USING HOMOTOPY METHODS TO SOLVE
NONSMOOTH EQUATIONS

by

Adam Speight

B.S., University of Colorado at Denver, 1997

A thesis submitted to the
University of Colorado at Denver
in partial fulfillment
of the requirements for the degree of
Master of Science
Applied Mathematics
1999
This thesis for the Master of Science
degree by
Adam Speight
has been approved
by

[Signature]
Stephen C. Billups

[Signature]
Harvey J. Greenberg

Gary A. Kochenberger

July 30, 1979
Date
Speight, Adam (M.S., Applied Mathematics)
Using Homotopy Methods to Solve Nonsmooth Equations
Thesis directed by Professor Stephen C. Billups

ABSTRACT

This thesis presents a new method for solving nonsmooth systems of equations, which is based on probability one homotopy techniques for solving smooth equations. The method is based on a new class of homotopy mappings on locally Lipschitz continuous operators that employ smoothing techniques to ensure the homotopy map is smooth on its natural domain. The new homotopy method is embedded in a hybrid algorithm. This is done so it may exploit the strong global convergence properties of homotopy methods, while relying on a Newton method for local converge to avoid potential numerical problems associated with nonsmoothness nearby a solution. Several theoretical results regarding the convergence behavior of this class of homotopy methods are presented.

The hybrid algorithm is applied to solve problems from a class of nonsmooth equations arising from a particular reformulation of the Mixed Complementarity Problem. This paper presents computational results from experimentation with the algorithm as well as issues related to the algorithm’s implementation.
In addition to presenting the hybrid algorithm, this thesis develops a suite of software routines in the MATLAB environment that allows developers to quickly create and evaluate homotopy-based solvers for nonsmooth equations. This software provides routines to help reformulate Mixed Complementarity Problems into nonsmooth equations, and an interface by which users may solve complementarity problems specified in the GAMS modeling language. It also includes interfaces to several sophisticated routines from HOMPACK90, a well-established collection of FORTRAN90 codes for implementing homotopy methods.

This abstract accurately represents the content of the candidate's thesis. I recommend its publication.

Signed _____________________________

Stephen C. Billups
ACKNOWLEDGMENTS

My most sincere thanks go to my advisor, Stephen Billups. For several years, he has been a constant source of advice, guidance, friendship, and inspiration, without which this thesis would never have been produced.

I would also like to offer my heartfelt thanks to Gary Kochenberger for serving on my thesis committee and Harvey Greenberg for the same and for providing many valuable comments, which greatly improved the quality of this thesis. Thanks also goes to Layne Watson for his valuable suggestions.

During the course of my education, several teachers have infused me with a drive to become more than I am and helped me acquire the means to do so. To these individuals, I am eternally grateful. In particular, would like to thank James Smith for helping me to discover a larger world, Allen Holder for acquainting me with it, and David Fischer, Burton Simon, and Stephen Medema for giving me the grand tour.

Thanks go to the faculty, staff, and students of the mathematics department at the University of Colorado at Denver who have provided me a supportive and pleasant environment from which to develop. Thanks also go to the Viotti family for helping me to keep perspective on what is really important in life.

Finally, I wish to thank my Family for years of support and prayers. Without them I surely would not enjoy the richness of life that I do today.
CONTENTS

Figures .......................................................... viii
Tables .......................................................... ix
1. Introduction ............................................... 1
2. Background Material .................................. 4
  2.1 Notation ............................................... 4
  2.2 Applications of Nonsmooth Equations ............ 5
    2.2.1 Inequality Feasibility Problem ............... 6
    2.2.2 Nonlinear Complementarity Problem ....... 6
    2.2.3 Variational Inequalities ...................... 7
  2.3 The Mixed Complementarity Problem ............... 8
    2.3.1 Walrasian Equilibrium ......................... 8
  2.4 MCP Reformulation .................................. 10
  2.5 Newton's Method ................................... 12
    2.5.1 Nonsmooth Newton Methods .................. 15
  2.6 Homotopy Methods .................................. 17
    2.6.1 Theory and Definitions ....................... 18
    2.6.2 Homotopy-based Algorithms .................. 22
  2.7 Dealing with nonsmoothness ....................... 28
    2.7.1 Complementarity Smoothers .................. 29
  3. The Algorithm ......................................... 33
## Contents

3.1 The Algorithm ........................................... 34  
3.2 Properties of $\rho_c$ ....................................... 35  
3.3 A Hybrid Algorithm ....................................... 38  
3.4 Solving Mixed Complementarity Problems ................. 40  
4. Implementation ............................................ 42  
4.1 HOMTOOLS .................................................. 42  
4.1.1 The Root Finding Package. ............................ 44  
4.1.2 The Utility Package. .................................. 49  
4.1.3 The Homotopy Smooth Package. ......................... 50  
4.1.4 The MCP Smooth Package. ............................. 51  
4.1.5 The GAMS MCP Package. .............................. 53  
4.1.6 The Algorithms Package. ............................... 54  
4.2 Solver Implementation .................................... 55  
5. Results and Conclusions .................................. 57  
5.1 Testing Procedure ...................................... 57  
5.2 Computational Results .................................. 61  
5.3 Conclusions .............................................. 61  
5.4 Avenues for Further Research ............................ 62  
References .................................................... 64
FIGURES

Figure

2.1 A nonsmooth Newton’s method with line search .......................... 14
2.2 A smoother for $\max(x, 0)$ ....................................................... 30
3.1 A hybrid algorithm ................................................................. 39
3.2 Procedure to evaluate an element of $\partial_B H(x)$ .......................... 41
TABLES

Table

4.1 Parameters for a HOMTOOLS compliant solver ............... 45
4.2 Overview of the HOMTOOLS public interface ............... 46
5.1 MCPLIB Test Problems ..................................... 58
5.2 MCPLIB Test Problems (continued) ......................... 59
5.3 MCPLIB Test Problems (continued) ......................... 60
1. Introduction

The problem of solving systems of equations is central to the field of applied mathematics. Many robust and efficient techniques exist to solve systems that are linear or differentiable. In recent years, however, there has been an increasing need for techniques to solve systems of equations that are nonsmooth, i.e., not continuously differentiable. Much progress has been made in generalizing Newton's method to solve such systems [16]. Techniques developed in this vein are attractive because they maintain the fast local convergence properties that are characteristic of Newton-based methods for smooth equations and perform with great efficacy when applied to nonsmooth problems that are nearly linear in the sense that their merit functions do not contain local minima that are not solutions. For problems that are highly nonlinear, however, these Newton-based schemes tend to fail or require a carefully chosen starting point to produce a solution. Finding such a point may require a great deal of human intervention.

To solve highly nonlinear systems of equations that are smooth, another class of algorithms known as homotopy methods may be employed, which often prove more robust than their Newton-based counterparts. To date very little progress has been made in extending homotopy methods into the domain of solving nonsmooth systems of equations. A principle objective of this thesis is to develop a class of techniques that extend conventional homotopy methods so they can be made to solve nonsmooth systems. A secondary objective of this thesis is to develop a set of software tools that allows for the rapid prototyping,
development, and testing of homotopy-based solvers for nonsmooth equations.

Chapter 2 presents some applications of nonsmooth systems of equations and describes a Newton-based method to solve them. It then presents some background material that will be necessary for the remainder of the thesis, including some theory on probability one homotopy methods for smooth equations, algorithms for implementing the same, and some theory about smoothing techniques for nonsmooth functions.

Chapter 3 describes our approach for using homotopy methods for solving nonsmooth equations. It presents some theoretical results, stating sufficient conditions under which the method should find a solution and discusses how the method can be incorporated into a hybrid algorithm, which makes use of a Newton-based method to achieve fast local convergence.

Chapter 4 details a suite of MATLAB software called HOMTOOLS. This package provides a uniform framework by which algorithm developers may write homotopy-based solvers for nonsmooth systems in MATLAB and other environments and test them against sets of problems coming from a variety of sources, including GAMS. It also makes several of the routines from the HOMPACK90 suite of codes available in the MATLAB environment. This package provides several sophisticated routines pertinent to homotopy-based algorithms that allow developers to leverage highly sophisticated routines and quickly write high-level algorithms. Chapter 4 also discusses how the homotopy method presented in Chapter 3 was implemented and integrated with HOMTOOLS.

Chapter 5 presents some computational results obtained from experimenting with the algorithm described in Chapters 3 and 4. It attempts to discern why the algorithm failed when it did and characterize the algorithm's
potential for usefulness in the future. This chapter concludes with a discussion on some future avenues for research.
2. Background Material

2.1 Notation

When discussing matrices, vectors and vector-valued functions, subscripts are used to indicate components, whereas superscripts are used to indicate the iteration number or some other label. For example, $A_i, A_j, A_{ij}$ refer to the $i$th row, $j$th column, and $(i,j)$th entry of a matrix $A$, respectively, whereas $x^k$ typically represents the $k$th iterate generated by an algorithm. In contrast to the above, for scalars or scalar-valued functions, we use subscripts to refer to labels so that superscripts can be reserved for exponentiation. The vector of all ones is represented by $e$.

Unless otherwise specified, $\| \cdot \|$ denotes the Euclidean norm. We use the notation $(\cdot)^+, (\cdot)^-$, and $|\cdot|$ to represent the plus, minus, and absolute value operators, respectively, for vectors. That is, $x_+ := (\max(x_1, 0); \ldots; \max(x_n, 0))$, $x_- := (\max(-x_1, 0); \ldots; \max(-x_n, 0))$, and $|x| := (|x_1|; \ldots; |x_n|)$.

The symbols $\mathbb{R}_+$ and $\mathbb{R}_{++}$ refer to the nonnegative real numbers and the positive real numbers respectively. The extended real numbers are denoted by $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$. The vectors $l$ and $u \in \overline{\mathbb{R}}^n$, specify a set of lower and upper bounds. Throughout this thesis we assume that $l < u$. The symbol $\mathcal{B}_{l,u}$ represents the box defined by $[l, u] := \{x \mid l \leq x \leq u\}$.

Real-valued functions are denoted with lower-case letters like $f$ or $\phi$ whereas vector-valued functions are represented by upper-case letters like $F$ or $\Phi$. For a function $F : C \subset \mathbb{R}^n \to \mathbb{R}^m$, we define $\nabla_i F_j(x) := \partial F_j(x)/\partial x_i$. $\nabla F(x)$ is the $n \times m$ matrix whose $ij$th element is $\nabla_i F_j(x)$. Thus, if $f$ is a
scalar valued function, then $\nabla f(x)$ is a row vector.

For a set $C \subset \mathbb{R}^n$ we denote its closure and boundary by $\overline{C}$ and $\partial C$ respectively. The projection operator for the set $C$ is denoted by $\pi_C(\cdot)$. That is, $\pi_C(x)$ represents the projection (with respect to the Euclidean norm) of $x$ onto the set $C$.

We will use the notation $o(\cdot)$ as follows: given a function $H : \mathbb{R}^n \to \mathbb{R}^m$, $o(H(x))$ represents any function $G : \mathbb{R}^n \to \mathbb{R}^m$ satisfying

$$\lim_{\|x\| \to 0} \frac{\|G(x)\|}{\|H(x)\|} = 0.$$ 

A sequence is said to converge \textit{quadratically} if

$$\limsup_{k \to \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} < \infty$$

and \textit{superlinearly} if

$$\limsup_{k \to \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} = 0.$$ 

We say that two sets, $A$ and $B$, are \textit{diffeomorphic} if there is a continuously differentiable bijection $\beta : A \to B$ such that $\beta^{-1}$ is also continuously differentiable. The term 'almost everywhere' will be used to modify to properties indicating that they hold on all elements of a set except on a subset having zero Lesbegue measure.

\textbf{2.2 Applications of Nonsmooth Equations}

The need to solve nonlinear equations arises frequently in the field of mathematical and equilibrium programming as well elsewhere in applied mathematics and in several engineering disciplines. In [16] a number of applications for nonsmooth equations are presented; some of which are described in this section.
2.2.1 Inequality Feasibility Problem  Finding solutions to systems of inequalities is an important and often difficult task in applied mathematics. Suppose $F : \mathbb{R}^n \to \mathbb{R}^n$ is a locally Lipschitzian function and $K$ is a polyhedral region in $\mathbb{R}^n$. The inequality feasibility problem is to find an $x \in K$ such that

$$F(x) \geq 0. \quad (2.1)$$

We can reformulate this into a system of equations by letting

$$H(x) = \min(0, F(x)),$$

where $\min(\cdot, \ldots, \cdot)$ is the componentwise minimum function of a finite number of vectors. It is clear that $H$ is a nonsmooth operator and that $x^*$ solves (2.1) if, and only if, $H(x^*) = 0$.

2.2.2 Nonlinear Complementarity Problem  The nonlinear complementarity problem (NCP) is a tool commonly used to model equilibrium problems arising in economics and game theory. It also serves as a generalized framework for many of the problems faced in mathematical programming. Given a continuously differentiable function $F : \mathbb{R}^n \to \mathbb{R}^n$, the problem NCP($F$) is to find some $x \in \mathbb{R}^n$ so that

$$0 \leq x \perp F(x) \geq 0 \quad (2.2)$$

where $x \perp F(x)$ means that $x^T F(x) = 0$.

We may take several approaches to reformulate the problem NCP($F$) into a nonsmooth system of equations. Define the functions

$$H^m(x) = \min(x, F(x)), \quad (2.3)$$

$$H^N(x) = F(x^+) - x^-, \quad \text{and} \quad (2.4)$$
\[ H^{FB}(x) = \left( x_i + F_i(x) - \sqrt{x_i^2 + F_i(x)^2} \right) \]  \hfill (2.5)

where the \( x_i^+ := \max(x_i, 0) \) and \( x_i^- := -\min(x_i, 0) \). One may easily verify that the following are equivalent:

(1) \( x^* \) solves (2.2),

(2) \( H^m(x^*) = 0 \),

(3) \( H^N(z) = 0 \) where \( x^* = z^+ \), and

(4) \( H^{FB}(x^*) = 0 \).

