = length(theta);
h = (b-a)/m;
w = h*ones(1,m);
case 'gauss'
theta = a:step:b;
m = length(theta);
u = 1:m-1;
u = u ./ sqrt(4*u.^2 - 1);
A = zeros(m,m);
A(2:m+1:m*(m-1)) = u;
A(m+1:m+1:m^2-1) = u;
[v,theta] = eig(A);
[theta,k] = sort(diag(theta));
w = 2 * v(1,k)'.^2;
theta = (b-a)/2 * theta + (a+b)/2;
w = (b-a)/2 * w;
end
function x1 = Tikhonov(U,S,V, alpha,x,x_initial,f)
Q1 = S'*S./((S'*S == 0) + S'*S + alpha);
```

```
$Q2 = eye(size(V))./((eye(size(V)) == 0) + S' * S + alpha) * S';$
$x1 = V * ((eye(size(V)) - Q1) * V' * x\_initial + Q1 * V' * x - Q2 * U' * f);$
function x2 = TSVD(U,S,V, alpha,x,x_initial,f)
$Q1 = (S' * S > alpha);$
$Q2 = (S' * S > alpha)./((S' * S \leq alpha) + S' * S) * S';$
$x2 = V * ((eye(size(V)) - Q1) * V' * x\_initial + Q1 * V' * x - Q2 * U' * f);$
function x3 = MTSVD(U,S,V, alpha,x,x_initial,f)
if $alpha > 0$
$Q1 = (S' * S > alpha) + ((S' * S \leq alpha)\&(S' * S > 0)). * sqrt(S' * S)/sqrt(alpha);$
$Q2 = Q1./((Q1 == 0) + S' * S) * S';$
$x3 = V * ((eye(size(V)) - Q1) * V' * x\_initial + Q1 * V' * x - Q2 * U' * f);$
else
x3 = TSVD(U,S,V, alpha,x,x_initial,f);
end
%————————————————%
%OUTPUT
%————————————————%
>> ipi_paper_svd_pictures
```

```
————————————————————————————————
n   tau   condFP   discrepancy   relative_error
————————————————————————————————
1.00 1.000000e-05 4.172758e+17 1.064699e-01 1.906265e-01
2.00 5.000000e-06 3.968754e+17 8.461777e-02 1.654506e-01
3.00 3.333333e-06 3.947039e+17 6.945104e-02 1.496108e-01
4.00 2.500000e-06 3.649768e+17 5.869065e-02 1.383317e-01
5.00 2.000000e-06 3.550779e+17 5.060410e-02 1.297258e-01
6.00 1.666667e-06 2.542985e+17 4.431824e-02 1.228629e-01
7.00 1.428571e-06 2.825516e+17 3.931439e-02 1.172196e-01
8.00 1.250000e-06 2.671250e+17 3.525092e-02 1.124732e-01
9.00 1.111111e-06 3.271916e+17 3.189328e-02 1.084107e-01
```

10.00	1.000000e-06	2.346281e+17	2.907655e-02	1.048849e-01
11.00	9.090909e-07	2.708658e+17	2.668231e-02	1.017897e-01
12.00	8.333333e-07	3.181833e+17	2.462379e-02	9.904641e-02
13.00	7.692308e-07	3.630155e+17	2.283619e-02	9.659503e-02
14.00	7.142857e-07	2.916202e+17	2.127019e-02	9.438900e-02
15.00	6.666667e-07	2.758940e+17	1.988770e-02	9.239146e-02
16.00	6.250000e-07	2.589157e+17	1.865881e-02	9.057279e-02
17.00	5.882353e-07	3.144060e+17	1.755974e-02	8.890886e-02
18.00	5.555556e-07	3.443417e+17	1.657134e-02	8.737980e-02
19.00	5.263158e-07	2.928321e+17	1.567804e-02	8.596909e-02
20.00	5.000000e-07	2.623987e+17	1.486705e-02	8.466283e-02
21.00	4.761905e-07	2.725642e+17	1.412774e-02	8.344929e-02
22.00	4.545455e-07	2.577826e+17	1.345124e-02	8.231844e-02
23.00	4.347826e-07	2.444738e+17	1.283008e-02	8.126168e-02
24.00	4.166667e-07	3.248478e+17	1.225791e-02	8.027161e-02
25.00	4.000000e-07	2.775851e+17	1.172931e-02	7.934179e-02
26.00	3.846154e-07	3.417919e+17	1.123961e-02	7.846660e-02
27.00	3.703704e-07	2.994056e+17	1.078479e-02	7.764112e-02
28.00	3.571429e-07	2.856020e+17	1.036135e-02	7.686103e-02
29.00	3.448276e-07	2.492863e+17	9.966229e-03	7.612248e-02
30.00	3.333333e-07	3.540624e+17	9.596754e-03	7.542210e-02
31.00	3.225806e-07	3.113255e+17	9.250561e-03	7.475686e-02
32.00	3.125000e-07	3.468285e+17	8.925562e-03	7.412405e-02
33.00	3.030303e-07	3.372025e+17	8.619905e-03	7.352125e-02
34.00	2.941176e-07	2.994399e+17	8.331940e-03	7.294629e-02
35.00	2.857143e-07	3.300415e+17	8.060193e-03	7.239719e-02
36.00	2.777778e-07	3.377787e+17	7.803347e-03	7.187218e-02
37.00	2.702703e-07	3.583612e+17	7.560216e-03	7.136965e-02
38.00	2.631579e-07	2.564656e+17	7.329734e-03	7.088813e-02
39.00	2.564103e-07	2.895489e+17	7.110940e-03	7.042628e-02

