Summer 2011

Full-Newton-Step Interior-Point Method for the Linear Complementarity Problems

H.K. Pubudu Kaluarachchi

FULL-NEWTON-STEP INTERIOR-POINT METHOD FOR THE LINEAR COMPLEMENTARITY PROBLEMS

by

H.K PUBUDU KALPANI KALUARACHCHI

(Under the Direction of Goran Lesaja)

ABSTRACT

In this thesis, we present a new Interior-Point Method (IPM) for monotone Linear Complementarity Problem (LPC). The advantage of the method is that it uses full Newton-steps, thus, avoiding the calculation of the step size at each iteration. However, by suitable choice of parameters the iterates are forced to stay in the neighborhood of the central path, hence, still guaranteeing the global convergence of the method under strict feasibility assumption. The number of iterations necessary to find ϵ-approximate solution of the problem matches the best known iteration bounds for these types of methods. The preliminary implementation of the method and numerical results indicate robustness and practical validity of the method.

INDEX WORDS: linear complementarity problem, interior-point method, full Newton-step, polynomial convergence

FULL-NEWTON-STEP INTERIOR-POINT METHOD FOR THE LINEAR COMPLEMENTARITY PROBLEMS

by

H.K PUBUDU KALPANI KALUARACHCHI

B.S special degree in Mathematics, University of Colombo, 2008, Sri Lanka

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment of the Requirement for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

2011

FULL-NEWTON-STEP INTERIOR-POINT METHOD FOR THE LINEAR COMPLEMENTARITY PROBLEMS

by

H.K PUBUDU KALPANI KALUARACHCHI

Major Professor: Goran Lesaja

Committee: Scott Kersey

Billur Kaymakcalan

Electronic Version Approved:

Summer, 2011

DEDICATION

This thesis is dedicated to my beloved father who was my very first Math teacher and who loves me mostly in this world.

ACKNOWLEDGMENTS

My deepest gratitude extends to Dr. Goran Lesaja for his eminent guidance and courage given to me throughout this thesis and also during entire my study at GSU. The success of this work is credited to his exceptional directions, and my respectful notice is that it was a privilege to have him as my advisor. My special thank goes to my committee members, Dr. Scott Kersey and Dr. Billur Kaymakcalan. I also acknowledge every member of the outstanding faculty and staff in Department of Mathematical Siences at Georgia Southern University, specially Dr. Martha Abell, Department Chair, and Dr. Yan Wu, Graduate Program Director, for all their utmost commitments toward the success of the students and the department.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Description of the Problem

The objective of the Linear Complementarity Problem (LCP) is finding a vector in a finite dimensional real vector space that satisfies a certain system of inequalities. More precisely, for a given vector $q \in R^n$ and matrix $M \in R^{n \times n}$ find a vector $x \in R^n$ (or show that no such vector exists), such that the following inequalities and equations are satisfied.

$$\begin{aligned} x &\geq 0 \\ q + Mx &\geq 0 \\ x^T(q + Mx) &= 0 \end{aligned} \tag{1.1}$$

We denote the above LCP by the pair (M,q), that is LCP(M,q). We will elaborate more on the problem formulation in the next chapter.

The LCP is not an optimization problem. However, it is closely related to optimization problems because Kurush-Kuhn-Tucker (KKT) optimality conditions for many optimization problems can be formulated as LCP. For example, KKT conditions of linear and quadratic optimization problems can be formulated as LCP (see 2.1 in Chapter 2). In addition, there are problems that can be directly formulated as LCP. This is the reason why LCP is often considered as a problem in the mathematical programming area. The applications include, but are not limited to, economics, engineering (game and equilibrium theory), transportation, and many other areas of operations research.

Specially significant is a connection of LCP to LP which is by for most used and theoretically examined optimization problem.

The LP can be formulated as follows,

$$\begin{aligned} min \;\; & c^T x \\ s.t \;\; & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.2}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$.

The minimization can be replaced by maximization. Thus, LP is a problem of finding $x \in \mathbb{R}^n$ that minimizes (or maximizes) objective function $z = c^T x$ subject to a set of constrains $Ax = b$, $x \geq 0$.

As we already mentioned, KKT conditions of LP can be formulated as LCP. See Section 2.3 in Chapter 2, Example 1. This provides a direct link between the two and enables generalization of methods and techniques used to solve LP to LCP and vice versa.

It is important to mention another well known fact and that is for the general type of matrix M, LCP (1.1) is a NP complete problem, which means that there is no efficient (polynomial) algorithm to solve LCP with general matrix M. Therefore, we need to consider classes of matrices M for which efficient algorithms do exist. See Section 2.2 in Chapter 2.

