SOLUTION OF A 1-D INVERSE PROBLEM BY THE CONVEXIFICATION METHOD

by

Ray Gabriel Abney

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Mathematics

Charlotte

2019

Approved by:

____

Dr. Mikhail Klibanov

____

Dr. Loc Nguyen

____

Dr. Kevin McGoff

ABSTRACT

RAY GABRIEL ABNEY. Solution of a 1-D inverse problem by the convexification method. (Under the direction of DR. MIKHAIL KLIBANOV)

In this thesis, we demonstrate a method outlined by Dr. Mikhail Klibanov for solving a 1-D coefficient inverse problem by the convexification method. Our inverse problem in question concerns finding buried bombs, where the dielectric constants of the bomb and the sand in which it is buried are represented by the coefficient function $c(x)$. The goal of our method is to approximate $c(x)$. In the method demonstrated in this thesis, we compute an orthonormal basis from the set $\{k^n e^k\}_{n=0}^{\infty}$ consisting of N vectors. Then we derive a series of boundary value problems from our coeficient inverse problem. Then we get a functional $J_{\lambda,\gamma}(V)$ that we wish to minimize. Then we find the unique minimizer V_{min}. And finally, having our V_{min}, we use it to compute an approximate solution $c_{\text{approx}}(x)$ for our coefficient inverse problem.

DEDICATION

I would like to dedicate this thesis to my mother, Patricia Abney, because she has been the most instrumental in getting me through college, and through school in general.

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr. Klibanov, and the other members of my committee, Drs. Nguyen and McGoff, for all of the help and guidance they have given me while doing this project.

I would also like to especially acknowledge all of the (very patient) guidance of Dr. Aleksandr Kolesov for breaking down for me all of the intricacies of our project. He was extremely helpful.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The problem discussed in this thesis comes in two parts, a forward problem and an inverse problem. Our forward problem, as quoted from [1], is as follows:

> Below $k > 0$ is the wave number. Also, for any $z \in \mathbb{C}$ we denote $\overline{z}$ its complex conjugate. Let $c_0 > 0$ be a positive number. Let $c(x), x \in \mathbb{R}$ be the function with the following properties:
>
> $$c \in C^2(\mathbb{R}), \qquad c(x) \geq c_0, \qquad \forall x \in \mathbb{R}, \tag{1.1}$$
>
> $$c(x) = 1, \qquad \forall x \notin (0,1). \tag{1.2}$$
>
> In our application $c(x)$ is the spatially distributed dielectric constant of the medium. Let $x_0 < 0$ be the position of the point source. The forward problem is:
>
> $$u'' + k^2 c(x) u = -\delta(x - x_0), \qquad x \in \mathbb{R}, \tag{1.3}$$
>
> $$\lim_{x \to \infty} (u' + iku) = 0, \qquad \lim_{x \to -\infty} (u' - iku) = 0. \tag{1.4}$$

The δ in equation (1.3) is a delta function. From [1], letting $u_0(x,k)$ be the solution of the problem defined by equations (1.3) and (1.4) with $c(x) \equiv 1$, we have

$$u_0(x,k) = \frac{\exp(-ik|x - x_0|)}{2ik}. \tag{1.5}$$

Thus, also as quoted from [1], we have our coefficient inverse problem:

> Let $[\underline{k}, \overline{k}] \subset (0, \infty)$ be an interval of wave numbers k. Reconstruct the function $c(x)$, assuming that the following function $g_0(k)$ is known
>
> $$g_0(k) = \frac{u(0,k)}{u_0(0,k)}, \qquad k \in [\underline{k}, \overline{k}]. \tag{1.6}$$

From [2], we have that the forward problem has a unique solution; and from [1], we have that $u(x,k) \neq 0$ for all $x > x_0$ and for all $k > 0$; and also from [1], we have that $g_0(k) \neq 0$ for all $k \in [\underline{k}, \overline{k}]$.

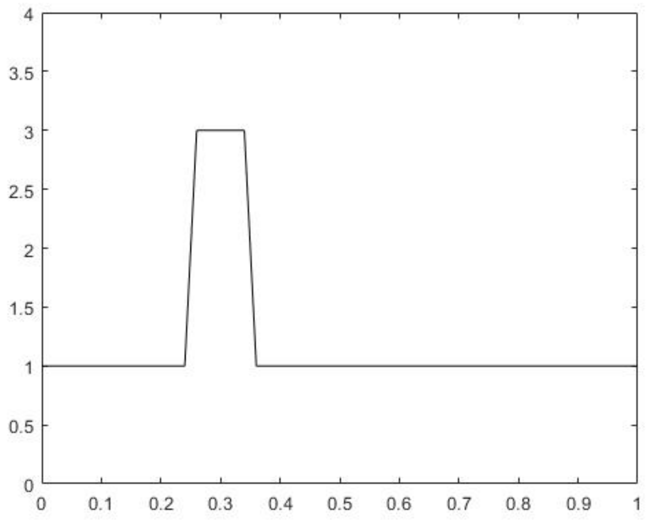


Figure 1.1: The step function $c(x)$ representing our buried explosive and the dirt surrounding it. Our bomb is in the inclusion where $c(x) = 3$.

Our motivation for finding $c(x)$ and thus solving our inverse problem is finding buried bombs. In our problem, $c(x)$ is a step function that models a buried bomb. For example, consider Figure 1.1. Sand has one dielectric constant while a bomb will have a greater dielectric constant. The areas where $c(x) = 1$ represent sand while the region where $c(x) > 1$ represents our bomb. And, as we can see from equation (1.6) above, we are supposed to find $c(x)$ using only the information that can be gleaned at $x = 0$, which is the surface of the ground. And the information $g_0(k)$ in equation (1.6) is supposed to be acquired using ground penetrating radar. Figure 1.2 is a schematic of how one is supposed to use ground penetrating radar to find $g_0(k)$. Radar works with microwaves, and microwaves are a form of electromagnetic radiation. If the incident electric field wave has only one nonzero component, this calls for the use of the Helmholtz equation above in equation 1.3, as noted in [3, 4], to model the propagation of the electric field through the sand.

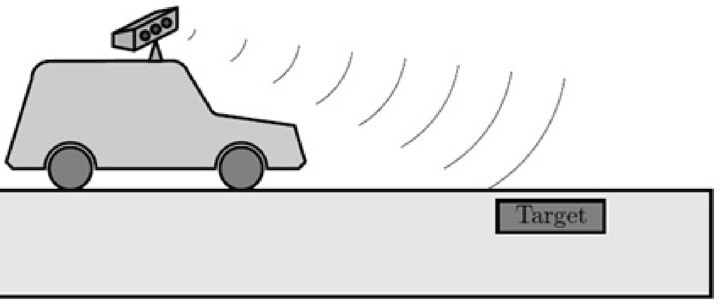


Figure 1.2: A schematic diagram from [5] showing how one is supposed to use radar to find a buried bomb.

Our problem is known as a coefficient inverse problem, abbreviated in the literature as CIP. While other (unsuccessful) ways to solve CIPs have been attempted, like using least squares according to [6], one way that works is to turn the CIP into a minimization problem that involves minimizing a strictly convex functional. We have, according to [2], that "CIPs are both highly nonlinear and ill-posed." And the reason that [7] gives as to why CIPs are ill-posed is that they have unstable solutions. When using Tikhonov functionals, our CIP being nonlinear and ill-posed will, according to [2], give us many local minima, like what we see in Figure 1.3. We are looking for a unique minimizer of our functional. If we have many local minima, we may confuse a local minima we have found with the global minimum we want. And with so many local minima, there is also the possibility that even the global minimum may not work as well as one of the local minima. This said, we want to use a strictly convex functional because such functionals have only one minimizer, and thus we will not have to deal with many local minima. From [1], this turning our inverse problem into a problem where we minimize a strictly convex functional is what is known as convexification.

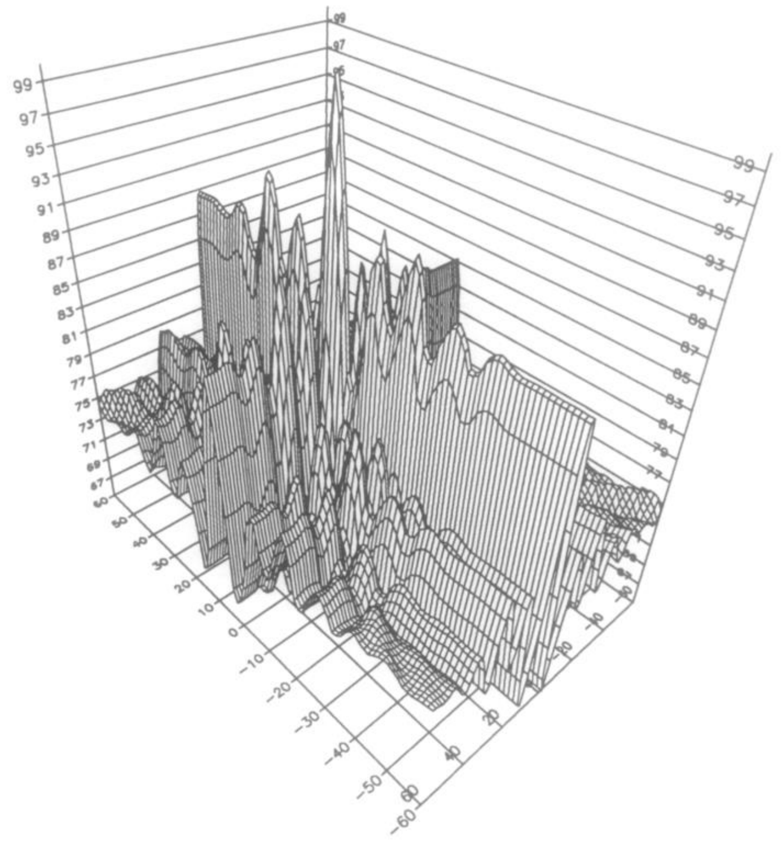


Figure 1.3: Why we want to use convexification rather than Tikhonov functionals. Note all of the local maxima and minima. This figure comes from [8].

