8-11-2014

Continuous Methods for Elliptic Inverse Problems

Corinne Teravainen

Continuous Methods for Elliptic Inverse Problems

By

Corinne Teravainen

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
College of Science
Rochester Institute of Technology

August 11, 2014

Advisor: Dr. Akhtar Khan

Co-Advisor: Dr. Baasansuren Jadamba

Committee members: Prof. Patricia Clark

Dr. Ephraim Agyingi

Dedicated to my loving, supportive family.

Abstract

Numerous mathematical models in applied and industrial mathematics take the form of a partial differential equation involving certain variable coefficients. These coefficients are known and they often 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 coefficient when a certain measurement of a solution of the partial differential equation is available. A commonly used approach to inverse problems is to solve an optimization problem whose solution is an approximation of the sought coefficient. Such optimization problems are typically solved by discrete iterative schemes. It turns out that most known iterative schemes have their continuous counterparts given in terms of dynamical systems. However, such differential equations are usually solved by specific differential equation solvers. The primary objective of this thesis is to test the feasibility of differential equations based solvers for solving elliptic inverse problems. We will use differential equation solvers such as Eulers Method, Trapezoidal Method, Runge-Kutta Methods and Adams-Bashforth Method. In addition, these solvers will also be compared to built-in MATLAB ODE solvers. The performance and accuracy of these methods to solve inverse problems will be thoroughly discussed.

Acknowledgements

I would like to thank my advisor, Dr. Akhtar Khan for all of his help through this process. His expertise and encouragement made the final step of my degree possible. Dr. Khan is a kind, caring professor who truly goes above and beyond for his students. I sincerely thank you, Dr. Khan.

In addition, I would like to thank the School of Mathematical Sciences at RIT, which is a wonderful, enriching program. My co-advisor and committee members, Dr. Baasansuren Jadamba, Prof. Patricia Clark, and Dr. Ephraim Agyingi, thank you for your assistance as well.

And finally, thank you to my parents for their continuous support and motivation throughout the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Inverse Problems

This chapter introduces the inverse problem. Within it, the problem will be defined and some simple examples will demonstrate the difficulties that arise in this problem. In addition, real world problems that contain inverse problems will be discussed, along with the main techniques available to solve them. And finally, the main aim and approach of this thesis will be identified.

1.1 Definition of Inverse Problems

Normally, cause and effect are studied in that order. Inverse problems begin by observing the effects and then try to determine the unknown causes it began with. Alternatively, the more common direct problem looks for the effects based on the information known about the causes.

Beginning with the following model in the vector space setting,

$$F(x) = y,$$

F is an operator that relates x and y. Here x, the physical parameters, are related to a set of data, y. With regard to the inverse problem at hand, this model is approached by estimating x using some measurement of y.

When the operator F is linear, the inverse problem in turn is in fact linear. Alternatively, the problem is nonlinear, whereas nonlinear inverse problems are considerably more difficult to solve. When collecting data, the data y is usually disrupted by some amount of noise. What we in fact have in this problem is:

$$y = y_{real} + \eta$$

where y_{real} satisfies $F(x) = y$ with x equal to x_{real} and η is clearly the error.

In inverse problems, finding a solution x for a very small η that influenced the data can have little or no correspondence at all to x_{real}. This is one glaring example of how direct problems and inverse problems behave very differently.

Most inverse problems are *ill-posed* because of their special characteristics. Jacques Hadamard, a mathematician known for his contribution to partial differential equations (among other topics), defined a problem to be *well-posed* if it had the following properties:

1. **Existence:** For a suitable data set, the problem has a solution.
2. **Uniqueness:** The solution is unique.
3. **Stability:** The solution depends continuously on the observations.

If any one of the above features is not met, then the problem is ill-posed. In the field of inverse problems, the main concern is its inability to meet the third condition.

The focus of this thesis is on the study of inverse problems of identifying physical parameters that appear in elliptic partial differential equations. This particular problem is also referred to as distributed parameter identification problems.

In the next section, we provide some examples of ill-posed problems' instability.

1.2 Ill-posedness: Examples

One characteristic of ill-posed problems is that the error between the exact and the noisy solution can be arbitrarily large even when the error in the data can be kept arbitrarily small. This situation is demonstrated in the following example.

1.2.1 Example 1: Differentiation of noisy functions

For this example we will compute the derivative of a noisy function. That is, instead of a function f, the noisy function f_δ is present and we wish to compute the derivative $\frac{df_\delta}{dx}$. Assume that

$$\begin{aligned} f_\delta(x) &= f(x) + e_\delta(x), \quad x \in S := [0,1] \\ f_\delta(0) &= f(0) = 0 \\ f_\delta(1) &= f(1) = 0 \end{aligned}$$

where $e_\delta(x)$ represents the data noise.
We choose

$$e_\delta = \sqrt{2}\delta \sin(2\pi n x)$$

for $n \in \mathbb{N}$. Notice that, for all n,

$$\int_0^1 |e_\delta(x)|^2 dx = \delta^2.$$

Moreover,

$$\frac{df_\delta}{dx}(x) = \frac{df}{dx}(x) + 2\sqrt{2}\pi\delta n \cos(2n\pi x).$$

Consequently, the $L_2(S)$ and the $L_\infty(S)$ errors, given by

$$\begin{aligned} \left\| \frac{df_\delta}{dx}(x) - \frac{df}{dx}(x) \right\|_{L_2(S)} &= 2\pi\delta n, \\ \left\| \frac{df_\delta}{dx}(x) - \frac{df}{dx}(x) \right\|_{L_\infty(S)} &= 2\sqrt{2}\pi\delta n, \end{aligned}$$

can be made arbitrarily large by choosing n large enough. □

A similar behavior is shown by the numerical differentiation in the next example.

1.2.2 Example 2: Numerical differentiation

Consider the following boundary value problem: Find $y : [0,1] \to \mathbb{R}$ to solve

$$\begin{aligned} -\frac{d^2y}{dx^2} &= f(x), \quad 0 < x < 1, \\ y(0) &= 0, \\ y(1) &= 0. \end{aligned}$$

The inverse problem we are interested in is:

Given $y : [0,1] \to \mathbb{R}$ such that $y(0) = y(1) = 0$, compute $f(x) = -y''(x)$.

To solve the above inverse problem, we discretize the above BVP by using a finite difference scheme. We establish a regular grid on the interval $[0,1]$ by defining $x_i = ih$, $i = 0, 1, \ldots, n$, $h = 1/n$. Then, restricting the differential equation $-y'' = f(x)$ to the grid points, we obtain

$$-y''(x_i) = \frac{-y(x_i - h) + 2y(x_i) - y(x_i + h)}{h^2} + O(h^2), \quad i = 1, 2, \ldots, n-1.$$

In the above, we have employed the central difference scheme and assumed that $y \in C^4[0,1]$.
The above equations can also be represented in a matrix-vector form,

$$Ly = f,$$

where

$$L = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} \in \mathbb{R}^{n-1}, \quad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{bmatrix} \in \mathbb{R}^{n-1},$$

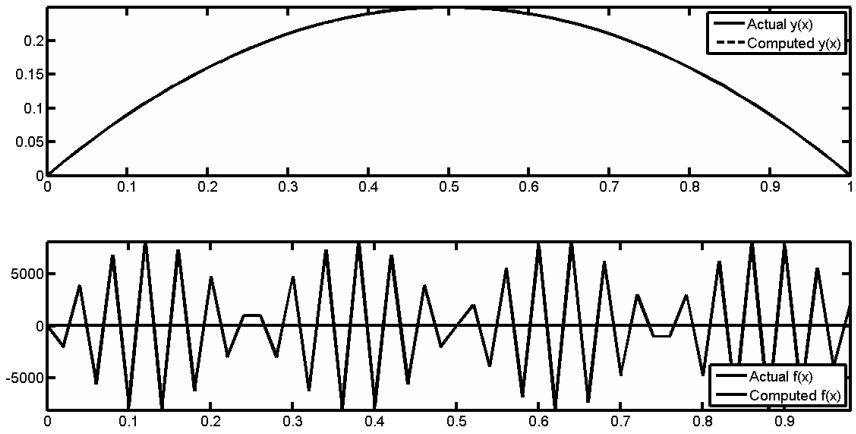
To depict the ill-posedness, we choose (discrete versions of)

$$\begin{aligned} y(x) &= x - x^2 \\ f(x) &= 2 \end{aligned} \tag{1.1}$$

For $n = 50$, we solve both the forward and the inverse problems. In each case, we add normally and independently distributed errors to the components of the data, and then compute the solution. The results are shown in Figure 1.1. As the graphs show, the noisy data does not significantly affect

the computed solution to the forward problem. On the other hand, the computed solution to the inverse problem is essentially useless. □

Figure 1.1: Exact and the computed solution for forward problem (top) and the inverse problem (bottom).



1.3 Explanation of the Problem

In this thesis we study the following two inverse problems:

1.3.1 Boundary Value Problem of Second-Order

A second-order boundary value problem is of the form:

$$\begin{aligned} -\frac{d}{dx}\left(a(x)\frac{d}{dx}u(x)\right) &= f(x) \quad x \in (0,1) \\ u(0) &= 0 \\ u(1) &= 0 \end{aligned} \tag{1.2}$$

The goal of inverse problems is to solve for the coefficient, $a(x)$, given a measurement of the solution $u(x)$, called z. This particular boundary value problem has Dirichlet conditions. The usual problem that is attempted, called the direct problem, is to solve for $u(x)$ given $a(x)$ and $f(x)$.

We know that the direct problem is well-posed, but the inverse problem is ill-posed. Starting from equation (1.1), to solve for $a(x)$ we can simply integrate both sides and then divide by $u(x)$ to obtain

$$a(x) = -\frac{1}{u'(x)} \int f(x).$$

As shown above, the coefficient $a(x)$ is not defined when $u'(x) = 0$.
Thus, in order to solve this type of problem, we first pose it as a minimization problem.

1.3.2 Boundary Value Problem of Fourth-Order

A fourth-order boundary-valued problem:

$$\begin{aligned} (a(x)u'')'' &= f(x), \quad \forall\, x \in \Omega, && \text{(1.3a)} \\ u(0) &= u'(0) = 0, && \text{(1.3b)} \\ u(1) &= u'(1) = 0, && \text{(1.3c)} \end{aligned}$$

where $\Omega = (0, 1)$, $a(x)$ is a variable coefficient and f is a suitable function.

The weak form of (1.3) reads: Find $u \in V$ such that

$$\langle a(x)u'', v'' \rangle = \langle f, v \rangle, \quad \text{for all } v \in V. \tag{1.4}$$

1.4 The Finite Element Method

The finite element method is needed to solve both the direct and inverse problem. It is made up of three parts:

1. The boundary value problem is written in its weak or variational form. This expresses the boundary value problem as infinitely many equations.

2. The Galerkin Method is then applied to solve the weak-form on a finite-dimensional subspace. Thus leading to a linear system.

3. A basis of piecewise polynomials is chosen for the finite-dimensional subspace.

1.4.1 The Weak Form

First, the linear space below must be defined, over the domain $\Omega = (0, 1)$.

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

To get to the weak form (or variational form) of the boundary value problem, begin by multiplying (1.1) by a test function $v \in V$. Then, integrate both sides using integration by parts.

$$\begin{aligned} -\frac{d}{dx}\left(a(x)\frac{d}{dx}u(x)\right) &= f(x) \\ -\int_0^1 (au')' v &= \int_0^1 fv \\ \left[-au'\right]_0^1 + \int_0^1 au'v' &= \int_0^1 fv \\ \int_0^1 au'v' &= \int_0^1 fv \end{aligned}$$

We now have the weak form,

$$T(a, u, v) = m(v) \quad \forall u, v \in V, \tag{1.5}$$

where

$$T(a, u, v) = \int_0^1 au'v' \tag{1.6a}$$

$$m(v) = \int_0^1 fv \tag{1.6b}$$

It is clear that $T(\cdot, \cdot, \cdot)$ is a trilinear form which is symmetric in u and v.For constants $\alpha, \beta > 0$, if the following conditions are held

$$T(a, u, v) \leq \alpha ||u|| \ ||v|| \tag{1.7a}$$

$$T(a, u, v) \geq \beta ||u||^2 \tag{1.7b}$$

then by the Lax-Milgram Lemma, the variational form is uniquely solvable.

1.5 Optimization Formulation of the Inverse Problem

The goal is to minimize the norm of the difference between $u(a)$ and z. This will give the most accurate coefficient $a(x)$.
There are several optimization approaches to compute this, but we will explore three of them:

1. Output Least-Squares (OLS)
2. Modified Output Least-Squares (MOLS)
3. Equation Error

Each approach aims to minimize the functional, J_i (i=1,2,3), shown below.

$$J_1(a) = \frac{1}{2}||u(a) - z||^2 \tag{1.8a}$$

$$J_2(a) = \frac{1}{2}\int_0^1 a(x)\left[\frac{d}{dx}\left(u(a) - z\right)\right]^2 dx \tag{1.8b}$$

$$J_3(a) = ||e(z,a)||^2_{H^1(\Omega)}, \tag{1.8c}$$

where $e(u,v)$ is such that

$$\langle e(u,a), v\rangle = T(a,u,v) - m(v), \quad \forall\ v \in V.$$

The functionals $J_1(a)$, $J_2(a)$ and $J_3(a)$ are the OLS, MOLS and Equation Error formulation, respectively. In the above, $||\cdot||$ represents a suitable norm and $u(a)$ is the solution that is obtained by solving the weak form for $a(x)$, the coefficient. A more complete explanation of these optimization approaches will be discussed later in the chapter.

In view of the ill-posed nature of the problem, the regularization of these functional is needed. Putting these two parts together, we now have

$$\min_{a\in\tilde{A}} J(a) + \epsilon R(a), \tag{1.9}$$

Where $R(a)$ is the regularization term $\epsilon > 0$ is the regularization parameter and $\tilde{A}$ is the set of all feasible coefficients. $J(a)$ can be either $J_1(a)$ or $J_2(a)$

or $J_3(a)$.
From here, we write the problem as a variational inequality, where the goal is to find $a^* \in \tilde{A}$.

$$\langle J'(a^*), a - a^* \rangle \geq 0 \quad \forall a \in \tilde{A} \tag{1.10}$$

The above is a necessary optimality condition for the minimization problem.

1.6 Finite Element Discretization

Now let V_n be the finite dimensional subspace of V which gives

$$T(a, u, v) = m(v) \quad \forall v \in V_n, \tag{1.11}$$

where u_n is a solution to the variational form. Next the aim to re-write the boundary value problem in matrix form. Let's begin with the bases of V_n, which will be denoted by

$$\{\phi_1, \phi_2, \phi_3, \ldots, \phi_n\}.$$

Substituting into (2.4) when $j = 1, 2, \ldots, n$ the following is known

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

Also, define the following

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

The solution u_n is unknown, but with the above definition, finding u_n is equivalent to finding the coefficients: U_i.
Finally, the stiffness matrix, K and load vector F are,

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

Bringing these three elements together, the matrix form of the variational problem is as follows

$$KU = F \tag{1.12}$$

Now the objective is to construct a finite dimensional subspace V_n consisting of piecewise linear functions.
To define V_n, let

$$0 = x_0 < x_1 < x_2 < \ldots < x_j < \ldots < x_n = 1$$

The points $x_0, x_1, \ldots, x_n$ are the nodes of the mesh we are creating. We are using a regular mesh, with $x_i = ih, \;\; h = \frac{1}{n}$. This makes a mesh that contains n-subintervals,

$$I_j = (x_{j-1}, x_j), \quad j = 1, 2, \ldots, n.$$

A function $p : [0, 1] \to R$ is piecewise linear if, for each $i = 1, 2, \ldots, n$, there exists constants a_i, b_i with

$$p(x) = a_i x + b_i \;\; \forall x \in (x_{i-1}, x_i)$$

Thus, we define, for a fixed mesh on $[0, 1]$,

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

The space V_n should have the following features:

1. Piecewise polynomials are easy to manipulate.
2. Smooth functions can be well approximated by piecewise linear functions.
3. The basis for V_n leads to a sparse K.

