

65026

IMPLEMENTATION OF A CONTINUATION
METHOD FOR NONLINEAR COMPLEMENTARITY
PROBLEMS VIA NORMAL MAPS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL
ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY

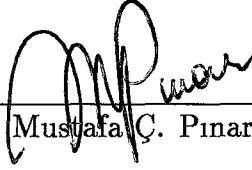
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By

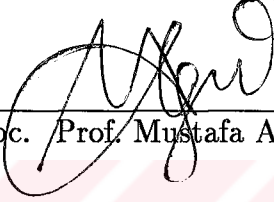
Ali Erkan

August, 1997

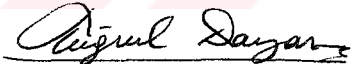
I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.


Assist. Prof. Mustafa Ç. Pinar (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.


Assoc. Prof. Mustafa Akgül

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.


Assist. Prof. Tuğrul Dayar

Approved for the Institute of Engineering and Sciences:


Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

IMPLEMENTATION OF A CONTINUATION METHOD FOR NONLINEAR COMPLEMENTARITY PROBLEMS VIA NORMAL MAPS

Ali Erkan

M.S. in Industrial Engineering

Supervisor: Assist. Prof. Mustafa Ç. Pınar

August, 1997

In this thesis, a continuation method for nonlinear complementarity problems via normal maps that is developed by Chen, Harker and Pınar [2] is implemented. This continuation method uses the smooth function to approximate the normal map reformulation of nonlinear complementarity problems. The algorithm is implemented and tested with two different plus-smoothing functions namely interior point plus-smooth function and piecewise quadratic plus-smoothing function. These two functions are compared. Testing of the algorithm is made with several known problems.

Key words: Smoothing Methods, Nonlinear Programming, Complementarity, Continuation Methods, Normal Maps

ÖZET

DOĞRUSAL OLMAYAN TAMAMLAYICI PROBLEMLER İÇİN NORMAL MEPLİ BİR SÜREKLİLİK YÖNTEMİNİN UYGULANMASI

Ali Erkan

Endüstri Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Doç. Mustafa Ç. Pınar

Ağustos, 1997

Bu tezde, Chen, Harker ve Pınar [2] tarafından doğrusal olmayan tamamlayıcı (complementarity) problemler için geliştirilen ve normal mepli bir süreklilik (continuation) yöntemi uygulandı. Bu süreklilik metodu, doğrusal olmayan tamamlayıcı problemlerin normal mepli modellenmesine yaklaşık sonuç bulabilmek için düzeltici (smoothing) fonksiyonlar kullanıldı. Algoritma iki farklı artı-düzleştirici fonksiyonla; iç noktalama (interior point) artı-düzleştirici fonksiyonu ve parçalı ikinci dereceli (piecewise quadratic) artı-düzleştirici fonksiyonuyla, uygulandı ve test edildi. Bu iki fonksiyon karşılaştırıldı. Algoritmanın testi birçok sayıda bilinen problemle yapıldı.

Anahtar sözcükler: Düzeltici (Smoothing) Yöntemler, Doğrusal Olmayan Programlama, Tamamlayıcılık (Complementarity), Süreklilik (Continuation) Yöntemleri, Normal Mep.

To my family



ACKNOWLEDGEMENT

I am mostly grateful to Mustafa Ç. Pinar for suggesting this interesting research topic, and who has been supervising me with patience and everlasting interest and being helpful in any way during my graduate studies.

I am also indebted to Musafa Akgül and Tuğrul Dayar for showing keen interest to the subject matter and accepting to read and review this thesis. Their remarks and recommendations have been invaluable.

I would like to thank to Micheal Ferris who has aided me with his valuable suggestions and his test problems and softwares.

I have to express my gratitude to the technical and academical staff of Bilkent University. I am especially thankful to Murat Saraç, Nesrin Çolak, Olcay Eraslan, H. Çağrı Sağlam, Feryal Erhun, Serkan Özkan, Alev Kaya, Hülya Emir for helping me in any way during the entire period of my M.S. studies and Çağrı Sağlam's mother, who I had not chance to meet, for giving us Çağrı. I can not forget the aids of who always supported me with my master's thesis.

Finally I have to express my sincere gratitude to anyone who have been of help, which I have forgotten to mention here.

Contents

1	INTRODUCTION	1
2	LITERATURE REVIEW	3
2.1	Overview	3
2.2	Mathematical Programming	5
2.2.1	Linear Programming	6
2.2.2	Quadratic Programming	7
2.2.3	Nonlinear Programming	10
2.3	The Homotopy Principle	11
2.3.1	Varieties of Homotopies	12
2.4	Lemke's Algorithm	13
2.5	Non-differentiable Equation Approach	17
2.5.1	The Damped-Newton algorithm	18
2.6	Continuation Methods	19

2.6.1	Interior-point continuation method for complementarity problems with uniform P -functions	20
2.6.2	Kanzow's algorithm:	20
2.7	Smoothing Methods	21
2.7.1	A class of smoothing functions	22
2.7.2	Application to The Nonlinear Complementarity Problem	24
2.7.3	Mixed Complementarity Problem	25
2.8	Non-interior Continuation Methods for the NCP	27
2.8.1	A Continuation Method for NCP	29
3	SMOOTH APPROXIMATION FOR NORMAL MAP FORMULATION OF NCP	32
3.1	Application to the Nonlinear Complementarity Problem	38
3.1.1	Existence of Solution Path	39
3.1.2	Boundedness of Trajectory T	40
3.1.3	Existence of Monotone Trajectory T	43
4	SUBPROBLEMS OF NEWTON CORRECTOR	47
5	THE ALGORITHM AND SOFTWARE	53
5.1	The Algorithm	53
5.1.1	Continuation Method	54
5.2	Software	55

5.2.1 Numerical Example	55
6 COMPUTATIONAL RESULTS	61
6.1 Test Problems	61
6.2 Experiment 1: Uniform Reduction of u	64
6.3 Experiment 2: A Hybrid Reduction Scheme for u	74
7 CONCLUSION	76
A User's Guide to NCPNMS: A Fortran 77 Package to solve NCP	86
A.1 Purpose	86
A.2 Specification of the Subroutine NCPNMS	86
A.3 Description of Parameters	87
A.4 UMFPACK2	90
A.5 BLAS	91
A.6 Harwell Routines	91
B Fortran 77 Code of Algorithm	92

List of Figures

6.1	Interior Plus-Smooth Function, CPU Time versus n	69
6.2	Piecewise Quadratic Plus-Smooth Function, CPU Time versus n	70
6.3	Piecewise Quadratic Plus-Smooth Function with Reduced Matrix	71
6.4	Comparison For CPU Time	72
6.5	Comparisons For CPU Time Per Iteration	73

List of Tables

6.1	Interior Plus-Smooth Function	66
6.2	Piecewise Quadratic Plus-Smooth Function	67
6.3	Piecewise Quadratic Plus-Smooth Function with Reduced Jacobian Matrix	68
6.4	Hybrid Reduction Scheme	75

Chapter 1

INTRODUCTION

Complementarity problems are important areas of applied mathematics due to their numerous applications: The complementarity theory derives its importance from the fact it unifies problems in fields such as: mathematical programming, game theory, the theory of equilibrium in a competitive economy, equilibrium of traffic flows, mechanics, engineering, lubricant evaporation in the cavity of a cylindrical bearing, elasticity theory, fluid flow through a semiimpermeable membrane, maximizing oil production, computation of fixed point etc.

Any mathematical programming model can be modeled as complementarity problem. Complementarity problem has been of great interest in the academic and professional communities ever since the path-breaking paper by Lemke and Howson. Especially after continuation methods and smoothing approaches complementarity problems became a very important subject. Because by using continuation methods and smoothing approaches, complementarity problems can be solved in computationally efficient time.

In this study, a continuation method for nonlinear complementarity problems via normal maps which was developed by Chen, Harker and Pinar [2]

is implemented. A software which is based on this algorithm is developed in ANSI FORTRAN 77 to solve numerous complementarity problems, and the software package uses a sparse linear system solver UMFPACK2.

The organization of the thesis is as follows. In the next chapter definitions of nonlinear complementarity problem (NCP) and a summary of the most well known algorithms to solve NCPs are given. Then the smoothing method approximation and application of smoothing method for NCPs are explained in the third chapter. The fourth chapter consists of subproblems of Newton corrector we study. Numerical example is in the fifth chapter, explanation of algorithm is in the chapter six and numerical testing and comparison is in the chapter seven. The thesis concludes with some remarks and suggestions for future research.

The following notation will be used throughout the thesis. All matrices are denoted by boldface letters. R^n, R_+^n, R_{++}^n denote, respectively, n dimensional Euclidean space, the nonnegative orthant of R^n , and the strictly positive orthant of R^n .

Chapter 2

LITERATURE REVIEW

2.1 Overview

In this section, Nonlinear Complementarity Problems (NCP) are defined mathematically and the related literature is briefly reviewed. To begin, let us define the problem under investigation:

Definition 2.1 *Let F be a mapping from R^n to itself. The nonlinear complementarity problem (NCP), denoted by $NCP(F)$ is to find a vector $x \in R^n$ such that:*

$$x \geq 0, \quad F(x) \geq 0 \quad \text{and} \quad F(x)^T x = 0.$$

The first two inequalities are called the feasibility conditions, and the equality is called the complementarity condition.

When F is affine, the NCP is reduced to the Linear Complementary Problem, which is defined as follows:

Definition 2.2 *Let $M \in R^{n \times n}$ and $q \in R^n$. The LCP is to find an $x \in R^n$ such that:*

$$x \geq 0, \quad w = Mx + q \geq 0 \quad \text{and} \quad w^T x = 0.$$

The following definitions of various special matrices and functions will be used in subsequent chapters.

Definition 2.3 The matrix $M \in \mathbb{R}^{n \times n}$ is said to be a

1. P_0 -matrix if for all $x \neq 0$, there exists an index i such that $x_i \neq 0$ and $x_i(Mx)_i \geq 0$,
2. copositive matrix if $x^T Mx \geq 0$ for all $x \geq 0$,
3. copositive-plus matrix if it is a copositive matrix and for all $x \geq 0$, if $x^T Mx = 0$, then $(M + M^T)x = 0$,
4. positive semi-definite matrix if $x^T Mx \geq 0$ for all x ,
5. P -matrix if for all $x \neq 0$, there exists an index i such that $x_i(Mx)_i > 0$,
6. positive definite matrix if $x^T Mx > 0$ for all $x \neq 0$,
7. R_0 -matrix if the following system has no solution:

$$x \geq 0,$$

$$M_{i,x} = 0 \text{ if } x_i > 0,$$

$$M_{i,x} \geq 0 \text{ if } x_i = 0,$$

8. Q -matrix if $LCP(M, q)$ has a solution for all q ,
9. Z -matrix if $m_{ij} \leq 0$ for all $i \neq j$.

Among the above special matrices, the following relations hold: positive definite matrix $\Rightarrow P$ -matrix $\Rightarrow P_0$ -matrix;

positive definite matrix $\Rightarrow$ positive semi-definite matrix $\Rightarrow P_0$ -matrix;

positive semi-definite matrix $\Rightarrow$ copositive plus matrix $\Rightarrow$ copositive matrix;

where $\Rightarrow$ denotes implication.

Definition 2.4 The mapping $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is said to be a

1. P_0 -function over a set X if

$$\max_{1 \leq i \leq n} [F_i(x) - F_i(y)]^T (x_i - y_i) \geq 0 \quad \forall x, y \in X, x \neq y,$$

2. P -function over a set X if

$$\max_{1 \leq i \leq n} [F_i(x) - F_i(y)]^T (x_i - y_i) > 0 \quad \forall x, y \in X, x \neq y,$$

3. *uniform P-function over a set X if there exists an $\alpha > 0$ such that*

$$\max_{1 \leq i \leq n} [F_i(x) - F_i(y)]^T (x_i - y_i) \geq \alpha \|x - y\|^2 \quad \forall x, y \in X,$$

4. *monotone function over a set X if*

$$[F(x) - F(y)]^T (x - y) \geq 0 \quad \forall x, y \in X,$$

5. *strictly monotone function over a set X if*

$$[F(x) - F(y)]^T (x - y) > 0 \quad \forall x, y \in X, x \neq y,$$

6. *strongly monotone function over a set X if*

$$[F(x) - F(y)]^T (x - y) \geq \alpha \|x - y\|^2 \quad \forall x, y \in X.$$

For the above definitions, the following relationships hold:

Strong monotonicity $\Rightarrow$ strict monotonicity $\Rightarrow$ monotonicity,

Uniform P-property $\Rightarrow$ P-property $\Rightarrow$ P_0 -property,

Strong monotonicity $\Rightarrow$ uniform P-property,

Strict monotonicity $\Rightarrow$ P-property,

Monotonicity $\Rightarrow$ P_0 -property [5].

Let us now consider some examples which lead to complementarity problems.

2.2 Mathematical Programming

In this section we consider several examples and models in finite dimensional spaces. Hence we consider the topological vector space R^n ordered by the pointed closed convex cone R_+^n and we denote by $\langle, \rangle$ the inner product, $\langle x, y \rangle = \sum_{i=1}^n x_i y_i; x = (x_i); y = (y_i)$ [23].

2.2.1 Linear Programming

Let $c = (c_i) \in R^n, b = (b_i) \in R^m$ be two vectors and let $\mathbf{A} = (a_{ij}) \in \mathbf{M}_{m \times n}(R)$ be a matrix.

Consider the primal linear program,

$$(P.L.P.) : \begin{cases} \text{minimize } \langle c, x \rangle \\ x \in F_1 \\ \text{where, } F_1 = \{x \in R_+^n \text{ and } \mathbf{A}x - b \in R_+^m\} \end{cases}$$

and its dual,

$$(D.L.P.) : \begin{cases} \text{maximize } \langle y, b \rangle \\ y \in F_2 \\ \text{where, } F_2 = \{y \in R_+^m \text{ and } \mathbf{A}^t y - c \in -R_+^n\} \end{cases}$$

A fundamental result of linear programming is the following.

Theorem 2.1 *If there exist $x_o \in F_1$ and $y_o \in F_2$ such that $\langle c, x_o \rangle = \langle y_o, b \rangle$ then x_o is a solution of problem (P.L.P.) and y_o is a solution of problem (D.L.P.).*

Using this result we can associate to the problem, (P.L.P.) and (D.L.P.) a complementarity problem.

Indeed, adding slack variables $u \in R^n$ and $v \in R^n$ such that, $\mathbf{A} - \mathbf{v} = b$ and $\mathbf{A}^t y + u = c$ and denoting,

$z = \begin{bmatrix} x \\ y \end{bmatrix}; w = \begin{bmatrix} u \\ v \end{bmatrix}; q = \begin{bmatrix} c \\ -b \end{bmatrix}; \mathbf{M} = \begin{bmatrix} \mathbf{0} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}$ we obtain the following complementarity problem,

$$(L.C.P.) : \begin{cases} \text{Find } z \in R^{n+m} \text{ such that} \\ z \in R_+^{n+m}, w = \mathbf{M}z + q \in R_+^{n+m} \text{ and } \langle z, w \rangle = 0. \end{cases}$$

We observe that this linear complementarity problem is equivalent to the couple primal-dual of linear programs (P.L.P.) - (D.L.P.).

Remarks:

- i-) Condition $\langle z, w \rangle = 0$ in the definition of problem (L.C.P.) was obtained observing that this condition expresses exactly the fact that $\langle c, x \rangle = \langle y, b \rangle$.
- ii-) The principal contribution of complementarity problem to the linear programming is that it transforms an optimization problem in an equation.

2.2.2 Quadratic Programming

Consider the quadratic programming problem,

$$(1) : \begin{cases} \text{Minimize } f(x) \\ x \in F \\ \text{where, } F = \{x \in R_+^n, b - Ax \in R_+^m\} \\ f(x) = \frac{1}{2} \langle x, Qx \rangle + \langle c, x \rangle, \\ c \in R^n, Q \in M_{n \times n}(R), (Q \text{ symmetric}), \\ A \in M_{m \times n}(R) \text{ and } b \in R^m \end{cases}$$

Denoting the Langrangian multiplier vectors of the constraints $Ax \leq b$ and $x \geq 0$ by $\lambda \in R^m$ and $u \in R^n$ respectively and denoting the vectors of slack variables by $v \in R^m$, the Kuhn-Tucker necessary optimality conditions could be written as:

$$(2) : \begin{cases} c + A^t \lambda + Qx - u = 0 \\ Ax + v = b \\ u \in R_+^n, v \in R_+^m, x \in R_+^n, \lambda \in R_+^m \\ \langle u, x \rangle = 0 \text{ and } \langle v, \lambda \rangle = 0. \end{cases}$$

Now, observe that conditions (2) can be written also as:

$$(3) : \begin{cases} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} + \begin{bmatrix} \mathbf{Q} & \mathbf{A}^t \\ -\mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} \\ \begin{bmatrix} u \\ v \end{bmatrix} \in R_+^{n+m}; \begin{bmatrix} x \\ \lambda \end{bmatrix} \in R_+^{n+m}; \langle \begin{bmatrix} u \\ v \end{bmatrix}, \begin{bmatrix} x \\ \lambda \end{bmatrix} \rangle = 0 \end{cases}$$

If we denote,

$$z = \begin{bmatrix} x \\ \lambda \end{bmatrix}; q = \begin{bmatrix} c \\ b \end{bmatrix}; \mathbf{M} = \begin{bmatrix} \mathbf{Q} & \mathbf{A}^t \\ -\mathbf{A} & \mathbf{0} \end{bmatrix} \text{ and } f(z) = \begin{bmatrix} u \\ v \end{bmatrix}$$

we obtain that the Kuhn-Tucker conditions (2) are equivalent to the following linear complementarity problem,

$$(4) : \begin{cases} \text{find } z \in R_+^{n+m} \text{ such that,} \\ z \in R_+^{n+m}; f(z) \in R_+^{n+m} \text{ and } \langle z, f(z) \rangle = 0 \end{cases}$$

It is remarkable to note that there exists another connection between linear programming, quadratic programming and the linear complementarity problem.

Consider the linear programming problem,

$$(5) : \begin{cases} \text{Minimize } \langle p, x \rangle \\ x \in F \\ \text{where: } F = \{x \in R^n | \mathbf{A}x - b \in R_+^m\} \\ p \in R^n, b \in R^m \text{ and } \mathbf{A} \in \mathbf{M}_{m \times n}(R) \end{cases}$$

Suppose that every row of $\mathbf{A}$ is different from zero and consider a quadratic perturbation on F of problem (5) of the form,

$$(6) : \begin{cases} \text{Minimize } [\langle p, x \rangle + \frac{\epsilon}{2} \langle x, x \rangle] \\ x \in F \end{cases}$$

In 1979 Mangasarian and Meyer proved the following result. [44].

Theorem 2.2 *If program (5) has an optimal solution then program (6) has a unique solution x_0 for every $\epsilon \in [0, \alpha]$ and some $\alpha > 0$.*

Moreover, the solution x_0 is independent of ϵ and it is also a solution of program (5).

Consider now a more general case, precisely the quadratic program,

$$(7) : \begin{cases} \text{Minimize } [\frac{1}{2} \langle x, \mathbf{Q}x \rangle + \langle p, x \rangle] \\ x \in F \\ \text{where; } F = \{x \in R^n | \mathbf{A}x - b \in R_+^m\}; \\ p \in R^n; b \in R^m; \mathbf{Q} \in M_{n \times n}(R) (\mathbf{Q} \text{ symmetric, positive definite}) \text{ and } \mathbf{A} \in M_{m \times n}(R) \end{cases}$$

The dual program of (7) is,

$$(8) : \begin{cases} \text{Maximize } [-\frac{1}{2} \langle x, \mathbf{Q}x \rangle + \langle b, u \rangle] \\ (x, u) \in F_1 \\ \text{where: } F_1 = \{(x, u) | x \in R^n, u \in R_+^m \text{ and } \mathbf{Q}x - \mathbf{A}^t u + p = 0\} \end{cases}$$

which under the positive definite assumption on $\mathbf{Q}$ is, (upon eliminating x , since from (8) $x = \mathbf{Q}^{-1}(\mathbf{A}^t u - p)$), equivalent to,

$$(9) : \begin{cases} \text{Minimize } (\frac{1}{2} \langle u, \mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^t u \rangle - \langle b + \mathbf{A}\mathbf{Q}^{-1}p, u \rangle) \\ u \in F_2 \\ \text{where: } F_2 = \{u \in R^m | u \in R_+^m\} \end{cases}$$

Since $\mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^t$ is positive semidefinite, (9) is equivalent to the following symmetric linear complementarity problem.

$$(S.L.C.P.) : \begin{cases} \text{Find } u \in R^m \text{ such that,} \\ u \in R_+^m; v = \mathbf{A}\mathbf{Q}\mathbf{A}^t u - (b + \mathbf{A}\mathbf{Q}^{-1}p) \in R_+^m \text{ and } \langle u, v \rangle = 0 \end{cases}$$

Readers can find more details on this subject in [43].

2.2.3 Nonlinear Programming

Consider the convex program,

$$(10) : \begin{cases} \text{Minimize } f(x) \\ x \in F \\ \text{where: } F = \{x \in R^n | x \geq 0 \text{ and } g_i(x) \leq 0; i = 1, 2, \dots, m\} \end{cases}$$

In this programming problem suppose all the functions convex and differentiable. The Lagrangian function $L(x, u)$ for (1) is given by,

$$L(x, u) = f(x) + \sum_{i=1}^m u_i g_i(x).$$

Hence, $u = (u_i) \in R^m$ and the Kuhn-Tucker necessary conditions for optimality can be written as:

$$(11) : \begin{cases} \frac{\partial L(x, u)}{\partial x_j} = h_j(x, u) \geq 0; j = 1, 2, \dots, n \\ -\frac{\partial L(x, u)}{\partial u_i} = h_{n+i}(x, u) \geq 0; i = 1, 2, \dots, m \\ x \geq 0, u \geq 0 \\ \sum_{j=1}^n x_j h_j(x, u) = 0 \text{ and } \sum_{i=1}^m u_i h_{n+i}(x, u) = 0 \end{cases}$$

If we denote, $z = \begin{bmatrix} x \\ u \end{bmatrix}$ and $h(z) = \begin{bmatrix} h_1(x) \\ \vdots \\ h_n(x) \\ \vdots \\ h_{n+m}(x) \end{bmatrix}$

then the Kuhn-Tucker conditions (11) may be stated as the following complementarity problem,

$$(C.P.) : \begin{cases} \text{find } z \in R^{n+m} \text{ such that,} \\ z \in R_+^{n+m}, h(z) \in R_+^{n+m} \text{ and} \\ \langle z, h(z) \rangle = 0 \end{cases}$$

Remark:

We have a similar construction for a nonlinear program (not necessarily convex), where f and $g_i (i = 1, \dots, m)$ are C^1 -functions on an open set U , such that $U \supset R_+^n$.

2.3 The Homotopy Principle

Suppose that a system of nonlinear equations is given for you to solve. Our immediate goal is to find a point $x = (x_1, \dots, x_n)$ that solves such a system. How might we accomplish this? One approach is to start with another system of equations to which we already know the solution. Usually, this is a particularly simple system that has an obvious solution. We then take this simple system and mathematically "bend" it into the original system. While bending the system we carefully watch the solution, as it also "bends" from the obvious solution into the solution we seek. This bending notion underlies a key idea that we shall develop shortly, the *homotopy* concept:

Let R^n denote *Euclidean n space*. A function $F : R^n \rightarrow R^n$ means that $F(x)$ has n components, $F(x) = (F_1(x), \dots, F_n(x))$, and that x has n components, $x = (x_1, \dots, x_n)$, so that

$$F_i(x) = F_i(x_1, \dots, x_n), \quad i = 1, \dots, n.$$

We are desirous of solving the $n \times n$ system of nonlinear equations

$$F(x) = 0$$

First set up a simple system

$$E(x) = 0.$$

Then define a special function, called a *homotopy function*, $H(x, t) : R^{n+1} \rightarrow R^n$, which has the original n variables plus an extra one, t . Here $(x, t) = (x_1, \dots, x_n, t) \in R^{n+1}$. The homotopy function H must be constructed so that

$$H(x, 0) = E(x)$$

$$H(x, 1) = F(x)$$

It follows that at $t = 0$,

$$H(x, 0) = 0$$

has a solution x^0 , which we already know, and at $t = 1$,

$$H(x, 1) = 0$$

has solution x^* , which we seek. In general for arbitrary t , $x(t)$ solves

$$H(x(t), t) = 0.$$

The idea is to start at $x(0) = x^0$ and then increase t until we reach $x(1) = x^*$. Generally, $x(t)$ will generate a path that we can follow from $t = 0$ to $t = 1$, thereby solving the original system [17].

2.3.1 Varieties of Homotopies

Newton Homotopy

$$H(x, t) = F(x) - (1 - t)F(x^0).$$

This form of homotopy is termed the *Newton homotopy* because some of the ideas behind it come from the work of Sir Isaac Newton himself. It is a particularly simple method to start. Pick an arbitrary point x^0 . Next calculate $F(x^0)$, and then let

$$E(x) = F(x) - F(x^0).$$

The function E , by construction, has solution x^0 . From first equation, the beginning of the path $x(0) = x^0$ is obvious and immediate. Since x^0 can be

selected arbitrarily, the Newton homotopy permits us to start the path from wherever we choose, a very nice feature.

Fixed-Point Homotopy

$$H(x, t) = (1 - t)(x - x^0) + tF(x).$$

This homotopy, called the *fixed – point homotopy*. Observe that

$$E(x) = x - x^0 = 0$$

so that $x(0) = x^0$. The fixed-point homotopy is thus also easy to start since x^0 can be chosen arbitrarily.

Both the fixed-point and Newton homotopies can be started from any point x^0 , which is one of the reasons that both are so widely employed.

The homotopy approach has been known to scientists at least since the nineteenth century. It is a standard tool in the theory of ordinary differential equations (Ficken [1951]). The first application to nonlinear equation systems seems to have been made by Lahaye [1948]. The first fail-safe general procedure for fixed point and related equation systems is due to Scarf [1967]. In actuality, Scarf did not use a homotopy approach but an entirely novel approach which he called primitive sets. An earlier work related to Scarf's fundamental work is Cohen [1967]. The underlying principle used by Scarf to prove convergence of his procedure is the "complementarity principle" of Lemke and Howson [1964].