To minimize our functional, we are after what is known as a globally convergent method. A globally convergent method, as defined in [2], is one which guarantees us that we will find a minimizer within a sufficiently small neighborhood of the true minimizer without any advanced knowledge of what this neighborhood is. Indeed, since we are trying to find a way to locate buried explosives, we need a way to minimize our functional without any knowledge of a neighborhood of where our minimizer is. Convexification is a globally convergent method according to [1].

There have been other methods to solve our inverse problem with convexification. One method is the quasi-reversibility method, outlined in [2]; and another method is the tail functions method, outlined in [5]. In this thesis, we discuss a new method for solving our inverse problem that involves finding the coefficients of a truncated

Fourier series. This method was outlined in [9]. In brief, we first want to find an orthonormal basis using the Gram-Schmidt process; then we want to obtain a series of differential equations; then we want to find a minimizer for a certain functional, a minimizer that is also a solution of one the differential equations we have obtained; and finally, having our minimizer, we want to use it to find $c(x)$.

Chapter 2 of this thesis discusses more deeply the theory behind our new method and the differences between our new method and past methods, Chapter 3 discusses a numerical experiment that tests our new method, and Chapter 4 discusses the results and conclusion of our numerical experiment.

CHAPTER 2: THEORY

The method of [9] is a general method for solving inverse problems by convexification. Here, we apply the method of [9] to our inverse problem.

As outlined in [9], the first step in our new method is to construct an orthonormal basis in $L^2(\underline{k}, \overline{k})$ from $\{k^n e^k\}_{n=0}^{\infty}$. In order to do this, though, we need to start off by finding an orthonormal basis in $L^2(0,1)$ from $\{k^n e^k\}_{n=0}^{\infty}$, just like the way [9] describes; then we abstract to having our orthonormal basis in $L^2(\underline{k}, \overline{k})$.

According to [9], what we want is a basis where the first derivative with respect to k for any element in our basis is not identically zero; and moreover, this first derivative with respect to k should be a linear combination of a finite number of elements from our basis. According to [9], a basis based on trigonometric functions or on orthonormal polynomials does not fit the bill. For a basis based on orthonormal polynomials, the first element in that basis is going to be a constant c_0. The first derivative of c_0 will obviously be identically zero. With trigonometric functions, if for $n \in \mathbb{N} \cup \{0\}$ we base our orthonormal basis on terms like $\cos \pi n k$, then the first term in our orthonormal basis will be a constant; and we also get constants as the first term in our basis if we base our basis on terms like $\sin \pi n k$ and $\exp i \pi n k$ for $n \in \mathbb{N} \cup \{0\}$. So [9] tells us that we want a basis that is similar to a basis based on Laguerre functions but works for $L^2(0,1)$ rather than $L^2(0,\infty)$.

To get our basis, we first use the Gram-Schmidt process on $\{k^n e^k\}_{n=0}^{\infty}$ and then normalize what vectors we get using the $L^2(0,1)$ norm. The vectors that result are of the form $P_m(k)e^k$, where $P_m(k)$ is a polynomial of degree m. For reasons which will later be explained in Chapter 3 of this thesis, we only need the first three vectors in our orthonormal basis. These vectors get very lengthy very quickly when we express them with exact coefficients, and using such cumbersome formulae in a computer program would be impractical. So, we will be giving them using approximate coefficients. Thus, our truncated orthonormal basis for $L^2(0,1)$ is

$$\{0.55949e^k, (-1.39851 + 2.1302k)e^k, (1.99714 - 9.25581k + 8.15893k^2)e^k\}.$$

Let us denote the mth vector in our orthonormal basis in $L^2(0,1)$ as $\phi_m(k)$.

To make the orthonormal basis we have just computed suitable for $L^2(\underline{k}, \overline{k})$, we have to shift our vectors by the amount $\underline{k}$. So, making the substitution $x = (k - \underline{k})/(\overline{k} - \underline{k})$ and realizing we have

$$\int_0^1 \phi_m(x)\phi_n(x)dx = \delta_{mn}, \tag{2.1}$$

where δ_{mn} is our Kronecker delta, we thus have

$$\frac{dx}{dk} = \frac{1}{\overline{k} - \underline{k}} \tag{2.2}$$

so that

$$\frac{1}{\overline{k} - \underline{k}} \int_{\underline{k}}^{\overline{k}} \phi_m\left(\frac{k - \underline{k}}{\overline{k} - \underline{k}}\right) \phi_n\left(\frac{k - \underline{k}}{\overline{k} - \underline{k}}\right) dk = \delta_{mn}. \tag{2.3}$$

Thus, what we end up with are orthonormal vectors ψ_m for $L^2(\underline{k}, \overline{k})$ of the form

$$\psi_m(k) = \frac{1}{\sqrt{\overline{k} - \underline{k}}}\phi_m\left(\frac{k - \underline{k}}{\overline{k} - \underline{k}}\right). \tag{2.4}$$

From [1] we have

Theorem 1. *We have*

$$a_{m,n} = \langle \psi_n', \psi_m \rangle_{L^2(\underline{k},\overline{k})} = \begin{cases} (\overline{k} - \underline{k})^{-1} & \textit{if } n = m, \\ 0 & \textit{if } n < m. \end{cases} \tag{2.5}$$

For an integer $N > 1$, let $M_N = (a_{m,n})_{(m,n)=(0,0)}^{(N-1,N-1)}$ be an $N \times N$ matrix. Then $\det M_N = (\overline{k} - \underline{k})^N \neq 0$, *which means that M_N is invertible.*

Theorem 1, as found in [1], is an adaptation of Theorem 2.4 of [9]; and the proof of Theorem 2.4 of [9] can be found in [9].

Remark. *Another reason why we cannot use orthonormal polynomials or trigonometric functions as the bases for our basis is that, according to [1], if we do, then the first column of our matrix M_N will be 0's going all the way down. This will keep our matrix from being invertible.*

The next thing we want to do, according to [9], is to derive a series of three boundary value problems from our inverse problem. We begin with $\beta(x) = c(x) - 1$, where

$$v_{xx} + k^2(v_x)^2 - 2ikv_x = -\beta(x). \tag{2.6}$$

From [1], equation (2.6) is the result of substituting

$$v(x,k) = \frac{\log w(x,k)}{k^2}, \tag{2.7}$$

where

$$w(x,k) = \frac{u(x,k)}{u_0(x,k)}, \tag{2.8}$$

into equation (1.3). So, corresponding to equations (3.1) and (3.2) of [9], we have

$$v_{xx} + k^2(v_x)^2 - 2ikv_x = -\beta(x), \tag{2.9}$$

$$v(0,k) = q_0(k), \qquad v_x(0,k) = q_1(k), \tag{2.10}$$

where, using the notation of [1],

$$q_0(k) = \frac{\log g_0(k)}{k^2}, \qquad q_1(k) = \frac{2i(g_0(k) - 1)}{g_0(k)k}, \tag{2.11}$$

and where $g_0(k)$ is as defined in equation (1.6). q_0 and q_1 are what they are because of equation (7)-(9) of [1]. From equations (10) and (11) of [1], we also have that

$$v_x(1,k) = 0. \tag{2.12}$$

We have that $\partial_k \beta(x) \equiv 0$. Thus, differentiating both sides of equation (2.9) with respect to k, we get what corresponds with equations (3.3) and (3.4) of [9],

$$v_{kxx} + 2k(v_x)^2 + 2k^2 v_x v_{xk} - 2iv_x - 2ikv_{kx} = 0, \tag{2.13}$$

$$v_k(0,k) = \partial_k q_0(k), \qquad v_{kx}(0,k) = \partial_k q_1(k), \qquad v_{kx}(1,k) = 0. \tag{2.14}$$

This is the first boundary value problem [9] says we should get.

The next thing that we want to do, having already obtained our orthonormal basis $\{\psi_n\}_{n=0}^{N-1}$ in the first step, is to represent $v(x,k)$ as the truncated Fourier series

$$v(x,k) = \sum_{n=0}^{N-1} v_n(x)\psi_n(k). \tag{2.15}$$

The $v_n(x)$'s, which are the inner products of our Fourier series, are unknown to us at the moment. Our mission is to find these $v_n(x)$'s so that we can compute equation (2.15) to substitute into equation (2.6) to ultimately compute $c(x)$. Substituting the Fourier series of equation (2.15) into equation (2.13), we get the nonlinear equation

$$\sum_{n=0}^{N-1} v_n''(x)\psi_n'(k) + 2k\left(\sum_{n=0}^{N-1} v_n'(x)\psi_n(k)\right)^2 + 2k^2\left(\sum_{n=0}^{N-1} v_n'(x)\psi_n(k)\right)\left(\sum_{n=0}^{N-1} v_n'(x)\psi_n'(k)\right) - 2i\sum_{n=0}^{N-1} v_n'(x)\psi_n(k) - 2ik\sum_{n=0}^{N-1} v_n'(x)\psi_n'(k) = 0. \tag{2.16}$$

By rearranging equation (2.16), then multiplying by $\psi_m(k)$, then integrating with respect to k on $[\underline{k}, \overline{k}]$; we have

$$\begin{aligned}\sum_{n=0}^{N-1} v_n''(x) \left\langle \psi_n'(k), \psi_m(k) \right\rangle_{L^2(\underline{k},\overline{k})} &= -\Bigg\langle 2k \left(\sum_{n=0}^{N-1} v_n'(x)\psi_n(k) \right)^2 + 2k^2 \left(\sum_{n=0}^{N-1} v_n'(x)\psi_n(k) \right) \\ &\times \left(\sum_{n=0}^{N-1} v_n'(x)\psi_n'(k) \right) - 2i \sum_{n=0}^{N-1} v_n'(x)\psi_n(k) \\ &- 2ik \sum_{n=0}^{N-1} v_n'(x)\psi_n'(k), \psi_m(k) \Bigg\rangle_{L^2(\underline{k},\overline{k})} \\ &= -\left\langle 2k(v_x)^2 + 2k^2 v_x v_{xk} - 2iv_x - 2ikv_{kx}, \psi_m(k) \right\rangle_{L^2(\underline{k},\overline{k})}. \end{aligned} \tag{2.17}$$

The right-hand side of equation (2.17) defines the mth component of a vector we shall denote as $F(x, V')$.