2.2.3 Variational Inequalities

Finite dimensional variational inequalities provide a framework from which to generalize the nonlinear complementarity problem as well as other problems in mathematical programming. A comprehensive treatment of the variational inequality is available in [12]. Suppose \( K \) is some closed convex subset of \( \mathbb{R}^n \) and \( F : D \to \mathbb{R}^n \) is a differentiable function over some open domain \( D \subset \mathbb{R}^n \) containing \( K \). The problem \( VI(K, F) \) is to find some \( x^* \in \mathbb{R}^n \) so that

\[
(y - x^*)^T F(x^*) \geq 0 \quad \text{for every } y \in K. \tag{2.6}
\]

To reformulate \( VI(F, K) \) into a system of equations we may define the following functions in terms of the Euclidean projection operator \( \pi \), which induces the nonsmoothness in the reformulations:

\[
H^m(x) = x - \pi_K(x - F(x)), \quad \text{and} \quad \tag{2.7}
\]

\[
H^N(x) = F(\pi_K(x)) + (x - \pi_K(x)). \tag{2.8}
\]

The notation for these functions is identical to those defined in (2.3) and (2.4) because if \( K = \mathbb{R}^n_+ \) the function \( H^m \) in (2.3) is equivalent to the function \( H^m \) in (2.7) and the function \( H^N \) in (2.4) is equivalent to \( H^N \) in (2.8). The function \( H^N \) is sometimes referred to as the normal map. That solving these equations is equivalent to solving \( VI(F, K) \) is shown in [12, Chapter 4].
2.3 The Mixed Complementarity Problem

One prominent class of problems we are interested in solving is the Mixed Complementarity Problem (MCP). Some prefer to refer to the MCP as the box-constrained variational inequality. This class of problems generalizes the NCP (Section 2.2.2) in that any component of the solution may be bounded above, bounded below, both, or unbounded altogether. We can reformulate the MCP into a system of nonsmooth equations in a fashion similar to the NCP reformulations. This thesis will pay particular attention to applying homotopy-based algorithms to a particular reformulation of this problem.

Given a rectangular region $B_{l,u} := \prod_{i=1}^{n} [l_i, u_i]$ in $\mathbb{R} \cup \{\pm \infty\}^n$ defined by two vectors, $l$ and $u$ in $\mathbb{R}^n$ where $-\infty \leq l < u \leq \infty$, and a function $F : B_{l,u} \to \mathbb{R}^n$, the Mixed Complementarity Problem $MCP(F, B_{l,u})$ is to find an $x \in B_{l,u}$ such that for each $i \in \{1, \ldots, n\}$, either

1. $x_i = l_i$, and $F_i(x) \geq 0$, or
2. $F_i(x) = 0$, or
3. $x_i = u_i$, and $F_i(x) \leq 0$.

This is equivalent to the condition that the componentwise median function, $\text{mid}(x-l, x-u, F(x)) = 0$. Sometimes when these conditions are satisfied we write $F(x) \perp x$ and say that $x$ is complementary to $F(x)$. Typically, one assumes that the function $F$ above is continuously differentiable on some open set containing $B_{l,u}$.

2.3.1 Walrasian Equilibrium MCPs are useful for modeling systems that have many agents, each of whom is optimizing a function. These functions may be interdependent in that the choice variables of all the agents may be parameters in the objective function of each agent. These interdependent systems of nonlinear programs may be reformulated into an MCP by
viewing the combined Karush-Kuhn-Tucker conditions of each agent’s problem as one large system of inequalities and noticing that together these systems form one large MCP. This reformulation is useful for recasting and solving many models arising in game theory and economics.

Of particular interest to microeconomists is the topic of general economic equilibrium. General equilibrium models usually incorporate many consumers, each of whom are maximizing their own utility and a production mechanism usually consisting of firms who are profit maximizers. A Walrasian equilibrium is characterized by having a set of prices, production activities, and consumption activities such that no good is in positive excess demand [19, Chapter 17.2].

Suppose there are \( m \) consumers in an economy and they may choose to purchase from a set of \( n \) goods, which can be purchased in arbitrarily fine quantities. If each consumer has well-behaved preferences then \( m \) quasi-convex utility functions \( (u_i : \mathbb{R}^n \to \mathbb{R}, \ i = 1 \ldots m) \) exist describing the consumers’ preferences[19]. If each consumer has an initial endowment of goods \( E^i \in \mathbb{R}^n, \ i = 1 \ldots m \), and \( p \in \mathbb{R}^n \) is a given vector of prices, then the \( i \)th consumer faces the following problem:

\[
\begin{align*}
\text{maximize}_{x \geq 0} & \quad u_i(x) \\
\text{subject to} & \quad p \cdot x \leq p \cdot E^i.
\end{align*}
\]  

(2.9)

Solving each of these \( m \) nonlinear programs parametrically in terms of the price vector \( p \) gives rise to demand operators for each consumer, \( d^i(p) \), where \( d^i_j(p) \) is the amount of good \( j \) that consumer \( i \) will consume at price level \( p \) [19, Chapter 7].

For the production side of the model, suppose there are \( k \) production sectors. Each sector \( j \) has an associated production level \( y_j \) that produces some
amount (possibly negative) of each good. It is usual to assume that production is linear; that is the set of producible goods is a convex cone in $\mathbb{R}^n$ [15]. If vectors of output $y$ and $z$ can be produced, then so can any combination $\alpha y + \beta z$ where $\alpha$ and $\beta$ are nonnegative scalars. Because there are a finite number of production sectors, the production cone is also an unbounded polyhedron, so we may represent the production set by a technology matrix $A \in \mathbb{R}^{n \times k}$. The element $A_{ij}$ represents the amount of good $i$ resulting in a unit of activity in sector $j$.

A set of equilibrium conditions adapted from [18] follows.

No activity earns a positive profit:  
$$p^T A \leq 0$$  \hspace{1cm} (1)

No good is in excess demand:  
$$\sum_{i=1}^{m} d^i(p) - \sum_{i=1}^{m} E^i - Ay \leq 0$$  \hspace{1cm} (2)

Positive prices and production:  
$$p \geq 0, \quad y \geq 0$$  \hspace{1cm} (3)

No losses or positive profits:  
$$(p^T A) y = 0$$  \hspace{1cm} (4)

Goods in excess supply are free and all goods with positive price are sold:  
$$p^T (\sum_{i=1}^{m} E^i + Ay - \sum_{i=1}^{m} d^i(p)) = 0$$  \hspace{1cm} (5)

The equations (4) and (5) indicate that the inequality (1) is complementary to $y$ and the inequality (2) is complementary to $p$. The above system can be viewed as a Nonlinear Complementarity Problem because each component of $y$ and $p$ is bounded by 0 and infinity.

2.4 MCP Reformulation

This section describes a class of operators such that any root of a particular operator is a solution to an associated MCP and any solution to that MCP is a root of its operator. Classes of both smooth and nonsmooth operators exist with this property, each with its own advantages. This section
focuses on a class of nonsmooth reformulations of a generic MCP.

**Definition 2.4.1** A function \( \phi : \mathbb{R}^2 \to \mathbb{R} \) is called an NCP function if \( \phi(a, b) = 0 \) if and only if \( \min(a, b) = 0 \).

These functions are so named because they are useful in reformulating NCPs (see Section 2.2.2). Equation (2.3) describes one way to reformulate an NCP using the component-wise minimum function, each component of which is trivially an NCP function. Indeed, this reformulation would still hold if the \( H \) defined in (2.3) were instead defined by

\[
H_i(x) = \phi(x_i, f_i(x))
\]

where \( \phi \) is any NCP function.

**Definition 2.4.2** A function \( \psi : \mathbb{R} \cup \{-\infty\} \times \mathbb{R} \cup \{\infty\} \times \mathbb{R}^2 \to \mathbb{R} \) is called an MCP function associated with \( l \) and \( u \) if \( \psi_{l,u}(x, f) := \psi(l, u, x, f) = 0 \) if and only if \( \text{mid}(x - l, x - u, f) = 0 \).

MCP functions are generalized NCP functions. If the lower bound \( l \) is zero and upper bound \( u \) is infinite, then the MCP function \( \psi_{l,u}(x, f) \) is also an NCP function because

\[
0 = \text{mid}(x - l, x - u, f) = \text{mid}(x, -\infty, f) = \min(x, f)
\]

if, and only if, \( \psi_{l,u}(x, f) = 0 \).

One very popular NCP function is the Fischer-Burmeister function \([9, 10]\),

\[
\phi_{FB}(a, b) := a + b - \sqrt{a^2 + b^2}.
\]  

(2.10)

A simple check reveals that this is an NCP function. While \( \phi_{FB} \) is not differentiable at the origin, \( \phi_{FB}^2 \) has the desirable property of being continuously differentiable everywhere. This property makes the Fischer-Burmeister function well suited for use in globalization strategies for Newton-based methods.
The function $\phi_{FB}$ is also semismooth (see Definition 2.5.3). This fact will be useful when we solve systems of semismooth equations incorporating the Fischer-Burmeister function.

Using either the pairwise minimum function or the Fischer-Burmeister function as $\phi$, Billups [1] was able to construct the MCP function:

$$
\psi(l, u, x, f) := \phi(x - l, -\phi(u - x, -f)).
$$

(2.11)

Constructing a function $H : \mathbb{R}^n \rightarrow \mathbb{R}^n$ from a box $\mathcal{B}_{l,u}$ defined by vectors $l$ and $u$ and a $C^2$ operator $F$,

$$
H_i(x) := \psi(l_i, u_i, x_i, F_i(x))
$$

(2.12)

yields a reformulation of the MCP$(F, \mathcal{B}_{l,u})$. $H(x) = 0$ if and only if $x$ is a solution to MCP$(\mathcal{B}_{l,u}, F)[1]$. Solving the nonsmooth equation $H(x) = 0$ will be the focus of much attention in later chapters.

2.5 Newton's Method

Newton's method is a standard technique for solving the equation

$$
F(x) = 0.
$$

(2.13)

If $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a continuously differentiable, or smooth, operator and $x^0$ is a starting point in $\mathbb{R}^n$, the sequence of iterates

$$
x^{k+1} := x^k - [\nabla F(x^k)]^{-1} F(x^k), \quad k = 0, 1, \ldots
$$

(2.14)

is well-defined and converges quadratically under conditions summarized in the following theorem.

Theorem 2.5.1 [7]. Let $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be continuously differentiable in an open convex set $D \subset \mathbb{R}^n$. Assume there is some $x^* \in \mathbb{R}^n$ and $r, \beta > 0$, so
that $\mathcal{B}_r(x^*) \subset D, F(x^*) = 0, \nabla F(x^*)^{-1}$ exists with $\|\nabla F(x^*)^{-1}\| \leq \beta$ and $\nabla F$ is Lipschitz continuous with constant $\gamma$ on $\mathcal{B}_r(x^*)$. Then, there exists some $\epsilon > 0$ such that for every $x^0 \in \mathcal{B}_r(x^*)$ the sequence generated by (2.14) is well defined, converges to $x^*$, and obeys

$$\|x^{k+1} - x^*\| \leq \gamma \beta \|x^k - x^*\|^2 \quad k = 0, 1, \ldots.$$  

This generic version of Newton’s method may perform poorly if $\nabla F$ is nearly singular at a solution or may fail altogether if it is started from a point that is too far from a solution. To improve the domain of convergence of the algorithm, a line search strategy can be added.

This is accomplished by defining the iterates as

$$x^{k+1} = x^k + \alpha_k d^k,$$

where $d^k$ solves

$$\nabla F(x^k)d^k = -F(x^k) \quad (2.15)$$

and $\alpha_k$ is chosen to ensure sufficient descent of the merit function

$$\theta(x) := \frac{1}{2} \|F(x)\|^2. \quad (2.16)$$

One popular criterion for choosing $\alpha_k$ is the Armijo rule; choose some $\alpha, \sigma \in (0, 1)$ and let $\alpha_k = \alpha^m$ where $m$ is the smallest integer such that

$$\theta(x^k + \alpha^m d^k) \leq \theta(x^k) - 2\sigma \alpha^m \theta(x^k).$$

This rule is used in the algorithm presented in Figure 2.1. More sophisticated line search techniques are presented in [14].
Figure 2.1. A nonsmooth Newton’s method with line search

Step 1 [Initialization] Choose line search parameters $\alpha$ and $\sigma$ in
$(0, 1)$, an initial point $x_0$ in $\mathbb{R}^n$ and a stopping tolerance $\epsilon$. Set
the iteration index $k = 0$.

Step 2 [Direction Selection] Pick some $V^k \in \partial_B F(x^k)$. If $V^k$ is
singular return a failure message along with the last iterate $x^k$
and stop. Otherwise let $d^k$ be the unique solution to

$$V^k d^k = -F(x^k). \tag{2.17}$$

Step 3 [Determine Step Length] Let $m_k$ be the smallest nonnegative integer $m$ so that

$$\theta(x^k + \alpha^m d^k) - \theta(x^k) \leq -2\sigma \alpha^m \theta(x^k).$$

Set $x^{k+1} = x^k + \alpha^m k$.

Step 4 [Check for Termination] If $\theta(x^k) < \epsilon$ then stop, returning
the solution $x^{k+1}$. Otherwise increment $k$ by one and return
to Step 2.
2.5.1 Nonsmooth Newton Methods  Developing versions of Newton’s method that require weaker assumptions regarding the differentiability of $F$ has been the focus of much research in recent years. Much of this research attempts to assume only the property of semismoothness about $F$ and employs the notion of a generalized Jacobian or subdifferential whenever $F$ is not differentiable.

**Definition 2.5.2** Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be a locally Lipschitzian function. By Rademacher’s Theorem, $F$ is differentiable except on a set of Lebesgue measure zero. Let $D_F$ be the set on which $F$ is differentiable. Define the $B$-subdifferential of $F$ by

$$\partial_B F(x) := \left\{ V \left| \exists \{x^k\} \to x, x^k \in D_F, V = \lim_{k \to \infty} \nabla F(x^k) \right. \right\}.$$ 

The Clarke subdifferential of $F$ is the convex hull of $\partial_B F(x)$ and is denoted as $\partial F(x)$.

It is clear that if $F$ is differentiable at a point $x$, then either characterizations of the subdifferential is simply the singleton set containing only $\nabla F(x)$. In this regard, the subdifferential generalizes the gradient.

**Definition 2.5.3** A function $F : \mathbb{R}^n \to \mathbb{R}^m$ is called semismooth at $x$ if

$$\lim_{V \in \partial F(x + th)} \{ Vh' \}$$

exists for every $h$ in \( \mathbb{R}^n \).

