6-10-2011

# Extragradient methods for elliptic inverse problems and image denoising

James Oleksyn

# Extragradient Methods for Elliptic Inverse Problems and Image Denoising

By

James Oleksyn

A thesis submitted in partial fulfillment of
the requirements for the degree of Master of Science
in Applied Mathematics
from the School of Mathematical Sciences
Rochester Institute of Technology

10 June 2011

Advisor: Dr. Akhtar Khan
Committee members: Dr. Patricia Clark
Dr. Baasansuren Jadamba
Dr. Fabio Raciti (University of Catania, Italy)

# Abstract

Numerous mathematical models in applied mathematics can be expressed as a partial differential equation involving certain coefficients. These coefficients are known and they describe some physical properties of the model. The direct problem in this context is to solve the partial differential equation. By contrast, an inverse problem asks for the identification of the variable coefficients when a certain measurement of a solution of the partial differential equation is available. One of the most commonly used approaches for solving this inverse problem is by posing a constrained minimization problem which can be written as a variational inequality.

The main contribution of this thesis is to employ various variants of extragradient methods to solve the inverse problem of parameter identification by posing it as a variational inequality. We present a thorough comparison of projected gradient method, scaled projected gradient method and several extragradient methods including the Marcotte variants, He-Goldstein type method, the projection-contraction methods proposed by Solodov and Tseng, and the hyperplane method developed by Iusem. We also test the performance of the extragradient methods for the image debluring problem.

Keywords: Inverse problems, parameter identification, regularization, projected gradient method, extragradient methods.

## Dedication

*I would like dedicate this to my family, first for their encouragement through the process, and second, for understanding that I would have to give up a lot of time with them in order to pursue my goals.*

-I love you

## Acknowledgement

I would like to thank my family, friends, and mentors for helping me through this learning process, your support was vital to my success.

I am especially grateful for my advisor, Dr. Akhtar Khan. In the two years I have been his student, he has been patient, kind, and understanding. Dr. Khan is extremely resourceful and spends much of his valuable time procuring guest speakers and other materials to better aid his students. The multiple viewpoints he provides to each problem is invaluable.

I would also like to thank the School of Mathematical Sciences at RIT for their help in ensuring my academic progress aligned with my military obligations. In particular, David Barth-Hart, Anna Fiorucci, and Carrie Koneski have been most helpful navigating me through this adventure.

Most importantly, I thank my parents for their encouragement for me to complete my degree; they can be most persuasive and motivating. Throughout college, I rarely had the opportunity to visit them for more than a few days at a time, and in a way, they have sacrificed more by having me stay in school. I love my family very much.

Special thanks to my committee members: Patricia Clark, Baasansuren Jadamba, and Fabio Raciti, and the director of graduate programs, Tamas Wiandt.

# Contents

# List of Figures

# Chapter 1

# Introduction

Inverse problems are well-studied for their applications to a wide variety of fields. In the past few decades, the development of powerful computers enabled engineers, mathematicians, and scientists to solve inverse problems computationally, leading to significant results in computer vision, medical imaging, physics and many other fields.
The scope of applications for the inverse problem has expanded to cover two main problems. These include determining past states or parameters of a physical system, and predicting the outcome of future states or parameters. Looking into past states and parameters is important in medical imaging. Solving the first type of problem enables us to locate the source of tumors because tumors are generally denser, and therefore resist pulling and pushing more than normal tissue. Studying the second problem is important in computer vision and other physical settings where we are estimating where objects are going to be at a specific time or when we want to steer the environment towards a specific outcome.
To describe the inverse problem that is the focus of this work, we consider the following one dimensional boundary value problem (BVP):

$$-\frac{d}{dx}\left(a(x)\frac{d}{dx}u(x)\right) = f(x), \qquad 0 < x < 1, \tag{1.1a}$$

$$u(0) = 0, \tag{1.1b}$$

$$u(1) = 0. \tag{1.1c}$$

The boundary conditions used here are Dirichlet boundary conditions. Other type of boundary conditions can also be used.

In the context of the above BVP, the *direct problem* is to find $u(x)$, given that $a(x)$ and $f(x)$ are known. On the other hand, our focus is on the *inverse problem* of finding the coefficient $a(x)$ when a measurement $z$ of the solution $u(x)$ is known.
In general inverse problems are more difficult to solve than the corresponding direct problems. This is partly due to the fact that they are ill-posed. To explain the meaning of the ill-posedness, we recall the notion of a well-posed problem. A problem is called *well-posed* if it possesses the following three features:

1. Existence
2. Uniqueness
3. Stability

The main issue here is the stability. A problem is stable if introducing small perturbations in the data do not lead to large perturbations in the solution. We remark that the direct problem associated to the above BVP is stable. We demonstrate this by constructing a simple example. We choose the following data for the above BVP:

$$\begin{aligned} a(x) &= 4x + 1 \\ f(x) &= 16x - 2. \end{aligned}$$

We solve the BVP by using the finite element approach (see Chapter 2 for details) by discretizing the problem using 100 uniformly spaced points on the interior of $[0, 1]$. The top left picture in Figure 1.1 shows the exact coefficient, the corresponding exact solution for $u(x)$ is shown in the top right. Then we add uniformly random noise to the coefficient (bottom left) and solve the direct problem using the finite element method (bottom right).
The figure shows the well-posedness of the direct problem. The system appears stable because moderate perturbations in the coefficient have little effect on the solution.
We remark that the inverse problem of finding the coefficient $a(x)$ cannot be solved directly by manipulating the BVP. To show some of the associated obstacles, we convert the BVP in the following form:

$$a(x) = -\frac{1}{\frac{du}{dx}} \int_{y=0}^{x} f(y)dy, \quad 0 < x < 1. \tag{1.2}$$

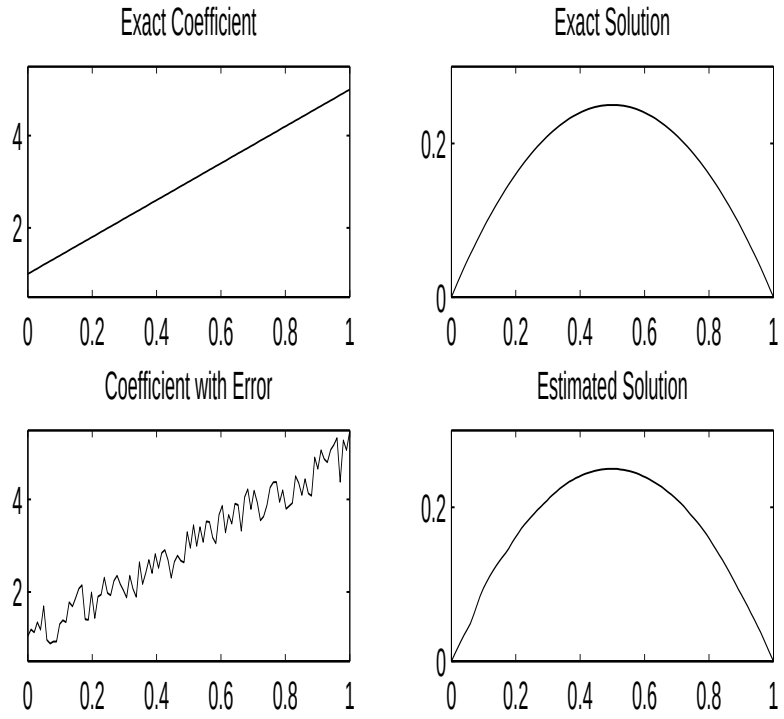


Figure 1.1: Stability of the Direct Problem

Notice that the coefficient is not even defined in the regions where $\frac{du}{dx} = 0$. Now consider solving equation (1.2) for the coefficient using the measurement $\tilde{u}(x)$ as the solution:

$$\tilde{u}(x) = u(x) + \epsilon \sin\left(\frac{x}{\epsilon^2}\right), \tag{1.3a}$$

$$\frac{d\tilde{u}}{dx} = \frac{du}{dx} + \frac{\cos\left(\frac{x}{\epsilon^2}\right)}{\epsilon}, \tag{1.3b}$$

where $\epsilon \to 0$.

In equation (1.3a), as $\epsilon \to 0$, since $\sin(x)$ is bounded, we have $\tilde{u}(x) \to u(x)$. However, in equation (1.3b), for sufficiently small $\epsilon$, there exists a smaller $\epsilon$ that will increase the magnitude of $d\tilde{u}/dx$ (the derivative is not bounded). Thus using equation (1.2) with the measurement $\tilde{u}(x)$, forces $a(x) \to 0$ as $\epsilon \to 0$ hence the problem is not stable for small perturbations in $u(x)$.

A commonly studied approach for solving inverse problems is to pose an equivalent optimization problem whose solution is an approximation of the

coefficient to be identified. The idea is to minimize the difference between the computed solution and the measured solution using some suitable norm. In other words, given a measurement $z$, the coefficient $a(x)$ should be chosen to minimize the output least squares (OLS) functional:

$$J_1(a) = \frac{1}{2}\|u(a) - z\|^2, \tag{1.4}$$

where $\|\cdot\|$ is some suitable norm and $u(a)$ is the solution obtained by solving the direct problem for the coefficient $a(x)$.
We remark that the above minimization problem is a constrained optimization problem where the BVP is the (implicit) constraint.
Since the BVP is not solvable for all $a(x)$, we also need to impose constraints on $a(x)$. We define the set of feasible coefficients by $\tilde{A}$, where $\tilde{A}$ is a closed and convex set.
Finally, to cope with the ill-posedness of the inverse problem at hand, we need to regularize the above minimization problem. Regularization consists of adding a suitable functional to the OLS functional. The regularization method penalizes numerical features that are not natural to our applications. Generally, these are high frequencies in the estimated coefficient to include rapidly changing derivative and non-smoothness. The regularization term is of the form:

$$R_\alpha(a) = \frac{\alpha}{2}\|a\|^2, \tag{1.5}$$

where $\alpha > 0$ and $\|\cdot\|$ is a suitable norm.
Besides the OLS, we will also employ the modified output least squares (MOLS) functional:

$$J_2(a) = \frac{1}{2}\int_0^1 a(x)\left[\frac{d}{dx}\left(u(a) - z\right)\right]^2 dx. \tag{1.6}$$

Combining all the ingredients discussed above, we have the following constrained optimization problem

$$\min_{a\in\tilde{A}} J(a) + R_\alpha(a), \tag{1.7}$$

where $J(a)$ is either $J_1(a)$ or $J_2(a)$.

In this work, we will solve the inverse problem by writing the above minimization problem as a variational inequality of finding $a^* \in \tilde{A}$ such that

$$\langle J'(a^*), a - a^* \rangle \geq 0 \qquad \forall\ a \in \tilde{A}, \tag{1.8}$$

where $J'$ is the derivative of $J$. We remark that the above variational inequality is a necessary optimality condition for the minimization problem. However, the variational inequality turns out to be necessary as well as sufficient optimality condition if the functional $J$ is convex.

This work is devoted to the gradient based methods. We employ variants of projected gradient methods and extragradient methods to solve the inverse problem of parameter identification by posing it as a variational inequality. We implement numerous algorithms and present a thorough comparison of projected gradient method, scaled projected gradient method and several extragradient methods including the Marcotte variants, He-Goldstein type method, the projection-contraction methods proposed by Solodov and Tseng, and the hyperplane method developed by Iusem. We also test the performance of the extragradient methods for the image debluring problem. During the last two decades numerous researches have focused on extragradient type methods. However, to the best of our knowledge this is the first instance when these methods have been thoroughly compared in the context of some applied problems.

Computation of the gradient for the objective functionals is a challenging task for the inverse problems. In this work, we will use the Adjoint-Stiffness method to calculate the gradient of the objective functionals.

# Chapter 2

# Direct Problem

In this chapter, we implement the finite element method (FEM) to solve the direct problem with Dirichlet boundary conditions.
The finite element method is based on three central ideas. First we formulate the boundary value problem into its variational or weak form. Then we discretize the weak form into a linear system of equations. However, as we increase the number of nodes, the amount of required storage greatly increases. Hence, we need to choose the basis functions in a manner that forces the matrix corresponding to our linear system of equations to be sparse.

## 2.1 Variational Form

Before we reformulate the problem, we need to introduce a few tools and conventions that will be needed. Throughout the paper, we will only consider the domain $\Omega = (0, 1)$. We define the linear space:

$$V := \{v : v \in H^1(\Omega)|\ v(0) = v(1) = 0\}.$$

We recall that we are dealing with the BVP:

$$\begin{aligned} -\frac{d}{dx}\left(a(x)\frac{d}{dx}u(x)\right) &= f(x) \qquad 0 < x < 1, && \text{(2.1a)} \\ u(0) &= 0, && \text{(2.1b)} \\ u(1) &= 0. && \text{(2.1c)} \end{aligned}$$

We will formulate the BVP into its variational form. Let us multiply each side of equation (2.1) by an arbitrary test function $v \in V$ and integrating each side over $(0,1)$ with respect to $x$. Then

$$-\int_0^1 \frac{d}{dx}\left(a(x)\frac{du}{dx}v(x)\right)dx = \int_0^1 f(x)v(x)dx \quad \forall v \in V \tag{2.2}$$

We modify the left hand side of equation (2.2) using integration by parts

$$\begin{aligned} -\int_0^1 \frac{d}{dx}\left(a(x)\frac{du}{dx}v(x)\right)dx &= -a(x)\frac{du}{dx}v|_{v=v(0)}^{v=v(1)} + \int_0^1 a(x)\frac{du}{dx}\frac{dv}{dx} \\ &= (0-0) + \int_0^1 a(x)\frac{du}{dx}\frac{dv}{dx}. \end{aligned}$$

Consequently, the variational form in an abstract form reads as:

$$T(a,u,v) = m(v) \qquad \forall\ u,v \in V, \tag{2.3}$$

where

$$T(a,u,v) = \int_0^1 a(x)\frac{du}{dx}\frac{dv}{dx}\,dx \qquad \forall\ u,v \in V, \tag{2.4a}$$

$$m(v) = \int_0^1 f(x)v(x)\,dx \qquad \forall\ v \in V. \tag{2.4b}$$

It is easy to check that $T(\cdot,\cdot,\cdot)$ is a trilinear form which is symmetric in $u$ and $v$. If the coefficients are so chosen that the trilinear form is continuous and coercive, that is, there are constants $\alpha > 0$ and $\beta > 0$ such that

$$T(a,u,v) \leq \alpha\|u\|\,\|v\|, \tag{2.5}$$

$$T(a,u,u) \geq \beta\|u\|^2, \tag{2.6}$$

then the Lax-Milgram Lemma ensures that the above variational problem is uniquely solvable.

## 2.2 Finite Element Discretization

Let $V_n$ be a finite dimensional subspace of $V$. By restricting the variational form to the finite dimensional subspace, we have

$$T(a,u,v) = m(v), \qquad \forall\ v \in V_n. \tag{2.7}$$

Let $u_n$ be a solution of the above variational form.
We will next convert the above BVP into a matrix form. For this, let

$$\{\phi_1, \phi_2, \ldots, \phi_n\}$$

be a bases for $V_n$. By substituting $v = \phi_j$ for $j = 1, \ldots, n$, into the above variational form, we obtain

$$\begin{aligned} T(a, u_n, \phi_1) &= m(\phi_1) \\ T(a, u_n, \phi_2) &= m(\phi_2) \\ &\vdots \\ T(a, u_n, \phi_n) &= m(\phi_n). \end{aligned}$$

Furthermore, we also have

$$u_n = \sum_{i=1^n} U_i \phi_i.$$

Since the solution $u_n$ is not known, the problem of finding $u_n$ is equivalent to finding the coefficients $U_i,\ i = 1, 2, \ldots, n$. By combining the above equations, we have

$$KU = P,$$

where

$$\begin{aligned} K_{ij} &= T(a, \phi_j, \phi_i) \\ P_i &= m(\phi_i). \end{aligned}$$

The matrix $K$ is the so-called stiffness matrix and the vector $P$ is the load vector.
The final ingredient of the finite element method is to choose the bases so that the resulting matrix $K$ is sparse. For this, we proceed as follows: We choose $N$ equally spaced points (called nodes) on the interior of $(0, 1)$ and define the meshsize $h$ as the distance between two adjacent nodes.
We define, for a fixed mesh on $[0, 1]$,

$$V_n = \{p : [0, 1] \to \mathbb{R} |\ p \text{ is continuous and piecewise linear, } p(0) = p(1) = 0\}.$$

Let us now construct a basis for $V_n$. For $n = 1, 2, \ldots, n-1$, we define $\phi_i \in V_n$ satisfying

$$\phi_i(x_j) = \begin{cases} 1 & \text{for} \quad j = i \\ 0 & \text{for} \quad j \neq i. \end{cases}$$

It is easy to check that $\{\phi_1, \phi_2, \ldots, \phi_{n-1}\}$ forms a basis for $V_n$. Furthermore, the precise form of the above bases functions can be computed by the following formulas:

$$\phi_j(x) = \begin{cases} \frac{x - x_{j-1}}{h_j} & \text{if } x \in [x_{j-1}, x_j] \\ \frac{x_{j+1} - x}{h_{j+1}} & \text{if } x \in [x_j, x_{j+1}] \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

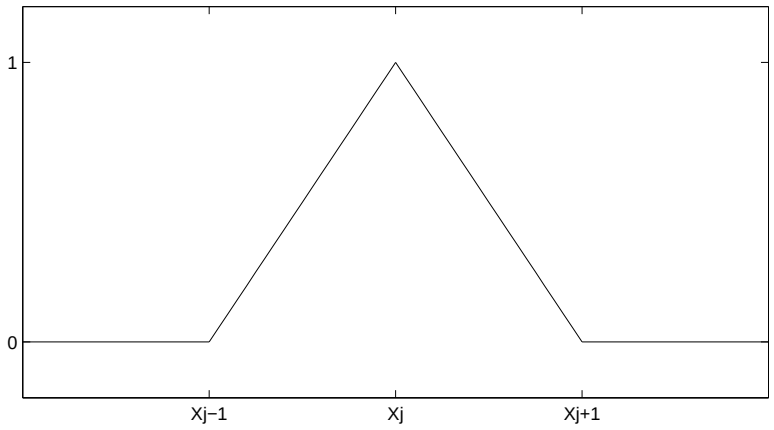


Figure 2.1: Typical Basis (Hat) Function: $\phi_j$

## 2.3 Matlab Implementation

We use the following Simpson's rule to calculate the integrals:

$$\int_b^a g(x)dx \approx \frac{b-a}{2}\left[g(a) + 4g\left(\frac{a+b}{2}\right) + g(b)\right]$$

### 2.3.1 Load Vector

The steps necessary to compute the load vector are as follows:

$$\begin{aligned}
P_j &= \frac{1}{h}\int_{x_{j-1}}^{x_j}(x-x_{j-1})f(x)dx - \frac{1}{h}\int_{x_j}^{x_{j+1}}(x-x_{j-1})f(x)dx \\
&= \frac{1}{h}\cdot\frac{h}{6}\left[(x_{j-1}-x_{j-1})f_{j-1} + 4\left[\frac{x_j+x_{j-1}}{2}-x_{j-1}\right]f\left(\frac{x_j+x_{j-1}}{2}\right) + (x_j-x_{j-1})f_j\right] \\
&- \frac{1}{h}\cdot\frac{h}{6}\left[(x_j-x_{j-1})f_j + 4\left[\frac{x_j+x_{j+1}}{2}-x_{j+1}\right]f\left(\frac{x_j+x_{j+1}}{2}\right) + (x_{j+1}-x_{j+1})f_{j+1}\right] \\
&= \frac{h}{3}\left[f\left(\frac{x_j+x_{j-1}}{2}\right) + f_j + f\left(\frac{x_j+x_{j+1}}{2}\right)\right] \\
P_j &= \frac{h}{6}\left[f_{j-1}+4f_j+f_{j+1}\right]
\end{aligned}$$

### 2.3.2 Stiffness Matrix

Due to the special basic functions the stiffness matrix $K(A)$ is sparse and tridiagonal. Indeed, for $i,j = 1,...,N$, if $|i-j| > 1$ then $K_{i,j} = 0$. We will use the convention: $a_j = a(x_j)$. For the main diagonal entries, we have

$$\begin{aligned}
K_{j,j} &= \int_{x_{j-1}}^{x_{j+1}} a(x)(\phi_j')^2 dx \\
&= \frac{1}{h^2}\int_{x_{j-1}}^{x_j} a(x)dx + \frac{1}{h^2}\int_{x_j}^{x_{j+1}} a(x)dx \\
&= \frac{1}{h^2}\cdot\frac{h}{6}\left[a_{j-1} + 4a\left(\frac{x_{j-1}+x_j}{2}\right) + a_j + a_j + 4a\left(\frac{x_j+x_{j+1}}{2}\right) + \kappa_{j+1}\right].
\end{aligned}$$

Using $a\left(\frac{a+b}{2}\right) = \left[a(a)+a(b)\right]/2$, we get:

$$K_{j,j} = \frac{1}{2h}\left[a_{j-1}+2a_j+a_{j+1}\right].$$

For the off diagonals, we have

$$\begin{aligned}
K_{j+1,j} &= \int_{x_j}^{x_{j+1}} a(x)\phi_{j+1}'\phi_j' dx \\
&= -\frac{1}{h^2}\int_{x_j}^{x_{j+1}} a(x)dx \\
&= -\frac{1}{6h}\left[a_j + 4\kappa\left(\frac{x_j+x_{j+1}}{2}\right) + a_{j+1}\right] \\
&= \frac{1}{2h}\left[a_j + a_{j+1}\right].
\end{aligned}$$

Notice that $K$ is symmetric, that is

$$K_{j+1,j} = K_{j,j+1}.$$

## 2.4 Examples

We solve the direct problem for some test examples, all examples are taken with 20 interior nodes.