Let

$$V(x) = \begin{bmatrix} v_0(x) \\ \vdots \\ v_{N-1}(x) \end{bmatrix} \tag{2.18}$$

be the Fourier coefficients from equation (2.15). Consider the diagonal matrix M_N of Theorem 1. With M_N, equation (2.17) simplifies to $M_N V''(x) = F(x, V')$. Thus, we have our second boundary value problem

$$V'' - M_N^{-1} F(x, V') = 0, \tag{2.19}$$

$$V(0) = V_0, \qquad V'(0) = V_1, \qquad V'(1) = 0. \tag{2.20}$$

V_0 and V_1 can be derived from the boundary conditions of equations (2.14) above. Note that our second boundary value problem is not equivalent to our first boundary value problem because we are now working with a truncated Fourier series. Our second boundary value problem is also not exact for this reason. And [1] says that our second boundary value problem has at most one solution.

Finally, [9] says to assume that there is a vector function $p \in C^2[0, 1]$ such that

$p(0) = p_0$ and $p'(0) = p_1$, where $p_0 = V_0$ and $p_1 = V_1$ as in equation (2.20). We can construct such a function by letting, as in [1], $\chi \in C^2[0,1]$ with

$$\chi(x) = \begin{cases} 1 & \text{if } x \in [0, 1/2], \\ 0 & \text{if } x \in [3/4, 1], \\ \in (0,1) & \text{if } x \in (1/2, 3/4). \end{cases} \tag{2.21}$$

Thus, we have the N-D function

$$p(x) = [V_0 + xV_1]\chi(x) \tag{2.22}$$

that meets the criteria of [9] above, with

$$p \in C^2[0,1], \qquad p(0) = V_0, \qquad p'(0) = V_1, \qquad p'(1) = 0. \tag{2.23}$$

Letting

$$W(x) = V(x) - p(x), \tag{2.24}$$

our second boundary value problem outlined by equations (2.19) and (2.20) turns into our third boundary value problem

$$L(x, p, W) = W'' + p'' - M_N^{-1} F(x, W' + p') = 0, \tag{2.25}$$

$$W(0) = W'(0) = W'(1) = 0. \tag{2.26}$$

In Step 3, we want to turn our last boundary value problem into a minimization problem. Let $H_0^2([0,1]) = \{W \in H^2([0,1]) \mid W(0) = W'(0) = W'(1) = 0\}$ and let $B_0(R) = \{W \in H^2([0,1]) \mid W(0) = W'(0) = W'(1) = 0, \|W\|_{H^2(0,1)} < R\}$, where $R > 0$. Naturally, $B_0(R) \subseteq H_0^2([0,1])$. From [9], the functional we should have and want to minimize on $\overline{B_0(R)}$ is

$$\Phi_{\lambda,\gamma}(W) = e^{2\lambda} \int_0^1 |L(x, p, W)|^2 e^{-2\lambda x} dx + \gamma \|W\|_{H^2(0,1)}^2, \tag{2.27}$$

where $\gamma \in (0,1)$ is the regularization parameter for our functional, where the term

$e^{-2\lambda x}$ is our Carleman weight function, and where $\lambda > 1$. (In our numerical experiment, $\gamma \in [0,1)$ and $1 \leq \lambda \leq 8$.) From [1, 9], the term $e^{2\lambda}$ is used to balance the two terms of the right-hand side of equation (2.27).

In our discussion of the theory, we will use $\Phi_{\lambda,\gamma}$ and its gradient $\Phi'_{\lambda,\gamma}$ with the gradient projection method to show that $\Phi_{\lambda,\gamma}$ can be minimized. However, in our experiment, since we do not have to start at a zero-point in $H_0^2([0,1])$, we use the conjugate gradient method to find the solution to our second boundary value problem (2.19)-(2.20). Namely, we want to find the minimizer of

$$J_{\lambda,\gamma}(V) = e^{2\lambda} \int_0^1 |V'' - M_N^{-1} F(x, V')|^2 e^{-2\lambda x} dx + \gamma \left\| V \right\|^2_{H^2(0,1)}, \tag{2.28}$$

on the set $\overline{B(R, V_0, V_1)}$, where $B(R, V_0, V_1) = \{V \in H^2([0,1]) \mid V(0) = V_0, V'(0) = V_1, V'(1) = 0, \|V - p\|_{H^2(0,1)} < R\}$, where $\gamma \in [0,1)$ and $1 \leq \lambda \leq 8$ as before. But conjugate gradient method or gradient projection method, we will show that we can solve our problem either way. If the gradient projection method can be used to solve our problem, then so can the conjugate gradient method.

Naturally, when one takes a measurement of anything, there will be some level of noise. Let $\delta \in (0,1)$ be the level of noise in our data. Drawing our discussion from [1], let V^* be the exact solution of our problem (2.19)-(2.20). Let

$$V^*(0) = V_0^*, \qquad {V^*}'(0) = V_1^* \tag{2.29}$$

as in (2.20) with $V^* \in B(R, V_0, V_1)$. Let

$$|V_0 - V_0^*| < \delta, \qquad |V_1 - V_1^*| < \delta. \tag{2.30}$$

Like with equation 2.22, let

$$p^*(x) = [V_0^* + xV_1^*]\chi(x). \tag{2.31}$$

Thus,

$$\|V - V^*\|_{C^2[0,1]} \le B\delta, \tag{2.32}$$

where $B = B(\chi) > 0$ depends on $\chi(x)$.

In [9] are listed a number of theorems, which [1] restates for our problem at hand, concerning our functional $\Phi_{\lambda,\gamma}$ and gradient $\Phi'_{\lambda,\gamma}$ and for our other functional $J_{\lambda,\gamma}$ and its gradient $J'_{\lambda,\gamma}$. Here, we restate without proof and almost verbatim the theorems, propositions, and corollaries found in [1]. The proofs of our theorems can be found in [1]. As in [1], $C_1 = C_1(R, F, N) > 0$ and $C_2 = C_2(R, F, N, \chi, V^*) > 0$ are different numbers that depend on the listed parameters.

Proposition 1. *Let $p(x)$ be defined by equation (2.22). Then $V - p \in B_0(R)$ if $V \in B(R, V_0, V_1)$, and $W + p \in B(R, V_0, V_1)$ if $W \in B_0(R)$.*

Theorem 2. *The functional $J_{\lambda,\gamma}(V)$ has a gradient $J'_{\lambda,\gamma}(V)$ for every $V \in B(2R, V_0, V_1)$. Also, there exists a number $\lambda_1 = \lambda_1(R, F, N) > 1$ depending only on the listed parameters such that for all $\lambda \ge \lambda_1$ the functional $J_{\lambda,\gamma}(V)$ is strictly convex on $\overline{B(R, V_0, V_1)}$; that is, for all $V^{(1)}, V^{(2)} \in \overline{B(R, V_0, V_1)}$, we have*

$$J_{\lambda,\gamma}(V^{(2)}) - J_{\lambda,\gamma}(V^{(1)}) - J'_{\lambda,\gamma}(V^{(1)})(V^{(2)} - V^{(1)}) \ge C_1 \left\|V^{(2)} - V^{(1)}\right\|^2_{H^2(0,1)}. \tag{2.33}$$

Corollary 1. *Consider the functional*

$$\Phi_{\lambda,\gamma}(W) = J_{\lambda,\gamma}(W + p), \qquad \forall p \in \overline{B_0(R)}. \tag{2.34}$$

Then a direct analog of Theorem 2 is valid for this functional. More precisely, the functional $\Phi_{\lambda,\gamma}(W)$ has the gradient $\Phi'_{\lambda,\gamma}(W)$ at each point $W \in B_0(2R)$. Let $\lambda_1 = \lambda_1(R, F, N) > 1$ be the number of Theorem 2. Then there exists a number $\lambda_2 = \lambda_2(R, F, N, \chi, p^*) \ge \lambda_1$ depending only on the listed parameters such that for all $\lambda \ge \lambda_2$ the functional $\Phi_{\lambda,\gamma}(W)$ is strictly convex on the set $\overline{B_0(R)}$; that is, for all $W^{(1)}, W^{(2)} \in$

$\overline{B_0(R)}$

$$\Phi_{\lambda,\gamma}(W^{(2)})-\Phi_{\lambda,\gamma}(W^{(1)})-\Phi'_{\lambda,\gamma}(W^{(1)})(W^{(2)}-W^{(2)}) \geq C_2 \left\|W^{(2)} - W^{(1)}\right\|^2_{H^2(0,1)}. \quad (2.35)$$

Theorem 3. *The gradients $J'_{\lambda,\gamma}(V)$ and $\Phi'_{\lambda,\gamma}(W)$ of both functionals $J_{\lambda,\gamma}$ and $\Phi_{\lambda,\gamma}$ are Lipschitz continuous on $B(2R, V_0, V_1)$ and $B_0(2R)$, respectively. In other words, there exists a constant $D = D(R, F, \lambda, \gamma) > 0$ depending only on the listed parameters such that for all $V^{(1)}, V^{(2)} \in B(2R, V_0, V_1)$*

$$\left\|J'_{\lambda,\gamma}(V^{(1)}) - J'_{\lambda,\gamma}(V^{(2)})\right\|_{H^2(0,1)} \leq D \left\|V^{(2)} - V^{(1)}\right\|_{H^2(0,1)} \quad (2.36)$$

and similarly for $\Phi'_{\lambda,\gamma}(W)$.