Thus, we must now construct a basis for V_n that upholds the third feature. A piecewise linear function can be determined in its entirety by its nodal values. For $n = 1, 2, \ldots, n-1$, define $\phi_i \in V_n$ to be a piecewise linear function satisfying the following

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

Additionally, each $p \in V_n$ satisfies

$$p(x) = \sum_{i=1}^{n-1} p(x_i)\phi_i(x)$$

And it is clear that every $p \in V_n$ can be written as a linear combination of $\{\phi_1, \phi_2, \ldots, \phi_n - 1\}$; this set spans V_n.
Recall now that the entries of the stiffness matrix K are

$$K_{ij} = T(a, \phi_j, \phi_i), \quad i = 1, 2, \ldots, n-1$$

Since most of the ϕ_i are zero on the interval $[0, 1]$, most of the values $T(a, \phi_j, \phi_i) = 0$ as well. The only place that ϕ_i is not zero is on the interval $[x_{i-1}, x_{i+1}]$. In other words, ϕ_i has the support $[x_{i-1}, x_{i+1}]$.
Thus,

$$\begin{aligned} T(a, \phi_{i-1}, \phi_i) &\neq 0 \\ T(a, \phi_i, \phi_i) &\neq 0 \\ T(a, \phi_{i+1}, \phi_i) &\neq 0 \end{aligned}$$

On the other hand, $|j - 1| > 1$ and thus $T(a, \phi_j, \phi_i) = 0$. In that case, for any $x \in [0, 1]$ we have that either $\frac{d\phi_i}{dx} = 0$ or $\frac{d\phi_j}{dx} = 0$. Therefore, the matrix K is tridiagonal.
The values of ϕ_j on (x_{j-1}, x_j) are computed by using the equation of a line segment. Given below,

$$\phi_j(x) = \begin{cases} \frac{1}{h}(x - x_{j-1}) & \text{for} \quad x \in (x_{j-1}, x_j) \\ -\frac{1}{h}(x - x_{j+1}) & \text{for} \quad x \in (x_j, x_{j+1}) \\ 0 & \text{otherwise} \end{cases}$$

From this we can easily compute the derivative of ϕ_j, which we will need in the computation of K, in 2.3.2.

$$\phi_j(x)' = \begin{cases} \frac{1}{h} & \text{for} \quad x \in (x_{j-1}, x_j) \\ -\frac{1}{h} & \text{for} \quad x \in (x_j, x_{j+1}) \\ 0 & \text{otherwise} \end{cases}$$

1.6.1 Computation of K and F

Load Vector

Using Simpson's Rule, shown here:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

the load vector, F, can be approximated below. Recall that we begin with,

$$\begin{aligned} F_j &= m(\phi_j) = \int_{x_{j-1}}^{x_{j+1}} f\phi_j \\ &= \frac{1}{h}\int_{x_{j-1}}^{x_j}(x - x_{j-1})f(x)dx - \frac{1}{h}\int_{x_j}^{x_{j+1}}(x - x_{j+1})f(x)dx \end{aligned}$$

$$\begin{aligned} &= \frac{h}{6h}\left[(x_{j-1} - x_{j-1})f_{j-1} + 4\left(\frac{x_{j-1}+x_j}{2} - x_{j-1}\right)f\left(\frac{x_{j-1}+x_j}{2}\right)\right] \\ &- \frac{h}{6h}\left[-(x_j - x_{j-1})f_j + (x_j - x_{j+1})f_j\right] \\ &- \frac{h}{6h}\left[4\left(\frac{x_j+x_{j+1}}{2} - x_{j+1}\right)f\left(\frac{x_j+x_{j+1}}{2}\right) + (x_{j+1} - x_{j+1})f_{j+1}\right] \\ &= \frac{1}{6}\left[4\left(\frac{x_j - x_{j-1}}{2}\right)f\left(\frac{x_{j-1}+x_j}{2}\right) + hf_j + hf_j\right] \\ &+ \frac{1}{6}\left[4\left(\frac{x_{j+1} - x_j}{2}\right)f\left(\frac{x_j+x_{j+1}}{2}\right)\right] \\ &= \frac{1}{6}\left[2hf\left(\frac{x_{j-1}+x_j}{2}\right) + 2hf_j + 2hf\left(\frac{x_j+x_{j+1}}{2}\right)\right] \\ &= \frac{h}{6}\left[2f\left(\frac{x_{j-1}+x_j}{2}\right) + 2f_j + 2f\left(\frac{x_j+x_{j+1}}{2}\right)\right] \\ &= \frac{h}{6}\left[f_{j-1} + f_j + 2f_j + f_j + f_{j+1}\right] \end{aligned}$$

Thus, we have the load vector as follows:

$$F_j = \frac{h}{6}\left[f_{j-1} + 4f_j + f_{j+1}\right]$$

Stiffness Matrix

Again, we will use Simpson's rule. The stiffness matrix, K, can be computed below. Recall we begin with,

$$\begin{aligned} K_{j,j} = T(a, \phi_j, \phi_j) &= \int_{x_{j-1}}^{x_{j+1}} a(x)\phi_j'\phi_j' dx \\ &= \frac{1}{h^2}\int_{x_{j-1}}^{x_j} a(x)dx + \frac{1}{h^2}\int_{x_j}^{x_{j+1}} a(x)dx \end{aligned}$$

$$= \frac{1}{h^2}\cdot\frac{h}{6}\left[a_{j-1} + 4a\left(\frac{x_{j-1}+x_j}{2}\right) + 2a_j + 4a\left(\frac{x_j+x_{j+1}}{2}\right) + a_{j+1}\right]$$

$$
\begin{aligned}
&= \frac{1}{6h}\left[a_{j-1} + 2a_{j-1} + 2a_j + 2a_j + 2a_j + 2a_{j+1} + a_{j+1}\right] \\
&= \frac{1}{6h}\left[3a_{j-1} + 6a_j + 3a_{j+1}\right] \\
K_{j,j} &= \frac{1}{2h}\left[a_{j-1} + 2a_j + a_{j+1}\right]
\end{aligned}
$$

$$
\begin{aligned}
K_{j+1,j} = T(a, \phi_{j+1}, \phi_j) &= \int_{x_j}^{x_{j+1}} a(x)\phi'_{j+1}\phi'_j dx \\
&= -\frac{1}{h^2}\int_{x_j}^{x_{j+1}} a(x)dx \\
&= -\frac{1}{h^2}\cdot\frac{h}{6}\left[a_j + 4a\left(\frac{x_j + x_{j+1}}{2}\right) + a_{j+1}\right] \\
&= -\frac{1}{6h}\left[a_j + 2a_j + 2a_{j+1} + a_{j+1}\right] \\
&= -\frac{1}{6h}\left[3a_j + 3a_{j+1}\right] \\
K_{j+1,j} &= -\frac{1}{2h}\left[a_j + a_{j+1}\right]
\end{aligned}
$$

Since K is symmetric,

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

1.7 The Inverse Problem

Recall the inverse problem we are aiming to solve:

$$
\begin{aligned}
-\frac{d}{dx}\left(a(x)\frac{d}{dx}u(x)\right) &= f(x) \quad x \in (0,1) \\
u(0) &= 0 \\
u(1) &= 0
\end{aligned}
$$

The coefficient $a(x)$ is unknown here. Remember that $f(x)$ and a measurement of $u(x)$, called z are known. To find the coefficient, $a(x)$ we will solve

the minimization problem below:

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

Let's explore the output least squares, or OLS, first.

1.7.1 Output Least Squares

Recall that V_n is the finite dimensional subspace of V. We will also be working in the finite dimensional subspace of the coefficient space. The system must be discretized, thus the basis functions being used are shown below:

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

From here, the finite-dimensional form for the OLS objective functional can be obtained

$$\begin{aligned} J_1(A) &= \frac{1}{2}\langle u - z, u - z \rangle + R(A) \\ &= \frac{1}{2}\langle \sum_{i=1}^{N} (U_i - Z_i)\phi_i, \sum_{j=1}^{N} (U_j - Z_j)\phi_j \rangle + R(A) \\ &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (U_i - Z_i)(U_j - Z_j)\langle \phi_i, \phi_j \rangle + R(A) \\ J_1(A) &= \frac{1}{2}(U - Z)^T M (U - Z) + R(A) \end{aligned}$$

Here, the mass matrix denoted by M, is defined as the following:

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

Recall that most of the values of $\langle \phi_i, \phi_j \rangle = 0$, (when $|i-j| > 1$). Thus making M a tridiagonal matrix.

$$\begin{aligned} M_{i,i} &= \langle \phi_i, \phi_i \rangle \quad \text{for} \quad i = 1, \ldots, N \\ &= \int_{x_{i-1}}^{x_{i+1}} \phi_i^2 dx \\ &= \frac{2}{h^2} \int_{x_{i-1}}^{x_i} (x - x_i)^2 dx \\ &= \left(\frac{2}{h^2}\right)\left(\frac{h^3}{3}\right) \\ M_{i,i} &= \frac{2h}{3} \end{aligned}$$

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

In order to use the Differential Equation Solver Method in Chapter 4, we must compute the gradient of the objective functional and also the Hessian, or second derivative. This is where the Adjoint Stiffness Matrix comes into play. We define the adjoint stiffness matrix by the following:

$$L(U)A = K(A)U$$

Recall that $K_{ij} = T(a, \phi_j, \phi_i)$. Thus,

$$K(A) = TA$$

Returning to $L(U)A = K(A)U$, we have:

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

Now we're ready to compute the gradient. To begin, we have $K(A)U = F$, and we will take the derivative of both sides:

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

Using the Adjoint Stiffness Matrix and the derivative of U, OLS can be summed up:
Objective functional:

$$J_1(A) = \frac{1}{2}(U - Z)M(U - Z)$$

Gradient:

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

Hessian:

$$\nabla^2 J_1(A) = B_1^T M B_1 + B_2 + B_2^T$$

$$B_1 = K(A)^1 L(U)$$

$$L_1 = L(K(A)^{-1} M(U - Z))$$

$$B_2 = L_1^T B_1$$

Details on the computation can be found in [23].

1.7.2 Modified Output Least Squares

To begin, our goal is to minimize the functional below:

$$J_2(A) = \frac{1}{2}||U(a) - Z||^2 + R(a)$$

Where $||\cdot||$ is the energy norm.

$$J_2(A) = \frac{1}{2}(U - Z)^T K(A)(U - Z) + R(a)$$

Let's differentiate
$J_2(A) = \frac{1}{2}(U-Z)^T K(A)(U-Z)$

$$\begin{aligned}
DJ_2(A)\delta A &= \frac{1}{2}[(\delta U)^T K(A)(U-Z) + (U-Z)^T K(a)\delta U \\
&\quad +(U-Z)^T DK(A)\delta A(U-Z)] \\
&= (\delta U)^T K(A)(U-Z) + \frac{1}{2}(U-Z)^T K(\delta A)(U-Z) \\
&= [-K(A)^{-1}K(\delta A)U]^T K(A)(U-Z) \\
&+ \frac{1}{2}(U-Z)^T K(\delta A)(U-Z) \\
&= -(\delta A)^T L(U)^T(-K(A)^{-1})^T K(A)(U-Z) \\
&+ \frac{1}{2}(U-Z)^T K(\delta A)(U-Z) \\
&= -(\delta A)^T L(U)^T(U-Z) + \frac{1}{2}(\delta A)^T L(U-Z)^T(U-Z) \\
\nabla J_2(A) &= -\frac{1}{2}(\delta A)^T L(U+Z)^T(U-Z)
\end{aligned}$$

Next it can be seen that,

$$\begin{aligned}
\nabla J_2(A) &= -\frac{1}{2}L(U+Z)^T(U-Z) \\
\nabla J_2(A) &= -\frac{1}{2}L(U)^T U + \frac{1}{2}L(Z)^T Z
\end{aligned}$$

Which will lead us to the hessian:

$$\begin{aligned}
DJ_2(A)\delta A &= -\frac{1}{2}\delta A \cdot L(U)^T U + \frac{1}{2}\delta A \cdot L(Z)^T Z \\
D^2 J_2(A)(\delta A, \delta A) &= -\frac{1}{2}\delta A \cdot L(\delta U)^T U - \frac{1}{2}\delta A \cdot L(U)^T \delta U \\
&= -\delta A \cdot L(U)^T \delta U \\
&= \delta A \cdot L(U)^T K(A)^{-1} L(V)\delta A \\
\nabla^2 J_2(A) &= L(U)^T K(A)^{-1} L(U)
\end{aligned}$$

In summation, we have:
Objective functional:

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

Gradient:

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

Hessian:

$$\nabla^2 J_2(A) = L(U)^T K(A)^{-1} L(U)$$

1.7.3 Equation Error

Details on the following computations can also be found in [23]
Objective functional:

$$J_3(A) = \frac{1}{2}\langle L(Z)A - F, P^{-1}(L(Z)A - F\rangle$$

Gradient:

$$\nabla J_3(A) = L(Z)^T P^{-1}(L(Z)A - F)$$

Hessian:

$$\nabla^2 J_3(A) = L(Z)^T P^{-1} L(Z)$$

$$\text{where} \qquad P = M + K$$

Chapter 2

Motivation and Literature Review

Many works have been dedicated to the field of unconstrained optimization and continuous methods. The following references do not encompass all the related works but do give a valid survey of previous work done. Ideas and motivation for this thesis came from several different papers presenting possible approaches to solve ordinary differential equations.

Botsaris [14] presented a family of newly formulated differential descent techniques for function minimization. Several algorithms were produced that, in a finite number of steps, minimize a quadratic function and efficiently minimize functions. In place of a ray, a broader curvilinear search path was utilized and the eigensystem of the Hessian matrix was connected with the minimization problem. To reach the curvilinear search paths, initial value systems of differential equations were solved. This brought about the idea to further improve current numerical integration methods for utilization in minimization of functions. The algorithms were applied to several test functions and results were noted along with possible research improvement areas.

Brown and Bartholomew-Biggs [15] studied several methods designed to solve equality constrained minimization problems. These methods followed a trajectory based on a system of differential equations. The numerical performance of several of these methods were compared with SQPs or sequential quadratic programming algorithms. In the paper, these methods were ap-

plied to eighteen test problems, which showed that the ODE techniques were in fact more successful than the SQP schemes. The authors proposed that these results proved the necessity for new research of these ODE methods in order to evaluate and further improve them, along with fortifying the already available SQP algorithms.

Another relevant work by Brown and Bartholomew-Biggs [16] surveyed methods for minimizing a function by following the solution curve of a system of ordinary differential equations. These methods were previously believed to be too expensive, but this paper showed through numerical tests and algorithms that this ODE technique can be applied in a way that allows it to compete with other popular methods.

In Schaffler [36], only the gradient of the objective function is used in this trajectory-following method. The results were then compared to Brown [16], and due to the simple step-size control, it measured up efficiently. Liao [32] also presented a gradient based continuous method for large scale optimization problems, where stiff and nonstiff methods were compared.

Flåm [20] studied nonsmooth convex programs with sharp constraints in a Hilbert space. Bounded and strictly feasible problems were considered. These programs were solved by a continuous trajectory brought about by a differential inclusion of a subgradient type.The steepest descent direction was implemented when possible. Fl åm showed that the considered algorithm converged to an ideal solution in finite time.

One of the first contributions in this interesting direction of research is by A. Antipin [6] the author focused on a convex programming problem of the following form: find $x^* = \text{Arg min}\{f(x)|x \in Q\}$ where f is a differentiable convex function and Q is a convex subset of R^n.. The problem is approached by continuous gradient projection methods of the first and second order. The first-order method connects the considered problem to the system of differential equations $\frac{dx}{dt} + x = \pi_Q(x - \alpha\nabla f(x)), x(t_0) = x^0$, where π_Q is the projection operator on Q and α is a positive step-length. It was shown that the trajectory $x(t)$ converges to x^* and, moreover, there exists $s = s(\alpha) > 0$ such that $x(t)$ satisfies $\|x(t) - x^*\|^2 Ce^{-st}$, $t \geq t_0$. Antipin also proposed a second-order technique and showed that the solution is exponentially stable.

H. Attouch studied several avenues of the dynamical system approach. A distinguished paper by Attouch and Cominetti [9] presented the asymptotic

behavior of the solutions to evolution equations of the form $0 \in u'(t) + \partial f(u(t), \epsilon(t)),\ u(0) =_0$ where $f(\cdot, \epsilon) : \epsilon > 0$ is a family of strictly convex functions whose minimum is attained at a unique point $x(\epsilon)$. Given the assumption that $x(\epsilon)$ converges to a point x_0 as $\epsilon \to 0$, and relying on the performance of the optimal trajectory $x(\epsilon)$, they derived sufficient conditions on the parametrization $\epsilon(t)$ which guaranteed that the solution $u(t)$ of the evolution equation converges to x_0 when $t \to \infty$. The outcomes were presented on three different penalty and viscosity-approximation schemes for convex minimization.

Glazos, Hui, and Żak [21] presented a class of dynamical systems that solved a convex optimization problem. They proved that all trajectories of these systems converged to optimum solutions. The Lyapunov stability theory for dynamical systems with discontinuous right-hand sides was used in the analysis.

Haraux and Jendoubi [28] studied the second-order gradient-like system $U_{tt} + g(U_t) = \nabla(U)$, where $F : R^n \to R$ is analytic and $g : R^n \to R^n$ is Lipschitz and coercive while $g(0) = 0$. The authors showed the system's convergence of global and bounded solutions to equilibrium points.

Alvarez and Pérez [5] considered the existence and asymptotic convergence of the trajectories from

$$\nabla^2 f(u(t), \epsilon(t))u'(t) + \epsilon'(t)\frac{\partial^2 f}{\partial \epsilon \partial x}(u(t), \epsilon(t)) + \nabla f(u(t), \epsilon(t)) = 0,$$

when $t \to \infty$ and where $\{f(\cdot, \epsilon)\}_{\epsilon > 0}$ is a parametric family of convex functions that approximate a given convex function f they try to minimize, and $\epsilon(t)$ is a parametrization such that $\epsilon(t) \to 0$ when $t \to \infty$. The authors got this technique from variational characterization of Newton's method shown here:: $(P_t^\epsilon)\ u(t) \in \text{Argmin}\{f(x, \epsilon(t)) - e^{-t}\langle \nabla f(u_0, \epsilon_0), x\rangle : x \in H\}$, where H is a real Hilbert space. They found conditions on the approximating group $f(\cdot, \epsilon)$ and the parametrization $\epsilon(t)$ to guarantee the norm convergence of the solution trajectories $u(t)$ in the direction of a certain minimizer of f. Rates of convergence were studied using the asymptotic estimates. Barrier and penalty techniques for linear programming were used to demonstrate application results. Along with this, viscosity methods for an abstract noncoercive variational problem were displayed. Lastly, the steepest descent method was used for comparison as well.

In [3], F. Alvarez studied the asymptotic behavior at infinity of solutions of the second-order evolution equations with damping and convex potentials. The results were given in a Hilbert space setting. Also refer to Aassila [1].