**Example 1:**

$$\begin{aligned} a(x) &= 1 + 4x \\ u(x) &= x - x^2 \\ f(x) &= 16x - 2. \end{aligned}$$

**Example 2:**

$$\begin{aligned} a(x) &= e^{-5(x-x^2)^2} \\ u(x) &= e^{5(x-x^2)^2} - 1 \\ f(x) &= 10 \end{aligned}$$

**Example 3:**

$$\begin{aligned} a(x) &= \cos(2\pi x) + 1 \\ u(x) &= 2\sin(2\pi x) \\ f(x) &= 8\pi^2 \sin(2\pi x)(1 + 2\cos(2\pi x)). \end{aligned}$$

**Example 4:**

$$\begin{aligned} a(x) &= ln(1 + x) \\ u(x) &= -x^2(x - \frac{1}{2})(x - 1) \\ f(x) &= -ln(1 + x)(-12x^2 + 9x - 1) - (-4x^3 + \frac{9}{2}x^2 - x)/(1 + x). \end{aligned}$$

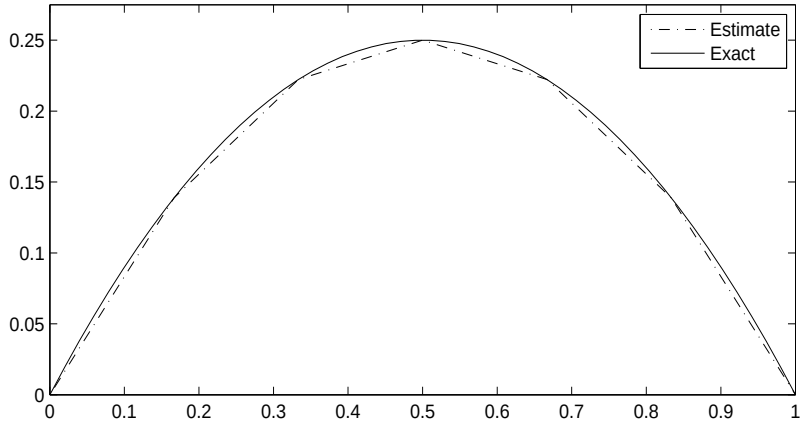


Figure 2.2: Example 1: Direct Problem

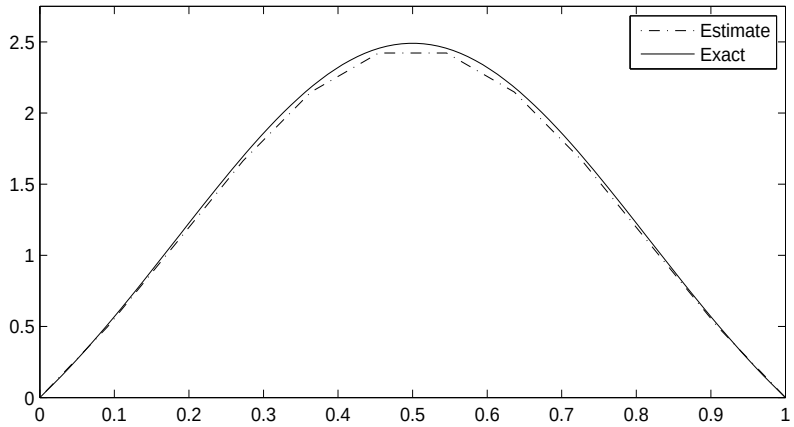


Figure 2.3: Example 2: Direct Problem

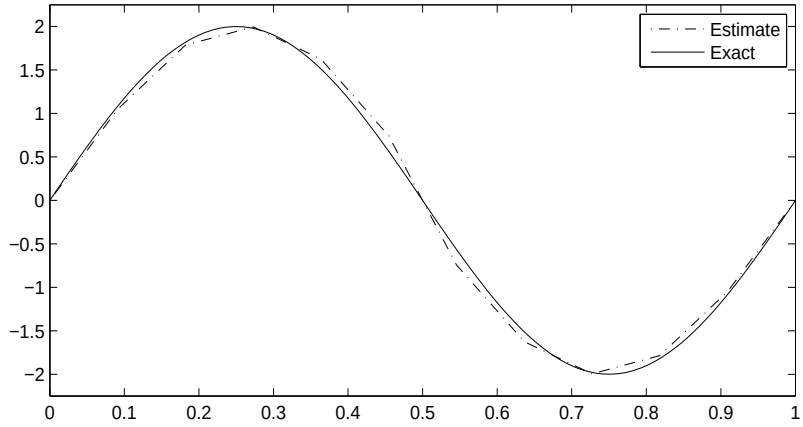


Figure 2.4: Example 3: Direct Problem

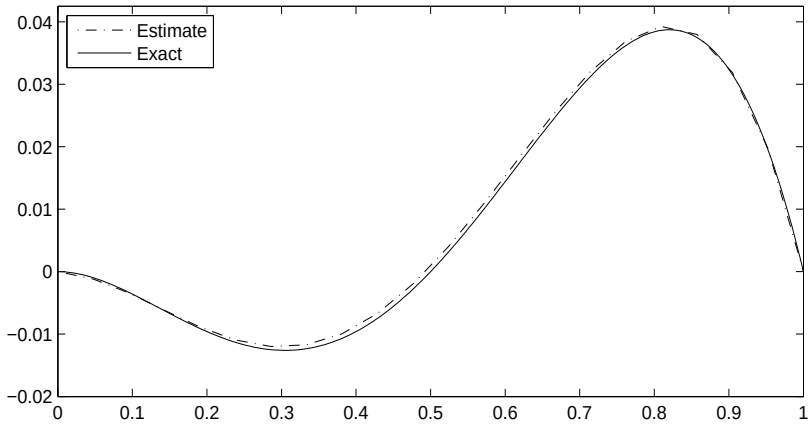


Figure 2.5: Example 4: Direct Problem

# Chapter 3

# The Inverse Problem

In this chapter we study the inverse problem. First we use finite element method to discretize the two objective functionals, then we find the gradient of each functional. We use the adjoint-stiffness approach for finding the gradient.
We assume that $f(x)$ and a measurement of $u(x)$, denoted by $z$, are known. Finding the coefficient is done by solving the following minimization problem:

$$J(a) = \min_{a \in \bar{A}} \frac{1}{2} \|u(a) - z\|^2. \tag{3.1}$$

We are given several options for the norm, we will implement output least squares, and a modified output least squares that captures characteristics of the energy functional.

## 3.1 Finite Element Discretization

### 3.1.1 Output Least Squares

Let $V_n$ be the finite dimensional subspace of the space $V$ and let $A_m$ be the finite dimensional subspaces of the coefficient space $B$. We use the same set of the basis functions.

We discretize the system using the basis functions

$$\begin{aligned} a(x) &= \sum_{i=1}^{N+2} A_i \phi_i \\ u(x) &= \sum_{j=1}^{N} U_j \phi_j \\ z(x) &= \sum_{k=1}^{N} Z_k \phi_k. \end{aligned}$$

We proceed to obtain a finite-dimensional form for the OLS objective functional. Recall that, we have

$$J(A) = \frac{1}{2}\langle u - z, u - z\rangle + \ R_\alpha(A), \tag{3.2}$$

where $R_\alpha(A)$ is a regularization parameter which we will discuss later.
We use (3.2) to get

$$\begin{aligned} \frac{1}{2}\langle u - z, u - z\rangle &= \frac{1}{2}\langle \sum_{i=1}^{N}(U_i - Z_i)\phi_i, \sum_{j=1}^{N}(U_j - Z_j)\phi_j\rangle \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(U_i - Z_i)(U_j - Z_j)\langle \phi_i, \phi_j\rangle, \end{aligned}$$

which has the matrix form:

$$J(A) = \frac{1}{2}(U - Z)^T M (U - Z) + R_\alpha(A), \tag{3.3}$$

where

$$U - Z = [U_1 - Z_1, U_2 - Z_2, ..., U_N - Z_N]^T$$

and

$$M_{i,j} = \langle \phi_i, \phi_j\rangle = \int_0^1 \phi_i \phi_j \, dx,$$

is the mass matrix.

Recall that $\langle \phi_i, \phi_j \rangle = 0$ if $|i-j| > 1$ hence M is tridiagonal. Furthermore,

$$\begin{aligned}
M_{i,i} &= \langle \phi_i, \phi_i \rangle \quad \text{for } i = 1, ..., N \\
&= \int_{x_{i-1}}^{x_{i+1}} \phi_i^2 dx \\
&= \frac{2}{h^2} \int_{x_{i-1}}^{x_i} (x - x_i)^2 dx \\
&= \frac{2}{h^2} \frac{h^3}{3} \\
&= \frac{2h}{3}. \\
M_{j,j+1} &= \langle \phi_j, \phi_{j+1} \rangle \quad \text{for } j = 1, ..., N-1 \\
&= \int_{x_j}^{x_{x+1}} \phi_j \phi_{j+1} dx \\
&= \frac{1}{h^2} \int_{x_j}^{x_{j+1}} (x_{j+1} - x)(x - x_j) dx \\
&= \frac{1}{h^2} \frac{h}{6} \left[ 0 + 4(x_{j+1} - x_{j+1/2})(x_{j+1/2} - x_j) + 0 \right] \\
&= \frac{h}{6}.
\end{aligned}$$

Consequently,

$$\begin{aligned}
M_{i,i} &= \frac{2h}{3} \qquad \text{for } i = 1, ..., N, \\
M_{j,j+1} &= \frac{h}{6} \qquad \text{for } j = 1, ..., N-1.
\end{aligned}$$

### 3.1.2 Modified Output Least Squares

Now we consider minimizing the following functional:

$$J_2(A) = \frac{1}{2} \|U(a) - Z\|_E^2 + R_\alpha(a). \tag{3.4}$$

It has the discretized form:

$$\begin{aligned}
J_2(A) &= \frac{1}{2} \langle a(x) \sum_{i=1}^{N} (u_i - z_i) \phi_i', \sum_{j=1}^{N} (u_j - z_j) \phi_j' \rangle + R_\alpha(a) \\
&= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (u_i - z_i)(u_j - z_j) \langle a(x) \phi_i', \phi_j' \rangle + R_\alpha(a) \\
&= \frac{1}{2} (U - Z)^T K(A) (U - Z) + R_\alpha(a),
\end{aligned}$$

where

$$K(A)_{i,j} = \langle a(x)\phi_i', \phi_j' \rangle.$$

We recall that, we have calculated the entries:

$$K(A)_{j,j} = \frac{1}{2h}\left[a_{j-1} + 2a_j + a_{j+1}\right]$$
$$K(A)_{j,j+1} = \frac{1}{2h}\left[a_j + a_{j+1}\right].$$

### 3.1.3 Regularization

In practice, regularization is very effective for handling inverse problems. This can be explained through convex analysis; the penalizing term $R_\alpha(A)$ makes the objective functional strictly convex, hence it has exactly one global minimum. However, if we choose $R_\alpha(A)$ poorly, then we may end up solving a completely different problem. Below, we present some good choices for $R_\alpha(A)$.

We have three choices for $R_\alpha(A)$. The L2 norm, the H1 norm, and the H1 semi-norm. The first choice is the L2 norm:

$$\begin{aligned} R_\alpha(A) &= \frac{\alpha}{2}\|a(x)\|^2 \\ &= \frac{\alpha}{2}A^T\tilde{M}A. \end{aligned}$$

However, unlike U, A is not zero at the boundary, therefore A has $n+2$ entries, which implies that this mass matrix $\tilde{M}$ has size $(n+2)\times(n+2)$. Furthermore, recall that the goal of the regularization term is to remove unnecessary features in the coefficient. Therefore, we should consider the derivative of the coefficient as a means to increase the smoothness of $A$ and thus removing unnecessary features. Hence, we present the $H_1$ norm semi-norm:

$$\begin{aligned} R_\alpha(A) &= \frac{\alpha}{2}\|a(x)'\|^2 \\ &= \frac{\alpha}{2}A^T\tilde{K}A, \end{aligned}$$

$\tilde{K}$ is a matrix of size $(n+2)\times(n+2)$. Finally, we combine the two, as the discrete $H_1$ norm:

$$R_\alpha(A) = \alpha(\|a(x)\|^2 + \|a(x)'\|^2). \tag{3.5}$$

## 3.2 Gradient of the Solution Map

### 3.2.1 The Adjoint-Stiffness Approach

We define $F : \Re^m \to \Re^n$ to be the finite element solution operator that maps a coefficient $a \in A_m$ to the approximate solution $u \in U_n$, here $m = n + 2$. Then $F(A) = U$, where U is defined by:

$$K(A)U = P. \tag{3.6}$$

Recall, $K(A) \in R^{n \times n}$ is the stiffness matrix and $P \in R^n$ is the load vector.

$$K(A)_{ij} = \int_0^1 a(x)\phi_j'\phi_i' dx \quad i, j = 1, ..., n$$
$$P_i = \int_0^1 f\phi_i \quad i = 1, ..., n.$$

This suggests another form, where we discretize $a(x)$

$$\begin{aligned} K(A)_{ij} &= \int_0^1 a(x)\phi_j'\phi_i' dx \\ &= \int_0^1 \left( \sum_{k=1}^m A_k \psi_k \right) \phi_j'\phi_i dx \\ &= \sum_{k=1}^m \left( \int_0^1 \psi_k \phi_j'\phi_i' \right) A_k \\ K(A)_{ij} &= T_{ijk} A_k, \end{aligned} \tag{3.7}$$

where

$$T_{ijk} = \int_0^1 \psi_k \phi_j'\phi_i' \quad i, j = 1, ..., n; k = 1, ..., m. \tag{3.8}$$

To arrive at the gradient formula for U, we define the adjoint-stiffness matrix $L(U)$ by the condition

$$L(U)A = K(A)U \qquad \forall\ A \in \Re^m,\ U \in \Re^n.$$

Substituting $K(A)$, we obtain

$$\begin{aligned} L(U)A &= (TA)U \\ \Rightarrow L(U) &= TU. \end{aligned}$$

Therefore

$$\begin{aligned} L(U)_{ik} &= \sum_{j=1}^{n} T_{ijk} U_j \\ &= T_{i1k} U_1 + T_{i2k} U_2 + ... + T_{ink} U_n. \end{aligned}$$

### 3.2.2 Adjoint Stiffness Matrix Computation

The size of $L(V)$ is $n \times m$. We are using the same basis functions as describe in Chapter 2. Two additional basis functions $\psi_0$ and $\psi_{n+1}$ are shown in the figure below. Recall that other basis functions where strategically chosen to

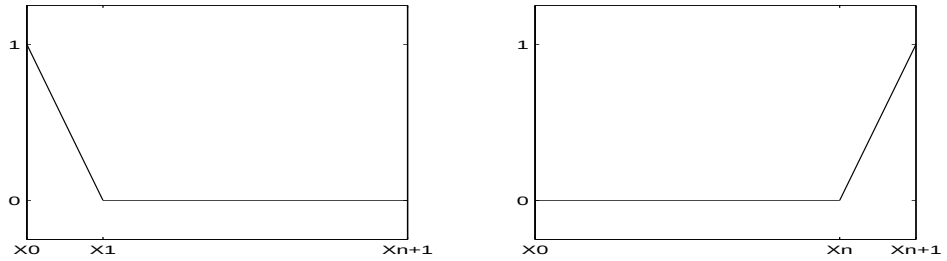


Figure 3.1: $\psi_0$ and $\psi_{n+1}$

be nonzero on exactly two adjacent subintervals. This means that the adjoint stiffness matrix will also be sparse, however, the tridiagonal structure will be shown in the main diagonal, the super diagonal, and the super-super diagonal. Considering the shape of $\psi_0$ and $\psi_{n+1}$, the first and last row of $L(U)$ will have a slightly different structure than the intermediate rows.

**Computation of Row 1:**

$$\begin{aligned} L_{1,0} &= \sum_{j=1}^{n} \int_0^1 \psi_0 \phi_j' \phi_1' u_j \\ &= \int_0^h \psi_0 \left(\phi_1'\right)^2 u_1 \\ &= \frac{h}{2} \left(\frac{1}{h}\right)^2 u_1 \\ &= \frac{1}{2h} u_1. \end{aligned}$$

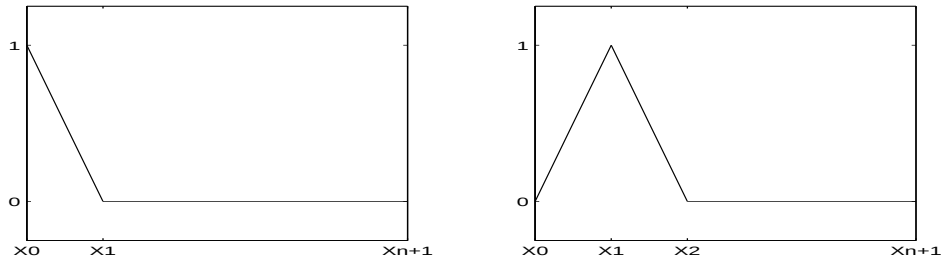


Figure 3.2: $\psi_0$ and $\psi_1$

$$\begin{aligned} L_{1,1} &= \sum_{j=1}^{n} \int_0^1 \psi_1 \phi_j' \phi_1' u_j \\ &= \int_0^1 \psi_1 \left(\phi_1'\right)^2 u_1 + \int_0^1 \psi_1 \phi_2' \phi_1' u_2 \\ &= 2 \cdot \frac{h}{2} \cdot \frac{1}{h^2} u_1 + \frac{h}{2} \cdot \frac{-1}{h^2} u_2 \\ &= \frac{1}{2h} \left[2u_1 - u_2\right]. \end{aligned}$$

$$\begin{aligned} L_{1,2} &= \sum_{j=1}^{n} \int_0^1 \psi_2 \phi_j' \phi_1' u_j \\ &= \int_0^1 \psi_2 \left(\phi_1'\right)^2 u_1 + \int_0^1 \psi_2 \phi_2' \phi_1' u_2 \\ &= \frac{u_2}{h^2} \int_h^{2h} \psi_2 + 2\left(\frac{u_2}{h^2}\right) \int_0^h \psi_2 \\ &= \frac{1}{2h} u_1 - \frac{1}{2h} u_2 \\ &= \frac{1}{2h} \left[u_1 - u_2\right]. \end{aligned}$$

**Computation of Row 2:**

$$\begin{aligned} L_{2,1} &= \sum_{j=1}^{n} \int_0^1 \psi_1 \phi_j' \phi_2' u_j \\ &= \int_0^1 \psi_1 \phi_1' \phi_2' u_1 + \int_0^1 \psi_1 \phi_2' \phi_2' u_2 \\ &= \frac{-1}{h^2} u_1 \int_0^1 \psi_1 + \frac{1}{h^2} u_2 \int_0^1 \psi_1 \\ &= \frac{-u_1}{h^2}\frac{h}{2} + \frac{u_2}{h^2}\frac{h}{2} \\ &= \frac{1}{2h}(-u_1 + u_2). \end{aligned}$$

$$\begin{aligned} L_{2,2} &= \sum_{j=1}^{n} \int_0^1 \psi_2 \phi_j' \phi_2' u_j \\ &= \int_0^1 \psi_2 \phi_1' \phi_2' u_1 + \int_0^1 \psi_2 \phi_2' \phi_2' u_2 + \int_0^1 \psi_2 \phi_3' \phi_2' u_3 \\ &= \left( \frac{-1}{h^2} u_1 + \frac{h}{h^2} u_2 + \frac{-1}{h^2} u_3 \right) \int_0^1 \psi_2 \\ &= \frac{1}{2h} \left[ -u_1 + 2u_2 - u_3 \right]. \end{aligned}$$

$$\begin{aligned} L_{2,3} &= \sum_{j=1}^{n} \int_0^1 \psi_3 \phi_j' \phi_2' u_j \\ &= 0 + \int_0^1 \psi_3 \phi_2' \phi_2' u_2 + \int_0^1 \psi_3 \phi_3' \phi_2' u_3 + 0 + ... + 0 \\ &= \frac{1}{2h} \left[ u_2 - u_3 \right]. \end{aligned}$$

**Computation of the Last Row:**

$$\begin{aligned} L_{n,n-1} &= \sum_{j=1}^{n} \int_0^1 \psi_{n-1} \phi_j' \phi_n' u_j \\ &= \int_0^1 \psi_{n-1} \phi_{n-1}' \phi_n' u_{n-1} + \int_0^1 \psi_{n-1} (\phi_n')^2 u_n \\ &= \frac{1}{2h} \left[ -u_{n-1} + u_n \right]. \end{aligned}$$

$$\begin{aligned} L_{n,n} &= \sum_{j=1}^{n} \int_0^1 \psi_n \phi_j' \phi_n' u_j \\ &= \int_0^1 \psi_n \phi_{n-1}' \phi_n' u_{n-1} + \int_0^1 \psi_n (\phi_n')^2 u_n \\ &= \frac{1}{2h} \left[ -u_{n-1} + 2u_n \right]. \end{aligned}$$

$$\begin{aligned} L_{n,n+1} &= \sum_{j=1}^{n} \int_0^1 \psi_{n+1} \phi_j' \phi_n' u_j \\ &= \int_0^1 \psi_{n+1} (\phi_n')^2 u_n \\ &= \frac{1}{2h} u_n. \end{aligned}$$