2.4 Lemke's Algorithm

Lemke's algorithm is designed to solve the LCP. Given q and $\mathbf{M}$, first choose a positive vector $d \in R^n$ such that $d + q > 0$. Then consider the LCP:

$$z = d + q + \mathbf{M}x$$

$$z \geq 0, \quad x \geq 0, \quad z^T x = 0.$$

Clearly this equation has a trivial solution $x = 0$, where $z = d + q$. Now define the homotopy:

$$H_i(x, t) = \min\{x_i, z_i\} = 0, \quad i = 1, \dots, n$$

for $z = (1 - t)d + q + \mathbf{M}x$. This equation is precisely equivalent to the LC problem

$$z^T x = 0, \quad x \geq 0, \quad z \geq 0$$

where

$$z = (1 - t)d + q + \mathbf{M}x.$$

At $t = 0$, notice that the trivial LC with trivial solution $x = 0$ is obtained, whereas when $t = 1$, we have the original LC.

Lemke's method traces the path in H^{-1} starting from $(x, t) = (0, 0)$ of the homotopy

$$H_i(x, t) = \min\{x_i, z_i\} = 0, \quad i = 1, \dots, n$$

where

$$z = (1 - t)d + q + Mx$$

To avoid degeneracies a regularity condition is required: For each $(x, t) \in H^{-1}$, at least $n - 1$ of the variables x, z are greater than zero.

Algorithm

Step 0

Initially, $(x^0, t^0) = (0, 0)$, $B^0 = (x_1, \dots, x_n)$. Increase t from zero in the system

$$z + td = d + q$$

$$z \geq 0, \quad t \in R^1$$

1. If t can be increased to 1, then $x = 0, z = q \geq 0$, is an LC solution.
2. Otherwise, some z_i becomes zero in above equation for $t = t^1$. Let $(x^1, t^1) = (0, t^1)$, $w_l = x_l$ be the distinguished variable, and

$$B^1 = \{x_1, \dots, x_{l-1}, z_l, x_{l+1}, \dots, x_n\}.$$

Go to step 1.

Step k, $k \geq 1$.

Let (x^k, t^k) be the current point, w_l the distinguished variable, and $B^k = (u_1, \dots, u_n)$ the zero set. Set $u_1 = \dots = u_n = 0$. Then increase w_l from zero in

$$A^k w + t d = d + q$$

$$w \geq 0, \quad t \in R^1$$

where $A_i^k = e^i$ and $w_i = z_i$.

1. If t becomes equal to 1, terminate. We have an LC solution at hand.
2. If w_l increase to infinity in above equation, terminate. We have a path diverging to infinity.
3. Otherwise, some w_j becomes zero in above equation when $w_l = \bar{w}_l > 0$.

Let (x^{k+1}, t^{k+1}) be the new point corresponding to $w_l = \bar{w}_l$, the complement of w_j be the distinguished variable, and $B^{k+1} = (u_1, \dots, u_{j-1}, w_j, u_{j+1}, \dots, u_n)$. Go to step $k + 1$.

Repeat this process until $t = 1$ or the path diverges to infinity.

Lemke's algorithm historically form the basis for path-following procedures. The LC homotopy is not differentiable but does have a special structure which by use of Lemke's algorithm produces a piecewise-linear path [17].

I divided solution methods and algorithms for NCP into four classes; non-differentiable equation approach, continuation methods, smoothing methods and non-interior continuation methods for review. However, these classes are not strictly independent. Non-interior continuation methods consists of smoothing methods and continuation methods, also smoothing methods consists of continuation methods, etc... Now let's look at some algorithms which were developed after Lemke's algorithm for NCP. Firstly, we give make some definitions.

Definition 2.5 *If a method maintains strictly feasible ($x^k > 0$ or $F(x)^k > 0$ (strictly greater than zero) for every iteration of solution procedure of NCP, then method can be called interior method. Otherwise, it is called a non-interior method.*

A lot of solution approaches exist for solving the $\text{NCP}(F)$. Kostreva [19, 31, 32] developed a block-pivoting algorithm in which multiple exchanges of basic and nonbasic variables are executed. This algorithm extended Murty's scheme [48] for LCP's to the nonlinear complementarity setting. This method must solve a system of nonlinear equations at each iteration, and does not contain a line search step.

Subramanian [62] used a Gauss-Newton method to solve the nonlinear complementarity problem when formulated as the continuously differentiable equation system first introduced by Mangasarian [42]. While deriving some convergence results, no extensive computational evidence was reported.

Kojima, Mizuno, Noma and Yoshise [30, 34, 40, 39] have developed interior-point algorithms for solving monotone linear and nonlinear complementarity problems. When the matrix is positive semi-definite in the linear case, they show that the complexity of their algorithm is polynomially bounded; some limited computational results for this case were also reported. For the nonlinear case, the convergence theory exists for uniform P-functions, but only limited convergence results exist when F is monotone.

In addition to these methods, non-interior continuation methods also were developed. Non-interior continuation methods are closely related to path-following interior point algorithms. Non-interior continuation methods take a different deformation of the complementarity condition. As a result, they did not have to restrict intermediate iterates to stay interior (detailed information can be found in section 2.8).

2.5 Non-differentiable Equation Approach

A successful algorithm for the general NCP is the Josephy-Newton [24, 25] method which is based on Robinson's generalized equation approach for analyzing problems such as the NCP. The basic idea is to linearize the function $F(x)$ around the current iterate x^k , and generate the next iterate x^{k+1} by solving the following linear complementarity problem (LCP):

$$\begin{aligned} F(x^k) + \nabla F(x^k)(x - x^k) &\geq 0, \quad x \geq 0, \\ [F(x^k) + \nabla F(x^k)(x - x^k)]^T x &= 0 \end{aligned}$$

The quadratic convergence rate of Newton's method is one of the most important features of this approach. However, the convergence theory for this method requires that one start close to the solution x^* . To overcome this problem, Mathiesen [46, 47] and Preckel [53] have introduced various heuristic line search procedures to widen the region of convergence.

The $\text{NCP}(F)$ can easily be formulated as a system of nonlinear equations as follows:

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

where the "min" operator is taken component-wise. While this system of equations is simple, it is not differentiable [50] and thus, the traditional Newton method does not apply. It is, however, B-differentiable. Robinson [55, 56, 57, 58] first studied B-differentiable functions and Newton's method for a class of such nonsmooth functions. Pang [51] presents a generalized Newton algorithm for the above system of B-differentiable equations which is globally convergent under certain assumptions. Harker and Pang [20] then used these results to develop a damped-Newton method for the linear complementarity problem, and showed through extensive computational experiments that this method is potentially very efficient as compared with Lemke's method.

Harker and Xiao [21] extended the work by Harker and Pang to the nonlinear complementarity problem. Instead of using "min" operator, Harker

and Xiao converted the nonlinear complementarity problem into a system of equations through the use of a Minty-map [30, 57, 58]. The algorithm presented was similar to Kostreva's direct block pivotal algorithm, except that they solved a linear system of equations at each iteration and performed a line search to minimize a given merit function. This merit function minimization is what generates the global convergence properties of the algorithm in much the same way as in solving F-differentiable equations [26]. Minty-map formulation [30, 57, 58]:

$$H(x) = F(x^+) + x^-,$$

where $x_i^+ = \max(x_i, 0)$, $x_i^- = \min(x_i, 0)$, $x^+ = (x_1^+, x_2^+, \dots, x_n^+)^T$ and $x^- = (x_1^-, x_2^-, \dots, x_n^-)^T$. It is easy to verify that there is a one-to-one correspondence between a solution of $\text{NCP}(F)$ and a solution of the system of equations

$$H(x) = F(x^+) + x^- = 0$$

In other words, x solves this equation if and only if x^+ solves the $\text{NCP}(F)$. Hence, the nonlinear complementarity problem can be formulated as a system of nonlinear equations different from first Minty map formulation [21]. Harker and Xiao used B-differentiable functions in their algorithm [21].

Definition 2.6 *A function $H : R^n \rightarrow R^n$ is said to be B-differentiable at the point x if H is Lipschitz continuous in a neighborhood of x and there exists a positive homogeneous function $\mathbf{BH}(x) : R^n \rightarrow R^n$, called the B-derivative of H at x such that*

$$\lim_{v \rightarrow 0} \frac{H(x+v) - H(x) - \mathbf{BH}(x)v}{\|v\|} = 0.$$

H is said to be B-differentiable in a set X if H is B-differentiable at all points $x \in X$.

2.5.1 The Damped-Newton algorithm

The damped-Newton algorithm [21] which is used to solve the above equation system (minty-map formulation) is stated as follows. Let $x^0 \in R^n$ be an arbitrary initial vector, let s, μ and σ be given scalars with $s > 0$, $\mu \in (0, 1)$

and $\sigma \in (0, \frac{1}{2})$. In general, given x^k with $\|H(x^k)\| > \varepsilon > 0$ where ε is the convergence tolerance, we generate x^{k+1} by performing the following two steps:
Step1. Solve the following Newton equation for the direction $d^k \in R^n$:

$$H(x^k) + \mathbf{B}H(x^k)d^k = 0$$

Step2. Let $\lambda_k = \mu^{m_k}s$ where m_k is the smallest nonnegative integer m which satisfies the following Armijo condition:

$$g(x^k) - g(x^k + \mu^m s d^k) \geq 2\sigma \mu^m s g(x^k).$$

where $g(x) = \frac{1}{2}H(x)^T H(x) = \|H(x)\|^2$. Set $x^{k+1} = x^k + \lambda_k d^k$. This procedure continues until $\|H(x^k)\| \leq \varepsilon$.

Pang [51] derived local and global convergence results for the damped-Newton method applied to a system of B-differentiable equations.

Although this algorithm doesn't use smoothing methods, its structure is similar to our algorithm [2]. Therefore it is included here to give a background to the readers.

2.6 Continuation Methods

Continuation methods study the relationship between a problem P , called the original problem, and the perturbed problems $P(\varepsilon)$ associated with P . In this context, ε is the perturbation parameter, which equals to zero for the original problem P . The content of continuation methods is rich. On the one hand, perturbation and parametric analysis discuss the behavior of the problem $P(\varepsilon)$ based on information of P , depending on whether ε is small or large, respectively. On the other hand, penalty related methods, proximal point algorithms and homotopy methods aim at finding a solution of P by successively solving the perturbed problems $P(\varepsilon)$.

Now, we will mention a continuation method which is developed by Kanzow [27]. This study is one of the recent studies and it is similar to previous Lemke's algorithm and our algorithm. Therefore we thought that it is suitable to give Kanzow's algorithm [27].

2.6.1 Interior-point continuation method for complementarity problems with uniform P -functions

This is based on [27]. Let $\mu \geq 0$ be given. The main tool used in this method is the function $\varphi_\mu : R^2 \rightarrow R$ defined by

$$\varphi_\mu(a, b) = a + b - \sqrt{(a - b)^2 + 4\mu}.$$

In the special case $\mu = 0$, $\varphi_\mu = \varphi_0$ reduces to

$$\varphi_0(a, b) = a + b - \text{sqrt}(a - b)^2 = a + b - |a - b| = 2 \min\{a, b\}.$$

The function $\min\{a, b\}$ has been used, e.g., by Pang [51] in order to characterize problem $\text{NCP}(F)$. Here, the function φ_μ is used to characterize problem $\text{PNCP}(F, \mu)$. For this, Kanzow defined the nonlinear operator $F_{\varphi_\mu} : R^{2n} \rightarrow R^{2n}$ by

$$F_{\varphi_\mu}(z) = F_{\varphi_\mu}(x, y) = \begin{pmatrix} F(x) - y \\ \varphi_\mu(x, y) \end{pmatrix}$$

where

$$\varphi_\mu(x, y) = (\varphi_\mu(x_1, y_1), \dots, \varphi_\mu(x_n, y_n))^T \in R^n.$$

Kanzow stated a theorem saying that a vector $z(\mu) = (x(\mu), y(\mu)) \in R^{2n}$ solves the perturbed nonlinear complementarity problem $\text{PCNP}(F, \mu)$ if and only if $z(\mu)$ solves the nonlinear system of equations $F_{\varphi_\mu}(z) = 0$ where $\text{PNCP}(F, \mu)$ is to find a solution $(x(\mu), y(\mu)) \in R^{2n}$ of the following system:

$$x \geq 0, y \geq 0, x_i y_i = \mu, y = F(x) \quad i = (1, \dots, n)$$

If $\mu = 0$ the problem $\text{PNCP}(f, \mu)$ reduces to the nonlinear complementarity problem.

2.6.2 Kanzow's algorithm:

Step0. Let $z^0 = (x^0, y^0) \in R^{2n}, k = 0$ and $\{\mu_k\}_{k \in N}$ be any sequence such that $\mu_k > 0 (k \in N)$ and $\lim_{k \rightarrow \infty} \mu_k = 0$.

Step1. If $\|F_{\varphi_{\mu}}(z^k)\| = 0$, stop: z^k solves $\text{NCP}(F)$.

Step2. Use a damped Newton method to determine a solution $z^{k+1} = z(\mu_{k+1})$ of the nonlinear system of equations

$$F_{\varphi_{\mu_{k+1}}}(z) = 0$$

Step3. Set $k = k + 1$ and go to *Step1* [27].

2.7 Smoothing Methods

The complementary condition

$$0 \leq x \perp y \leq 0.$$

where x and y are vectors in R^n and the symbol $\perp$ denotes orthogonality, plays a fundamental role in mathematical programming. Many problems can be formulated by using this complementarity condition. For example, most optimality conditions of mathematical programming [49] as well as variational inequalities [59] and extended complementarity problems [45, 18, 67] can be so formulated. It is obvious that the vectors x and y satisfy complementarity condition if and only if

$$x = (x - y)_+$$

where the plus function $(\cdot)_+$ is defined as

$$(\varepsilon)_+ = \max\{\varepsilon, 0\},$$

for a real number ε . For a vector $\mathbf{x}$, the vector $(\mathbf{x})_+$ denotes the plus function applied to each component of $\mathbf{x}$. In this sense, the plus function plays an important role in mathematical programming. But one big disadvantages of the plus function is that is not smooth because it is not differentiable. Thus numerical methods that use gradients cannot be directly applied to solve a problem involving a plus function. However, the smooth function approximation to plus function can solve this problem. With this approximation, many efficient algorithms, such as the Newton method, can be

employed.

Smoothing techniques have already been applied to different problems, such as, l_1 -minimization problems [41], multi-commodity flow problems [52], nonsmooth programming [68, 33], linear and convex inequalities [10], and linear complementarity problems [6], [10] and [27]. These successful techniques motivate a systematic study of the smooth approach.

2.7.1 A class of smoothing functions

In this section we review a class of smooth approximations to the fundamental function $(x)_+ = \max\{x, 0\}$ [11]. Notice first that $(x)_+ = \int_{-\infty}^x \sigma(y)dy$, where $\sigma(x)$ is the step function:

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

The step function $\sigma(x)$ can in turn be written as, $\sigma(x) = \int_{-\infty}^x \delta(y)dy$, where $\delta(x)$ is the Dirac delta function which satisfies the following properties

$$\delta(x) \geq 0, \quad \int_{-\infty}^{+\infty} \delta(y)dy = 1$$

The fact that the plus function is obtained by twice integrating the Dirac delta function, prompts us to propose probability density functions as a means of smoothing the Dirac delta function and its integrals. Chen and Mangasarian [11] considered the piecewise continuous function $d(x)$ with finite number of pieces which is a density function. That is, it satisfies

$$d(x) \geq 0 \text{ and } \int_{-\infty}^{\infty} d(x)dx = 1.$$

To parametrize the density function Chen and Mangasarian defined

$$\hat{t}(x, \beta) = \frac{1}{\beta} d\left(\frac{x}{\beta}\right)$$

where β is a positive parameter. When β goes to 0, the limit of $\hat{t}(x, \beta)$ is the Dirac delta function $\delta(x)$. This motivates a class of smooth approximations as

follows:

$$\hat{s}(x, \beta) = \int_{-\infty}^x \hat{t}(t, \beta) dt \approx \sigma(x)$$

and

$$\hat{p}(x, \beta) = \int_{-\infty}^x \hat{s}(t, \beta) dt \approx (x)_+$$

Therefore, Chen and Mangasarian got an approximate plus function by twice integrating a density function. In fact, this is the same as defining

$$\hat{p}(x, \beta) = \int_{-\infty}^x (x - t) \hat{t}(t, \beta) dt.$$

Now, let's look at some smooth plus function examples:

Example 2.1 *Neural Networks Smooth Plus Function [10]: Let*

$$d(x) \frac{e^{-x}}{(1 + e^{-x})^2}$$

Here $D_1 = \log 2$, $D_2 = 0$ and $\text{supp}d(x) = R$, where

$$D_1 = \int_{-\infty}^0 |x| d(x) dx$$

and

$$D_2 = \max\left\{\int_{-\infty}^{+\infty} x d(x) dx, 0\right\}$$

Integrating $\alpha = \frac{1}{\beta}$, we have

$$p(x, \alpha) = \hat{p}(x, \frac{1}{\alpha}) = \int s(\xi, \alpha) d\xi = x + \frac{1}{\alpha} \log(1 + e^{-\alpha x})$$

$$s(x, \alpha) = \hat{s}(x, \frac{1}{\alpha}) = \frac{1}{1 + e^{-\alpha x}} = \int t(\xi, \alpha) d\xi$$

$$t(x, \alpha) = \hat{t}(x, \frac{1}{\alpha}) = \frac{\alpha e^{-\alpha x}}{(1 + e^{-\alpha x})^2} = \alpha s(x, \alpha)(1 - s(x, \alpha)).$$

Example 2.2 *Chen-Harker-Kanzow-Smale Smooth Plus Function [60], [27], and [6]:*

Let

$$d(x) = \frac{2}{(x^2 + 4)^{\frac{3}{2}}}$$

Here $D_1 = 1$, $D_2 = 0$, $\text{supp}\{d(x)\} = R$ and

$$\hat{p}(x, \beta) = \frac{x + \sqrt{x^2 + 4\beta^2}}{2}$$

Example 2.3 *Pinar-Zenios Smooth Plus Function [52]:*

Let

$$d(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Here $D_1, D_2 = \frac{1}{2}$, $\text{supp}\{d(x)\} = [0, 1]$ and

$$\hat{p}(x, \beta) = \begin{cases} 0 & \text{if } x < 0 \\ \frac{x^2}{2\beta} & \text{if } 0 \leq x \leq \beta \\ x - \frac{\beta}{2} & \text{if } x > \beta \end{cases}$$

Example 2.4 *Zang Smooth Plus Function [68]:*

let

$$d(x) = \begin{cases} 1 & \text{if } -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Here $D_1 = \frac{1}{8}$, $D_2 = 0$, $\text{supp}\{d(x)\} = [-\frac{1}{2}, \frac{1}{2}]$ and

$$\hat{p}(x, \beta) = \begin{cases} 0 & \text{if } x < -\frac{\beta}{2} \\ \frac{1}{2\beta}(x + \frac{\beta}{2})^2 & \text{if } |x| \leq \frac{\beta}{2} \\ x & \text{if } x > \frac{\beta}{2} \end{cases}$$

2.7.2 Application to The Nonlinear Complementarity Problem

Recall that NCP is the problem of finding an x in R^n such that

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

Here $F(x)$ is a differentiable function from R^n to R^n . By using the smooth function $\hat{p}(x, \beta)$ introduced before, smooth nonlinear equation

$$R(x) = x - \hat{p}(x - F(x), \beta) = 0$$

is proposed as an approximation to the following nonsmooth equivalent reformulation of the NCP

$$x = (x - F(x))_+$$

And now we can specify Chen and Mangasarian's algorithm [11]. The algorithm consists of a Newton method with an Armijo line search with parameters δ and σ such that $0 < \delta < 1$ and $0 < \sigma < \frac{1}{2}$.

Newton NCP Algorithm: [11]

Given $x_0 \in R^n$ and let $k = 0$.

(1) **Direction** d_k

$$d_k = -\nabla R(x_k)^{-1} R(x_k)$$

(2) **Stepsize** λ_k (Armijo)

$$x_{k+1} = x_k + \lambda_k d_k, \lambda_k = \max\{1, \delta, \delta^2, \dots\}, s.t.$$

$$f(x_k) - f(x_{k+1}) \geq \sigma \lambda_k |d_k^T \nabla f(x_k)|$$

$k = k + 1$ go to step(1).

The above algorithm is well defined for a monotone NCP with a continuously differentiable $F(x)$.

2.7.3 Mixed Complementarity Problem

The mixed complementarity problem (MCP) is defined as follows [14]: Given a differentiable $F : R^n \rightarrow R^n$, $l, u \in \bar{R}^n$, $l < u$, where $\bar{R} = R \cup \{+\infty, -\infty\}$, find $x, w, v \in R^n$, such that

$$F(x) - w + v = 0$$

$$0 \leq x - l \perp w \geq 0$$

$$0 \leq v \perp u - x \geq 0$$

This MCP model includes many classes of mathematical programming problems, such as nonlinear equations, nonlinear programming, nonlinear complementarity problems and variational inequalities.

By using the smooth function $\hat{p}(x, \beta)$ instead of the plus function, Chen and Mangasarian reformulated the MCP approximately as follows [11]. For $i = 1, \dots, n$:

Case 1. $l_i = -\infty$ and $u_i = \infty$:

$$F_i(x) = 0$$

Case 2. $l_i > -\infty$ and $u_i = \infty$:

$$x_i - l_i - \hat{p}(x_i - l_i - F_i(x), \beta) = 0$$

Case 3. $l_i = -\infty$ and $u_i < \infty$:

$$x_i - u_i + \hat{p}(u_i - x_i + F_i(x), \beta) = 0$$

Case 4. l_i and $u_i < \infty$:

$$F_i(x) - w_i + v_i = 0$$

$$x_i - l_i - \hat{p}(x_i - l_i - w_i, \beta) = 0$$

$$u_i - x_i - \hat{p}(u_i - x_i - v_i, \beta) = 0.$$

Chen and Mangasarian denoted the above 4 cases collectively by the nonlinear equation

$$R(x, w, v) = 0$$

Note that the natural residual for the MCP is given by the left hand side of above relation with the $\hat{p}$ function replaced by the plus function.

Let $f(x, w, v)$ be the residual function of the nonlinear equation defined as follows

$$f(x, w, v) = \frac{1}{2} R(x, w, v)^T R(x, w, v)$$

Chen and Mangasarian's Smooth Algorithm for MCP [11]:

Input tolerance ϵ , parameter $\nu > 1$ and initial guess $x_0 \in R^n$

(1) **Initialization** For $1 \leq i \leq n$ of Case 4, let $w_0^i = (F_i(x_0))_+$, $v_0^i = (-F_i(x_0))_+$, $k = 0$ and $\alpha_0 = \alpha(x_0, w_0, v_0)$. Choose $\alpha_{max} = \frac{2}{\epsilon}$

(2) **Newton Armijo Step** Find $(x_{k+1}, w_{k+1}, v_{k+1})$ by a Newton-Armijo step applied to

$$R(x, w, v) = 0.$$

(3) **Parameter Update** If $\alpha(x_{k+1}, w_{k+1}, v_{k+1}) \geq \nu\alpha_k$, set

$$\alpha_{k+1} = \alpha(x_{k+1}, w_{k+1}, v_{k+1}),$$

otherwise if $\|\nabla f(x_{k+1}, w_{k+1}, v_{k+1})\|_2 \leq \epsilon$, set

$$\alpha_{k+1} = \nu\alpha_k$$

If $\alpha_{k+1} > \alpha_{max}$, set $\alpha_{k+1} = \alpha_{max}$. Let $k = k + 1$, go to step(2).

2.8 Non-interior Continuation Methods for the NCP

Finally, let's look at non-interior continuation methods. Non-interior continuation methods are closely related to path-following interior point algorithms. Both methods deform the complementarity condition by a parameterized systems of smooth nonlinear equations, then solve the deformed NCP by Newton's method approximately, and adjust the parameter to refine the deformation. Both feasible and infeasible interior point path following algorithms have been developed to solve linear complementarity problems (LCPs) and NCPs (see for example [36, 37, 38, 35, 64, 65]). Wright and Ralph [65] proposed to alternate between the Newton step for the NCP and the centering step for the deformed NCP to achieve global and local superlinear convergence. However, no global convergence rate was given for their algorithm. Tseng [64] took a different approach by choosing certain combination of the above two steps as a search direction at each iteration. For the first time, he showed both global linear convergence and local superlinear and quadratic convergence for monotone NCPs with some additional assumptions. All interior point algorithms, however, share a common feature: they require each intermediate iterates to stay interior (positive).

Non-interior continuation methods take a different deformation of the complementarity condition. As a result, they did not have to restrict intermediate iterates to stay interior. The first non-interior method was introduced by Chen and Harker [6], where the authors concentrated on establishing the structural properties of the central path for LCPs with P_0 and R_0 matrices. The method was later improved by Kanzow [27], where the author refined the smooth function and established the convergence for the continuation method under similar assumptions. However, both methods lack a systematic procedure to reduce the continuation (or smooth) parameter to zero, even though they have shown impressive numerical performance

[6, 27] compared with interior point algorithms. As a result, no rate of convergence results were obtained. This gap was closed recently by Burke and Xu [3]. Inspired by many path following interior point algorithms, the authors introduced a notion of neighborhood around the central path for their non-interior continuation methods. All intermediate iterates are required to stay within the neighborhood and this provides a systematic procedure to reduce the smooth parameter. This important addition to the continuation methods allowed them to establish the global linear convergence for both LCPs with P_0 and R_0 matrices [3] and NCPs with uniform P functions [3]. In addition, their computational experiments have shown further improvement over previous non-interior continuation methods have been developed to solve linear and quadratic programs [7], complementarity problems [28], and variational inequalities [8, 29].