Theorem 4. *Let $\lambda_1 = \lambda_1(R, F, N) > 1$ and $\lambda_2 = \lambda_2(R, F, N, \chi, V^*) \geq \lambda_1$ be the numbers of Theorem 2 and Corollary 1, respectively. Let $p(x)$ be the function defined by 2.22. Then, for any $\lambda \geq \lambda_2$ and for any $\gamma \in (0, 1)$, there exists a unique minimizer V_{min} of the functional $J_{\lambda,\gamma}(V)$ on the set $\overline{B(R, V_0, V_1)}$. In addition, for these values of λ and γ, there exists unique minimizer W_{min} of the functional $\Phi_{\lambda,\gamma}(W)$ on the set $\overline{B_0(R)}$. Furthermore, $W_{min} = V_{min} - p$ and*

$$\Phi'_{\lambda,\gamma}(W_{min,\lambda,\gamma})(W_{min,\lambda,\gamma} - W) \leq 0, \qquad \forall W \in \overline{B_0(R)}, \quad (2.37)$$

$$J'_{\lambda,\gamma}(V_{min,\lambda,\gamma})(V_{min,\lambda,\gamma} - V) \leq 0, \qquad \forall V \in \overline{B(R, V_0, V_1)}. \quad (2.38)$$

Theorem 5. *Assume that the exact solution V^* of the problem (2.19)-(2.20) exists and $V^* \in B(R, V_0^*, V_1^*)$. Also, assume that (2.29) and (2.30) hold. Let $W^* = V^* - p^* \in B_0(R)$. In addition, assume that there exists the exact solution $c^*(x)$ of our CIP and this function satisfies the conditions (1.1) and (1.2). Suppose that the function $c^*(x)$ can be found from the vector function $V^*(x)$ from the formulae (2.15) and (2.6) and the definition $c(x) = \beta(x) + 1$. Let $\lambda_1 = \lambda_1(R, F, N) > 1$ and $\lambda_2 = \lambda_2(R, F, N, \chi, V^*) \geq \lambda_1$ be the numbers of Theorem 2 and Corollary 1, respectively. Consider the number δ_0 such that $\delta_0 \in (0, e^{-4\lambda_2})$. For any $\delta \in (0, \delta_0)$ we set*

$\lambda = \lambda(\delta) = \ln(\delta^{-1/4}) > \lambda_2$ and $\gamma = \gamma(\delta) = \sqrt{\delta}$. Let $V_{min,\lambda,\gamma} \in B(R, V_0, V_1)$ and $W_{min,\lambda,\gamma} = V_{min,\lambda,\gamma} - p_{min,\lambda,\gamma} \in B_0(R)$ be the unique minimizers of the functionals $J_{\lambda,\gamma}(V)$ and $\Phi_{\lambda,\gamma}(W)$ on $B(R, V_0, V_1)$ and $B_0(R)$, respectively. Then the following accuracy estimates hold:

$$\|W_{min,\lambda,\gamma} - W^*\|_{H^2(0,1)} \leq C_2\delta^{1/4}, \tag{2.39}$$

$$\left\|V_{min,\lambda(\delta),\gamma(\delta)} - V^*\right\|_{H^2(0,1)} \leq C_2\delta^{1/4}, \tag{2.40}$$

$$\left\|c_{min,\lambda(\delta),\gamma(\delta)} - c^*\right\|_{L_2(0,1)} \leq C_2\delta^{1/4}, \tag{2.41}$$

where the function $c_{min,\lambda(\delta),\gamma(\delta)}$ is found from $V_{min,\lambda(\delta),\gamma(\delta)}$ with equations (2.15), (2.6), and the definition $c(x) = \beta(x) + 1$.

Let $P_{\overline{B_0}} : H_0^2(0,1) \to \overline{B_0(R)}$ be the projection operator from $H_0^2(0,1)$ to $\overline{B_0(R)}$. Let W_0 be our initial guess for our gradient projection method. Let $\zeta > 0$ be a number and define a recursive sequence $\{W_n\}$ with

$$W_n = P_{\overline{B_0}}(W_{n-1} - \zeta\Phi'_{\lambda,\gamma}(W_{n-1})). \tag{2.42}$$

Thus, we have

Theorem 6. *Let the conditions of Theorem 5 about the exact solutions V^* and c^* hold. Let the numbers $\lambda_1, \lambda_2, \delta, \lambda(\delta), \alpha(\delta)$ be the same as in Theorem 5. Let an arbitrary point $W_0 \in B_0(R)$ be the starting point of the gradient projection method of equation (2.42), and let $V_n = W_n + p$. Then there exists a number $\xi_0 = \xi_0(R, F, N, \chi, V^*, \delta) \in (0,1)$ depending only on listed parameters such that for any $\xi \in (0, \xi_0)$ there exists a number $q = q(\xi) \in (0,1)$ such that the following convergence estimates are valid:*

$$\left\|W_n - W_{min,\lambda(\delta),\gamma(\delta)}\right\|_{H^2(0,1)} \leq q^n \left\|W_0 - W_{min,\lambda(\delta),\gamma(\delta)}\right\|_{H^2(0,1)}, \tag{2.43}$$

$$\left\|V_n - V_{min,\lambda(\delta),\gamma(\delta)}\right\|_{H^2(0,1)} \leq q^n \left\|V_0 - V_{min,\lambda(\delta),\gamma(\delta)}\right\|_{H^2(0,1)}, \tag{2.44}$$

$$\|V_n - V^*\|_{H^2(0,1)} \leq C_2\delta^{1/4} + q^n \left\|V_0 - V_{min,\lambda(\delta),\gamma(\delta)}\right\|_{H^2(0,1)}, \tag{2.45}$$

$$\|c_n - c^*\|_{L^2(0,1)} \leq C_2\delta^{1/4} + q^n \left\|V_0 - V_{min,\lambda(\delta),\gamma(\delta)}\right\|_{H^2(0,1)}, \tag{2.46}$$

where functions $c_n(x)$ are found from the V_n's.

There are two advantages to using the gradient projection method and the conjugate gradient method: First, we can begin with any $W_0, V_0 \in H_0^2([0,1])$ and end up with $W_{\min} \in B_0(R)$ and $V_{\min} \in B(R, V_0, V_1)$; and secondly, we do not have to know the radius R in advance.

Then comes Step 4, where, having our minimizer, we use it to compute equation (2.15) and substitute our results into equation (2.6) to have $\beta(x)$ with which we compute our $c(x)$.

CHAPTER 3: NUMERICAL EXPERIMENT

Before we can begin discussing the details of our numerical experiment testing the method of [9], we have to address the number of ψ_n's needed for our experiment and how we determine this number. Determining how many ψ_n's we need is a small numerical experiment in its own right. We can figure out how many ψ_n's we need by first defining the step function $c(x)$ according to the rule

$$c(x) = \begin{cases} c_{\text{coeff}} & x \in [x_{\text{loc}} - d/2, x_{\text{loc}} + d/2], \\ 1 & \text{elsewhere,} \end{cases} \tag{3.1}$$

where the dielectric constant of the bomb we want to detect is $c_{\text{coeff}} \in [3, 6]$, where the location of the center of the bomb $x_{\text{loc}} \in [0.1, 0.4]$, and where $d = 0.1$ is the width of our bomb. Then, for this $c(x)$, we solve the forward problem by solving the 1-D Lippmann-Schwinger equation (which, according to [1] is equivalent to the forward problem)

$$u(x,k) = \frac{\exp\left(-ik|x - x_0|\right)}{2ik} + k^2 \int_0^1 \frac{\exp(-ik|x-\xi|)}{2ik}(c(\xi) - 1)u(\xi, k)d\xi, \tag{3.2}$$

where $x_0 = -1$. Having $u_0(x,k)$ as defined by equation (1.5), we compute $w(x,k)$ defined by equation (2.8) and then we compute $v(x,k)$ as defined by equation (2.7). We want to express equation (2.7) as the Fourier series of equation (2.15). Having $v(x,k)$ from equation (2.7) and all of the ψ_n's we could possibly need at our disposal; we just need to find the $v_n(x)$'s, which, for our purposes at this moment, we can treat as $L^2([\underline{k}, \overline{k}])$ inner products. In fact, here we are going to calculate the $v_n(x)$'s as

$$v_n(x) = \langle v(x,k), \psi_n(k) \rangle_{L^2([\underline{k},\overline{k}])} . \tag{3.3}$$

Then, for a given value of N, we substitute equation (2.15) into equation (2.6); and, having $\beta(x)$, we compute $c_{\text{approx}}(x) = \text{Re}[\beta(x)] + 1$. This short experiment is a qualitative experiment. If $c_{\text{approx}}(x)$ looks enough like $c(x)$, then we have enough ψ_n's to use in our main experiment. From [1] and from our work, we have determined that we need the first three or four ψ_n's. After the first four ψ_n's, our approximations start getting too tall. Figure 3.1 illustrates the graphs of our first three ψ_n's and Figure 3.2 illustrates what approximations we get for $c(x)$ with $N = 1, \ldots, 5$.

And now we discuss our main experiment, our experiment testing the method of [9]. Our general goal with this experiment was to see how good the method of [9] was at solving our problem. In our experiment, we used a MATLAB script that was a translation of a Python script written by Dr. Aleksandr Kolesov.

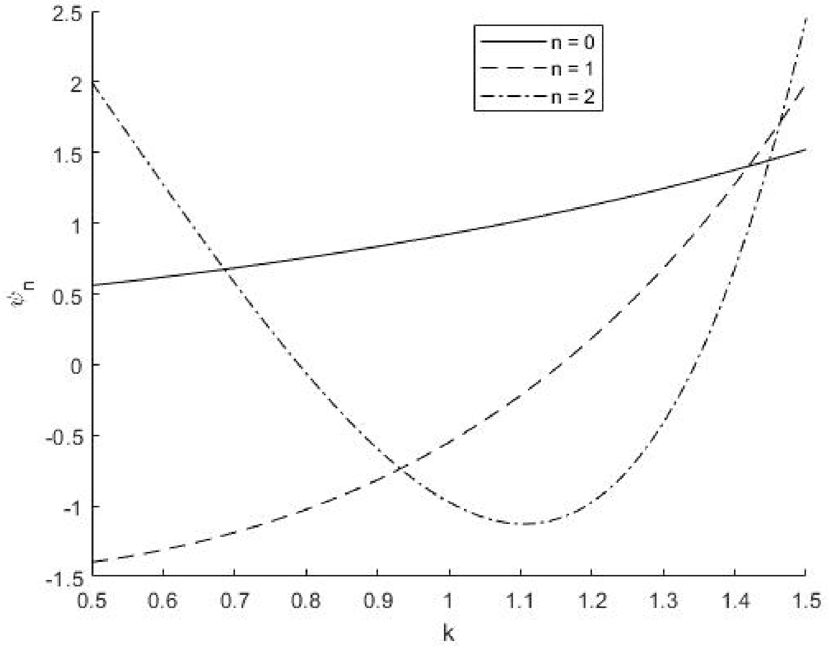


Figure 3.1: The graphs of ψ_0, ψ_1, and ψ_2.