### 3.2.3 Computing the Derivative of $U(A)$

We have

$$\delta U = DF(A)\delta A. \tag{3.9}$$

To find $\delta U$, we begin by differentiating

$$K(A)U = P.$$

We find the derivative of the left hand side using chain rule, yielding

$$(DK(A)\delta A)U + K(A)\delta U = 0.$$

Now we exploit that $K$ is linear in $A$ and simplify to get

$$\begin{aligned} (K(A)\delta A)U + K(A)\delta U &= 0 \\ \Rightarrow K(A)\delta U &= -DK(A)(\delta A)U \\ \Rightarrow K(A)\delta U &= -K(\delta A)U \\ \Rightarrow \delta U &= -(K(A))^{-1} K(\delta A)U. \end{aligned}$$

### 3.2.4 Output Least Squares

Recall that

$$J_1(A) = \frac{1}{2}(U - Z)^T M (U - Z). \tag{3.10}$$

Using chain rule, we have

$$\begin{aligned} DJ_1(A)\delta A &= \frac{1}{2}(\delta U)^T M(U-Z) + \frac{1}{2}(U-Z)^T M(\delta U) \\ &= (\delta U)^T M(U-Z) \\ &= \left[-K(A)^{-1}K(\delta A)U\right]^T M(U-Z) \\ &= \left[-K(A)^{-1}L(U)\delta A\right]^T M(U-Z) \\ &= -\delta A^T L(U)^T \left(K(A)^{-1}\right)^T M(U-Z) \\ &= -\delta A^T L(U)^T K(A)^{-1} M(U-Z). \end{aligned}$$

Simplifying, we have

$$\nabla J_1(A) = -L(U)^T K(A)^{-1} M(U-Z). \tag{3.11}$$

### 3.2.5 Modified Output Least Squares

Recall that

$$J_2(A) = \frac{1}{2}(U-Z)^T K(A)(U-Z). \tag{3.12}$$

Now we differentiate (applying chain rule)

$$\begin{aligned} DJ_2(A)\delta A &= \frac{1}{2}(\delta U)^T K(A)(U-Z) + \frac{1}{2}(U-Z)^T K(A)\delta U \\ &+ \frac{1}{2}(U-Z)^T DK(A)\delta A(U-Z) \\ &= (\delta U)^T K(A)(U-Z) + \frac{1}{2}(U-Z)^T K(\delta A)(U-Z) \\ &= \left[-K(A)^{-1}K(\delta A)U\right]^T K(A)(U-Z) + \frac{1}{2}(U-Z)^T K(\delta A)(U-Z) \\ &= -(\delta A)^T L(U)^T(-K(A)^{-1})^T K(A)(U-Z) + \frac{1}{2}(U-Z)^T K(\delta A)(U-Z) \\ &= -(\delta A)^T L(U)^T(U-Z) + \frac{1}{2} - (\delta A)^T L(U-Z)^T(U-Z) \\ &= -\frac{1}{2}(\delta A)^T L(U+Z)^T(U-Z). \end{aligned}$$

Simplifying, we have

$$\nabla J_2(A) = -\frac{1}{2}L(U+Z)^T(U-Z). \tag{3.13}$$

# Chapter 4

# Gradient and Extragradient Methods

In this work, we will implement and test the numerical performance of the following iterative schemes for solving the inverse problem of parameter identification:

1. Gradient Projection Using Armijo Line Search
2. Scaled Gradient Projection Using Barzilai-Borwein Rules
3. Khobotov Extragradient Method Using Marcotte Rules (3 Variants)
4. Solodov-Tseng's Projection-Contraction Method
5. Improved He-Goldstein Type Extragradient Method
6. Hyperplane Extragradient Method

## 4.1 Test Problems

All the above methods will be tested on the following suite of test problems:
**Test Problem 1:**

$$\begin{aligned} a(x) &= 1 + 4x \\ u(x) &= x - x^2 \\ f(x) &= 16x - 2. \end{aligned}$$

**Test Problem 2:**

$$\begin{aligned} a(x) &= e^{-5(x-x^2)^2} \\ u(x) &= e^{5(x-x^2)^2} - 1 \\ f(x) &= 10. \end{aligned}$$

**Test Problem 3:**

$$\begin{aligned} a(x) &= \cos(2\pi x) + 1 \\ u(x) &= 2\sin(2\pi x) \\ f(x) &= 8\pi^2 \sin(2\pi x)(1 + 2\cos(2\pi x)). \end{aligned}$$

**Test Problem 4:**

$$\begin{aligned} a(x) &= ln(1+x) \\ u(x) &= -x^2(x - \frac{1}{2})(x-1) \\ f(x) &= -ln(1+x)(-12x^2 + 9x - 1) - (-4x^3 + \frac{9}{2}x^2 - x)/(1+x) \end{aligned}$$

For all the experiments, we will use $N = 100$.

## 4.2 Variational Inequalities

In this work, we will solve the inverse problem of identifying variable coefficients in BVP by formulating the regularized output-least squares and regularized modified output least squares functional as a variational inequality of finding $A^* \in \tilde{A}$ such that

$$\langle J'(A^*), A - A^* \rangle \geq 0, \quad \forall\ A \in \tilde{A}. \tag{4.1}$$

The above optimality condition has a unique solution if $J'$ is strongly monotone (i.e.)

$$\langle J'(x) - J'(y), x - y \rangle \geq\ l\|x - y\|^2 \ \ \forall x, y \in \tilde{A}, \tag{4.2}$$

and Lipschitz continuous:

$$\|J'(x) - J'(y)\| \leq\ L\|x - y\| \quad \forall x, y \in \tilde{A}. \tag{4.3}$$

We can convert the VI to a fixed point problem (FPP)

$$A^* = P_{\tilde{A}}(A^* - \alpha J'(A^*)),$$

where $P_{\tilde{A}}$ is the project onto $\tilde{A}$.
We recall the projection theorem in the following:
**Projection Theorem:** Let $\tilde{A}$ be a closed and convex subset of $\mathbb{R}^m$. Then for each $x \in \mathbb{R}^m$ there is a unique $A^* \in \tilde{A}$ such that:

$$\|x - A^*\| = \inf_{A \in \tilde{A}} \|x - A\|.$$

We call $A^*$ the projection of x onto $\tilde{A}$; $A^* = P_{\tilde{A}}(x)$.

## 4.3 Gradient Projection Method

The gradient projection is an iterative algorithm for the FPP

$$A^{k+1} = P_{\tilde{A}}(A^k - \alpha J'(A^k)) \tag{4.4}$$

Convergence requires that

$$\alpha \in (0, \frac{2l}{L^2}). \tag{4.5}$$

Recall that this implies that $J'$ is strongly monotone.

Note that we do not have information about $l$ and $L$ and hence need to use a method to determine the step-length. Also, since the gradient projection method assumes steepest descent convergence given boxed constraints we must introduce other conditions on $\alpha$ to ensure convergence. To avoid the convergence pitfalls of the steepest descent algorithm, we use the following condition to guarantee that the change in the gradient is not proportional to the change in the coefficient:

Find the largest possible $\alpha$ such that

$$J(A^{k+1}) - J(A^k) < -\alpha\lambda\|J'(A^k)\|^2, \tag{4.6}$$

where $\lambda \in (0, 1)$.

We use Armijo line search to backtrack (reduce $\alpha$) until the above condition.

Now we discuss the convergence properties of the gradient projection.
Recall the OLS functional:

$$J_1(a) = \frac{1}{2}\|u(a) - z\|^2.$$

It can be shown that

$$\langle J_1'(x) - J_1'(y), x - y\rangle \geq -m\|x - y\|^2 \quad \forall x, y \in \tilde{A}. \tag{4.7}$$

Since regularization term $R$ (with parameter $\epsilon$) is strongly convex, by definition, we have

$$\langle R'(x) - R'(y), x - y\rangle \geq \epsilon\|x - y\|^2 \quad \forall x, y \in \tilde{A} \tag{4.8}$$

We are guaranteed convergence when

$$-m + \epsilon = m_1 > 0 \tag{4.9}$$

For each functional, the corresponding $m$ is fixed, therefore, we require $\epsilon$ to be large to ensure convergence. $\epsilon$ is large in the sense that we are adding sufficient noise and hence are solving a different optimization problem.

To contrast the OLS functional, it can also be shown that MOLS without a regularizer is monotone. Therefore we get better results by minimizing the MOLS functional since MOLS requires a smaller $\epsilon$ than OLS. In the next few sections we will discuss how extragradient methods relax the condition of strong monotonicity which will enable us to choose a lower $\epsilon$ for each objective functional.

Gradient Projection: Example 1

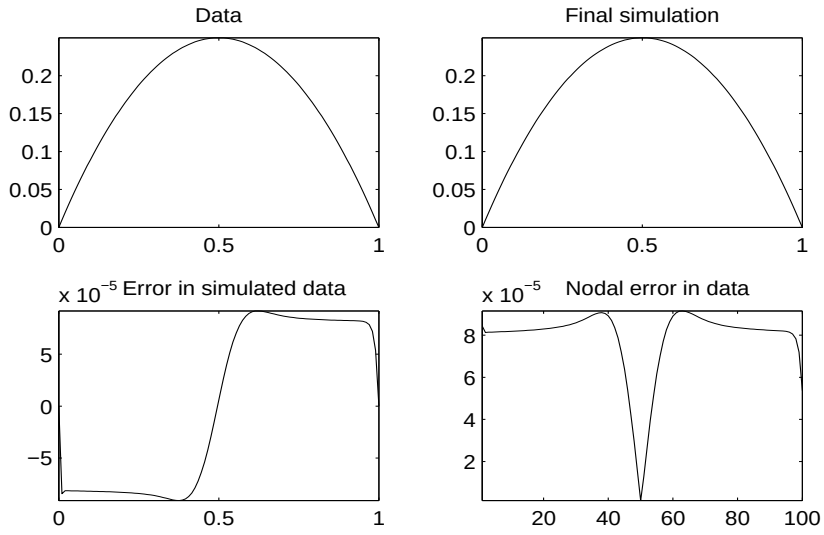


Figure 4.1: Example 1: Solution by Gradient Projection

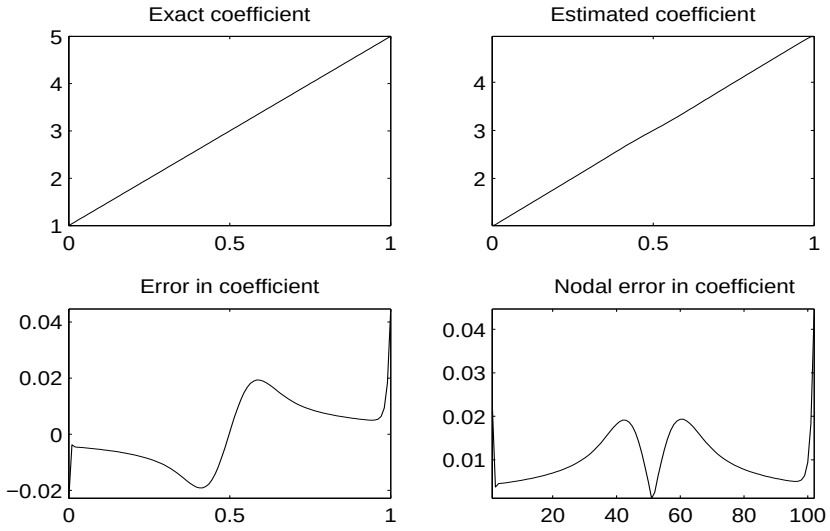


Figure 4.2: Example 1: Coefficient by Gradient Projection

Gradient Projection: Example 2

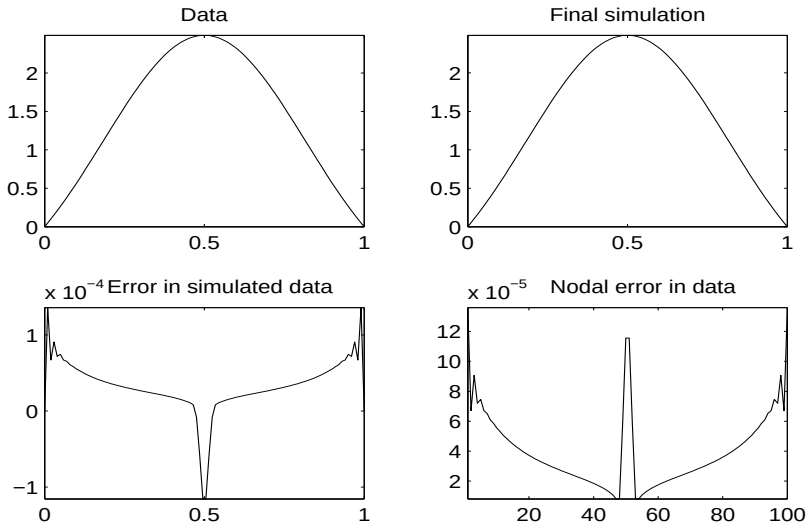


Figure 4.3: Example 2: Solution by Gradient Projection

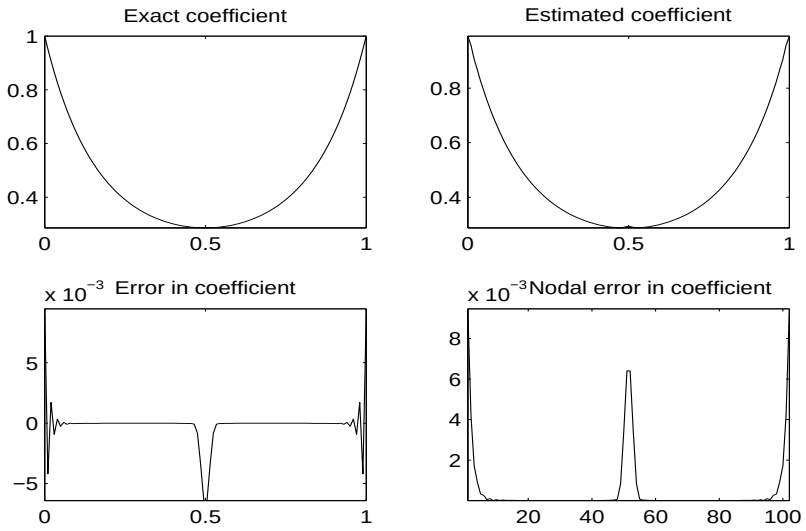


Figure 4.4: Example 2: Coefficient by Gradient Projection

Gradient Projection: Example 3

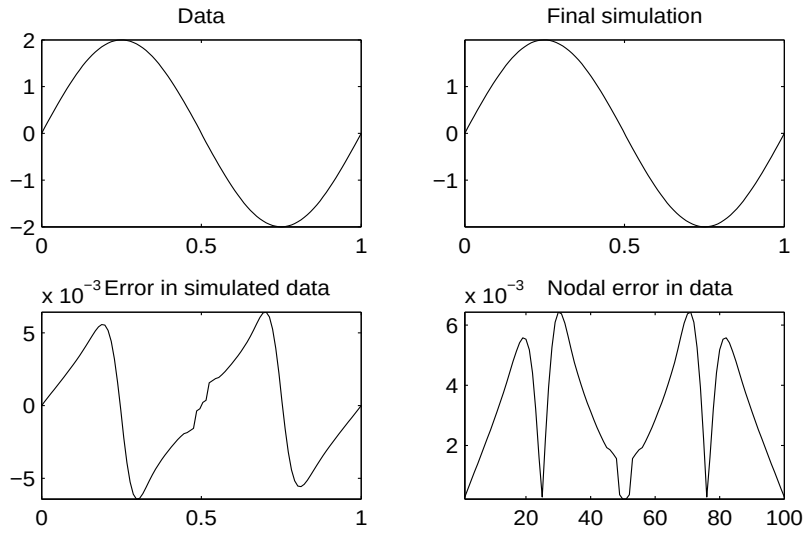


Figure 4.5: Example 3: Solution by Gradient Projection

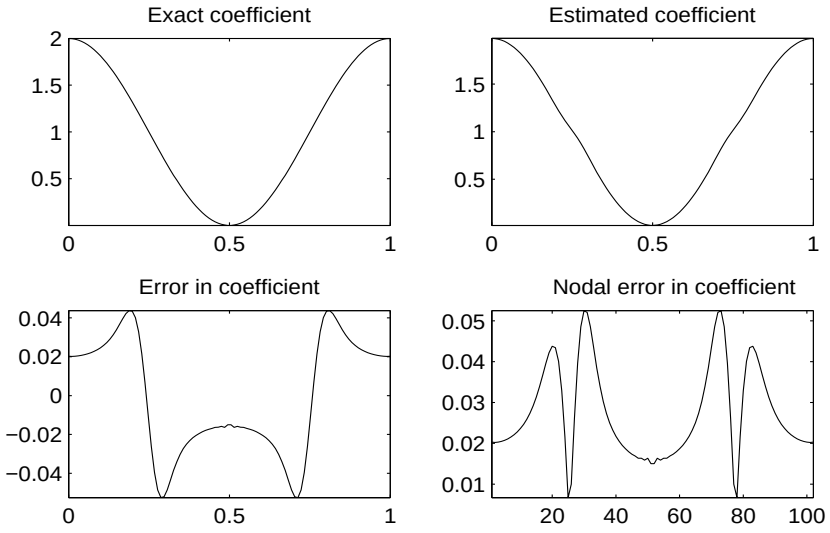


Figure 4.6: Example 3: Coefficient by Gradient Projection

Gradient Projection: Example 4

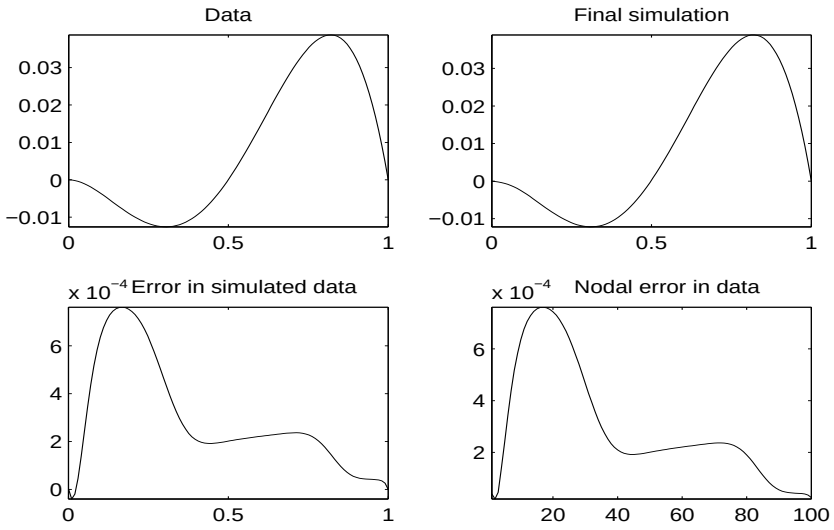


Figure 4.7: Example 4: Solution by Gradient Projection

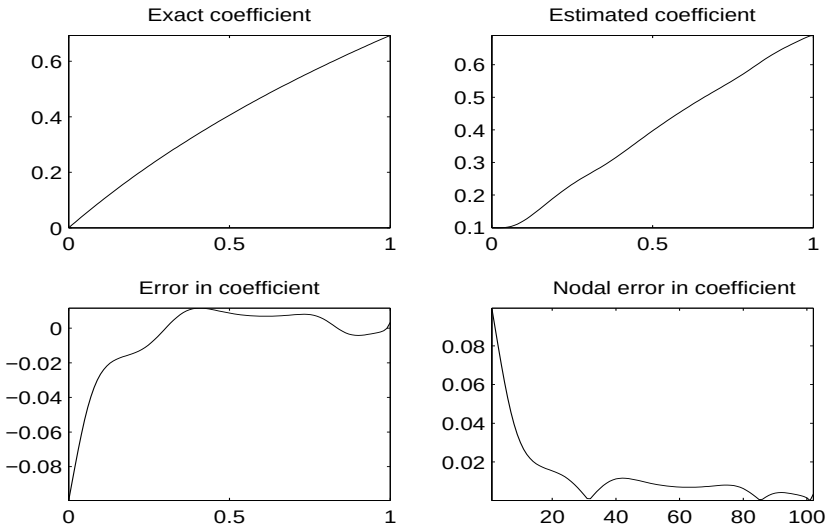


Figure 4.8: Example 4: Coefficient by Gradient Projection

## 4.4 Scaled Gradient Projection

The scaled gradient projection (SGP) method has the following iterative scheme:

$$A^{k+1} = P_{\tilde{A}}(A^k - \alpha_k D_k J'(A^k)) \tag{4.10}$$

It is common practice to take the scaling matrix $D_k$ as the main diagonal of the Hessian of $A^k$ with all other entries equal to zero.