```
40.00 2.500000e-07 2.829836e+17 6.902964e-03 6.998288e-02
41.00 2.439024e-07 3.001817e+17 6.705020e-03 6.955681e-02
42.00 2.380952e-07 3.430567e+17 6.516393e-03 6.914704e-02
43.00 2.325581e-07 2.800737e+17 6.336433e-03 6.875263e-02
44.00 2.272727e-07 2.623542e+17 6.164550e-03 6.837271e-02
45.00 2.222222e-07 3.486253e+17 6.000205e-03 6.800648e-02
46.00 2.173913e-07 2.952412e+17 5.842905e-03 6.765319e-02
47.00 2.127660e-07 2.838876e+17 5.692199e-03 6.731216e-02
48.00 2.083333e-07 3.147181e+17 5.547676e-03 6.698275e-02
49.00 2.040816e-07 2.614544e+17 5.408956e-03 6.666436e-02
50.00 2.000000e-07 2.850056e+17 5.275691e-03 6.635645e-02
51.00 1.960784e-07 3.117330e+17 5.147561e-03 6.605850e-02
52.00 1.923077e-07 2.717882e+17 5.024272e-03 6.577003e-02
53.00 1.886792e-07 2.914612e+17 4.905550e-03 6.549060e-02
54.00 1.851852e-07 3.419683e+17 4.791144e-03 6.521978e-02
55.00 1.818182e-07 2.603948e+17 4.680822e-03 6.495718e-02
56.00 1.785714e-07 3.244850e+17 4.574366e-03 6.470243e-02
57.00 1.754386e-07 3.113161e+17 4.471577e-03 6.445520e-02
58.00 1.724138e-07 2.487163e+17 4.372268e-03 6.421515e-02
59.00 1.694915e-07 2.607306e+17 4.276266e-03 6.398197e-02
60.00 1.666667e-07 2.800289e+17 4.183409e-03 6.375539e-02
61.00 1.639344e-07 3.052787e+17 4.093547e-03 6.353512e-02
62.00 1.612903e-07 2.438589e+17 4.006538e-03 6.332092e-02
63.00 1.587302e-07 3.092931e+17 3.922250e-03 6.311254e-02
64.00 1.562500e-07 2.836918e+17 3.840560e-03 6.290974e-02
65.00 1.538462e-07 2.910509e+17 3.761352e-03 6.271233e-02
66.00 1.515152e-07 2.647448e+17 3.684517e-03 6.252008e-02
67.00 1.492537e-07 2.579317e+17 3.609952e-03 6.233280e-02
68.00 1.470588e-07 3.833041e+17 3.537562e-03 6.215031e-02
69.00 1.449275e-07 3.120587e+17 3.467254e-03 6.197244e-02
```

70.00	1.428571e-07	2.562627e+17	3.398945e-03	6.179901e-02
71.00	1.408451e-07	3.556867e+17	3.332552e-03	6.162987e-02
72.00	1.388889e-07	2.708776e+17	3.267999e-03	6.146487e-02
73.00	1.369863e-07	3.240601e+17	3.205214e-03	6.130386e-02
74.00	1.351351e-07	2.747644e+17	3.144128e-03	6.114671e-02
75.00	1.333333e-07	3.041537e+17	3.084676e-03	6.099328e-02
76.00	1.315789e-07	2.902871e+17	3.026797e-03	6.084346e-02
77.00	1.298701e-07	3.094201e+17	2.970432e-03	6.069711e-02
78.00	1.282051e-07	3.531358e+17	2.915525e-03	6.055414e-02
79.00	1.265823e-07	3.070662e+17	2.862024e-03	6.041442e-02
80.00	1.250000e-07	3.357713e+17	2.809878e-03	6.027785e-02
81.00	1.234568e-07	3.276178e+17	2.759039e-03	6.014434e-02
82.00	1.219512e-07	3.381402e+17	2.709461e-03	6.001380e-02
83.00	1.204819e-07	3.526932e+17	2.661101e-03	5.988611e-02
84.00	1.190476e-07	2.889435e+17	2.613918e-03	5.976121e-02
85.00	1.176471e-07	3.217209e+17	2.567871e-03	5.963901e-02
86.00	1.162791e-07	2.407772e+17	2.522923e-03	5.951942e-02
87.00	1.149425e-07	2.823983e+17	2.479037e-03	5.940237e-02
88.00	1.136364e-07	3.088887e+17	2.436179e-03	5.928779e-02
89.00	1.123596e-07	3.279013e+17	2.394316e-03	5.917560e-02
90.00	1.111111e-07	3.027763e+17	2.353416e-03	5.906573e-02
91.00	1.098901e-07	2.998306e+17	2.313447e-03	5.895812e-02
92.00	1.086957e-07	2.982107e+17	2.274382e-03	5.885270e-02
93.00	1.075269e-07	2.790644e+17	2.236193e-03	5.874942e-02
94.00	1.063830e-07	2.781357e+17	2.198851e-03	5.864821e-02
95.00	1.052632e-07	2.932521e+17	2.162332e-03	5.854902e-02
96.00	1.041667e-07	2.483180e+17	2.126611e-03	5.845179e-02
97.00	1.030928e-07	2.883230e+17	2.091665e-03	5.835647e-02
98.00	1.020408e-07	3.210273e+17	2.057469e-03	5.826302e-02
99.00	1.010101e-07	3.157231e+17	2.024003e-03	5.817137e-02