The apparent awkwardness of the definition of semismoothness is due to the fact that much of nonsmooth analysis requires a condition weaker than differentiability to be useful and stronger than local Lipschitz continuity to be powerful. Semismoothness is such a property. In [17], Qi presents some alternate characterizations of semismoothness. The most intuitive interpretation,
however, is that semismoothness is equivalent to the uniform convergence of directional derivatives in all directions. It is also worth noting that the class of semismooth functions is closed under composition.

In [17], a version of Newton’s method based on the Clarke subdifferential was presented. To find a solution to the equation $F(x) = 0$, starting from a point $x^0$, the iterative map

$$x^{k+1} := x^k - [V^k]^{-1} F(x^k),$$

(2.18)

where $V^k \in \partial F(x^k)$, may be followed.

An algorithm presented in [1, Figure 1] is outlined in Figure 2.1. It presents an algorithm similar to the above Newton’s method but adds a line search strategy and uses a merit function $\theta$ (for example $\theta(x) := \frac{1}{2} \|F(x)\|^2$). It also differs in that $V^k$ is restricted to be in the B-subdifferential, whereas [17] uses the Clarke subdifferential. The convergence analysis for this particular method uses the concepts of semicontinuity and BD-regularity.

**Definition 2.5.4** Suppose that $F : \mathbb{R}^n \to \mathbb{R}^m$ is B-differentiable in a neighborhood of $x$. We say that the directional derivative $F'(\cdot; \cdot)$ is semicontinuous at $x$ if, for every $\epsilon > 0$, there exists a neighborhood $N$ of $x$ such that, for all $x + h \in N$,

$$\|F'(x + h; h) - F'(x; h)\| \leq \epsilon \|h\|.$$

We say that $F'(\cdot; \cdot)$ is semicontinuous of degree 2 at $x$ if there exist a constant $L$ and a neighborhood $N$ of $x$ such that, for all $x + h \in N$,

$$\|F'(x + h; h) - F'(x; h)\| \leq L \|h\|^{2}.$$

The following definition generalizes the notion of nonsingularity. In nonsmooth analysis, regularity assumptions on functions are highly analogous to the nonsingularity of Jacobians in the analysis of smooth operators.
Definition 2.5.5 A semismooth operator $F$ is called BD-regular at $x$ if every element of $\partial_B F(x)$ is nonsingular.

The local convergence results for the nonsmooth Newton’s method presented in Figure 2.1 are presented in the following theorem.

Theorem 2.5.6 [1] Suppose that $x^*$ is a solution to $F(x) = 0$, and that $F$ is semismooth and BD-regular at $x^*$. Then, the iteration method defined by $x^{k+1} = x^k + d^k$, where $d^k$ is given by (2.17) is well defined and converges to $x^*$ superlinearly in a neighborhood of $x^*$. In addition, if $F(x^k) \neq 0$ for all $k$, then

$$\lim_{k \to \infty} \frac{\|F(x^{k+1})\|}{\|F(x^k)\|} = 0.$$  

If, in addition, $F$ is directionally differentiable in a neighborhood of $x^*$ and $F''(\cdot; \cdot)$ is semicontinuous of degree 2 at $x^*$, then the convergence of the iterations is quadratic.

Newton-based methods perform well on problems whose merit functions do not have non-zero local minima. For problems with higher degrees of nonlinearity, Newton-based methods tend to stall by converging to a local minimum of their merit function, or otherwise fail by diverging altogether. The local convergence theory presented above guarantees that they will produce a solution within an arbitrary tolerance provided that they are started sufficiently near to a solution. For many highly nonlinear problems this is not practical, and a more robust, globally convergent method is desired.

2.6 Homotopy Methods

A homotopy is a topological construction representing a function that continuously deforms an element of a function space into another element of that space. Suppose we are trying to solve the smooth equation $F(x) = 0$
where $F : \mathbb{R}^n \to \mathbb{R}^n$. An example of a simple homotopy on $F$ is

$$\rho(a, \lambda, x) := \lambda F(x) + (1 - \lambda)G_a(x)$$

where $\lambda \in [0, 1]$ is called the homotopy parameter and $a$ is some fixed vector in $\mathbb{R}^n$. Here, $G_a$ is some trivial class of operators on $\mathbb{R}^n$. An example may be $G_a(x) = x - a$. It is clear that when $\lambda = 1$ we have that $\rho(a, 1, x) = F(x)$ and when $\lambda = 0$, $\rho(a, 0, x) = G_a(x)$.

Much progress has been made in using the theory of homotopy mappings to construct techniques for finding zeros of smooth functions. This section describes some theory of probability one homotopy methods and describes a standard algorithm to implement them. The term ‘probability one’ applies because under appropriate conditions, a path from the starting point, which is uniquely determined by the parameter $a$, to a solution will fail to exist only on a set of those $a$ having zero Lesbegue measure.

2.6.1 Theory and Definitions  Much of the theory from probability one homotopy methods can be derived from generalizations of a result in fixed-point theory called Sard’s Theorem. This class of theorems uses a notion of a regular value.

**Definition 2.6.1** Let $U \subset \mathbb{R}^n$ be open and $F : \mathbb{R}^n \to \mathbb{R}^m$ be a smooth function. We say that $y \in \mathbb{R}^m$ is a regular value of $F$ if

$$\text{Range } \nabla F(x) = \mathbb{R}^m, \quad \text{for all } x \in F^{-1}(y).$$

**Theorem 2.6.2 (Parameterized Sard’s Theorem) [5]** Let $V \subset \mathbb{R}^p$, $U \subset \mathbb{R}^m$ be open sets and let $Y : V \times U \to \mathbb{R}^p$ be of class $C^r$, where $r > \max\{0, m - p\}$. If $0 \in \mathbb{R}^p$ is a regular value of $Y$, then for almost every $a \in V$, 0 is a regular value of the function $Y_a$ defined by $Y_a(\cdot) := Y(a, \cdot)$. 

18
Corollary 2.6.3 [5] Assume the conditions of the previous theorem. If $m = p + 1$, then each component of $\Gamma^{-1}_a({0})$ is a smooth curve for almost every $a \in V$ with respect to $q$-dimensional Lebesgue measure.

The following proposition gives conditions under which a well-behaved zero curve will exist. It is similar to results presented in [20] and [5, Theorem 2.4]. The path $\gamma_a$ defined in the proposition ‘reaches a zero of $F$’ in the sense that it contains a sequence $\{(\lambda_k, x^k)\}$ that converges to $(1, \bar{x})$, where $\bar{x}$ is a zero of $F$.

Proposition 2.6.4 Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a Lipschitz continuous function and suppose there is a $C^2$ map

$$\rho : \mathbb{R}^m \times [0, 1) \times \mathbb{R}^n \to \mathbb{R}^n$$

such that

1. $\nabla \rho(a, \lambda, x)$ has rank $n$ on the set $\rho^{-1}(\{0\})$,
2. the equation $\rho_a(0, x) = 0$, where $\rho_a(\lambda, x) := \rho(a, \lambda, x)$, has a unique solution $x^a \in \mathbb{R}^n$ for every fixed $a \in \mathbb{R}^m$,
3. $\nabla_x \rho_a(0, x^a)$ has rank $n$ for every $a \in \mathbb{R}^m$,
4. $\rho$ is continuously extendable (in the sense of Buck [3]) to the domain $\mathbb{R}^m \times [0, 1] \times \mathbb{R}^n$, and $\rho_a(1, x) = F(x)$ for all $x \in \mathbb{R}^n$ and $a \in \mathbb{R}^m$, and
5. $\gamma_a$, the connected component of $\rho_a^{-1}(\{0\})$ containing $(0, x^a)$, is bounded for almost every $a \in \mathbb{R}^m$.

Then for almost every $a \in \mathbb{R}^m$ there is a zero curve $\gamma_a$ of $\rho_a$, along which $\nabla \rho_a$ has rank $n$, emanating from $(0, x^a)$ and reaching a zero $\bar{x}$ of $F$ at $\lambda = 1$.

Further, $\gamma_a$ does not intersect itself and is disjoint from any other zeros of $\rho_a$.

Also, if $\gamma_a$ reaches a point $(1, \bar{x})$ and $F$ is regular at $\bar{x}$, then $\gamma_a$ has finite arc length.
Proof: Assumption 1 and the Parameterized Sard's Theorem give that 0 is a regular value of $\rho_a$ for almost every $a \in \mathbb{R}^m$. By Corollary 2.6.3 each component of $(\rho_a^{-1}(0))^{-1}(\{0\})$ is a smooth curve and, as such, either diffeomorphic to a circle or an interval [5]. Because $\nabla_x \rho_a(0, x^a)$ has rank $n$, the Implicit Function Theorem gives $x$ in terms of $\lambda$ in a neighborhood of $(0, x^a)$. This implies that $\gamma_a$ is not diffeomorphic to a circle. Since $\rho_a$ is of class $C^2$, all limit points of $\gamma_a$ must lie in $\rho_a^{-1}(\{0\})$ (with $\rho_a$'s extended domain $\mathbb{R}^n \times [0, 1] \times \mathbb{R}^m$), so the only limit point of $\gamma_a$ in $\{0\} \times \mathbb{R}^n$ is $(0, x^a)$. Furthermore, $\gamma_a \cap ((0, 1) \times \mathbb{R}^n)$ is diffeomorphic to an interval and $(0, x^a)$ corresponds to one end of it. Since $\gamma_a$ is bounded, there is some bounded closed set $B \subset \mathbb{R}^n$ such that $\gamma_a \subset [0, 1] \times B$. By the Implicit Function Theorem, $\gamma_a$ cannot terminate (have an end point) in $(0, 1) \times \mathbb{R}^n$, and by compactness $\gamma_a$ has finite arc length in every compact subset of $[0, 1) \times B$. (For each point $z \in \gamma_a$, there is a neighborhood of $z$ in which $\gamma_a$ has finite arc length by the Implicit Function Theorem and the fact that 0 is a regular value of $\rho_a$. For $z \notin \gamma_a$, there is a neighborhood of $z$ disjoint from $\gamma_a$ since $\mathbb{R}^n$ is normal. These neighborhoods form an open covering of $[0, 1) \times B$, for which the finite subcovering of a compact subset $[0, 1 - \epsilon) \times B$ yields finite arc length for the portion of $\gamma_a$ in $[0, 1 - \epsilon) \times B$.) Therefore $\gamma_a$ must leave every set $[0, 1 - \epsilon) \times B$, $\epsilon > 0$, and have a limit point in $[1) \times B$. By continuity of $\rho_a$ if $(1, x^*)$ is a limit point of $\gamma_a$, then $x^*$ is a zero of $F$.

That $\gamma_a$ does not intersect itself or any other component of $\rho_a^{-1}(\{0\})$ follows from the Implicit Function Theorem and the full rank of $\nabla \rho_a(\lambda, x)$ on $\rho_a^{-1}(\{0\})$: for each $z \in \gamma_a$ there is a neighborhood $B(z)$ such that $B(z) \cap \gamma_a$ is diffeomorphic to an open interval.

If $(1, \bar{x})$ is a limit point of $\gamma_a$ and $F$ is regular at $\bar{x}$, then every element of $\pi_x \partial \rho_a(\lambda, x)$ is nonsingular. By the Implicit Function Theorem for
nonsmooth functions (see [6][Corollary, Page 256]), \( \gamma_a \) can be described as a Lipschitz continuous function of \( \lambda \) in a neighborhood of \((1, \bar{x})\), and therefore has finite arc length in this neighborhood. Further, the zero curve has finite arc length outside this neighborhood by the above argument, and hence finite arc length everywhere. \( \blacksquare \)

A function such as \( \rho \) is called a homotopy mapping for \( F \). The zero curve \( \gamma_a \) described above can be characterized as the connected component of the set \( \rho_a^{-1}(\{0\}) \) containing the point \((0, x^a)\). Because the path \( \gamma_a \) is a smooth curve, it can be parameterized by its arc length away from \((0, x^a)\). This yields a function \( \gamma_a(s) \), the point on \( \gamma_a \) of arc length \( s \) away from \((0, x^0)\).

Given a homotopy mapping \( \rho \), we may construct a globally convergent algorithm as follows. Choose an arbitrary \( a \in \mathbb{R}^m \). This determines a unique starting point \( x^0 \). With probability one, we may then track the zero curve \( \gamma_a \) to a point \((1, \bar{x})\), where \( \bar{x} \) is a solution.

A simple and particularly useful homotopy mapping is \( \rho : \mathbb{R}^m \times [0,1) \times \mathbb{R}^n \to \mathbb{R}^n \) given by

\[
\rho(a, \lambda, x) := \lambda F(x) + (1 - \lambda)(x - a) .
\]  

If \( F \) is a \( C^2 \) operator then \( \rho \) satisfies properties (1), (2), (3), and (4) but not necessarily (5) of Proposition 2.6.4. Properties (2), (3), and (4) are satisfied trivially. That \( \rho \) satisfies property (1) can be seen in [21]. The following theorem gives conditions on \( F \) under which the fifth condition is satisfied.

**Theorem 2.6.5** [21] Let \( F : \mathbb{R}^n \to \mathbb{R}^n \) be a \( C^2 \) function such that for some \( \bar{x} \in \mathbb{R}^n \) and \( r > 0 \),

\[
(x - \bar{x})^T F(x) \geq 0 \text{ whenever } \|x - \bar{x}\| = r .
\]  

21
Then $F$ has a zero in a closed ball of radius $r$ about $\bar{x}$, and for almost every $a$ in the interior of this ball there is a zero curve $\gamma_a$ of

$$\rho_a(\lambda, x) := \lambda F(x) + (1 - \lambda)(x - a),$$

along which $\nabla \rho_a(\lambda, x)$ has full rank, emanating from $(0, a)$ and reaching a zero $\bar{x}$ of $F$ at $\lambda = 1$. Further, $\gamma_a$ has finite arc length if $\nabla F(\bar{x})$ is nonsingular.

The actual statement of the theorem in [21] fixes $\bar{x} = 0$. However, the proof can be modified trivially to yield the more general statement above.