---

Algorithm SGP

---

Choose $A^0 \in A_m, \beta, \theta \in (0,1), 0 < \alpha_{min} < \alpha_{max}, M > 0$
For $k = 0, 1, 2, ...$ Do the following steps:
  Step 1: Choose $\alpha_k \in [\alpha_{min}, \alpha_{max}]$ and $D_k$;
  Step 2: Projection $Y^k = P_{\tilde{A}}(A^k - \alpha D_k J'(A^k))$;
    If $Y^k = A^k$ Stop;
  Step 3: Descent direction: $d^k = Y^k - A^k$
  Step 4: Set $\lambda_k = 1$ and $f_{max} = max_{0 \le j \le \min(k, M-1)} J(A^{k-j})$
  Step 5: Backtracking loop:
    If $J(A^k + \lambda_k d^k) \le f_{max} + \beta \lambda_k J'(A^k)^T d^k$ Then
      Go to step 6;
    Else
      Set $\lambda_k = \theta \lambda_k$ and go to step 5;
    EndIf
  Step 6: $A^{k+1} = A^k + \lambda_k d^k$
END

---

Here we choose $\alpha_k$ using the Barzilai-Borwein rules:
Let $r^{k-1} = A^k - A^{k-1}$ and $z^{k-1} = J'(A^k) - J'(A^{k-1})$ with the following rules:

$$\alpha_k^{(1)} = \frac{r^{(k-1)^T} D_k^{-1} D_k^{-1} r^{(k-1)}}{r^{(k-1)^T} D_k^{-1} z(k-1)} \tag{4.11a}$$

$$\alpha_k^{(2)} = \frac{r^{(k-1)^T} D_k z^{(k-1)}}{z^{(k-1)^T} D_k^2 z^{(k-1)}}. \tag{4.11b}$$

Determining $\alpha_k$: take a prefixed non-negative integer $M_\alpha$ and $\tau_1 \in (0,1)$:

If $\alpha_k^{(2)} / \alpha_k^{(1)} \le \tau_k$ then
$\alpha_k = \min \left( \alpha_j^{(2)}, j = \max(1, k - M_\alpha), ..., k \right)$

$\tau_{k+1} = 0.9\tau_k$;
Else
$\alpha_k = \alpha_k^{(1)}$;
$\tau_{k+1} = 1.1\tau_k$
EndIf

Example 1: Scaled Gradient Projection

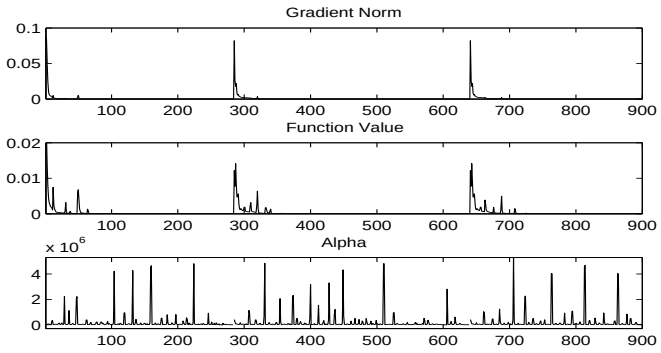


Figure 4.9: Example 1: History Output for SGP Method

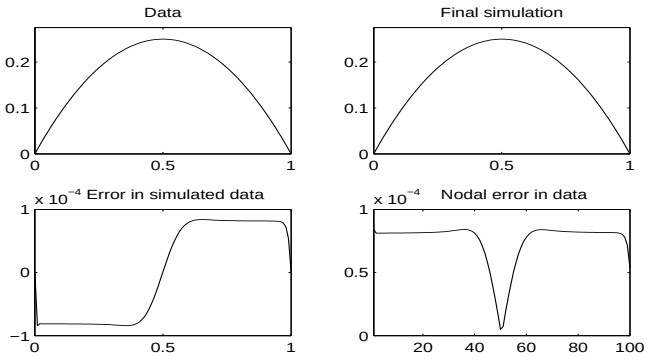


Figure 4.10: Example 1: Solution by SGP Method

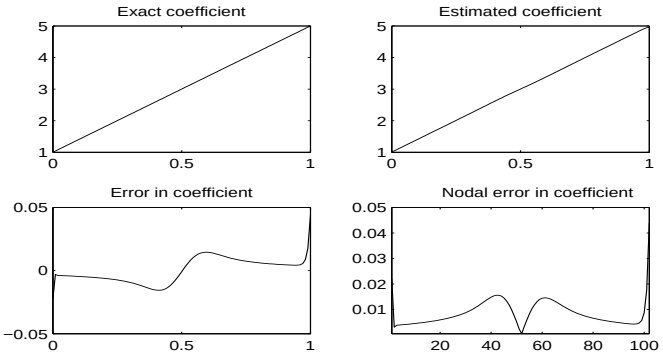


Figure 4.11: Example 1: Coefficient by SGP Method

Example 2: Scaled Gradient Projection

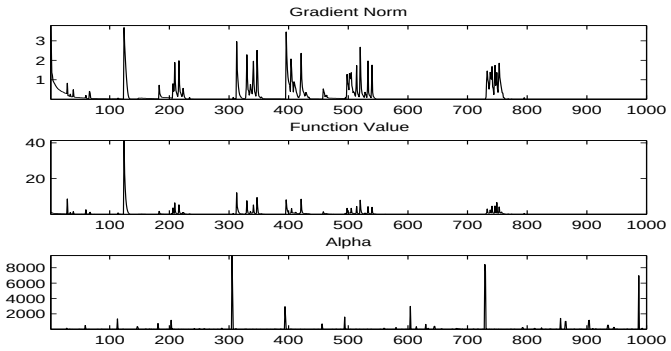


Figure 4.12: Example 2: History Output for SGP

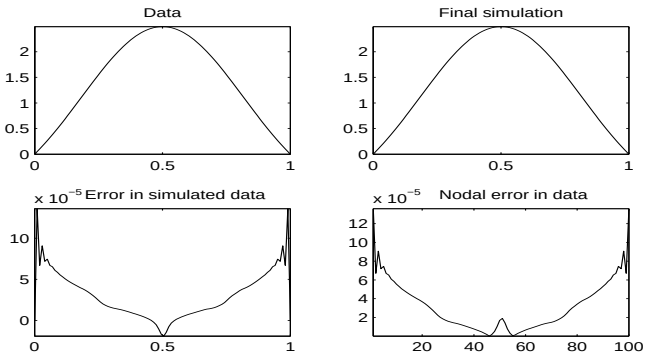


Figure 4.13: Example 2: Solution by SGP Method

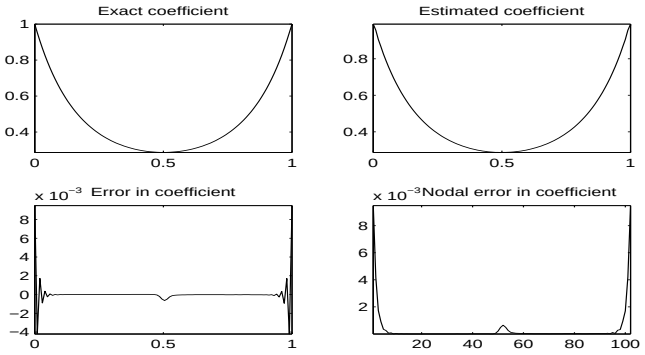


Figure 4.14: Example 2: Coefficient by SGP Method

Example 3: Scaled Gradient Projection

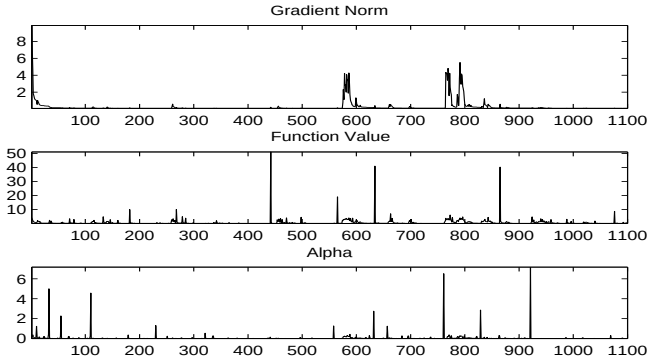


Figure 4.15: Example 3: History Output for SGP

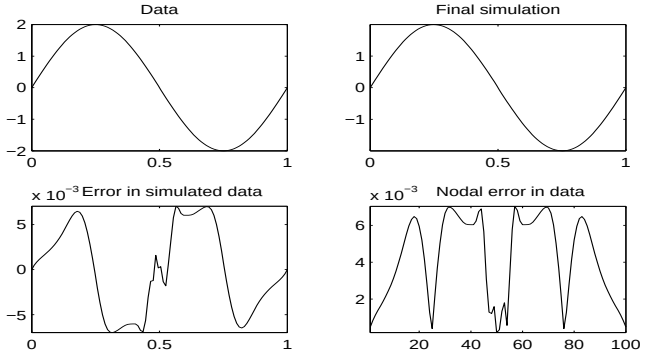


Figure 4.16: Example 3: Solution by SGP Method

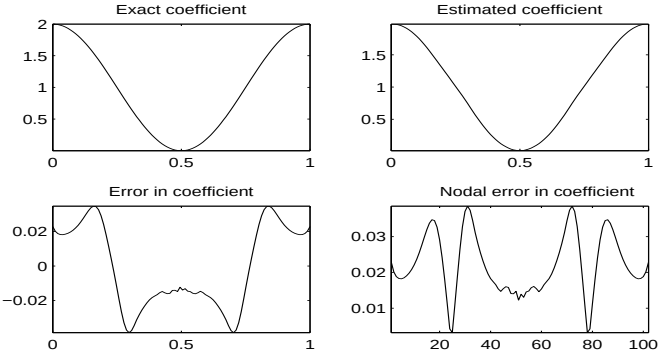


Figure 4.17: Example 3: Coefficient by SGP Method

Example 4: Scaled Gradient Projection

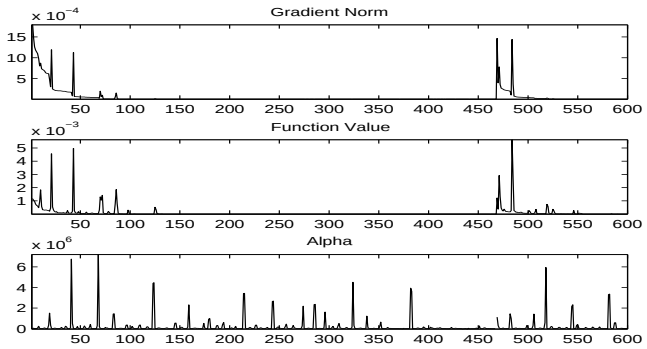


Figure 4.18: Example 4: History Output for SGP Method

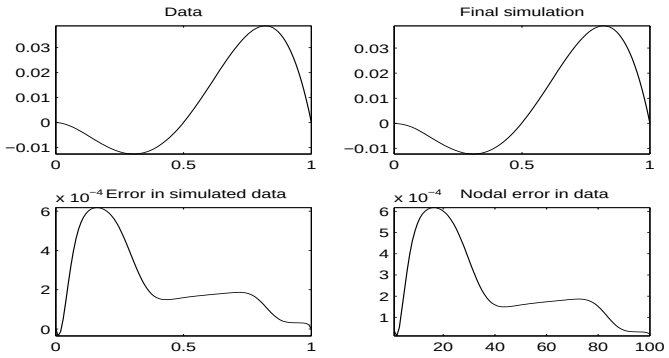


Figure 4.19: Example 4: Solution by SGP Method

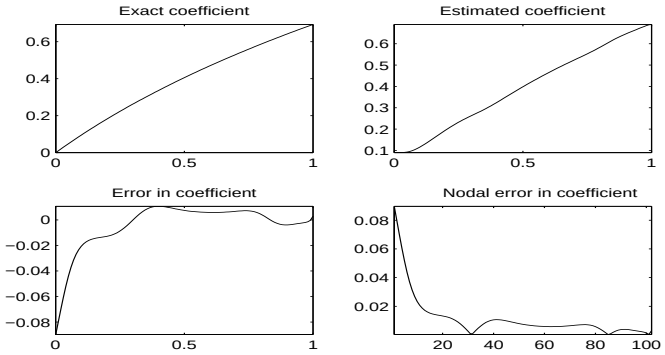


Figure 4.20: Example 4: Coefficient by SGP Method

## 4.5 Extragradient Method

Now we explore the extra-gradient method proposed in [26] to relax the conditions on convergence for the projection method:

$$\bar{A}^k = P_{\tilde{A}}(A^k - \alpha J'(A^k)) \tag{4.12a}$$
$$A^{k+1} = P_{\tilde{A}}(A^k - \alpha J'(\bar{A}^k)), \tag{4.12b}$$

where $\alpha$ is constant for all iterations.
In [4], convergence is proved under the conditions that $\tilde{A}$ is non-empty, $J'$ is monotone and Lipshitz (with constant L) and $\alpha \in (0, 1/L)$.
Convergence problems with $\alpha$ will be similar to the projection gradient method. When $L$ is unknown, we may have difficulties choosing an appropriate $\alpha$. As in the gradient projection method, if $\alpha$ is too small then the algorithm will converge slowly and if $\alpha$ is too big then it may not converge at all. Thus we will consider extragradient methods where $\alpha$ is an adaptive step-length.

### 4.5.1 Khobotov Extragradient Method

We are going to implement the adaptive steplength first introduced in [23] to remove the constraint that $J'$ must be Lipshitz continuous. The adaptive algorithm is of the form:

$$\bar{A}^k = P_{\tilde{A}}(A^k - \alpha_k J'(A^k)) \tag{4.13a}$$
$$A^{k+1} = P_{\tilde{A}}(A^k - \alpha_k J'(\bar{A}^k)). \tag{4.13b}$$

Intuitively, we get better convergence when $\alpha$ gets smaller between iterations, however, it is obvious that we must also control how the sequence of $\{\alpha_k\}$ goes to zero.
We use the following reduction rule for $\alpha_k$ given in [23]:

$$\alpha_k > \beta \frac{A^k - \bar{A}^k}{J'(A^k) - J'(\bar{A}^k)}, \tag{4.14}$$

where $\beta \in (0, 1)$. Results from [37] and [23] show that $\beta$ is usually 0.8 or 0.9 which our results support. The Khobotov extragradient algorithm has the following general algorithm:

| Khobotov-type Extragradient Algorithm |
|---|
| Step 1: Choose $\alpha$, $A^0$, and $\beta$ |
| Step 2: Compute $J'(A^k)$ |
| Step 3: Compute $\bar{A}^k = P_{\tilde{A}}(A^k - \alpha J'(A^k))$ |
| Step 4: Compute $J'(\bar{A}^k)$ |
| If $J'(\bar{A}^k) = 0$, STOP |
| Step 5: If $\alpha > \beta \frac{A^k - \bar{A}^k}{J'(A^k) - J'(\bar{A}^k)}$ |
| Then reduce $\alpha_k$ and go to step 5; |
| Step 6: Compute $A^{k+1} = P_{\tilde{A}}(A^k - \alpha J'(\bar{A}^k))$ |
| Step 7: If $\|A^{k+1} - A^k\| <$ TOL Then STOP, Else go to (2) |

### 4.5.2 Marcotte Choices for Steplength

The above algorithm requires a way to reduce $\alpha_k$. We are going to implement the Marcotte rule [30] and its variants [37]. The original Marcotte rule incorporates the sequence $a_k = a_{k-1}/2$ and forcing $\alpha_k$ to satisfy step 5 above:

$$\alpha_k = \min\left\{\frac{\alpha_{k-1}}{2}, \frac{\|A^k - \bar{A}^k\|}{\sqrt{2}\|J'(A^k) - J'(\bar{A}^k)\|}\right\} \tag{4.15}$$

Potentially we can still choose $\alpha$ small enough that we never reduce $\alpha_k$. In this case we get very slow convergence. To avoid this, we want the sequence $\alpha_k$ to have the ability to increase if $\alpha_k$ if $\alpha_{k-1}$ is smaller than optimal. The first and second modified versions of Marcotte have this rule:
First modified version of Marcotte

$$\alpha_k = \alpha_{k-1} + \left(\beta \frac{\|A^{k-1} - \bar{A}^{k-1}\|}{\|J'(A^{k-1}) - J'(\bar{A}^{k-1})\|} - \alpha_{k-1}\right)\gamma, \tag{4.16}$$

where $\gamma \in (0, 1)$.
Second modified version of Marcotte

$$\alpha_k = \max\left\{\hat{\alpha}, \min\left\{\xi \cdot \alpha_{k-1}, \beta \frac{\|A^{k-1} - \bar{A}^{k-1}\|}{\|J'(A^{k-1}) - J'(\bar{A}^{k-1})\|}\right\}\right\} \tag{4.17}$$

where $\xi \in (0, 1)$, and $\hat{\alpha}$ is some lower limit for $\alpha_k$, generally, no less than $1.0 \times 10^{-4}$.

Example 1: Marcotte methods (all three versions)

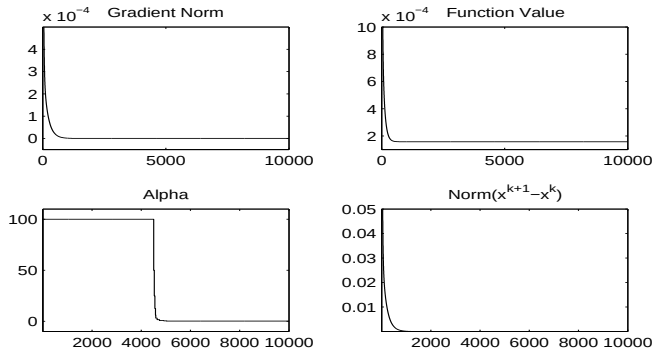


Figure 4.21: Example 1: History Output for Marcotte Method

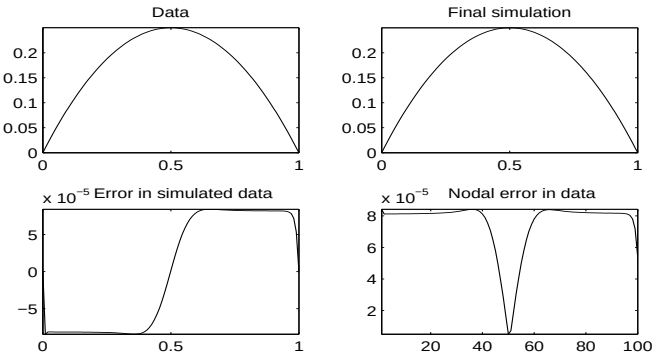


Figure 4.22: Example 1: Solution by Marcotte Methods

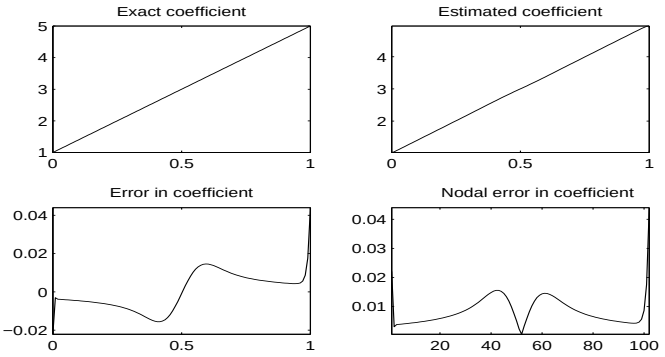


Figure 4.23: Example 1: Coefficient by Marcotte Methods

Example 2: Marcotte Methods (all three versions)

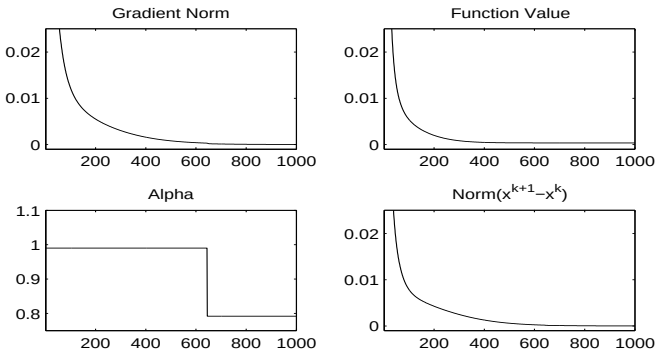


Figure 4.24: Example 2: History Output for Marcotte SMV Method

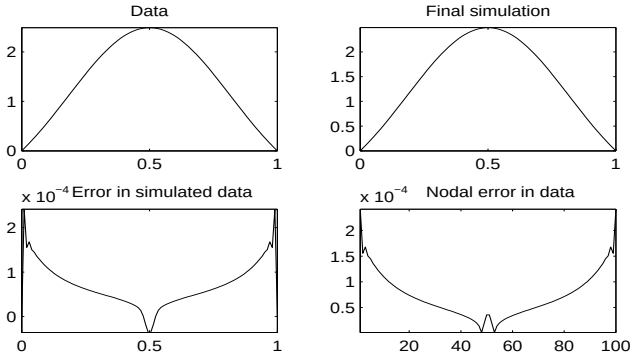


Figure 4.25: Example 2: Solution by Marcotte Methods

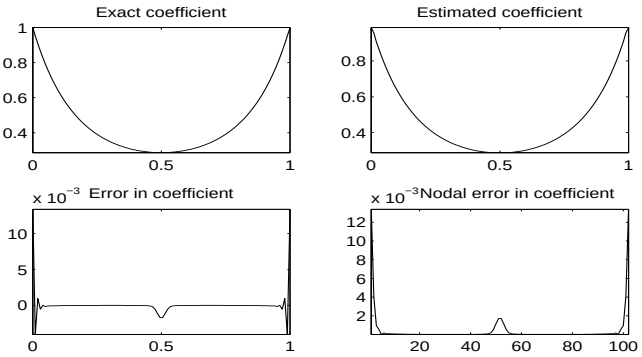


Figure 4.26: Example 2: Coefficient by Marcotte Methods

Example 3: Marcotte Method (similar to the first variant)