Attouch, Goudou, and Redont [10] worked on the problem shown here: Let H be a real Hilbert space and $\Phi : H \to R$ a continuously differentiable function whose gradient is Lipschitz continuous on bounded sets. The authors studied the nonlinear dissipative dynamical system $x'(t)+\lambda x'(t)+\nabla\Phi(x(t)) = 0$, $\lambda > 0$, along with Cauchy data, focusing on the unconstrained minimization of the function Φ. Additional results regarding the convergence of a solution to a critical point were given in several circumstances, coupled with convex Φ or is a Morse function; a counterexample demonstrated that, without particular assumptions, a trajectory might not converge. A scheme for studying local minima of Φ was found by following the trajectories. A singular perturbation analysis connected the results pertaining to gradient systems. See also Attouch and Alvarez [7] and Attouch, Bolte, and Redont [8].

Schropp [37] considered an ODE approach to solve smooth minimization problems with equality constraints. Slack variables were used to change the inequality constraints into equality constraints. The ODE was reformulated as differential algebraic equations to reach efficient implementation. Then the approach was linked to the SQP approach. Numerical examples were given for both methods and compared.

Refer to Coffey, Kelley, and Keyes [17]. Diener and Schaback [19] presented a numerical realization of an extended continuous Newton scheme defined by Diener [18]. The technique followed a connected set of locally one-dimensional trajectories that had all critical points of a smooth function $f : R^n \to R$. This method was found to be useful and efficient.

Shi [38] presented an innovative multi-step curve search method for solving unconstrained minimization problems. The scheme used information from previous iterative steps and a curve search rule to generate new iterative points. The curve search was a generalization of the line search. The method was proved to be globally convergent and the rate of convergence was linear under certain assumptions. The technique guaranteed stability of convergence and it can be applied to solving large-scale problems. The large-scale problem numerical experiment results were showed and compared with the conjugate-gradient methods from Fletcher and Reeves, Polak and

Ribiere, and Hestenes and Stiefel.

Attouch and Teboulle [13] presented a new class of dynamical systems associated with qualitative and numerical characteristics of optimization. It combined the rudimentary algorithm of the continuous gradient method for minimization with a Lotka-Volterra nonlinear differential system contained by a logarithmic-quadratic proximal method. They concentrated on proving global existence, viability results, asymptotic performance of the trajectory, and making the global convergence of the trajectory to a minimizer of the accompanying convex optimization problem over the nonnegative orthant.

Alvarez and Cabot [4] considered in the Hilbert space, the problem of minimizing a twice continuously differentiable real valued function that's bounded from below. They reviewed the gradient differential equation with variable scaling. The scaling factor matched up with a quadratic approximation to a linear search in the negative gradient direction. This caused the gradient to decrease dramatically alongside the integral curves. They looked at the situation if a convex function that is either nonsmooth or not *strongly* convex, then a group of smooth strongly convex approximations of the original function was used. Consequently, the approximation scheme was matched up with the scaled gradient DE Subsequently, the authors couple the latter approximation scheme with the above-mentioned scaled gradient differential equation. Barrier methods in linear programming and viscosity techniques were applied to the problem and evaluated.

Liao, Qi, and Tam [32] explored the gradient-based continuous scheme for solving the minimization problem and its numerical outcomes. The following: $\min f(x)$, where $f : R^n \to R$ is a continuously differentiable function, is the minimization problem at hand.The scheme came about from the technique that is made up of finding the zero-gradient points of f while equilibrium points of the ODE $\frac{dx(t)}{dt} = \nabla(x(t))$ were sought. The authors provided proof of a convergence result by Q. Han et al. [27]. Then they report and discuss numerical results gathered from solving 27 test problems with n very large, up to 106. The ODE problem is solved by the LSODAR solver of ODEPACK from [29].

Liao [31] described an effective continuous method for convex programming problems that can be applied to practical problems. By converting the convex problem into a variational inequality, it was then solved using the

continuous method which does not require a Lipschitz condition, but still maintains strong convergence of the ODE solution.

Ou [34] considered a trust region scheme for solving a system of nonsmooth equations of type $F(x) = 0$ where F is a locally Lipschitz continuous function. F can be decomposed into the sum of a smooth and nonsmooth part. An algorithm consisting of two unique traits in comparison with previous ideas was proposed. The algorithm used the derivative of the smooth part and the function values of F in the formation of an approximation of the Hessian. This scheme mandated solving a linear system of equations instead of working at every iteration a quadratic programming problem. The authors proved global convergence and local superlinear convergence of the adopted scheme, given that F is semismooth and the Hessian approximation is bounded.

Ou, Zhou, and Lin [35] considered solving unconstrained optimization problems using a trust region algorithm. It is a mixture of a trust region method, fixed step-length and ODE techniques. A system of linear equations is solved at each iteration to gain a trial step, if the trial step is not accepted, the technique is to generate an iterative point whose step-length is determined by a given formula. Given certain assumptions, the algorithm is globally convergent and locally superlinear convergent as well.

Shikhman and Stein [39] presented a dynamical system approach to solve general constrained optimization problems. The first of two parts of the approach was to replace the problem by an ODE. Secondly, the equation was discretized by an appropriate method. Inequality constraints were converted to equality constraints by the use of quadratic slack variables. Unfortunately, this approach led to an exponential number of critical points in the number of inactive constraints. This was fixed by using a differential equation from the dynamical system given by Jongen and Stein (2003). The authors computed the projected gradient to obtain the ODE in the original variables. Finally, basic properties of the dynamical system were discussed, along with the relationship between the new ODE method and the beginning optimization problem.

A real valued, continuous convex function f on a Banach space X was studied by Aizicovici, Reich, and Zaslavski in [2]. f was equipped with a complete metric space of vector fields $V : X \to X$ such that the directional

derivative $f_0(x, Vx) \leq 0 \quad \forall x \in X$. Gradient-like iterative procedures correlate to each vector field in such cases. Related research proved that for an everywhere-dense subset of the space of vector fields, convergence can be established in the sense that the iterative processes generate sequences x_n such that $f(x_n) \to \inf(f)$ on everywhere-dense $G\delta$ subsets of X. The authors' paper reinforced these results for the case in which f is Lipschitz continuous on bounded subsets of X and f has a sharp minimum over X. Convergence was achieved over an open everywhere-dense set.

Hajba [26] presented continuous versions of the Fletcher-Reeves iteration for minimization described by a system of second-order differential equations. This problem was studied in previous papers [24] and [25] which used the assumption that the minimizing function is strongly convex. In this paper, strong convexity was replaced with only convexity of the minimizing function. The Tikhonov regularization [40], [41] was used to reach the minimal norm solution as the asymptotically stable limit point of the trajectories.

Attouch and Svaiter [12] considered nonautonomous continuous dynamical systems that were connected to the Newton and LevenbergMarquardt schemes. The authors focused on solving inclusions controlled by maximal monotone operators in Hilbert spaces. Dependent upon the Minty representation of maximal monotone operators as Lipschitzian manifolds, they demonstrated that in time differential systems, these dynamics can be expressed as first-order. This was applicable to the CauchyLipschitz theorem. The trajectories were proved to converge weakly to equilibria by way of Lyapunov techniques. New understanding of Newton's method for solving monotone inclusions was given by way of algorithms with time discretization of these dynamics. Refer to Attouch, Peypouquet, and Redont [11] as well.

Ou [33] presented an ODE method to solve unconstrained optimization problems. This method combined the IMPBOT algorithm with a fixed step-length. Details on the IMPBOT algorithm can be found in [16]. This technique solved a lower dimensional system of linear equations just once at each iteration to get a trial step. When the trial step was not accepted, this method used minimization of a convex overestimation, which avoided a line search for step-length. It was shown to be most effective on small scale optimization problems. With reasonable assumptions held, this scheme proved to be globally convergent.

In this work, the paper Zhang, Kelley, and Liao [42] was implemented. As will be discussed in more complete detail later in this work, a new continuous optimization method was introduced. In summary, it combined Newton's and the steepest descent direction for a new dynamical system to solve the unconstrained optimization problem. This method defined the trajectory in such a way that line search is not needed, singularities of the Hessian are avoided and global convergence is shown through numerous test problems.

Chapter 3

Continuous Method Using OLS and MOLS

In this chapter, we investigate the inverse problem which is posed as a dynamical system. We use the output least-squares as well as modified output least squares formulation to convert the inverse problem to an optimization problem. As an optimality condition we obtain a variational inequality defined over the set of feasible coefficients. By using the well-known properties of the projection operator, we convert the variational inequality into a nonlinear equation which then motivates the study of a dynamical system. We give convergence analysis of the proposed dynamical system.

3.1 An Abstract Formulation

In this section we present an abstract formulation for the inverse problem of identifying parameters in general variational problems. In the following, X and Y are two real Hilbert spaces, K is a nonempty closed and convex subset of X, $T : X \times Y \times Y \to \mathbb{R}$ a trilinear form and $m : Y \to \mathbb{R}$ a bounded linear functional. Here the space X hosts the parameters and the set K is the set of feasible parameters. We suppose that there exists positive constants α, β such that for all u, $v \in Y$ and for all $a \in K$, the following estimates hold:

$$T(a,u,v) \leq \beta\|a\|_X\|u\|_Y\|v\|_Y \tag{3.1}$$

$$T(a,u,u) \geq \alpha\|u\|_Y^2. \tag{3.2}$$

By the Riesz representation theorem, the variational equation

$$T(a, u, v) = m(v) \quad \text{for all } v \in Y \tag{3.3}$$

has a unique solution u for each $a \in A$. Consequently, we define $F : A \to Y$ by the condition that $u = F(a)$ is the solution to (3.3).

Notice that for a fixed $(a, u) \in X \times Y$, by considering the map $v \to T(a, u, v)$ and keeping in mind the linearity and continuity of the trilinear form, we get the existence of a mapping $\mathcal{T} : X \times Y :\to \mathbb{R}$ such that

$$\langle \mathcal{T}(a, u), v\rangle_Y = T(a, u, v) \quad \text{for all } v \in Y.$$

This equation allows us to interpret the results of Kluge [30], which are in the context of the parametric equation $\mathcal{T}(a, u) = m$, in the present setting.

We begin with recalling some useful properties of the solution map F.

Lemma 3.1.1. *For $a, b \in K$, we have*

$$F(b) - F(a) = F'(a)(b - a) + Q(a, b - a)$$

with

$$\begin{aligned}
\|F(a)\| &\leq \frac{1}{\alpha}\|m\| \\
\|F'(a)\| &\leq \frac{\beta}{\alpha^2}\|m\| \\
\|F(a) - F(b)\| &\leq \frac{\beta}{\alpha^2}\|m\|\,\|a - b\| \\
\|F'(a) - F'(b)\| &\leq \frac{2\beta^2}{\alpha^3}\|m\|\,\|a - b\| \\
\|Q(a, b - a)\| &\leq \frac{\beta^2}{\alpha^3}\|m\|\,\|a - b\|^2.
\end{aligned}$$

Proof. See Kluge [30]. □

3.2 Formulation of the inverse problem

Suppose that a (possibly noisy) measurement z of $\bar{u}$ is available, where $\bar{u}$ and $\bar{a}$ jointly satisfy variational problem (3.3). The objective of this chapter is to

propose and analyze a method for estimating $\bar{a}$ from the given data z. We define the functional $J_0 : K \to \mathbb{R}$ by

$$J_0(a) := H(F(a)) \tag{3.4}$$

with

$$H(u) = \frac{1}{2}\|u - z\|_Y^2 \tag{3.5}$$

where $u = F(a)$ is the unique solution of (3.3) corresponding to a. The functional J_0 is related to output least-squares functionals considered by numerous authors.

We will estimate $\bar{a}$ by minimizing J_0 over the set of admissible parameters K, that is, by solving the following optimization problem: Find $a \in K$ such that

$$J_0(a) \leq J_0(b) \quad \text{for all } b \in K. \tag{3.6}$$

However, since the inverse problem under consideration is severely ill-posed, it is necessary to regularize the OLS functional J_0. In the setting, the following observation also justifies the need of regularization.

The following variational inequality is a necessary optimality condition for $\bar{a} \in K$ to be a solution of the optimization problem (3.6)

$$D_+J_0(\bar{a})(b - \bar{a}) \geq 0 \quad \text{for all } b \in K \tag{3.7}$$

where

$$D_+J_0(a)(b - a) = \lim_{t \downarrow 0} \frac{J_0(a + t(b - a)) - J_0(a)}{t} \quad a,\, b \in K$$

provided that the derivative exists (see [30]).

In this chapter, we intend to employ some dynamical system methods based on the variational inequality formulation. However, in the context of variational inequalities, the dynamical system approach demands for some kind of strengthen monotonicity of $D_+J_0(\cdot)$ for convergence results. To verify the availability of this stronger requirement, we begin by exploiting the form of H. Evidently, for $J_0(\cdot)$ given in (3.4), we have

$$D_+J_0(a)(b - a) = \langle F'(a)^*(F(a) - z), b - a\rangle \tag{3.8}$$

where we have used the fact that $H'(u) = u - z$.

We begin by noticing that H' is monotone, that is

$$\langle H'(u) - H'(v), u - v\rangle \geq 0, \quad \text{for all } u,\, v \in Y.$$

By setting $u = F(a_k)$ and $v = F(\bar{a})$ in the above inequality, we obtain

$$\langle H'(F(a_k)) - H'(F(\bar{a})), F(a_k) - F(\bar{a})\rangle \geq 0,$$

which by using Lemma 3.1.1 yields

$$0 \leq \langle H'(F(a_k)), F(a_k) - F(\bar{a})\rangle - \langle H'(F(\bar{a})), F'(\bar{a})(a_k - \bar{a}) + Q(\bar{a}, a_k - \bar{a})\rangle$$

and consequently

$$\begin{aligned}
\langle H'(F(\bar{a})), F'(\bar{a})(a_k - \bar{a})\rangle &\leq \langle H'(F(a_k)), F(a_k) - F(\bar{a})\rangle \\
&\quad -\langle H'(F(\bar{a})), Q(\bar{a}, a_k - \bar{a})\rangle \\
&= -\langle H'(F(a_k)), F'(a_k)(\bar{a} - a_k)\rangle \\
&\quad -\langle H'(F(a_k)), Q(a_k, \bar{a} - a_k)\rangle \\
&\quad -\langle H'(F(\bar{a})), Q(\bar{a}, a_k - \bar{a})\rangle.
\end{aligned}$$

Summarizing the above calculation, we have

$$\langle F'(a_k)^* H'(F(a_k)) - F'(\bar{a})^* H'(F(\bar{a})), a_k - \bar{a}\rangle \geq$$

$$\langle H'(F(a_k)), Q(a_k, \bar{a} - a_k)\rangle + \langle H'(F(\bar{a})), Q(\bar{a}, a_k - \bar{a})\rangle.$$

Now by using Lemma 3.1.1 once again, we have

$$\begin{aligned}
\langle H'(F(a_k)), Q(a_k, \bar{a} - a_k)\rangle &\leq \|H'(F(a_k))\|\, \|Q(a_k, \bar{a} - a_k)\| \\
&= \|F(a_k) - z\|\, \|Q(a_k, \bar{a} - a_k)\| \\
&\leq \frac{\beta^2}{\alpha^4}\|m\|(\|m\| + \alpha\|z\|)\|a_k - \bar{a}\|^2
\end{aligned}$$

Analogously, we have

$$\langle H'(F(\bar{a})), Q(\bar{a}, a_k - \bar{a})\rangle \leq \frac{\beta^2}{\alpha^4}\|m\|(\|m\| + \alpha\|z\|)\|a_k - \bar{a}\|^2.$$

By combining the above three inequalities, we obtain

$$\langle F'(a_k)^* H'(F(a_k)) - F'(\bar{a})^* H'(F(\bar{a})), a_k - \bar{a}\rangle \geq -\kappa\, \|a_k - \bar{a}\|^2 \tag{3.9}$$

where

$$\kappa = \frac{2\beta^2}{\alpha^4}\|m\|(\|m\| + \alpha\|z\|) \tag{3.10}$$

Consequently, the must desired strong monotonicity argument cannot be confirmed. Kluge [30] suggested to incorporate a strongly convex and differentiable regularization operator R so that $\tau J' + \rho R'$, with $R'(\cdot)$ as the derivative of R, becomes strongly monotone. Here τ and ρ are strictly positive constants.

To be specific, we define $J : A \to \mathrm{R}$ by

$$J(a) = \tau J_0(a) + \rho R(a)$$

where R is a strongly convex Gateaux differentiable functional, that is, R' is strongly monotone with modulus of monotonicity as κ_0.

We note that the condition

$$\rho\kappa_0 - \tau\kappa > 0 \tag{3.11}$$

now ensures that $\tau F'(\cdot)^* H'(\cdot) + \rho R'$ is strongly monotone, that is, for all a_1, $a_2 \in K$, we have

$$\langle \tau F'(a_1)^* H'(a_1) + \rho R'(a_1) - \tau F'(a_2)^* H'(a_2) - \rho R'(a_2), a_1 - a_2 \rangle \geq$$

$$(\rho\kappa_0 - \tau\kappa)\|a_1 - a_2\|^2. \tag{3.12}$$

Although all the arguments here are valid for an arbitrary strongly convex functional, we shall stick to the choice

$$R(a) = \frac{1}{2}\|a - \tilde{a}\|^2,$$

for some $\tilde{a} \in X$. That is, $\kappa_0 = 1$ in (3.11).

Therefore, instead of (3.6) we consider the following regularized optimization problem: Find $a \in A$ such that

$$J(a) \leq J(b) \quad \text{for all } b \in K. \tag{3.13}$$

By standard arguments it follows that the above problem leads to the following variational inequality: Find $a \in A$ such that

$$\langle \tau\, F'(a)^*(F(a) - z) + \rho R'(a), b - a \rangle \geq 0 \ \text{ for all } b \in K. \tag{3.14}$$

In fact by using the employed notion of the one-sided directional derivative, it can be shown that (see [30, Lemma 4.1]) under condition (3.11), the functional $J(a)$ is strongly convex and hence the above variational inequality is necessary as well as sufficient optimality condition for the regularized minimization problem (3.13).