All non-interior continuation methods mentioned above are based on smooth functions derived from $x_i y_i = \mu$, the deformed complementarity condition used for interior point algorithms. Many other smooth functions exist. Indeed, Chen and Mangasarian [4] have proposed a broad class of smooth functions for the plus function $z_+ = \max\{z, 0\}$ as mentioned above.

The non-interior continuation methods are also closely related to a broader class of algorithms called smoothing methods, which have attracted much attention recently. In particular, Gabriel and Moré further generalized the Chen-Mangasarian smooth function family and applied their smooth functions to mixed complementarity problems [16]. Chen, Qi and Sun [66] designed a smooth Newton method to solve a system of non-smooth equations and showed global and locally superlinear convergence for their method. Their results are based on a even broader class of smooth functions than the Gabriel-More family. The method was then applied to solve general box constrained variational inequalities. More recently, Chen [12] developed a smoothing quasi-Newton method for non-smooth equations and established superlinear convergence for the algorithm.

2.8.1 A Continuation Method for NCP

A recent contribution was made by Chen and Xiu [9], which gave a globally linearly and locally quadratically convergent algorithm. This recent non-interior method is conceptually important and therefore we put it here.

Recall that NCP can be written as:

$$\min\{x, y\} = 0 \text{ or } x - (x - y)_+ = 0,$$

where the plus function is taken component-wise. By deforming the plus function with the Chen-Mangasarian smooth function p_μ , we obtain the following smoothed complementarity condition:

$$\Psi_\mu(x, y) = x - P_\mu(x - y) = 0,$$

where $\mu \geq 0$ is a smooth parameter, $P_\mu(x - y) = \text{vec}\{p_\mu(x_i - y_i)\}$, and $\Psi_\mu(x, y) = \text{vec}\{\psi_\mu(x_i, y_i)\}$.

The smoothed NCP then becomes:

$$H_\mu(x, y) = \begin{bmatrix} F(x) - y \\ \Psi_\mu(x, y) \end{bmatrix} = 0,$$

and the NCP conditions (1)-(2) can be written as $H_0(x, y) = 0$ [9]. The idea behind continuation methods is to solve the smoothed NCP $H_\mu(x, y) = 0$ “approximately” for each given smooth parameter $\mu > 0$ and gradually reduce μ to zero. Hopefully, as μ approaches zero, the solution of the smoothed NCP approaches a solution of the NCP.

Chen and Xiu [9] mentioned importance of the structure of the Jacobian matrix $\nabla H_\mu(z)$ for the convergence analysis:

Denote $P'_\mu(z) = \text{diag}\{p'_\mu(z_i)\}$. By definition,

$$\nabla_x \Psi_\mu(x, y) = I - P'_\mu(x - y), \text{ and } \nabla_y \Psi_\mu(x, y) = P'_\mu(x - y).$$

and

$$I - P'_\mu(x - y) = P'_\mu(y - x), \quad 0 < P'_\mu(x - y) < I, \quad 0 < P'_\mu(y - x) < I.$$

Thus, the Jacobian matrix can be written as

$$\nabla H_\mu(x, y) = \begin{bmatrix} \nabla F(x) & -I \\ P'_\mu(y - x) & P'_\mu(x - y) \end{bmatrix}$$

It is well known that the Jacobian $\mu(x, y)$, due to its special structure, is nonsingular if and only if matrix $P'_\mu(y - x) + P'_\mu(x - y)(x)$ is nonsingular.

Merit function for $H_\mu(x, y)$:

$$\rho_\mu(x, y) = \|F(x) - y\| + \|\Psi_\mu(x, y)\|.$$

Let the central path(s) of the NCP be the set of solutions of $H_\mu(x, y) = 0$ for all $\mu > 0$. To construct an implementable continuation method for the NCP, Chen and Xiu introduced a neighborhood around the central path:

$$N(\beta) = \{(x, y) : \rho_\mu(x, y) \leq \beta\mu, \mu > 0\},$$

where parameter $\beta > 0$ is called the width of the neighborhood. In addition, they defined the slice of neighborhood with $\mu \in U$ as

$$N(\beta, U) = \{(x, y) : \rho_\mu(x, y) \leq \beta\mu, \mu \in U\}.$$

For simplicity, if $U = \mu$, they wrote the slice as $N(\beta, \mu)$.

Chen and Xiu's Continuation Algorithm [9]:

Given $\sigma \in (0, 1)$, and $\alpha_i \in (0, 1)$, for $i = 1, 2, 3$.

Step 0 (Initialization)

Set $k = 0$. Choose $\mu_0 > 0$, $(x^0, y^0) \in R^{2n}$, and $\beta > nB$ such that $(x^0, y^0) \in N(\beta, \mu_0)$.

Step 1 (Calculate Centering Step)

If $H_0(x^k, y^k) = 0$, stop. (x^k, y^k) is a solution of the NCP; otherwise, if $\mu_k(x^k, y^k)$ is singular, stop. The continuation method fails; otherwise, let $(\Delta\tilde{x}^k, \Delta\tilde{y}^k)$ solve the equation

$$H_{\mu_k}(x^k, y^k) + \mu_k(x^k, y^k)^T(\Delta x, \Delta y)^T = 0.$$

Step 2 (Line Search For Centering Step)

If $\rho_{\mu_k}(x^k, y^k) = 0$, set $(\tilde{x}^{k+1}, \tilde{y}^{k+1}) = (x^k, y^k)$; otherwise, let λ_k be the maximum of the values $1, \alpha_1, \alpha_1^2, \dots$ such that

$$\rho_{\mu_k}(x^k + \lambda_k \Delta \tilde{x}^k, y^k + \lambda_k \Delta \tilde{y}^k) \leq (1 - \sigma \lambda_k) \rho_{\mu_k}(x^k, y^k).$$

Set $(\tilde{x}^{k+1}, \tilde{y}^{k+1}) = (x^k, y^k) + \lambda_k (\Delta \tilde{x}^k, \Delta \tilde{y}^k)$.

Step 3 (μ Reduction Based on Centering Step)

Let γ_k be the maximum of the values $1, \alpha_1, \alpha_2^2, \dots$ such that

$$(\tilde{x}^{k+1}, \tilde{y}^{k+1}) \in N(\beta, (1 - \gamma_k) \mu_k).$$

Set $\tilde{\mu}_{k+1} = (1 - \gamma_k) \mu_k$.

Step 4 (Calculate Approximate Newton Step)

Let $(\Delta \tilde{x}^k, \Delta \tilde{y}^k)$ solve the equation

$$H_0(x^k, y^k) +_{\mu_k} (x^k, y^k)^T (\Delta x, \Delta y)^T = 0.$$

Set $(\hat{x}^{k+1}, \hat{y}^{k+1}) = (x^k, y^k) + (\Delta \tilde{x}^k, \Delta \tilde{y}^k)$.

Step 5 (μ Reduction Based on Approximate Newton Step)

If $(\tilde{x}^{k+1}, \tilde{y}^{k+1}) \notin N(\beta, \tilde{\mu}_{k+1})$, set

$$\mu_{k+1} = \tilde{\mu}_{k+1}, \quad (x^{k+1}, y^{k+1}) = (\tilde{x}^{k+1}, \tilde{y}^{k+1}),$$

and $k = k + 1$, Return to Step 1. Otherwise, let η_k be the maximum of the values $1, \alpha_3, \alpha_3^2, \dots$ such that

$$(\tilde{x}^{k+1}, \tilde{y}^{k+1}) \in N(\beta, \eta_k \tilde{\mu}_{k+1}).$$

Set

$$\mu_{k+1} = \eta_k \tilde{\mu}_{k+1}, \quad (x^{k+1}, y^{k+1}) = (\tilde{x}^{k+1}, \tilde{y}^{k+1}),$$

and $k = k + 1$. Return to Step 1.

It is very easy to initialize the above continuation method. One may simply choose any $\mu_0 > 0$, $(x^0, y^0) \in R^{2n}$, and $\beta > \max\{\rho_{\mu_0}(x^0, y^0)/\mu_0, nB\}$.

No computational experience is available for this method yet. This can be an interesting line of future work.

Chapter 3

SMOOTH APPROXIMATION FOR NORMAL MAP FORMULATION OF NCP

In the present chapter we introduce the normal map formulation of the NCP and apply smoothing techniques to it. The material of the present chapter and Chapter 4 are from [2].

Given a mapping $f : R^n \rightarrow R^n$, the NCP with respect to f , $\text{NCP}[f]$, finds an $x \in R^n$ such that

$$x \geq 0, \quad f(x) \geq 0, \quad \text{and} \quad x^T f(x) = 0$$

It is well known that $\text{NCP}[f]$ can be reformulated as a system of nonsmooth equations by using either min map or normal map. In either reformulation, the plus function $z_+ = \max\{0, z\}$ is involved, where the max is taken component-wise. The nonsmooth equation of the min formulation is given by

$$\min\{x, f(x)\} = x - (x - f(x))_+ = x$$

and x solves $\text{NCP}[f]$ if and only if it solves the above equation.

Chen and Mangasarian [4] proposed a class of parametric smooth functions, called plus-smooth functions, to approximate the plus function (detailed information can be found in literature review). The plus-smooth function is obtained by twice integrating a parameterized probability distribution function d . With the help of the plus-smooth function, the nonsmooth min reformulation is approximated as a system of smooth nonlinear equations. An approximate solution can be obtained by solving these equations. Their numerical experiment indicates that the smoothing approach is very effective and efficient. Chen and Harker [8] refined the plus-smooth function proposed by Chen and Mangasarian and the refinements allowed them to establish the existence, uniqueness, and some trajectory properties of parametric approximations to the NCP's.

The normal map reformulation relates $\text{NCP}[f]$ to the following system of nonsmooth equations, called Normal Map Equation (NME):

$$f(z_+) + z_- = 0$$

where $z_- = \min\{0, z\}$. It is well known that $x = z_+$ solves $\text{NCP}[f]$ if z solves the NME and $z = x - f(x)$ solves the NME if x solves $\text{NCP}[f]$. Unlike the min reformulation, the normal map reformulation only requires that f be defined on R_+^n instead of R^n .

Chen, Harker and Pınar [2] applied the plus-smooth to approximate both z_+ and z_- in the NME and proposes a continuation method to solve $\text{NCP}[f]$. This chapter studies the properties of the plus-smooth function based on [2]. Unlike previous papers, Chen, Harker and Pınar [2] classified the smooth function by whether it is derived from a density function with finite or infinite

support. Section 3.1 analyzes the smooth approximation to the NME. Sufficient conditions are provided to guarantee the boundedness and the monotonicity of the solution trajectory for the continuation method, respectively. Next chapter gives the structure of the subproblems of a Newton corrector based continuation method which was investigated in [2]. It is shown that the smooth function with finite support results in subproblems of reduced dimension, an advantage shared by many B-differentiable approaches to solve complementarity related problems (see [21] for example). Chapter 6 reports our numerical experiments of the continuation method using smoothing functions with finite and infinite support.

Many complementarity related problems can be reformulated as a system of equations or an optimization problem. The plus function z_+ is involved in many of these reformulations. However, since the plus function is nonsmooth, many of the resulting equations or optimization problems are nonsmooth. They cannot be solved directly by the traditional techniques for smooth problems. To overcome the difficulty, Chen and Mangasarian [8] introduced a class of plus-smooth function $p(z, u)$ that novelly approximates the fundamental plus function z_+ by twice integrating a probability density function with parameter $0 \leq u < \infty$. More specifically, the plus-smooth function is given by

$$p(z, u) = \int_{-\infty}^z \int_{-\infty}^t \frac{1}{u} d\left(\frac{x}{u}\right) dx dt$$

where $d(x)$ is a probability density function satisfies certain assumptions. Clearly, as u approaches zero, the probability density $\frac{1}{u}d(\frac{x}{u})$ approaches the delta function with all the masses concentrated at origin and the double integration $p(z, u)$ approaches the plus function z_+ . In this regard, $p(z, u)$ can be considered as a natural approximation of the plus function z . Indeed, it has been shown by Chen and Harker [8] that any "well" behaved smooth approximation of the plus function must be a double integration of a probability density function.

The smooth approximation $p(z, u)$ preserves many structural properties of

the plus function z_+ , which will be explored next. Chen, Harker and Pinar's characterization is based on whether the plus-smooth function p has a finite or an infinite support. p is said to have a finite (an infinite) support if the probability density function d it derives from has a finite (an infinite) support. A probability density function d is supported on range $[a, b]$ if $d(x) > 0$ for all $x \in [a, b]$ and $d(x) = 0$ otherwise. If both a and b are finite numbers, we called d has a finite support; otherwise, d has an infinite support. Some of the following assumptions on the probability density function d will be used to characterize the plus-smooth function p :

- (C1) $d(x)$ is symmetric and piecewise continuous with finite number of pieces.
- (C2) $E(|x|) < \infty$.
- (C3) $\lim_{x \rightarrow \infty} x^3 d(x) < \infty$.
- (C4) $d(x)$ has a finite support on $[-s, s]$ for some $0 < s < \infty$.
- (C5) $d(x)$ has an infinite support.

The symmetric assumptions in (C1) is made only for the convenience of presentation. After all, all plus-smooth functions proposed and implemented so far are derived from symmetric probability density functions. Assumptions (C2) ensures that the integration of d exists. Assumptions (C3) requires that both tails of d to be thin enough, a property to be used to establish the boundedness of solution later. Clearly, assumption (C4) implies (C3), which in turn implies (C2).

The following results characterizes the plus-smooth function $p(z, u)$ under various assumptions of the probability density function d :

Proposition 3.1 *Let $p(z, u)$ be defined above with $u > 0$ and the probability density function d satisfying assumptions (C1) and (C2).*

1. $p(z, u)$ is continuously differentiable, nondecreasing, and convex.
2. $\lim_{u \rightarrow -0} p(z, u) = p(z, 0) = z_+$ for all z .
3. $\lim_{z \rightarrow -\infty} p(z, u) = 0$ and $\lim_{z \rightarrow \infty} p(z, u) = z$ for all $u > 0$.
4. $0 \leq p'(z, u) \leq 1$ and $p'(-z, u) = 1 - p'(z, u)$.

5. $0 \leq P(z, u) - (z)_+ \leq Du$ for all z , where $D > 0$ is a fixed constant depending on d .
6. Equation $p(z, u) = b$ has a unique solution for all $u \geq 0$ and $b > 0$. 7. If in addition d satisfies (C4), then $p(z, u)$ is strictly increasing and convex, $0 < p'(z, u) < 1$, and $z_+ < p(z, u)$.
8. If in addition d satisfies (C4), then $p(z, u)$ is strictly increasing and convex in $(-su, su)$, $z_+ < p(z, u)$ and $0 < p'(z, u) < 1$ for all $z \in (-su, su)$, and $p(z, u) = z_+$ otherwise.
9. If in addition d satisfies (C3), then $p(z, u)p(-z, u) < \infty$ for all z [2].

Proof. Results (1),(5), and the first two parts of result (6) have been shown in [11] and results (3) and the last part of result (6) have been shown in [8]. Results (2),(4), and the last part of result (8) are true by definition of $p(z, u)$ and the fact that $\frac{1}{u}d(\frac{x}{u})$ is a probability density function. To show result (6), suppose on the contrary that the equation $p(z, u) = b$ has two solutions $z_1 < z_2$. By result (3), there exists a $z_0 < z_1$ such that $p(z_0, u) = b_0 < b$. Clearly, $p(z_1, u)$ is strictly greater than the linear interpolation of $p(z_0, u)$ is a convex function. To show the first two parts of results (8), notice that if the probability density function $d(x)$ is supported on $[-s, s]$ then the functional $\frac{1}{u}d(\frac{x}{u})$ has a support on $[-us, us]$. The remaining proof is almost identical to that of result (7). It remains to show result (9). Since d is assumed to be symmetric, it suffices to show that $\lim_{z \rightarrow \infty} p(z, u)p(-z, u) < \infty$. Indeed,

$$\begin{aligned}
 &= \lim_{z \rightarrow \infty} \frac{p(-z, u)}{\frac{1}{p(z, u)}} \\
 &= \lim_{z \rightarrow \infty} \frac{\int_{-z}^{-\infty} \frac{1}{u} d(\frac{x}{u}) dx}{\int_{-\infty}^z \frac{1}{u} d(\frac{x}{u}) dx} p^2(z, u) \\
 \lim_{z \rightarrow \infty} p(z, u)p(-z, u) &= \lim_{z \rightarrow \infty} z^2 \int_{-z}^{-\infty} \frac{1}{u} d(\frac{x}{u}) dx \\
 &= \lim_{z \rightarrow \infty} u^{-2} z^3 d(\frac{-z}{u}) \\
 &= \lim_{z \rightarrow \infty} u x^3 d(x) \\
 &< \infty
 \end{aligned}$$

Here the l'Hospital's rule is used to obtain the second and the fourth equalities. The third equality is true since d is a probability density function and $\lim_{z \rightarrow \infty} p(z, u) = z$ by result (2) of Proposition 3.1. $\square$

Below are several examples of the plus-smooth function derived from a probability function with an infinite support. In all examples, the function d satisfies assumptions (C1),(C2),(C3), and (C5).

Example 1. Neural network plus-smoothing function (Chen and Mangasarian [11]):

$$d(x) = \frac{e^{-x}}{(1 + e^{-x})^2}, \quad p(z, u) = z + u \log(1 + e^{-\frac{z}{u}}).$$

Example 2. Interior point plus-smooth function (Smale [60]):

$$d(x) = \frac{2}{(x^2 + 4)^{1.5}}, \quad p(z, u) = \frac{z + \sqrt{z^2 + 4u}}{2}.$$

Notice that the above probability density function d is a scaled variant of the t distribution with parameter $n = 2$. It has been shown [60] that the central path of many interior point algorithms can be characterized as solution of the parametric smooth equations obtained by applying this plus-smooth function to the min reformulation $\text{NCP}[f]$ (1).

Example 3. Normal plus-smooth function:

$$d(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

No close form expression for $p(z, u)$.

Although it is possible to derive a plus-smooth function from many probability density functions with finite support, the following plus-smooth function with finite support seems to be a natural choice. It satisfies assumptions (C1), (C2), (C3), and (C4).

Example 4. Uniform plus-smooth function (Zang [68]):

$$d(x) = \begin{cases} 1 & \text{if } -1/2 \leq x \leq 1/2 \\ 0 & \text{otherwise} \end{cases} \quad p(z, u) = \begin{cases} 0 & \text{if } z < -u/2 \\ \frac{(z+u/2)^2}{2u} & \text{if } |z| \leq u/2 \\ z & \text{if } z > u/2 \end{cases}$$

3.1 Application to the Nonlinear Complementarity Problem

In this section, the plus-smooth functions introduced in the previous section are applied to the NME, the normal map reformulation of $\text{NCP}[f]$. As a result, the NME is approximated by a series of parametric smooth equations. The properties of the solution path(s), consisting of solutions of the smooth equations, are investigated. Sufficient conditions are provided to ensure the existence and the monotonicity of the solution path. Some of the results have been established in literature (see [36]), but only for a specific choice of plus-smooth functions, such as the interior point plus-smooth function. In the remaining study, the probability density function d that derives the plus-smooth function is assumed to satisfy at least assumptions (C1) and (C2).

As mentioned in the introduction, $x = z_+$ solves $\text{NCP}[f]$ if z solves the NME: $f(z_+) + z_- = 0$. Since z_+ and z_- can be approximated by $p(z, u)$ and $-p(z, u)$, respectively, the NME can be approximated by the following parametric equations, called Smooth Normal Map Equation (SNME):

$$H(z, u) = (1 - u)f(p(z, ua)) - p(-z, ua) + ub = 0,$$

where $a \in \mathbb{R}_+^n$ and $b \in \mathbb{R}_+^n$ are fixed parameters and $p(z, ua)$ and $p(-z, ua)$ are column vectors with components $p(z_i, ua_i)$ and $p(-z_i, ua_i)$, $i = 1, 2, \dots, n$, respectively. If the vector a is strictly positive, i.e., $a > 0$, $H(z, u)$ is C^1 by result (1) of Proposition 3.1. If some of the components of $a \geq 0$ are zeros, $H(z, u)$ is in general piecewise C^1 .

At $u = 1$, the SNME reduces to

$$H(z, 1) = -p(-z, ua) + b = 0$$

The equation has a unique solution by result (7) of Proposition 3.1. On the other hand, at $u = 0$, the SNME reduces to the NME

$$H(z, 0) = f(z_+) + z_- = 0$$

Therefore, if there exists a path from the unique solution at $u = 1$ to a solution at $u = 0$, we could apply standard homotopy techniques to find the solution of the NME and thus a solution of $\text{NCP}[f]$.

3.1.1 Existence of Solution Path

We start by defining S as the set of all solutions of the SNME, i.e.,

$$S = \{(z, u) \in \mathbb{R}^n * (0, 1] : H(z, u) = 0\},$$

and the solution path T as the connected components of S emanating from the unique solution of $H(z, 1) = 0$. The following result establishes the existence of a solution path from the unique starting point to a solution of the NME. It is a straight forward extension of a similar result established by Kojima et al. [36] for the interior point plus-smooth function:

Theorem 3.1 *Let $a \geq 0$ be fixed. Then for almost every $\epsilon > 0$ the set T forms a trajectory, a 1-dimensional manifold which is homeomorphic to $(0, 1]$, such that*

$$T = \{(\epsilon(t), \tau(t)) : 0 < t \leq 1\}$$

and $\lim_{t \rightarrow 0} \tau(t) = 0$ whenever T is bounded. Here, $\epsilon : (0, 1] \rightarrow \mathbb{R}^n$, $\tau(t) : (0, 1] \rightarrow (0, 1]$ are piecewise C^1 -mappings or C^1 -mappings when $a > 0$.

Proof. To prove the result, we are only interested in the set of solutions $(z, u) \in S$ with $0 < u \leq 1$. Hence, defining a piecewise C^1 -mapping $P : \mathbb{R}^n * (0, 1] \rightarrow \mathbb{R}^n$:

$$P(z, u) = [(1 - u)f(p(z, ua)) - p(-z, ua)]/u$$

we can rewrite the SNME as:

$$P(z, u) = -b.$$

Clearly, the mapping P is C^1 if $a > 0$ and piecewise C^1 if some of the components of $a \geq 0$ are zeros. Consequently, the theorem follows from the result on regular values of piecewise C^1 -mappings. Almost every $-b < 0$ is a regular value of the piecewise C^1 -mapping P .

And if $-b < 0$ is a regular value of the piecewise C^1 -mapping P then S is a disjoint union of smooth 1-dimensional manifolds: specifically its connected component T forms a piecewise smooth trajectory (or a smooth trajectory when $a > 0$) such that either $\|z\|$ tends to infinity or u tends to 0 along the trajectory T . $\square$

The significance of the above result has been discussed in [36]: The set T generically forms a smooth or piecewise smooth trajectory. Furthermore, if the trajectory T is bounded, there exists at least one limit point as u tends to 0 along the trajectory, and every limit point is a solution of the NME. Sufficient conditions to ensure the boundedness of the solution set S and therefore the trajectory T is discussed next.

3.1.2 Boundedness of Trajectory T

We show that the solution set S is bounded if f is a monotone function and $\text{NCP}[f]$ has a strictly feasible solution or f is an R_0 -function (to be defined).

Let D be a nonempty subset of R^n . A continuous mapping $f:D \rightarrow R^n$ is said to be *monotone* over D if

$$[f(x) - f(y)]^T(x - y) \geq 0 \text{ for all } x, y \in D$$

Proposition 3.2 *Let $a \geq 0$ and the plus-smooth function p satisfy assumptions (C3). If f is monotone over R_+^n and $NCP[f]$ has a strictly positive solution, then the solution set S , and therefore the trajectory T , is bounded.*

Proof. Let z be any point of set S for some $u \in (0, 1]$. We need to show that $\|z\|$ is bounded. Denote $x = p(z, ua) \geq 0$ and $y = p(-z, ua) \geq 0$. By result (5) of Proposition 3.1,

$$|z_i| = (z_i)_+ - (z_i)_- \leq p(z_i, ua_i) + p(-z_i, ua_i).$$

It suffices to show that $1^T x + 1^T y$ is bounded. By assumption there exist a point $(\tilde{x}, \tilde{y}) > 0$ such that $y = f(x)$. Define the positive numbers ϵ and ω by

$$\epsilon = \min\{b_i, \tilde{x}_i, \tilde{y}_i : i = 1, \dots, n\}, \quad \omega = \max\{b_i, \tilde{x}_i, \tilde{y}_i : i = 1, \dots, n\}.$$

It has been shown by Kojima *et al.* (page 953 of [36]) that if f is a monotone function then

$$1^T x + 1^T y \leq [x^T y + n\omega^2]/\epsilon.$$

Since $x^T y = p(z, ua)^T p(-z, ua)$ is bounded by assumption (C3) and result (9) of Proposition 3.1, $1^T x + 1^T y$ is bounded and result follows. $\square$

We now present another condition for S to be bounded. Let D be a nonempty subset of R^n . A continuous mapping $f: D \rightarrow R^n$ is a R_0 -function over the set D if for any sequence $\{x^k\}$ in D satisfying $\{\|x^k\|\} \rightarrow \infty$ and

$$\liminf_{k \rightarrow \infty} \frac{\min_i x_i^k}{\|x^k\|} \geq 0, \quad \liminf_{k \rightarrow \infty} \frac{\min_i f_i(x^k)}{\|x^k\|} \geq 0,$$

there exists an index j such that $\{x_j^k\} \rightarrow \infty$ and $\{f_j(x^k)\} \rightarrow \infty$.