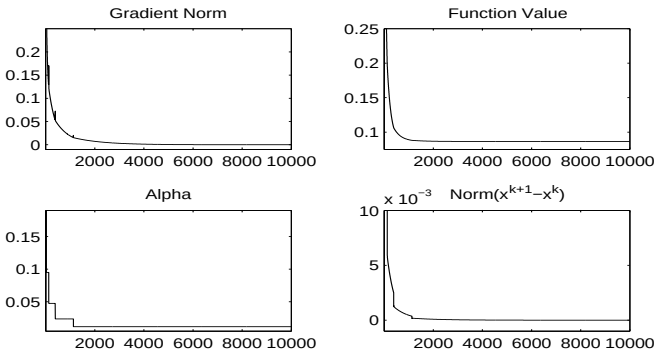


Figure 4.27: Example 3: History Output for Marcotte Method

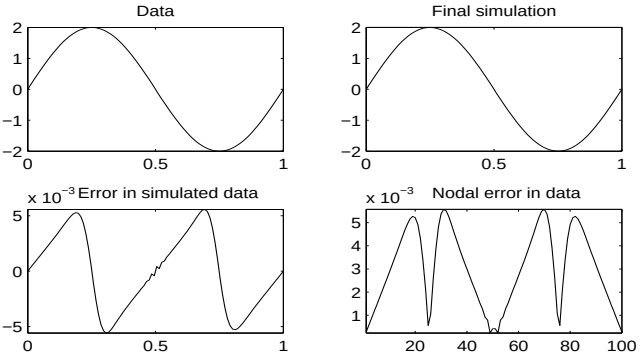


Figure 4.28: Example 3: Solution by Marcotte Method

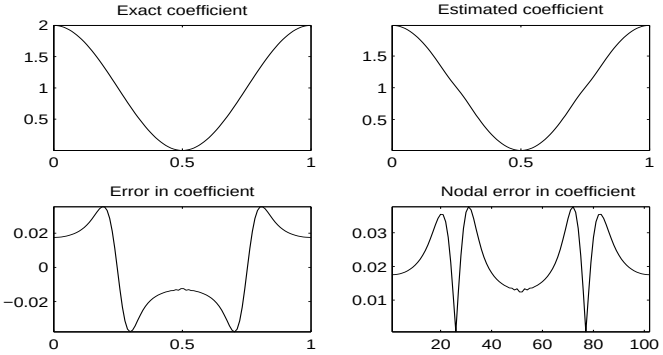


Figure 4.29: Example 3: Solution by Marcotte Method

Example 3: Second Marcotte Method

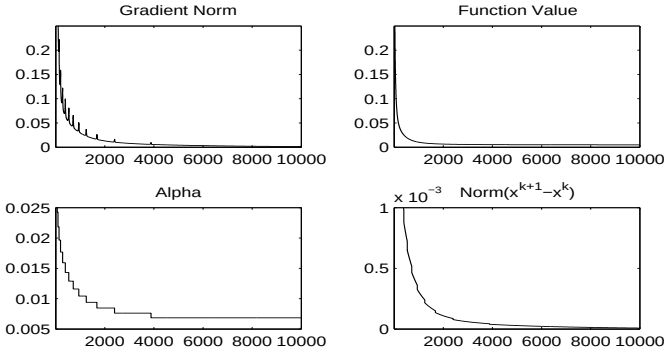


Figure 4.30: Example 3: History Output for Marcotte SMV Method

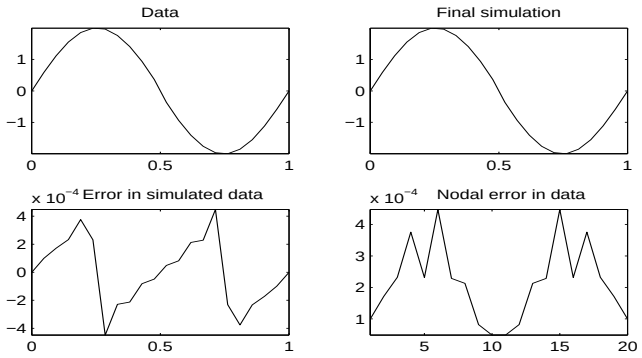


Figure 4.31: Example 3: Solution by Marcotte SMV Method

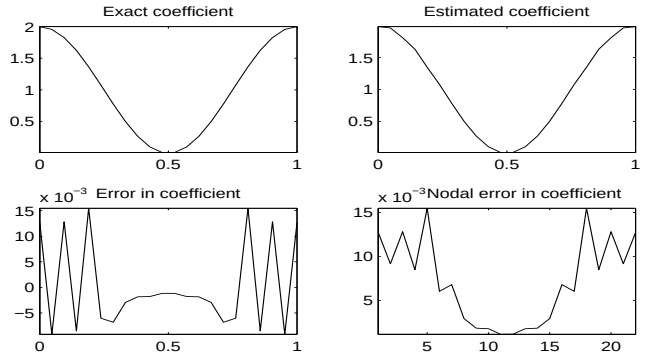


Figure 4.32: Example 3: Solution by Marcotte SMV Method

Example 4: Marcotte Method (similar first variant)

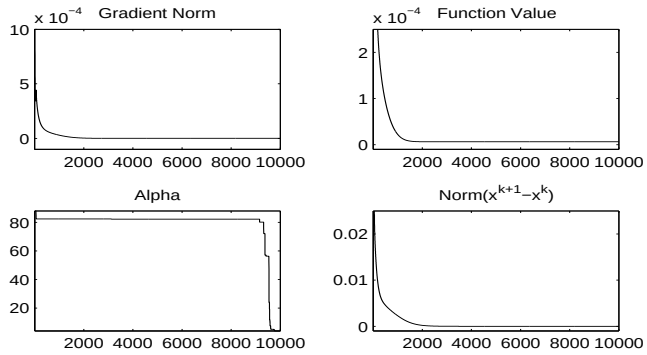


Figure 4.33: Example 4: History Output for Marcotte FMV Method

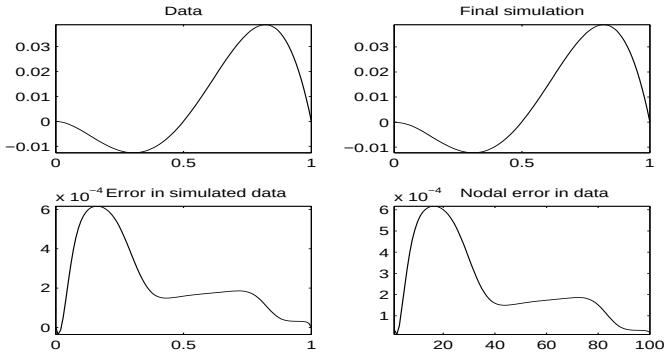


Figure 4.34: Example 4: Solution by Marcotte Method

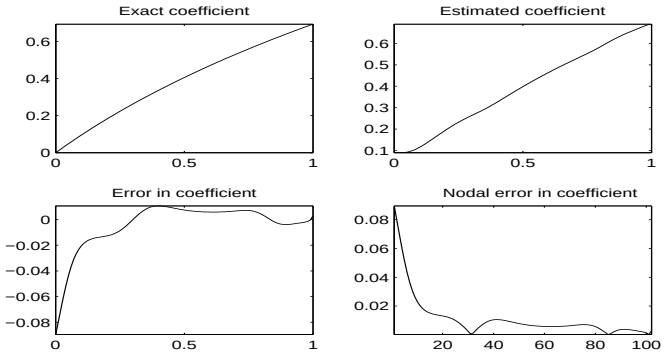


Figure 4.35: Example 4: Solution by Marcotte Method

Example 4: Second Marcotte Method

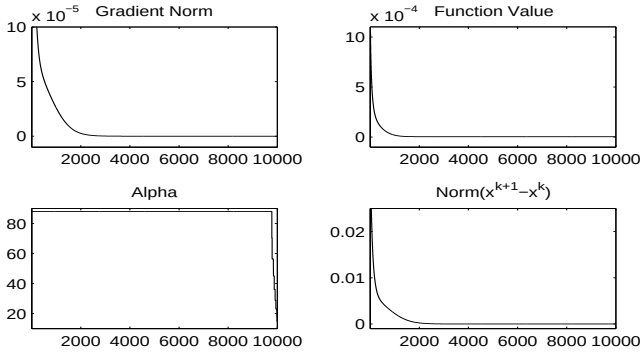


Figure 4.36: Example 4: History Output for Marcotte SMV Method

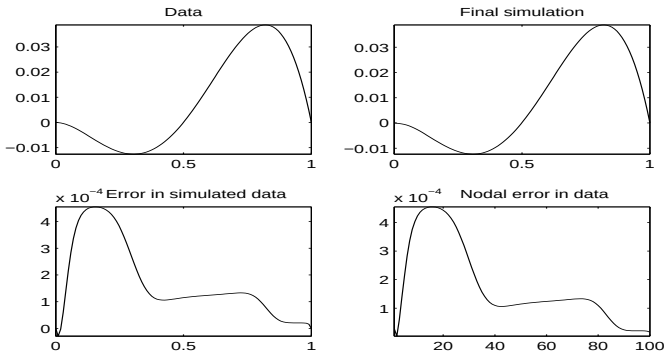


Figure 4.37: Example 4: Solution by Marcotte SMV Method

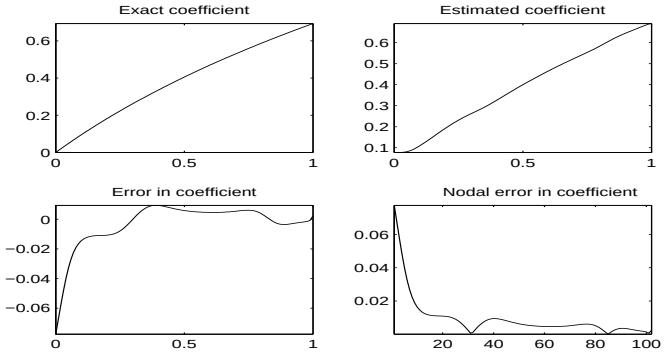


Figure 4.38: Example 4: Solution by Marcotte SMV Method

## 4.6 Scaled Extragradient Method

Now we consider a projection-contraction type extragradient method presented by Solodov and Tseng in [34]. It involves a scaling matrix to accelerate convergence.

$$\bar{A}^k = P_{\tilde{A}}(A^k - \alpha_k J'(A^k)) \tag{4.18}$$

$$A^{k+1} = A^k - \gamma M^{-1}(T_\alpha(A^k) - T_\alpha(P_{\tilde{A}}(\bar{A}^k)) \tag{4.19}$$

where $\gamma \in R^+$ and $T_\alpha = (I - \alpha J')$; here, $I$ is the identity matrix, and $\alpha$ is chosen such that $T_\alpha$ is strongly monotone.
Additional discussion of the scaling matrix is given in [37], however, in both [37] and [34], test problems take $M$ equal to the identity matrix.

Solodov-Tseng Method

Step 1: Choose $A^0, \alpha_{-1}, \theta \in (0,2), \rho, \beta \in (0,1), M \in R^{m\times m}$
Step 2: $\bar{A}^0 = 0, k = 0, rx = ones(m,1)$
Step 3: if $\|rx\| <$ TOL then STOP
  else $\alpha = \alpha_{k-1}$, $flag = 0$;
Step 4: if $J'(A^k) = 0$ then STOP
Step 5: While $\alpha(A^k - \bar{A}^k)^T(J'(A^k) - J'(\bar{A}^k)) > (1-\rho)\|A^k - \bar{A}^k\|^2$ or $flag = 0$
  If $flag \neq 0$ Then $\alpha = \alpha_{k-1}\beta$ endif
  update $\bar{A}^k = P_{\tilde{A}}(A^k - \alpha J'(A^k))$, compute $J'(\bar{A}^k)$
  $flag = flag + 1$;
endwhile
Step 6: update $\alpha_k = \alpha$;
Step 7: compute $\gamma = \theta\rho\|A^k - \bar{A}^k\|^2 / \|M^{1/2}(A^k - \bar{A}^k - \alpha_k J'(A^k) + \alpha_k J'(\bar{A}^k))\|^2$;
Step 8: compute $A^{k+1} = A^k - \gamma M^{-1}(A^k - \bar{A}^k - \alpha_k J'(A^k) + \alpha_k J'(\bar{A}^k))$
Step 9: $rx = A^{k+1} - A^k, k = k+1$ go to step (3)

The Solodov-Tseng method suggests a more general form for the advanced extragradient methods:

$$\bar{A}^k = P_{\tilde{A}}(A^k - \alpha_k J'(\alpha^k)) \tag{4.20a}$$

$$A^{k+1} = P_{\tilde{A}}(A^k - \eta_k J'(\bar{A}^k)), \tag{4.20b}$$

where $\alpha_k$ and $\eta_k$ are chosen using different rules.