We conclude by recalling the notion of the projection.

Theorem 3.2.1. *Let K be a closed and convex subset of a Hilbert space H. Then for every $x \in H$ there is a unique $z \in K$ such that*

$$\|x - z\| = \inf_{w \in K} \|x - w\|.$$

Proof. See Kinderlehrer and Stampacchia (1980). □

Remark 3.2.1. *The element z in the above result is the projection of x onto K and is denoted by $z = P_K(x)$.*

The following result gives some useful properties of the projection map.

Theorem 3.2.2. *Let K be a closed and convex subset of a Hilbert space H. Then $z = P_K(x)$ if and only if*

$$\langle x - z, w - z \rangle \leq 0, \quad \text{for every } w \in K. \tag{3.15}$$

Furthermore, the projection map $P_K(\cdot)$ is nonexpansive, that is,

$$\|P_K(x_1) - P_K(x_2)\| \leq \|x_1 = x_2\|, \quad \text{for every } x_1, x_2 \in H. \tag{3.16}$$

3.3 The Continuous Method

We now write the variational inequality (3.14) as a nonlinear operator equation which would then give rise to a dynamical system.

Let $P_K : X \to K$ be the projection map onto the closed and convex set of admissible parameters K. It follows from (3.15) that $a \in K$ is a solution of variational inequality if and only if

$$a = P_K(a - \alpha(\tau\, F'(a)^*(F(a) - z) + \rho R'(a)))), \quad \alpha > 0.$$

The above equation motivates to consider the following dynamical system:

$$\begin{aligned} \dot{a}(t) + a(t) &= P_K(a(t) - \alpha(t)[\tau\, F'(a(t))^*(F(a(t)) - z) + \rho R'(a(t))]), & (3.17a) \\ a(t_0) &= a_0 \in H, & (3.17b) \end{aligned}$$

where $\alpha(t)$ is a positive continuous function for $t \geq t_0 \geq 0$ and

$$0 < \alpha_0 \leq \alpha(t) \leq \alpha_1. \tag{3.18}$$

The following result is based on the ideas given in [6, 22, 30]

Theorem 3.3.1. *Assume that the conditions* (3.21) *and* (3.23) *hold. Then* $u(t)$ *converges strongly to the unique solution of variational inequality as* $t \to \infty$. *Moreover,*

$$\|u(t) - x\| \leq \|u_0 - x\| exp[-\ell(t - t_0)].$$

Proof. In the following for simplicity, we set

$$A(a(t)) := [\tau\, F'(a(t))^*(F(a(t)) - z) + \rho R'(a(t))].$$

We define

$$F(t, a) = P_K(a(t) - \alpha(t)A(a(t))) - a(t).$$

By virtue of the property (3.16), we have

$$\begin{aligned} \|F(t, a_1) - F(t, a_2)\| &\leq \| - a_1(t) + P_K(a_1(t) - \alpha(t)A(a_1(t))) \\ &\quad -(-a_2(t) + P_K(a_2(t) - \alpha(t)Aa_2(t)))\| \\ &= \|a_1(t) - a_2(t)\| \\ &+ \|P_K(a_1(t) - \alpha(t)A(a_1(t))) - P_K(a_2(t) \\ &- \alpha(t)A(a_2(t)))\| \\ &\leq \|a_1(t) - a_2(t)\| + \|a_1(t) - a_2(t) \\ &+ \alpha(t)[A(a_1(t))) - A(a_2(t))]\| \\ &\leq [2 + \alpha_1 L]\|a_1 - a_2\|, \end{aligned}$$

where $L > 0$ is constant such that

$$\|A(a_1(t))) - A(a_2(t))\| \leq L\|a_1(t) - a_2(t)\|.$$

The precise value of the constant will be estimated shortly. Therefore, the dynamical system has a Lipschitz continuous and consequently standard existence results ensure that

$$\dot{a}(t) = F(t, a)$$

is uniquely solvable. Furthermore, the solution is of class $C^1[t_0, \infty[$.

In view of the equivalence of the following two statements

$$\begin{aligned} z &= P_K x \\ \langle z - x, z - y\rangle &\leq 0, \quad \forall\ y \in K, \end{aligned}$$

the identity

$$\dot{a}(t) + a(t) = P_K(a(t) - \alpha(t)Aa(t)),$$

ensures that for all $b \in K$, we have

$$\langle (\dot{a}(t) + a(t)) - (a(t) - \alpha(t)Aa(t)), b - (\dot{a}(t) - a(t))\rangle \geq 0,$$

or

$$\langle \dot{a}(t) + \alpha(t)Aa(t), b - \dot{a}(t) - a(t)\rangle \geq 0, \quad \forall\ b \in K. \tag{3.19}$$

In view of $\dot{a}(t) + a(t) = P_K(a(t) - \alpha(t)Aa(t))$, we have

$$\dot{a}(t) + a(t) \in K, \quad t \geq t_0.$$

Let $\bar{a}$ be the unique solution of the variational inequality. Then we obtain the following two variational inequalities

$$\begin{aligned} \langle A(\bar{a}), \dot{a}(t) + a(t) - x\rangle &\geq 0 \\ \langle \dot{a}(t) + \alpha(t)Aa(t), \bar{a} - \dot{a}(t) - a(t)\rangle &\geq 0. \end{aligned}$$

We multiply the first of above inequalities by $\alpha(t)$ on both sides and add to the second inequality to obtain

$$\langle \dot{a}(t) + \alpha(t)[Aa(t) - A(\bar{a})], \bar{a} - a(t) - \dot{a}(t)\rangle \geq 0,$$

which can be written as

$$\langle \dot{a}(t), \bar{a} - a(t)\rangle + \alpha(t)\langle A(a) - A\bar{a}, \dot{a}(t)\rangle \geq \|\dot{a}(t)\|^2 + \alpha(t)\langle Aa(t) - A(\bar{a}), a(t) - \bar{a}\rangle. \tag{3.20}$$

We now define the function

$$\sigma(t) = \frac{1}{2}\|a(t) - \bar{a}\|^2,$$

and notice that

$$\begin{aligned} \sigma'(t) &= \langle \dot{a}(t), a(t) - \bar{a} \rangle \\ \sigma(t_0) &= \frac{1}{2}\|a_0 - \bar{a}\|^2 = \sigma_0. \end{aligned}$$

In view of the strong monotonicity of $A := \tau F'(\cdot)^* H'(\cdot) + \rho R'$ (see (3.12)), we have

$$\langle Aa(t) - A\bar{a}, a(t) - \bar{a} \rangle \geq 2M\sigma(t)$$

with

$$M := (\rho - \tau\kappa).$$

Note that

$$\|F'(a(t))^* H'(F(a_k)) - F'(\bar{a})^* H'(F(\bar{a}))\|^2$$

$$\begin{aligned} &\leq 2\left\{\|(F'(a(t))^* - F'(\bar{a})^*)(F(\bar{a}) - z)\|^2\right\} \\ &+ 2\left\{\|(F'(\bar{a})^*(F(a(t)) - F(\bar{a})\|^2\right\} \\ &\leq 2\left\{\frac{4\beta^4}{\alpha^6}\|m\|^2\|a(t) - \bar{a}\|^2\|F(a(t)) - z\| + \frac{\beta^4}{\alpha^8}\|m\|^4\|a(t) - \bar{a}\|^2\right\} \\ &\leq 2\left\{\frac{4\beta^4}{\alpha^8}\|m\|^2(\|m\| + \alpha\|z\|)^2\|a(t) - \bar{a}\|^2 + \frac{\beta^4}{\alpha^8}\|m\|^4\|a(t) - \bar{a}\|^2\right\} \\ &\leq \left\{\frac{8\beta^4}{\alpha^8}\|m\|^2(\|m\| + \alpha\|z\|)^2 + \frac{2\beta^4}{\alpha^8}\|m\|^4\right\}\|a(t) - \bar{a}\|^2 \\ &= L_0\|a(t) - \bar{a}\|^2, \end{aligned}$$

where

$$L_0 := \left\{\frac{8\beta^4}{\alpha^8}\|m\|^2(\|m\| + \alpha\|z\|)^2 + \frac{2\beta^4}{\alpha^8}\|m\|^4\right\}.$$

Furthermore,

$$\begin{aligned} \langle A\bar{a} - Aa(t), \dot{a}(t) \rangle &\leq \|A\bar{a} - Aa(t)\|\|\dot{a}(t)\| \\ &\leq L_1\|\bar{a} - a(t)\|\|\dot{u}(t)\| \\ &\leq \frac{L_1}{2}\left(2\sigma(t) + \|\dot{u}(t)\|^2\right), \end{aligned}$$

where

$$L_1 := \rho + \tau\sqrt{L_0}$$

and we used the identity

$$ab \leq \frac{a^2}{2} + \frac{b^2}{2}.$$

Consequently,

$$\|\dot{u}(t)\|^2 \left[1 - \frac{1}{2}(L_1\alpha(t))\right] + \dot{\sigma}(t) + \sigma(t)(2M\alpha(t) - L_1\alpha(t)) \leq 0.$$

Assuming that

$$L_1 \leq \frac{2}{\alpha_1}, \tag{3.21}$$

we can drop the first term in the above inequality and obtain

$$\dot{\sigma}(t) + \sigma(t)(2M\alpha(t) - L_1\alpha(t)) \leq 0. \tag{3.22}$$

Moreover, assuming that

$$\ell := (2M - L_1) > 0, \tag{3.23}$$

we have

$$\dot{\sigma}(t) \leq -\ell\sigma(t),$$

and consequently

$$\sigma(t) \leq \|a_0 - \bar{a}\|^2 e^{-\ell(t-t_0)},$$

or

$$\|a(t) - \bar{a}\|^2 \leq \sigma_0 e^{-\ell(t-t_0)}.$$

This ensures that $a(t)$ converges strongly to $\bar{a}$ as $t \to \infty$. The final assertion is a direct consequence of the above inequality. The proof is complete. □

Chapter 4

Differential Equation Solver Method

4.1 Continuous Newton-type Method

Recall from Chapter 1, that we were aiming to minimize the functional J. Thus, as a minimization problem, a continuous Newton-type method for unconstrained optimization can be utilized. The ideas from this method come from the Zhang and Kelley paper [42]. In general, we are looking for the solution to the following minimization problem

$$\min_{a \in R^n} J(a)$$

The most basic way to solve this problem is the continuous method using the simplest trajectory: the continuous gradient method, also known as the steepest descent direction.

$$\begin{cases} \frac{da}{dt} &= -\nabla J(a(t)) \\ a(t_0) &= a_0 \end{cases}$$

Another trajectory to implement is Newton's direction, which uses the hessian in its computation,

$$\begin{cases} \frac{da}{dt} & = & -(\nabla^2 J(a(t)))^{-1}\nabla J(a(t)) \\ a(t_0) & = & a_0 \end{cases}$$

Because the hessian is used, the issue of singularity arises. The Kelley paper adopts a method that can implement either the Newton direction, the negative gradient, or a modified mixture of the two methods. Depending on the minimum eigenvalue, one of the three trajectories were used,

$$\begin{cases} \frac{da}{dt} & = & g(a) \\ a(t_0) & = & a_0 \end{cases}$$

Here $g(a)$ is defined as the following

$$g(a) =$$

$$\begin{cases} -(\nabla^2 J(a(t)))^{-1}\nabla J(a(t)) & \text{if} \quad \lambda_{min}(a) > \delta_2 \\ -\alpha(t)(\nabla^2 J(a(t)))^{-1}\nabla J(a(t)) - \beta(t)\nabla J(a(t)) & \delta_1 \leq \lambda_{min}(a) \leq \delta_2 \\ -\nabla J(a(t)) & \lambda_{min}(a) < \delta_1 \end{cases}$$

The minimum eigenvalue of the hessian of $J(a)$ is represented by $\lambda_{min}(a)$, and $\delta_2 > \delta_1 > 0$. Also,

$$\begin{cases} \alpha(a) & = & \frac{\lambda_{min}(a)-\delta_1}{\delta_2-\delta_1} \\ \beta(a) & = & 1-\alpha(a) = \frac{\delta_2-\lambda_{min}(a)}{\delta_2-\delta_1} \end{cases}$$

The idea shown above for the mixed method comes from a convex combination of the emphasis placed on the Newton direction versus the continuous gradient direction. In other words, when $\lambda_{min}(a)$ is between δ_1 and δ_2, if its value is closer to δ_2 then $\alpha(a)$ will be larger, in turn putting more emphasis on the Newton direction. On the other hand, if $\lambda_{min}(a)$ is closer to δ_1 then $\beta(a)$ will be the larger term, putting more emphasis on the continuous gradient direction.

Applied to four second order test examples, the following methods were used.

1. Euler
2. Trapezoid
3. Runge-Kutta
4. Matlab ODE 45
5. Adams-Bashforth
6. Matlab ODE 113

Euler method is a first-order ODE solver. This is the simplest method in which the tangent line is used. The coefficient $a(t)$ is found in a one step approximation:

$$a_{i+1} = a_i + hg(a_i)$$

Clearly, the process finds the slope (tangent line) at the initial condition $a(t_0) = a_0$, then at each iteration re-evaluates the slope at the new point. Connecting all of these line segments gives an approximation of the coefficient. This one-step solver is quite efficient but may lack accuracy when an initial guess is not chosen carefully. Because this method is so rudimentary, more methods should be employed. Thus, having Euler's Method results gives us a good basis of comparison to other solvers.

The next ODE solver modifies Euler's method. This is called the Explicit Trapezoid Method (or modified Euler); where in place of the slope (from Euler's method) we use the average between

$$g(a_i) \qquad \text{and} \qquad g(a_i + hg(a_i)).$$

This is another one-step solver whose algorithm's quick output is beneficial.

While the Runge-Kutta Method of order 4 is only a one-step method, it is more accurate than Euler and Trapezoid methods because it is higher order. This scheme improves on Euler and Trapezoid method by using an

improved guess for the slope to decide what trajectory to follow. In the following scheme below, c_1 is simply from Euler's method (slope), while c_2 and c_3 is based on the slope at the midpoint of the interval. Lastly, c_4 is based on the slope at the end of the interval. In summation, the followed trajectory has improved from the previous two methods.

$$c_1 = g(a_i)$$
$$c_2 = g\left(a_i + \frac{h}{2}c_1\right)$$
$$c_3 = g\left(a_i + \frac{h}{2}c_2\right)$$
$$c_4 = g\left(a_i + hc_3\right)$$
$$a_{i+1} = a_i + \frac{h}{6}\left(c_1 + 2c_2 + 2c_3 + c_4\right)$$

At this point it is worth mentioning that the next solver on the list is in fact a built-in Matlab solver called ODE45. ODE45 is based on the Runge-Kutta method, so it's useful to compare the results from these two ODE solvers. Of course keeping in mind that the built-in solver uses different step-size and other parameters that may influence its efficiency.

Next, we will use Adams-Bashforth Two-Step Method. This multi-step method uses two function evaluations per iteration, resulting in an even more accurate solver. Being a multi-step method, it needs a one-step method (in this particular case, RK4 was used) within its algorithm to get it started.

$$c_1 = hg(a_0)$$
$$c_2 = hg\left(a_0 + \frac{c_1}{2}\right)$$
$$c_3 = hg\left(a_0 + \frac{c_2}{2}\right)$$
$$c_4 = hg(a_0 + c_3)$$
$$a_1 = a_0 + \frac{1}{6}\left(c_1 + 2c_2 + 2c_3 + c_4\right)$$

$$c_5 = hg(a_1)$$
$$c_6 = hg\left(a_1 + \frac{c_5}{2}\right)$$
$$c_7 = hg\left(a_1 + \frac{c_6}{2}\right)$$
$$c_8 = hg(a_1 + c_7)$$
$$a_2 = a_1 + \frac{1}{6}(c_5 + 2c_6 + 2c_7 + c_8)$$
$$a_{i+2} = a_{i+1} + h\left(\frac{3}{2}g(a_{i+1}) - \frac{1}{2}g(a_i)\right)$$

Similar to RK4's connection with ODE45, it is noteworthy to mention the parallel between Adams-Bashforth and ODE113. ODE113 is a built-in Matlab ODE solver, which is based on Adams-Bashforth and Adams-Moulton method. Thus, comparing these two numerical methods is useful in the results section of this work.

4.2 MOLS Results

The following tables and images represent the results obtained from each example below. For all examples, $n = 100$, the initial guess, a_0: a vector of ones was implemented and the step-size h remained fixed. Note also, that $\delta_2 = 1000 * \delta_1$. The column labeled error represents the L2 error.
In addition, in the accompanying tables, H represents the number of times the following part of the DE solver formula was implemented:

$$-(\nabla^2 J^{-1} \nabla J)$$

Recall that this means that the minimum eigenvalue was greater than δ_2.
Secondly, under M, it shows the number of times this piece of the solver was used:

$$-\alpha(t)(\nabla^2 J^{-1} \nabla J - \beta(t) \nabla J)$$

Where the conditions met were that the minimum eigenvalue was between δ_1 and δ_2.

Lastly, G gives the number of times the continuous gradient being used:

$$-\nabla J$$

In this case, the minimum eigenvalue was less than δ_1.