For convenience, this thesis shall refer to (2.20) as the *global monotonicity* property as it is similar to the definition of monotonicity but ignores local behavior. If a $C^2$ operator $F$ possesses this property, these theoretical results have some profound implications. We are guaranteed the existence of a path of finite arc length between almost any starting point and a solution to $F(x) = 0$. In theory, to find a solution, one must simply follow the path from start to a limit point of $\gamma_a$, where $\lambda = 1$. In practice, however, the task of constructing an algorithm that can reliably track all types of paths is very difficult.

### 2.6.2 Homotopy-based Algorithms

Many of the homotopy-based algorithms for solving smooth systems of equations assume the conditions about rank, smoothness, and global monotonicity presented in Proposition 2.6.4 and Theorem 2.6.5. These assumptions guarantee the existence of a well-behaved zero curve, allowing developers of algorithms to focus on methods for tracking this curve from beginning to a solution.

Many packages exist to solve root finding problems using homotopy-based techniques [23]. This work will make use of the FIXPNF routine from the HOMPACK90 suite of software [22] [23, Section 3], which tracks the zero curve of a homotopy mapping specified by the user. This section describes the
Then $F$ has a zero in a closed ball of radius $r$ about $\tilde{x}$, and for almost every $a$ in the interior of this ball there is a zero curve $\gamma_a$ of

$$\rho_a(\lambda, x) := \lambda F(x) + (1 - \lambda)(x - a),$$

along which $\nabla \rho_a(\lambda, x)$ has full rank, emanating from $(0, a)$ and reaching a zero $\tilde{x}$ of $F$ at $\lambda = 1$. Further, $\gamma_a$ has finite arc length if $\nabla F(\tilde{x})$ is nonsingular. The actual statement of the theorem in [21] fixes $\tilde{x} = 0$. However, the proof can be modified trivially to yield the more general statement above.

For convenience, this thesis shall refer to (2.20) as the global monotonicity property as it is similar to the definition of monotonicity but ignores local behavior. If a $C^2$ operator $F$ possesses this property, these theoretical results have some profound implications. We are guaranteed the existence of a path of finite arc length between almost any starting point and a solution to $F(x) = 0$. In theory, to find a solution, one must simply follow the path from start to a limit point of $\gamma_a$, where $\lambda = 1$. In practice, however, the task of constructing an algorithm that can reliably track all types of paths is very difficult.

### 2.6.2 Homotopy-based Algorithms

Many of the homotopy-based algorithms for solving smooth systems of equations assume the conditions about rank, smoothness, and global monotonicity presented in Proposition 2.6.4 and Theorem 2.6.5. These assumptions guarantee the existence of a well-behaved zero curve, allowing developers of algorithms to focus on methods for tracking this curve from beginning to a solution.

Many packages exist to solve root finding problems using homotopy-based techniques [23]. This work will make use of the FIXPNF routine from the HOMPACK90 suite of software [22] [23, Section 3], which tracks the zero curve of a homotopy mapping specified by the user. This section describes the
FIXPNF algorithm and interface in great deal because the algorithm and its parameters will be referred to later.

FIXPNF takes the following input parameters:

- **N** - the dimension of the problem’s domain.
- **Y** - an array of size N+1 and contains the point at which to start tracking the curve. Y(1) is assumed to be zero and represents the λ component and Y(2:N+1) represents the x components.
- **IFLAG** - sets the mode of FIXPNF to finding fixed points or roots. It should be -2 for the zero finding problem with a user-specified homotopy.
- **A** - the parameter a in Proposition 2.6.4.
- **ARCRE, ARCAE, ANSRE, ANSAE** - real valued parameters corresponding to the absolute and relative errors in the curve tracking and the answer tolerances. These are discussed in more detail later.
- **SSPAR** - a vector of eight real numbers. They allow the user to have a high level of control over the stepsize estimation process. They are discussed in detail later.

FIXPNF provides the following output variables:

- **Y** - an array of size N+1. Y(2:N+1) contains the solution.
- **NFE** - number of function evaluations performed during the routine’s execution.
- **ARCLEN** - an approximation of the arc length of the zero curve that was traversed.
- **IFLAG** - parameter indicating the error status of the routine. A value of 1 is normal.

The core of the FIXPNF routine uses an algorithm consisting of the
following phases: prediction, correction, and stepsize estimation. The curve tracking algorithm generates a sequence of iterates \((\lambda_k, x^k)\) along the zero curve \(\gamma_a\) beginning with \((0, x^0)\), where \(x^0\) is determined by item (2) of Proposition 2.6.4 and converging to a point \((1, x^*)\) where \(x^*\) solves \(F(x) = 0\). Since \(\gamma_a\) can be parameterized by its arc length from its starting point by differentiable functions \((\lambda(s), x(s))\), each iterate \((\lambda_k, x^k)\) has an associated \(s_k\) such that \((\lambda_k, x^k)\) is \(s_k\) units away from \((0, x^0)\) along \(\gamma_a\).

For the general iteration in the prediction phase suppose that we have generated the two most recent points

\[ P^1 = (\lambda(s_1), x(s_1)) \quad \text{and} \quad P^2 = (\lambda(s_2), x(s_2)) \]

along \(\gamma_a\) and associated tangent vectors \(\left(\frac{d\lambda}{ds}, \frac{dx}{ds}\right)\) \((s_1)\) and \(\left(\frac{d\lambda}{ds}, \frac{dx}{ds}\right)\) \((s_2)\). We have also determined some \(h > 0\) to be used as an optimal step size in arc length to take along \(\gamma_a\) from \((\lambda(s_2), x(s_2))\). Noting that because the function \((\lambda(s), x(s))\) is the point on \(\gamma_a\) that is \(s\) units of arclength away from \((0, x^0)\), it must be that

\[
\left\| \left(\frac{d\lambda}{ds}, \frac{dx}{ds}\right)\right\|_2 = 1 \quad (2.21)
\]

for any \(s\). This can be seen because for \(t > 0\) we have that

\[ ||(\lambda(s + t), x(s + t)) - (\lambda(s), x(s))||_2 = t + o(t) \]

by the definition of \(\gamma_a(s)\). Dividing both sides by \(t\) and letting \(t \downarrow 0\) gives (2.21).

As a predictor for the next point on \(\gamma_a\), FIXPNF uses

\[ Z^0 := p(s_2 + h), \quad (2.22) \]

where \(p(s)\) is the unique Hermite cubic polynomial that interpolates \(P^1\) and
$P^2$. Formally,
\[
p(s_1) = (\lambda(s_1), x(s_1)), \quad \frac{dp}{ds}(s_1) = \left( \frac{d\lambda}{ds}, \frac{dx}{ds} \right)(s_1)
\]
\[
p(s_2) = (\lambda(s_2), x(s_2)), \quad \frac{dp}{ds}(s_2) = \left( \frac{d\lambda}{ds}, \frac{dx}{ds} \right)(s_2)
\]
and each component of $p(s)$ is a cubic polynomial.

The corrector phase is where the normal flow algorithm derives its name. As the vector $a$ varies, the paths $\gamma_a$ change. This generates a family of paths known as the Davidenko flow. The predicted point $Z^*$ will lie on a particular zero curve corresponding to some other path. The correction phase converges to a point on $\gamma_a$ along a path, which is normal to the Davidenko flow.

The corrector phase of FIXPNF uses a basic Newton-like method to converge from the predicted point to a point, $Z^0$, on $\gamma_a$. The standard Newton's method may not be used because $\nabla \rho(\lambda, x)$ is a rectangular matrix. Numerical issues related to efficiently computing these iterates are discussed in [22]. Beginning with $Z^0$ defined by (2.22) the corrector iteration is
\[
Z^{k+1} := Z^k - \left[ \nabla \rho_a(Z^k) \right]^\dagger \rho_a(Z^k),
\]
where $\left[ \nabla \rho_a(Z^k) \right]^\dagger$ denotes the Moore-Penrose pseudoinverse of the rectangular matrix $\nabla \rho_a(Z^k)$ [11, Chapter 5]. Because of the computational expense incurred by matrix factorizations in the Newton steps, the corrector phase reports failure if it has not converged to within the user specified tolerances in four iterations. The iterate $Z^k$ is assumed to have converged if
\[
\left\| Z^{k-1} - Z^k \right\| \leq ARCAE + ARCRE \left\| Z^k \right\|.
\]

The stepsize estimation phase of this algorithm is an elaborate process, which is very customizable via changes to values in the input parameter
array SSPAR. It attempts to carefully balance progress along $\gamma_{\alpha}$ with effort expended in the correction phase. This phase of the algorithm calculates an optimal step size $h_*$ for the next time through the algorithm. The SSPAR array contains eight floating point numbers,

$$(\bar{L}, \bar{R}, \bar{D}, h_{\min}, h_{\max}, B_{\min}, B_{\max}, q).$$

To find the optimal step size for the next iteration we define a *contraction factor*

$$L = \frac{\|Z^2 - Z^1\|}{\|Z^1 - Z^0\|},$$

*a residual factor*

$$R = \frac{\|\rho_\alpha(Z^1)\|}{\|\rho_\alpha(Z^0)\|}, \text{ and}$$

*a distance factor*

$$D = \frac{\|Z^1 - Z^4\|}{\|Z^0 - Z^4\|}.$$

The first three components of SSPAR represent ideal values for these quantities.

If we have that $h$ is the current step size, the goal is to achieve

$$\frac{\bar{L}}{L} \approx \frac{\bar{R}}{R} \approx \frac{\bar{D}}{D} \approx \frac{h^F}{h^*},$$

for some $q$. Define

$$\hat{h} = \left(\min\{\bar{L}/L, \bar{R}/R, \bar{D}/D\}\right)^{1/q} h$$

to be the smallest allowable choice for $h_*$. Now $h_*$ is chosen to be

$$h_* = \min\{\max\{h_{\min}, B_{\min}, \hat{h}\}, B_{\max} h, h_{\max}\}.$$

It remains to describe how the algorithm begins and ends. In the general prediction step, the algorithm uses a cubic predictor. This is possible because there are two points and corresponding tangent vectors from which
to interpolate. So to enter the general prediction phase we must generate two points on \( \gamma_a \) and find their associated tangent vectors. The first point is \((0, x^0)\) corresponding to the arc length \(s_0 = 0\). To generate the point \((\lambda_1, x^1)\) we make a linear approximation of \( \gamma_a \) from \((0, x^0)\), given an initial step size \(h\), by

\[
Z^0 = h \left( \frac{d\lambda}{ds}, \frac{dx}{ds} \right)(s_0) + (0, x^0).
\]

We may then use the correction phase to generate the point \((\lambda_1, x^1)\). If the correction phase fails, the stepsize is reduced by half and the process is repeated.

To evaluate \( \left( \frac{d\lambda}{ds}, \frac{dx}{ds} \right) \) at an \( s \) corresponding to a particular point on \( \gamma_a; (\lambda, x) \), we must solve the system

\[
\nabla \rho_a(\lambda, x)d = 0.
\]

If \( \nabla \rho_a(\lambda, x) \) has full rank, this equation determines a one dimensional subspace. However, by (2.21), we may take \( \|d\| = 1 \) so that \( d \) is only undetermined in its sign. For the initial point \((0, x^0)\) we may take the sign of \( d \) to be that which makes \( \frac{d\lambda}{ds}(0) > 0 \). For subsequent points we choose the sign of \( d \) so that \( d^T d^{last} > 0 \) where \( d^{last} \) was the direction generated on the previous prediction step. This heuristic prevents the path from bending too quickly. It is well justified because \((\lambda(s), x(s))\) is a differentiable function, which implies there will always be a stepsize small enough to ensure the acuteness of these two directions.

The FIXPNF routine determines it has reached an ‘end game’ state when it produces a pair of iterates \((\lambda_k, x^k)\) and \((\lambda_{k+1}, x^{k+1})\) such that

\[
\lambda_k \leq 1 \leq \lambda_{k+1}.
\]

The solution now lies on \( \gamma_a \) somewhere in between the last two generated points. The algorithm then builds Hermite cubic predictors interpolating \((\lambda_k, x^k)\) and
\((\lambda_{k+1}, x^{k+1})\) as above and attempts to generate progressively smaller brackets of points on \(\gamma_\alpha\) containing \( (1, x^*) \). The algorithm terminates when the end game either converges or fails.

2.7 Dealing with nonsmoothness

Since the functions considered in this paper are not of class \( C^2 \), we can not apply a homotopy method directly to them. Instead, these functions need to be approximated by a family of smooth functions.

Suppose we are interested in solving the system \( F(x) = 0 \) where \( F \) is a nonsmooth operator. Suppose we are given a family of functions \( F^\mu \) parameterized by a smoothing parameter \( \mu \), so that \( \lim_{\mu \to 0} F^\mu = F \) in some sense. We would like the solutions to the systems \( F^\mu(x) = 0 \) to converge to a solution to \( F(x) = 0 \) along a smooth trajectory. Under suitable conditions this can be achieved [4].

**Definition 2.7.1** Given a nonsmooth function \( \varphi : \mathbb{R}^p \to \mathbb{R} \), a smoother for \( \varphi \) is a function \( \tilde{\varphi} : \mathbb{R}^p \times \mathbb{R}_+ \to \mathbb{R} \) such that

1. \( \tilde{\varphi}(x, 0) = \varphi(x) \), and
2. \( \tilde{\varphi} \) is twice continuously differentiable with respect to \( x \) on the set \( \mathbb{R}^p \times \mathbb{R}_+ \).

For convenience, we shall use the notation \( \varphi^\mu(x) \) to denote the smoother \( \tilde{\varphi}(x, \mu) \).

**Example 2.7.2** As a simple example, consider the following function:

\[
\xi(x) := \max(x, 0).
\]

Since \( \xi \) is nonsmooth at zero, we define the following smoother, which is consistent with Definition 2.7.1:

\[
\xi^\mu(x) := \xi(x, \mu) := \frac{\sqrt{x^2 + \mu} + x}{2}.
\] (2.23)
As shown in Figure 2.2, $\xi^\mu(x)$ is smooth for all $\mu > 0$ and converges uniformly to $\xi(x)$ as $\mu \downarrow 0$.

To define smoothers for operators, we say that $F^\mu : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^n$ is a smoother for an operator $F$ if for each $i \in \{1 \ldots n\}$, $F_i^\mu$ is a smoother for $F_i$.