```
100.00 1.000000e-07 3.251040e+17 1.991245e-03 5.808149e-02

--------------------------------------------------------------

n   tau   condFP   discrepancy   relative_error   value

--------------------------------------------------------------

1.00 1.000000e+02 4.172758e+17 1.234523e-01 2.332520e-01 1.020000e-02

2.00 5.000000e+01 3.825896e+17 1.035868e-01 2.216895e-01 1.014142e-02

3.00 3.333333e+01 3.361013e+17 8.827032e-02 2.101989e-01 1.011547e-02

4.00 2.500000e+01 3.192411e+17 7.633794e-02 1.994606e-01 1.010000e-02

5.00 2.000000e+01 3.699447e+17 6.688533e-02 1.896696e-01 1.008944e-02

6.00 1.666667e+01 3.803425e+17 5.931079e-02 1.808399e-01 1.008165e-02

7.00 1.428571e+01 3.335209e+17 5.317432e-02 1.729137e-01 1.007559e-02

8.00 1.250000e+01 3.477186e+17 4.814699e-02 1.658086e-01 1.007071e-02

9.00 1.111111e+01 3.783361e+17 4.398472e-02 1.594372e-01 1.006667e-02

10.00 1.000000e+01 3.384950e+17 4.050590e-02 1.537153e-01 1.006325e-02

11.00 9.090909e+00 2.434713e+17 3.757292e-02 1.485652e-01 1.006030e-02

12.00 8.333333e+00 2.771826e+17 3.507905e-02 1.439165e-01 1.005774e-02

13.00 7.692308e+00 2.687392e+17 3.293984e-02 1.397067e-01 1.005547e-02

14.00 7.142857e+00 3.462757e+17 3.108779e-02 1.358807e-01 1.005345e-02

15.00 6.666667e+00 2.743516e+17 2.946887e-02 1.323905e-01 1.005164e-02

16.00 6.250000e+00 2.437219e+17 2.804003e-02 1.291948e-01 1.005000e-02

17.00 5.882353e+00 3.183261e+17 2.676723e-02 1.262581e-01 1.004851e-02

18.00 5.555556e+00 2.723825e+17 2.562380e-02 1.235502e-01 1.004714e-02

19.00 5.263158e+00 3.059973e+17 2.458888e-02 1.210452e-01 1.004588e-02

20.00 5.000000e+00 2.707266e+17 2.364622e-02 1.187211e-01 1.004472e-02

21.00 4.761905e+00 3.274699e+17 2.278312e-02 1.165590e-01 1.004364e-02

22.00 4.545455e+00 2.963298e+17 2.198949e-02 1.145425e-01 1.004264e-02

23.00 4.347826e+00 3.144238e+17 2.125730e-02 1.126577e-01 1.004170e-02

24.00 4.166667e+00 2.731039e+17 2.057997e-02 1.108922e-01 1.004082e-02

25.00 4.000000e+00 3.071595e+17 1.995202e-02 1.092353e-01 1.004000e-02

26.00 3.846154e+00 2.830223e+17 1.936885e-02 1.076776e-01 1.003922e-02
```

```
27.00 3.703704e+00 2.773620e+17 1.882644e-02 1.062106e-01 1.003849e-02
28.00 3.571429e+00 3.541009e+17 1.832132e-02 1.048270e-01 1.003780e-02
29.00 3.448276e+00 2.762170e+17 1.785037e-02 1.035201e-01 1.003714e-02
30.00 3.333333e+00 2.371461e+17 1.741084e-02 1.022839e-01 1.003651e-02
31.00 3.225806e+00 3.513015e+17 1.700023e-02 1.011131e-01 1.003592e-02
32.00 3.125000e+00 2.216735e+17 1.661627e-02 1.000027e-01 1.003536e-02
33.00 3.030303e+00 3.031272e+17 1.625691e-02 9.894853e-02 1.003482e-02
34.00 2.941176e+00 2.675750e+17 1.592030e-02 9.794649e-02 1.003430e-02
35.00 2.857143e+00 3.901123e+17 1.560471e-02 9.699299e-02 1.003381e-02
36.00 2.777778e+00 2.474517e+17 1.530861e-02 9.608474e-02 1.003333e-02
37.00 2.702703e+00 2.621143e+17 1.503056e-02 9.521875e-02 1.003288e-02
38.00 2.631579e+00 3.068063e+17 1.476928e-02 9.439226e-02 1.003244e-02
39.00 2.564103e+00 3.090935e+17 1.452357e-02 9.360275e-02 1.003203e-02
40.00 2.500000e+00 4.195955e+17 1.429234e-02 9.284793e-02 1.003162e-02
----------------------------------------------------------------------
n   tau   condFP   discrepancy   relative_error   value
----------------------------------------------------------------------
1.00 1.000000e+03 4.172758e+17 1.493007e-01 2.411702e-01 5.316228e-02
2.00 5.000000e+02 7.250062e+17 1.427437e-01 2.385607e-01 5.223607e-02
3.00 3.333333e+02 5.171690e+17 1.350246e-01 2.351204e-01 5.182574e-02
4.00 2.500000e+02 4.089583e+17 1.272103e-01 2.311332e-01 5.158114e-02
5.00 2.000000e+02 6.541351e+17 1.198823e-01 2.268141e-01 5.141421e-02
6.00 1.666667e+02 4.537081e+17 1.132446e-01 2.223145e-01 5.129099e-02
7.00 1.428571e+02 2.852355e+17 1.072937e-01 2.177383e-01 5.119523e-02
8.00 1.250000e+02 3.171461e+17 1.019509e-01 2.131576e-01 5.111803e-02
9.00 1.111111e+02 3.239210e+17 9.713253e-02 2.086231e-01 5.105409e-02
10.00 1.000000e+02 3.784011e+17 9.277286e-02 2.041710e-01 5.100000e-02
11.00 9.090909e+01 4.695933e+17 8.882474e-02 1.998267e-01 5.095346e-02
12.00 8.333333e+01 3.578964e+17 8.525277e-02 1.956074e-01 5.091287e-02
13.00 7.692308e+01 3.659629e+17 8.202718e-02 1.915241e-01 5.087706e-02
```