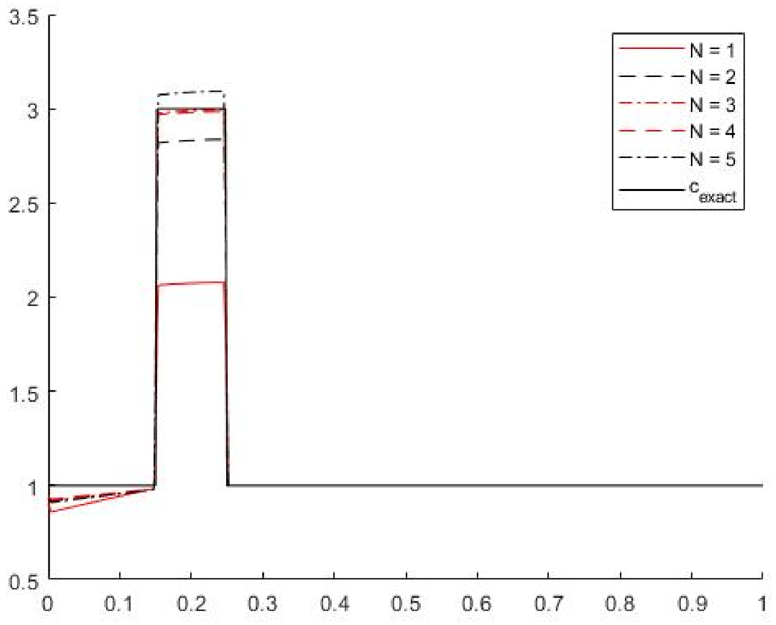


Figure 3.2: Our approximations of $c(x)$ computed with $N = 1, \ldots, 5$. The approximations with $N = 3$ and $N = 4$ very closely resemble $c(x)$ while the approximation with $N = 5$ is just a little too tall. Here, $c_{\text{coeff}} = 3$, $x_{\text{loc}} = 0.2$, $k = 0.5$, and the stepsize for both x and k was 0.002.

At the beginning of our script, the most fundamental variables are defined. First, we say we want the first three orthonormal vectors ψ_n. We then define $\underline{k} = 0.5$ and $\overline{k} = 1.5$ and we have $N_k = 3$ values of k altogether with $h_k = 0.5$ being the increment between each value of k: $k_0 = \underline{k} = 0.5 < \cdots < 1.5 = \overline{k} = k_{N_k-1}$. We have $N_x = 51$ values of x from 0 to 1 in increments of $h_x = 0.02$. And at this point we define λ and γ. Since we want to test our method with multiple values of λ and γ, we loop λ and γ with for-loops for eight different values of λ from 1 to 8 in increments of 1 and ten different values of γ with values from 0 to 0.9 in increments of 0.1.

The next thing we want to do is to calculate the values of our orthonormal vectors ψ_n and their derivatives which we will define here as φ_n (not to be confused with ϕ_n of the previous chapter). This is done using the Gram-Schmidt process. And with our discrete values of $\psi_n(k_m)$ and $\varphi_j(k_i)$, with $0 \leq i, j, m, n \leq N_k - 1 = N - 1$, we get the square $N_k \times N$ matrices

$$\Psi = \begin{bmatrix} \psi_0(k_0) & \psi_1(k_0) & \psi_2(k_0) \\ \psi_0(k_1) & \psi_1(k_1) & \psi_2(k_1) \\ \psi_0(k_2) & \psi_1(k_2) & \psi_2(k_2) \end{bmatrix} \qquad \text{and} \qquad \Phi = \begin{bmatrix} \varphi_0(k_0) & \varphi_1(k_0) & \varphi_2(k_0) \\ \varphi_0(k_1) & \varphi_1(k_1) & \varphi_2(k_1) \\ \varphi_0(k_2) & \varphi_1(k_2) & \varphi_2(k_2) \end{bmatrix}.$$

And at this point we also calculate Ψ^T. Ψ^T and Φ will be used to calculate M_N.

At this point in the program, we define the function $c(x)$ that we want to approximate on $[0, 1]$ according to equation (3.1). In our program, c_{coeff} is looped for the values 3, 4, 5, and 6 while x_{loc} is looped for the values 0.1, 0.2, 0.3, and 0.4. This gives us a total of sixteen $c(x)$'s that we want to approximate.

Having our $c(x)$, we solve the forward problem by solving the Lippmann-Schwinger equation (3.2). We solve the Lippmann-Schwinger equation for u at $x = 0$ and at the aforementioned points $k_0, ..., k_{N_k-1}$. After we have our u, we then calculate $g_0(k)$ defined by equation (1.6); and having $g_0(k)$, we then calculate $q_0(k)$ as defined in the boundary conditions (2.11). We would like to have $v_{\text{exact}}(x, k)$ as defined by equation (2.7) so that we can make a comparison between the v_{exact} we wish to emulate and the v_{approx} we will have. This said, we also solve the Lippmann-Schwinger equation at the points $k_0, ..., k_{N_k-1}$ for all N_x points $x \in [0, 1]$.

The numerical experiment of [1] (which is almost just like the experiment done in this thesis) used noise; that is, $\delta = 0.05$ in the numerical experiment of [1]. However, in our experiment, $\delta = 0$; which means that we did not use noise. However, if we had, then this is the point where we would have factored noise into our experiment. Namely, as [1] has it, we would have had

$$g_{0,\delta}(k_m) = g_0(k_m)(1.0 + \delta\sigma(k_m)), \tag{3.4}$$

where

$$\sigma = \sigma_r(k_m) + i\sigma_i(k_m), \tag{3.5}$$

where σ_r and σ_i are random numbers between -1.0 and 1.0 and where $0 \leq m \leq N_k - 1$.

For us to use the conjugate gradient method, we need an initial guess. But, before we can compute our initial guess, we need to estimate x_{loc}. This is where our location estimator comes in. Before we describe how our program's location estimator works, we need to talk a little bit about the theory behind our location estimator. Drawing our discussion from [1], we let $v(x,k)$ be as in equation (2.7). Letting $s(x,k) = v_k(x,k)$, let

$$v(x,k) = -\int_k^{\overline{k}} s(x,\tau)d\tau + v(x,\overline{k}), \tag{3.6}$$

where the tail function $v(x,k)$ is unknown. Now we substitute equation (3.6) into equation (2.13) and we get an integro-differential equation with respect to $s(x,k)$. From

$$\log w(x,k) = -\frac{1}{4}\ln c(x) - ik\left(\int_{x_0}^{x}\sqrt{c(\xi)}d\xi - x + x_0\right) - \ln(2k) - i\frac{\pi}{2} + \log(1+\hat{w}(x,k)), \tag{3.7}$$

where $w(x,k)$ is as in equation (2.8) and where

$$\hat{w}(x,k) \equiv O\left(\frac{1}{k}\right), \qquad k \to \infty, \tag{3.8}$$

and from

$$\log(1+\hat{w}(x,k)) = \sum_{n=1}^{\infty}(-1)^{n-1}\frac{(\hat{w}(x,k))^n}{n}; \tag{3.9}$$

we have that

$$v(x,k) = \frac{r(x)}{k} + O\left(\frac{\ln(2k)}{k^2}\right), \qquad k \to \infty. \tag{3.10}$$

If we assume that $\overline{k} \gg 1$ is sufficiently large, and if we get rid of the term $O(\ln(2k)/k^2)$, and if we substitute $v(x,k) = r(x)/k$ into our integro-differential equation with respect to $s(x,k)$ we get from substituting equation (3.6) into equation (2.13), letting $k = \overline{k}$; we end up with $r''(x) = 0$, whose solution for $x \in (0,1)$ gives us an approximate solution of the tail function.

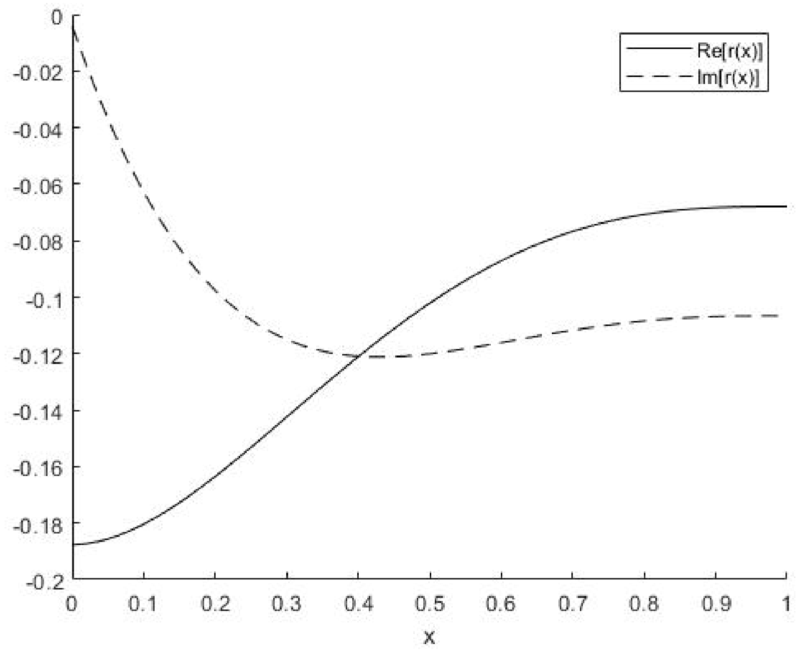


Figure 3.3: Re$[r(x)]$ and Im$[r(x)]$ plotted against x. Here, $c_{\text{coeff}} = 6$, $x_{\text{loc}} = 0.4$, $\alpha = 90$, and $x_{\text{est}} = 0.42$.