Example 1: Solodov-Tseng Method

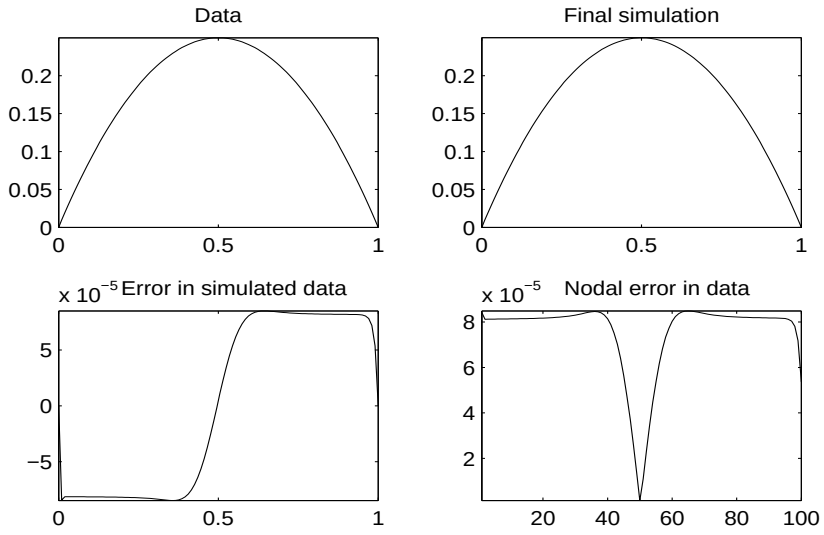


Figure 4.39: Example 1: Solution by Solodov-Tseng Method

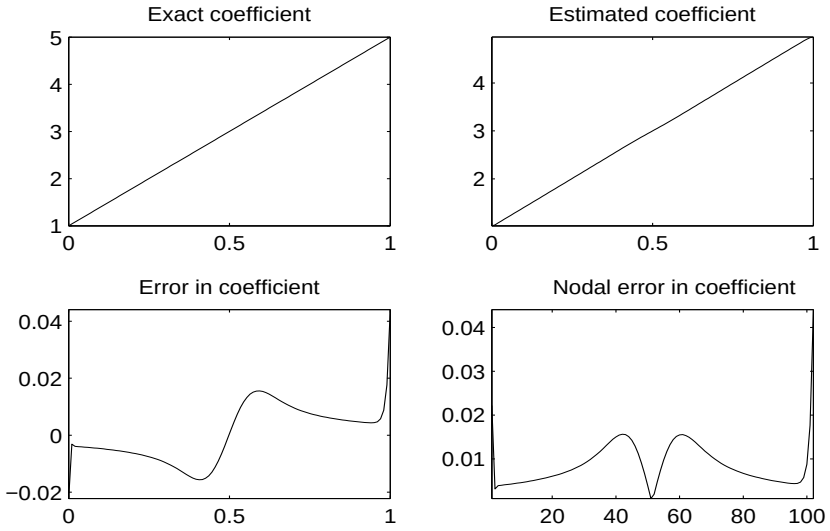


Figure 4.40: Example 1: Coefficient by Solodov-Tseng Method

Example 2: Solodov-Tseng method

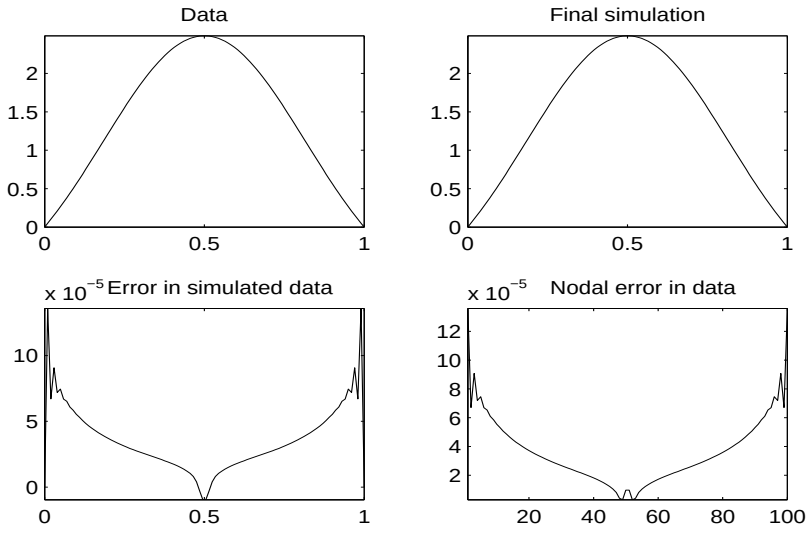


Figure 4.41: Example 2: Solution by Solodov-Tseng Method

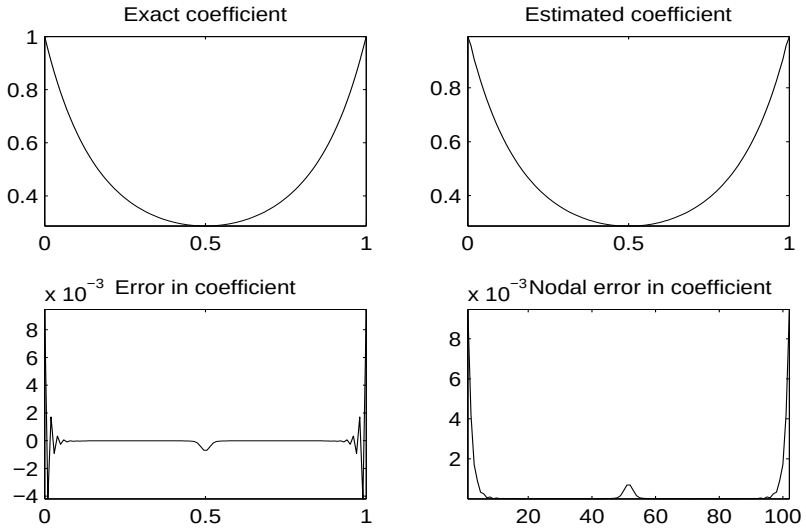


Figure 4.42: Example 2: Coefficient Solodov-Tseng Method

Example 3: Solodov-Tseng method

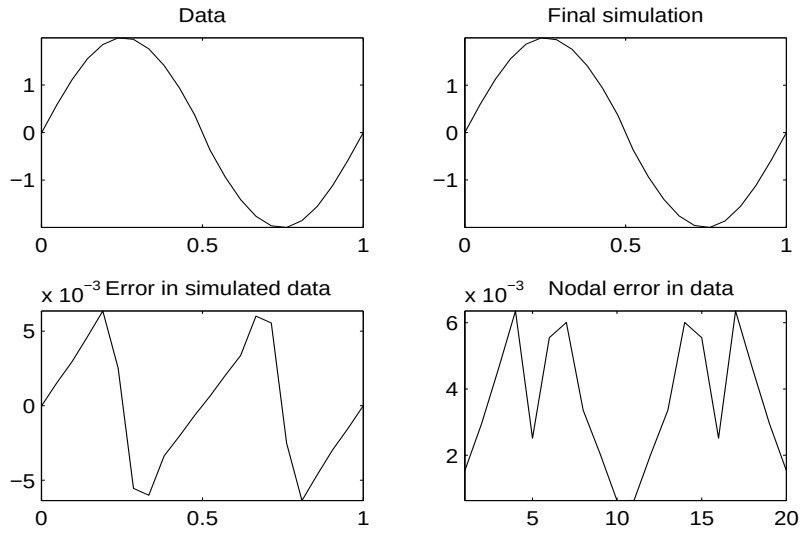


Figure 4.43: Example 3: Solution by Solodov-Tseng Method

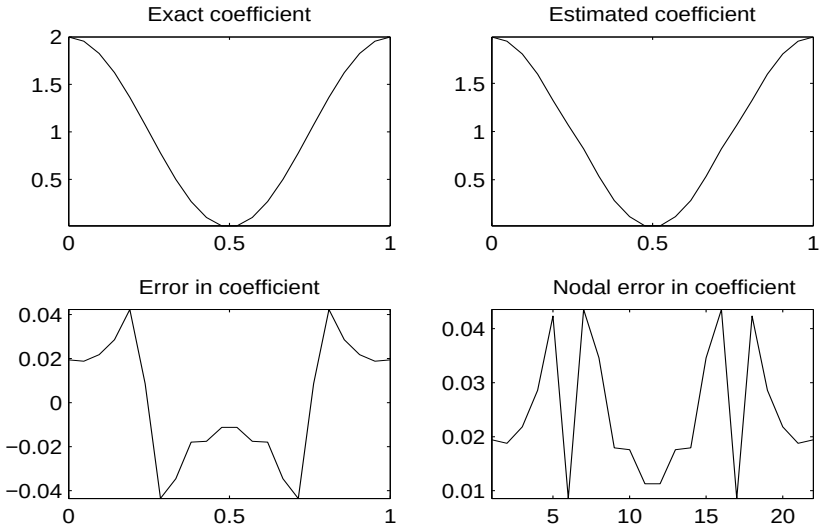


Figure 4.44: Example 3: Coefficient by Solodov-Tseng Method

Example 4: Solodov-Tseng method

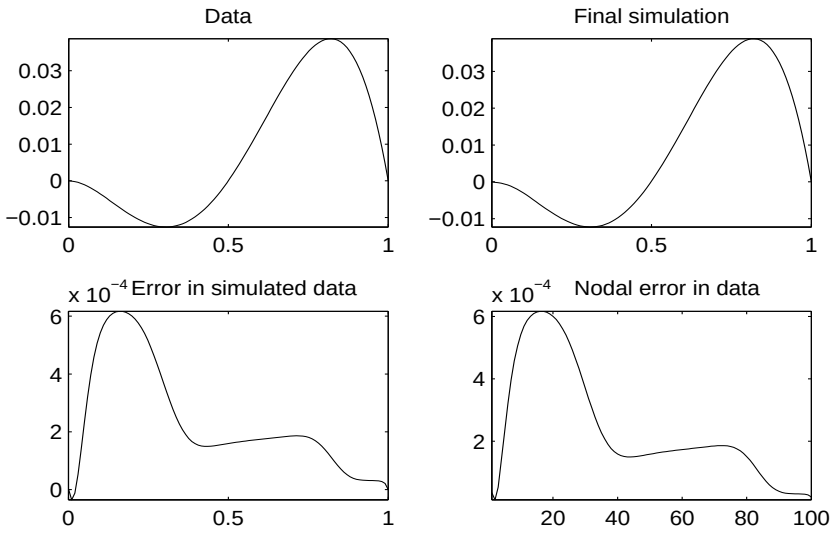


Figure 4.45: Example 4: Solution by Solodov-Tseng Method

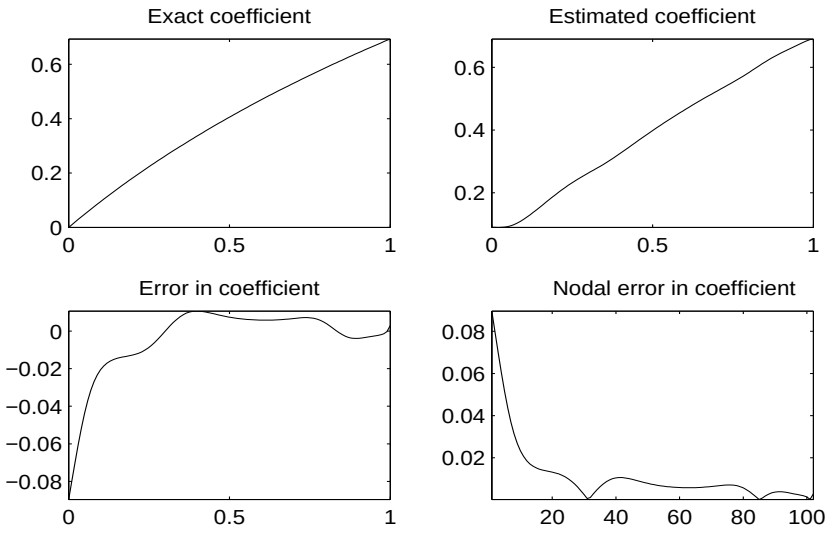


Figure 4.46: Example 4: Coefficient by Solodov-Tseng Method

## 4.7 Goldstein-Type Methods

The classical Goldstein projection method presented in [27] is of the form:

$$A^{k+1} = P_{\tilde{A}}[A^k - \beta_k J'(A^k)]. \tag{4.21}$$

The He-Goldstein method, an extragradient method that requires Lipschitz continuity and strong monotonicity of $J'$ is of the form:

$$\bar{A}^k = P_{\tilde{A}}(J'(A^k) - \beta_k A^k) \tag{4.22a}$$

$$A^{k+1} = A^k - \frac{1}{\beta_k}\{J'(A^k) - \bar{A}^k\}. \tag{4.22b}$$

It can also be expressed:

$$r(A^k, \beta_k) = \frac{1}{\beta_k}\{J'(A^k) - P_{\tilde{A}}[J'(A^k) - \beta_k A^k]\} \tag{4.23a}$$

$$A^{k+1} = A^k - r(A^k, \beta_k). \tag{4.23b}$$

We are going to implement the more general version of equation (4.23) presented in [27], it allows us to control the second projection (choosing $\eta_k$).

| Improved Goldstein-type method |
|---|
| Initialize: choose $\beta_U > \beta_L > 1/(4\tau), \gamma \in (0,2), \epsilon > 0, A^0, \beta_0 \in [\beta_L, \beta_U], k = 0$; |
| Step 1: Compute: |
| $r(A^k, \beta_k) = \frac{1}{\beta_k}\{J'(A^k) - P_{\tilde{A}}[J'(A^k) - \beta_k A^k]\}$ |
| If $\|r(A^k, \beta_k) \leq \epsilon$ then STOP |
| Step 2: $A^{k+1} = A^k - \gamma\alpha_k r(A^k, \beta_k)$ where $\alpha_k := 1 - \frac{1}{4\beta_k\tau}$ |
| Step 3: Update $\beta_k$ |
| $\omega_k = \frac{\|J'(A^{k+1} - J'(A^k)\|}{\beta_k \|A^{k+1} - A^k\|}$ |
| If $\omega_k < \frac{1}{2}$ Then $\beta_{k+1} = max\{\beta_L, \frac{1}{2}\beta_k\}$ |
| Else if $\omega_k > \frac{3}{2}$ Then $\beta_{k+1} = min\{\beta_U, \frac{6}{5}\beta_k\}$ |
| Step 4: $k = k + 1$, go to step (1); |

Example 2: Goldstein Extragradient Method

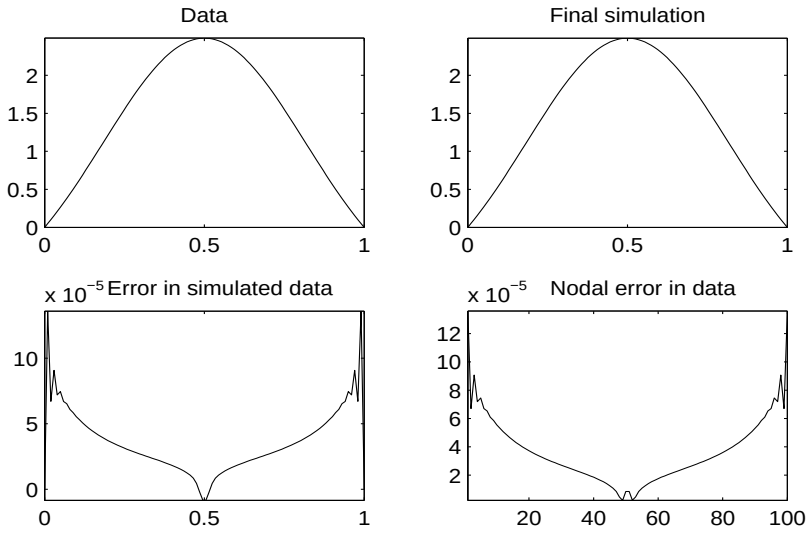


Figure 4.47: Example 2: Solution by Goldstein extragradient variant

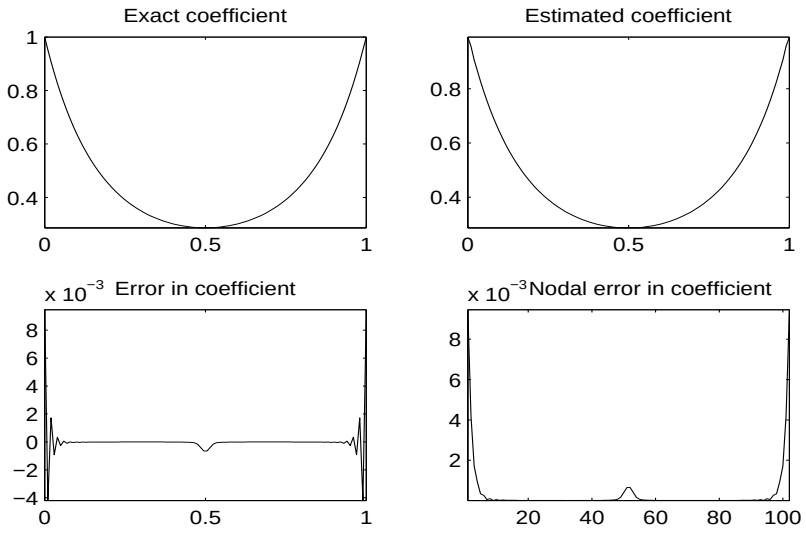


Figure 4.48: Example 2: Coefficient by Goldstein Extragradient Variant

Example 3: Goldstein Extragradient Method

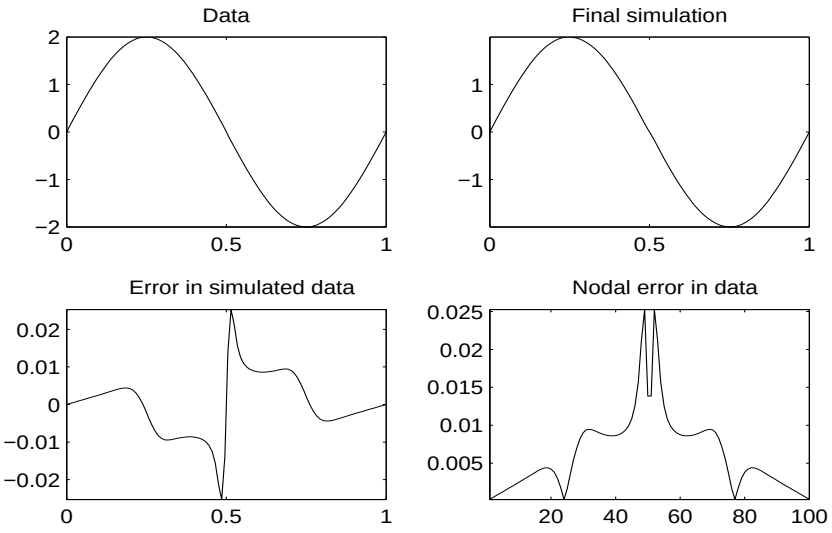


Figure 4.49: Example 3: Solution by Goldstein Extragradient Variant

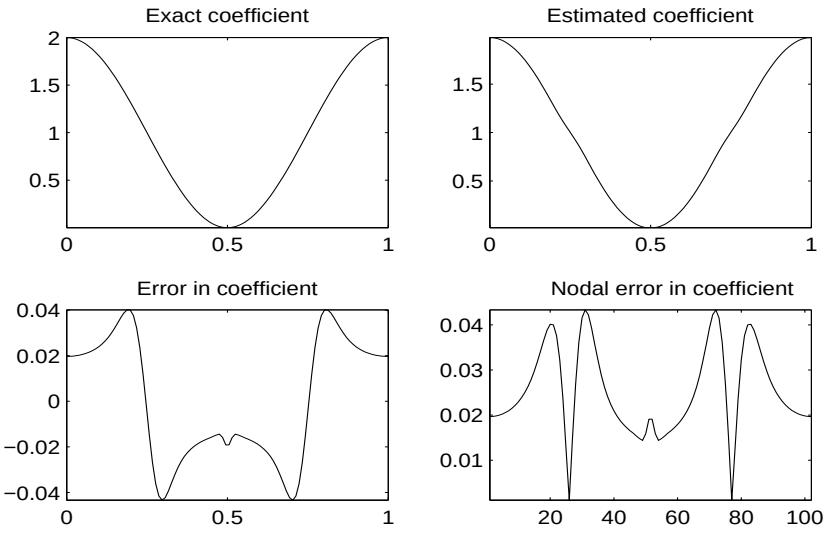


Figure 4.50: Example 3: Coefficient by Goldstein Extragradient Variant

Example 4: Goldstein Extragradient Method

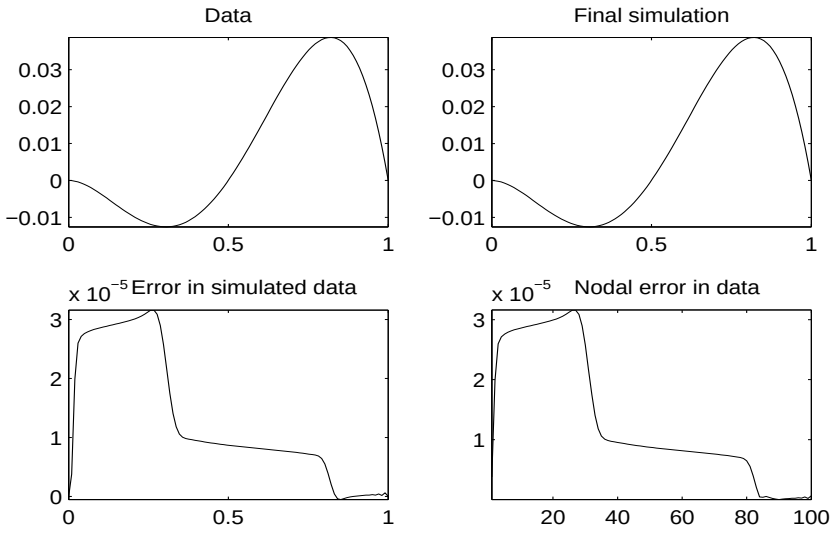


Figure 4.51: Example 4: Solution by Goldstein Extragradient Variant

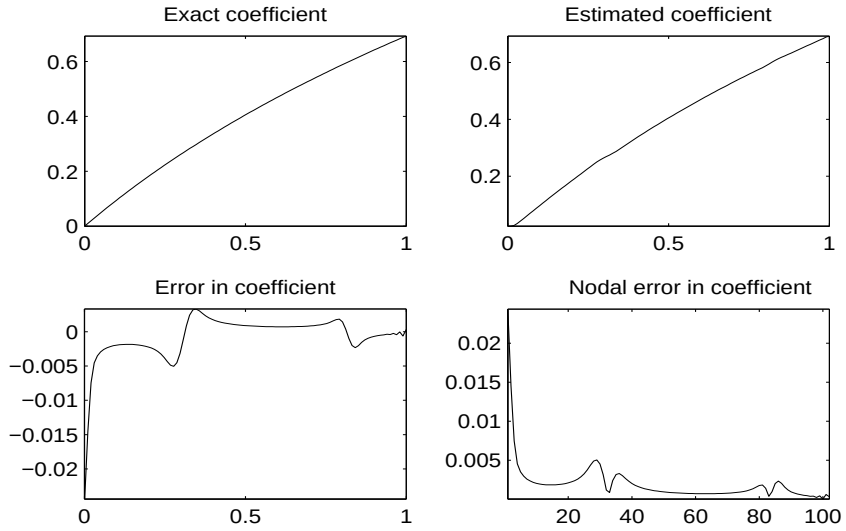


Figure 4.52: Example 4: Coefficient by Goldstein Extragradient Variant

## 4.8 Hyperplane Extragradient Method

We choose $\eta_k$ using the following rule from [37]:

$$\eta_k = \frac{\langle J'(\bar{A}^k), A^k - \bar{A}^k \rangle}{\|J'(\bar{A}^k)\|^2} \tag{4.24}$$

The idea here is that the hyperplane of all solutions $A$ such that

$$\langle J'(\bar{A}^k), \bar{A}^k - A \rangle = 0, \tag{4.25}$$

separates all the solutions onto one side of the hyperplane. Using our definitions of the variational inequality, we know which side the solutions fall onto:

$$\langle J'(A^*), \bar{A}^k - A^* \rangle \geq 0 \tag{4.26}$$

Consequently, if $J'$ is monotone, then we also have

$$\langle J'(\bar{A}^k), \bar{A}^k - A^* \rangle \geq 0. \tag{4.27}$$

Thus if

$$\langle J'(\bar{A}^k), \bar{A}^k - A^k \rangle < 0, \tag{4.28}$$

then we know that we have to look for the solution on the other side of the hyperplane.

This method, presented by Iusem, requires three constants, $\epsilon \in (0, 1)$ and $\tilde{\alpha} \geq \hat{\alpha} > 0$ such that the sequence $\alpha_k$ is computed such that

$$\langle J'(\bar{A}^k), \bar{A}^k - A^k \rangle \leq 0,$$

when $\alpha_k \in [\hat{\alpha}, \tilde{\alpha}]$.

| Hyperplane method |
|---|
| Step 1: Choose $A^0, \epsilon, \hat{\alpha}, \tilde{\alpha}$ |
| Step 2: Set $k = 0, rx = ones(m, 1)$ |
| Step 3: If $\|rx\| <$ TOL then STOP |
| Step 4: Choose $\tilde{\alpha}_k$ using a finite bracketing procedure |
| Step 5: Compute $\tilde{A}^k = P_{\tilde{A}}(A^k - \tilde{\alpha}_k J'(A^k))$ and $J'(\tilde{A}^k)$ |
| Step 6: If $J'(\tilde{A}^k) = 0$ then STOP |
| Step 7: If $\|J'(\tilde{(A)}^k - J'(A^k)\| \leq \frac{\|\tilde{A}^k - A^k\|^2}{2\tilde{\alpha}_k^2 \|J'(A^k)\|}$ |
| Then $\bar{A}^k = \tilde{A}^k$ |
| Else find $\alpha_k \in (0, \tilde{\alpha}_k)$ such that |
| $\epsilon \frac{\|\tilde{A}^k - A^k\|^2}{2\tilde{\alpha}_k^2 \|J'(A^k)\|} \leq \|J'(P_{\tilde{A}}(A^k - \alpha_k J'(A^k))) - J'(A^k)\| \leq \frac{\|\tilde{A}^k - A^k\|^2}{2\tilde{\alpha}_k^2 \|J'(A^k)\|}$ |
| Step 8: Compute $\bar{A}^k = P_{\tilde{A}}(A^k - \alpha_k J'(A^k))$ |
| Step 9: If $J'(\bar{A}^k) = 0$ then STOP |
| Step 10: Compute $\eta_k$ |
| Step 11: Compute $A^{k+1} = P_{\tilde{A}}(A^k - \eta J'(\bar{A}^k))$ |
| Step 12: $rx = A^{k+1} - A^k, k = k + 1$; go to (3); |