```
14.00 7.142857e+01 5.673544e+17 7.912017e-02 1.875828e-01 5.084515e-02
15.00 6.666667e+01 3.278490e+17 7.650448e-02 1.837861e-01 5.081650e-02
16.00 6.250000e+01 2.734522e+17 7.415324e-02 1.801337e-01 5.079057e-02
17.00 5.882353e+01 3.116315e+17 7.204045e-02 1.766236e-01 5.076696e-02
18.00 5.555556e+01 3.491766e+17 7.014144e-02 1.732522e-01 5.074536e-02
19.00 5.263158e+01 3.665586e+17 6.843334e-02 1.700152e-01 5.072548e-02
20.00 5.000000e+01 3.055799e+17 6.689528e-02 1.669080e-01 5.070711e-02
21.00 4.761905e+01 3.562748e+17 6.550846e-02 1.639253e-01 5.069007e-02
22.00 4.545455e+01 3.376410e+17 6.425612e-02 1.610620e-01 5.067420e-02
23.00 4.347826e+01 3.821333e+17 6.312344e-02 1.583129e-01 5.065938e-02
24.00 4.166667e+01 2.765196e+17 6.209734e-02 1.556730e-01 5.064550e-02
25.00 4.000000e+01 2.810676e+17 6.116634e-02 1.531373e-01 5.063246e-02
26.00 3.846154e+01 3.189172e+17 6.032033e-02 1.507010e-01 5.062017e-02
27.00 3.703704e+01 2.931593e+17 5.955048e-02 1.483598e-01 5.060858e-02
28.00 3.571429e+01 3.295283e+17 5.884899e-02 1.461091e-01 5.059761e-02
29.00 3.448276e+01 3.247685e+17 5.820901e-02 1.439449e-01 5.058722e-02
30.00 3.333333e+01 2.863090e+17 5.762446e-02 1.418632e-01 5.057735e-02
31.00 3.225806e+01 3.365979e+17 5.708999e-02 1.398604e-01 5.056796e-02
32.00 3.125000e+01 3.674327e+17 5.660082e-02 1.379326e-01 5.055902e-02
33.00 3.030303e+01 3.066352e+17 5.615268e-02 1.360767e-01 5.055048e-02
34.00 2.941176e+01 3.312647e+17 5.574177e-02 1.342892e-01 5.054233e-02
35.00 2.857143e+01 2.839444e+17 5.536465e-02 1.325671e-01 5.053452e-02
36.00 2.777778e+01 3.121810e+17 5.501823e-02 1.309074e-01 5.052705e-02
37.00 2.702703e+01 3.020804e+17 5.469972e-02 1.293072e-01 5.051988e-02
38.00 2.631579e+01 2.501730e+17 5.440659e-02 1.277638e-01 5.051299e-02
39.00 2.564103e+01 2.947907e+17 5.413655e-02 1.262747e-01 5.050637e-02
40.00 2.500000e+01 2.936434e+17 5.388752e-02 1.248373e-01 5.050000e-02
Done! Press Any Key to Continue...
```

A.2 MATLAB CODE 2

ITERATIVELY TRUNCATED NEWTON

```
function output = test_truncated_1
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;
delta_set = [0 0.05 0.1 0.15 0.2];
[sd,td,Sd,Td,wnd,wmd] = quadrature2d('trap',a,b,c,d,nd,md);
[s,t,S,T,wn,wm] = quadrature2d('trap',a,b,c,d,n,m);
% ————————————————————————%
% DIRECT PROBLEM
% ————————————————————————%
% SET X TO SOME KNOWN FUNCTION ON A FINE GRID
TTd = (Td-c)/(d-c); % domain normalization
SSd = (Sd-a)/(b-a); % domain normalization
Xd = -sin(abs(10*TTd-5)-abs(10*SSd-5))/3 + 1;
Xk_delta = ones(m,n,5);
Fk_delta = zeros(md,nd,5);
kk = 0;
na=1;
fexact = F(Td,Sd,td,sd,wmd,wnd,Xd);
svalues = zeros(m*n,kmax,4);
```

```
delta_k = 0;
for delta = delta_set
alpha0 = alphavector(na);
kk = kk+1;
fprintf('Delta = %1.4E\n', delta);
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));
fprintf('Absolute Error on the right - hand side = %1.4E\n', abs_err_rhs);
rel_err_rhs = norm(delta*rand(md*nd,1),'fro')/norm(fexact,'fro');
fprintf('Relative Error on the right - hand side = %1.4E\n', rel_err_rhs);
% SET X TO SOME KNOWN FUNCTION ON A COARSE 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
f = MtoV(Fm,m,n);
TT = (T-c)/(d-c); % domain normalization
SS = (S-a)/(b-a); % domain normalization
X = -sin(abs(10*TT-5)-abs(10*SS-5))/3 + 1;
x = MtoV(X,m,n);
% ————————————————————————————%
% INVERSE PROBLEM
% ————————————————————————————%
% INITIAL APPROXIMATION
X0 = 0.1*ones(m,n);
Xk = X0;
```

```
x0 = MtoV(X0,m,n);
xk=x0;
relerr = norm(x0-x,'fro')/norm(x,'fro');
fprintf('Relative  Error = %1.4E\n', relerr);
for k = 1:kmax
fprintf('Iteration  k = %d', k);
Xklast = xk;
discreplast = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro')/norm(f, 'fro');
% ITERATIVELY REGULARIZED SCHEME
beta = 1;
alpha = alpha0*(k^-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);
if delta ≥ 0
[UM, SM, VM] = svd(FP);
sv = diag(SM);
I=ones(m*n,1);
sinv = 1./sv;
fprintf('alpha = %1.4E\n', alpha);
count=0; %for counting singular values which are less than alpha.
for i = 1:m*n
if sv(i) < alpha
sinv(i) = 0;
I(i)=0;
count=count+1;
end
end
fprintf('#  of  singular  values  which  are  cut  off :  %1.4E\n', count);
```