So, to estimate the location of x_{loc}, we need to solve the boundary value problem

$$r'' = 0, \qquad x \in (0,1), \tag{3.11}$$

$$r(0) = q_0(\overline{k}), \qquad r'(0) = q_1(\overline{k}) \qquad r'(1) = 0, \tag{3.12}$$

where q_0 and q_1 are defined by the boundary conditions (2.11). This ordinary differential equation outlined by (3.11) and (3.12) is solved using the quasi reversibility method that is outlined in [2]. What we do is that we minimize the functional

$$I_\alpha(r) = \frac{1}{2}\left\|r''\right\|^2_{L^2(0,1)} + \frac{1}{2}\alpha\left\|r\right\|^2_{L^2(0,1)}, \tag{3.13}$$

where α is our regularization parameter for our location estimator. In our experiment, α is looped for eleven different values from 0 to 100 in increments of 10. The reason why we need α to be so great is to keep the first term of equation (3.13) from dominating. Once we have the complex-valued $r(x)$ that minimizes (3.13), we find the $x \in (0,1)$ where $\mathrm{Im}[r(x)]$ has its minimum value and we call it x_{est}. Figure 3.3 is a graph of $\mathrm{Re}[r(x)]$ and $\mathrm{Im}[r(x)]$ against x.

If $x_{\text{est}} \leq 0.1$, we use

$$v(0,k) = \frac{\log g_0(k)}{k^2}$$

calculated with $g_0(k)$ from equation (1.6) to compute our boundary conditions to compute our initial guess. Thus, we calculate

$$v(0,k) = q_0(k), \qquad v_x(0,k) = q_1(k), \qquad v(1,k) = v(0,k) + h_x v_x(0,k). \tag{3.14}$$

And thus we define the initial guess $v_0(x,k)$ for $v(x,k)$ according to the rule

$$v_0(x,k) = \begin{cases} v(0,k) & \text{if } x = 0, \\ v(1,k) & \text{if } 0.02 \leq x \leq 1, \end{cases} \tag{3.15}$$

or equivalently

$$v_0(x,k) = \begin{cases} q_0(k) & \text{if } x = 0, \\ q_0(k) + h_x q_1(k) & \text{if } 0.02 \leq x \leq 1. \end{cases} \tag{3.16}$$

On the other hand, if $x_{\text{est}} > 0.1$, we, as [1] puts it, propagate $g_0(k)$ from $x = 0$ to $x_{\text{tar}} = x_{\text{est}} - 0.1$, thus having $g_{\text{prop}}(k)$. Having x_{tar}, we first calculate $u_0(0,k)$, then we have $D_2(k) = u_0(0,k)$, then we have $D_1(k) = u_0(0,k)(g_0(k) - 1)$, then we have $u_{\text{prop}}(x_{\text{tar}}, k) = D_1(k) \exp(ikx_{\text{tar}}) + D_2(k) \exp(-ikx_{\text{tar}})$, and finally we have

$$g_{\text{prop}}(k) = \frac{u_{\text{prop}}(x_{\text{tar}}, k)}{u_0(x_{\text{tar}}, k)}; \tag{3.17}$$

and we calculate $g_{\text{prop}}(k)$ for $k = k_0, ..., k_{N_k - 1}$. Having $g_{\text{prop}}(k)$, we compute our boundary conditions (3.14), treating $g_{\text{prop}}(k)$ like $g_0(k)$. And, having our boundary conditions, we compute an initial guess the same way just like in the case where $x_{\text{loc}} \leq 0.1$. In either case, in this program, the initial guess $v_0(x,k)$ is an $N_k \times N_x$ matrix with k-values going down and x-values going across.

Then we compute $M_N = \Psi^T \Phi$, whose inverse M_N^{-1} will be used to calculate $J'_{\lambda,\gamma}$.

Then we compute the initial value of $V_n(x)$, which would be $V_0(x)$, by having

$$V_0(x) = \Psi^T v_0(x, k). \tag{3.18}$$

And then we compute the Carleman weight function $\exp(-2\lambda x)$, which is part of our functional $J_{\lambda,\gamma}(V)$ and our gradient $J'_{\lambda,\gamma}(V)$.

Now comes the main part of our program, the conjugate gradient method. Here, we use the nonlinear conjugate gradient method. Since the nonlinear conjugate gradient method is so widely known, we will not be getting into the finer details of how it works; instead, we will discuss what things are specific to our use of the conjugate gradient method. The chief thing we would like to discuss is the gradient $J'_{\lambda,\gamma}(V)$, namely how to compute it. Similarly to what was done in [10], we use Kronecker deltas (a method outlined in [11]) to make a formula for $J'_{\lambda,\gamma}(V)$. Since $V(x)$ has both real and imaginary parts, we can decompose it as

$$V(x) = Q(x) + iP(x), \tag{3.19}$$

where $Q(x) = \mathrm{Re}[V(x)]$ and $P(x) = \mathrm{Im}[V(x)]$; and, naturally,

$$Q(x) = \begin{bmatrix} q_0(x) \\ \vdots \\ q_{N-1}(x) \end{bmatrix} \quad \text{and} \quad P(x) = \begin{bmatrix} p_0(x) \\ \vdots \\ p_{N-1}(x) \end{bmatrix}.$$

Our conjugate cogradient operator (defined in [11]) is

$$\frac{\partial}{\partial V} = \frac{1}{2} \begin{bmatrix} \frac{\partial}{\partial q_0} + i\frac{\partial}{\partial p_0} \\ \vdots \\ \frac{\partial}{\partial q_{N-1}} + i\frac{\partial}{\partial p_{N-1}} \end{bmatrix} = \frac{1}{2}\left[\frac{\partial}{\partial Q} + i\frac{\partial}{\partial P}\right]. \tag{3.20}$$

Thus,

$$\begin{aligned} J'_{\lambda,\gamma}(V) &= \frac{\partial}{\partial V} J_{\lambda,\gamma}(V) \\ &= \frac{1}{2} \begin{bmatrix} \frac{\partial}{\partial q_0} J_{\lambda,\gamma}(V) + i \frac{\partial}{\partial p_0} J_{\lambda,\gamma}(V) \\ \vdots \\ \frac{\partial}{\partial q_{N-1}} J_{\lambda,\gamma}(V) + i \frac{\partial}{\partial p_{N-1}} J_{\lambda,\gamma}(V) \end{bmatrix} \\ &= \frac{1}{2} \left[\frac{\partial}{\partial Q} J_{\lambda,\gamma}(V) + i \frac{\partial}{\partial P} J_{\lambda,\gamma}(V) \right], \end{aligned} \tag{3.21}$$

where

$$\frac{\partial}{\partial Q} J_{\lambda,\gamma}(V) = e^{2\lambda} \frac{\partial}{\partial Q} \int_0^1 |V''(x) - M_N^{-1} F(x, V')|^2 e^{-2\lambda x} dx + \gamma \frac{\partial}{\partial Q} \|V\|^2_{H^2([0,1])} \tag{3.22}$$

and

$$\frac{\partial}{\partial P} J_{\lambda,\gamma}(V) = e^{2\lambda} \frac{\partial}{\partial P} \int_0^1 |V''(x) - M_N^{-1} F(x, V')|^2 e^{-2\lambda x} dx + \gamma \frac{\partial}{\partial P} \|V\|^2_{H^2([0,1])} \tag{3.23}$$

Our gradient is a complex, vector-valued function of $x \in [0, 1]$; and, in the MATLAB script, it is an $N \times N_x$ matrix.

We will not derive all of the terms of equations (3.22) and (3.23) because it would be long and superfluous to do so. However, we will derive only the second terms of (3.22) and (3.23) just to give the reader an idea of how the conjugate cogradient operator is used to derive $J'_{\lambda,\gamma}(V)$. We have that

$$\begin{aligned}
\|V\|^2_{H^2([0,1])} &= \langle V, V\rangle_{H^2([0,1])} \\
&= \sum_{n=0}^{N-1} \int_0^1 \partial_{xx} v_n(x)\partial_{xx}\overline{v_n(x)} + \partial_x v_n(x)\partial_x\overline{v_n(x)} + v_n(x)\overline{v_n(x)}dx \\
&= \sum_{n=0}^{N-1} \int_0^1 (q_n''(x) + ip_n''(x))(q_n''(x) - ip_n''(x)) + (q_n'(x) + ip_n'(x))(q_n'(x) - ip_n'(x)) \\
&+ (q_n(x) + ip_n(x))(q_n(x) - ip_n(x))dx \\
&= \sum_{n=0}^{N-1} \int_0^1 q_n''(x)^2 + p_n''(x)^2 + q_n'(x)^2 + p_n'(x)^2 + q_n(x)^2 + p_n(x)^2 dx.
\end{aligned} \tag{3.24}$$

Thus, each component of $\frac{\partial}{\partial Q}\|V\|^2_{H^2([0,1])}$ is

$$\begin{aligned}
\frac{\partial}{\partial q_n}\|V\|^2_{H^2([0,1])}(x_m) &= \int_0^1 \frac{\partial}{\partial q_n}[q_n''(x)^2] + \frac{\partial}{\partial q_n}[q_n'(x)^2] + \frac{\partial}{\partial q_n}[q_n(x)^2]dx \\
&= 2\int_0^1 q_n''(x)\frac{\partial}{\partial q_n}q_n''(x) + q_n'(x)\frac{\partial}{\partial q_n}q_n'(x) + q_n(x)\frac{\partial}{\partial q_n}q_n(x)dx \\
&\approx 2h_x \sum_{j=2}^{N_x-1}\left[q''_{n,j}\frac{\delta_{m,j+1} - 2\delta_{m,j} + \delta_{m,j-1}}{h_x^2} + q'_{n,j}\frac{\delta_{m,j+1} - \delta_{m,j-1}}{2h_x} + q_{n,j}\delta_{m,j}\right] \\
&= 2h_x\left(\frac{q''_{n,m+1} - 2q''_{n,m} + q''_{n,m-1}}{h_x^2} - \frac{q'_{n,m+1} - q'_{n,m-1}}{2h_x} + q_{n,m}\right),
\end{aligned} \tag{3.25}$$

where

$$\frac{\partial}{\partial q_n}q_n''(x) = \frac{\delta_{m,j+1} - 2\delta_{m,j} + \delta_{m,j-1}}{h_x^2}, \qquad \frac{\partial}{\partial q_n}q_n'(x) = \frac{\delta_{m,j+1} - \delta_{m,j-1}}{2h_x}, \qquad \frac{\partial}{\partial q_n}q_n(x) = \delta_{m,j}, \tag{3.26}$$

where we have the finite differences

$$q''_{n,j} = \frac{q_{n,j+1} - 2q_{n,j} + q_{n,j-1}}{h_x^2}, \qquad q'_{n,j} = \frac{q_{n,j+1} - q_{n,j-1}}{2h_x}, \qquad q_{n,j} = q_n(x_j), \tag{3.27}$$

and where $1 \leq m \leq N_x$. Similarly,

$$\frac{\partial}{\partial p_n} \|V\|^2_{H^2([0,1])}(x_m) = 2h_x \left(\frac{p''_{n,m+1} - 2p''_{n,m} + p''_{n,m-1}}{h_x^2} - \frac{p'_{n,m+1} - p'_{n,m-1}}{2h_x} + p_{n,m} \right). \tag{3.28}$$