2.7.1 Complementarity Smoothers The Fischer-Burmeister function is useful for reformulating complementarity problems. For convenience, we define it again as follows:

$$\phi_{FB}(a, b) := a + b - \sqrt{a^2 + b^2}.$$  

The Fischer-Burmeister function is not differentiable at the origin, so a smoother will be necessary to use it in a homotopy method. Following is the Kanzow smoother for the Fisher-Burmeister function [13].

$$\phi^\mu(a, b) := \tilde{\phi}(a, b, \mu) := a + b - \sqrt{a^2 + b^2 + 2\mu} \tag{2.24}$$

Another function used in the reformulation of MCPs is (2.11)

$$\psi(l, u, x, f) := \phi(x - l, -\phi(u - x, -f))$$

whose associated smoother is called the Kanzow MCP smoother and defined as follows:

$$\psi^\mu(l, u, x, f) := \tilde{\psi}(l, u, x, f, \mu) := \phi^\mu(x - l, -\phi^\mu(u - x, -f)) \tag{2.25}.$$  

The following rather technical definitions are used to qualify NCP and MCP functions so that we can construct operators satisfying the global monotonicity property presented in (2.20).

**Definition 2.7.3** An NCP function $\phi$ is called positively oriented if

$$\text{sign}(\phi(a, b)) = \text{sign}(\min(a, b)).$$

29
Figure 2.2. A smoother for $\max(x, 0)$
An MCP function $\psi$ is called positively oriented if

$$\text{sign}(\psi_{l,u}(x, f)) = \text{sign}(\text{mid}(x - l, x - u, f)).$$

**Definition 2.7.4** A positively oriented MCP function $\psi$ is called median-bounded if there are positive constants $M$ and $m$ such that

$$m|\text{mid}(x - l, x - u, f)| \leq |\psi_{l,u}(x, f)| \leq M|\text{mid}(x - l, x - u, f)|.$$

Given the smoother $\psi^\mu$ it is possible to construct a smoother for $H$ in (2.12), the function that recasts MCP($F, B_{i,a}$) into a zero finding problem. The smoother for $H$ is

$$H_i^\mu(x) := \psi^\mu(l_i, u_i, x, F(x)).$$ \hspace{1cm} (2.26)

The following weak assumption about the smoothers in this paper will be useful.

**Assumption 2.7.5** There is a nondecreasing function $\xi : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ satisfying $\lim_{\mu \downarrow 0} \xi(\mu) = 0$ such that

$$|\tilde{\phi}(x, \mu) - \tilde{\phi}(x, 0)| \leq \xi(\mu)$$

for every $(x, \mu) \in \mathbb{R}^p \times \mathbb{R}_+$.

It will be important to note [1, Proposition 2.14] that the Kanzow MCP smoother (2.25) satisfies Assumption 2.7.5 with

$$\xi_K(\mu) := 3\sqrt{2}\mu.$$ \hspace{1cm} (2.27)

The following theorem asserts that, when all bounds are finite, $H^\mu$ has the property of global monotonicity discussed in Theorem 2.6.5 and, as such, is a nice candidate to be used in homotopy-based methods.
Theorem 2.7.6  [1] Let $\psi$ be a positively oriented median-bounded MCP function, and let $\tilde{\psi}$ be a smoother for $\psi$ satisfying Assumption 2.7.5. Suppose $B_{l,u}$ defined by vectors $l$ and $u$ is bounded, choose $\mu > 0$, and let $H^\mu : \mathbb{R}^n \to \mathbb{R}^n$ be defined by (2.26). Then, $H^\mu$ satisfies condition (2.20).
3. The Algorithm

This chapter presents a new homotopy-based hybrid algorithm for solving nonsmooth systems of equations. It contrasts a method presented by Billups, which is another hybrid Newton-Homotopy method[1]. Billups’ method begins by using a nonsmooth version of the damped-Newton’s method described in Figure 2.1 to solve the root finding problem $F(x) = 0$. If the Newton algorithm stalls, a standard homotopy method is invoked to solve a particular smoothed version of the original problem, $F^\mu(x) = 0$. The smoothing parameter $\mu$ a fixed value chosen based on the level of a merit function on $F$ at the last point $\hat{x}$ generated by the Newton method. Starting from $\hat{x}$, a homotopy method is carried out until it produces a point that yields a better merit value than the previous Newton iterate. The Newton method is then started from this new point and the process repeats until a point is produced that is close enough to a solution or the homotopy method fails. One key feature of this hybrid method is that each time the Newton method stalls, a different homotopy mapping is used. This class of maps is defined by

$$\rho^\mu_\hat{x}(\lambda, x) := \lambda F^\mu(x) + (1 - \lambda)(x - \hat{x}),$$

where $\hat{x}$ is the point at which Newton’s method last failed.

Computational experience with that algorithm indicates that starting the homotopy algorithm from a point at which Newton’s method failed often results in zero curves that are very difficult to track. In fact, picking a random starting point from which to start a homotopy algorithm usually produces better results. We therefore propose a different hybrid algorithm. This algorithm
again uses Newton's method, but chooses a single homotopy zero curve to follow for the duration of the algorithm. It attempts to follow this single curve into the domain of convergence of the Newton method.

3.1 The Algorithm

The basic idea of the new algorithm follows. Given a function $F$ and an associated smoother, $F^\nu$ consistent with Definition 2.7.1, construct a single homotopy mapping on $F$ whereby the smoothing parameter $\mu$ is a function of the homotopy parameter $\lambda$ so that $\mu \downarrow 0$ as $\lambda \uparrow 1$. If this homotopy satisfies the conditions in Proposition 2.6.4, a well behaved path exists from almost any starting point to a solution, and we can use standard curve tracking techniques to reliably solve the equation $F(x) = 0$.

Throughout this chapter we shall assume that $F$ is an operator on $\mathbb{R}^n$ and that $F^\nu$ is a smoother for $F$. We will take $\mu(\lambda)$ to be a nondecreasing differentiable function such that $\lim_{\lambda \to 1} \mu(\lambda) = 0$. For simplicity we assume

$$\mu(\lambda) := \alpha(1 - \lambda)^2$$  \hspace{1cm} (3.1)

for some fixed value of a parameter $\alpha > 0$, although one need not be so specific about the form of this function. We define a homotopy on $F$ in terms of its smoother $F^\nu$,

$$\varrho_\alpha(\lambda, x) := \lambda F^\mu(\lambda) + (1 - \lambda)(x - a)$$  \hspace{1cm} (3.2)

and define $\gamma_\alpha$ to be the connected component of the set $\varrho_\alpha^{-1}({0})$ that contains $(0,a)$.

Since the smoothing parameter $\mu(\lambda)$ converges to zero as $\lambda \uparrow 1$, the function $F^{\mu(\lambda)}$ may be nearly nonsmooth near $\lambda = 1$. By this we mean that the curvature at certain points in some components of $F^{\mu(\lambda)}$ may be very large. This behavior may result in the zero curve $\gamma_\alpha$ having severe turns as it
approaches a point where $\lambda = 1$. Such erratic turning poses great difficulties for standard curve tracking algorithms, which are designed to track zero curves induced by homotopies on smooth functions. In addition to the potential erratic behavior of the zero curve near $\lambda = 1$, there is the problem that the smoothers we are interested in are not defined for $\mu < 0$. The curve tracking algorithm in FIXPNF (see Section 2.6.2) actually tracks the zero curve until it finds a point where $\lambda > 1$. While this poses no difficulty for the smoothing parameter in (3.1), using a function like

$$\nu(\lambda) := \alpha(1 - \lambda)$$

for a smoothing parameter would yield disastrous results if a point were ever evaluated such that $\lambda > 1$.

Because of the issues presented above, we are interested in using an algorithm similar to the general phase of the FIXPNF routine to track the zero curve $\gamma_\alpha$ to a point somewhere short of $\lambda = 1$, and using a different ‘end game’ strategy to converge to a solution. This chapter will give conditions under which a point $(\lambda, x)$ on the zero curve with $\lambda$ sufficiently close to one, will guarantee that $x$ be arbitrarily close to a solution. This result will allow for the construction of a hybrid algorithm, which exploits the fast local convergence of Newton’s method for the ‘end game’ phase of the algorithm and maintains the global convergence properties of homotopy methods.

3.2 Properties of $\varrho_\alpha$

In order to ensure that $\gamma_\alpha$ is almost surely a well-behaved path that leads to a solution, we must state conditions on $F$ and its smoother so that Proposition 2.6.4 can be invoked. The following weak assumption on our smoother will be useful in the theory that follows.
Assumption 3.2.1 There is a nondecreasing function $\eta : \mathbb{R}_+ \to \mathbb{R}_+$ satisfying
\[ \lim_{\nu \to 0} \eta(\nu) = 0 \] such that for all $x$ in $\mathbb{R}^n$ and all $\nu$ in $\mathbb{R}_+$

\[ \| F^\nu(x) - F(x) \|_\infty \leq \eta(\nu). \]

Lemma 3.2.2 Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a Lipschitz continuous function such that
for some fixed $r > 0$ and $\tilde{x} \in \mathbb{R}^n$,

\[ (x - \tilde{x})^T F(x) \geq 0 \text{ whenever } \| x - \tilde{x} \| = r, \]

and such that $F^\mu$ satisfies Assumption 3.2.1. Further, suppose that the smoothing parameter $\mu(\lambda)$ is such that

\[ \eta(\mu(\lambda)) < \frac{1 - \lambda}{\lambda} M \text{ for } 0 < \lambda \leq 1 \] (3.3)

for some $M \in (0, r)$. Then $\gamma_a$ is bounded for almost every $a \in \mathbb{R}^n$ such that

$\| a - \tilde{x} \| < \tilde{r} := r - M$.

Proof: Consider any point $(\lambda, x)$ with $0 < \lambda < 1$, $\| x - \tilde{x} \| = r$, and let

$\| a - \tilde{x} \| < \tilde{r}$. Starting with

$\varrho_a(\lambda, x) = \lambda F^{\mu(\lambda)}(x) + (1 - \lambda)(x - a),$ 

multiplying by $x - \tilde{x}$ and dividing by $1 - \lambda$ results in

\[ \frac{\varrho_a(\lambda, x)^T (x - \tilde{x})}{1 - \lambda} = \frac{\lambda}{1 - \lambda} F^{\mu(\lambda)}(x)^T (x - \tilde{x}) + (x - a)^T (x - \tilde{x}). \] (3.4)

By assumption

\[ F^{\mu(\lambda)}(x)^T (x - \tilde{x}) = (F^{\mu(\lambda)}(x) - F(x))^T (x - \tilde{x}) + F(x)^T (x - \tilde{x}) \] (3.5)

\[ \geq - \| F^{\mu(\lambda)}(x) - F(x) \| \| x - \tilde{x} \| + 0 \]

\[ \geq - \eta(\mu(\lambda)) \| x - \tilde{x} \| \]

\[ \geq - \frac{1 - \lambda}{\lambda} M r. \]
Combining (3.4) and (3.5) gives
\[
\frac{g_a(\lambda, x)^T(x - \bar{x})}{1 - \lambda} \geq -\frac{\lambda}{1 - \lambda} M r + (x - a)^T(x - \bar{x}) \\
\geq -Mr + ((x - \bar{x}) + (\bar{x} - a))(x - \bar{x}) \\
> -Mr + r^2 - \bar{r} = 0.
\]
Therefore \( g_a \) is not zero on the set
\[
B := \{ (\lambda, x) | \| x - \bar{x} \| = r, \; \lambda \in [0, 1] \}.
\]
Since \((0, a) \in \gamma_a \) and \( \| a - \bar{x} \| < \bar{r} < r \), \( \gamma_a \) is bounded (being contained in the convex hull of \( B \)).

A direct application of Proposition 2.6.4 gives our main convergence result.

**Corollary 3.2.3** Under the assumptions of Lemma 3.2.2, \( F \) has a zero in a closed ball of radius \( r \) about \( \bar{x} \), and for almost every \( a \) in the interior of a ball of radius \( \bar{r} \) about the \( \bar{x} \), there is a zero curve \( \gamma_a \) of
\[
g(a, \lambda, x) := g_a(\lambda, x) := \lambda F^{\mu(\lambda)}(x) + (1 - \lambda)(x - a),
\]
along which \( \nabla g_a(\lambda, x) \) has full rank, emanating from \((0, a) \) and reaching a zero \( \bar{x} \) of \( F \) at \( \lambda = 1 \). Further, \( \gamma_a \) has finite arc length if \( F \) is regular at \( \bar{x} \).

**Proof:** The existence and behavior of the zero curve \( \gamma_a \) will follow from Proposition 2.6.4. \( g \) satisfies conditions (2)-(4) of this proposition trivially. It satisfies property (5) by Lemma 3.2.2. It only remains to show that \( \nabla g \) has rank \( n \) on \( g^{-1}(\{0\}) \). For this we observe that for \( \lambda < 1 \), \( \nabla g(\lambda, x, a) \) is given by
\[
\nabla g := \left[ (\lambda - 1)I \quad F^{\mu(\lambda)}(x) + \lambda \nabla_\lambda F^{\mu(\lambda)}(x) - x + a \quad \nabla_x F^{\mu(\lambda)}(x) + (1 - \lambda)I \right].
\]
Since the first \( n \) columns are linearly independent for \( \lambda < 1 \), \( \nabla g \) has rank \( n \) on \( g^{-1}(\{0\}) \).
Observe that in applications, the $r$ in Lemma 3.2.2 can be arbitrarily large, hence so can $\tilde{r} = r - M$, and thus $|a - \tilde{x}| < \tilde{r}$ is really no restriction at all.

We now establish conditions under which a homotopy mapping can generate a zero curve that may be followed arbitrarily close to a solution.

**Theorem 3.2.4** Suppose $F$ and $F^{\mu(\lambda)}$ satisfy the conditions in Lemma 3.2.2. Then for all $\varepsilon > 0$ there is some $\delta > 0$ such that whenever $(\lambda, x) \in \gamma_a$ with $\lambda \in (1 - \delta, 1)$ we have that

$$||x - x^*|| < \varepsilon$$

for some solution $x^*$.