Example 1: Hyperplane Extragradient Method

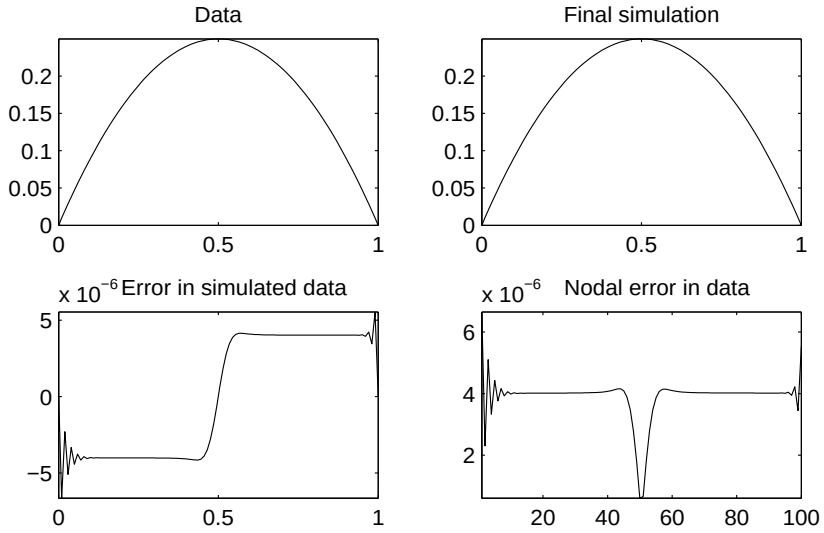


Figure 4.53: Example 1: solution by Hyperplane Extragradient Variant

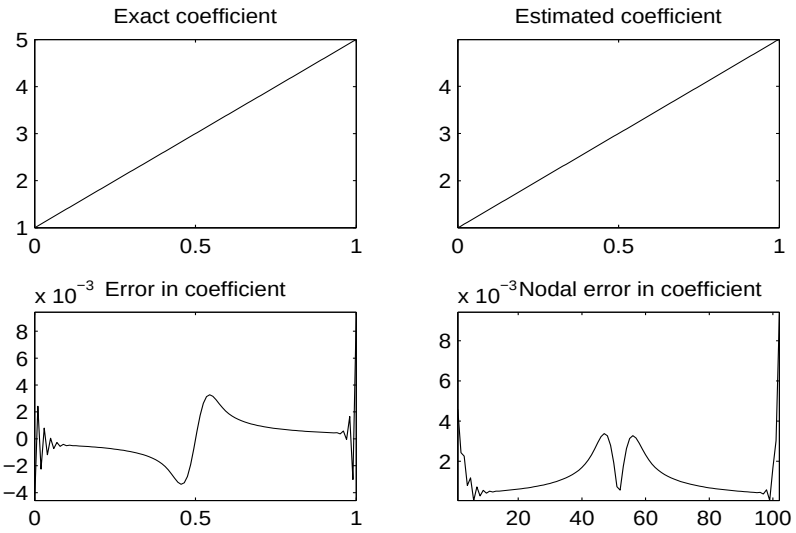


Figure 4.54: Example 1: Coefficient by Hyperplane Extragradient Variant

Example 2: Hyperplane Extragradient Method

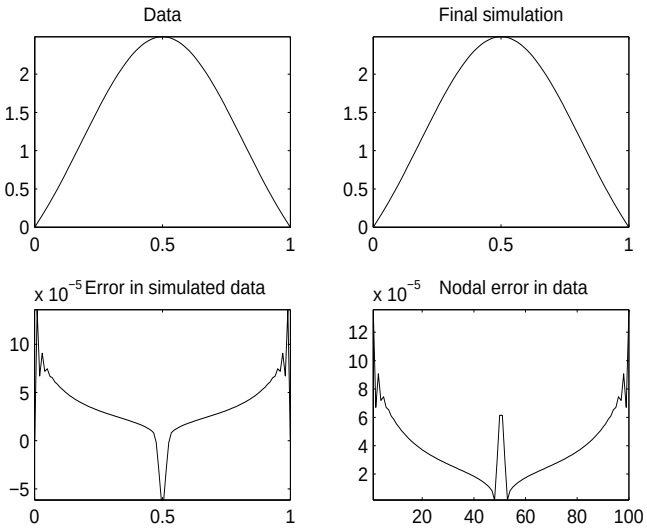


Figure 4.55: Example 2: Solution by Hyperplane Extragradient Variant

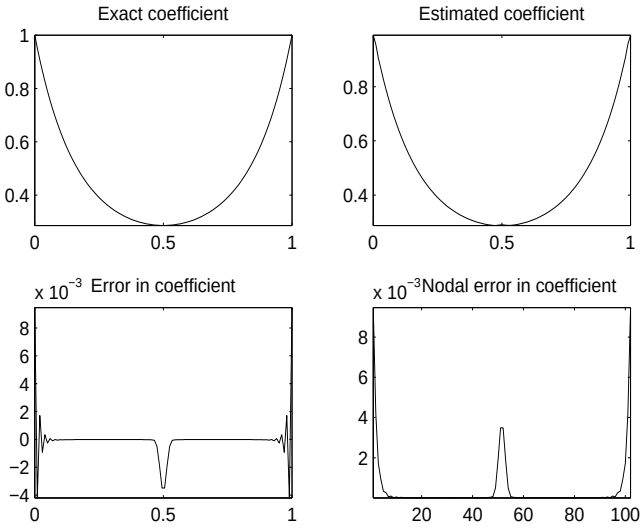


Figure 4.56: Example 2: Coefficient by Hyperplane Extragradient Variant

Example 3: Hyperplane Extragradient Method

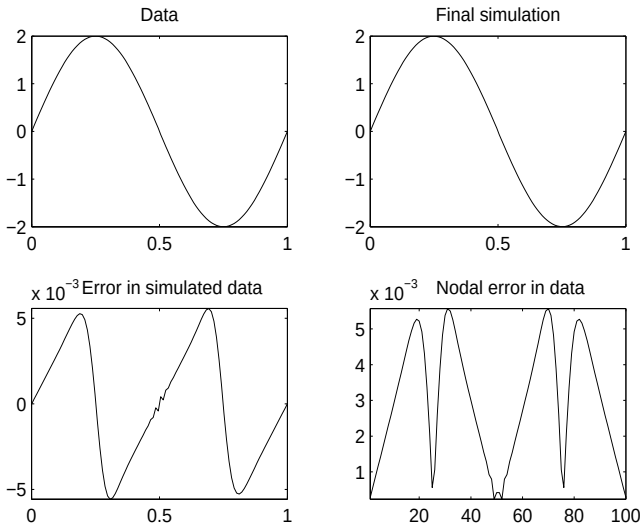


Figure 4.57: Example 3: Solution by Hyperplane Extragradient Variant

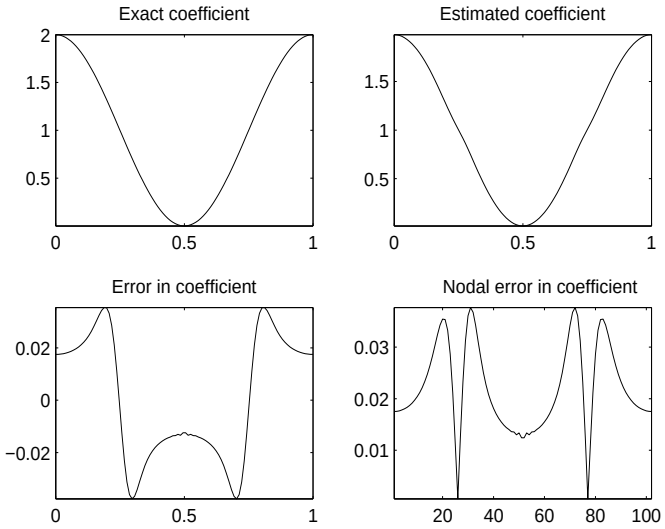


Figure 4.58: Example 3: Coefficient by Hyperplane Extragradient Variant

Example 4: Hyperplane Extragradient Method

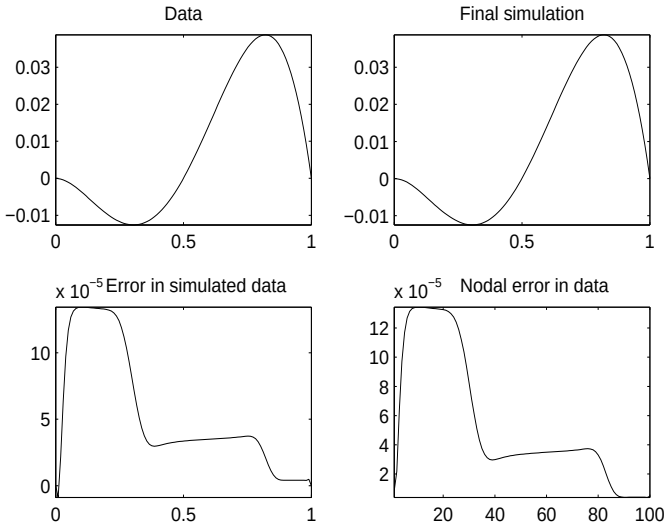


Figure 4.59: Example 4: Solution by Hyperplane Extragradient Variant

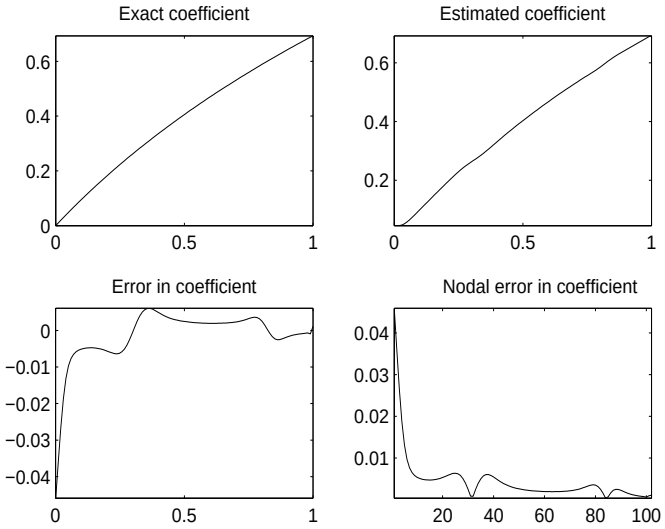


Figure 4.60: Example 4: Coefficient by Hyperplane Extragradient Variant

# Chapter 5

# Performance Analysis

In this section, we present a comparison for the following methods:

1. Gradient Projection Using Armijo Line Search
2. Scaled Gradient Projection Using Barzilai-Borwein Rules
3. Khobotov Extragradient Method Using Marcotte Rules (3 Versions)
4. Solodov-Tseng (Projection-Contraction) Method
5. Improved He-Goldstein Type Extragradient Method
6. Hyperplane Extragradient Method

## 5.1 Results

### Example 1

| N=100 | $L_2$-Error | Iterations | $\epsilon$ |
|---|---|---|---|
| Gradient Projection | $2.24e-5$ | 100000 | 6.0e-6 |
| SGP | $1.80e-5$ | 10000 | 5.0e-6 |
| Marcotte methods | $2.63-6$ | 10977 | 1.0e-6 |
| Solodov-Tseng | $2.63e-6$ | 30159 | 1.0e-6 |
| Hyperplane | $2.63e-6$ | 5464 | 1.0e-6 |

- Set gradient tolerance to 1.0e-10
- Parameters for examples:
- Gradient Projection: gradient did not reach 1.0e-10 after 100000 iterations.
- Marcotte: $\alpha = 100, \beta = 0.8$
- First variant: $\alpha = 70, \beta = 0.8, \gamma = 0.9$
- Second variant: $\alpha = 70, \beta = 0.8, \gamma = 0.8, \alpha_{max} = .0004$
- SGP: $\beta = 0.8, \theta = 0.9, \alpha_{min} = .001, \alpha_{max} = 1000, \tau = .9$
- Solodov-Tseng: $\alpha_{-1} = 10, \theta = 1.5, \rho = 0.9, \beta = 0.7$
- Hyperplane: $\epsilon = 1.0e-4, \hat{\alpha} = 35, \tilde{\alpha} = 75$

## Example 2

| N=100 | $L_2$ Error | $\epsilon$ |
|---|---|---|
| Gradient Projection | $5.08e-7$ | 4.0e-5 |
| SGP | $5.14e-7$ | 4.0e-5 |
| Marcotte Methods | $3.34e-8$ | 1.0e-6 |
| Solodov-Tseng | $3.13e-8$ | 1.0e-6 |
| He-Goldstein variant | $3.18e-8$ | 1.0e-6 |
| Hyperplane | $3.11e-8$ | 1.0e-6 |

- Set gradient tolerance $= 1.0e-10$
- Parameters for examples:
- Gradient Projection: Did not reach gradient tolerance after 30000 iterations
- Marcotte: $\alpha = .99, \beta = 0.9$
- First variant: $\alpha = .99, \beta = 0.8, \gamma = 0.8$
- Second variant: $\alpha = .99, \beta = 0.8, \gamma = 0.8, \alpha_{max} = .001$
- Solodov-Tseng $\alpha_{-1} = 1.1, \theta = 1.6, \rho = 0.8, \beta = 0.8$
- SGP: $\beta = 0.8, \theta = 0.9, \alpha_{min} = .001, \alpha_{max} = 1000, \tau = .9$
- He-Goldstein variant: $\gamma = 0.8, \tau = 2, \beta = 0.8, \beta_L = 1/(2 * \tau), \beta_U = 25$
- Hyperplane: $\epsilon = 1.0e-7, \hat{\alpha} = .06, \tilde{\alpha} = .1$

## Example 3

| N=100 | $L_2$ Error | $\epsilon$ |
|---|---|---|
| Gradient Projection | $8.84e-4$ | 8.0e-3 |
| SGP | $5.56e-4$ | 9.0e-3 |
| Marcotte | $3.41e-4$ | 5.0e-4 |
| Solodov-Tseng | $6.61e-4$ | 9.0e-3 |
| He-Goldstein variant | $7.29e-5$ | 9.0e-3 |
| Hyperplane | $5.57e-5$ | 9.0e-3 |

- Gradient tolerance $= 1.0e-8$
- Parameters for examples:
- Marcotte: $\alpha = .19, \beta = 0.7$
- First variant: $\alpha = .1, \beta = 0.8, \gamma = 0.9$
- Second variant: $\alpha = .1, \beta = 0.8, \gamma = 0.9, \alpha_{max} = .0005$
- SGP: $\beta = 0.8, \theta = 0.7, \alpha_{min} = .001, \alpha_{max} = 0.1, \tau = .9$
- Solodov-Tseng: $\alpha_{-1} = .01, \theta = 1.5, \rho = 0.9, \beta = 0.7$
- He-Goldstein variant: $\gamma = 0.8, \tau = 2, \beta = 0.8, \beta_L = 1/(2 * \tau), \beta_U = 25$
- Hyperplane: $\epsilon = 0.5, \hat{\alpha} = .01, \tilde{\alpha} = .05$

## Example 4

| N=100 | $L_2$ Error | $\epsilon$ |
|---|---|---|
| Gradient Projection | $4.80e-4$ | 4.0e-5 |
| SGP | $3.57e-4$ | 4.0e-5 |
| Marcotte | $2.37e-4$ | 3.0e-5 |
| Solodov-Tseng | $3.56e-4$ | 3.0e-5 |
| He-Goldstein variant | $1.10e-5$ | 4.0e-7 |
| Hyperplane | $5.57e-5$ | 4.0e-6 |

- Gradient tolerance $= 1.0e-10$
- Parameters for examples:
- Marcotte: $\alpha = .19, \beta = 0.7$
- First variant: $\alpha = .1, \beta = 0.8, \gamma = 0.9$
- Second variant: $\alpha = .1, \beta = 0.8, \gamma = 0.9, \alpha_{max} = .0005$
- SGP: $\beta = 0.8, \theta = 0.9, \alpha_{min} = .001, \alpha_{max} = 1000, \tau = .9,$
- Solodov-Tseng: $\alpha_{-1} = .01, \theta = 1.5, \rho = 0.9, \beta = 0.7$
- He-Goldstein variant: $\gamma = 0.8, \tau = 2, \beta = 0.8, \beta_L = 1/(2 * \tau), \beta_U = 25$
- Hyperplane: $\epsilon = 1.0e-4, \hat{\alpha} = .001, \tilde{\alpha} = .05$

## 5.2 Analysis

In this chapter, we compare the results for each method and discuss the benefits and potential problems of each method as well as parameter selection. We start with the Marcotte methods. Of the three methods presented, the first variant is the best, while the second variant is generally slightly worse, while the simple Marcotte method performs generally worse. Visually it is nearly impossible to distinguish between the resulting coefficients of each method (See Appendix). This can be explained by the way that the sequence of $\alpha_k$ are chosen.

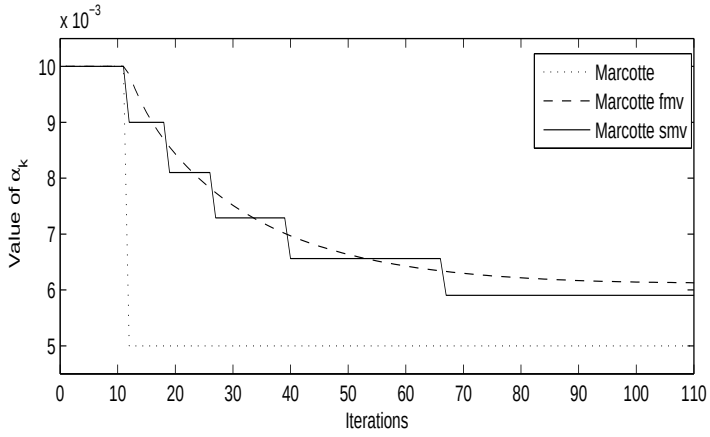


Figure 5.1: $\alpha_k$ for Various Marcotte Methods

Although these methods perform better than gradient projection, the scaled gradient projection method easily out-performs the Marcotte methods. However, we expect this because the scaled gradient projection method uses the Hessian to improve convergence. However, we also need to consider the cost of computing the Hessian (the main diagonal entries using the finite difference method). Although SGP finds solutions with fewer iterations, we look into the extragradient methods to improve the runtime even though some of the methods (i.e. Marcotte methods) will require more iterations to converge. Rather than writing the codes to stop after a fixed iteration, we modified several codes to stop when the gradient reached a certain tolerance (generally $10^{-8}$). This was an excellent way to balance the cost-benefits of each method, and to fully understand their convergence properties. This is especially important in the SGP method where there are spikes in the gradient

(due to spikes in $\alpha$).

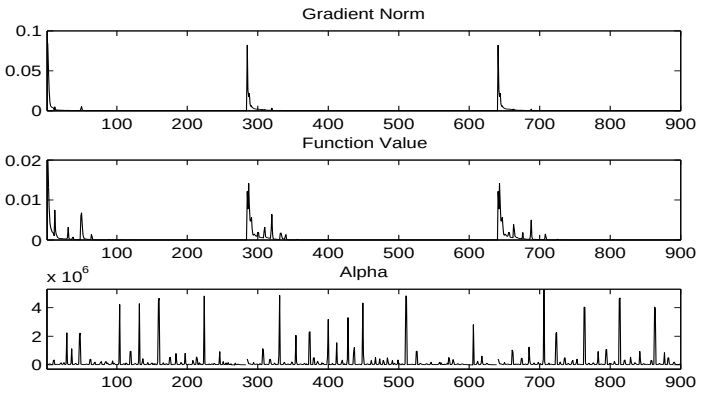


Figure 5.2: Spikes in the Scaled Gradient Projection Method

The Solodov-Tseng method is not as good as the SGP method for several important reasons. First, the scaling matrix $M$ is not updated at each iterate, second, the Solodov-Tseng method required several parameters for initialization. These parameters are considerably harder to determine than other methods. Generally speaking, choosing poor parameters results in either an error, or forces you to take a really high regularization parameter which takes away from the accuracy of the coefficient. A low $\alpha_{-1}$ is most influential in affecting the regularization parameter. In some cases, the S-T method converges fast, but obtains less accuracy than other methods.

The He-Goldstein variant obtains similar accuracy as the S-T method, however, the He-Goldstein variant is a much better algorithm. First and foremost, there are less parameters to guess. Second, these parameters are less sensitive. Overall, the He-Goldstein variant is a good algorithm.

The last method to discuss is the hyperplane method. It quickly approaches the solution however, there are instances when the algorithm gets "stuck" near the hyperplane. Specifically, when $\bar{A}^k$ is near the hyperplane, then the difference between $A^k$ and $A^{k+1}$ is minimal. Sometimes, the sequence of $\alpha's$ or $\eta's$ stops being adaptive and the solution begins to converge slowly. As in Example 1, we obtained convergence ($grad = 10^{-10}$) in 5465 iterations, however, at 1000 iterations, the solution doesn't seem to be converging very fast. Overall, the method is best at 10000 iterations.

# Chapter 6

# Methods for Image Denoising

In this chapter, we explore the following methods for image denoising:

1. Projected Landweber Method
2. ISRA Method (Scaled gradient projection)
3. Khobotov Extragradient Using Marcotte Rules
4. Solodov-Tseng Method

Throughout the chapter, we are going to use images and techniques from [2]. The author provides a detailed analysis of conjugate gradient least squares method which we can use for comparisons.

## 6.1 Iterative Methods

Now we consider the least squares approach to nonnegative image de-blurring.

$$\min_{m \geq 0} J_d(m) = \frac{1}{2}\|Gm - d\|_2^2, \tag{6.1}$$

where G is a blurring operator that introduces Gaussian noise into the system and d is the blurred image. We also add a small amount of random noise into d. It is important that G is very sparse. For a picture of size 200 by 200 pixels, a dense matrix would require about 13 gigabytes of storage (the matrix is $200^2 \times 200^2$) but a matrix with less than .1 percent nonzero elements, which

is sufficient to add blurring to the system, only requires about 12 megabytes of storage [2].
To set up this system as a variational inequality, we need know $\nabla J_d(m)$.

$$\nabla J_d(m) = G^T(Gm - d). \tag{6.2}$$

**Landweber Iteration:**
This method is very simple and is given in [2].

$$m^{k+1} = m^k - \omega G^T(Gm^k - d). \tag{6.3}$$

We can use projections on this method:

$$m^{k+1} = P_C(m^k - \omega G^T(Gm^k - d)). \tag{6.4}$$

This suggests that we implement the projected Landweber (PL) method with inactive constraints which enables our codes to solve equation 6.3.

**ISRA Method:**
[10] suggests another common technique. The ISRA method has semi-convergence. Since regularization is not built in to this method, we still obtain good results in relatively few number of iterations. The method is equivalent to the scaled gradient projection method of using the Hessian

$$m^{k+1} = m^k - \frac{m^k}{G^T G m^k}\nabla J_d(m^k) = m^k \frac{G^T d}{G^T G m^k}, \tag{6.5}$$

where multiplication and division of vectors are in the Hadamard sense, i.e. pixel by pixel.