```
num = zeros(m*n,1);
svco = zeros(m*n,1);
for i=1:count
num(i)=m*n-count+i;
svco(i)=sv(m*n-count+i);
end
if delta_k ≥ 1  &&  delta_k < 5
svalues(:,2*k-1,delta_k) = num; %store the cut off singular values of each iteration.
svalues(:,2*k,delta_k) = svco;
end
PIS = diag(sinv);
PIFP = VM*PIS*UM’;
IS = diag(I);
PIP = UM*IS*UM’;
PIPC = UM*(diag(ones(m*n,1))-IS)*UM’;
end
Pk = - PIFP*G;
xk = PIP*xk + PIPC*x0 + Pk;
Xk = VtoM(xk,m,n);
% COMPUTE THE RELATIVE ERROR
relerr = norm(x-xk,’fro’)/norm(x,’fro’);
disp(sprintf('Relative  Error = %1.4E', relerr));
output(k+1,:) = [k relerr];
% STOP IF CONVERGENCE OR DIVERGENCE DETECTED
if ((norm(xk − Xklast,' fro') < 1E − 5) | (relerr < 1E − 10))
disp(sprintf('Convergence  Detected!'));
break;
else
if ((norm(xk − Xklast,' fro') > 100) | (relerr > 2))
fprintf('Divergence  Detected!');
```

```
xk = Xklast;
Xk = VtoM(xk,m,n);
break;
end
end
discrep = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro')/norm(f, 'fro');
if (discreplast < discrep)
xk = Xklast;
Xk = VtoM(xk,m,n);
discrep = discreplast;
break;
end
end
Xk_delta(:,:,kk) = Xk;
discrep = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro')/norm(f, 'fro');
fprintf('Discrepancy = %1.4E\n', discrep);
singular = svd(FP);
minimum = min(singular);
maximum = max(singular);
fprintf('Minimum  singular  value = %1.4E\n', minimum);
fprintf('Maximum  singular  value = %1.4E\n', maximum);
na=na+1; %move to next alpha0 for higher level of noise.
delta_k = delta_k+1;
end %delta
%————————————————————————————————%
fig6 = figure;
subplot(2,2,1)
plot(svalues(:,9,1),svalues(:,10,1),'co',svalues(:,11,1),svalues(:,12,1),'b+',svalues(:,13,1),
svalues(:,14,1),'k*',svalues(:,15,1),svalues(:,16,1),'mo',svalues(:,17,1),svalues(:,18,1),'g.')
title('Relative level of noise 2.5%');
```

```
legend('5th iteration', '6th iteration','7th iteration','8th iteration','9th iteration',0);
subplot(2,2,2)
plot(svalues(:,9,2),svalues(:,10,2),'co',svalues(:,11,2),svalues(:,12,2),'b+',svalues(:,13,2),
svalues(:,14,2),'k*',svalues(:,15,2),svalues(:,16,2),'mo',svalues(:,17,2),svalues(:,18,2),'g.')
title('Relative level of noise 5%');
legend('5th iteration', '6th iteration','7th iteration','8th iteration','9th iteration',0);
subplot(2,2,3)
plot(svalues(:,9,3),svalues(:,10,3),'co',svalues(:,11,3),svalues(:,12,3),'b+',svalues(:,13,3),
svalues(:,14,3),'k*',svalues(:,15,3),svalues(:,16,3),'mo',svalues(:,17,3),svalues(:,18,3),'g.')
title('Relative level of noise 7.5%');
legend('5th iteration', '6th iteration','7th iteration','8th iteration','9th iteration',0);
subplot(2,2,4)
plot(svalues(:,9,4),svalues(:,10,4),'co',svalues(:,11,4),svalues(:,12,4),'b+',svalues(:,13,4),
svalues(:,14,4),'k*',svalues(:,15,4),svalues(:,16,4),'mo',svalues(:,17,4),svalues(:,18,4),'g.')
title('Relative level of noise 10%');
legend('5th iteration', '6th iteration','7th iteration','8th iteration','9th iteration',0);
print ex4s.eps
figure(fig6);
% 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('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('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('Relative level of noise 2.5%');
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('Relative level of noise 5%');
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('Relative level of noise 7.5%');
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('Relative level of noise 10%');
set(gca,'GridLineStyle','-','linewidth',[1])
print ex1s.eps
figure(fig1);
fig2 = figure;
axisName = 's';
axisValue = 10.0;
switch (axisName)
```

```
case {'s', 'x'}
indx = find(abs(s-axisValue) == min(abs(s-axisValue)));
c=plot(t,X(:,indx),'s-',t,Xk_delta(:,indx,1),'^-',t,Xk_delta(:,indx,2),'*-',
t,Xk_delta(:,indx,3),'p-',t,Xk_delta(:,indx,4),'d-',t,Xk_delta(:,indx,5),'o-',t,X0(:,indx),'k--');
set(c,'linewidth',[2]);
case {'t', '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-',s,Xk_delta(indx,:,4),'d-',s,Xk_delta(indx,:,5),'o-',s,X0(indx,:),'k--');
set(c,'linewidth',[2]);
end
xlabel(sprintf('%c = %.2f',axisName,axisValue));
title('Cross-Sectional Comparison');
legend('exact solution', 'noise-free reconstr','2.5% noise', '5% noise','7.5% noise', '10%
noise','initial guess',0);
print ex2s.eps
figure(fig2);
fig3 = figure;
axisName = 't';
axisValue =4.0;
switch (axisName)
case {'s', 'x'}
indx = find(abs(s-axisValue) == min(abs(s-axisValue)));
c=plot(t,X(:,indx),'s-',t,Xk_delta(:,indx,1),'^-',t,Xk_delta(:,indx,2),'*-',
t,Xk_delta(:,indx,3),'p-',t,Xk_delta(:,indx,4),'d-',t,Xk_delta(:,indx,5),'o-',t,X0(:,indx),'k--');
set(c,'linewidth',[2]);
case {'t', '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-',s,Xk_delta(indx,:,4),'d-',s,Xk_delta(indx,:,5),'o-',s,X0(indx,:),'k--');
```