Having $J'_{\lambda,\gamma}(V)$, we can use the conjugate gradient method. In our experiment, we have four conditions for when to terminate the conjugate gradient method. The first is when the step size goes below a certain minimum, which is 10^{-19} in our program. then second condition is when $\left\| J'_{\lambda,\gamma}(V) \right\|$, the matrix 2-norm of $J'_{\lambda,\gamma}(V)$, dips below a certain minimum, which we have set to be 0.001 in our program. The third condition is when our program has gone through the maximum amount of iterations, which is 20,000 in this program. And finally, the fourth condition is when the gradient $J'_{\lambda,\gamma}(V)$ stops changing, which we determine to be the case whenever the step size is less than or equal to 10^{-15} and when $J_{\lambda,\gamma}(V_{n+1}) - J_{\lambda,\gamma}(V_n) \leq 10^{-20}$.

In our program, we start off with an initial step size of 10^{-5}; and for every one thousand iterations, we multiply the step size by 10. We do not use an exact line search to compute the step size. Instead, if both Wolfe conditions are not satisfied on a given iteration, the step size is divided by 10 and we go right back to the beginning of the iteration and start all over.

Once the conjugate gradient method computes for us a $V_{\min}(x)$, we then compute

$$v_{\text{approx}}(x,k) = \Psi V_{\min}(x). \tag{3.29}$$

Note that Ψ and Ψ^T are inverses. Having our $v_{\text{approx}}(x,k)$, and letting $k = k_0 = 0.5$, we substitute it into equation (2.6) and compute $\beta(x)$. (Instead of just letting $k = k_0 = 0.5$, we could loop k and have it cycle through $k_0, ..., k_{N_k-1}$. However, this is going to give us more graphs than we can comfortably study.) Now, our program will produce graphs that have some extra humps that we want to get rid of. First, we take the real part of $\beta(x)$ and we put $\beta(x) = 0$ for all $x \in [0,1]$ with $\text{Re}[\beta(x)] < 0$. Then we find the $x_{\max}$ where $\text{Re}[\beta(x)]$ has a maximum. Having our truncation factor

$\rho = 0.8$, we put $\beta(x) = 0$ for all of those $x \in [0, 1]$ for which $\mathrm{Re}[\beta(x)] < \rho\mathrm{Re}[\beta(x_{\mathrm{max}})]$. (Again, to limit the number of approximations we have to sort through, we will only consider approximations with $\rho = 0.8$.)

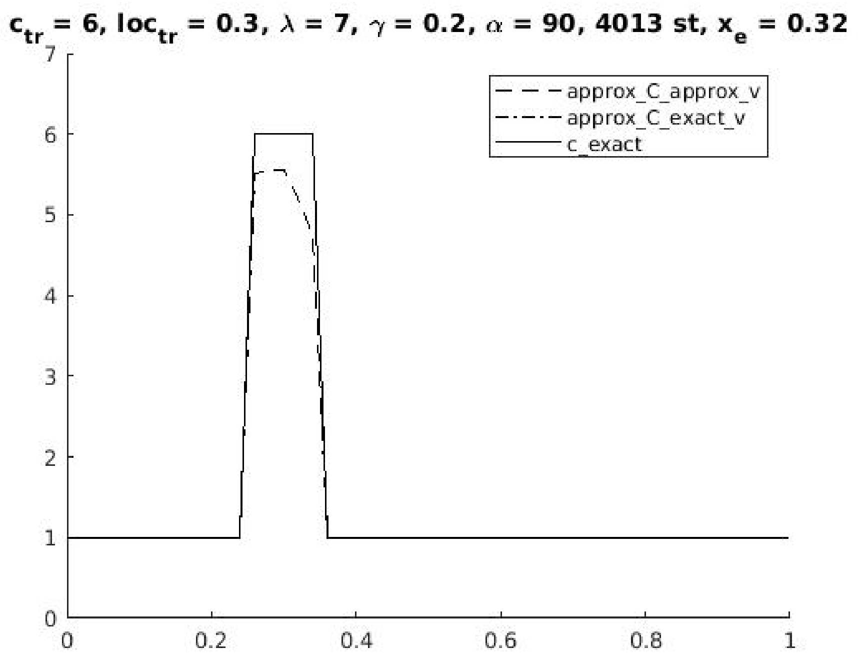
Having our $\beta(x)$, we then compute $c(x) = \mathrm{Re}[\beta(x)] + 1$. Now, if we plot $c(x)$ at this point, regardless of how we define x_{loc} in our program, the approximation seems to always center itself at $x = 0.1$. To fix this, we simply translate our approximation of $c(x)$ to the right by x_{tar} if $x_{\mathrm{est}} > 0.1$.

Finally, having a complete approximation of $c(x)$, our program gives the approximation a filename bearing all of the approximation's important information. This filename will say which $c(x)$ was supposed to be approximated, which values of λ, γ, and α were used, and how many iterations of the conjugate gradient method it took to compute the approximation. This allows us to determine which values of λ, γ, and α give the best results.

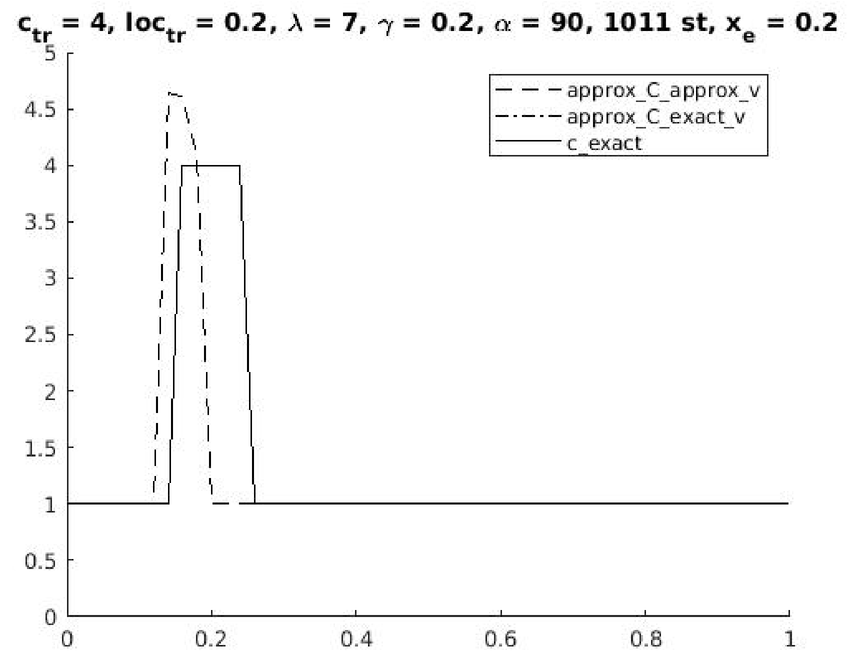
CHAPTER 4: RESULTS AND CONCLUSION

Looping through eight values of λ, ten values of γ, and eleven values of α for each of the sixteen $c(x)$'s we wanted to approximate; our program gave us 14,080 approximations, 880 approximations for each $c(x)$. (If we had also looped for k as we suggested in the previous chapter, then we would have had to sort through 42,240 approximations; and we would have had even more approximations if we would have also looped for the truncation factor ρ!) Having all of our many graphs, our duty was to sort through them all, separating the good approximations from the bad ones.

After spending a few weeks sorting through them all, we (initially) classified 4,607 graphs as being good and the other 9,473 as being bad. Judging good approximations from bad ones was a bit subjective. An approximation was classified as being a good approximation of a function $c(x)$ if the approximation looked enough like the $c(x)$; and bad approximations were ones that were too far to the left or right of where they needed to be, or were too tall, or otherwise did not bear the resemblance to $c(x)$ they should have borne. Figure 4.1 gives examples of good approximations and Figure 4.2 gives examples of bad approximations. While all sorts of values of λ and γ were found among the 4,607 good graphs, the minimum value for α among the good graphs was 40; which means that all graphs with $\alpha \leq 30$ were classified as being bad.

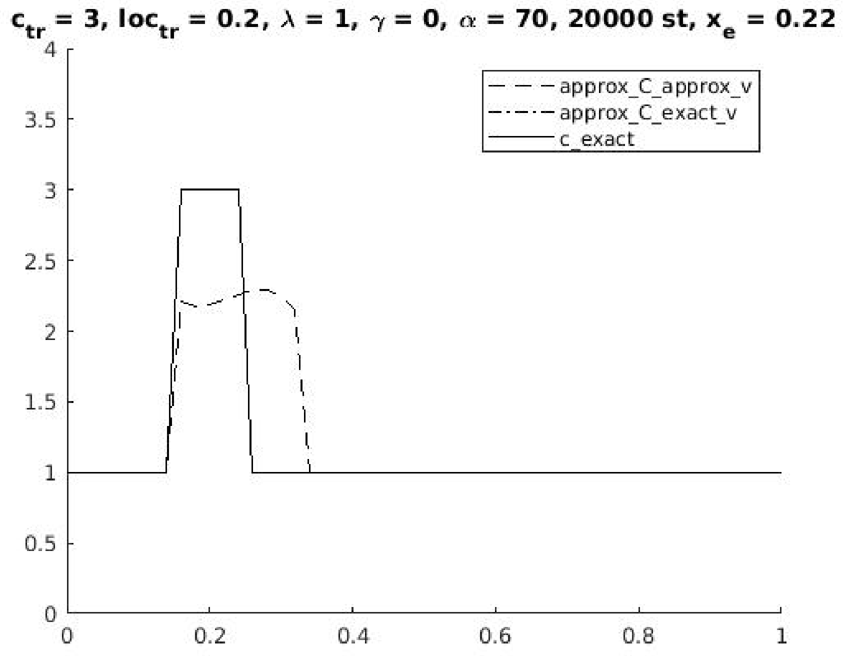


(a) This is a good because it almost totally fills up the space under $c_{\text{exact}}(x)$.