**Proof:** Suppose towards a contradiction that for some $\varepsilon > 0$ there is no $\delta > 0$ such that for all $(\lambda, x) \in \gamma_a$ with $\lambda \in (1 - \delta, 1)$ we have $||x - x^*|| < \varepsilon$. Define the set

$$S_\varepsilon := \{x \mid ||x - x^*|| < \varepsilon \text{ for some solution } x^*\}.$$

By supposition, there is some sequence $\{(\lambda_k, x^k), \{x^k\}, \{x^{kj}\}\}$ such that $\lambda_k \uparrow 1$ but the tail of $x^k$ is not contained in $S_\varepsilon$, so there is some subsequence $x^{kj}$ that is disjoint from $S_\varepsilon$. Since $\gamma_a$ is bounded, $x^{kj}$ is bounded and must have a limit point $\bar{x}$ that is not contained in $S_\varepsilon$. Since $\rho_a$ is continuously extendable to a domain where $\lambda \in [0, 1]$, $\bar{x}$ must be a zero of $F$ and therefore must be in $S_\varepsilon$. But this contradicts the above assertion that $\bar{x} \notin S_\varepsilon$. \hfill \blacksquare

### 3.3 A Hybrid Algorithm

One way to use the previously described homotopy approach in a hybrid algorithm is to combine it with a nonsmooth version of Newton's method such as the one presented in Figure 2.1. In order to solve the system $F(x) = 0$, 

38
the nonsmooth Newton's method requires that $F$ be semismooth. If in addition $F$ is BD-regular at a solution $x^*$, the method will converge superlinearly in some neighborhood about $x^*$. To use the homotopy approach, $F$ should satisfy the global monotonicity property and should ideally be nonsingular at all solutions. This guarantees that the homotopy's zero curve crosses the hyperplane given by $\lambda = 1$ transversally rather than tangentially and ensures that the zero curve will have finite arc length.

**Figure 3.1.** A hybrid algorithm

- **Step 1** [Initialization] Pick parameters $\sigma$ and $\delta$ in $(0, 1)$ and a tolerance $\epsilon > 0$. Let $k = 0$, $\lambda_0 = 0$ and choose some $x^0 \in \mathbb{R}^n$.
- **Step 2** [Homotopy] Starting from the point $(\lambda_k, x_k)$, use a curve tracking algorithm to find a point $(\lambda_{k+1}, x_{k+1})$ in $\gamma_\delta$ such that $\lambda_{k+1} \in (1 - \delta, 1)$.
- **Step 3** [Local Convergence] Run $A(x^{k+1}, \epsilon)$. If a point $\tilde{x}$ is produced such that $\theta(\tilde{x}) < \epsilon$ report $\tilde{x}$ as a solution and terminate. Otherwise proceed to Step 4.
- **Step 4** [Adjustment] Let $\delta := \sigma \delta$, increment $k$ by one and go to Step 2.

We shall denote the nonsmooth Newton's method in Figure 2.1 as $A(x, \epsilon)$, which stops when it produces a point such that $\theta(x) < \epsilon$ or when it converges to a local minimum of $\theta$ that is not a solution. Figure 3.1 describes a hybrid algorithm, which will produce a solution provided that the assumptions in Theorem 3.2.4 are satisfied and the curve tracking algorithm is able to perform accurately. If the algorithm fails, it will do so in the homotopy step when it can no longer make progress along the zero curve.

One notable feature of this hybrid algorithm is that if the local convergence step (3) fails, the homotopy step (2) will resume starting from the
last point found on $\gamma_\alpha$. This contrasts the algorithm in [1] and is motivated by the fact that when Newton’s method stalls, it usually gets stuck in a local minimum of the merit function, which may cause the homotopy method to perform poorly.

3.4 Solving Mixed Complementarity Problems

When all bounds are finite, the reformulation of the Mixed Complementarity Problem into a zero finding problem by the nonsmooth operator $H$ presented in (2.12) and its smoother $H^\mu$ in (2.26) are particularly well suited for use with the algorithm in this chapter. If all bounds are finite on the MCP that defines $H$, all of the conditions for convergence are automatically satisfied except for (3.3), which can be met by choosing a smoother that closely approximates $H$. If in addition $H$ is BD-regular at all solutions, the Newton component of the hybrid algorithm will converge superlinearly.

To use this algorithm to solve the problem $\text{MCP}(F, B_{l,u})$, by solving the equation $H(x) = 0$, it only remains to show how to generate elements of the subdifferential $\partial_B H(x)$ to be used in the nonsmooth Newton’s method presented in Figure 2.1. Such a procedure is provided in Figure 3.2. In [1] it is proved that this procedure will produce an element of $\partial_B H(x)$. Since we are using this procedure only in Newton’s method to solve $H(x) = 0$, we may consider just the case when $\mu = 0$ in this procedure.
Figure 3.2. Procedure to evaluate an element of $\partial_p H(x)$

Step 1 Set $\beta_1 := \{ i \mid x_i - l_i = 0 = F_i(x) \}$ and $\beta_u := \{ i \mid u_i - x_i = 0 = F_i(x) \}$.

Step 2 Choose $z \in \mathbb{R}^n$ such that $z_i \neq 0$ for all $i \in \beta_1 \cup \beta_u$.

Step 3 For each $i$, if $i \notin \beta_u$, or $\mu \neq 0$, set

\[
    c_i(x) := \frac{x_i - u_i}{\sqrt{(x_i - u_i)^2 + F_i(x)^2 + 2\mu}} + 1
\]

\[
    d_i(x) := \frac{F_i(x)}{\sqrt{(x_i - u_i)^2 + F_i(x)^2 + 2\mu}} + 1;
\]

else if $\mu = 0$ and $i \in \beta_u$, set

\[
    c_i(x) := \frac{z_i}{\|(z_i, \nabla F_i(x)z)\|} + 1
\]

\[
    d_i(x) := \frac{\nabla F_i(x)z}{\|(z_i, \nabla F_i(x)z)\|} + 1.
\]

Step 4 For each $i$, if $i \notin \beta_1$ or $\mu \neq 0$, set

\[
    a_i(x) := 1 - \frac{x_i - l_i}{\sqrt{(x_i - l_i)^2 + \phi(u_i - x_i, -F_i(x))^2 + 2\mu}}
\]

\[
    b_i(x) := 1 - \frac{\phi(u_i - x_i, -F_i(x))}{\sqrt{(x_i - l_i)^2 + \phi(u_i - x_i, -F_i(x))^2 + 2\mu}};
\]

else if $\mu = 0$ and $i \in \beta_1$, set

\[
    a_i(x) := 1 - \frac{z_i}{\|(z_i, c_i(x)z_i + d_i(x)\nabla F_i(x)z)\|}
\]

\[
    b_i(x) := 1 - \frac{c_i(x)z_i + d_i(x)\nabla F_i(x)z}{\|(z_i, c_i(x)z_i + d_i(x)\nabla F_i(x)z)\|};
\]

Step 5 For each $i$, set

\[
    V_i := (a_i(x) + b_i(x)c_i(x))e^iT + b_i(x)d_i(x)\nabla F_i(x).
\]

41
4. Implementation

4.1 HOMTOOLS

The algorithm described in the previous chapter is but one of many possible approaches for using homotopy-based techniques to solve nonsmooth equations. Our ultimate interest lies in exploring variants of this algorithm. To aid in the development of such algorithms, this thesis develops a suite of MATLAB routines called HOMTOOLS, which allows algorithm developers to prototype and evaluate homotopy-based solvers in a MATLAB environment. HOMTOOLS provides a uniform interface by which clients can request a problem to be solved, and several convenience routines to aid in the construction of solvers. Some of these routines are MATLAB Mex interfaces into routines from HOMPACK90, which allow developers to build solvers from highly sophisticated atomic routines. For the client, HOMTOOLS provides an interface to the MCPLIB library of complementarity problems, which serves as a convenient test library for the solvers. Table 4.2 outlines the basic structure of the six packages of the HOMTOOLS interface. The prefixes on the functions have been omitted for ease of reading.

The Root Finding package provides clients a way to specify the homotopy map and solver to be used and provides a uniform interface through which users may invoke the specified solver. It provides solvers the ability to recover functions implementing the homotopy map.

The Homotopy Smooth package creates the homotopy map defined in (3.2) on a function and its smoother and initializes the Root Finding package.
The *Utility* package provides functions that can be used to reformulate Mixed Complementarity Problems as specified in Section 2.4.

The *MCP Smooth* package uses the *Utility* package to construct the function $H$ and its smoother $H^u$ described in Section 2.7.1 given an MCP defined by a function and upper and lower bound vectors. This package uses these functions to initialize the *Homotopy Smooth* package.

The *GAMS MCP* package interfaces with GAMS [8] to make available problems from MCPLIB, a large test library of complementarity problems defined in GAMS. It obtains handles to functions and bound vectors which are used to initialize the *MCP Smooth* package. This package allows problems from MCPLIB to be solved simply by specifying the model name and a starting point.

The *Algorithms* package contains convenience routines to help developers build solvers. Some of these routines are Mex interfaces into the HOMPACK90 suite of codes.

A more detailed description of HOMTOOLS with examples follows. In order to function as a homotopy-based solver for the equation

$$ F(x) = 0, \quad (4.1) $$

a routine must be provided with functions to evaluate a homotopy mapping and its Jacobian:

$$ \rho_a(\lambda, x) \quad \text{and} \quad (4.2) $$

$$ \nabla \rho_a(\lambda, x). \quad (4.3) $$

The functions $F(x)$ and $\nabla F(x)$ may be evaluated by taking $\rho_a(1, x)$ and $\nabla \rho_a(1, x)$ respectively, however for convenience and efficiency the solver should be able to evaluate these functions directly, especially if $F$ is nonsmooth. To
be as general as possible, the solver routine should be provided with an open set of parameters. We will assume that solvers used with HOMTOOLS have the following MATLAB function signature:

\[
[x \ njac \ arclen \ status] = \text{solverName}(n,x0,A,
\text{arcre,arcae,ansre,ansae,axnfe,sspar})
\]  

(4.4)

where the parameters are defined in Table 4.1.

The remainder of this section describes each package of HOMTOOLS in detail and presents some examples of how to use each package. These packages are presented so that each package directly depends and builds on the previous one excepting the *Utility* package, which is independent of the rest of HOMTOOLS and the *Algorithms* package, which only depends on the *Root Finding* package. One should also note that the packages in HOMTOOLS were designed so that they depend on one another through interface only. If any package’s implementation is unsatisfactory for a particular application, it may be replaced with a different package that offers the same function names, signatures, and semantics.

### 4.1.1 The *Root Finding* Package.

At the base of the HOMTOOLS suite is a package called *Root Finding*. This package is responsible for associating a user-specified solver with MATLAB functions implementing (4.2) and (4.3) and providing a uniform interface by which clients may solve equations. The core set of functions in the *Root Finding* package follow.

\[
H\_initRootFinding(fName,fjacName,rhoName,rhojacName)
\]  

(4.5)

This function takes as input the names of MATLAB routines that can be used to evaluate \( F, \rho_0 \), and their Jacobians. These arguments are passed in as single quoted strings so that they may later be invoked through a call to the `feval()`
Table 4.1. Parameters for a HOMTOOLS compliant solver

<table>
<thead>
<tr>
<th>Input:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>problem size</td>
</tr>
<tr>
<td>x0</td>
<td>starting point</td>
</tr>
<tr>
<td>A</td>
<td>parameter in (4.2)</td>
</tr>
<tr>
<td>arcre</td>
<td>relative error for curve tracking</td>
</tr>
<tr>
<td>arcae</td>
<td>absolute error for curve tracking</td>
</tr>
<tr>
<td>ansre</td>
<td>relative error for end game</td>
</tr>
<tr>
<td>ansae</td>
<td>absolute error for end game</td>
</tr>
<tr>
<td>maxnfe</td>
<td>maximum number of Jacobian evaluations to be performed</td>
</tr>
<tr>
<td>sspar</td>
<td>any MATLAB object - defaults to HOMPACK90 conventions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>alleged answer</td>
</tr>
<tr>
<td>njac</td>
<td>number of Jacobian evaluations performed</td>
</tr>
<tr>
<td>arclen</td>
<td>approximate length of the homotopy curve that was tracked</td>
</tr>
<tr>
<td>status</td>
<td>solver dependent error flag</td>
</tr>
<tr>
<td>Package</td>
<td>Function</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Root Finding</td>
<td>initRootFinding()</td>
</tr>
<tr>
<td></td>
<td>setImpl()</td>
</tr>
<tr>
<td></td>
<td>findRoot()</td>
</tr>
<tr>
<td></td>
<td>getDefaultArgs()</td>
</tr>
<tr>
<td></td>
<td>getArgs()</td>
</tr>
<tr>
<td></td>
<td>setArgs()</td>
</tr>
<tr>
<td></td>
<td>merit()</td>
</tr>
<tr>
<td></td>
<td>F() Fjac() rho() rhojac()</td>
</tr>
<tr>
<td>Homotopy Smooth</td>
<td>initHomotopySmooth()</td>
</tr>
<tr>
<td>util</td>
<td>mu() mujac()</td>
</tr>
<tr>
<td></td>
<td>phi() phijac()</td>
</tr>
<tr>
<td></td>
<td>psi() psijac()</td>
</tr>
<tr>
<td></td>
<td>Alpha()</td>
</tr>
<tr>
<td>MCP Smooth</td>
<td>initMCPSmooth()</td>
</tr>
<tr>
<td></td>
<td>solveMCP()</td>
</tr>
<tr>
<td></td>
<td>testSoin()</td>
</tr>
<tr>
<td>GAMS MCP</td>
<td>solveGamsMCP()</td>
</tr>
<tr>
<td>Algorithms</td>
<td>fixpnf() - from HOMPACK90</td>
</tr>
<tr>
<td></td>
<td>stepnf() - from HOMPACK90</td>
</tr>
</tbody>
</table>
function in MATLAB. The necessary signatures of these routines are the same as those in (4.6) – (4.9).

\[
y = Hrf.F(x) \tag{4.6}
\]
\[
dy = Hrf.Fjac(x) \tag{4.7}
\]
\[
z = Hrf.rho(lambda,x) \tag{4.8}
\]
\[
dz = Hrf.rhojac(lambda,x) \tag{4.9}
\]

In these routines, \(dy\) and \(dz\) are Jacobian matrices of dimensions \(n \times n\) and \(n \times (n + 1)\) respectively.

\[
Hrf.setImpl(solverName) \tag{4.10}
\]

This function takes as input the name of a MATLAB routine as a single quoted string. The function specified by \(\text{solverName}\) must have a calling signature identical to (4.4).

\[
[x, njac, arcLen, status] = Hrf.findRoot(n, x0, A,
\quad \text{arcre, arcae, ansre, ansae, maxnfe, sspar}) \tag{4.11}
\]

This function invokes the specified solver and returns the results. A call can be made to \(Hrf.findRoot()\) only after \(Hrf.initRootFinding()\) and \(Hrf.setImpl()\) are called. The parameters in this function have the same meanings as in (4.4). The first three parameters are mandatory and the remaining arguments have defaults that can be specified through a call to \(Hrf.setArgs()\) and recovered through a call to \(Hrf.getArgs()\). These functions simply allow the user to specify and the solver to recover the parameters \(\text{arcre}, \text{arcae}, \text{ansre}, \text{ansae}, \text{maxnfe},\) and \(\text{sspar}\).

\[
Hrf.setArgs(arcre, arcae, ansre, ansae, maxnfe, sspar) \tag{4.12}
\]
\[ [\text{arcre, arcae, ansre, ansea, maxnfe, sspar}] = \text{Hrf.getArgs() } \quad (4.13) \]

**Example 4.1.1** Suppose we have written a MATLAB function with the same signature as (4.4) called `mySolver` and routines `myF`, `myFJac`, `myRho`, and `myRhoJac` with signatures corresponding to (4.6) – (4.9). To initialize the `Root Finding` package the following code may be executed.

\begin{verbatim}
\text{Hrf.setImpl('mySolver');}
\text{H_initRootFinding('myF', 'myFJac', 'myRho', 'myRhoJac');}
\end{verbatim}

These calls associate the MATLAB functions implementing \( F \) and \( \rho_a \) with the solver whose implementation is contained in the MATLAB function called `mySolver`. The solver has visibility to the functions `Hrf.F`, `Hrf.FJac`, `Hrf.rho`, and `Hrf.rhojac` in the `Root Finding` package. These routines have the same signature as the respective arguments passed into (4.5), and calls to them will result in the invocation of those functions passed into `H_initRootFinding()`.