```
set(c,'linewidth',[2]);
end
xlabel(sprintf('%c = %.2f',axisName,axisValue));
title('Cross-Sectional Comparison');
legend('exact solution', 'noise-free reconstr','2.5% noise', '5% noise','7.5% noise', '10%
noise','initial guess',0);
print ex2s.eps
figure(fig3);
fig4 = figure;
axisName = 'sd';
axisValue = 10.0;
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,2),'*-',td,Fk_delta(:,indx,3),'p-',
td,Fk_delta(:,indx,4),'d-',td,Fk_delta(:,indx,5),'o-');
set(c,'linewidth',[2]);
case {'td', 'y'}
indx = find(abs(t-axisValue) == min(abs(t-axisValue)));
c=plot(sd,Fk_delta(indx,:,1),'^-',sd,Fk_delta(indx,:,2),'*-',sd,Fk_delta(indx,:,3),'p-',
sd,Fk_delta(indx,:,4),'d-',sd,Fk_delta(indx,:,5),'o-');
set(c,'linewidth',[2]);
end
xlabel(sprintf('%c = %.2f',axisName,axisValue));
legend( 'noise-free right-hand side','2.5% relative noise', '5% relative noise','7.5% relative
noise', '10% relative noise',0);
print ex3s.eps
figure(fig4);
fig5 = figure;
axisName = 'td';
```

```
axisValue = 4.0;
switch (axisName)
case {'sd', 'x'}
indx = find(abs(sd-axisValue) == min(abs(sd-axisValue)));
c=plot(td,Fk_delta(:,indx,1),'ˆ-',td,Fk_delta(:,indx,2),'*-',td,Fk_delta(:,indx,3),'p-',
td,Fk_delta(:,indx,4),'d-',td,Fk_delta(:,indx,5),'o-');
set(c,'linewidth',[2]);
case {'td', 'y'}
indx = find(abs(td-axisValue) == min(abs(td-axisValue)));
c=plot(sd,Fk_delta(indx,:,1),'ˆ-',sd,Fk_delta(indx,:,2),'*-',sd,Fk_delta(indx,:,3),'p-',
sd,Fk_delta(indx,:,4),'d-',sd,Fk_delta(indx,:,5),'o-');
set(c,'linewidth',[2]);
end
xlabel(sprintf('%c = %.2f',axisName,axisValue));
legend( 'noise-free right-hand side','2.5% relative noise', '5% relative noise', '7.5% rela-
tive noise', '10% relative noise',0);
print ex3s.eps
figure(fig5);
load gong;
sound(y,Fs);
fprintf('Done! Press Any Key to Continue...');
pause;
close all;
format;
% ——————————————————————%
% FUNCTION DEFINITIONS
% ——————————————————————%
function vect = K(T,S,xsi,nu,x)
global H
vect = x*(((T-xsi).ˆ2+(S-nu).ˆ2+xˆ2).ˆ(-1.5))- ...
```

```
H*(((T-xsi).^2+(S-nu).^2+H^2).^(-1.5));
function vect = Kprime(t,s,XSI,NU,X)
vect = ((t-XSI).^2+(s-NU).^2-2*X.^2).*(((t-XSI).^2+(s-NU).^2+X.^2).^(-5/2));
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);
```

```
%——————————————————————%
%OUTPUT
%——————————————————————%
>> test_truncated_1
Delta = 0.0000E+00
Absolute Error on the right - hand side = 0.0000E+00
Relative Error on the right - hand side = 0.0000E+00
Relative Error = 9.0530E-01
Iteration k = 1  alpha = 5.0000E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.6329E-01
Iteration k = 2  alpha = 2.5000E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.0344E-01
Iteration k = 3  alpha = 1.6667E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 7.0541E-01
Iteration k = 4  alpha = 1.2500E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 4.8339E-01
Iteration k = 5  alpha = 1.0000E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 1.3443E-01
Iteration k = 6  alpha = 8.3333E-02
# of singular values which are cut off: 2.5100E+02
Relative Error = 4.5948E-02
Iteration k = 7  alpha = 7.1429E-02
# of singular values which are cut off: 3.1800E+02
Relative Error = 4.5232E-02
Iteration k = 8  alpha = 6.2500E-02
```

```
# of singular values which are cut off: 3.0800E+02
Relative Error = 3.9372E-02
Iteration k = 9  alpha = 5.5556E-02
# of singular values which are cut off: 2.9200E+02
Relative Error = 3.5406E-02
Iteration k = 10  alpha = 5.0000E-02
# of singular values which are cut off: 2.8100E+02
Relative Error = 3.4042E-02
Iteration k = 11  alpha = 4.5455E-02
# of singular values which are cut off: 2.7000E+02
Relative Error = 2.7715E-02
Iteration k = 12  alpha = 4.1667E-02
# of singular values which are cut off: 2.6400E+02
Relative Error = 2.6408E-02
Iteration k = 13  alpha = 3.8462E-02
# of singular values which are cut off: 2.5900E+02
Relative Error = 2.5053E-02
Iteration k = 14  alpha = 3.5714E-02
# of singular values which are cut off: 2.5000E+02
Relative Error = 2.5191E-02
Iteration k = 15  alpha = 3.3333E-02
# of singular values which are cut off: 2.4700E+02
Relative Error = 2.4786E-02
Iteration k = 16  alpha = 3.1250E-02
# of singular values which are cut off: 2.4000E+02
Relative Error = 2.3115E-02
Iteration k = 17  alpha = 2.9412E-02
# of singular values which are cut off: 2.3000E+02
Relative Error = 2.1900E-02
Iteration k = 18  alpha = 2.7778E-02
```