Proposition 3.3 *Let $a \geq 0$. If f is a R_0 function over R_+^n , then the solution set S , and therefore the trajectory T , is bounded.*

Proof. Suppose on the contrary that S is unbounded. Then there exists an unbounded sequence $\{z^k\}$ such that $H(z^k, u^k) = 0$ with $u^k \in [0, 1]$. Denote

$$x^k = p(z^k, u^k a) \geq 0 \text{ and } y^k = p(-z^k, u^k a) \geq 0.$$

Then

$$(1 - u^k)f(x^k) - y^k + u^k b = 0.$$

We claim that sequence $\{x^k\}$ must also be unbounded. Otherwise, to satisfy last equation, sequence $\{y^k\}$ must be bounded and so is $\{z^k\}$ since $|z^k| \leq x_i^k + y_i^k$ from the proof of the previous result. However, this contradicts the assumption that sequence $\{z^k\}$ is unbounded. We now consider the limiting behavior of unbounded sequence $\{x^k\}$ and the associated sequence $\{f(x^k)\}$. For any index i ,

- If z_i^k is bounded, then y_i^k is bounded by definition, and $f_i(x^k)$ bounded by equation (5);
- If $z_i^k \rightarrow \infty$, then $x_i^k \rightarrow \infty$, $y_i^k \rightarrow 0$ by result (3) of Proposition 3.1, and $f_i(x^k)$ is bounded by equation (5);
- If $z_i^k \rightarrow -\infty$, then, by the same argument, $x_i^k \rightarrow 0$, $y_i^k \rightarrow \infty$, and $f_i(x^k) \rightarrow \infty$.

It follows that sequence $\{x^k\}$ satisfies the assumptions of R_0 -function:

$$\liminf_{k \rightarrow \infty} \frac{\min_i x_i^k}{\|x^k\|} \geq 0, \quad \liminf_{k \rightarrow \infty} \frac{\min_i f_i(x^k)}{\|x^k\|} \geq 0.$$

By definition of R_0 -function, there exists an index j such that $x_j^k \rightarrow \infty$ and $y_i^k = f_i(x^k) \rightarrow \infty$. However, $x_j^k \rightarrow \infty$ implies $z_j^k \rightarrow \infty$, $y_i^k \rightarrow -\infty$ implies $z_j^k \rightarrow -\infty$. This leads to a contradiction. $\square$

3.1.3 Existence of Monotone Trajectory T

In general, the trajectory T defined in the previous two subsections above is not necessarily monotone with respect to the parameter u . i.e., to follow the trajectory from the starting point at $u = 1$ to a solution at $u = 0$, u does not always decrease monotonically. More sophisticated techniques (see e.g. [1]) are needed to trace the trajectory. However, u is decreased monotonically in most implementations of smoothing methods and interior point methods for complementarity related problems. This subsection provides a set of sufficient conditions under which there exists a unique monotone trajectory. Needed definitions are in literature review part of thesis.

It is well known that the P_0 -property is implied by both the P -property and the monotonicity; the P -property is in turn implied by the uniform P -property. In addition, it has been shown [2] that the R_0 -property introduced in the previous section is also implied by the uniform P -property. The uniqueness of solution to the SNME is studied first.

Proposition 3.4 *Let $a > 0$ and the smooth function p have an infinite support (condition (C5)). If f is a P_0 -function over $\mathbb{R}_+^n$, then the SNME has at most one solution for each $u \in (0, 1]$.*

Proof. The result is true for $u = 1$, as indicated in subsection 3.1. Suppose on the contrary that equation $H(z, u) = 0$ has two different solutions $z^1 \neq z^2$ for some $u \in (0, 1)$. Denote $x^k = p(z^k, ua)$ and $y^k = p(-z, ua)$ for $k = 1, 2$. Since p has an infinite support, it is strictly increasing and convex by result (7) of Proposition 3.1. Thus, $x^1 \neq x^2$ and $y^1 \neq y^2$. In addition, we have

$$f(x^k) = \frac{1}{1-u}y^k + \frac{u}{1-u}b \text{ for } k = 1, 2.$$

Since f is a P_0 -function, there exists an index i such that

$$x_i^1 \neq x_i^2 \text{ and } (f_i(x^1) - f_i(x^2))(x_i^1 - x_i^2) \geq 0,$$

or equivalently,

$$x_i^1 \neq x_i^2 \text{ and } (y_i^1 - y_i^2)(x_i^1 - x_i^2) \geq 0.$$

Without loss of generality, one may assume $x_i^1 > x_i^2$. Then the above inequality implies that $y_i^1 \geq y_i^2$. However, since p has finite support, $x_i^1 > x_i^2$ implies $z_i^1 > z_i^2$, $y_i^1 \geq y_i^2$ implies $z_i^1 \leq z_i^2$. This leads to a contradiction. $\square$

Similar uniqueness result can be established for the NME when $\mathbf{a} \geq 0$ has zero components or the plus-smooth function involved has finite support.

Proposition 3.5 *Let $a \geq 0$ and the smooth function p have a finite support (condition (C_4)). If f is a P -function then the SNME has at most one solution for all $u \in [0, 1]$.*

Proof. The proof is a slight modification of the previous one. Suppose on the contrary that equation $H(z, u) = 0$ has two different solutions $z^1 \neq z^2$ for some $u \in [0, 1]$. Let x^k and y^k be defined the same as in the previous proof. We claim that $x^1 \neq x^2$. Since otherwise $y^1 = y^2 = (1 - u)f(x^k) + ub$ which implies that $z^1 = z^2$. However, this contradicts to the assumption. Since f is a P -function and $x^1 \neq x^2$, by the same argument as in the previous proof, there exists an index i such that

$$x_i^1 \neq x_i^2 \text{ and } (y_i^1 - y_i^2)(x_i^1 - x_i^2) > 0.$$

Without loss of generality, one may assume $x_i^1 > x_i^2$. Then the above inequality implies that $y_i^1 > y_i^2$. However, for any plus-smooth function, $x_i^1 > x_i^2$ implies $z_i^1 > z_i^2$, $y_i^1 > y_i^2$ implies $z_i^1 > z_i^2$. This leads to a contradiction. $\square$

We now provide a set of sufficient conditions to guarantee the existence of a monotone trajectory T for plus-smooth functions with finite or infinite supports.

- (A1) $a > 0$, p satisfies (C4), and f is both P_0 and R_0 -function;
- (A2) $a > 0$, p satisfies (C4) and (C5), f is a monotone function over R_+^n , and $\text{NCP}[f]$ has strictly positive solution;
- (A3) $a \geq 0$ and f is both P and R_0 -function.

Theorem 3.2 *If one of the three assumptions (A1)-(A3) is satisfied, the following statements are true: 1. For each $u \in (0, 1]$, the SNME has a unique solution $z(u)$; hence the trajectory can be rewritten as $T = \{(z(u), u) : u \in (0, 1]\}$, which is monotone with respect to u . 2. Trajectory T is bounded; hence there is at least one limit point of $z(u)$ as $u \rightarrow \infty$. 3. Let z^* be any limit point of $z(u)$ as $u \rightarrow 0$. Then z_+^* is a solution of the NME. In particular, the solution is unique under assumption (A3).*

Proof. To prove the first result, in view of Proposition 3.1 and 3.5, it suffices to show the existence of solution. Recall that the SNME $H(z, u) = 0$ has a unique solution for $u = 1$. Let $0 \leq \bar{u} \leq 1$ be the supremum of u 's such that the SNME has a solution for every $u \in [\bar{u}, 1]$. Then there exists a sequence $\{(z^k, u^k)\}$ such that z^k solves the SNME with parameter u^k and $\lim_{k \rightarrow \infty} u^k = \bar{u}$. Proposition 3.2 and 3.3 ensure that the sequence $\{z^k\}$ is bounded under all three assumptions (A1)-(A3). Hence, one may assume that it converges to some $\bar{z} \in R^n$. since function H is continuous on z , it must happen that $H(\bar{z}, \bar{u}) = 0$. Hence, if $\bar{u} = 0$, the desired result follows. Assume on the contrary that $\bar{u} > 0$. Under any of the assumptions (A1)-(A3), the generalized Jacobian $\partial H(z, u)$ is nonsingular (see Proposition 4.1 and 4.2) for all $z \in R^n$ and $u > 0$. Hence, the SNME has a unique solution for every u sufficiently close to $\bar{u}$. However, this contradicts the definition of $\bar{u}$. Therefore, the SNME has a unique solution for

all $u \in (0, 1]$ and the trajectory T is monotone. The second result is a direct consequence of the first result and the fact that T is bounded. To prove the last result, let z^* be any limiting point of $z(t)$ as $t \rightarrow \infty$. By the continuity of the mapping H , we have $H(z^*, 0) = 0$ or $f(z_+^*) + z_-^* = 0$. Hence z^* is a solution of the NME. In particular, by Proposition 3.5, z^* is unique if f is a P -function. $\square$

The above result clearly applies to an NCP with a uniform P -function, since the uniform P -property implies both R_0 -property and P -property. Moreover, as a corollary to the above result, we have also established the existence of solutions for the NCP with a P_0 and R_0 -function. In addition, if the trajectory T is monotone, the implementation of a continuation method could be simplified: A simple lift step on u can be used as a predictor.



Chapter 4

SUBPROBLEMS OF NEWTON CORRECTOR

From the discussion of previous chapter, $z(u)$, the solution of SNME $H(z, u) = 0$, in general forms a trajectory with a unique starting point at $u = 1$ and an ending point at a solution of the NME. Therefore, a continuation method can be designed to follow the trajectory and locate a solution of the NME. Most continuation methods perform predictor and corrector steps alternatively. The predictor step starts from a point close to the trajectory and moves along the trajectory approximately. Depending on the property of the trajectory, one may chose to use more complicated Euler predictor or lift predictor by simply reducing u . The corrector step brings back the newly moved point to a neighbor of the trajectory to prepare for the next predictor step. Newton's method is often used as a corrector in many continuation methods. We will study the subproblems of the Newton corrector in this chapter.

Let $a \geq 0$ and $0 < u \leq 1$. As discussed in Subsection 3.1. $H(z, u)$ is C^1 if $a > 0$ and piecewise C^1 if some components of a are zeros. At a given point z , the Newton's direction d is obtained by solving the following generalized Newton equations:

$$\mathbf{V}\mathbf{d} + \mathbf{H}(\mathbf{z}, \mathbf{u}) = 0$$

where $\mathbf{V} \in \partial\mathbf{H}(\mathbf{z}, \mathbf{u})$ with $\partial\mathbf{H}(\mathbf{z}, \mathbf{u})$ being the generalized Jacobian (see [13]) of H defined at $(\mathbf{z}, \mathbf{u})$. The method and its convergence properties has been studied by [54].

Although it is in general difficult to represent a generalized Jacobian explicitly, it can be done for the SNME, due to its special structure.

$$\partial\mathbf{H}(\mathbf{z}, \mathbf{u}) = \begin{cases} \{(1-u)\nabla\mathbf{f}(\mathbf{p}(\mathbf{z}, \mathbf{u}))\mathbf{D} + \mathbf{I} - \mathbf{D} \mid \mathbf{D} = \text{diag}\{\mathbf{D}_i\}, \\ D_i = p'(z_i, ua_i) \text{ if } a_i > 0, \\ D_i = 1 \text{ if } a_i = 0, z_i > 0, \\ D_i \in [0, 1] \text{ if } a_i = 0, z_i = 0, \\ D_i = 0 \text{ if } a_i = 0, z_i < 0 \} \end{cases}$$

The following two lemmas were introduced by Chen, Harker and Pınar [2] for establishing nonsingularity of generalized Jacobian $\partial\mathbf{H}(\mathbf{z}, \mathbf{u})$.

Lemma 4.1 *Let $\mathbf{D}_i, i = 1, 2, 3$, be a positive diagonal matrices of appropriate dimension as defined in matrix $\mathbf{M}'$ below*

$$\mathbf{M}' = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12}\mathbf{D}_1 & \mathbf{0} \\ \mathbf{M}_{21} & \mathbf{M}_{22}\mathbf{D}_1 + \mathbf{D}_2 & \mathbf{0} \\ \mathbf{M}_{31} & \mathbf{M}_{32}\mathbf{D}_1 & \mathbf{D}_3 \end{pmatrix}$$

$\mathbf{M}'$ is nonsingular if $\mathbf{M}_{11}$ is nonsingular and $\mathbf{M}_{22} - \mathbf{M}_{21}\mathbf{M}_{11}^{-1}\mathbf{M}_{12}$ is a P_0 -matrix.

Proof. Let $\mathbf{x} = (x_1, x_2, x_3)^T$ be any vector such that $\mathbf{M}'\mathbf{x} = \mathbf{0}$. Since $\mathbf{M}_{11}$ is nonsingular, we have, after some algebraic calculations,

$$[(\mathbf{M}_{22} - \mathbf{M}_{21}\mathbf{M}_{11}^{-1}\mathbf{M}_{12})\mathbf{D}_1 + \mathbf{D}_2]\mathbf{x}_2 = \mathbf{0}$$

By assumption, $\mathbf{M}_{22} - \mathbf{M}_{21}\mathbf{M}_{11}^{-1}\mathbf{M}_{12}$ is a P_0 -matrix. It follows that the whole matrix in front of x_2 is a P -matrix and thus nonsingular. This implies $x_2 = 0$. Following similar calculations, we have

$$x_1 = -\mathbf{M}_{11}^{-1}\mathbf{M}_{12}\mathbf{D}_1x_2 = 0, \quad x_3 = -\mathbf{D}_3^{-1}(\mathbf{M}_{31}x_1 + \mathbf{M}_{23}\mathbf{D}_1x_2) = 0$$

Therefore $\mathbf{x} = \mathbf{0}$ and $\mathbf{M}'$ is nonsingular by definition. $\square$

Lemma 4.2 *Let $\mathbf{M}$ be a P -matrix. Then $\mathbf{MD} + \mathbf{I} - \mathbf{D}$ is nonsingular for all diagonal matrix $\mathbf{D}$ such that $0 \leq D_i \leq 1$ for all i .*

Proof. Let $\mathbf{D}$ be any diagonal matrix such that $0 \leq D_i \leq 1$ for all i . Define the following index set associated with $\mathbf{D}$:

$$\alpha = \{i : D_i = 1\}, \quad \beta = \{i : 0 < D_i < 1\}, \quad \gamma = \{i : D_i = 0\}$$

Then matrix $\mathbf{MD} + \mathbf{I} - \mathbf{D}$ can be simplified as:

$$\begin{pmatrix} \mathbf{M}_{\alpha\alpha} & \mathbf{M}_{\alpha\beta}\mathbf{D}_\beta & \mathbf{0}_\alpha \\ \mathbf{M}_{\beta\alpha} & \mathbf{M}_{\beta\beta}\mathbf{D}_\beta + \mathbf{I}_\beta - \mathbf{D}_\beta & \mathbf{0}_\beta \\ \mathbf{M}_{\gamma\alpha} & \mathbf{M}_{\gamma\beta}\mathbf{D}_\beta & \mathbf{I}_\gamma \end{pmatrix}$$

It is well known that any submatrix of P -matrix is also a P -matrix, and any Schur-complement of a P -matrix is also a P -matrix. Thus $\mathbf{M}_{\alpha\alpha}$ is nonsingular and $\mathbf{M}_{\beta\beta} - \mathbf{M}_{\beta\alpha}\mathbf{M}_{\alpha\alpha}^{-1}\mathbf{M}_{\alpha\beta}$ is a P -matrix, since $\mathbf{M}$ is a P -matrix. By Lemma 4.1, the whole matrix is nonsingular. $\square$

The following result provides a sufficient condition for all $\mathbf{V} \in \partial\mathbf{H}(\mathbf{z}, \mathbf{u})$ to be nonsingular.

Proposition 4.1 *Let $a \geq 0$ and $u \geq 0$. If f is a P -function over R_+^n then all $\mathbf{V} \in \partial\mathbf{H}(\mathbf{z}, \mathbf{u})$ is nonsingular for all $\mathbf{z} \in R^n$.*

Proof. Since f is a P -function over R_+^n , $\nabla f(p(z, ua))$ is a P -matrix for all $z \in R^n$. The result then follows from Lemma 4.2. $\square$

The nonsingularity condition can be weakened if a is chosen to be strictly positive. Then $H(z, u)$ is differentiable for all $u > 0$ and the generalized Jacobian $\partial H(z, u)$ reduces to the regular Jacobian $\nabla H(z, u)$. If the plus-smooth function has an infinite support, the Jacobian of H is given by:

$$\nabla H(z, u) = (1 - u)\nabla f(p(z, ua))\text{diag}\{p'(z_i, ua_i)\} + \text{diag}\{p'(-z_i, ua_i)\}$$

The Newton corrector subproblem is obviously a system of linear equations of full dimension n . The nonsingularity condition is given as follows.

Proposition 4.2 *Let $a > 0$ and the plus-smooth function p have an infinite support. Then $\nabla H(z, u)$ is nonsingular for all $z \in R^n$ if f is a P_0 -function over R_+^n .*

Proof. From result (7) of Proposition 3.1, $0 < p'(z_i, ua_i) < 1$ for all $z \in R$ and $u > 0$. Since f is a P_0 -function over R_+^n , $\nabla f(p(z, ua))$ is a P_0 -matrix for all $z \in R^n$. It follows that $\nabla H(z, u)$ is a P -matrix and therefore, is nonsingular. Now consider the case that the plus-smooth function p has a finite support and that $a > 0$. Without loss of generality, assume that the support of the plus-smooth function is on $[-1, 1]$, i.e., $s = 1$. To study the structure of the Newton corrector subproblem, we define the following index sets at a given point $(z, u) \in R^n \times (0, 1]$:

$$\begin{aligned}\alpha(z, u) &= \{i : z_i \geq ua_i\} \\ \beta(z, u) &= \{i : -ua_i < z_i < ua_i\} \\ \gamma(z, u) &= \{i : z_i \leq -ua_i\}\end{aligned}$$

For simplicity, the argument (z, u) of all index sets will be dropped. By definition and result (8) of Proposition 3.1,

$$\begin{aligned}
p'(z_i, ua_i) &= 1 & \text{for all } i \in \alpha, \\
0 < p'(z_i, ua_i) &< 1 & \text{for all } i \in \beta, \\
p'(z_i, ua_i) &= 0 & \text{for all } i \in \gamma
\end{aligned}$$

Let $\mathbf{p}' = \text{diag}\{p'(z_i, ua_i), i \in \beta\}$ and denote $\nabla \mathbf{f}_{st}$ as a submatrix with elements $\partial f_i(p(z, ua))/\partial z_i$, $i \in s$ and $j \in t$. Then, after some algebraic manipulations, we have

$$\nabla \mathbf{H}(\mathbf{z}, \mathbf{u}) = (1 - u) \begin{pmatrix} \nabla \mathbf{f}_{\alpha\alpha} & \nabla \mathbf{f}_{\alpha\beta} \mathbf{p}'_{\beta} & \mathbf{0}_{\alpha} \\ \nabla \mathbf{f}_{\beta\alpha} & \nabla \mathbf{f}_{\beta\beta} \mathbf{p}'_{\beta} + (\mathbf{I}_{\beta} - \mathbf{p}'_{\beta})/(1 - u) & \mathbf{0}_{\beta} \\ \nabla \mathbf{f}_{\gamma\alpha} & \nabla \mathbf{f}_{\gamma\beta} \mathbf{p}'_{\beta} & \mathbf{I}_{\gamma}/(1 - u) \end{pmatrix}$$

□

Piecewise quadratic smooth plus function provides us this special structured jacobian matrix. In the software package, this special structure is used for piecewise quadratic function and it reduces CPU time significantly for some test cases (see figures and tables). In the software only a small matrix with reduced dimension $|\alpha| + |\beta|$ is solved (detailed information is given in Chapter 6). The next result provides a condition for the Jacobian to be nonsingular.

Proposition 4.3 *Let $a > 0$ and the plus-smooth function p have a finite support. Then $\nabla \mathbf{H}(\mathbf{z}, \mathbf{u})$ is nonsingular if at $(\mathbf{z}, \mathbf{u}) \in \mathbb{R}^n \times (0, 1]$*

1. $\nabla \mathbf{f}_{\alpha\alpha}$ is nonsingular and
2. the Schur-complement $\nabla \mathbf{f}_{\beta\beta} - \nabla \mathbf{f}_{\beta\alpha} \nabla \mathbf{f}_{\alpha\alpha}^{-1} \nabla \mathbf{f}_{\alpha\beta}$ is a P_0 -matrix.

Proof. The result follows directly from Lemma 4.1. □

Based on the structure of the above Jacobian, it is clear that the Newton corrector subproblem of a continuation method using a plus-smooth function with a finite support is system of linear equations of reduced dimension $|\alpha| + |\beta|$. It compares favorably to the continuation method using a plus-smooth function with a infinite support, where linear equations of full dimension need to be solved as a Newton corrector subproblem. If a solution of the NME is nondegenerate and if a continuation method converges to the solution, the correct active set α and γ are identified in finite steps.

Proposition 4.4 *Let z^* be a solution of the NME and that $z_i^* \neq 0$ for all i . If the plus-smooth function p has an finite support, then there exists $\bar{u} > 0$ such that $H(z^*, ua) = 0$ for all $0 \leq u \leq \bar{u}$.*

Proof. By result (8) Proposition 3.1, for any $z \in 0$, there exists a $\bar{u} > 0$ such that $p(z, u) = z_+$ and $p(-z, u) = -z_-$ for all $0 \leq u \leq \bar{u}$. The result then follows directly. As a result, a continuation method using a plus-smooth function with a finite support converges to a nondegenerate solution of the NME in finite step (if Newton corrector is used). $\square$



Chapter 5

THE ALGORITHM AND SOFTWARE

In this chapter we look at a simple version of a continuation method (Chen, Harker and Pınar [2]), summarize the computational results and describe the software.

5.1 The Algorithm

The algorithm monotonically reduces u at each iteration (predictor step) and uses Damped Newton's method as a corrector. A nonmonotone line search procedure is chosen instead of the traditional Armijo line search. Our experience indicates that the nonmonotone line search may significantly reduce the number of function evaluations and improve the convergence. For each corrector step, define the merit function $\Psi(z, u) \equiv \|H(z, u)\|_2^2$.

The nonmonotone line search consists of finding the smallest $i_k = 0, 1, 2, \dots$ and thus $\alpha^k = 2^{-i_k}$ such that

$$\Psi(z^k + 2^{-i_k} d^k, u^k) \leq W + \beta 2^{-i_k} \nabla \psi(z^k, u^k)^T d^k,$$

where W is any value satisfying

$$\Psi(z^k, u^k) \leq W \leq \max_{j=0,1,\dots,M^k} \Psi(z^{k-j}, u^{k-j})$$

and M^k is a nonnegative integer with large enough k so as to guarantee the occurrence of nonnegative indices. Our implementation of the nonmonotone search is as follows:

- Set $W = \Psi(z^0, u^0)$ at the beginning of the algorithm,
- keep the value of W fixed as long as

$$\Psi(z^k, u^k) \leq \min_{j=0,1,\dots,5} \Psi(z^{k-j}, u^{k-j}),$$

- If this equation is not satisfied at iteration k , set $W = \Psi(z^k, u^k)$.

Interested reader can find more detailed description of the nonmonotone line search at [63]. The continuation method is now described in detail.

5.1.1 Continuation Method

Step 0 Set $k = 0$. Choose z^k, u^k , and a and b .

Step 1 If $\|H(z^k, u^k)\| \leq \epsilon$, stop.

Step 2 If $\nabla H(z^k, u^k)$ is nonsingular solve for d^k in

$$H(z^k, u^k) + \nabla H(z^k, u^k)d^k = 0.$$

Otherwise, let $d^k = -H(z^k, u^k)$.

Step 3 Compute a step length α^k using the nonmonotone line search.

Step 4 Set

$$z^{k+1} = z^k + \alpha^k d^k,$$

$$u^{k+1} = u^k * \beta.$$

$$k \leftarrow k + 1$$

Go to Step 1. In the above algorithm we choose $\epsilon = 10^{-6}$, $\beta \in (0, 1)$, $a = 1$ and $b = 1$.

5.2 Software

A software which is based on the algorithm is developed in ANSI FORTRAN 77, it uses a sparse linear system solver UMFPACK2 and its name is NCPNMS (Nonlinear Complementarity Problem Normal Map Solver). The NCPNMS can exploit sparsity. Therefore it doesn't carry zero entries of jacobian matrices. It firstly calculates initial function values and its jacobian matrix and $\|H(z, u)\|_2$ by using the starting point. After this initialization, in a loop (loop continues until termination point is reached), it use damped-Newton algorithm and nonmonotone line search (firstly it finds step length, then updates z). In addition to this, for piecewise quadratic function, NCPNMS doesn't solve $\nabla H * dz = h$ with full ∇H jacobian matrix. Due to special structure of ∇H matrix of piecewise quadratic function, software solve only

$$(1 - u) \begin{bmatrix} \nabla f_{\alpha\alpha} & \nabla f_{\alpha\beta} p'_\beta \\ \nabla f_{\beta\alpha} & \nabla f_{\beta\beta} p'_\beta + (I_\beta - p'_\beta)/(1 - u) \end{bmatrix}$$

part of ∇H matrix. For this purpose, NCPNMS make arrangement on ∇H matrix and it makes new indexes for α , β and γ . After solving this equation system with reduced matrix, for γ part, software calculates

$$d_\gamma = h_\gamma - (1 - u) * \nabla f_{\gamma\alpha} d_\alpha - (1 - u) \nabla f_{\gamma\beta} p'_\beta d_\beta.$$

Other parts are identical to parts of the software package with interior plus function except that we maintain index set α , β , γ for the piecewise quadratic function. Numerical testing shows that solving this reduced matrix is more efficient according to computational time. Expecially for big problems (large n), this difference can be seen more precisely.