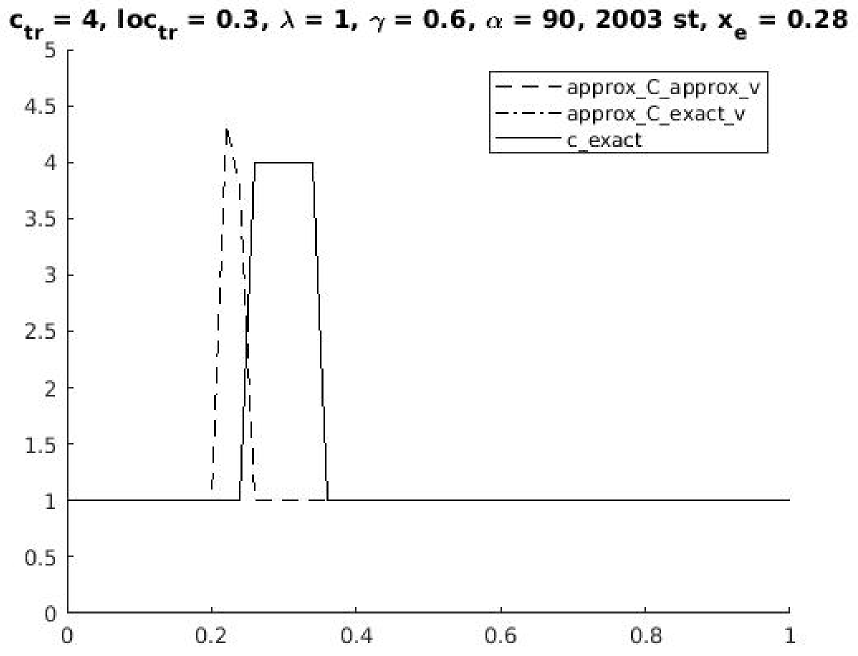


(b) Even though this one is a little tall and slightly off-center, it is still a good graph for being where it is supposed to be and for being roughly the same height as $c_{\text{exact}}(x)$.

Figure 4.1: Some examples of good approximations.



(a) This is a bad approximation for being too wide and not looking enough like the $c_{\text{exact}}(x)$ it is supposed to be approximating.



(b) This approximation is too far to the left to be a good approximation.

Figure 4.2: Some examples of bad approximations.

What we wanted was a combination of λ, γ, and α that gives us good approximations for all sixteen of our $c(x)$'s. Since for the $c(x)$ with $c_{\text{coeff}} = 6$ and $x_{\text{loc}} = 0.4$ there were no good graphs with $\alpha \leq 70$, we decided to look for our perfect combination among the good results with $\alpha \geq 80$. Since there were 2,829 approximations classified as good with $\alpha \geq 80$, we needed to write a MATLAB script that could search through the filenames of all of these graphs and find us a combination of λ, γ, and α that gives us good approximations for all sixteen of our $c(x)$'s.

Unfortunately, searching through these 2,829 approximations, our program did not find any combinations that gave good approximations for all sixteen $c(x)$'s. So we wrote another MATLAB program that searches the filenames of these 2,829 approximations, determines which combinations were used to compute these approximations, and then gives the number of approximations made by each combination. Using this program, we found sixteen combinations that were able to approximate well fifteen $c(x)$'s and fifty combinations that did a good job approximating fourteen $c(x)$'s, which amounted to sixty-six combinations. Thus, with the hope of finding a combination that gives us good approximations for all sixteen $c(x)$'s, we looked among the 116 graphs classified as being bad computed by these sixty-six combinations to see if we could find some graphs that were either looked acceptable or had been mistakenly classified as bad. This said, there were fourteen combinations we thought gave decent results. They are listed in Table 4.1.

Table 4.1: Our good combinations of λ, γ, and α. With any of these combinations, we can obtain good approximations.

λ	γ	α
2	0	90
2	0.8	80
4	0.1	90
4	0.2	90
5	0	90
7	0.1	90
7	0.2	90
7	0.3	90
7	0.4	90
7	0.6	80
8	0.2	90
8	0.3	90
8	0.4	90
8	0.5	90

Thus, from Table 4.1, we can conclude that our program, and thus our method, gives mostly good results when $7 \leq \lambda \leq 8$, when $0.2 \leq \gamma \leq 0.4$, and when $\alpha \approx 90$. We say that $\alpha \approx 90$ rather than $\alpha = 90$ because our program cycled through values of α in increments of 10. To find a more precise value of α, we would have to change our program so that it cycles through values of $\alpha \in [80, 100]$ in increments of 1. Figure 4.3, as an example, illustrates some members of a family of $c(x)$'s computed with $\lambda = 7$, $\gamma = 0.2$, and $\alpha = 90$.

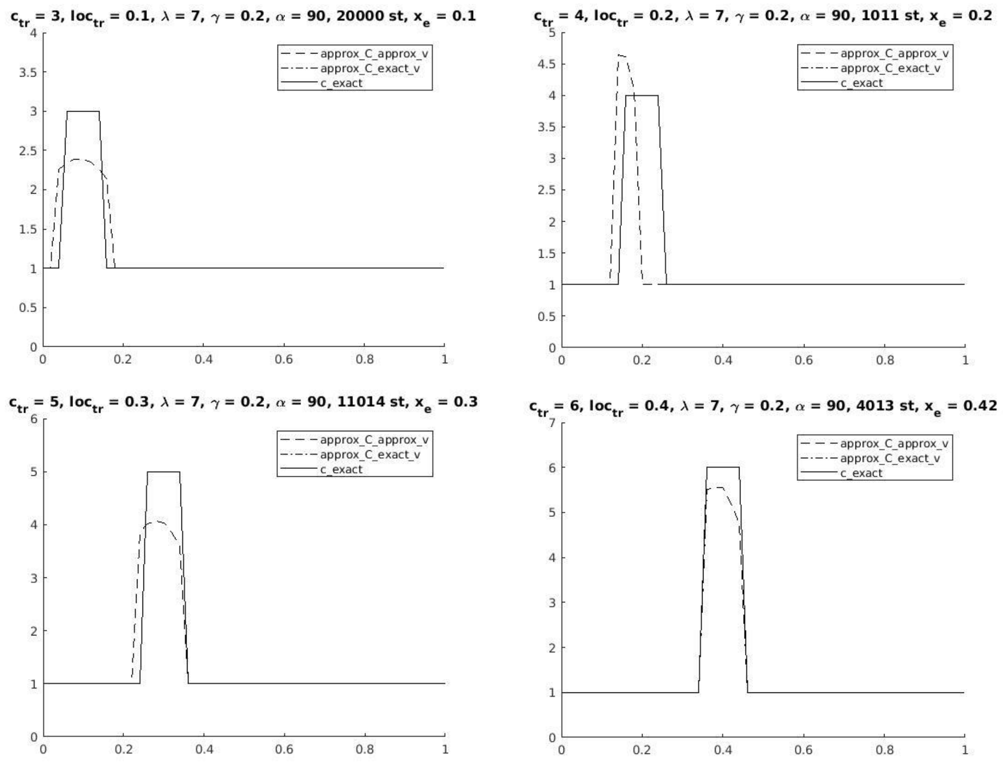


Figure 4.3: Some members of a family of $c(x)$'s computed with $\lambda = 7$, $\gamma = 0.2$, and $\alpha = 90$.

REFERENCES

[1] M. V. Klibanov, A. E. Kolesov, A. Sullivan, and L. Nguyen, “A new version of the convexicification method for a 1-d coefficient inverse problem with experimental data,” 2018.

[2] M. V. Klibanov, L. H. Nguyen, A. Sullivan, and L. Nguyen, “A globally convergent numerical method for a 1-d inverse medium problem with experimental data,” 2016.

[3] M. V. Klibanov and A. E. Kolesov, “Convexification of a 3-d coefficient inverse scattering problem,” *Computers and Mathematics with Applications*, 2018.

[4] M. V. Klibanov, D. L. Nguyen, and L. H. Nguyen, “A coefficient inverse problem with a single measurement of phaseless scattering data,” 2017.

[5] M. V. Klibanov, A. E. Kolesov, L. Nguyen, and A. Sullivan, “Globally strictly convex cost functional for a 1-d inverse medium scattering problem with experimental data,” 2017.

[6] A. B. Bakushinskii, M. V. Klibanov, and N. A. Koshev, “Carleman weight functions for a globally convergent numerical method for ill-posed cauchy problems for some quasilinear pdes,” 2016.

[7] L. Beilina and M. V. Klibanov, *Approximate global convergence and adaptivity for coefficient Inverse problems*. Springer, 2012.

[8] J. A. Scales, M. L. Smith, and T. L. Fischer, “Global optimization methods for multidimensional inverse problems,” *Journal of Computational Physics*, vol. 103, no. 2, pp. 258–268, 1991.

[9] M. V. Klibanov, “Convexification of a restricted dirichlet-to-neumann map,” *Journal of Inverse and Ill-Posed Problems*, 2017.

[10] A. V. Kuzhuget and M. V. Klibanov, “Global convergence for a 1-d inverse problem with application to imaging of land mines,” *Journal of Computational Physics*, vol. 89, no. 1, pp. 125–157, 2010.

[11] L. Sorber, M. V. Barel, and L. D. Lathauwer, “Unconstrained optimization of real functions in complex variables,” *SIAM Journal on Optimization*, vol. 22, no. 3, pp. 879–898, 2012.