A client may then invoke the solver with a call to

\[[x \text{ njac arclen status}] = \text{H_findRoot(length(x), x, A)};\]

which uses default arguments, or by the extended form:

\[[x \text{ njac arclen status}] = \text{H_findRoot(n, x0, A,}
\text{ arcre, arcae, ansre, ansea, maxnfe, sspar);}\]

The advantage of the approach taken in this package is that it decouples the code implementing the solver from the code that uses it. The solver must only be aware of the interfaces to the functions in the `Root Finding` package, while clients are free to vary the implementation and names of the functions invoked by this package. This type of decoupling is a dominant theme in the design of HOMTOOLS.

48
Passing the names of functions in this manner is meant to simulate the use of function pointers that are available in a language such as C. Unfortunately this practice is more hazardous in the MATLAB environment because it lacks compiler type safety and a namespacing or scoping mechanism. An incorrect calling sequence will, in most cases, produce a meaningful runtime error, however.

4.1.2 The Utility Package. The utility package contains routines that are used by the remaining sections. They are germane to the reformulation of a complementarity problem into a parameterized smoother \( H^{p(\lambda)}(x) \) (2.26) for \( H(x) \) (2.12) as discussed in Section 2.7. It contains the following routines.

\[
\begin{align*}
\mu &= \text{Hutil\_mu}(\lambda) \\
\frac{d\mu}{d\lambda} &= \text{Hutil\_muJ}(\lambda) \\
\alpha &= \text{Hutil\_Alpha}(\alpha) \\
\rho &= \text{Hutil\_phi}(\mu, a, b) \\
\frac{d\rho}{d\lambda} &= \text{Hutil\_phiJ}(\mu, a, b) \\
\rho &= \text{Hutil\_psi}(\mu, l, u, x, f) \\
\frac{d\rho}{d\lambda} &= \text{Hutil\_psiJ}(\mu, l, u, x, f)
\end{align*}
\]

Currently this package implements \text{Hutil\_mu} as

\[
\mu_\alpha(\lambda) := \alpha(1 - \lambda)^2
\]

where the parameter \( \alpha \) is obtained through a call to \text{Hutil\_Alpha()}\), which may be specified by the user. If an argument is provided to \text{Hutil\_Alpha()}\) the package takes that to be the current \( \alpha \). If no argument is provided the function simply returns the current value of \( \alpha \).
\texttt{Hutil.phi} is implemented as the smoother for the Fischer-Burmeister function (2.24)

\[ \phi_\mu(a, b) := a + b - \sqrt{a^2 + b^2 + 2\mu}. \]

The \texttt{Hutil.psi} routine implements the Kanzow MCP smoother (2.25) as

\[ \psi_\mu(l, u, x, f) := \phi_\mu(x - l, -\phi_\mu(u - x, -f)). \]

The implementation of these functions is particular to a specific reformulation of the Mixed Complementarity Problem. If these functions are not suitable for a particular reformulation they may be replaced with functions, which have more appropriate implementations. For example, the component-wise median function is a valid NCP function (see Definition 2.4.1), so any implementation of a smoother for that function may be substituted for \texttt{Hutil.phi} and \texttt{Hutil.phijac}.

\textbf{4.1.3 The Homotopy Smooth Package.} Although the Root Finding package may be used directly, clients must supply it with meaningful homotopies. The \textit{Homotopy Smooth} package is used to create those homotopies for nonsmooth systems that we are considering in this thesis. To do this it uses the homotopy map presented in Chapter 3

\[ \rho_\alpha(\lambda, x) := \lambda F^{\mu_\alpha(\lambda)}(x) + (1 - \lambda)(x - \alpha) \]

(4.21)

where the smoothing parameter is governed by

\[ \mu_\alpha := \alpha(1 - \lambda)^2, \]

which is implemented by the \texttt{Hutil.mu()} (4.14) function in the \textit{Utility} package. Here it is assumed that \( F \) has a valid smoother \( F^\nu \).
To construct a function implementing the homotopy mapping, the package must be provided with routines to evaluate $F$, $\nabla F$, and their associated smoothers $F^\nu$ and $\nabla F^\nu$. This is accomplished by initializing the package through a call to

$$H\_\text{initHomotopySmooth(FName,FJacName,F\_muName,FJac\_muName)} \tag{4.22}$$

where FName and FJacName are the single quoted string names of MATLAB functions implementing $F$ and $\nabla F$, and F\_muName and FJac\_muName represent their smoothers. This call provides the following functions with the implementations provided as inputs. For example, a call to Hhs.F will result in a call to the function specified in the above FName parameter.

$$y = Hhs.F(x) \tag{4.23}$$

$$dy = Hhs.FJac(x,k) \tag{4.24}$$

$$y = Hhs.rho(lambda,x) \tag{4.25}$$

$$dy = Hhs.rhojac(lambda,x,k) \tag{4.26}$$

Here $dy$ is an $n \times n$ matrix. The call to H\_initHomotopySmooth() also initializes the Root Finding package by calling H\_initRootFinding() and passing in the function names of (4.23) - (4.26) as arguments, thus providing a homotopy mapping for dealing with nonsmooth systems. Because the function $F$ need not be smooth, the function Hhs.FJac is required to return an element of the B-subdifferential of $F$.

**4.1.4 The MCP Smooth Package.** The purpose of the MCP Smooth package is to reformulate an MCP into a system of equations using a smoothed version of the reformulation presented in Section 2.4. To reformulate
the problem \( \text{MCP}(F, \mathcal{B}_{l,u}) \) (Section 2.3) this package needs to be provided with MATLAB functions through a call to

\[
\text{H.initMCPSmooth}(\text{FName}, \text{FJacName}) \tag{4.27}
\]

where \text{FName} and \text{FJacName} are the names of MATLAB functions implementing \( F \) and its Jacobian \( \nabla F \). This function initializes the *Homotopy Smooth* package with a call to \text{H.initHomotopySmooth()} with the following functions as arguments.

\[
y = \text{Hmcp.H}(x) \tag{4.28}
\]

\[
dy = \text{Hmcp.HJjac}(x) \tag{4.29}
\]

\[
y = \text{Hmcp.Hmu}(x) \tag{4.30}
\]

\[
dy = \text{Hmcp.HJjac.mu}(x) \tag{4.31}
\]

To solve an MCP, simply initialize the package and invoke the function

\[
[z \ njac \ arclen \ status] = \text{H.solveMCP}(z0, 1, u) \tag{4.32}
\]

where \( z0 \) is a starting point for the problem and \( 1 \) and \( u \) are the vectors of lower and upper bounds that define the box \( \mathcal{B}_{l,u} \). The output parameters are identical to those in (4.11).

Since the stopping criteria for the algorithm is solver dependent, an independent check for solution quality is necessary. This is provided by the function

\[
\text{residual} = \text{Hmcp.testSoln}(z) \tag{4.33}
\]

which must be invoked after the problem has been initialized and run. Since \( z \) solves \( \text{MCP}(F, \mathcal{B}_{l,u}) \) if and only if \( \text{mid}(x - l, x - u, F(x)) = 0 \), the residual vector \( r \) is computed as

\[
r = |\text{mid}(x - l, x - u, F(x))|
\]
where $\cdot$ is the componentwise absolute value function. The output parameter residual should be nearly zero in every component if a solution to the problem defined by previous calls to `H_initMCPSmooth()` and `H_solveMCP()` is given as an input argument.

**4.1.5 The GAMS MCP Package.** The GAMS MCP package is a special extension to MCP Smooth, which allows a complementarity problem to be specified in GAMS and imported into and solved in MATLAB. This package is implemented using a GAMS to MATLAB interface presented in [8]. It allows developers to test solvers against entire libraries of complementarity problems specified in the GAMS environment [2].

The GAMS MCP package offers clients only one function:

$$[z \ njac \ arclen \ status] = H\_solveGamsMCP(num,literal,name).$$  
(4.34)

The input parameter name is a single quoted string containing the name of a GAMS model. The num parameter is an integer specifying the point at which point to start; this point is provided by the GAMS to MATLAB interface described in [8]. Alternatively, one could pass in a vector into the num parameter. If it is of correct dimension and the literal argument is not zero, the routine will use it as a starting point for the MCP. The output parameters have the same meanings as in (4.11).

**Example 4.1.2** A typical case that illustrates how to use the GAMS MCP package with HOMTOOLS follows.

```matlab
Hrf_setImpl('H_hompack');
Hutil_Alpha(0.8);
Hrf_setArgs(10e-2,10e-2,10e-12,10e-12,1000,20.0,[0 0 0 0 0 0 0]);
[z njac arclen status] = H\_solveGamsMCP(2,0,'josephy');
```

53
residual = Hmcp_testSoln(z);
This MATLAB program will attempt to solve the josephy model from the
MCPLIB library using a solver called H_hompack and report if the problem was
solved. The client need only specify which solver to use and which problem
to solve. The other calls serve to customize the solver. This customization is
not strictly necessary as the parameters have reasonable defaults; however, it
is usually desirable to do so.

4.1.6 The Algorithms Package. This package currently con-
tains two routines exported from HOMPACK90. The H_fixpnf() algorithm is
a Mex interface to the routine described in detail in Section 2.6.2. It has the
following interface.

$$[z \ njac \ arclen \ iflag] = \ H\_fixpnf(n,x0,A,$$
$$\quad \text{arcre,arcae,ansre,ansae,iflag,sspar}); \tag{4.35}$$

The H_stepnf() routine generates the iterates inside the H_fixpnf() routine.
It has the following signature.

$$[y \ yp \ yold \ ypold\ h \ hold \ njac \ arclen \ iflag] =$$
$$\ H\_stepnf(n,A,y,yp,yold,abser,relerr,iflag,start) \tag{4.36}$$

This function takes y to be the current point on the zero curve and yp to be
the associate tangent vector. ypold is a tangent vector to which yp must be
acute. The abser and relerr parameters are used to detect convergence in
the corrector phase of the algorithm. The start parameter should be set to
1 the first time the routine is called. iflag is an error status documented
in HOMPACK90. It should be -2 on input and 0 on output. Should iflag
return greater than 0, then the algorithm has failed and iflag is an error code
documented in the HOMPACK90 source code.
4.2 Solver Implementation

To implement the hybrid algorithm presented in Figure 3.1, we used HOMTOOLS in the MATLAB environment. The nonsmooth Newton’s method used for local convergence in Step 3 of the algorithm is a straight-forward implementation of the algorithm presented in Figure 2.1.

To implement Step 2 of the hybrid algorithm, we use a routine similar to HOMPACK90’s FIXPNF (see Section 2.6.2) to track the zero curve of the homotopy map. This routine is a MATLAB function that uses a MEX interface to the STEPNF routine from HOMPACK90 to generate iterates along the zero curve. Its implementation differs from FIXPNF’s in that it omits the end game strategy. The routine takes an input parameter $\delta \in (0, 1)$ so that as soon as it produces a point $(\lambda, x)$ on the zero curve with $\lambda \in (1 - \delta, 1)$ it returns and lets the Newton method take care of the end game convergence. If the STEPNF routine produces an iterate such that its $\lambda$ component is greater than one, the algorithm simply throws away the point and calls STEPNF again with the maximum allowable stepsize reduced by some prescribed factor (default is 0.5).

As with the end game strategy, our curve tracking algorithm differs from the FIXPNF routine in that it uses a different heuristic to control how tightly the iterates follow the zero curve when it detects that the curve is particularly nonlinear. Assume that the previous iterate on the zero curve we have accepted is $(\lambda_0, x^0)$ and that the STEPNF routine indicates that $(\lambda, x)$ should be the next. This algorithm will accept this point as the next iterate provided that the angle between $(\lambda_0, x^0)$ and $(\lambda, x)$ is smaller that some prescribed angle (it defaults to 60 degrees). This strategy is intended to ensure that the iterates follow the zero curve more closely when the curve is highly nonlinear. The intent is to prevent the algorithm from producing a point that is on a different
component of $\varphi^{-1}_{a}(\{0\})$ and to make sure that it does not start tracking the curve in the wrong direction of the correct component.