5.2.1 Numerical Example

Let us consider a Modified Mathiesen problem as an example of NCP(Piecewise quadratic plus-smooth function is used). The dimension of this problem is 4 and

starting point is (1,1,1,1) and starting $\mu = 1$. Function vector:

$$\begin{aligned} f(1) &= x(2) + x(3) + x(4) \\ f(2) &= x(1) - (4.5 \times x(3) + 2.7 \times x(4)) / (x(2) + 1) \\ f(3) &= 5 - x(1) - (0.5 \times x(3) + 0.3 \times x(4)) / (x(3) + 1) \\ f(4) &= 3 - x(1) \end{aligned}$$

and the Jacobian matrix of the function:

$$\nabla f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{4.5 \times x(3) + 2.7 \times x(4)}{(x(2) + 1)^2} & \frac{4.5}{x(2) + 1} & \frac{2.7}{x(2) + 1} \\ 1 & 0 & \frac{-0.5 + 0.3 \times x(4)}{(x(3) + 1)^2} & \frac{-0.3}{x(3) + 1} \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

For starting point $z = (1, 1, 1, 1)$:

$$p'(z, 1) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

and

$$p'(-z, 1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

and

$$p(z, 1) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

Jacobian matrix of function:

$$\nabla f = \begin{bmatrix} 0 & -1 & 1 & 1 \\ 1 & 1.8 & -2.25 & -1.35 \\ -1 & 0 & -0.05 & -0.15 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

due to this matrix :

$$\nabla H(z,1) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$f(p(z,1)) = \begin{bmatrix} 1 \\ -2.6 \\ 3.6 \\ 2 \end{bmatrix}$$

and

$$p(-z,1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

due to these values:

$$H(z,1) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

and $\|H(z,1)\|_2^2 = 4$ Therefore solution of $\nabla H(z,1) \times d = -H(z,1)$:

$$d = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

Now we must find step length, in the first iteration Armijo condition is satisfied at beginning with $\alpha = 1$. Therefore new point is

$$z = z + \alpha \times d$$

$$z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Due to this new point, new $p(z,1)$ and $f(p(z,1))$ is calculated again. And for every entry of these vectors, minimums are selected. Norm of this new vector is calculated. This norm is the error term and its value is smaller than 10^{-6} the algorithm stops with optimal point, otherwise the algorithm proceeds. Due to current point error term is 0.70887 and this is greater than 10^{-6} , therefore continue...(now new $\mu = 0.1$)

Due to this z ,

$$p'(z, 0.1) = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

and

$$p'(-z, 0.1) = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

and

$$p(z, 0.1) = \begin{bmatrix} 0.0626 \\ 0.0625 \\ 0.0625 \\ 0.0625 \end{bmatrix},$$

Jacobian matrix of function:

$$\nabla f = \begin{bmatrix} 0 & -1 & 1 & 1 \\ 1 & 0.39862 & -4.2353 & -2.5412 \\ -1 & 0 & -0.4263 & -0.28235 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

due to this matrix :

$$\nabla H(z, 1) = \begin{bmatrix} 0.5 & -0.25 & 0.25 & 0.25 \\ 0.25 & 0.59966 & -1.0588 & -0.6353 \\ -0.25 & 0 & 0.3934 & -0.070588 \\ -0.25 & 0 & 0 & 0.5 \end{bmatrix}$$

and

$$f(p(z, 0.1)) = \begin{bmatrix} 0.0625 \\ -0.361 \\ 4.89 \\ 2.9375 \end{bmatrix}$$

and

$$p(-z, 0.1) = \begin{bmatrix} 0.0625 \\ 0.0625 \\ 0.0625 \\ 0.0625 \end{bmatrix}$$

due to these values:

$$H(z, 0.1) = \begin{bmatrix} 0.46875 \\ 0.25698 \\ 2.8827 \\ 1.90625 \end{bmatrix}$$

and $\|H(z, 0.1)\|_2^2 = 12.23$ Therefore solution of $\nabla H(z, 0.1) \times d = -H(z, 0.1)$:

$$d = \begin{bmatrix} -4.73 \\ -25.2 \\ -11.44 \\ -6.18 \end{bmatrix}$$

Now we must find step length, in this case Armijo condition is not satisfied with $\alpha = 1$, $\alpha = 0.5$ and $\alpha = 0.25$. After two inner iteration for step length for $\alpha = 0.125$ Armijo condition is satisfied. Therefore new point is

$$z = z + \alpha \times d$$

$$z = \begin{bmatrix} -0.5912 \\ -3.1504 \\ -1.4301 \\ -0.7722 \end{bmatrix}$$

Due to this new point, new $p(z, 0.1)$ and $f(p(z, 0.1))$ is calculated again. And for every entry of these vectors, minimums are selected. Norm of this new

vector is calculated and this norm is error term and this value is smaller than 10^{-6} stop, optimal point is reached otherwise continue. Due to current point error term is 0 and this is smaller than 10^{-6} , therefore stop optimal point is reached. After 2 iterations solution is reached. The solution vectors are:

$$x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } f(x) = \begin{bmatrix} 0 \\ 0 \\ 5 \\ 3 \end{bmatrix}.$$



Chapter 6

COMPUTATIONAL RESULTS

The algorithm was implemented in FORTRAN 77 and tested on a Sparc Station Classic with one 60 Mhz Micro SPARC 8-CPU and 1 GB main memory running UNIX. I report the results of two experiments below. The first experiment was conducted using a uniform reduction scheme for u . That is, we reduce u by the same factor after each nonmonotone search. The second experiment consists of using a hybrid strategy: reduce slowly at the beginning, then reduce fast. I used test problems from De Luca [63] and Ferris and Rutherford [15]. Next, we describe briefly the test problem characteristics.

6.1 Test Problems

FORTTRAN versions of below problems can be obtained from the author.

Kojima-Shindo: This is a 4-variable problem. F is not a P_0 -function. The starting points are: (a) (0,0,0,0), (b) (1,1,1,1).

Kojima-Josephy: This is a 4-variable problem. F is not a P_0 -function. The starting point are: (a) (0,0,0,0), (b) (1,1,1,1).

HS34: This problems represents the KKT conditions of a nonlinear program. F is monotone on the positive orthant but not even P_0 on R^n . The dimension

is 8. Starting point is (a) $(0, 1.05, 2.9, 0, 0, 0, 0, 0)$.

HS35: This problem represents the KKT conditions for the 35th problem in [22]. F is linear and monotone but not strictly monotone. The dimension is 4. Starting point is (a) $(0.5, 0.5, 0.5, 0)$.

HS66: This problem represents the KKT conditions for the 66th problem in [22]. F is monotone on the positive orthant but not even P_0 on R^n . The dimension is 8. Starting point is (a) $(0, 1.05, 2.9, 0, 0, 0, 0, 0)$.

HS76: This problem represents the KKT conditions for the 76th problem in [22]. F is linear and monotone but not strictly monotone. The dimension is 7. Starting point is (1) $(0.5, 0.5, 0.5, 0.5, 0, 0, 0)$.

Watson3: This is linear complementarity problem with $F(x) = Mx + q$. M is not even semi-monotone. This is known to be a hard problem. In fact, De Luca *et al.*] indicate that none of the standard algebraic techniques can solve this problem. We choose $q = (-1, 0, \dots, 0)$ as in [22]. The dimension is 10. Starting point is (a) $(0, 0, \dots, 0)$.

Watson4: This problem represents the KKT conditions for a convex programming problem involving exponentials. F is monotone on the positive orthant but not even P_0 on R^n . The dimension is 5. Starting point is (a) $(0, 0, \dots, 0)$.

Nash-Cournot: F is not twice continuously differentiable. F is a P -function on the strictly positive orthant. The dimension is 10. Starting points are (a) $(10, 10, \dots, 10)$, (b) $(1, 1, \dots, 1)$.

Modified Mathiesen: F is not defined everywhere and does not belongs to any known class of functions. The dimension is 4. Starting points are (a) $(1, 1, 1, 1)$, (b) $(10, 10, 10, 10)$.

Spatial: This is a spatial equilibrium model. F is a P -function. The dimension is 42. Starting points are (a) $(0, 0, \dots, 0)$, (b) $(1, 1, \dots, 1)$.

Traffic: This is a traffic equilibrium problem. The dimension is 50. Starting point is (a). All the components are 0 except $z_1, z_2, z_3, z_{10}, z_{11}, z_{20}, z_{21}, z_{22}, z_{29}, z_{30}, z_{40}, z_{45}$ which are 1, $z_{39}, z_{42}, z_{43}, z_{46}$ which are 7, $z_{41}, z_{47}, z_{48}, z_{50}$ which are 6, z_{44} and z_{49} which are 10.

Pies: Energy Equilibrium Model. The dimension is 42. Starting point $(0, \dots, 0)$.

Ehl-kost: Elasto-hydrodynamic lubrication. The dimension is 101. The

starting point is $(1, \dots, 1)$.

Colvdual: Dual of Colville Problem Number 2. The dimension is 20. The starting point is $(1, \dots, 1)$.

MyPowell: Powell's Nonlinear Program. The dimension is 16. The starting point is $(1, \dots, 1)$.

Cycle: Cycling example. The dimension is 1. The starting point is 1.

Powell-mcp: Powell's nonlinear program, MCP form. The dimension is 8. The starting point is $(1, \dots, 1)$.

Sppe: Spatial Price Equilibrium Example. The dimension is 30. The starting point is $(1, \dots, 1)$.

Explcp: A sample Linear Complementarity Problem. The dimension is 16. The starting point is $(1, \dots, 1)$.

Bertsekas: Bertsekas problem as a 15-variable NCP. The dimension is 15. The starting point is $(1, \dots, 1)$.

Freebert: Bertsekas problem as a 15-variable MCP. The dimension is 15. The starting point is $(1, \dots, 1)$.

Hanskoop: Hansen /Koopmans invariant capital stock problem. The dimension is 14. The starting point is $(1, \dots, 1)$.

Vonthmcp: General Equilibrium Variant of the Vonthunene Model. The dimension is 126. The starting point is $(1, \dots, 1)$.

Gafni: This problem is based on a traffic assignment problem. The dimension is 5. The starting point is $(1, \dots, 1)$.

Colvnpc: Colville Problem. The dimension is 15. The starting point is $(0, \dots, 0)$.

Hydroc06: Distillation problem. The dimension is 29. The starting point is $(1, \dots, 1)$.

Hydroc20: Distillation problem. The dimension is 99. The starting point is $(1, \dots, 1)$.

Methan08: Distillation problem. The dimension is 31. The starting point is $(1, \dots, 1)$.

Scarfanum: Walrasian problem. The dimension is 14. The starting point is $(0, \dots, 0)$.

Scarfbnum: Walrasian problem. The dimension is 40. The starting point is

$(1, \dots, 1)$.

Pgvon106: Von Thunen model. The dimension is 106. The starting point $(1, \dots, 1)$ and $(0, \dots, 0)$.

6.2 Experiment 1: Uniform Reduction of u

I summarize below in Table 6.1, Table 6.2 and Table 6.3 our experiments with the interior smooth function with infinite support, the piecewise quadratic smooth function with finite support respectively and piecewise quadratic function with reduced $\nabla \mathbf{H}$ jacobian matrix. The column headings are as follows. u^0 denotes the starting point value of u , RS refers to the u reduction strategy, SP stands for the starting point in reference to the previous section, IT stands for iteration number, FE for the number of function evaluations and CPU refers to CPU times. We use two alternative strategies for reducing u :

1 u by $u \leftarrow u/10$

2 u by $u \leftarrow u/2$.

From tables and figures (figure 6.1, 6.2, 6.3, 6.4 and 6.5), we can see that especially piecewise quadratic plus-smooth function with reduced $\nabla \mathbf{H}$ has less CPU time for a lot of problems. However, it is not possible to say that interior plus-smooth function is better than piecewise quadratic plus-smooth function or vice versa. Because for some problems interior function wants less number of iterations and for some other problems piecewise quadratic function wants less number of iterations. But piecewise quadratic function with reduced $\nabla \mathbf{H}$ can be preferred instead of interior. Because for a lot of problems it has less CPU time than interior function even when piecewise quadratic function requires a larger number of iterations. Especially for problems Spatial Equilibrium, Ehl_kost, Explcp, Hanskoop, Sppe algorithm of piecewise quadratic function with reduced matrix reduced CPU time to approximately half of CPU time of piecewise quadratic function with full matrix. These tables don't contain all

the experiments with fail results. Therefore some experiments with different starting points, different starting μ are not in tables.

All problems are converted to ANSI Fortran 77 code from GAMS and MATLAB files. Some problems are in MCP format. Therefore they have upper and lower bounds. However, upper and lower bounds can not be used by normal maps. Therefore reasons of our fails for some problems may be trying to solve problems, whose original types have upper and lower bounds, without upper and lower bounds .



PROBLEMS	u	RS	SP	IT	FE	CPU Time	n
Kojima-Shindo	1	10	a	14	19	0.042318	4
	10	10	b	15	17	0.044265	4
Kojima-Josephy	10	10	b	9	15	0.024996	4
HS34	1	10	a	10	31	0.0435	8
HS35	1	10	a	8	9	0.024433	4
HS66	10	10	a	9	24	0.035842	8
HS76	10	10	a	9	10	0.035105	7
Watson3	1	2	a	22	30	0.106676	10
Watson4	1	2	a	39	39	0.091447	5
Nash-Cournot	1	10	a	9	15	0.059771	10
	10	10	a	10	13	0.056845	10
	10	10	b	11	13	0.062692	10
Mod. Mathiesen	1	2	a	22	24	0.057889	4
	1	10	b	8	19	0.027408	4
Spatial Eq.	10	2	a	27	36	0.660758	42
	1	2	b	24	31	0.552334	42
Traffic Eq.	1	10	a	15	27	0.44834	50
Bertsekas	10	10	1	23	56	0.36294	15
Colvdual	1	10	1	19	83	0.18376	20
Colvnp	10	2	1	20	139	0.165205	15
	10	2	1	20	139	0.159597	15
Cycle	1	10	0	7	12	0.015487	1
Ehl_kost	10	2	0	42	53	0.562476	101
Explcp	10	2	1	27	73	0.222058	16
	10	10	0	9	14	0.079298	16
	1	10	0	8	11	0.072527	16
	1	2	0	23	23	0.184209	16
Freebert	1	2	0	Fail			
	1	2	0	Fail			
	1	2	0	Fail			
	1	10	1	Fail			
	1	10	0	Fail			
	1	2	1	Fail			
	1	2	0	Fail			
Gafni	1	10	1	10	14	0.071293	5
Hanskoop	10	2	1	38	94	0.58473	14
Hydroc06	1	10	1	13	23	0.26231	29
Hydroc20	10	10	1	24	52	0.92843	99
Methan08	10	2	1	12	22	0.27453	31
Pgvon106	10	10	1	Fail			
	10	2	10	Fail			
	1	10	1	Fail			
Pies	10	10	10	17	174	0.42634	42
	10	10	0	Fail			
	10	10	1	Fail			
	1	10	1	Fail			
Powell	10	2	1	16	22	0.23345	16
Powell_mcp	10	10	1	9	11	0.11273	8
	10	2	1	Fail			
	10	2	0	Fail			
Scarfnum	1	10	0	18	45	0.4234	14
Scarfbnum	1	10	0	23	61	0.53476	40
	1	10	1	Fail			
	10	10	1	Fail			
Sppe	10	2	0	Fail			
	10	10	0	12	102	0.159562	30
	10	10	1	9	45	0.115617	30
	10	2	1	Fail			
Vonthunen	1	10	0	Fail			
	1	10	1	Fail			
	1	2	1	Fail			
	10	10	1	Fail			
	10	10	10	Fail			
	10	2	0	Fail			
Total CPU Time (nly one CPU Time (minimum) for every problem) =						6.83362	

Table 6.1
Interior Plus-Smooth Function

PROBLEMS	u	RS	SP	IT	FE	CPU Time	n
Kojima-Shindo	1	10	a	8	13	0.028735	4
	1	10	b	9	14	0.029376	4
Kojima-Josephy	1	10	b	8	12	0.026557	4
HS34	1	2	a	23	27	0.066008	8
HS35	1	10	a	8	10	0.022996	4
HS66	1	2	a	23	24	0.072293	8
HS76	1	10	a	8	11	0.028064	7
Watson3	10	2	a	26	35	0.109984	10
Watson4	10	2	a	26	26	0.062708	5
Nash-Cournot	10	10	a	9	10	0.048967	10
	10	10	b	11	14	0.059474	10
Mod. Mathiesen	1	2	a	2	5	0.011336	4
Mod. Mathiesen	10	2	b	3	10	0.016192	4
Spatial Eq.	10	2	a	27	58	0.454667	42
	10	10	b	18	30	0.293117	42
Traffic Eq.	1	10	a	28	87	0.548375	50
Bertsekas	1	10	1	27	113	0.508476	15
Colvdual	10	2	0	Fail			
	10	10	1	18	94	0.161101	20
	1	10	1	Fail			
Colvnlp	10	2	1	26	148	0.248946	15
	10	10	1	Fail			
	10	10	0	Fail			
Cycle	1	1	0	8	12.56	0.021833	1
Ehl_kost	10	10	1	38	82	0.582622	101
Explcp	10	2	1	37	170	0.24322	16
	1	10	1	9	89	0.061505	16
	10	10	0	Fail			
	1	10	0	Fail			
	10	10	1	Fail			
Freebert				Fail			
Gafni	10	2	0	9	12	0.075854	5
Hanskoop	10	10	1	34	102	0.63484	14
Hydroc06	1	10	1	11	21	0.218484	29
Hydroc20	10	10	1	18	94	0.91747	99
Methan08	1	10	1	14	18	0.33359	31
Pgvon106				Fail			
Pies	10	2	1	14	41	0.436278	42
Powell	1	10	1	11	31	0.168748	16
Powell_mcp	10	10	10	10	17	0.12938	8
Scarfnum	1	2	0	20	71	0.53427	14
Scarfbnum	10	10	1	24	95	0.63573	40
Sppe	10	10	0	12	35	0.125023	30
	1	2	1	23	75	0.237154	30
	1	10	1	Fail			
	10	10	1	Fail			
	1	10	1	Fail			
Vonthunen				Fail			
Total CPU Time (only one CPU Time (minimum) for every problem =						7.27484	

Table 6.2
Piecewise Quadratic Function

PROBLEMS	u	RS	SP	IT	FE	CPU Time	n
Kojima-Shindo	1	10	a	8	13	0.029754	4
	1	10	b	9	14	0.031909	4
Kojima-Josephy	1	10	b	8	12	0.028536	4
HS34	1	2	a	23	27	0.054523	8
HS35	1	10	a	8	10	0.021307	4
HS66	1	2	a	23	24	0.050484	8
HS76	1	10	a	8	11	0.023181	7
Watson3	10	2	a	26	35	0.085872	10
Watson4	10	2	a	26	26	0.062898	5
Nash-Cournot	10	10	a	9	10	0.049203	10
	10	10	b	11	14	0.059618	10
Mod. Mathiesen	1	2	a	2	5	0.011407	4
Mod. Mathiesen	10	2	b	3	10	0.015638	4
Spatial Eq.	10	2	a	27	58	0.342144	42
	10	10	b	18	30	0.210287	42
Traffic Eq.	1	10	a	28	87	0.43645	50
Bertsekas	1	10	1	27	113	0.41956	15
Colvdual	10	2	0	Fail			
	10	10	1	18	94	0.117174	20
	1	10	1	Fail			
Colvnlp	10	2	1	26	148	0.179478	15
	10	2	1	Fail			
	10	10	1	Fail			
	10	10	0	Fail			
Cycle	1	1	0	8	12.56	0.022735	1
Ehl_kost	10	10	1	38	82	0.39534	101
Exlcp	10	10	0	Fail			
	1	10	0	Fail			
	1	10	1	9	89	0.034063	16
	10	10	1	Fail			
	10	2	1	37	170	0.147422	16
Freebert				Fail			
Gafni	10	2	0	9	12	0.06345	5
Hanskoop	10	10	1	34	102	0.49756	14
Hydroc06	1	10	1	11	21	0.175434	29
Hydroc20	10	10	1	18	94	0.725634	99
Methan08	1	10	1	14	18	0.26353	31
Pgvon106				Fail			
Ples	10	2	1	14	41	0.30465	42
Powell	1	10	1	11	31	0.14846	16
Powell_mcp	10	10	10	10	17	0.973452	8
Scarfnum	1	2	0	20	71	0.337645	14
Scarfbnum	10	10	1	24	95	0.419837	40
Sppe	1	10	1	Fail			
	10	10	0	12	35	0.077807	30
	10	10	1	Fail			
	1	10	1	Fail			
	1	2	1	23	75	0.137129	30
Vonthunen				Fail			
Total CPU Time (only one CPU Time (minimum) for every problem) =						6.33307	