Example 1:

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

Figure 4.1: Example 1 - Euler

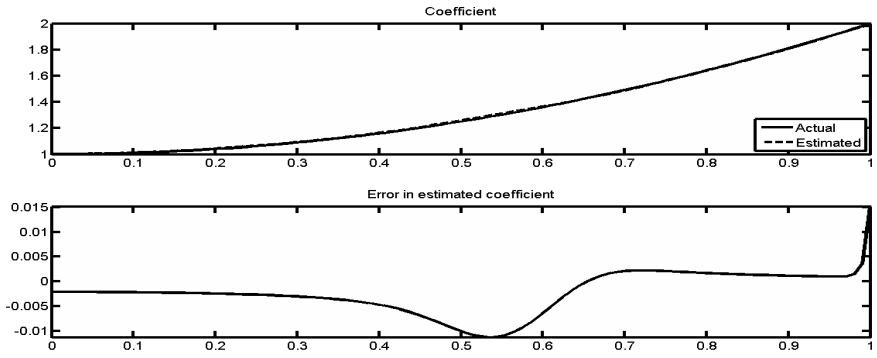


Figure 4.2: Example 1 - Trapezoid

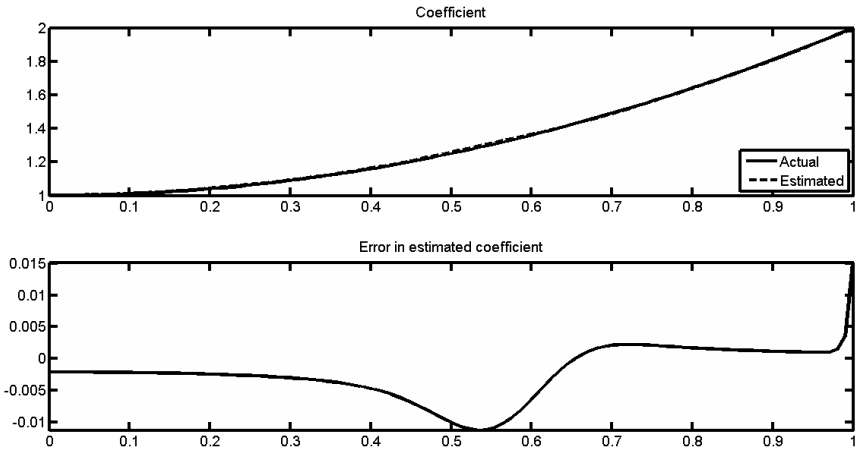


Figure 4.3: Example 1 - Adams-Bashforth

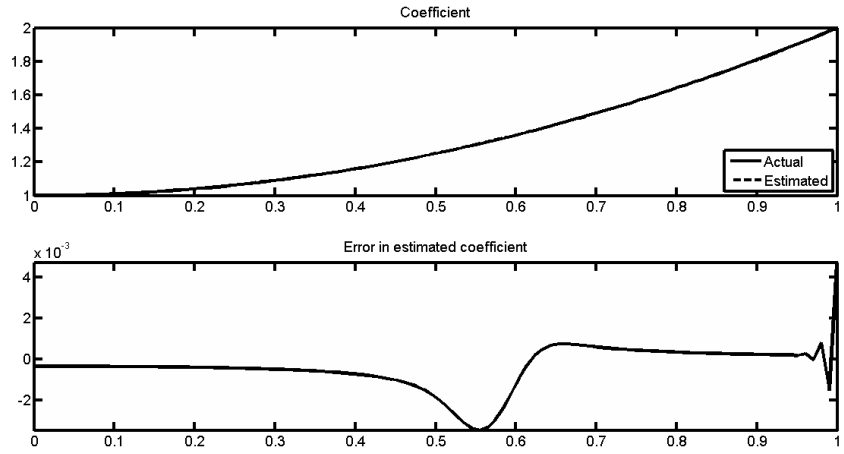


Figure 4.4: Example 1 - ODE113

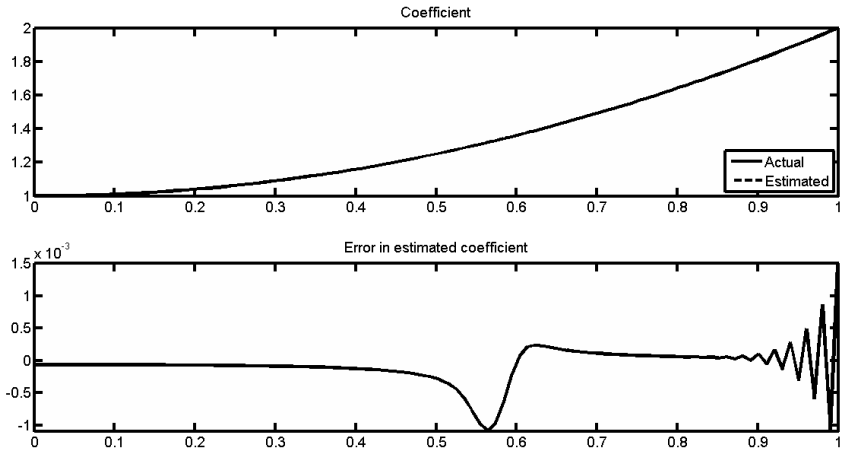


Figure 4.5: Example 1 - Runge-Kutta

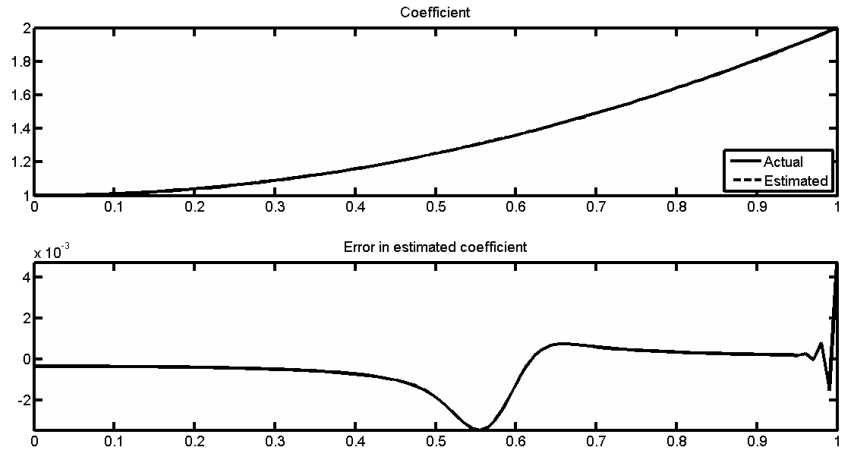


Figure 4.6: Example 1 - ODE 45

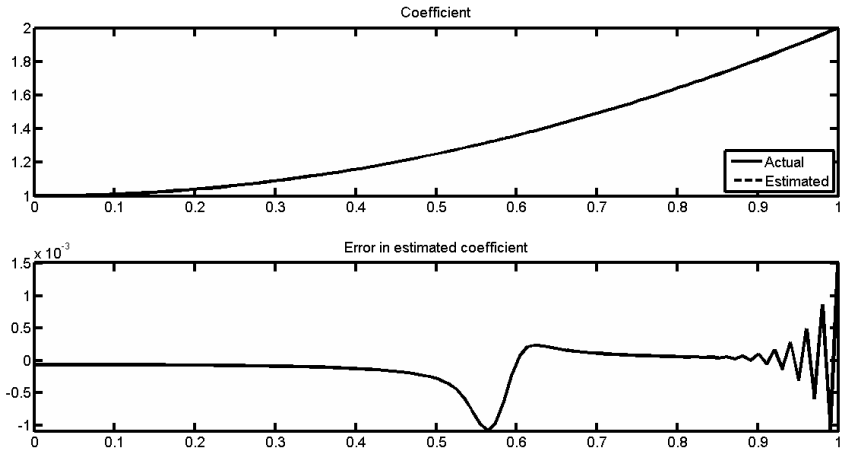


Table 4.1: Example 1 Results

Method	H	M	G	Reg Par	δ_1	Error	Time (secs)	λ_{min}
Euler	12	0	0	10^{-3}	10^{-9}	0.2136	2.0201	8.2631e-04
	15	0	0	10^{-4}	10^{-9}	0.0477	2.1165	2.7950e-04
	1	0	9999	10^{-5}	10^{-9}	fails	fails	0
Trapezoid	43	0	0	10^{-3}	10^{-9}	0.2136	5.7318	8.2631e-04
	43	0	0	10^{-4}	10^{-9}	0.0477	4.3338	2.7950e-04
	1	0	9999	10^{-5}	10^{-9}	fails	fails	0
R-K	29	0	0	10^{-3}	10^{-9}	0.2136	5.7272	8.2631e-04
	33	0	0	10^{-4}	10^{-9}	0.0477	5.9361	2.7950e-04
	30	0	0	10^{-5}	10^{-9}	0.0115	4.3791	9.1050e-05
	0	0	10000	10^{-6}	10^{-9}	fails	fails	0
ODE45	481	0	0	10^{-3}	10^{-9}	0.2136	17.7657	8.2631e-004
	499	0	0	10^{-4}	10^{-9}	0.0477	12.7612	2.7950e-004
	553	0	0	10^{-5}	10^{-9}	0.0115	16.6563	9.1050e-005
	673	0	0	10^{-6}	10^{-9}	0.0033	33.0203	2.9221e-005
A-B	1962	0	0	10^{-3}	10^{-9}	0.2136	177.5276	8.2631e-04
	1975	0	0	10^{-4}	10^{-9}	0.0477	101.4992	2.7950e-04
	1829	0	0	10^{-5}	10^{-9}	0.0115	96.7698	8.9808e-05
	1	0	10	10^{-6}	10^{-9}	fails	fails	0
ODE113	251	0	0	10^{-3}	10^{-9}	0.2136	10.7794	8.2631e-004
	293	0	0	10^{-4}	10^{-9}	0.0477	9.2306	2.7950e-004
	329	0	0	10^{-5}	10^{-9}	0.0115	9.8003	9.1050e-005
	457	0	0	10^{-6}	10^{-9}	0.0033	22.5014	2.9221e-005

4.2.1 Interpretation of Example 1

All four of the differential equation solver methods succeeded in finding an acceptable estimate of $a(x)$. For this example, dependent on the conditions imposed: the regularization parameter, δ_1, δ_2 the initial guess of a vector of ones; different levels of accuracy were achievable. Surprisingly, Euler's method could get the same results as the other methods. In addition, the fact that the scope of this problem is only 1-D may also attribute to the lack of difference in Euler to other more advanced methods. Further investigation on higher dimension problems may explain this behavior further. Although, notice in the chart that Runge-Kutta and Adams-Bashforth were able to reach a higher level of accuracy by using a smaller regularization parameter, while Euler and Trapezoid failed at these parameters due to computational problems.

The built in Matlab solvers compared well to the solvers we implemented here. When observing the initial results from Runge-Kutta compared with ODE45, they both were able to find the solution to the same degree of accuracy. In fact Runge-Kutta reached this level of accuracy in fewer steps than the built-in solver, as seen in the table. Both solvers ran through the differential equation method that satisfied the Newton direction; where the minimum eigenvalue of the hessian was greater than δ_2. Similarly to Runge-Kutta, the Adams-Bashforth scheme was able to reach the same accuracy as ODE113 from Matlab, but it took more iterations. This makes sense because Adams-Bashforth was a very basically written algorithm while the build-in solver uses advanced methods to pick a variable step-size and also switches between Adams-Bashforth and Adams-Moulton to solve. Again, Newton's direction was implemented for both.

By examining the minimum eigenvalue, it can be seen that the smaller the value, the more accurate the solver becomes.

Example 2:

$$\begin{aligned} f(x) &= 18x \\ a(x) &= e^{-3(x-x^3)} \\ u(x) &= e^{3(x-x^3)} - 1 \end{aligned}$$