```
# of singular values which are cut off: 2.2700E+02

Relative Error = 2.1914E-02

Iteration k = 19  alpha = 2.6316E-02

# of singular values which are cut off: 2.2000E+02

Relative Error = 2.1683E-02

Iteration k = 20  alpha = 2.5000E-02

# of singular values which are cut off: 2.1400E+02

Relative Error = 2.1618E-02

Iteration k = 21  alpha = 2.3810E-02

# of singular values which are cut off: 2.1000E+02

Relative Error = 2.1787E-02

Iteration k = 22  alpha = 2.2727E-02

# of singular values which are cut off: 2.0800E+02

Relative Error = 2.1528E-02

Iteration k = 23  alpha = 2.1739E-02

# of singular values which are cut off: 2.0200E+02

Relative Error = 2.1709E-02

Iteration k = 24  alpha = 2.0833E-02

# of singular values which are cut off: 2.0100E+02

Relative Error = 2.1657E-02

Iteration k = 25  alpha = 2.0000E-02

# of singular values which are cut off: 1.9700E+02

Relative Error = 2.0407E-02

Discrepancy = 3.1968E-04

Minimum singular value = 3.2064E-04

Maximum singular value = 3.1531E+00

Delta = 5.0000E-02

Absolute Error on the right - hand side = 2.1374E+00

Relative Error on the right - hand side = 2.5906E-02

Relative Error = 9.0530E-01
```

```
Iteration k = 1  alpha = 1.0000E+00
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.6329E-01
Iteration k = 2  alpha = 5.0000E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.0344E-01
Iteration k = 3  alpha = 3.3333E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 7.0538E-01
Iteration k = 4  alpha = 2.5000E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 4.8329E-01
Iteration k = 5  alpha = 2.0000E-01
# of singular values which are cut off: 1.0000E+01
Relative Error = 1.3759E-01
Iteration k = 6  alpha = 1.6667E-01
# of singular values which are cut off: 3.4200E+02
Relative Error = 9.0253E-02
Iteration k = 7  alpha = 1.4286E-01
# of singular values which are cut off: 3.9800E+02
Relative Error = 7.4815E-02
Iteration k = 8  alpha = 1.2500E-01
# of singular values which are cut off: 3.8500E+02
Relative Error = 7.5435E-02
Iteration k = 9  alpha = 1.1111E-01
# of singular values which are cut off: 3.7200E+02
Relative Error = 6.9148E-02
Iteration k = 10  alpha = 1.0000E-01
# of singular values which are cut off: 3.5900E+02
Relative Error = 7.0647E-02
```

```
Iteration k = 11  alpha = 9.0909E-02
# of singular values which are cut off: 3.5200E+02
Relative Error = 7.4626E-02
Discrepancy = 1.4704E-02
Minimum singular value = 2.5501E-04
Maximum singular value = 3.1807E+00
Delta = 1.0000E-01
Absolute Error on the right - hand side = 4.2585E+00
Relative Error on the right - hand side = 5.1638E-02
Relative Error = 9.0530E-01
Iteration k = 1  alpha = 1.5000E+00
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.6329E-01
Iteration k = 2  alpha = 7.5000E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.0345E-01
Iteration k = 3  alpha = 5.0000E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 7.0544E-01
Iteration k = 4  alpha = 3.7500E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 4.8359E-01
Iteration k = 5  alpha = 3.0000E-01
# of singular values which are cut off: 2.4000E+01
Relative Error = 1.4783E-01
Iteration k = 6  alpha = 2.5000E-01
# of singular values which are cut off: 4.0200E+02
Relative Error = 1.3322E-01
Iteration k = 7  alpha = 2.1429E-01
# of singular values which are cut off: 4.5500E+02
```

```
Relative Error = 1.0278E-01

Discrepancy = 3.9158E-02

Minimum singular value = 1.3915E-04

Maximum singular value = 3.3371E+00

Delta = 1.5000E-01

Absolute Error on the right - hand side = 6.4047E+00

Relative Error on the right - hand side = 7.7415E-02

Relative Error = 9.0530E-01

Iteration k = 1  alpha = 2.0000E+00

# of singular values which are cut off: 0.0000E+00

Relative Error = 8.6330E-01

Iteration k = 2  alpha = 1.0000E+00

# of singular values which are cut off: 0.0000E+00

Relative Error = 8.0348E-01

Iteration k = 3  alpha = 6.6667E-01

# of singular values which are cut off: 0.0000E+00

Relative Error = 7.0553E-01

Iteration k = 4  alpha = 5.0000E-01

# of singular values which are cut off: 0.0000E+00

Relative Error = 4.8400E-01

Iteration k = 5  alpha = 4.0000E-01

# of singular values which are cut off: 4.9000E+01

Relative Error = 1.5745E-01

Iteration k = 6  alpha = 3.3333E-01

# of singular values which are cut off: 4.4800E+02

Relative Error = 1.5135E-01

Iteration k = 7  alpha = 2.8571E-01

# of singular values which are cut off: 4.9300E+02

Relative Error = 1.2326E-01

Discrepancy = 5.8913E-02
```

```
Minimum singular value = 1.3825E-04
Maximum singular value = 5.4886E+00
Delta = 2.0000E-01
Absolute Error on the right - hand side = 8.5650E+00
Relative Error on the right - hand side = 1.0291E-01
Relative Error = 9.0530E-01
Iteration k = 1  alpha = 2.5000E+00
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.6328E-01
Iteration k = 2  alpha = 1.2500E+00
# of singular values which are cut off: 0.0000E+00
Relative Error = 8.0341E-01
Iteration k = 3  alpha = 8.3333E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 7.0525E-01
Iteration k = 4  alpha = 6.2500E-01
# of singular values which are cut off: 0.0000E+00
Relative Error = 4.8284E-01
Iteration k = 5  alpha = 5.0000E-01
# of singular values which are cut off: 7.5000E+01
Relative Error = 1.6492E-01
Iteration k = 6  alpha = 4.1667E-01
# of singular values which are cut off: 4.8900E+02
Relative Error = 1.8881E-01
Iteration k = 7  alpha = 3.5714E-01
# of singular values which are cut off: 5.0400E+02
Relative Error = 1.5551E-01
Discrepancy = 1.3363E-01
Minimum singular value = 1.1180E-04
Maximum singular value = 2.0368E+01
```