Table 6.3
 Piecewise Quadratic Function with reduced matrix

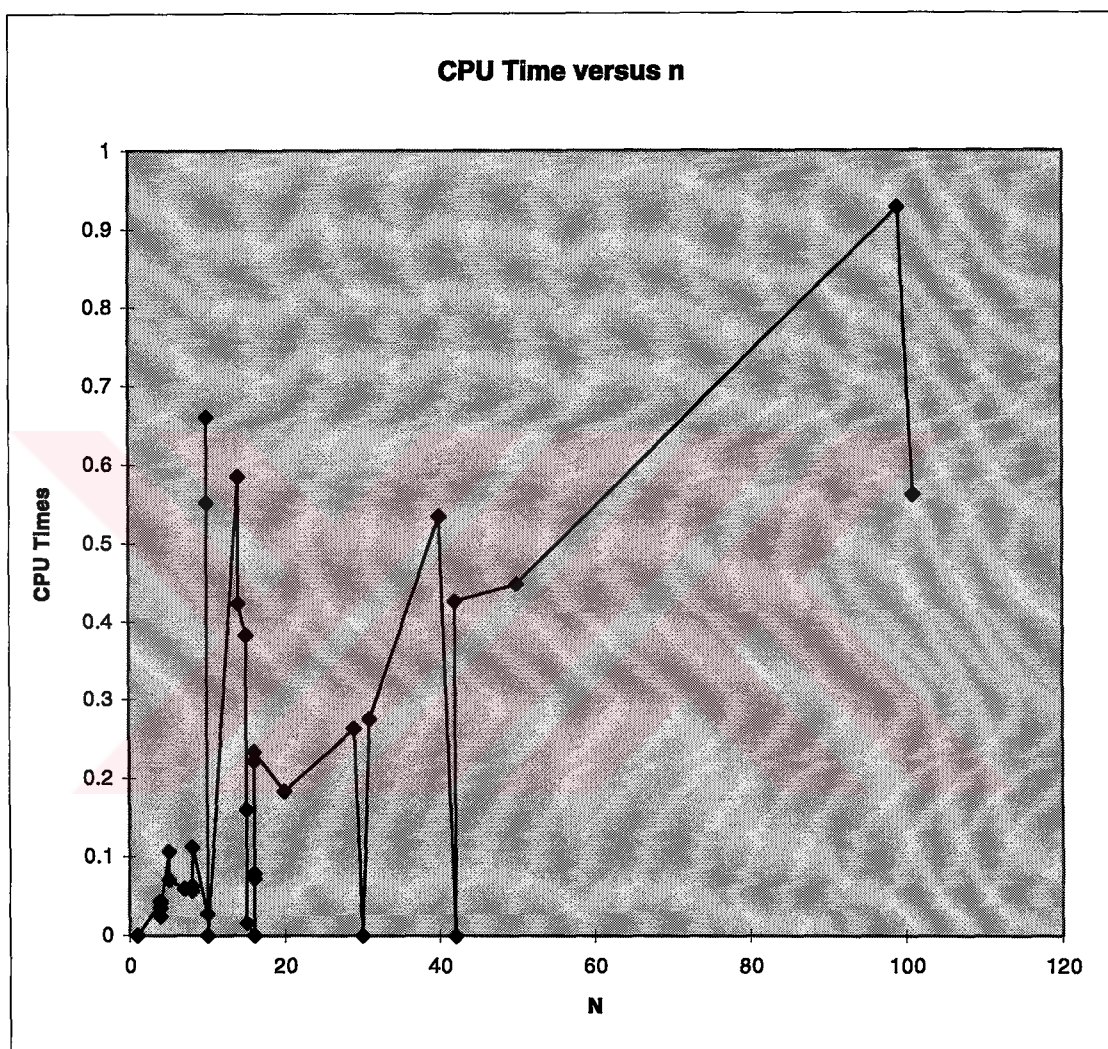


Figure 6.1
Interior plus function
CPU Time versus n

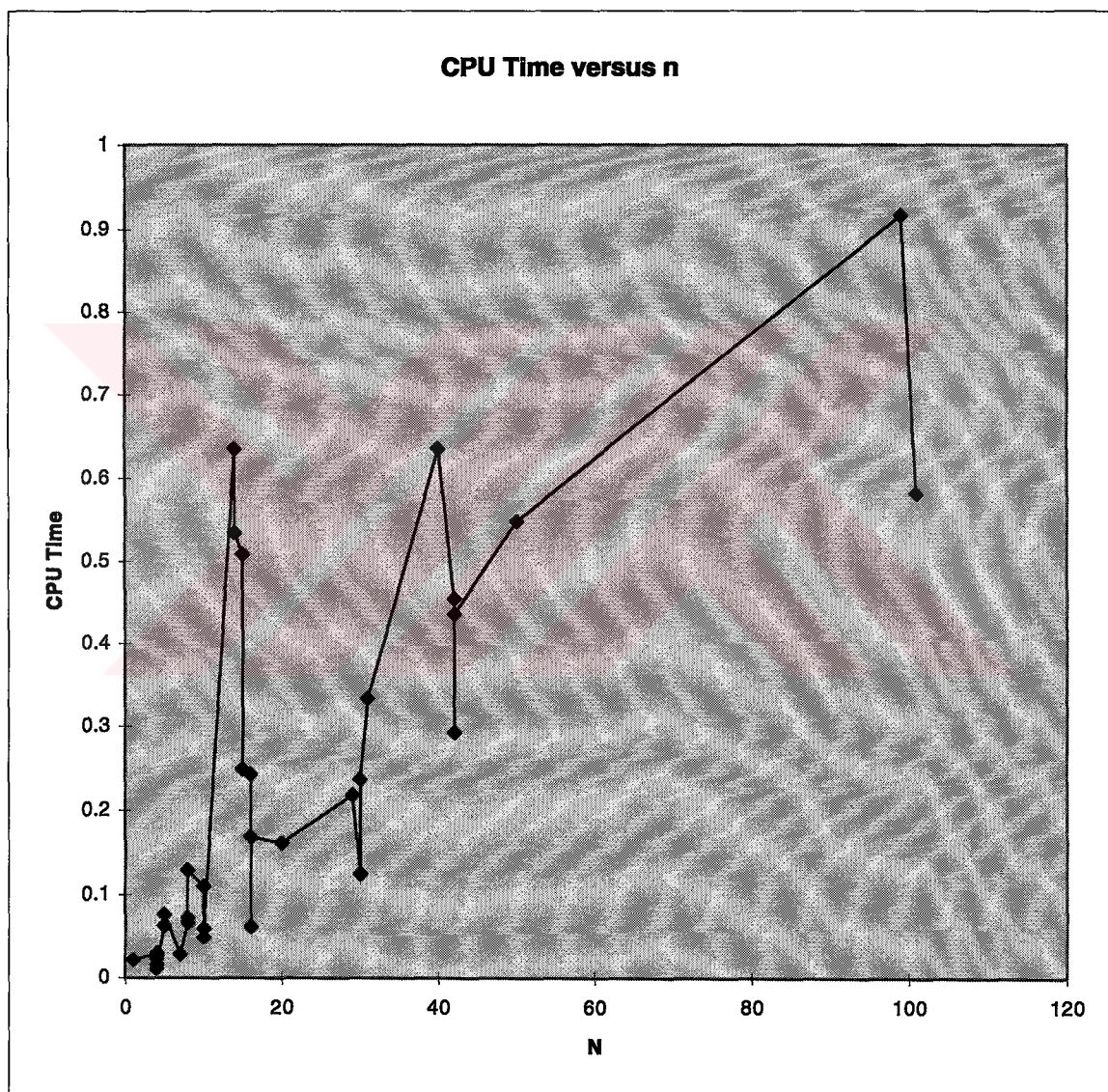


Figure 6.2
CPU Time versus n
Piecewise Quadratic Function

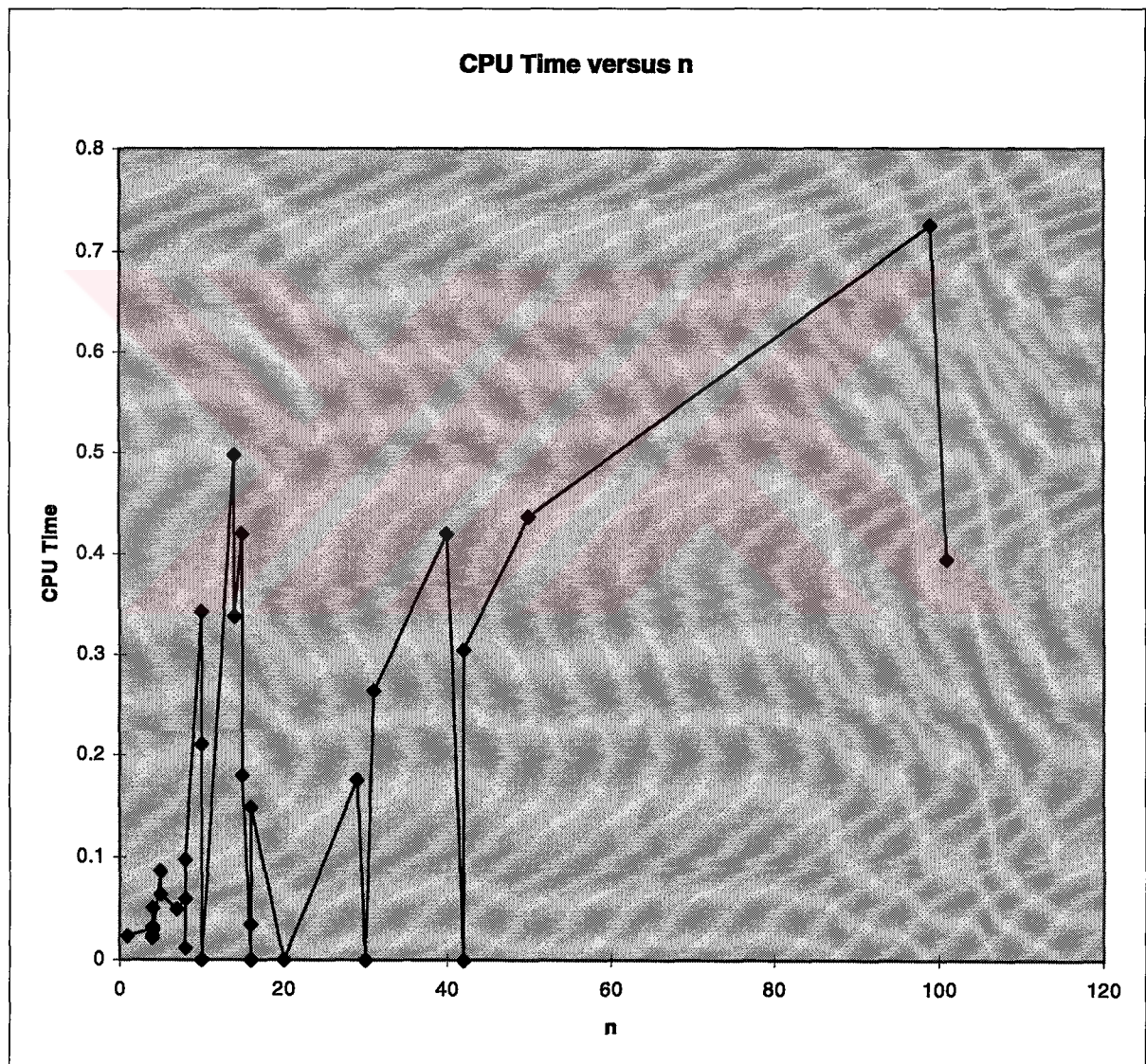


Figure 6.3
CPU Time versus n
Piecewise Quadratic Function with reduced matrix

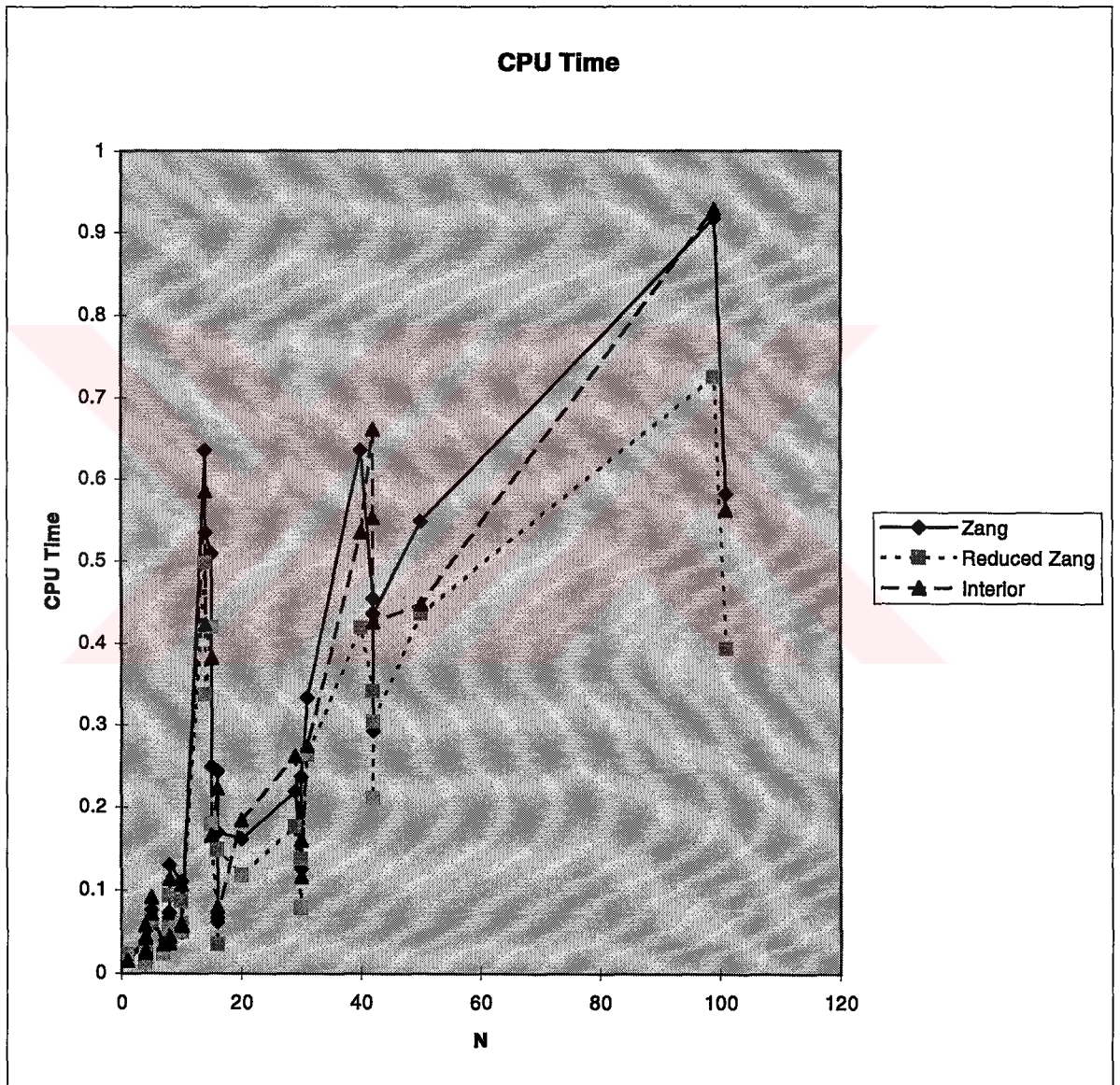


Figure 6.4
Comparison For CPU Time

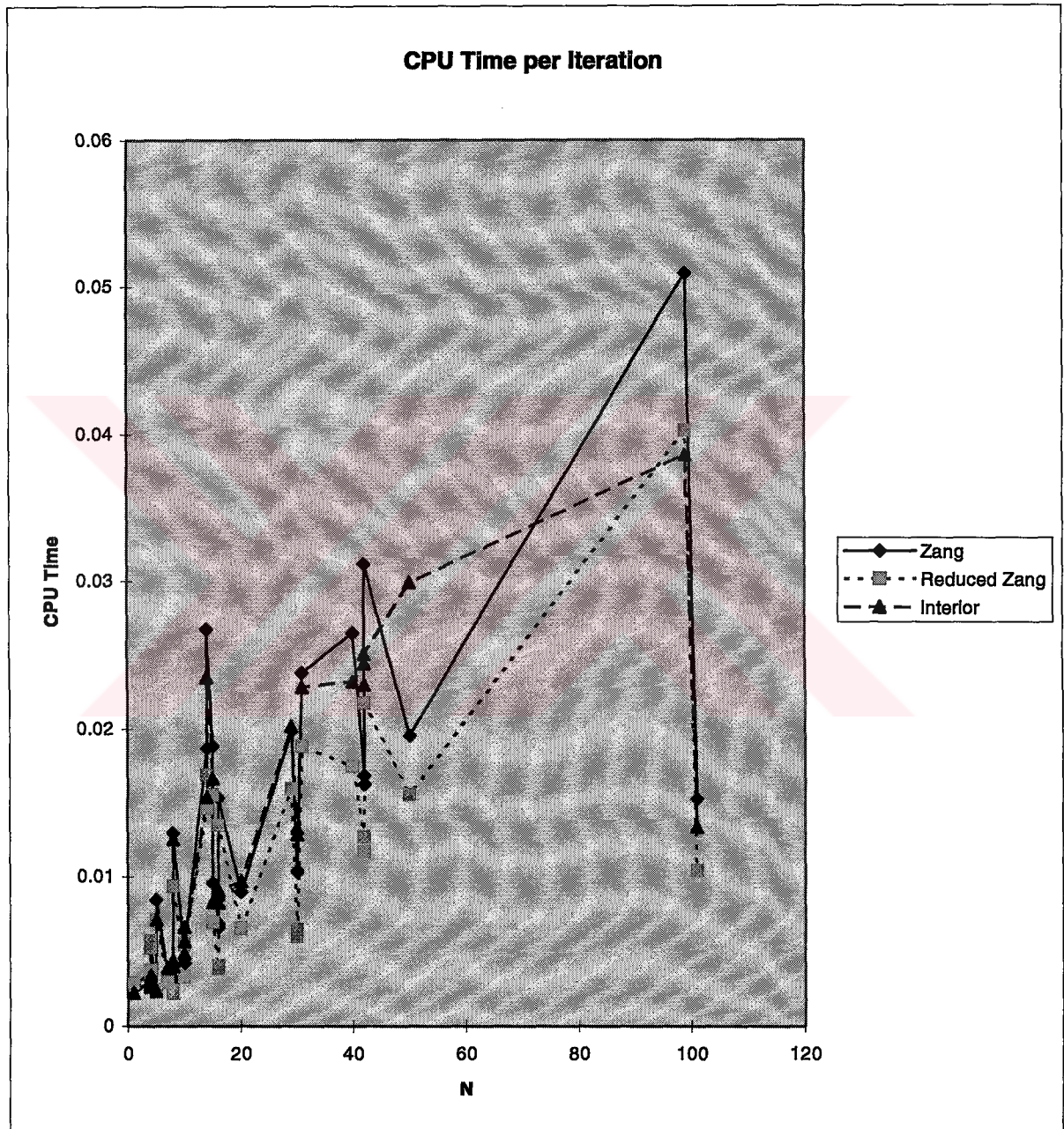


Figure 6.5
Comparisons For CPU Time per Iteration

6.3 Experiment 2: A Hybrid Reduction Scheme for u

I observed in our experiments reported in the previous section that some test problems require that u be reduced slowly at the start of the algorithm. However, this slow reduction of u seems to slow down the tail convergence rate for these problems. Therefore, we tested a hybrid u -reduction scheme where u is reduced slowly at the beginning ($u \leftarrow u/2$) until the complementarity error is under a certain threshold. Then u is reduced faster. We have tested two alternatives at the faster reduction phase:

Strategy 1 $u \leftarrow u/100$,

Strategy 2 $u \leftarrow u/10$.

In table 6.4 below, we report the results of this experiment on problems where some improvement could be expected. The pair of numbers under the column heading RS refer to the u reduction strategy and the threshold value of the error at which we start to reduce faster, respectively.

However, for a lot of problems hybrid reduction scheme for u doesn't affect number of iterations.

Results using the interior smooth function					
Problem	u	RS	SP	IT	FE
Mod. Math	1	(1,1)	a	5	7
Watson4	1	(1,10)	a	19	19
Nash-C.	1	(1,100)	a	9	15
Watson3	1	(1,1)	a	7	13
Spatial Eq	10	(2,10)	a	24	33
Results using the piecewise quadratic smooth function					
Problem	u	RS	SP	IT	FE
Watson4	1	(2,100)	a	18	18
Watson3	1	(1,1)	a	13	22
Spatial Eq	10	(2,10)	b	29	47

Table 6.4

Hybrid u -reduction scheme

Chapter 7

CONCLUSION

In this thesis, a continuation method algorithm which using smoothing methods is implemented. As a plus-smoothing function interior-point function and piecewise quadratic function is used and this functions are compared by using in the same algorithm. Especially due to piecewise quadratic function, $\nabla \mathbf{H}(\mathbf{z}, \mathbf{u})$ jacobian matrix has special structure and this structure reduces computational efforts very largely.

For different problems, number of iterations which are needed to reach termination point can change due to whether interior function or piecewise quadratic function is used. For some problems interior-point wants less iteration (less CPU time), for some problems piecewise quadratic function wants less iteration (less CPU time). Therefore, we couldn't conclude that interior-point method is more powerful than piecewise quadratic function or vice versa. However, due to special structure of $\nabla \mathbf{H}(\mathbf{z}, \mathbf{u})$ with piecewise quadratic function, if this special structure is used in the algorithm, reducing in the computational time (CPU time) is very important. Also choosing of starting point and μ affects number of iterations and hybrid reducing strategy can reduce number of iterations, at least for some problems. And unfortunately, algorithm couldn't reach optimal solution for a few problems. Because we can not use upper and lower bounds of some problems and these may be reasons

of fail results.

Also hybrid reduction scheme reduced number of iterations for only a few problems. Therefore we can not conclude that hybrid reduction scheme is better.

For future research, we would suggest to increase the performance of the algorithm for especially unsuccessful results for some problems. Also it would be interesting to implement a continuation method with Euler predictor step.



Bibliography

- [1] E. Allgower and K. George. *Numerical Continuation Methods: An introduction*. Springer-Verlag, New York, 1990.
- [2] J. Burke and S. Xu. The global linear convergence of a non-interior path-following algorithm for linear complementarity problem. Technical report, Department of Mathematics, University of Washington, Seattle, WA 98195, 1996.
- [3] C.Chen and O.L. Mangasarian. A class of smoothing functions for nonlinear and mixed complementarity problems. *Comp. Optim. and Appl.*, (5):97–138, 1996.
- [4] B. Chen. *A Continuation Method For Monotone Variational Inequality And Complementary Problems: With Application To Linear And Non-linear Programming*. PhD thesis, Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104-6366, 1990.
- [5] B. Chen and P.T. Harker. A non-interior-point continuation method for linear complementarity problems. *SIAM Journal on Matrix Analysis and Applications*, (14):1168–1190, 1993.
- [6] B. Chen and P.T. Harker. A non-interior-point continuation method for quadratic and linear programs. *SIAM J. Optim.*, (3):503–515, 1993.
- [7] B. Chen and P.T. Harker. A continuation method for monotone variational inequalities. *Mathematical Programming*, (69):237–253, 1995.

- [8] B. Chen, P.T. Harker, and M.C. Pinar. Continuation method for nonlinear complementarity problems via normal maps. Technical report, Department of Industrial Engineering, Bilkent University, Ankara, Turkey, (mustafap@bilkent.edu.tr), December 1995.
- [9] B. Chen and N. Xiu. A global linear and local quadratic non-interior continuation method for nonlinear complementarity problems based on chen-mangasarian smoothing functions. Technical report, Department of Management and Systems, Washington State University, Pullman, WA 99164-4736, U.S.A., (chenbi@wsu.edu), February 1997.
- [10] C. Chen and O.L. Mangasarian. Smoothing methods for convex inequalities and linear complementarity problems. *Mathematical Programming*, (71):51–70.
- [11] C. Chen and O.L. Mangasarian. A class of smoothing functions for nonlinear and mixed complementarity problems. *Computational Optimization and Applications*, (5(2)):97–138, 1996.
- [12] X. Chen. Superlinear convergence of smoothing quasi-newton methods for non-smooth equations. Technical report, School of Mathematics, The University of New South Wales, Sydney 2052, Australia, 1996.
- [13] X. Chen, L. Qi, and D. Sun. Global and superlinear convergence of the smoothing newton method and its application to general box constrained variational inequalities. Technical report, School of Mathematics, The University of New South Wales, Sydney 2052, Australia, 1996. AMR 96/26, Applied Mathematics Report.
- [14] F.H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley, New York, 1983.
- [15] S.P. Dirkse and M.C. Ferris. The path solver: A non-monotone stabilization scheme for mixed complementarity problems. Technical report, Computer Science Department, University of Wisconsin, 1993. 1179.

- [16] M.C. Ferris and T.F. Rutherford. Accessing realistic mixed complementarity problems within matlab. Technical report, University of Wisconsin, Madison, June 1995. ftp.cs.wisc.edu:math-prog/.
- [17] S.A. Gabriel and J.J. More. Smoothing of mixed complementarity problems. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1995. MCS-P541-0995.
- [18] C.B. Garcia and W.I. Zangwill. *Pathways to Solutions Fixed Points, And Equilibria*. Prentice-Hall, Englewood Cliffs, N.J., 1981.
- [19] F. Giannessi, R.W. Cottle, and J.-L. Lions. *Variational Inequalities and Complementarity Problems*. Prentice-Hall, Englewood Cliffs, N.J., 1993.
- [20] M.S. Gowda. On the extended linear complementarity problem. Technical report, Department of Mathematics Statistics, University of Maryland Baltimore County, Baltimore, Maryland, 1994.
- [21] G.J. Habetler and M.M. Kostreva. On a direct algorithm for nonlinear complementarity problems. *SIAM Journal of Control and Optimization*, (16):504–511, 1978.
- [22] S.J. Hammarling, S.P. Datardina, J.J. Du Croz, and M.W. Pont. A proposed specification of blas routines in c. Technical report, NAG Limited, Oxford, 1991.
- [23] P.T. Harker and J.-S. Pang. Finite-dimensional varitional inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. *Mathematical Programming*, (48):161–220, 1990.
- [24] P.T. Harker and B. Xiao. Newton’s method for the nonlinear complementarity problem: A b-differentiable equation approach. *Mathematical Programming*, 48:339–357, 1990.
- [25] W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*. Lecture Notes in Economics and Mathematical Systems 187, (Springer-Verlag, Berlin), 1971.

- [26] G. Isac. *Complementarity Problems*. Springer-Verlag, Berlin, 1991.
- [27] N.H. Josephy. Newton's method for generalized equations. Technical Report 1965, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1979.
- [28] N.H. Josephy. Quasi-newton method for generalized equations. Technical Report 1966, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1979.
- [29] J.E. Dennis Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, N.J., 1983.
- [30] C. Kanzow. A new approach to continuation methods for complementarity problems with uniform p-functions. Technical report, Institute of Mathematics, University of Hamburg, November 1994.
- [31] C. Kanzow and M. Fukushima. A new approach to continuation methods for complementarity problems with uniform p-function. Technical report, Institute of Applied Mathematics, University of Hamburg, Bundesstrasse 55 D-20146 Hamburg, Germany, 1996.
- [32] C. Kanzow and H. Jiang. A continuation method for (strongly) monotone variational inequalities. Technical report, Institute of Applied Mathematics, University of Hamburg, Bundesstrasse 55 D-20146 Hamburg, Germany, 1996.
- [33] M. Kojima, S. Mizuna, and T. Noma. Limiting behavior of trajectories generated by a continuation method for monotone complementarity problems. Technical Report B-199, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1988.
- [34] M. Kojima, S. Mizuna, and T. Noma. A new continuation method for complementarity problems with uniform p-functions. *Mathematical Programming*, (43):107–113, 1989.

- [35] M. Kojima, S. Mizuna, and A. Yoshise. A polynomial-time algorithm for a class of linear complementarity problems. *Mathematical Programming*, (44):1–26, 1989.
- [36] M. Kojima and S. Shindo. Extension of newton and quasi-newton methods to systems of pc equations. *Journal of the Operations Research Society of Japan*, (29):352–374, 1986.
- [37] M.M. Kostreva. *Direct algorithms for complementarity problems*. PhD thesis, Department of Mathematics, Rensselaer Polytechnic Institute, Troy, NY, 1976.
- [38] M.M. Kostreva. Block pivoting methods for solving the complementarity problem. *Linear Algebra and its Applications*, (21):207–215, 1978.
- [39] J. Kreimer and R.Y. Rubinstein. Nondifferentiable optimization via smooth approximation: General analytical approach. *Annals of Operations Research*, (39):97–119, 1992.
- [40] T. De Luca, F. Facchinei, and C. Kanzow. A semismooth equation approach to the solution of nonlinear complementarity problems. Technical report, Institute of Applied Mathematics, University of Hamburg, 1995. Preprint 93.
- [41] K. Madsen and H.B. Nielsen. A finite smoothing algorithm for linear l_1 estimation. *SIAM Journal on Optimization*, (3(2)):223–235, May 1993.
- [42] O.L. Mangasarian. Equivalence of the complementarity problem to a system of nonlinear equations. *SIAM Journal on Applied Mathematics*, (31):89–92, 1976.
- [43] O.L. Mangasarian. Solution of symmetric complementarity problems by iterative methods. *J. Optimization Theory Appl.*, (7,4):465–485, 1977.
- [44] O.L. Mangasarian and R.R. Meyer. Nonlinear perturbation of linear programs. *SIAM J. Control and Optimization*, 17(6):745–752, 1979.

- [45] O.L. Mangasarian and J.-S. Pang. The extended linear complementarity problem. (1188), November 1993. Technical Report, Computer Sciences Department, University of Wisconsin.
- [46] L. Mathiesen. Computation of economic equilibria by a sequence of linear complementarity problems. *Mathematical Programming Study*, (23):144–162, 1985.
- [47] L. Mathiesen. Computational experience in solving equilibrium models by a sequence of linear complementarity problems. *Operations Research*, (33):1225–1250, 1985.
- [48] N. Megiddo, M. Kojima, and S. Mizuna. A general framework of continuation methods for complementarity problems. *Math. of Oper. Res.*, (18):945–963, 1993.
- [49] N. Megiddo, M. Kojima, and T. Noma. Homotopy continuation methods for nonlinear complementarity problems. *Math. of Oper. Res.*, (16):754–774, 1991.
- [50] K.G. Murty. Note on a bard-type scheme for solving the complementarity problem. *Opsearch*, (11):123–130, 1974.
- [51] K.G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Helderman-Verlag, Berlin, 1988.
- [52] T. Noma, M. Kojima, N. Megiddo, and A. Yoshise. *A unified approach to interior point algorithms for linear compleментарity problems*. Springer-Verlag, Berlin, Germany, 1991.
- [53] T. Noma, M. Kojima, and A. Yoshise. Global convergence and detecting infeasibility in interior-point algorithms. Technical report, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1992.
- [54] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970.
- [55] J.S. Pang. Newton’s method for b-differentiable equations. *Mathematics of Operations Research*, (15):331–341, 1990.

- [56] M.C. Pinar and S.A. Zenios. On smoothing exact penalty functions for convex constrained optimization. *SIAM Journal on Optimization*, (4):486–511, 1994.
- [57] P.V. Preckel. A modified newton method for the nonlinear complementarity problem. Paper presented at the ORSA/TIMS Joint National Meeting, Miami Beach, Florida, October 1986.
- [58] L. Qi. Convergence analysis of some algorithms for solving nonsmooth equations. *Mathematics of Operations Research*, (18):227–244, 1993.
- [59] S.M. Robinson. Implicit b-differentiability in generalized equations. Technical Report 2854, Mathematics Research Center, University of Wisconsin, Madison, WI, 1985.
- [60] S.M. Robinson. Local structure of feasible sets in nonlinear programming, part iii: stability and sensitivity. *Mathematical Programming Study*, (30):45–66, 1987.
- [61] S.M. Robinson. An implicit function theorem for b-differentiable functions. Technical report, Department of Industrial Engineering, University of Wisconsin, Madison, WI, 1988.
- [62] S.M. Robinson. Newton's method for a class of nonsmooth functions. Technical report, Department of Industrial Engineering, University of Wisconsin, Madison, WI, 1988.
- [63] S. Smale. Algorithms for solving equations. *Proceedings of the International Congress of Mathematicians*, pages 172–195, 1987.
- [64] P.K. Subramanian. A note on the least two norm solutions of monotone complementarity problems. Technical Report 2845, Mathematics Research Center, University of Wisconsin, Madison, W.I., 1985.
- [65] P. Tseng. An infeasible path-following method for monotone complementarity problem. Technical report, Department of Mathematics, University of Washington, WA 98195, 1994.