PL Method

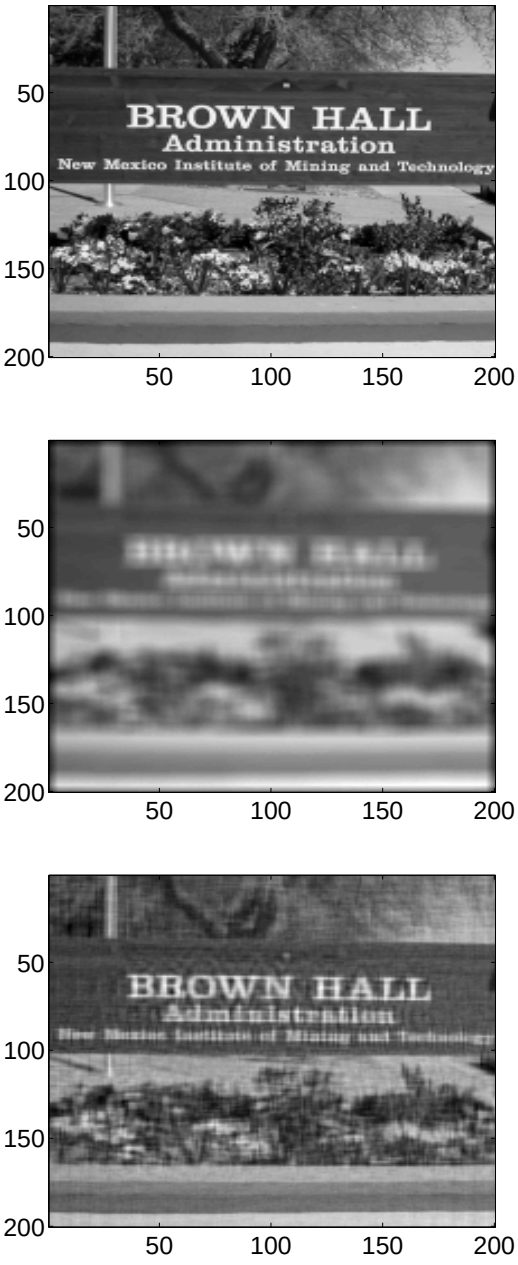


Figure 6.1: PL Method: Original (top), blurred (middle), restored (bottom)

ISRA Method

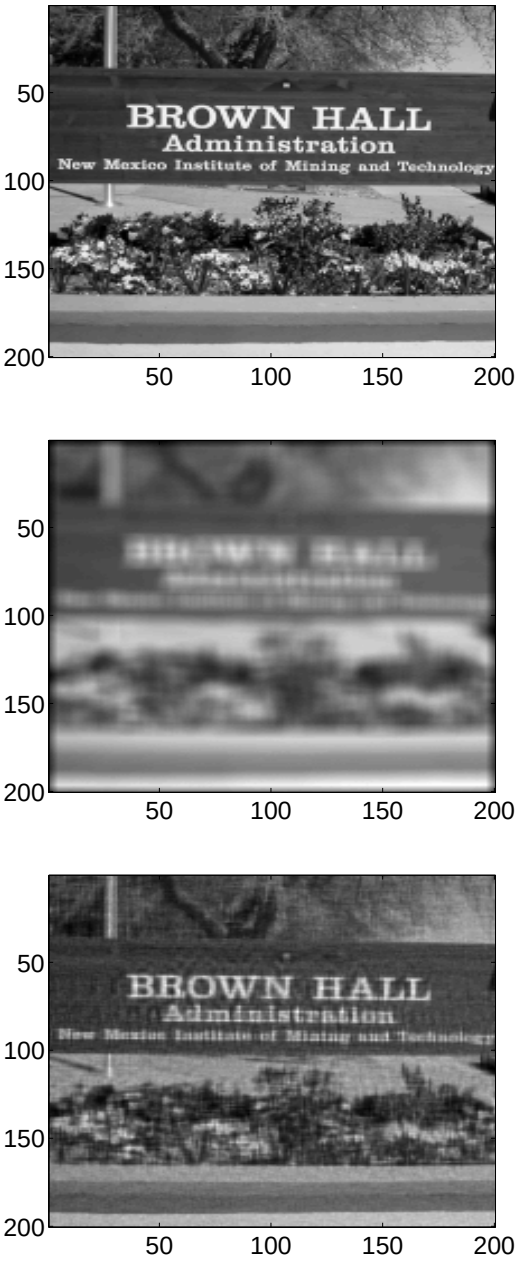


Figure 6.2: ISRA: Original (top), blurred (middle), restored (bottom)

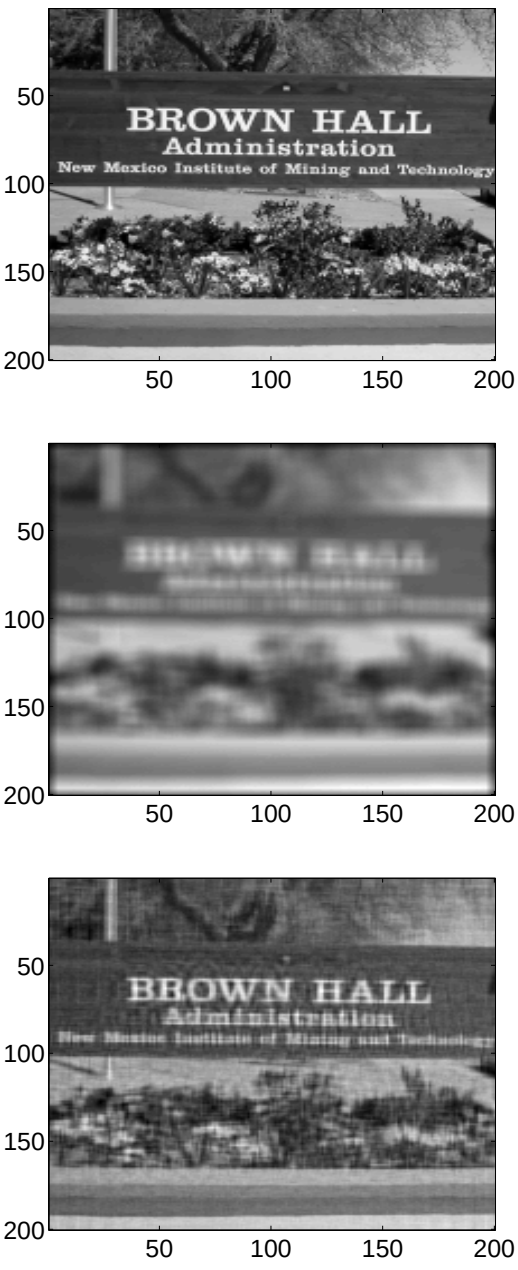


Figure 6.3: Marcotte: Original (top), blurred (middle), restored (bottom)

Marcotte Method: First Variant

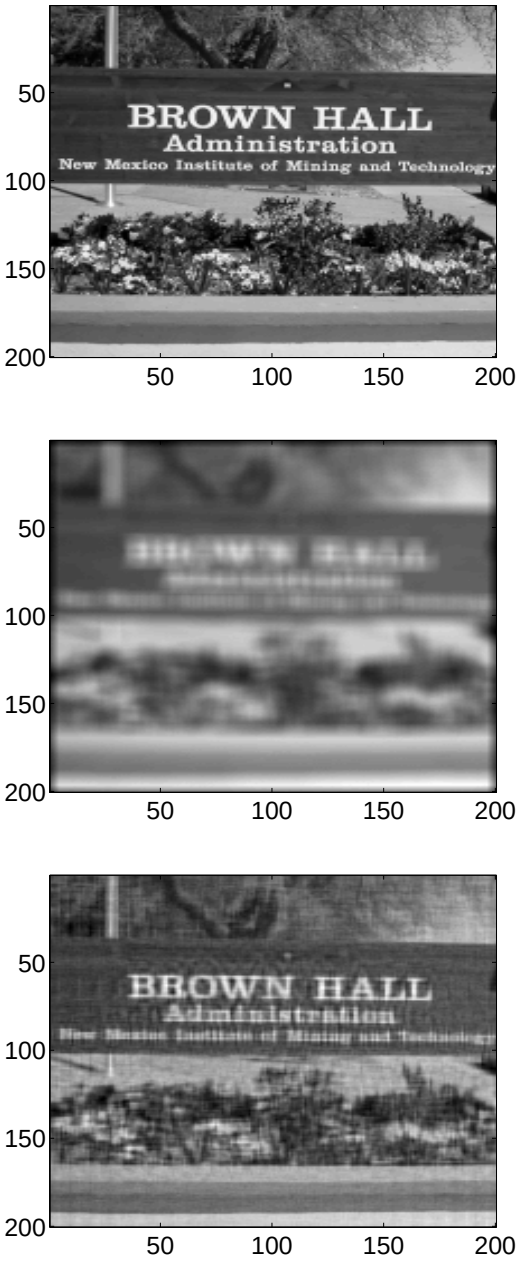


Figure 6.4: FMV: Original (top), blurred (middle), restored (bottom)

Marcotte Method: Second Variant

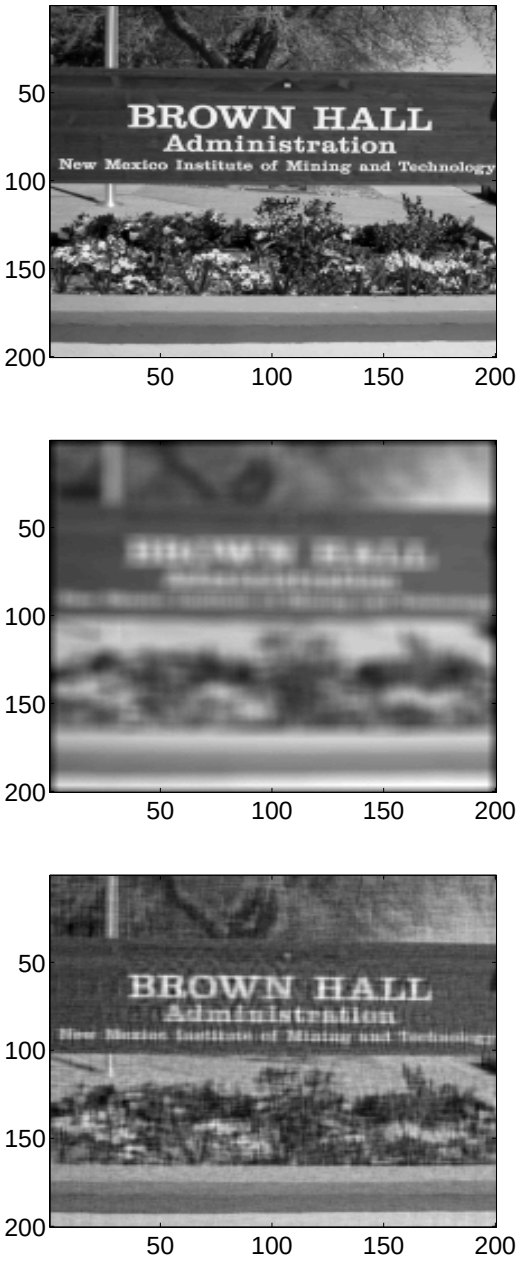


Figure 6.5: SMV: Original (top), blurred (middle), restored (bottom)

Solodov-Tseng Method

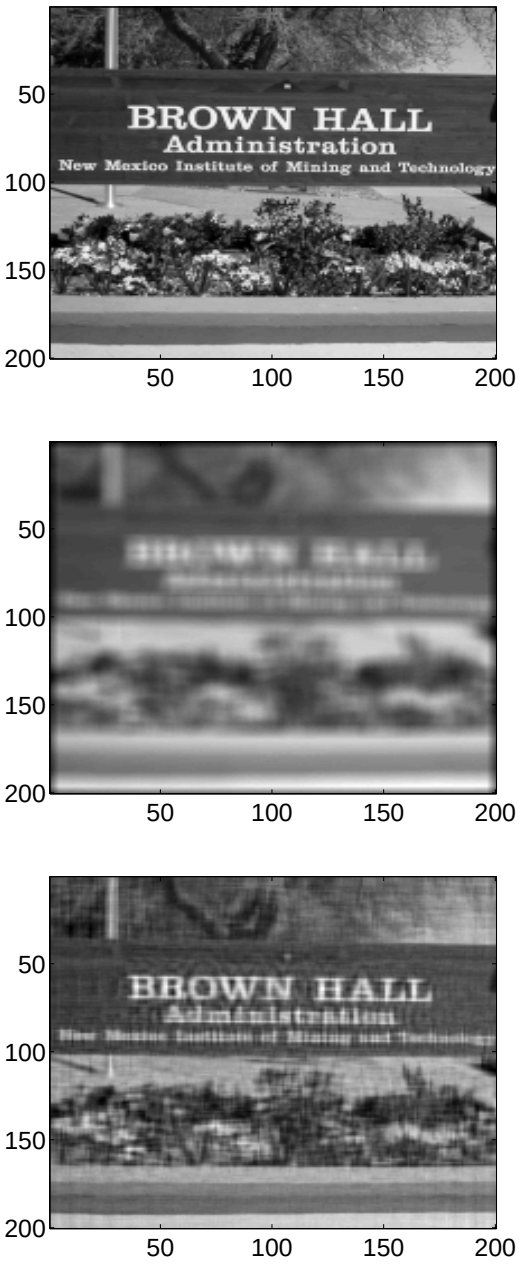


Figure 6.6: S-T: Original (top), blurred (middle), restored (bottom)

ISRA Method

Figure 6.7: ISRA: Original (top), blurred (middle), restored (bottom)

ISRA Method


Figure 6.8: ISRA: Original (top), blurred (middle), restored (bottom)

# Bibliography


[1] R. Acar, Identification of the coefficient in elliptic equations, *SIAM J. Control Optim.*, **31** (1993), 1221–1244.

[2] R. C. Aster, B. Brian and H.T. Clifford, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press 2005.

[3] D.P. Bersekas, *On the Goldstein-Levitin-Polyak Gradient Projection Method.* IEEE Transactions on Automatic Control, Vol. AC-21, No. 2, pp. 174-184, April 1976.

[4] D. P. Bersekas and J.N. Tsitsikis, *Parallel and distributed Computation, Numerical Method*, Prentice-Hall, London 1989.

[5] D. P. Bertsekas, *Nonlinear Programming, Second Edition.* Authena Scientific, 1999.

[6] A. Bnouhachem, X.L. Fu, M.H. Xu and S. Zhaohan, *Modified extragradient methods for solving variational inequalities.* Computers and Mathematics with Applications, 57, pp. 230-239, Elsevier, 2009.

[7] A. Bnouhachem, X.L. Fu, M.H. Xu and S. Zhaohan, *New extragradient-type methods for solving variational inequalities* Applied Mathematics and Computation 216 pp. 2430-2440, Elsevier, 2010.

[8] A. Bnouhachem, M.A. Noor and Zhang Hao, *Some new extragradient iterative methods for variational inequalities.* Nonlinear Analysis, 70, pp.1321-1329, Elsevier, 2009.

[9] Bonettini, S., R. Zanella, L. Zanni, *A scaled gradient projection method for constrained image deblurring.* IOP Publishing, Inverse Problems 25 (2009) 015002 (23pp).

[10] S. F. Bonettini, F. Benvenuto, R. Zanella, L Zanni and M. Bertero, *Gradient Projection Approaches for Optimization Problems in Image Deblurring and*

*Denoising*. Proceedings of the 17th European Signal Processing Conference (EUSIPCO), 1384-1388, 2009.

[11] L.C. Ceng and J.C. Yao, *An extragradient-like approximation method for variational inequalities and fixed point problems*. Applied Mathematics and Computation, 190, pp. 205-215, Elsevier, 2007.

[12] J. Y. Cruz, J.Y. Bello, A.N. Iusem, *Full convergence of an approximate projection method for nonsmooth variational inequalities*. Mathematics and Computers in Simulation, Elsevier, 2010.

[13] L. F. Demkowicz and J. T. Oden, *Applied Functional Analysis*. CRC Press, 1996.

[14] H. Engl, M. Hanke and A. Neubauer, *Regularization of Inverse Problems* Mathematics and Its Applications, Kluwer Academic Publishers, 2000.

[15] H. Gao, Q. Yang and J. Zhao, *A note on Solodov and Tseng's methods for maximal monotone mappings*. Journal of Computational and Applied Mathematics, 234, pp. 1522-1527, Elsevier, 2010.

[16] M.S. Gockenback and A. A. Khan, *An Abstract Framework for Elliptical Inverse Problems: Part 1. An Output Least-Squares Approach*. SAGE Publications, 2007.

[17] D. Han, *A New Class of Projection and Contraction Methods for Solving Variational Inequality Problems*. Computers and Mathematics with Applications, 51, pp. 937-950, Elsevier, 2006.

[18] P.T. Harker, *Accelerating the convergence of teh diagonalization and projection algorithms for finite-dimensional variational inequalities*. Mathematical Programming 41, 1988 (29-59).

[19] Y. He, *A new double projection algorithm for variational inequalities*. Journal of Computational and Applied Mathematics, 185, pp. 166-173, Elsevier, 2006.

[20] A. N. Iusem, R. Luis and R. Perez, *An Extragradient-type Algorithm for Non-Smooth Variational Inequalities*. Optimization, 2000, Vol. 48, pp. 309-332. Overseas Publishers Association, 2000.

[21] P. Katchang and P. Kumam, *A viscosity of extragradient approximation method for finding equilibriu, problems, variational inequalities and finxed point problems for nonexpansive mappings*. Nonlinear Analysis: Hybrid Systems 3, pp. 475-486, Elsevier, 2009.

[22] C. T. Kelly, *Iterative Methods for Optimization*. SIAM Frontiers in Applied Mathematics, 1999.

[23] E. N. Khobotov, *Modification of the Extra-gradient Method for Solving Variational Inequalities and Certain Optimization Problems*. Pergamon Press plc, 1989.

[24] D. Kinderleher and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. SIAM Classics In Applied Mathematics, 31, 2000.

[25] I.V. Konnov, S. Kum and G. M. Lee, *On convergence of descent methods for variational inequalities in Hilbert space*. Mathematical Methods of Operations Research, 55, pp. 371-382, 2002.

[26] G. M. Korpelevich, *The Extragradient Method For Finding Saddle Points and Other Problems*. Nauka Publishers (USSR), 1976.

[27] M. Li, L.Z. Liao and X.M. Yuan, *Some Goldstein's type methods for cocoervice variant variational inequalities*. Applied Numerical Mathematics, 61, pp. 216-228, Elsevier, 2011.

[28] M. Li and X.M. Yuan, *An APPA-based descent method with optimal stepsizes for monotone variational inequalities*. European Journal of Operational Research, 186, pp. 486-495, Elsevier, 2008.

[29] L.Z. Liao, B. He and X. Wang, *Steplengths in the extragradient type methods*. Journal of Computational and Applied Mathematics, 233, pp. 2925-2939, Elsevier, 2010.

[30] P. Marcotte, *Application of Khobotov's algorithm to variational inequalities and network equilibrium problems*, INFORM, 29 (1991).

[31] J.J. More, B.S. Garbow and K. E. Hillstrm, *Testing Unconstrained Optimization Software*, ACM Transactions on Mathematical Software, 7:1, pp 17-41, 1981.

[32] D. Poole, *Linear Algebra: A Modern Introduction, Second Ed.* Thomson Brooks/ Cole, Canada 2006.

[33] L.D. Popov, *Schemes for Generation of a Leading Sequence in Regularized Extragradient Solution Method for Variational Inequalities*. Russian Mathematics (Iz. VUZ), Vol. 48, No. 1, pp. 67-76, 2004.

[34] M.V. Solodov and P. Tseng, *Modified projection type methods for monotone variational inequalities*. SIAM Journal of Control Optimization, Vol. 34, N.5, 1996 (1914-1830).

[35] M.V. Solodov, *Convergence Analysis of Perturbed Feasible Descent Methods*. Journal of Optimization Theory and Applications, Vol. 93, No. 2, pp. 337-353, May 1997.

[36] P. Tseng, *On linear convergence of iterative methods for the variational inequality problem*. Journal of Computational and Applied Mathematics, 60, pp. 237-252, Elsevier, 1995.

[37] F. Tinti, *Numberical Solution for Pseudomonotone Variational Inequality Problems by Extragradient Methods*. Italian FIRB Project, Grant n. RBAU01JYPN, 2003.

[38] C.R. Vogel, *Computational Methods for Inverse Problems*. Frontiers in Applied Mathematics, SIAM, 2002.

[39] Y.J. Wang, N.H. Xiu and J.Z. Zhang, *Modified Extragradient Method for Variational Inequalities and Verification of Solution Existance*. Plenum Publishing Corporation, 2003.

[40] Y. Wang, N. Xiu and C. Wang, *A New Version of Extragradient Method for Variational Inequality Problems*. Computers and Mathematics with Applications, 42, pp. 969-979, Pergamon, 2001.

[41] N.H. Xui and Z. Zhang, *Local Convergence Analysis of Projection-Type Algorithms: Unified Approach*. Plenum Publishing Corporation, 2002.

[42] R. Zanella, P. Boccacci, L. Zanni, and M. Bertero, *Efficient gradient projection methods for edge-preserving removal of Poisson noise*. IOP Publishing, Inverse Problems 25 (2009) 045010 (24pp).

[43] L. Zanni, *On the Convergence Rate of Two Projection Methods for Variational Inequalities in RN*. University of Modena, Italy, 1991.