```
Done! Press Any Key to Continue...

ans =

0 0

1.0000 0.8633

2.0000 0.8034

3.0000 0.7053

4.0000 0.4828

5.0000 0.1649

6.0000 0.1888

7.0000 0.1555

8.0000 0.0754

9.0000 0.0691

10.0000 0.0706

11.0000 0.0746

12.0000 0.0264

13.0000 0.0251

14.0000 0.0252

15.0000 0.0248

16.0000 0.0231

17.0000 0.0219

18.0000 0.0219

19.0000 0.0217

20.0000 0.0216

21.0000 0.0218

22.0000 0.0215

23.0000 0.0217

24.0000 0.0217

25.0000 0.0204
```

A.3 MATLAB CODE 3

```
% THIS FUNCTION TESTS THE ITERATIVELY REGULARIZED GAUSS-NEWTON
% ALGORITHM FOR SOLVING THE NONLINEAR MAGNETOMETRY PROBLEM.
function output = test19
global H
format long;
warning off;
% PARAMETERS
a = 2.8; b = 20.0;
c = 0; d = 8.0;
m = 40; n = 86;
H = 2.0;
kmax =50;
delta = 0.1;
alpha0 = 0.1;
% COMPUTE THE QUADRATURE ELEMENTS
[s, t, S, T, wn, wm] = quadrature2d('midpt', a, b, c, d, n, m);
% ——————————————————————————%
% DIRECT PROBLEM
% ——————————————————————————%
% SET X TO SOME KNOWN FUNCTION
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);
disp(sprintf('Delta = %1.4E', delta))
% CALCULATE f DIRECTLY FROM X
f = F(T, S, t, s, wm, wn, X) + delta ∗ rand(m ∗ n, 1);
RHS = VtoM(f, m, n);
```

```
F0 = VtoM(F(T,S,t,s,wm,wn,X),m,n);
rel_err_rhs = norm(delta*rand(m*n,1),'fro')/norm(F(T,S,t,s,wm,wn,X),'fro');
disp(sprintf('Relative Error on the right hand side = %1.4E', rel_err_rhs))
% ——————————————————%
% INVERSE PROBLEM
% ——————————————————%
% INITIAL SOLUTION
X0 = 0.1 * ones(m,n);
x0 = MtoV(X0,m,n);
relerr = norm(x0-x,'fro')/norm(x,'fro');
disp(sprintf('Relative Error = %1.4E', relerr));
output(1,:) = [0relerr];
Xk = X0;
xk = x0;
for k = 1:kmax
disp(sprintf('Iteration k = %d', k));
Xklast = xk;
discreplast = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro');
% ITERATIVE REGULARIZATION SCHEME
alpha = alpha0*log(1+k^-0.25);
% 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);
% FINISH THE ITERATION
Pk = −(FP + alpha * eye(m * n, m * n))\(G + alpha * (xk − x0));
xk = xk + Pk;
Xk = VtoM(xk,m,n);
% COMPUTE THE RELATIVE ERROR
relerr = norm(x-xk,'fro')/norm(x,'fro');
```

```
disp(sprintf('Relative Error = %1.4E', relerr));
output(k+1,:) = [k relerr];
% STOP IF CONVERGENCE OR DIVERGENCE DETECTED
if ((norm(xk − Xklast,′fro′) < 1E − 5) | (relerr < 0.01))
disp(sprintf('Convergence Detected!'));
break;
else
if ((norm(xk − Xklast,′fro′) > 100) | (relerr > 2))
disp(sprintf('Divergence Detected!'));
break;
end;
end;
discrep = norm(F(T,S,t,s,wm,wn,Xk) - f, 'fro');
if (discreplast < discrep)
xk = Xklast;
Xk = VtoM(xk,m,n);
discrep = discreplast;
break;
end
end;
disp(sprintf('Discrepancy = %1.4E', discrep));
fig1 = figure;
subplot(2,2,1);
surf(s,t,X);
subplot(2,2,2);
surf(s,t,Xk);
subplot(2,2,3)
surf(s,t,F0);
subplot(2,2,4)
surf(s,t,RHS);
```

```
figure(fig1);
disp(sprintf('Done! Press Any Key to Continue...'));
pause;
close all;
format;
% ——————————————————————%
% FUNCTION DEFINITIONS
% ——————————————————————%
function vect = K(T,S,xsi,nu,x)
global H
vect = x*(((T-xsi).^2+(S-nu).^2+x^2).^(-1.5))- ...
H*(((T-xsi).^2+(S-nu).^2+H^2).^(-1.5));
function vect = Kprime(t,s,XSI,NU,X)
vect = ((t-XSI).^2+(s-NU).^2-2*X.^2).*(((t-XSI).^2+(s-NU).^2+X.^2).^(-5/2));
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.2̂ - 1);
A = zeros(n,n);
A(2:n+1:n*(n-1)) = u;
A(n+1:n+1:nˆ2-1) = u;
[v,x] = eig(A);
```

```
[x,k] = sort(diag(x));
w = 2 * v(1,k)'.^2;
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);
%——————————————————————%
%OUTPUT
%——————————————————————%
>> test19
Delta = 1.0000E-01
Relative Error on the right - hand side= 4.9932E-02
Relative Error = 9.0068E-01
Iteration k = 1
Relative Error = 8.4837E-01
Iteration k = 2
Relative Error = 6.9552E-01
Iteration k = 3
Relative Error = 3.8027E-01
Discrepancy = 5.8545E+01
Done! Press Any Key to Continue...
ans =
0 0.9007
1.0000 0.8484
2.0000 0.6955
3.0000 0.3803
```