- [66] S. Wright and D. Ralph. A superlinear infeasible-interior-point algorithm for monotone complementarity problems. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1993. Preprint MCS-P344-1292.
- [67] Y. Ye. A fully polynomial-time approximation algorithm for computing a stationary point of the general linear complementarity problem. *Mathematics of Operations Research*, (18):334–345, 1993.
- [68] I. Zang. A smoothing-out technique for min-max optimization. *Mathematical Programming*, (19):61–77, 1980.



Appendix A

User's Guide to NCPNMS: A Fortran 77 Package to solve NCP

A.1 Purpose

NCPNMS is a FORTRAN 77 subroutine designed to solve nonlinear complementarity problems. The subroutine can exploit sparsity. Therefore, it is suitable for large problems with very sparse constraint matrices. The target problem and the description of the algorithm are provided in the main body.

A.2 Specification of the Subroutine NCP- NMS

The heading of subroutine NCPNMS is given below.

```
SUBROUTINE NCPNMS(N,MU,SZ,SOL,ITER,FITER)  
INTEGER N,ITER,FITER
```

DOUBLE PRECISION MU, SZ(N), SOL(N)

Now, we give a description of the parameters.

A.3 Description of Parameters

N INTEGER

On entry N must specify the number of variables n . Unchanged on exit.

MU DOUBLE PRECISION

On entry mu must specify first value of mu.

SZ DOUBLE PRECISION ARRAY OF DIMENSION (N)

On entry SZ is the starting vector.

SOL DOUBLE PRECISION ARRAY OF DIMENSION (N)

SOL is the solution vector.

ITER INTEGER

ITER is number of iteration.

FITER INTEGER

FITER is number of inner loop iteration.

In addition to this, user must support two subroutines. First one is CALCF() for formulation of NCP problem, second one is CALCG() for jacobian of NCP problem. Now, we give examples for CALCF and CALCG.

Example for CALCF:

f specifies the function vector.

C compute the function F for problem spatial equilib.;

C Mustafa C. Pinar, Bilkent, Sept. 26, 1995

C input : x .. variables

C Kojima-Josephy

Subroutine Calcf(f,x,n)

Double Precision x(*),f(*)

Integer n,i,j

$$f(1) = 3*x(1)**2 + 2*x(1)*x(2) + 2*x(2)**2 + x(3) + 3*x(4) - 6$$

$$f(2) = 2*x(1)**2 + x(1) + x(2)**2 + 3*x(3) + 2*x(4) - 2$$

$$f(3) = 3*x(1)**2 + x(1)*x(2) + 2*x(2)**2 + 2*x(3) + 3*x(4) - 1$$

$$f(4) = x(1)**2 + 3*x(2)**2 + 2*x(3) + 3*x(4) - 3$$

Return

End

Example for CALCG:

DF1 specifies number of row of entries of Jacobian ∇f matrix.

DF2 specifies number of column of entries of Jacobian ∇f matrix.

VDF specifies value of entries of Jacobian ∇f matrix.

ne specifies number of nonzero entries of matrix.

DH1 specifies number of row of entries of ∇H matrix.

DH2 specifies number of column of entries of ∇H matrix.

VDH specifies value of entries of ∇H matrix.

C Compute the Jacobian DF of F

C input : x .. variables

C Kojima-Josephy

Subroutine CALCG(DF,x,n)

Integer n,i,j

Double Precision DF(n,n),x(*)

C First row

$$VDF(1) = 6*x(1) + 2*x(2)$$

$$VDF(2) = 2*x(1) + 4*x(2)$$

$$VDF(3) = 1$$

$$VDF(4) = 3$$

C Second row

$$\text{VDF}(5) = 4 * x(1) + 1$$

$$\text{VDF}(6) = 2 * x(2)$$

$$\text{VDF}(7) = 3$$

$$\text{VDF}(8) = 2$$

C Third row

$$\text{VDF}(9) = 6 * x(1) + x(2)$$

$$\text{VDF}(10) = x(1) + 4 * x(2)$$

$$\text{VDF}(11) = 2$$

$$\text{VDF}(12) = 3$$

C Fourth row $\text{VDF}(13) = 2 * x(1)$

$$\text{VDF}(14) = 6 * x(2)$$

$$\text{VDF}(15) = 2$$

$$\text{VDF}(16) = 3$$

ne=16

$$\text{DF1}(1) = 1$$

$$\text{DF2}(1) = 1$$

$$\text{DF1}(2) = 1$$

$$\text{DF2}(2) = 2$$

$$\text{DF1}(3) = 1$$

$$\text{DF2}(3) = 3$$

$$\text{DF1}(4) = 1$$

$$\text{DF2}(4) = 4$$

$$\text{DF1}(5) = 2$$

$$\text{DF2}(5) = 1$$

$$\text{DF1}(6) = 2$$

$$\text{DF2}(6) = 2$$

$$\text{DF1}(7) = 2$$

$$\text{DF2}(7) = 3$$

$$\text{DF2}(8) = 4$$

$$\text{DF1}(9) = 3$$

$$\text{DF2}(9) = 1$$

$$\text{DF1}(10) = 3$$

```

DF2(10) = 2
DF1(11) = 3
DF2(11) = 3
DF1(12) = 3
DF2(12) = 4
DF1(13) = 4
DF2(13) = 1
DF1(14) = 4
DF2(14) = 2
DF1(15) = 4
DF2(15) = 3
DF1(16) = 4
DF2(16) = 4
k=1
Do i=1,n
Do j=1,n
DH1(k)=i
DH2(k)=j
VDH(k)=0
k=k+1
End do
End do
Return
End

```

A.4 UMFPACK2

UMFPACK2 Version 2.0 is a package for solving systems of sparse linear systems, $Ax = b$, where A is sparse and can be unsymmetric. It is written in ANSI Fortran 77. There are options for choosing a good pivot order, factorizing a subsequent matrix with the same pivot order and nonzero pattern

as a previously factorized matrix, and solving systems of linear equations with the factors (with A, L, or U, or their transposes). Iterative refinement, with sparse backward error estimates, can be performed. Both single and double precision routines are available.

A.5 BLAS

In 1973, Hanson, Krogh and Lawson described the advantages of adopting a set of basic routines for problems in linear algebra. The original basic linear algebra subprograms, commonly referred to as the BLAS or, in view of subsequent developments, the Level 1 BLAS, have been very successful and have been used in a wide range of software including LINPACK. Subsequently a set of Level 2 BLAS for matrix-vector operations, motivated by the development of vector-processing machines, was proposed by Dongarra. More recently a set of Level 3 BLAS for matrix-matrix operations, motivated by the development of hierarchical memory and parallel machines, has been proposed by Dongarra [61].

A.6 Harwell Routines

The MA38 Package in the Harwell Subroutine Library (HSL) has equivalent functionality (and identical calling interface) as UMFPACK. It is available for commercial use. Technical reports, information on HSL, and matrices are available via the World Wide Web at <http://www.cis.rl.ac.uk/struct/ARCD/NUM.html>, or by anonymous ftp at [seamus.cc.rl.ac.uk/pub](ftp://seamus.cc.rl.ac.uk/pub).

Appendix B

Fortran 77 Code of Algorithm



B1. CODE OF SOFTWARE FOR INTERIOR PLUS-SMOOTH FUNCTION

```

C *****
C Nonlinear Complementary Problem
C 10.1996
C Ali Erkan
C Industrial Engineering Department
C Bilkent University
C *****
C
C This program uses a continuation algorithm which consists smoothing
C methods developed by Bintong Chen, Patrick T. Harker and Mustafa C. Pinar.
C Program gets function model and gradient formulation of problem as
C subroutines namely calcf.f and calcg.f.
C User must support this subroutines, starting point, mu, n, nemax, Maxiter.
C Program uses UMFPACK2 which is developed by Timothy A. Davis and Iain
C S. Duff in 1995 to solve sparse linear systems ( $Ax = b$ ). Also it is written in
C ANSI FORTAN 77.

```

Subroutine NCPNMS(n,nemax,mu,z,dz,iter,sum,)

```

Integer Optflag, Maxiter
Parameter (Maxiter = 50)
Double Precision W(Maxiter),WW(Maxiter),Phi(Maxiter),
$ R(Maxiter),M(Maxiter),E(Maxiter),Lsc(Maxiter)
Double Precision z(n),xp(n),f(n),
$ xm(n),h(n),y(n),dz(n)
Integer i,iter,n,ne,DH1(nemax),DH2(nemax),DF1(nemax),DF2(nemax)
Double Precision dot,func1,eps,phi0,arw,mueps,mu,sum
Double Precision uvec(n),VDH(nemax),
$ VDF(nemax),dp(n),dpm(n),grad(n),
$ ff(n),lamda(n),zd(n),yd(n),err(n)
Real t1(2),t2(2),tstart,tend,cputime

```

```

C *****
eps=2.204E-16
optflag=1
Do i=1,maxiter
    W(i)=0
    WW(i)=0
    Phi(i)=0
    R(i)=0

```

```

        M(i)=0
        E(i)=0
        Lsc(i)=0
    End do
    iter=0
    Do i=1,n
        y(i)=-z(i)
    End do

C    Calculate plus-smooth function  $p(z, \mu_a)$  ( $a=1$ ) for starting point.
    Call CALCP(xp,z,mu,n)

C    Calculate function value  $f(p(z, \mu_a))$  ( $a=1$ ) for starting point.
    Call CALCF(f,xp,n)

C    Calculate plus-smooth function  $p(-z, \mu_a)$  ( $a=1$ ) for starting point.
    Call CALCP(xm,y,mu,n)

C    Calculate 2-norm of  $H(z, \mu) = (1-\mu)*f(p(z, \mu_a)) - p(-z, \mu_a) + ub$ 
C    for starting point.  $a=1, b=1$ .
    Call NORMH(func1,h,xm,f,mu,n,uvec)

C    Initialization for line search.
    phi0=func1
    arw=phi0
    w(1)=arw
    mueps=eps

C    Starting time of CPU.
    tstart = etime(t1)
C    *****

C    *****
C    Until reach termination point such that error < 0.000001 continue
    Do While ((optflag.eq.1) .and. (Iter.lt.Maxiter))
        Iter=Iter+1
        M(Iter)=Mu

C    Damped-Newton's Method as corrector
    Call NEWTON(func1,h,dz,dot,z,mu,n,DH1,DH2,VDH,
$    DF1,DF2,VDF,y,dp,dpm,grad,xp,f,xm,uvec,ne)
        phi(iter) = func1

C    Nonmonotone line search
C    Calculate step length
    Call NONMON(iter,phi,phi0,func1,arw,ww,w)

C    Line search, control procedure

```

```

    Call ARMIJO(z,iter,n,mu,arw,dot,dz,lsc,ff,
$   lamda,zd,yd,h,xp,f,xm,uvec)

C   Control subroutine whether optimal point is reached or not.
    Call OPT(eps,z,mu,n,optflag,err,xp,f)

C   If optimal point is not reached and mu become very small number
C   then stop program, optimal point is not reached.
    If (mu.le.mueps) Then
        optflag=0
    Endif

C   Reducing mu.
    mu=mu/10
End do

C   *****

C   *****
C   End time of CPU.
    tend=etime(t2)
C   CPU time.
    cputime = tend - tstart
    Write(*,*) 'Iterations', Iter

C   Total number of inner loop iterations.
    sum=0
    Do i=1,iter
        sum=sum+lsc(i)
    End do

C   *****

    Write(*,*) 'Inner loop iteration = ', Sum
    Write(*,*) ' Cpu Time = ',cputime
End

C   *****
C   NEWTON.F
C   Compute the Newton step
C   Input: z,mu,n
C   Output: dz,grad,dot
C   *****

Subroutine NEWTON(func1,h,dz,dot,z,mu,n,
$   DH1,DH2,VDH,DF1,DF2,VDF,y,dp,dpm,grad,xp,f,xm,uvec,ne)
Double Precision func1,h(*),dz(*),dot,z(*),mu
Integer n,ne,DF1(*),DF2(*),DH1(*),DH2(*)

```

Double Precision VDF(*),VDH(*),y(*),dp(*),dpm(*)
 Double Precision grad(*),xp(*),f(*),xm(*)
 Integer i,j
 Double Precision uvec(*)

dot=0
 Do i=1,n
 y(i)=-z(i)
 grad(i)=0
 End do

- C Calculate d(z) of plus-smooth function p(z,ua).
 Call PPRIME(dp,z,mu,n)
- C Calculate d(-z) of plus-smooth function p(-z,mua).
 Call PPRIME(dpm,y,mu,n)
- C Calculate plus-smooth function p(z,mua).
 Call CALCP(xp,z,mu,n)
- C Calculate gradient Matrix of function f(p(z,mua)).
 Call CALCG(DF1,DF2,VDF,DH1,DH2,VDH,xp,n,ne)
- C Calculate Jacobian dH(z,mu)
 Call JACOB(DH1,DH2,VDH,DF1,DF2,VDF, dp, dpm, mu,n,ne)
- C Calculate function value f(p(z,mua)).
 Call CALCF(f,xp,n)
- C Calculate plus-smooth function p(-z,mua).
 Call CALCP(xm,y,mu,n)
- C Calculate 2-norm of $H(z,\mu) = (1-\mu)f(p(z,ua)) - p(-z,mua) + \mu b$
 a=1, b=1.
 Call NORMH(func1,h,xm,f,mu,n,uvec)
- C Solve $DH * dz = h$ and find dz by using UMFPACK2.
 Call CALLUMF(n,DH1,DH2,VDH,dz,h,ne)
- C Calculate $grad = DH' * h'$
 Do i=1,n
 Do j=1,ne
 If (i.eq.DH2(j)) then
 grad(i)=grad(i)+VDH(j)*h(DH1(j))
 End if
 End do
 End do

```

C      Calculate dot = -grad'*dz
      Do i=1,n
            dot=dot+grad(i)*dz(i)
      End do
      dot=-dot

      write(*,*) 'dot= ',dot
      Return
End

```

```

C      *****
C      CALLUMF.F
C      Input and output for UMFPACK2 to solve DH * dz =h.
C      UMFPACK2: Sparse linear systems solver
C      By Timothy A. Davis and Iain S. Duff.
C      *****

```

```

      Subroutine CALLUMF(n,DH1,DH2,VDH,dz,h,ne)
      Integer n,ne,DH1(*),DH2(*)
      Double Precision VDH(*),dz(*),h(*)

```

```

C      This part is needed by UMFPACK2.
      Integer nmax,nemax,lvalue,index
      Parameter(nmax=5000,nemax=100000,lvalue=400000,index=700000)
      Integer keep(20),index(lindex),info(40),i,j,icntl(20),
$      ai1(nemax),ai2(nemax)
      Double Precision b(nmax),w(4*nmax),value(lvalue),
$      cntl(10),rinfo(20)

```

```

C      Nonzero values of dH(z,mu) are converted for UMFPACK2.
      Do i=1,ne
            Index(i)=DH1(i)
            Index(ne+i)=DH2(i)
            Value(i) = VDH(i)
      End do

```

```

C      If all entries are zero.
      If (ne.eq.0) then
            Do i=1,n
                  dz(i) = -h(i)
            End do

```

```

C      If not all entries are zero.
      Else

```

```

C      Initialization part of UMFPACK2.

```

```

Call UMD2IN(icntl,cntl,keep)
Icntl(4)=0

C      Control for whether matrix dH(z,mu) is singular or not.
      Call UMD2FA(n,ne,0,.false.,lvalue,lindex,value,index,
$      keep,cntl,icntl,info,rinfo)
C      If there is a mistake in matrix then stop.
      If (info(1).lt.0) stop

C      If matrix is singular.
      If (info(1).ne.0) then
        Do i=1,n
          dz(i) = -h(i)
        End do

C      If matrix is nonsingular.
      Else
        Icntl(8)=10
        Do i=1,n
          b(i)=-h(i)
        End do
C      Solve linear system if dH(z,mu) is nonsingular.
      Call UMD2SO(n,0,.false.,lvalue,lindex,value,index,keep,
$      b,dz,w,cntl,icntl,info,rinfo)
      if (info(1).lt.0) stop
      End if
      End if
End

C      *****
C      NONMON.F
C      Nonmonotone line search
C      Find minimum value 2-norm ||H(z,mu)||
C      to calculate step length alpha.
C      *****

Subroutine NONMON(iter,phi,phi0,func1,arw,ww,w)
Integer iter
Double Precision phi(*),phi0,func1,arw,ww(*),w(*)

Integer kk,kkm5,i
Double Precision ww1

If (Iter.gt.1) then
  kk = Iter
Endif

```

```

      If (kk.gt.5) then
      kkm5 = kk - 5
      Else
      kkm5 = 1
      Endif
      ww1 = phi(kkm5)
      Do 100 i=kkm5+1,kk
      If (ww1.gt.phi(i)) then
      ww1 = phi(i)
      Endif
100  Continue
      WW(iter) = ww1
      If (kk.lt.5) then
      If (ww1.gt.phi0) then
      ww1 = phi0
      Endif
      Endif
      If (Func1.gt.ww1) then
      arw = func1
      Endif
      W(iter) = arw
      Return
      End

```

```

C *****
C  ARMIJO.F
C  Perform the Armijo linesearch
C  Input:z,mu,n,arw,dot
C  Output:ff(),lsc(),z,delf,rhs
C *****

```

```

$ Subroutine ARMIJO(z,iter,n,mu,arw,dot,dz,lsc,
  ff,lamda,zd,yd,h,xp,f,xm,uvec)
  Double Precision z(*),mu,arw,dot,dz(*),lsc(*)
  Integer iter,n

  Double Precision func2,deltam,delta,lsitmx,sigma,delf,rhs
  Integer m,i,lsiter
  Double Precision ff(*),lamda(*),zd(*)
  Double Precision yd(*),h(*),xp(*),f(*),xm(*)
  Double Precision uvec(*)

```

```

C  For Armijo line search experiment values.
C  This values can be changed.
  delta = .5
  lsitmx = 20
  m = 0

```

```

lsiter = 1
sigma = .4

deltam = delta**m
lamda(lsiter) = deltam

C   Direction  $z(k+1) = z(k) + \alpha(k)*d(k)$ 
    Do i=1,n
      zd(i) = z(i) + deltam*dz(i)
      yd(i) = -zd(i)
    End do

C   Calculate plus-smooth function  $p(zd, \mu)$ .
    Call CALCP(xp,zd,mu,n)

C   Calculate function value  $f(p(zd, \mu))$ .
    Call CALCF(f,xp,n)

C   Calculate plus-smooth function  $p(-zd, \mu)$ .
    Call CALCP(xm,yd,mu,n)

C   Calculate 2-norm of  $H(zd, \mu)$ .
    Call NORMH(func2,h,xm,f,mu,n,uvec)
    ff(lsiter) = func2
    rhs = 2*sigma*deltam*dot
    delf = arw - func2

C   Continue until armijo line search finished.
    Do while (delf.lt.rhs )
      m=m + 1
      lsiter = lsiter + 1
      deltam = delta**m
      lamda(lsiter) = deltam
      Do 200 i=1,n
        zd(i) = z(i) + deltam*dz(i)
        yd(i) = -zd(i)
      200 Continue

C   Calculate plus-smooth function  $p(zd, \mu)$ .
    Call CALCP(xp,zd,mu,n)

C   Calculate function value  $f(p(zd, \mu))$ .
    Call CALCF(f,xp,n)

C   Calculate plus-smooth function  $p(-zd, \mu)$ .
    Call CALCP(xm,yd,mu,n)

C   Calculate 2-norm of  $H(zd, \mu)$ .

```

```

Call NORMH(func2,h,xm,f,mu,n,uvec)
ff(lsite) = func2
rhs = 2*sigma*deltam*dot
delf = arw - func2
End do
lsc(iter) = lsite

```

```

C   New point after line search.
    Do i=1,n
      z(i) = z(i) + deltam*dz(i)
    End do
    Return
  End

```

```

C   *****
C   OPT.F
C   Check optimality
C   *****

```

```

Subroutine OPT(eps,z,mu,n,optflag,err,xp,f)
Double Precision eps,z(*),mu
Integer n,optflag

Double Precision ecomp,nerr,err(*),xp(*),f(*)
Integer i

```

```

C   Calculate plus-smooth function p(z,mu).
    Call CALCP(xp,z,mu,n)

```

```

C   Calculate function value f(p(z,mu)).
    Call CALCF(f,xp,n)

```

```

    nerr=0
    Do i=1,n
      err(i) = 1/eps
    End do

```

```

C   Calculate minimum of error.
    Do i=1,n
      ecomp = min(xp(i),f(i))
      err(i) = min(ecomp,err(i))
      nerr=nerr+err(i)**2
    End do

```

```

C   2-norm of error.
    nerr=nerr**0.5
    Write(2,*)' Error in opt ', nerr
    if (nerr.le.1e-6) then

```

```

Write(*,*) '*****'
Write(*,*) '-----> Successful Termination <-----'
Write(*,*) '*****'
optflag = 0
Endif
Return
End

```

```

C *****
C CALCP.F
C Compute the interior smooth function
C *****

```

```

Subroutine Calcp(p,r,mu,n)
Double Precision p(*),r(*),mu
Integer n

```

```

Double Precision Term1
Integer i

```

```

Do i=1,n
p(i) = 0
End do

```

```

C Plus-smooth function.
Do i=1,n
Term1 = (r(i)**2 + 4*mu)**0.5
p(i) = 0.5*(r(i) + term1)
End do
Return
End

```

```

C *****
C NORMH.F
C Compute the vector function h and its norm;
C nh,h:output
C xm,f,mu,n:input
C *****

```

```

Subroutine NORMH(nh,h,xm,f,mu,n,uvec)
Integer n
Double Precision nh,h(*),xm(*),f(*)

```

```

Integer i
Double Precision uvec(*),mu

```

```

      nh=0
      Do i=1,n
C      mu*b, b=1.
      uvec(i)=mu*1

C      Calculate  $H(z,\mu) = (1-\mu)*f(p(z,\mu)) - p(-z,\mu) + \mu$ .
      h(i) = (1-mu)*f(i) - xm(i) + uvec(i)

C      Calculate 2-norm of H(z,mu)
      nh=nh+h(i)**2

      End do
      Return
      End

C      *****
C      PPRIME.F
C      Compute the interior point smooth function's gradient
C      *****

      Subroutine PPRIME(dp,r,mu,n)
      Double Precision dp(*),r(*),mu
      Integer n

      Integer i
      Double Precision term1, term2

C      Calculate gradient of smooth function.
      Do i=1,n
      dp(i)=0
      term1 = (r(i)**2 + 4*mu)**0.5
      term2 = r(i)/term1
      dp(i) = 0.5*(1+term2)
      End do
      Return
      End

C      *****
C      JACOB.F
C      Compute the Jacobian matrix DH;
C      *****

C      VDH: value of entries of dH(z,mu).
C      DH1: number of row of dH(z,mu).
C      DH2: number of column of dH(z,mu).
C      VDF: value of entries of df(p(z,mu)).

```

- C DF1: number of row of $df(p(z,\mu))$.
 C DF2: number of column of $df(p(z,\mu))$.

Subroutine JACOB(DH1,DH2,VDH,DF1,DF2,VDF,dp,dpm,mu,n,ne)
 Integer n,ne,DH1(*),DH2(*),DF1(*),DF2(*)
 Double Precision VDH(*),VDF(*),dp(*),dpm(*),mu

Integer i,j

- C Calculate $dH(z,\mu) = (1-\mu)*df(p(z,\mu))*diag\{p'(z,\mu)\} - diag\{p'(-z,\mu)\}$.
 Do i=1,n
 Do j=1,ne
 If (i.eq.DF2(j)) then
 $VDH(j) = VDH(j) + (1-\mu)*VDF(j)*dp(i)$
 End if
 End do
End do
Do j=1,ne
 If (DH1(j).eq.DH2(j)) then
 $VDH(j) = VDH(j) + dpm(DH1(j))$
 End if
End do
Return
End

- C *****
 C CALCF.F
 C Compute the function F for problem spatial equilib.
 C input : x .. variables
 C Kojima-Josephy
 C *****

Subroutine Calcf(f,x,n)
 Double Precision x(*),f(*)
 Integer n,i,j

$f(1) = 3*x(1)**2 + 2*x(1)*x(2) + 2*x(2)**2 + x(3) + 3*x(4) - 6$
 $f(2) = 2*x(1)**2 + x(1) + x(2)**2 + 3*x(3) + 2*x(4) - 2$
 $f(3) = 3*x(1)**2 + x(1)*x(2) + 2*x(2)**2 + 2*x(3) + 3*x(4) - 1$
 $f(4) = x(1)**2 + 3*x(2)**2 + 2*x(3) + 3*x(4) - 3$
 Return
 End

- C *****
 C CALCG.F
 C Compute the Jacobian DF of F for traffic equilibrium C CALCGTR.F

```

C      Input : x .. variables
C      Kojima-Josephy
C      ****

Subroutine CALCG(DF1,DF2,VDF,DH1,DH2,VDH,x,n,ne)
Integer n,i,j,ne,DF1(*),DF2(*),DH1(*),DH2(*),k
Double Precision VDF(*),x(*),VDH(*)

C      Do i=1,n
C      Do j=1,n
C      DF(i,j)=0
C      End do
C      End do
C      First row
VDF(1) = 6*x(1) + 2*x(2)
VDF(2) = 2*x(1) + 4*x(2)
VDF(3) = 1
VDF(4) = 3
C      Second row
VDF(5) = 4*x(1) + 1
VDF(6) = 2*x(2)
VDF(7) = 3
VDF(8) = 2
C      Third row
VDF(9) = 6*x(1) + x(2)
VDF(10) = x(1) + 4*x(2)
VDF(11) = 2
VDF(12) = 3
C      Fourth row
VDF(13) = 2*x(1)
VDF(14) = 6*x(2)
VDF(15) = 2
VDF(16) = 3
ne=16
DF1(1) = 1
DF2(1) = 1
DF1(2) = 1
DF2(2) = 2
DF1(3) = 1
DF2(3) = 3
DF1(4) = 1
DF2(4) = 4
DF1(5) = 2
DF2(5) = 1
DF1(6) = 2
DF2(6) = 2
DF1(7) = 2
DF2(7) = 3

```

```
DF1(8) = 2
DF2(8) = 4
DF1(9) = 3
DF2(9) = 1
DF1(10) = 3
DF2(10) = 2
DF1(11) = 3
DF2(11) = 3
DF1(12) = 3
DF2(12) = 4
DF1(13) = 4
DF2(13) = 1
DF1(14) = 4
DF2(14) = 2
DF1(15) = 4
DF2(15) = 3
DF1(16) = 4
DF2(16) = 4
k=1
Do i=1,n
  Do j=1,n
    DH1(k)=i
    DH2(k)=j
    VDH(k)=0
    k=k+1
  End do
End do
Return
End
```

B2. CODE OF SOFTWARE FOR PIECEWISE QUADRATIC PLUS-SMOOTH FUNCTION WITH REDUCED JACOBIAN MATRIX

```

C *****
C Nonlinear Complementary Problem
C 7.1997
C Ali Erkan
C Industrial Engineering Department
C Bilkent University
C *****
C
C This program uses a continuation algorithm which consists smoothing
C methods developed by Bintong Chen, Patrick T. Harker and Mustafa C. Pinar.
C Program gets function model and gradient formulation of problem as
C subroutines namely calcf.f and calcg.f.
C User must support this subroutines, starting point, mu, n, nemax, Maxiter.
C Program uses UMFPACK2 which is developed by Timothy A. Davis and Iain
C S. Duff in 1995 to solve sparse linear systems ( $Ax = b$ ). Also it is written in
C ANSI FORTAN 77.