Figure 4.7: Example 2 - Euler

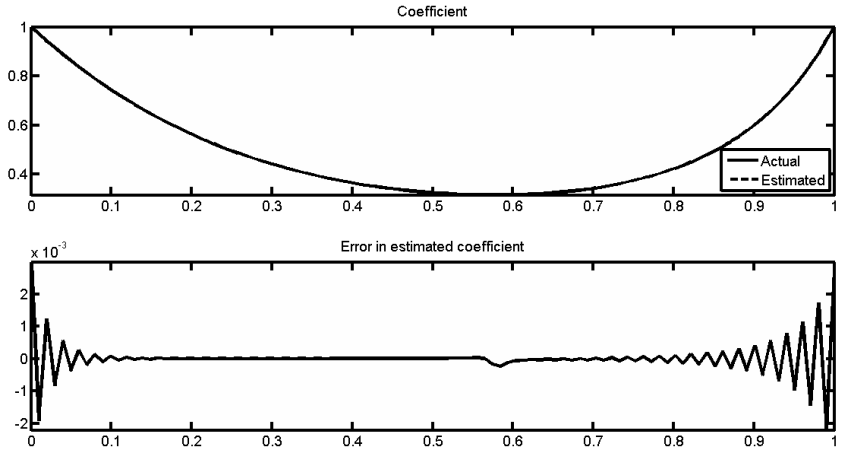


Figure 4.8: Example 2 - Trapezoid

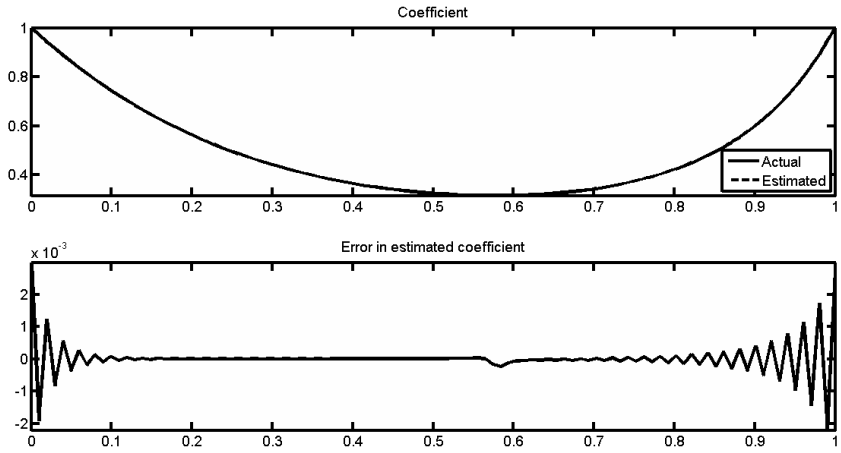


Figure 4.9: Example 2 - Adams-Bashforth

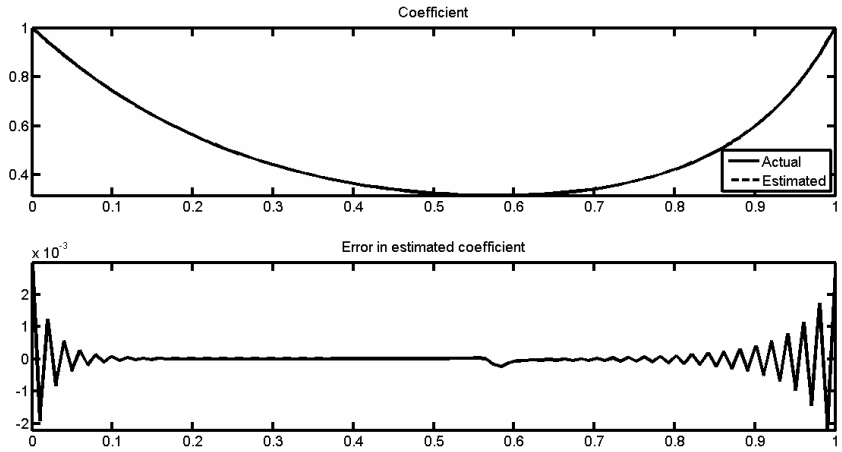


Figure 4.10: Example 2 - ODE113

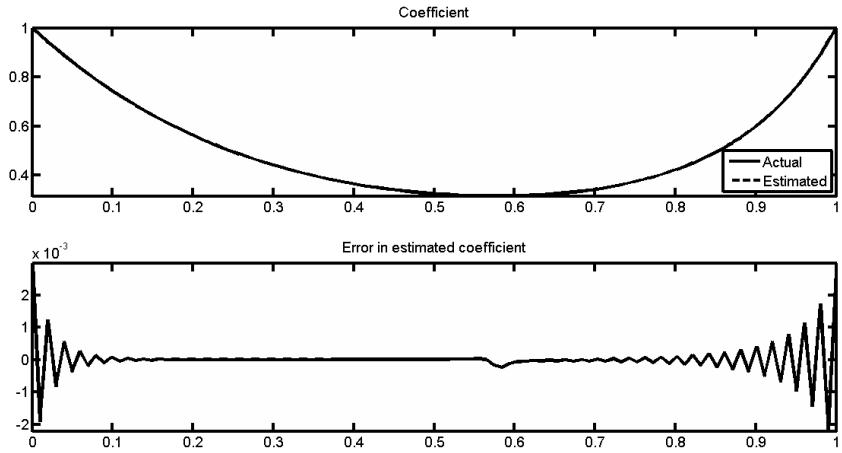


Figure 4.11: Example 2 - Runge-Kutta

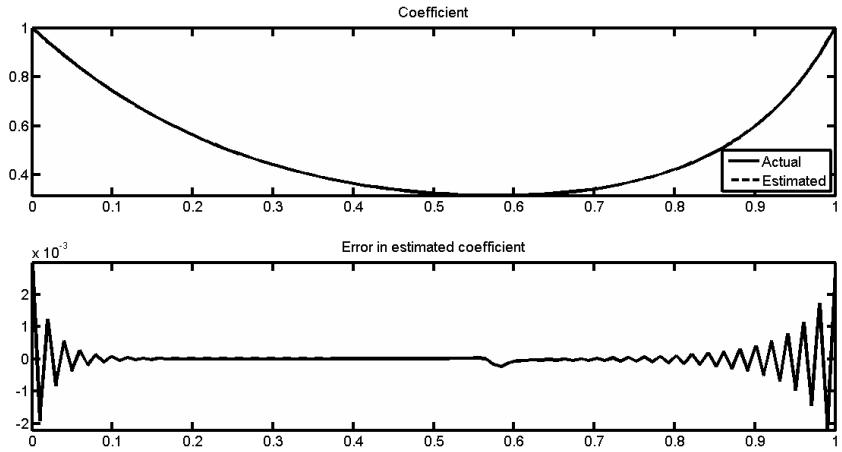


Figure 4.12: Example 2 - ODE 45

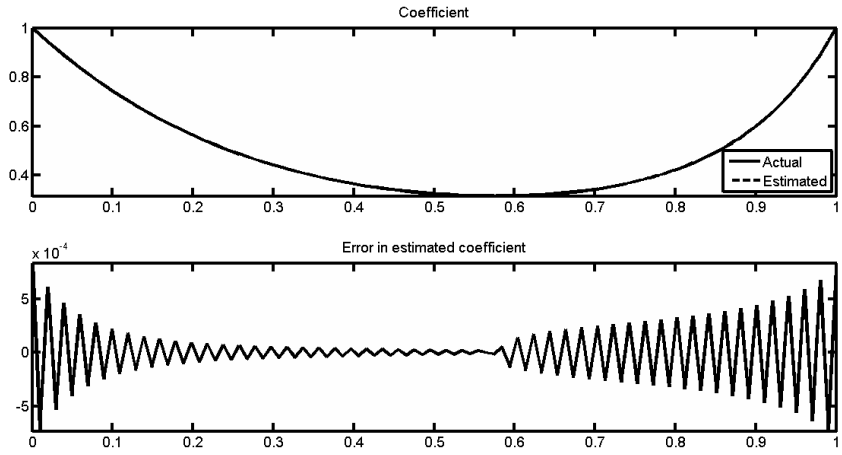


Table 4.2: Example 2 Results

Method	H	M	G	Reg Par	δ_1	Error	Time (secs)	λ_{min}
Euler	0	169	0	10^{-3}	10^{-4}	0.0433	6.1692	0.0031
	0	497	0	10^{-4}	10^{-4}	0.0148	24.0222	0.0010
	0	1476	0	10^{-5}	10^{-4}	0.0061	197.0011	3.2049e-04
Trapezoid	0	186	0	10^{-3}	10^{-4}	0.0433	12.3950	0.0031
	0	512	0	10^{-4}	10^{-4}	0.0148	48.2988	0.0010
	0	1486	0	10^{-5}	10^{-4}	0.0061	252.1864	3.2049e-04
R-K	0	185	0	10^{-3}	10^{-4}	0.0433	15.1927	0.0038
	0	511	0	10^{-4}	10^{-4}	0.0148	62.4613	0.0011
	0	1490	0	10^{-5}	10^{-4}	0.0061	520.1852	3.3244e-04
ODE45	631	0	0	10^{-3}	10^{-9}	0.0433	13.4235	0.0148
	625	0	0	10^{-4}	10^{-9}	0.0148	19.3598	0.0052
	613	0	0	10^{-5}	10^{-9}	0.0061	49.1400	0.0017
	613	0	0	10^{-6}	10^{-9}	0.0031	96.0823	3.9329e-004
A-B	0	420	0	10^{-3}	10^{-4}	0.0433	18.5956	0.0038
	0	501	0	10^{-4}	10^{-4}	0.0148	35.2438	0.0011
	0	1480	0	10^{-5}	10^{-4}	0.0061	257.3643	3.3244e-04
ODE113	359	0	0	10^{-3}	10^{-9}	0.0433	8.8912	0.0148
	387	0	0	10^{-4}	10^{-9}	0.0148	13.1963	0.0052
	384	0	0	10^{-5}	10^{-9}	0.0061	31.3867	0.0017

4.2.2 Interpretation of Example 2

The same general results hold for Example 2 as in Example 1. One difference is the fact that using $\delta_1 = 10^{-9}$ fails. So, in following the suggestion of [42], instead, $\delta_1 = 10^{-4}$ was used. In addition, due to this change in the problem and δs, now for Euler, Trapezoid, Runge-Kutta, and Adams-Bashforth, the mixed Newton's direction and continuous gradient was implemented. This proved to be slower and take more iterations than the previous example, when using the Newton's direction. This can be attributed to either the complexity of the example or the fact that mixing the two directions in a convex combination simply takes more time than the direct Newton's direction.

Example 3:

$$\begin{aligned} f(x) &= 16\pi^2 \sin(4\pi x)(\cos(4\pi x) + 1) \\ a(x) &= \frac{1}{2}(\cos(4\pi x) + 1) \\ u(x) &= \sin(4\pi x) \end{aligned}$$

Figure 4.13: Example 3 - Euler

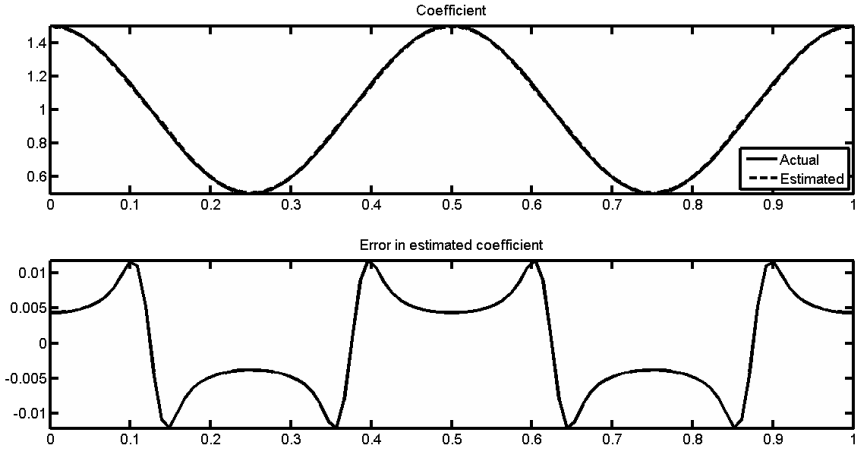


Figure 4.14: Example 3 - Trapezoid

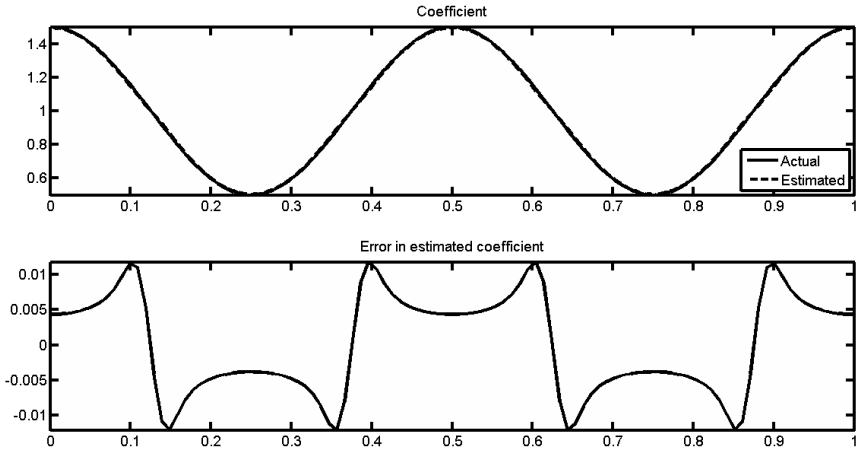


Figure 4.15: Example 3 - Adams-Bashforth

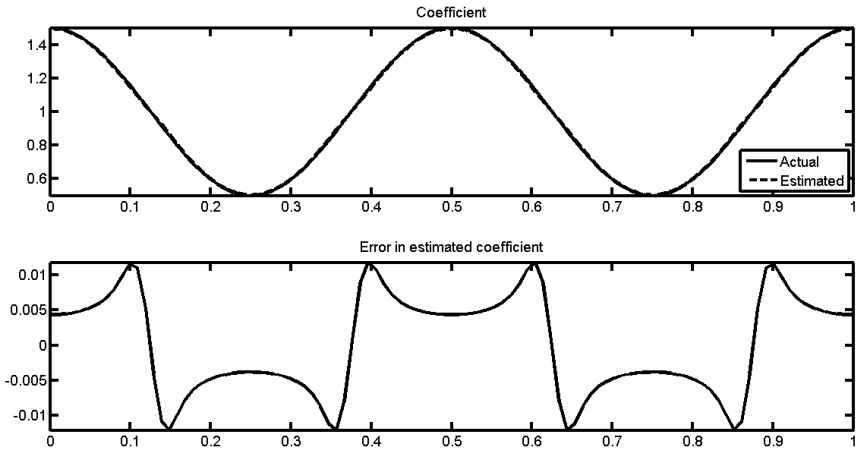


Figure 4.16: Example 3 - ODE113

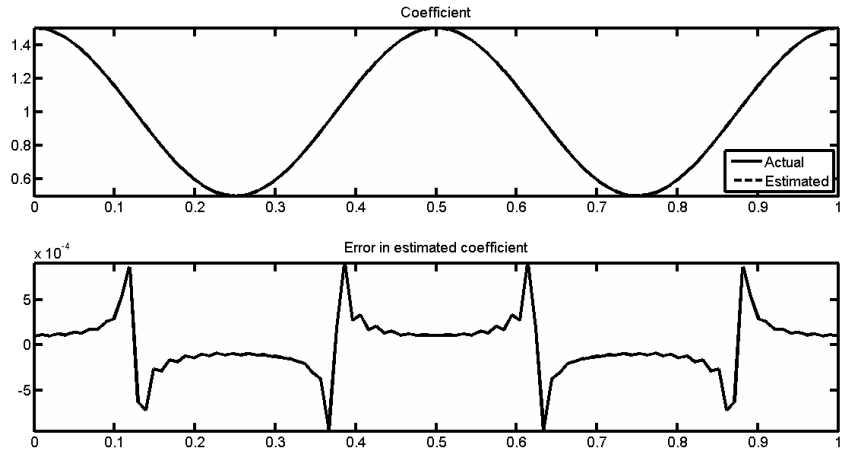


Figure 4.17: Example 3 - Runge-Kutta

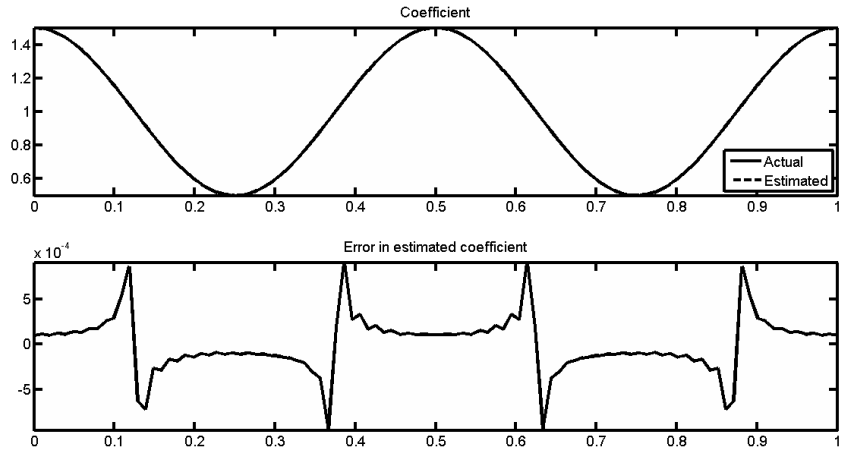


Figure 4.18: Example 3 - ODE 45

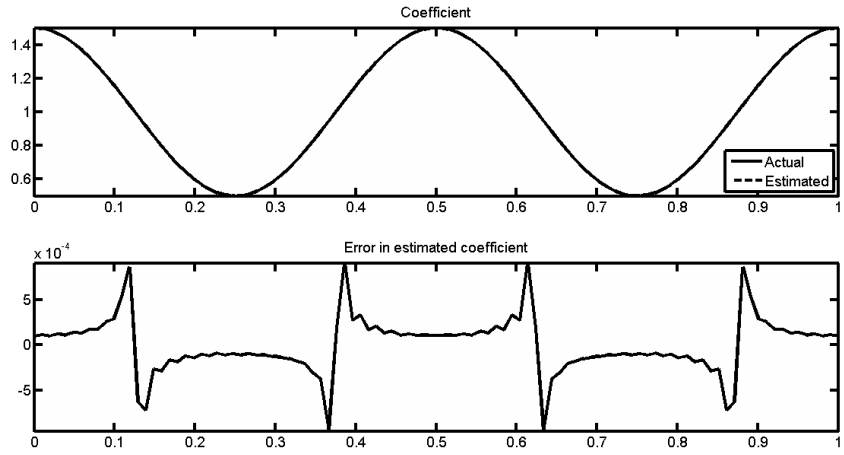


Table 4.3: Example 3 Results

Method	H	M	G	Reg Par	δ_1	Error	Time (secs)	λ_{min}
Euler	10	0	0	10^{-2}	10^{-9}	0.2998	1.8706	0.1398
	13	0	0	10^{-3}	10^{-9}	0.0707	2.0313	0.0486
Trapezoid	46	0	0	10^{-2}	10^{-9}	0.2998	4.8511	0.1398
	46	0	0	10^{-3}	10^{-9}	0.0707	5.8137	0.0486
R-K	33	0	0	10^{-2}	10^{-9}	0.2998	4.7942	0.1486
	33	0	0	10^{-3}	10^{-9}	0.0707	5.9509	0.0486
	0	164	0	10^{-4}	10^{-4}	0.0160	52.3007	0.0155
	0	646	0	10^{-5}	10^{-4}	0.0032	719.2322	0.0037
ODE45	517	0	0	10^{-2}	10^{-9}	0.2998	16.5789	0.1493
	541	0	0	10^{-3}	10^{-9}	0.0707	22.6468	0.0486
	607	0	0	10^{-4}	10^{-9}	0.0160	73.2117	0.0155
	829	0	0	10^{-5}	10^{-9}	0.0032	295.1082	0.0037
A-B	1930	0	0	10^{-2}	10^{-9}	0.2998	75.0973	0.1493
	1947	0	0	10^{-3}	10^{-9}	0.0707	100.3444	0.0483
ODE113	278	0	0	10^{-2}	10^{-9}	0.2998	10.0676	0.1493
	301	0	0	10^{-3}	10^{-9}	0.0707	13.5218	0.0486
	366	0	0	10^{-4}	10^{-9}	0.0160	31.9196	0.0155
	504	0	0	10^{-5}	10^{-9}	0.0032	150.8617	0.0037

Example 4:

$$\begin{aligned} f(x) &= \ln(x+2)(12x^2-6x) + \frac{1}{(x+2)(4x^3-3x^2)} \\ a(x) &= \ln(x+2) \\ u(x) &= x^4 - x^3 \end{aligned}$$