To solve complementarity problems, we used the MATLAB to GAMS interface described in [8]. This interface allows one to specify an MCP in the GAMS environment and obtain the bounds and evaluate the function and associated Jacobian that define the MCP from the MATLAB environment. These functions and bounds are accessed through the GAMS MCP package in HOMTOOLS. The function implementing our hybrid algorithm is passed into the Hrf.setImpl() function in the Root Finding package.
5. Results and Conclusions

To test the performance of the algorithm presented in Chapter 3 we used the implementation discussed in Section 4.2 along with the FIXPNF routine from HOMPACK90 on the MCPLIB, a set of Mixed Complementarity Problems specified in the GAMS language. The performance of several complementarity solvers has been tested and documented against this library [2]. Because the HOMTOOLS package can incur a large overhead and currently only supports the use of dense Jacobians, we tested the algorithm on all problems in the MCPLIB having fewer than 100 variables at all starting points provided by the GAMS to MATLAB interface in [8].

5.1 Testing Procedure

For initial testing, we chose a set of default parameters to use in the hybrid algorithm, which takes as input the same arguments as the FIXPNF routine described in Section 2.6.2 as well as those specified in Step 1 of the algorithm in Figure 3.1. For the curve tracking tolerance parameters we chose values of $arcae = arcre = 10^{-2}$. These loose curve tracking parameters were chosen so that reasonable progress would be made along the homotopy’s zero curve. The answer tolerances $\epsilon$, $ansae$, and $ansre$ were chosen to be $10^{-10}$. The $sspar$ parameters from FIXPNF (also used in STEPNF, described in Section 2.6.2) are taken as their default values in HOMPACK90.

We tested the method in three ways. First as a generalized Newton’s method. The $\delta$ parameter is chosen to be 1.0 on initialization. This causes the Newton method to be executed before the homotopy phase during the first
<table>
<thead>
<tr>
<th>Problem Name</th>
<th>st. pt.</th>
<th>size</th>
<th>( \delta = 1 )</th>
<th>( \delta = 0.01 )</th>
<th>FIXPNF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Step 2 calls</td>
<td>Jac evals</td>
<td>Step 2 calls</td>
</tr>
<tr>
<td>badfree</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>bertsekas</td>
<td>1</td>
<td>15</td>
<td>7</td>
<td>424</td>
<td>3</td>
</tr>
<tr>
<td>bertsekas</td>
<td>2</td>
<td>15</td>
<td>7</td>
<td>562</td>
<td>3</td>
</tr>
<tr>
<td>bertsekas</td>
<td>3</td>
<td>15</td>
<td>12</td>
<td>fail</td>
<td>3</td>
</tr>
<tr>
<td>billups</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>box</td>
<td>1</td>
<td>44</td>
<td>1</td>
<td>fail</td>
<td>1</td>
</tr>
<tr>
<td>colvdual</td>
<td>1</td>
<td>20</td>
<td>1</td>
<td>53</td>
<td>1</td>
</tr>
<tr>
<td>colvdual</td>
<td>2</td>
<td>20</td>
<td>1</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>colvlp</td>
<td>1</td>
<td>15</td>
<td>1</td>
<td>61</td>
<td>1</td>
</tr>
<tr>
<td>colvlp</td>
<td>2</td>
<td>15</td>
<td>0</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>colvtemp</td>
<td>1</td>
<td>20</td>
<td>1</td>
<td>53</td>
<td>1</td>
</tr>
<tr>
<td>degen</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>ehl.k40</td>
<td>1</td>
<td>41</td>
<td>1</td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>ehl.k60</td>
<td>1</td>
<td>61</td>
<td>1</td>
<td>fail</td>
<td>1</td>
</tr>
<tr>
<td>ehl.k80</td>
<td>1</td>
<td>81</td>
<td>1</td>
<td>423</td>
<td>1</td>
</tr>
<tr>
<td>explcp</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>freebert</td>
<td>1</td>
<td>15</td>
<td>7</td>
<td>614</td>
<td>3</td>
</tr>
<tr>
<td>freebert</td>
<td>2</td>
<td>15</td>
<td>0</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>freebert</td>
<td>3</td>
<td>15</td>
<td>7</td>
<td>253</td>
<td>3</td>
</tr>
<tr>
<td>freebert</td>
<td>4</td>
<td>15</td>
<td>7</td>
<td>582</td>
<td>3</td>
</tr>
<tr>
<td>freebert</td>
<td>5</td>
<td>15</td>
<td>0</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>freebert</td>
<td>6</td>
<td>15</td>
<td>7</td>
<td>433</td>
<td>3</td>
</tr>
<tr>
<td>gafni</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>gafni</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>gafni</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>games</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>hanskoop</td>
<td>1</td>
<td>14</td>
<td>2</td>
<td>36</td>
<td>1</td>
</tr>
<tr>
<td>hanskoop</td>
<td>2</td>
<td>14</td>
<td>1</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>hanskoop</td>
<td>3</td>
<td>14</td>
<td>1</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>hanskoop</td>
<td>4</td>
<td>14</td>
<td>1</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>hanskoop</td>
<td>5</td>
<td>14</td>
<td>2</td>
<td>87</td>
<td>1</td>
</tr>
<tr>
<td>hydroc06</td>
<td>1</td>
<td>29</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 5.2. MCPLIB Test Problems (continued)

<table>
<thead>
<tr>
<th>Problem Name</th>
<th>st. pt.</th>
<th>size</th>
<th>Step 2 calls</th>
<th>Jac evals</th>
<th>Step 2 calls</th>
<th>Jac evals</th>
<th>FIXPNF Jac evals</th>
</tr>
</thead>
<tbody>
<tr>
<td>jcl</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>120</td>
<td>123</td>
</tr>
<tr>
<td>josephy</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>josephy</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>josephy</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>279</td>
<td>1</td>
<td>208</td>
<td>217</td>
</tr>
<tr>
<td>josephy</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>josephy</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>josephy</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>21</td>
<td>16</td>
</tr>
<tr>
<td>kojshin</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>22</td>
<td>31</td>
</tr>
<tr>
<td>kojshin</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>210</td>
<td>238</td>
</tr>
<tr>
<td>kojshin</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>210</td>
<td>15</td>
</tr>
<tr>
<td>kojshin</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>26</td>
<td>19</td>
</tr>
<tr>
<td>kojshin</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>23</td>
<td>17</td>
</tr>
<tr>
<td>mathinum</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>mathinum</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>22</td>
<td>13</td>
</tr>
<tr>
<td>mathinum</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>mathinum</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>mathisum</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>mathisum</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>mathisum</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>29</td>
<td>23</td>
</tr>
<tr>
<td>mathisum</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>19</td>
<td>15</td>
</tr>
<tr>
<td>methan08</td>
<td>1</td>
<td>31</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>fail</td>
<td>fail</td>
</tr>
<tr>
<td>multi-v</td>
<td>1</td>
<td>48</td>
<td>1</td>
<td>fail</td>
<td>1</td>
<td>fail</td>
<td>fail</td>
</tr>
<tr>
<td>nash</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>32</td>
<td>28</td>
</tr>
<tr>
<td>nash</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>115</td>
<td>fail</td>
</tr>
<tr>
<td>ne-hard</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>25</td>
<td>1</td>
<td>251</td>
<td>fail</td>
</tr>
<tr>
<td>pgvon105</td>
<td>1</td>
<td>105</td>
<td>2</td>
<td>203</td>
<td>3</td>
<td>339</td>
<td>fail</td>
</tr>
<tr>
<td>pgvon105</td>
<td>2</td>
<td>105</td>
<td>1</td>
<td>110</td>
<td>1</td>
<td>86</td>
<td>fail</td>
</tr>
<tr>
<td>pgvon105</td>
<td>3</td>
<td>105</td>
<td>1</td>
<td>101</td>
<td>1</td>
<td>110</td>
<td>fail</td>
</tr>
<tr>
<td>pgvon106</td>
<td>1</td>
<td>106</td>
<td>12</td>
<td>fail</td>
<td>5</td>
<td>fail</td>
<td>fail</td>
</tr>
<tr>
<td>pies</td>
<td>1</td>
<td>42</td>
<td>0</td>
<td>15</td>
<td>1</td>
<td>251</td>
<td>fail</td>
</tr>
</tbody>
</table>
### Table 5.3. MCPLIB Test Problems (continued)

<table>
<thead>
<tr>
<th>Problem Name</th>
<th>st. pt.</th>
<th>size</th>
<th>Step 2 calls</th>
<th>Jac evals</th>
<th>δ = 1</th>
<th>Step 2 calls</th>
<th>Jac evals</th>
<th>δ = 0.01</th>
<th>Step 2 calls</th>
<th>Jac evals</th>
<th>FIXPNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>powell</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>99</td>
<td>fail</td>
<td>fail</td>
<td>25</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>powell</td>
<td>2</td>
<td>16</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>powell</td>
<td>3</td>
<td>16</td>
<td>1</td>
<td>242</td>
<td>1</td>
<td>265</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>powell</td>
<td>4</td>
<td>16</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>62</td>
<td>226</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>powell_mcp</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>24</td>
<td>22</td>
<td></td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>powell_mcp</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>26</td>
<td>25</td>
<td></td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>powell_mcp</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>42</td>
<td>45</td>
<td></td>
<td>45</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>powell_mcp</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>29</td>
<td>28</td>
<td></td>
<td>28</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>qp</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>16</td>
<td>13</td>
<td></td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>scarbsum</td>
<td>1</td>
<td>40</td>
<td>3</td>
<td>fail</td>
<td>1</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfanum</td>
<td>1</td>
<td>13</td>
<td>1</td>
<td>39</td>
<td>1</td>
<td>37</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfanum</td>
<td>2</td>
<td>13</td>
<td>0</td>
<td>13</td>
<td>1</td>
<td>39</td>
<td>38</td>
<td></td>
<td>38</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>scarfanum</td>
<td>3</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>3</td>
<td>69</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfasum</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfasum</td>
<td>2</td>
<td>14</td>
<td>1</td>
<td>100</td>
<td>1</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfasum</td>
<td>3</td>
<td>14</td>
<td>1</td>
<td>64</td>
<td>1</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfbnum</td>
<td>1</td>
<td>39</td>
<td>12</td>
<td>589</td>
<td>5</td>
<td>224</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfbnum</td>
<td>2</td>
<td>39</td>
<td>10</td>
<td>523</td>
<td>5</td>
<td>224</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfsbum</td>
<td>1</td>
<td>40</td>
<td>1</td>
<td>74</td>
<td>2</td>
<td>234</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>scarfbsum</td>
<td>2</td>
<td>40</td>
<td>7</td>
<td>273</td>
<td>3</td>
<td>183</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>shubik</td>
<td>1</td>
<td>42</td>
<td>20</td>
<td>fail</td>
<td>11</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>simple-ex</td>
<td>1</td>
<td>17</td>
<td>1</td>
<td>42</td>
<td>1</td>
<td>39</td>
<td>40</td>
<td></td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>simple-red</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>30</td>
<td>31</td>
<td></td>
<td>31</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>sppe</td>
<td>1</td>
<td>27</td>
<td>1</td>
<td>69</td>
<td>1</td>
<td>114</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>sppe</td>
<td>2</td>
<td>46</td>
<td>1</td>
<td>46</td>
<td>3</td>
<td>87</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td></td>
</tr>
<tr>
<td>tobin</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>82</td>
<td>1</td>
<td>82</td>
<td>83</td>
<td></td>
<td>83</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>tobin</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>79</td>
<td>78</td>
<td></td>
<td>78</td>
<td>78</td>
<td></td>
</tr>
</tbody>
</table>
time through the hybrid algorithm. If the Newton method solves the problem, the homotopy method is never invoked. The $\delta$-reduction factor $\sigma$ was chosen to be 0.5. The second configuration used the same hybrid algorithm, except with the initial $\delta = 0.01$. This causes the homotopy method to do most of the actual work, using Newton's method only for the 'end game' phase. If no solution was produced within 30 iterations of the hybrid method the algorithm was terminated. If Step 3 produced more than 1000 points on the homotopy curve without returning that phase was terminated. Finally, we ran the problem set with the FIXPNF routine from HOMPACK, including the standard 'end game' strategy from FIXPNF.

5.2 Computational Results

Using a single set of parameters the implementation of the algorithm described in Chapter 3 was able to solve many problems on which Newton's method alone failed. There were also some problems that Newton's method solved and the hybrid algorithm could not solve by using the homotopy method. The FIXPNF routine failed on many problems. This is can be attributed to the fact that the standard 'end game' strategy of FIXPNF was not designed to handle nonsmooth functions.

5.3 Conclusions

While this algorithm solved most of the test problems, compared to other complementarity solvers [2] its current implementation is very slow in terms of the number of Jacobian evaluations required to produce a solution. A degree of slowness is characteristic of homotopy based algorithms, however, what they lack in speed they tend to make up for in robustness. The theoretical foundation presented in Chapter 3 almost surely (in the sense of Lesbegue
measure), under very weak assumptions, guarantees the existence of a zero curve that can be tracked to a solution. This theoretical foundation ensures that any algorithm that can accurately track the zero curve will converge to a solution.

In spite of the solid theory supporting the algorithm, this method failed altogether on a number of problems and required carefully chosen parameters to solve several others. For some of these problems, the failure was due to the fact that the Jacobian of the problem's reformulation was singular at the solution. In this case the theory suggests the algorithm could fail because the zero curve of the homotopy map may have infinite arc length.

The homotopy-based approach for solving nonsmooth systems presented in this paper is neither extremely robust nor efficient in its current implementation. It does, however, bring to bear some powerful theory, which suggests that it should be very robust for a large class of problems. It remains to determine exactly why the algorithm performs poorly or fails altogether on many problems. The poor performance might be due to the difficulty of accurately tracking the homotopy curves or it could be that some problems are violating a theoretical requirement.

5.4 Avenues for Further Research

At present this algorithm could be used in cases where more conventional methods fail. To improve the robustness and performance of the algorithm, there are two approaches that can be taken: develop homotopy maps that induce a more well-behaved zero curve, or develop a better approach for tracking the curve induced by the map presented in Chapter 3. One way to accomplish the latter task might be to develop a new curve tracking paradigm whereby the zero curve is tracked with increasing tolerance as progress is made
along it. This contrasts the approach presented in HOMPACK90's FIXPNF algorithm, which tracks the zero curve with a uniform tolerance all the way until the end game phase. Such an approach may be more efficient because the algorithm can take longer steps along the zero curve during the beginning of the algorithm and may be able to make good progress even when the zero curve has sharp turns. This approach may also dove-tail more elegantly into the end game phase of the algorithm since the curve tracking tolerance shrinks as it approached a solution.
REFERENCES