```

```

Subroutine NCPNMS(n,nemax,mu,z,dz,iter,sum)

```

```

Integer Optflag, Maxiter

```

```

Parameter (Maxiter = 50)

```

```

Double Precision W(Maxiter),WW(Maxiter),Phi(Maxiter),

```

```

$ R(Maxiter),M(Maxiter),E(Maxiter),Lsc(Maxiter)

```

```

Double Precision z(n),xp(n),f(n),

```

```

$ xm(n),h(n),y(n),dz(n)

```

```

Integer i,iter,n,ne,DH1(nemax),DH2(nemax),DF1(nemax),DF2(nemax),

```

```

$ s1(n),s2(n),s3(n),n1,n2,n3,nDH

```

```

Double Precision dot,func1,eps,phi0,arw,mueps,mu,sum

```

```

Double Precision uvec(n),VDH(nemax),

```

```

$ VDF(nemax),dp(n),dpm(n),grad(n),

```

```

$ ff(n),lamda(n),zd(n),yd(n),err(n)

```

```

Real t1(2),t2(2),tstart,tend,cputime

```

```

C *****

```

```

eps=2.204E-16

```

```

optflag=1

```

```

Do i=1,maxiter

```

```

W(i)=0

```

```

WW(i)=0

```

```

Phi(i)=0

```

```

R(i)=0

```

```

M(i)=0

```

```

E(i)=0

```

```

Lsc(i)=0

```

```

End do

iter=0
Do i=1,n
    y(i)=-z(i)
End do

C    Calculate plus-smooth function  $p(z,\mu)$  of starting point.
    Call CALCP(xp,z,mu,n)

C    Calculate function value  $f(p(z,\mu))$  of starting point.
    Call CALCF(f,xp,n)

C    Calculate plus-smooth function  $p(-z,\mu)$ .
    Call CALCP(xm,y,mu,n)

C    Calculate 2-norm of  $H(z,\mu)$ .
    Call NORMH(func1,h,xm,f,mu,n,uvec)

    phi0=func1
    arw=phi0
    w(1)=arw
    mueps=eps
    tstart = etime(t1)

C    *****
C    Continue until termination point is reached, error < 0.000001.
    Do While ((optflag.eq.1) .and. (Iter.lt.Maxiter))
        Iter=Iter+1
        M(Iter)=Mu

C    Damped-Newton's Method as corrector.
        Call NEWTON(func1,h,dz,dot,z,mu,n,DH1,DH2,VDH,
$    DF1,DF2,VDF,y,dp,dpm,grad,xp,f,xm,uvec,ne,
$    s1,s2,s3,n1,n2,n3,nDH)
        phi(iter) = func1

C    Nonmonotone line search
C    Calculate step length
        Call NONMON(iter,phi,phi0,func1,arw,ww,w)

C    Line search, control procedure
        Call ARMIJO(z,iter,n,mu,arw,dot,dz,lsc,ff,
$    lamda,zd,yd,h,xp,f,xm,uvec)

C    Control subroutine whether optimal point is reached or not.
        Call OPT(eps,z,mu,n,optflag,err,xp,f)

```

```

C      If optimal point is not reached and mu become very small number
C      then stop program, optimal point is not reached.
      If (mu.le.mueps) Then
          optflag=0
      Endif

C      Reducing mu.
      mu=mu/10
      End do

C      *****

C      *****
C      End time of CPU.
      tend=etime(t2)
C      CPU time.
      cputime = tend - tstart

C      Total number of inner loop iterations.
      sum=0
      Do i=1,iter
          sum=sum+lsc(i)
      End do
C      *****

      Write(*,*) 'Iterations', Iter
      Write(*,*) 'Inner loop iteration = ', Sum
      Write(*,*) ' Cpu Time = ',cputime
End

C      *****
C      NEWTON.F
C      Compute the Newton step
C      Input: z,mu,n
C      Output: dz,grad,dot
C      *****

      Subroutine NEWTON(func1,h,dz,dot,z,mu,n,
$      DH1,DH2,VDH,DF1,DF2,VDF,y,dp,dpm,grad,xp,f,xm,uvec,ne,
$      s1,s2,s3,n1,n2,n3,nDH)
      Double Precision func1,h(*),dz(*),dot,z(*),mu
      Integer n,ne,DF1(*),DF2(*),DH1(*),DH2(*)
      Integer s1(*),s2(*),s3(*),n1,n2,n3,nDH

      Double Precision VDF(*),VDH(*),y(*),dp(*),dpm(*)
      Double Precision grad(*),xp(*),f(*),xm(*)
      Integer i,j

```

Double Precision uvec(*)

dot=0

Do i=1,n

 y(i)=-z(i)

 grad(i)=0

End do

C Calculate d(z) of plus-smooth function p(z,mu).
 Call PPRIME(dp,z,mu,n)

C Arrange for alpha,beta,gamma.

C Find index whose dp-value is 1 for alpha.

C Find index whose dp-value is 0 for gamma.

C Find index whose dp-value is between 1 and 0 for beta.

 Call ZPRIME(dp,n,s1,s2,s3,n1,n2,n3)

C Calculate d(-z) of plus-smooth function p(-z,mu).

 Call PPRIME(dpm,y,mu,n)

C Calculate plus-smooth function p(z,mu).

 Call CALCP(xp,z,mu,n)

C Calculate gradient Matrix of function f(p(z,mu)).

 Call CALCG(DF1,DF2,VDF,DH1,DH2,VDH,xp,n,ne)

C Calculate Jacobian dH(z,mu)

 Call JACOB(DH1,DH2,VDH,DF1,DF2,VDF,dp,dpm,mu,n,

\$ ne,s1,s2,s3,n1,n2,n3,nDH)

C Calculate function value of f(p(z,mu)).

 Call CALCF(f,xp,n)

C Calculate plus-smooth function p(-z,mu).

 Call CALCP(xm,y,mu,n)

C Calculate 2-norm of $H(z,mu) = (1-mu)f(p(z,mu)) - p(-z,mu) + mu b$

C a=1, b=1.

 Call NORMH(func1,h,xm,f,mu,n,uvec)

C Solve $DH * dz = h$ and find dz by using UMFPACK2.

 Call CALLUMF(n,DH1,DH2,VDH,dz,h,ne,s1,s2,s3,n1,n2,

\$ n3,nDH,DF1,DF2,VDF,dp,mu)

C Calculate grad = $DH' * h'$

 Do i=1,n

 Do j=1,ne

 If (i.eq.DH2(j)) then

```

                                grad(i)=grad(i)+VDH(j)*h(DH1(j))
                        End if
                End do
        End do

C      Calculate dot = -grad'*dz
        Do i=1,n
                dot=dot+grad(i)*dz(i)
        End do
        dot=-dot
        write(*,*) 'dot= ',dot
        Return
End

C      *****
C      CALLUMF.F
C      Input and output for UMFPACK2 to solve DH * dz =h.
C      UMFPACK2: Sparse linear systems solver
C      By Timothy A. Davis and Iain S. Duff.
C      *****

        Subroutine CALLUMF(n,DH1,DH2,VDH,dz,h,ne,s1,s2,s3,n1,n2,
$      n3,nDH,DF1,DF2,VDF,dp,mu)
        Integer n,ne,DH1(*),DH2(*),s1(*),s2(*),s3(*),
$      n1,n2,n3,nDH,DF1(*),DF2(*)
        Double Precision VDH(*),dz(*),h(*),VDF(*),dp(*),mu

C      This part is needed by UMFPACK2.
        Integer nmax,nemax,lvalue,lindex
        Parameter(nmax=5000,nemax=100000,lvalue=400000,lindex=700000)
        Integer keep(20),index(lindex),info(40),i,j,icntl(20),
$      ai1(nemax),ai2(nemax),k,l
        Double Precision b(nmax),x(nmax),w(4*nmax),value(lvalue),
$      cntl(10),rinfo(20)

C      Nonzero values of alpha-beta part of dH(z,mu)
C      are converted for UMFPACK2.
        Do i=1,nDH
                Index(i)=DH1(i)
                Index(nDH+i)=DH2(i)
                Value(i) = VDH(i)
        End do

C      If all entries are zero.
        If (ne.eq.0) then
                Do i=1,n
                        dz(i) = -h(i)

```

```

                End do
C      If not all entries are zero.
      Else

C      Initialization part of UMFPACK2.
      Call UMD2IN(icntl,cntl,keep)
      Icntl(4)=0

C      Control for whether matrix dH(z,mu) is singular or not.
      Call UMD2FA(n1+n2,nDH,0,.false.,lvalue,lindex,value,index,
$      keep,cntl,icntl,info,rinfo)
C      If there is a mistake in matrix then stop
      If (info(1).lt.0) stop

C      If matrix is singular
      If (info(1).ne.0) then
        Do i=1,n
          dz(i) = - h(i)
        End do

C      If matrix is nonsingular
      Else

C      Alpha part
      Do i=1,n1
        b(i) = -h(s1(i))
      End do

C      Beta part
      Do i=n1+1,n1+n2
        b(i) = - h(s2(i-n1))
      End do

C      Gamma part
      Do i=n1+n2+1,n1+n2+n3
        b(i) = - h(s3(i-n1-n2))
      End do
      Icntl(8)=10

C      Solve linear system if dH(z,mu) is nonsingular.
      Call UMD2SO(n1+n2,0,.false.,lvalue,lindex,value,index,keep,
$      b,x,w,cntl,icntl,info,rinfo)
      if (info(1).lt.0) stop

C      Gamma part dz(n2+1) - - dz(n3)
      Do i=n1+n2+1,n1+n2+n3
        x(i) = 0
      End do

```

```

Do k=1,ne
Do i=1,n1
Do j=1,n3
If (DF1(k).eq.s3(i) .and. s1(j).eq.DF2(k)) then
x(n1+n2+i) = x(n1+n2+i) + VDF(k)*x(j)
End if
End do
Do j=n1+1,n1+n2
If (s3(i).eq.DF1(k) .and. s2(j).eq.DF2(k)) then
x(n1+n2+i) = x(n1+n2+i) + VDF(k)*dp(s2(j))*x(j)
End if
End do
End do
Do i=n1+n2+1,n1+n2+n3
x(i) = b(i) - (1-mu)*x(i)
End do
Do i=1,n1
dz(s1(i)) = x(i)
End do
k=1
Do i=n1+1,n2
dz(s2(k)) = x(i)
k=k+1
End do
l=1
Do i=n1+n2+1,n1+n2+n3
dz(s3(l)) = x(i)
l=l+1
End do
End if
End

```

```

C *****
C NONMON.F
C Nonmonotone line search
C Find minimum value 2-norm ||H(z,mu)||
C to calculate step length alpha.
C *****

```

```

Subroutine NONMON(iter,phi,phi0,func1,arw,ww,w)
Integer iter
Double Precision phi(*),phi0,func1,arw,ww(*),w(*)

Integer kk, kkm5,i
Double Precision ww1

```

```

    If (Iter.gt.1) then
    kk = Iter
    Endif
    If (kk.gt.5) then
    kkm5 = kk - 5
    Else
    kkm5 = 1
    Endif
    ww1 = phi(kkm5)
    Do 100 i=kkm5+1,kk
    If (ww1.gt.phi(i)) then
    ww1 = phi(i)
    Endif
100 Continue
    WW(iter) = ww1
    If (kk.lt.5) then
    If (ww1.gt.phi0) then
    ww1 = phi0
    Endif
    Endif
    If (Func1.gt.ww1) then
    arw = func1
    Endif
    W(iter) = arw
    Return
    End

```

```

C *****
C ARMIJO.F
C Perform the Armijo linesearch
C Input:z,mu,n,arw,dot
C Output:ff(),lsc(),z,delf,rhs
C *****

```

```

Subroutine ARMIJO(z,iter,n,mu,arw,dot,dz,lsc,
$ ff,lamda,zd,yd,h,xp,f,xm,uvec)
Double Precision z(*),mu,arw,dot,dz(*),lsc(*)
Integer iter,n

Double Precision func2,deltam,delta,lsitmx,sigma,delf,rhs
Integer m,i,lsiter
Double Precision ff(*),lamda(*),zd(*)
Double Precision yd(*),h(*),xp(*),f(*),xm(*)
Double Precision uvec(*)

```

```

C   For Armijo line search experiment values.
C   This values can be changed.
    delta = .5
    lsitmx = 20
    m = 0
    lsiter = 1
    sigma = .4

    deltam = delta**m
    lamda(lsiter) = deltam

C   Direction  $z(k+1) = z(k) + \alpha(k)*d(k)$ 
    Do i=1,n
    zd(i) = z(i) + deltam*dz(i)
    yd(i) = -zd(i)
    End do

C   Calculate plus-smooth function  $p(zd, \mu)$ .
    Call CALCP(xp,zd,mu,n)

C   Calculate function value  $f(p(zd, \mu))$ .
    Call CALCF(f,xp,n)

C   Calculate plus-smooth function  $p(-zd, \mu)$ .
    Call CALCP(xm,yd,mu,n)

C   Calculate 2-norm of  $H(zd, \mu)$ .
    Call NORMH(func2,h,xm,f,mu,n,uvec)
    ff(lsiter) = func2
    rhs = 2*sigma*deltam*dot
    delf = arw - func2

C   Continue until armijo line search finished.
    Do while (delf.lt.rhs )
    m=m + 1
    lsiter = lsiter + 1
    deltam = delta**m
    lamda(lsiter) = deltam
    Do 200 i=1,n
    zd(i) = z(i) + deltam*dz(i)
    yd(i) = -zd(i)
200 Continue

C   Calculate plus-smooth function  $p(zd, \mu)$ .
    Call CALCP(xp,zd,mu,n)

C   Calculate function value  $f(p(zd, \mu))$ .
    Call CALCF(f,xp,n)

```

```

C      Calculate plus-smooth function  $p(-z_d, \mu)$ .
      Call CALCP(xm,yd,mu,n)

C      Calculate 2-norm of  $H(z_d, \mu)$ .
      Call NORMH(func2,h,xm,f,mu,n,uvec)
      ff(lsite) = func2
      rhs = 2*sigma*deltam*dot
      delf = arw - func2
      End do
      lsc(iter) = lsite

C      New point after line search.
      Do i=1,n
      z(i) = z(i) + deltam*dz(i)
      End do
      Return
      End

C      *****
C      OPT.F
C      Check optimality
C      *****

      Subroutine OPT(eps,z,mu,n,optflag,err,xp,f)
      Double Precision eps,z(*),mu
      Integer n,optflag

      Double Precision ecomp,nerr,err(*),xp(*),f(*)
      Integer i

C      Calculate plus-smooth function  $p(z, \mu)$ .
      Call CALCP(xp,z,mu,n)

C      Calculate function value  $f(p(z, \mu))$ .
      Call CALCF(f,xp,n)

      nerr=0
      Do i=1,n
      err(i) = 1/eps
      End do

C      Calculate minimum of error.
      Do i=1,n
      ecomp = min(xp(i),f(i))
      err(i) = min(ecomp,err(i))
      nerr=nerr+err(i)**2

```

```

End do
C 2-norm of error.
nerr=nerr**0.5
Write(2,*)' Error in opt ', nerr
if (nerr.le.1e-6) then
Write(*,*) '*****'
Write(*,*) '-----> Successful Termination <-----'
Write(*,*) '*****'
optflag = 0
Endif
Return
End

C *****
C CALCP.F
C Compute the interior smooth function
C *****

Subroutine Calcp(p,r,mu,n)
Double Precision p(*),r(*),mu
Integer n

Integer i

C Piecewise quadratic function.
Do i=1,n
If (r(i).lt. -mu/2) then
p(i) = 0
Else if (r(i).gt.mu/2) then
p(i) = r(i)
Else
p(i) = (r(i) + mu/2)**2/(2*mu)
End if
End do
Return
End

C *****
C NORMH.F
C Compute the vector function h and its norm;
C nh,h:output
C xm,f,mu,n:input
C *****

Subroutine NORMH(nh,h,xm,f,mu,n,uvec)
Integer n
Double Precision nh,h(*),xm(*),f(*)

```

```

Integer i
Double Precision uvec(*),mu

nh=0
Do i=1,n
C   mu*b, b=1.
    uvec(i)=mu*1

C   Calculate  $H(z,\mu) = (1-\mu)*f(p(z,\mu)) - p(-z,\mu) + \mu$ .
    h(i) = (1-mu)*f(i) - xm(i) + uvec(i)

C   Calculate 2-norm of H(z,mu)
    nh=nh+h(i)**2

End do
Return
End

C *****
C PPRIME.F
C Compute the interior point smooth function's gradient
C *****

Subroutine PPRIME(dp,r,mu,n)
Double Precision dp(*),r(*),mu
Integer n

Integer i,j,k,l
Double Precision term1

C Calculate gradient of piecewise quadratic function.
Do i=1,n
    If (r(i).lt. -mu/2) then
        dp(i) = 0
    Else if (r(i).gt. mu/2) then
        dp(i) = 1
    Else
        term1 = r(i) + mu/2
        dp(i) = 2*term1/(2*mu)
    End if
End do
Return
End

```

```

C *****
C JACOB.F
C Compute the Jacobian matrix DH;
C *****

C VDH: value of entries of dH(z,mu).
C DH1: number of row of dH(z,mu).
C DH2: number of column of dH(z,mu).
C VDF: value of entries of df(p(z,mu)).
C DF1: number of row of df(p(z,mu)).
C DF2: number of column of df(p(z,mu)).
C s1: index for alpha part.
C s2: index for beta part.
C s3: index for gamma.
C n1: number of alpha entries.
C n2: number of beta entries.
C n3: number of gamma entries.

      Subroutine JACOB(DH1,DH2,VDH,DF1,DF2,VDF,dp,dpm,mu,n,
$      ne,s1,s2,s3,n1,n2,n3,nDH)
      Integer n,ne,DH1(*),DH2(*),DF1(*),DF2(*),s1(*),s2(*),
$      s3(*),n1,n2,n3,nDH
      Double Precision VDH(*),VDF(*),dp(*),dpm(*),mu

      Integer i,j,k,l

C      df(alpha,alpha) part of dH(z,mu).
      l=1
      Do k=1,ne
        Do i=1,n1
          Do j=1,n1
            If (s1(i).eq.DF1(k) .and. s1(j).eq.DF2(k)) then
              VDH(l) = VDH(l) + (1-mu)*VDF(k)
              DH1(l) = i
              DH2(l) = j
              l=l+1
            End if
          End do
        End do
      End do

C      Calculate (1-mu)*df(alpha,beta)*p'(z,mu) part of dH(z,mu)
      Do k=1,ne
        Do i=1,n1
          Do j=1,n2
            If (s1(i).eq.DF1(k) .and. s2(j).eq.DF2(k)) then
              VDH(l) = VDH(l) + (1-mu)*VDF(k)*dp(s2(j))
              DH1(l) = i

```

```

        DH2(l) = j + n1
        l=l+1
    End if
End do
End do
End do

C    Calculate (1-mu)*df(beta,alpha) part of dH(z,mu).
Do k=1,ne
    Do i=1,n2
        Do j=1,n1
            If (s2(i).eq.DF1(k) .and. s1(j).eq.DF2(k)) then
                VDH(l) = VDH(l) + (1-mu)*VDF(k)
                DH1(l) = i + n1
                DH2(l) = j
                l=l+1
            End if
        End do
    End do
End do

C    Calculate (1-mu)*df(beta,beta)*p'(z,mu)(beta) + p'(-z,mu)(beta)
C    part of dH(z,mu).
Do k=1,ne
    Do i=1,n2
        Do j=1,n2
            If (s2(i).eq.DF1(k) .and. s2(j).eq.DF2(k)) then
                VDH(l) = VDH(l) + (1-mu)*VDF(k)*dp(s2(j))
                DH1(l) = i + n1
                DH2(l) = j + n1
                If (DF1(k).eq.DF2(k)) then
                    VDH(l) = VDH(l) + dpm(s2(j))
                End if
            End if
            l=l+1
        End do
    End do
End do
nDH=l-1

C    Calculate (1-mu)*df(gamma,alpha) part of dH(z,mu).
Do k=1,ne
    Do i=1,n3
        Do j=1,n1
            If (s3(i).eq.DF1(k) .and. s1(j).eq.DF2(k)) then
                VDH(l) = VDH(l) + (1-mu)*VDF(k)
                DH1(l) = i + n1 + n2
                DH2(l) = i
            End if
        End do
    End do
End do

```

```

                                l=l+1
                                End if
                            End do
                        End do
                    End do
                End do
            End do
        End do
    End do
    C   Calculate (1-mu)*df(gamma,beta)*p'(z,mu) part
    C   of dH(z,mu).
        Do k=1,ne
        Do i=1,n3
        Do j=1,n2
            If (s3(i).eq.DF1(k) .and. s2(j).eq.DF2(k)) then
                VDH(l) = VDH(l) + (1-mu)*VDF(k)*dp(s2(j))
                DH1(l) = i + n1 + n2
                DH2(l) = i + n1
                l=l+1
            End if
        End do
    End do
End do
End do
End do
Do k=1,ne
Do i=1,n3
Do j=1,n3
    If (s3(i).eq.DF1(k) .and. s3(j).eq.DF2(k)) then
        If (DF1(k).eq.DF2(k)) then
            VDH(l) = 1
            DH1(l) = i + n1 + n2
            DH2(l) = i + n1 + n2
            l=l+1
        End if
    End if
End do
End do
End do
Return
End

```

```

C   *****
C   CALCF.F
C   Compute the function F for problem spatial equilib.;
C   Input : x .. variables
C   Kojima-Josephy
C   *****

```

```

Subroutine Calcf(f,x,n)
Double Precision x(*),f(*)
Integer n,i,j

```

```

f(1) = 3*x(1)**2 + 2*x(1)*x(2) + 2*x(2)**2 + x(3) + 3*x(4) - 6
f(2) = 2*x(1)**2 + x(1) + x(2)**2 + 3*x(3) + 2*x(4) - 2
f(3) = 3*x(1)**2 + x(1)*x(2) + 2*x(2)**2 + 2*x(3) + 3*x(4) - 1
f(4) = x(1)**2 + 3*x(2)**2 + 2*x(3) + 3*x(4) - 3
Return

```

End

```

C *****
C CALCG.F
C Compute the Jacobian DF of F for traffic equilibrium C CALCGTR.F
C Input : x .. variables
C Kojima-Josephy
C *****

```

```

Subroutine CALCG(DF1,DF2,VDF,DH1,DH2,VDH,x,n,ne)
Integer n,i,j,ne,DF1(*),DF2(*),DH1(*),DH2(*),k
Double Precision VDF(*),x(*),VDH(*)

```

```

C First row
VDF(1) = 6*x(1) + 2*x(2)
VDF(2) = 2*x(1) + 4*x(2)
VDF(3) = 1
VDF(4) = 3
C Second row
VDF(5) = 4*x(1) + 1
VDF(6) = 2*x(2)
VDF(7) = 3
VDF(8) = 2
C Third row
VDF(9) = 6*x(1) + x(2)
VDF(10) = x(1) + 4*x(2)
VDF(11) = 2
VDF(12) = 3
C Fourth row
VDF(13) = 2*x(1)
VDF(14) = 6*x(2)
VDF(15) = 2
VDF(16) = 3
ne=16
DF1(1) = 1
DF2(1) = 1
DF1(2) = 1
DF2(2) = 2
DF1(3) = 1

```

```
DF2(3) = 3
DF1(4) = 1
DF2(4) = 4
DF1(5) = 2
DF2(5) = 1
DF1(6) = 2
DF2(6) = 2
DF1(7) = 2
DF2(7) = 3
DF1(8) = 2
DF2(8) = 4
DF1(9) = 3
DF2(9) = 1
DF1(10) = 3
DF2(10) = 2
DF1(11) = 3
DF2(11) = 3
DF1(12) = 3
DF2(12) = 4
DF1(13) = 4
DF2(13) = 1
DF1(14) = 4
DF2(14) = 2
DF1(15) = 4
DF2(15) = 3
DF1(16) = 4
DF2(16) = 4
k=1
Do i=1,n
  Do j=1,n
    DH1(k)=i
    DH2(k)=j
    VDH(k)=0
    k=k+1
  End do
End do
Return
End
```