Figure 4.19: Example 4 - Euler

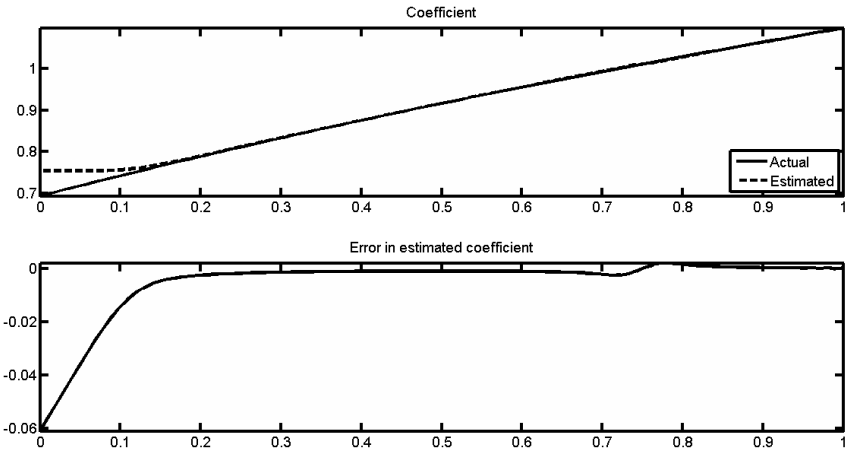


Figure 4.20: Example 4 - Trapezoid

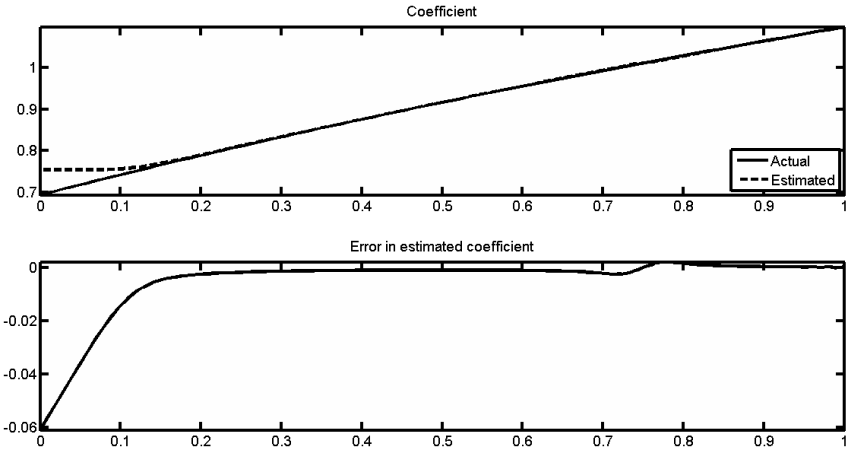


Figure 4.21: Example 4 - Adams-Bashforth

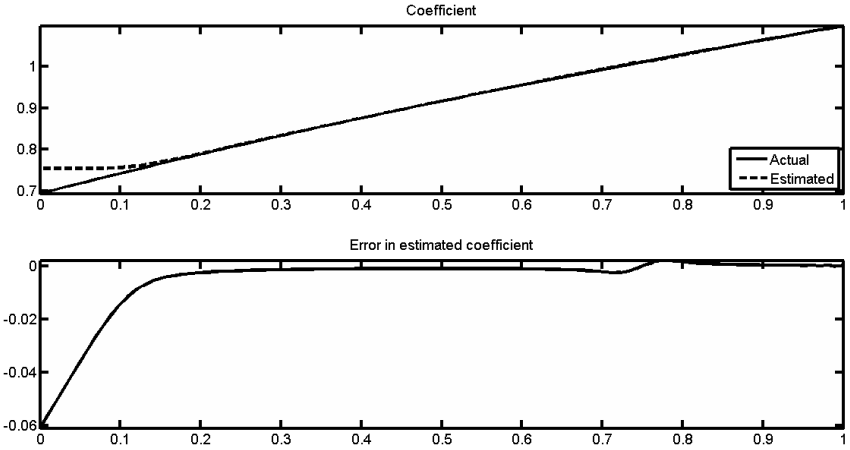


Figure 4.22: Example 4 - ODE113

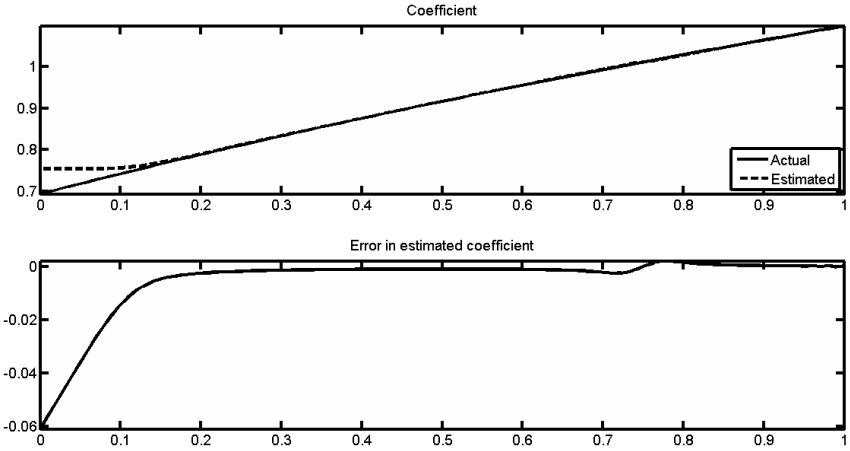


Figure 4.23: Example 4 - Runge-Kutta

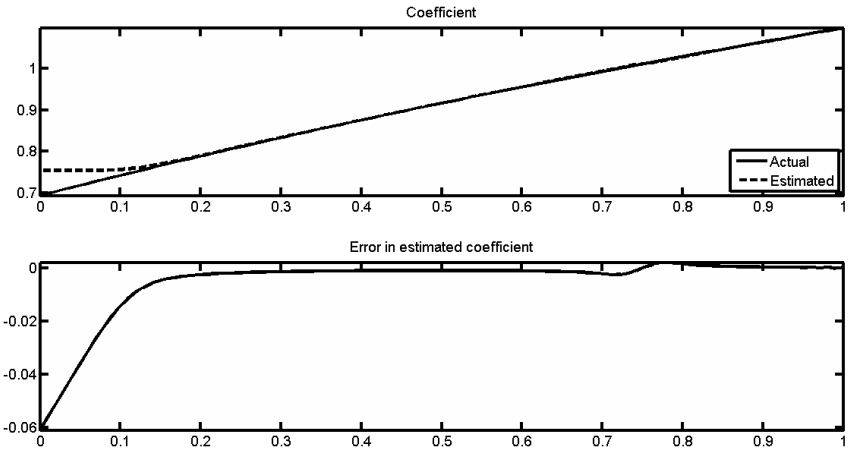


Figure 4.24: Example 4 - ODE 45

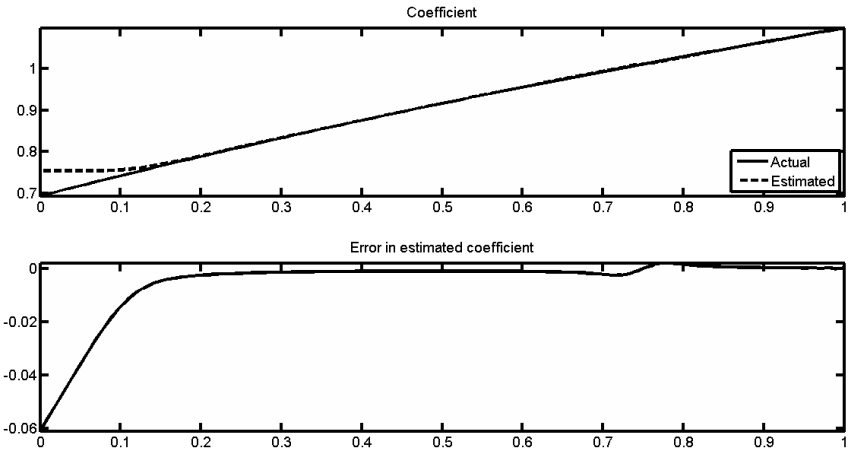


Table 4.4: Example 4 Results

Method	H	M	G	Reg Par	δ_1	Error	Time (secs)	λ_{min}
Euler	11	0	0	10^{-4}	10^{-9}	0.4387	2.3078	2.2496e-05
	67	0	0	10^{-5}	10^{-9}	0.2371	4.2472	5.8247e-06
	39	0	0	10^{-6}	10^{-9}	0.1334	3.6849	1.3498e-06
Trapezoid	43	0	0	10^{-4}	10^{-9}	0.4387	9.9559	2.2496e-05
	46	0	0	10^{-5}	10^{-9}	0.2371	8.2569	5.6714e-06
	44	0	0	10^{-6}	10^{-9}	0.1334	10.9617	1.3498e-06
R-K	33	0	0	10^{-4}	10^{-9}	0.4387	7.3837	2.3939e-05
	30	0	0	10^{-5}	10^{-9}	0.2371	5.5857	5.7427e-06
	42	0	0	10^{-6}	10^{-9}	0.1334	9.4129	1.3498e-06
ODE45	439	0	0	10^{-4}	10^{-9}	0.4387	22.3847	2.5301e-05
	445	0	0	10^{-5}	10^{-9}	0.2371	14.5081	6.0478e-06
	511	0	0	10^{-6}	10^{-9}	0.1334	21.8680	1.3498e-06
A-B	1939	0	0	10^{-4}	10^{-9}	0.4387	207.2714	2.3939e-05
	2188	0	0	10^{-5}	10^{-9}	0.2371	163.6906	5.7427e-06
	1859	0	0	10^{-6}	10^{-9}	0.1334	171.3274	1.2955e-06
ODE113	261	0	0	10^{-4}	10^{-9}	0.4387	11.9599	2.5301e-05
	255	0	0	10^{-5}	10^{-9}	0.2371	9.2590	6.0478e-06
	304	0	0	10^{-6}	10^{-9}	0.1334	14.6249	1.3498e-06

4.2.3 Interpretation of Example 3 & 4

Acceptable results were shown in Example 3 and 4 as well. Dependent on the problem, either $\delta_1 = 1.0e-9$ or $\delta_1 = 1.0e-4$. And dependent on δ_1, δ_2 and $\lambda_{min}(a)$ one of the three trajectories was chosen. Notice that none of the examples using Modified Output Least Squares implemented the third trajectory: continuous gradient method. With MOLS being strongly convex, this is to be expected because of the acceptability of the hessian.

Table 4.5: Continuous Gradient Method

	Method	Regularization Parameter	Iterations	Time (secs)	Error
1	Euler	$1.0e-4$	10000	98.8505	0.0742
	Trap	$1.0e-4$	10000	168.7705	0.0743
	Runge-Kutta	$1.0e-4$	10000	266.5713	0.0743
	Adams-Bashforth	$1.0e-4$	9999	148.2903	0.0743

Figure 4.25: Continuous Gradient - Euler

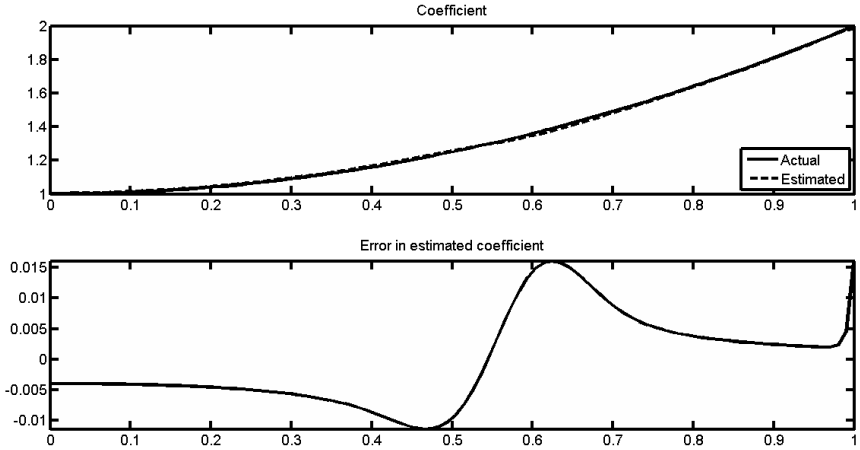


Figure 4.26: Continuous Gradient - Trapezoid

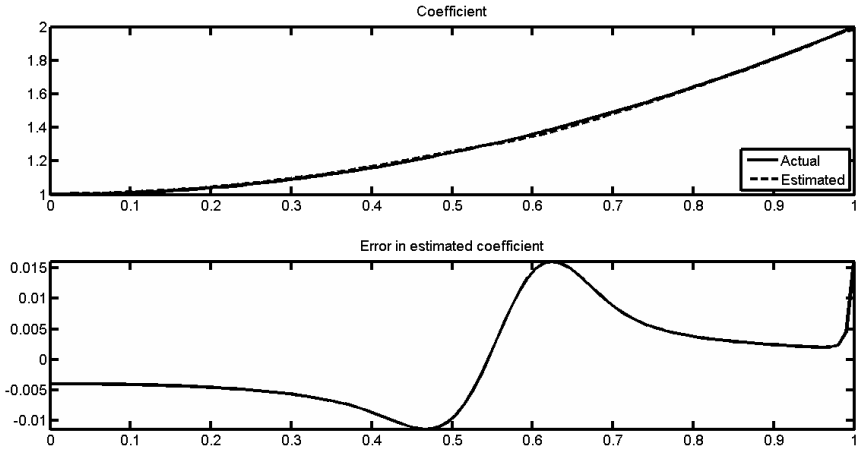


Figure 4.27: Continuous Gradient - Adams-Bashforth

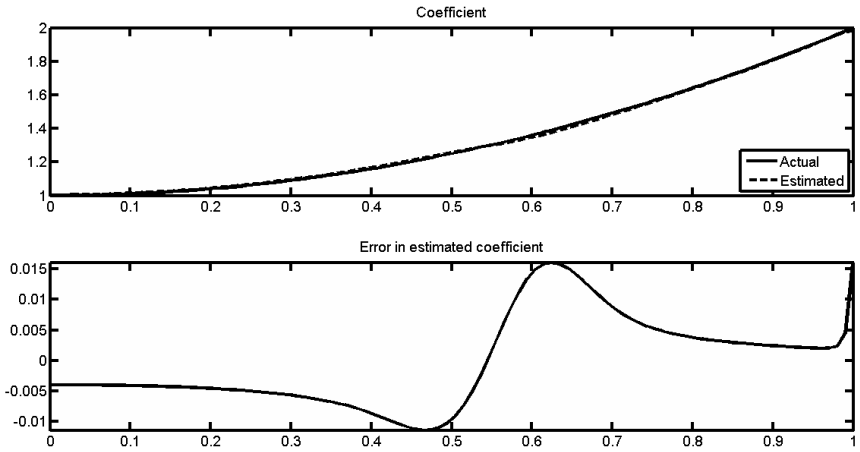
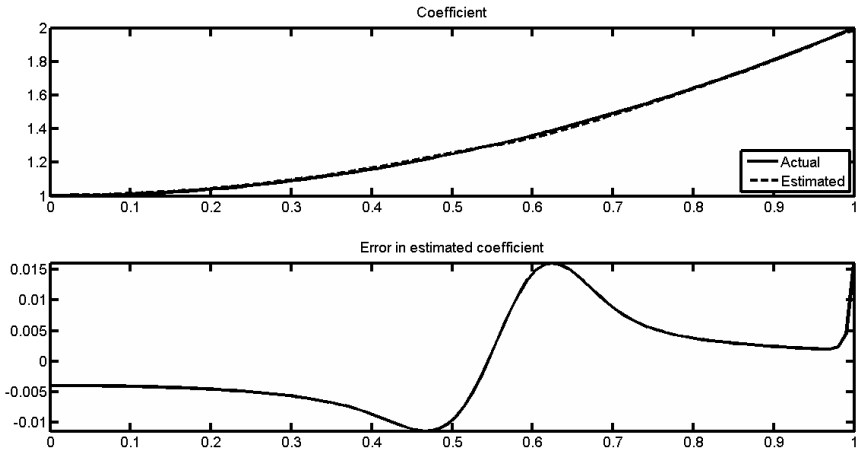


Figure 4.28: Continuous Gradient - Runge-Kutta



4.2.4 Comparison of Continuous Gradient Method with Newton-Type Method

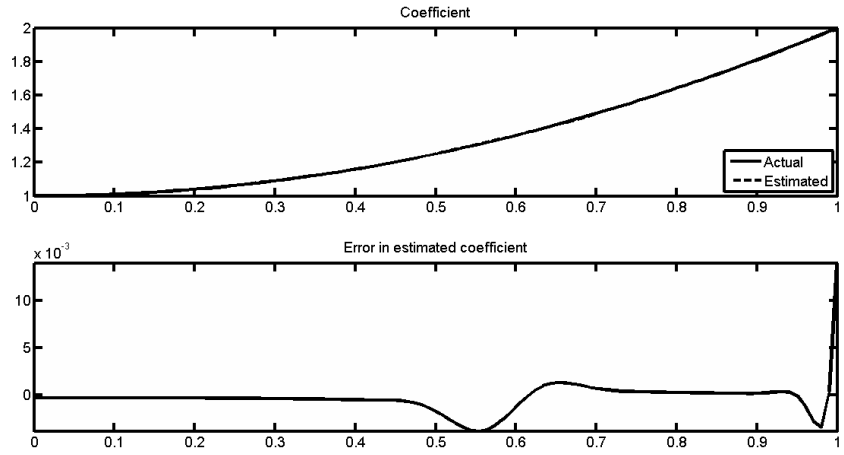
Clearly, the continuous gradient method performs at the same degree of accuracy as the implemented Newton-Type Method. But, the draw-back comes from the fact that the gradient method almost always used the maximum number of iterations to reach this level of accuracy (10,000). Therefore, the benefits of this new method allow for improvements in solving optimization problems via ODEs.

4.3 OLS Results

Table 4.6: OLS Results

	Method	H	M	G	Reg Par	δ_1	Error	Time (secs)	λ_{min}
1	Euler	0	428	942	10^{-8}	10^{-9}	0.0182	60.8834	-1.7848e-05
	Trapezoid	0	438	927	10^{-8}	10^{-9}	0.0182	123.3485	-1.7848e-05
	R-K	0	443	926	10^{-8}	10^{-9}	0.0182	224.3870	-1.7747e-05
	A-B	0	434	948	10^{-8}	10^{-9}	0.0182	114.8002	-1.7747e-05

Figure 4.29: Example 1 - OLS



4.4 Equation Error Results

Table 4.7: Equation Error Results

	Method	H	M	G	Reg Par	δ_1	Error	Time (secs)	λ_{min}
1	Euler	6	0	0	10^{-8}	10^{-9}	$8.3342e-4$	2.9104	$3.4411e-6$
	Trapezoid	27	0	0	10^{-8}	10^{-9}	$8.3341e-4$	3.2057	$3.4411e-6$
	R-K	19	0	0	10^{-8}	10^{-9}	$8.3341e-4$	3.8358	$3.4411e-6$
	A-B	1193	0	0	10^{-8}	10^{-9}	$8.3341e-4$	54.4534	$3.4411e-6$
2	Euler	7	0	0	10^{-8}	10^{-9}	0.0025	1.6316	$3.9565e-6$
	Trapezoid	34	0	0	10^{-8}	10^{-9}	0.0025	2.4885	$3.9565e-6$
	R-K	24	0	0	10^{-8}	10^{-9}	0.0025	2.8894	$3.9565e-6$
	A-B	1525	0	0	10^{-8}	10^{-9}	0.0025	46.6141	$3.9565e-6$
3	Euler	10	0	0	10^{-8}	10^{-9}	2.0391e-4	1.6288	$3.8708e-6$
	Trapezoid	46	0	0	10^{-8}	10^{-9}	2.0391e-4	2.8118	$3.8708e-6$
	R-K	32	0	0	10^{-8}	10^{-9}	2.0391e-4	3.3339	$3.8708e-6$
	A-B	2004	0	0	10^{-8}	10^{-9}	2.0391e-4	60.0253	$3.8708e-6$
4	Euler	0	10000	0	10^{-10}	10^{-9}	0.0151	221.4075	$2.5969e-9$
	Trapezoid	0	10000	0	10^{-10}	10^{-9}	0.0151	379.0178	$2.5969e-9$
	R-K	0	10000	0	10^{-10}	10^{-9}	0.0151	641.0123	$2.5969e-9$
	A-B	0	9999	0	10^{-10}	10^{-9}	0.0151	401.1844	$2.5969e-9$

Figure 4.30: Example 1 - EE

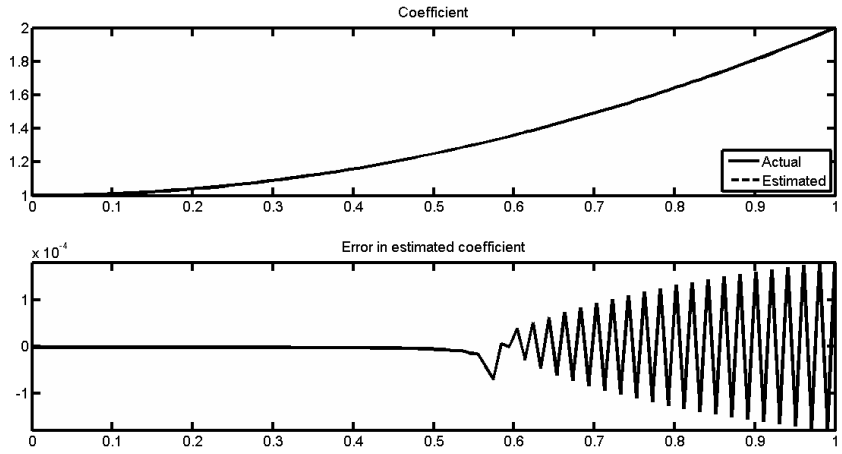


Figure 4.31: Example 2 - EE

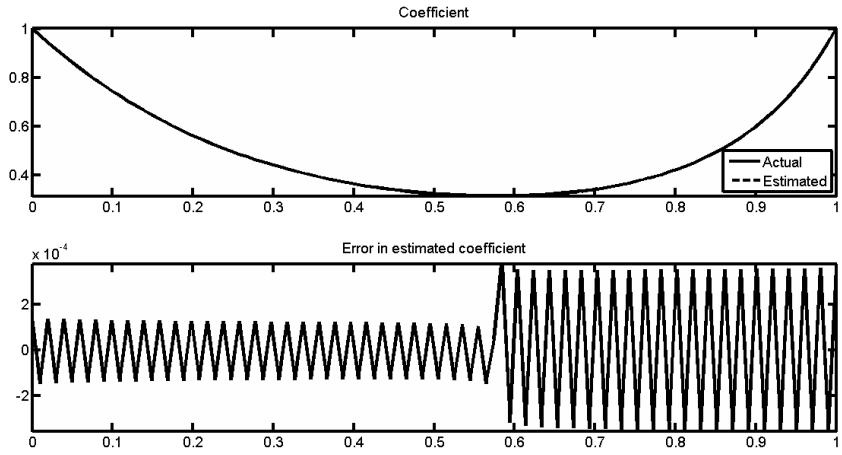


Figure 4.32: Example 3 - EE

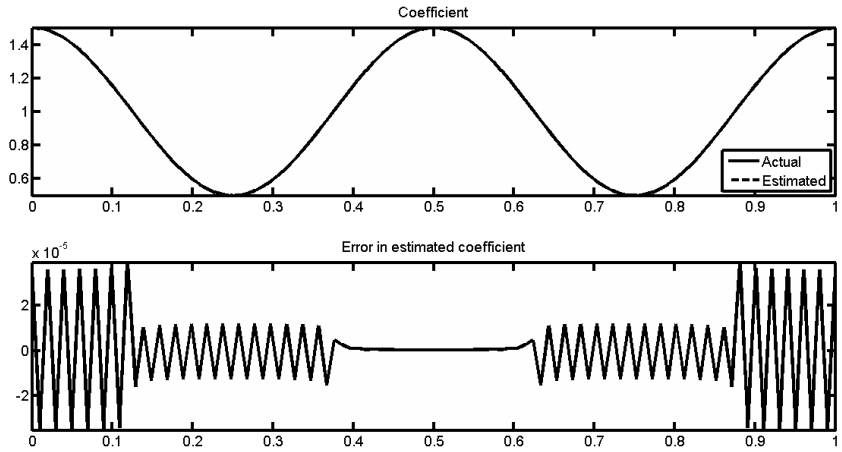
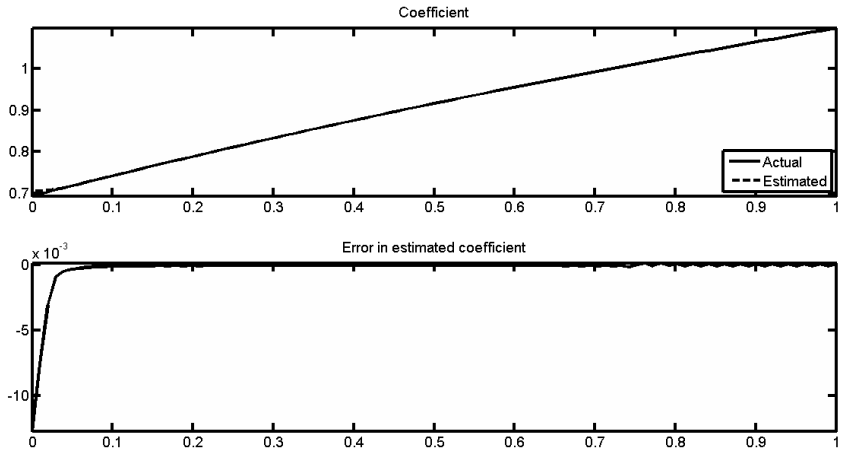


Figure 4.33: Example 4 - EE

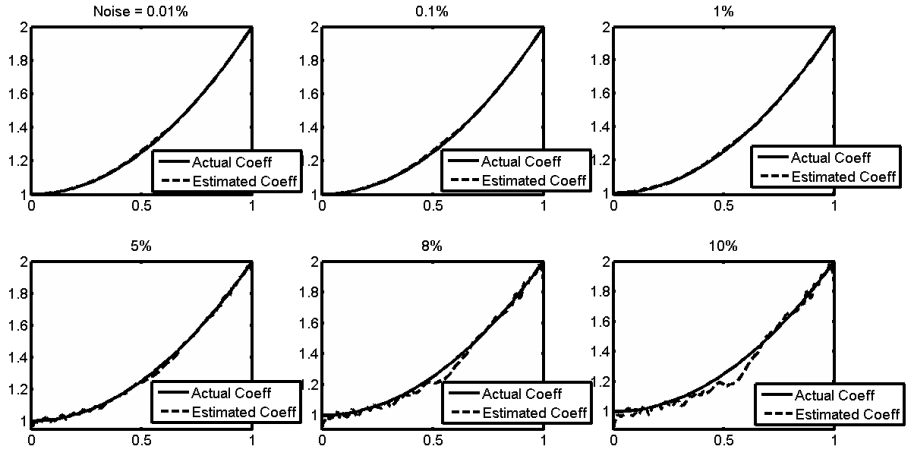


4.5 Noise added to Example 1

Table 4.8: Runge-Kutta Added Noise

Regularization Parameter	Noise	MOLS	OLS	Equation Error
$1.0e-4$	$1.0e-4$	0.0477	0.9109	0.0374
$1.0e-4$	0.001	0.0483	0.9108	0.0379
$1.0e-4$	0.01	0.0563	0.9105	0.0606
$1.0e-4$	0.05	0.1555	0.9092	0.2967
$1.0e-4$	0.08	0.3183	0.9092	0.5912
$1.0e-4$	0.1	0.4679	0.9080	0.8331
$1.0e-5$	$1.0e-4$	0.0118	0.2484	0.0093
$1.0e-5$	0.001	0.0160	0.2485	0.0152
$1.0e-5$	0.01	0.0898	0.2498	0.1108
$1.0e-5$	0.05	0.4953	0.2563	0.7086
$1.0e-5$	0.08	0.9250	0.2623	1.3286
$1.0e-5$	0.1	fails	0.2667	1.7604
$5.0e-5$	$1.0e-4$	0.0309	0.6405	0.0243
$5.0e-5$	0.001	0.0319	0.6405	0.0255
$5.0e-5$	0.01	0.0527	0.6411	0.0657
$5.0e-5$	0.05	0.2293	0.6438	0.3838
$5.0e-5$	0.08	0.4518	0.6462	0.7586
$5.0e-5$	0.1	0.6463	0.6479	1.0545

Figure 4.34: Added Noise using MOLS (reg par = 10^{-4})



4.6 A Fourth-Order Example

$$
\begin{aligned}
a(x) &= x+1 \\
u(x) &= -\cos(2\pi x)+1 \\
f(x) &= -16\pi^3 \sin(2\pi x) - 16\pi^4 (x+1)\cos(2\pi x)
\end{aligned}
$$

Figure 4.35: Equation Error - Runge-Kutta

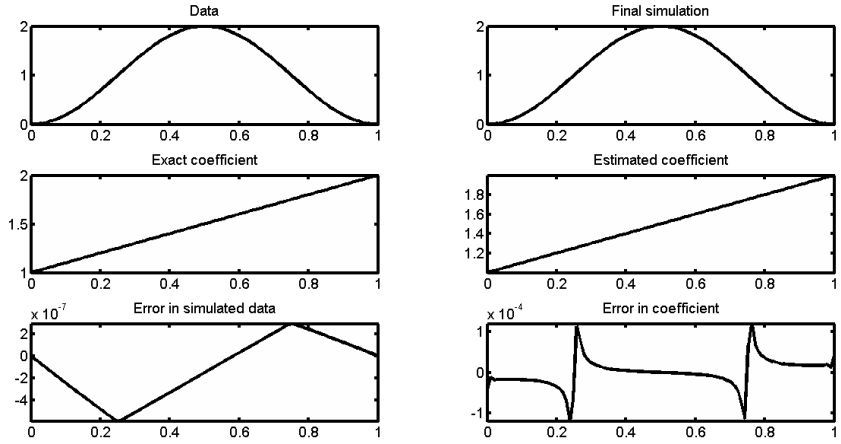


Table 4.9: Fourth-Order Example (EE)

Method	H	M	G	Reg Par	δ_1	Error	Time (secs)	λ_{min}
Euler	6	0	0	$1.0e-4$	$1.0e-9$	$3.5136e-4$	0.6761	0.0206
Trapezoid	1752	0	0	$1.0e-4$	$1.0e-9$	$3.5139e-4$	94.8177	0.0206
Runge-Kutta	23	0	0	$1.0e-4$	$1.0e-9$	$3.5136e-4$	2.2873	0.0206
Adams-Bashforth	1302	0	0	$1.0e-4$	$1.0e-9$	$3.5136e-4$	50.5602	0.0206

Bibliography

[1] M. Aassila. A dynamical approach for the stability of second order dissipative systems. *Differential Integral Equations*, 15(4):463–475, 2002.

[2] S. Aizicovici, S. Reich, and A.J. Zaslavski. Minimizing convex functions by continuous descent methods. *Electron. J. Differential Equations*, pages No. 19, 7, 2010.

[3] F. Alvarez. On the minimizing property of a second order dissipative system in Hilbert spaces. *SIAM J. Control Optim.*, 38(4):1102–1119 (electronic), 2000.

[4] F. Alvarez and A. Cabot. Steepest descent with curvature dynamical system. *J. Optim. Theory Appl.*, 120(2):247–273, 2004.

[5] F. Alvarez D. and J. M. Pérez C. A dynamical system associated with Newton's method for parametric approximations of convex minimization problems. *Appl. Math. Optim.*, 38(2):193–217, 1998.

[6] A. S. Antipin. Minimization of convex functions on convex sets by means of differential equations. *Differentsial˘nye Uravneniya*, 30(9):1475–1486, 1652, 1994.

[7] H. Attouch and F. Alvarez. The heavy ball with friction dynamical system for convex constrained minimization problems. In *Optimization (Namur, 1998)*, volume 481 of *Lecture Notes in Econom. and Math. Systems*, pages 25–35. Springer, Berlin, 2000.

[8] H. Attouch, J. Bolte, and P. Redont. Optimizing properties of an inertial dynamical system with geometric damping. Link with proximal

methods. *Control Cybernet.*, 31(3):643–657, 2002. Well-posedness in optimization and related topics (Warsaw, 2001).

[9] H. Attouch and R. Cominetti. A dynamical approach to convex minimization coupling approximation with the steepest descent method. *J. Differential Equations*, 128(2):519–540, 1996.

[10] H. Attouch, X. Goudou, and P. Redont. The heavy ball with friction method. I. The continuous dynamical system: global exploration of the local minima of a real-valued function by asymptotic analysis of a dissipative dynamical system. *Commun. Contemp. Math.*, 2(1):1–34, 2000.

[11] H. Attouch, J. Peypouquet, and P. Redont. A dynamical approach to an inertial forward-backward algorithm for convex minimization. *SIAM J. Optim.*, 24(1):232–256, 2014.

[12] H. Attouch and B. F. Svaiter. A continuous dynamical Newton-like approach to solving monotone inclusions. *SIAM J. Control Optim.*, 49(2):574–598, 2011.

[13] H. Attouch and M. Teboulle. Regularized Lotka-Volterra dynamical system as continuous proximal-like method in optimization. *J. Optim. Theory Appl.*, 121(3):541–570, 2004.

[14] C. A. Botsaris. Differential gradient methods. *J. Math. Anal. Appl.*, 63(1):177–198, 1978.

[15] A. A. Brown and M. C. Bartholomew-Biggs. ODE versus SQP methods for constrained optimization. *J. Optim. Theory Appl.*, 62(3):371–386, 1989.

[16] A. A. Brown and M. C. Bartholomew-Biggs. Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations. *J. Optim. Theory Appl.*, 62(2):211–224, 1989.

[17] T. S. Coffey, C. T. Kelley, and D. E. Keyes. Pseudotransient continuation and differential-algebraic equations. *SIAM J. Sci. Comput.*, 25(2):553–569, 2003.

[18] I. Diener. On the global convergence of path-following methods to determine all solutions to a system of nonlinear equations. *Math. Programming*, 39(2):181–188, 1987.

[19] I. Diener and R. Schaback. An extended continuous Newton method. *J. Optim. Theory Appl.*, 67(1):57–77, 1990.

[20] S. D. Flåm. Solving convex programs by means of ordinary differential equations. *Math. Oper. Res.*, 17(2):290–302, 1992.

[21] M. P. Glazos, S. Hui, and S. H. Żak. Sliding modes in solving convex programming problems. *SIAM J. Control Optim.*, 36(2):680–697 (electronic), 1998.

[22] M. S. Gockenbach and A. A. Khan. Identification of Lamé parameters in linear elasticity: a fixed point approach. *J. Ind. Manag. Optim.*, 1(4):487–497, 2005.

[23] M.S. Gockenback, B. Jadamba, and A.A. Khan. A comparative numerical study of optimisation approaches for elliptic inverse problems. *JMI Inter. J. Math. Sci.*, 1(1):35–56, 2010.

[24] T. Hajba. Optimization methods modeled by second order differential equation. *Ann. Univ. Sci. Budapest. Sect. Comput.*, 26:145–158, 2006.

[25] T. Hajba. Parameter analysis of optimization methods modeled by second order differential equation. *Alkalmazott Mat. Lapok*, 25(2):61–79, 2008.

[26] T. Hajba. Optimizing second-order differential equation systems. *Electron. J. Differential Equations*, pages No. 44, 16, 2011.

[27] Q. Han, L. Z. Liao, H. Qi, and L. Qi. Stability analysis of gradient-based neural networks for optimization problems. *J. Global Optim.*, 19(4):363–381, 2001.

[28] A. Haraux and M. A. Jendoubi. Convergence of solutions of second-order gradient-like systems with analytic nonlinearities. *J. Differential Equations*, 144(2):313–320, 1998.

[29] A.C. Hindmarsh. Odepack, a systematized collection of ode solvers. *Scientific Computing*, pages 55–64, 1982.

[30] Reinhard Kluge. An inverse problem for "coefficients" in linear equations. Uniqueness and iterative solution. In *Nonlinear analysis (Berlin, 1979)*, volume 2 of *Abh. Akad. Wiss. DDR, Abt. Math. Naturwiss. Tech., 1981*, pages 139–148. Akademie-Verlag, Berlin, 1981.

[31] L. Z. Liao. A continuous method for convex programming problems. *J. Optim. Theory Appl.*, 124(1):207–226, 2005.

[32] Li-Zhi Liao, Liqun Qi, and Hon Wah Tam. A gradient-based continuous method for large-scale optimization problems. *J. Global Optim.*, 31(2):271–286, 2005.

[33] Yi-gui Ou and Guan-shu Wang. A hybrid ODE-based method for unconstrained optimization problems. *Comput. Optim. Appl.*, 53(1):249–270, 2012.

[34] Yigui Ou. A superlinearly convergent ODE-type trust region algorithm for nonsmooth nonlinear equations. *J. Appl. Math. Comput.*, 22(3):371–380, 2006.

[35] Yigui Ou, Qian Zhou, and Haichan Lin. An ODE-based trust region method for unconstrained optimization problems. *J. Comput. Appl. Math.*, 232(2):318–326, 2009.

[36] S. Schäffler and H. Warsitz. A trajectory-following method for unconstrained optimization. *J. Optim. Theory Appl.*, 67(1):133–140, 1990.

[37] J. Schropp. A dynamical systems approach to constrained minimization. *Numer. Funct. Anal. Optim.*, 21(3-4):537–551, 2000.

[38] Z.-J. Shi. Convergence of multi-step curve search method for unconstrained optimization. *J. Numer. Math.*, 12(4):297–309, 2004.

[39] V. Shikhman and O. Stein. Constrained optimization: projected gradient flows. *J. Optim. Theory Appl.*, 140(1):117–130, 2009.

[40] A. N. Tihonov. On the regularization of ill-posed problems. *Dokl. Akad. Nauk SSSR*, 153:49–52, 1963.

[41] A. N. Tihonov and F. P. Vasil˜ev. Methods for the solution of ill-posed extremal problems. In *Mathematical models and numerical methods (Papers, Fifth Semester, Stefan Banach Internat. Math. Center, Warsaw, 1975)*, volume 3 of *Banach Center Publ.*, pages 297–342. PWN, Warsaw, 1978.

[42] Lei-Hong Zhang, C. T. Kelley, and Li-Zhi Liao. A continuous Newton-type method for unconstrained optimization. *Pac. J. Optim.*, 4(2):259–277, 2008.