1.2 A Brief Historical Overview

Some instances of the LCP can be traced back to the early 1940's; however, larger interest in LCP was taken in the early to mid 1960's in conjunction with the rapid development of theory and methods for LP.

In 1947, George Dantzig proposed a famous SM to solving the LP. Basically, the main idea of the SM is to travel along from vertex to vertex on the boundary of the feasible region. The method constantly increases (or decreases) the objective function until either an optimal solution is found or the SM concludes that such an optimal solution does not exist.

Theoretically, the algorithm could have a worse-case scenario of 2^n iteration, with n being the size of the problem, which is an exponential number. This was shown in 1972 by Klee and Minty [8]. However, on behalf of the SM, it is remarkably efficient in practice and an exponential number of iterations has never been observed in practice. It usually requires $O(n)$ iterations to solve a particular problem. There exists many resources and excellent software for the SM.

Another great advancement in the area of solving convex optimization problems was the *ellipsoid method*. This method was introduced by Nemirovsky and Yudin [24] in 1976 and by Shor [20] in 1977. The algorithm works by encapsulating the minimizer of a convex function in a sequence of ellipsoids whose volume decreases at each iteration. Later Khachiyan [7] showed in 1984 that the ellipsoid method can be used to solve the LP in polynomial time. This was the first polynomial time algorithm for the LP. Unfortunately, in practice, the method was far surpassed by the SM. Nevertheless, the theoretical importance of the ellipsoid method is hard to

neglect.

In 1984, Karmarkar [6] introduced an *Interior-Point Method* (IPM) for LP. Karmarkar used the efficiency of the simplex method with the theoretical advantages of the ellipsoid method to create his efficient polynomial algorithm. The algorithm is based on projective transformations and the use of Karmarkar's primal potential function. This new algorithm sparked much research, creating a new direction in optimization - the field of IPMs. Unlike the SM, which travels from vertex to vertex along the edges of the feasible region, the IPM follows approximately a central path in the interior of the feasible region and reaches the optimal solution only asymptotically. As a result of finding the optimal solution in this fashion, the analysis of the IPMs become substantially more complex than that of the SM.

Since the first IPM was developed, many new and efficient IPMs for solving LP have been created. Many researches have proposed different interior-point methods, which can be grouped into two different groups: potential reduction algorithms and path-following algorithms. Each of the two groups contains algorithms based on primal, dual, or primal-dual formulations of the LP. Also, computational results show that the primal-dual formulation is superior to either the primal or dual formulation of the algorithm. We will focus on the *primal-dual* path-following IPMs, which have become the standard of efficiency in practical applications. These primal-dual methods are based on using Newton's method in a careful and controlled manner.

Soon after the SM was developed, a similar method for solving LCP was introduced by Lemke [10]. It is a pivoting algorithm similar to the SM. Unfortunately, Lemke's algorithm can sometimes fail to produce a solution even if one exists. Nevertheless, Lemke's algorithm was extremely useful. However, researchers kept searching

for other methods for the LCP. Much later, in the 1990's, the tradition of immediate generalizations from LP to LCP continued even more strongly in the case of the IPMs and many efficient IPMs have been proposed for LCP.

In this thesis, we will focus on extending a class of IPMs, from LP to LCP. The main features of this class of methods is that at each iteration a full Newton-step is taken, i.e., it is not necessary to calculate a step size. These type of IPMs are called Full-Newton-step IPM (FNS-IPM). They were first discussed for LP by Roos in [18].

In addition, IPMs have been generalized to solve many other important optimization problems, such as semidefinite optimization, second order cone optimization, and general convex optimization problems. The unified theory of IPMs for general convex optimization problems was first developed by Nesterov and Nemirovski [15] in 1994.

The first comprehensive monograph that considers in-depth analysis of the LCP and methods for solving it is the monograph of Cottle, Pang, and Stone [3]. More recent results on the LCP as well as nonlinear complementarity problems and variational inequalities are contained in the monograph of Facchinei and Pang [5].

CHAPTER 2

LINEAR COMPLEMENTARITY PROBLEM

In this chapter the linear complementarity problem (LCP) is introduced, defined, and discussed. Also, several direct applications of the linear complementarity problem are presented and discussed.

2.1 Linear Complementarity Problem

LCP is a problem of finding a particular vector in a finite real vector space that satisfies a certain system of inequalities. Mathematically, given a vector $q \in \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{nxn}$, we want to find a vector $x \in \mathbb{R}^n$ (or to show such a vector does not exist) such that

$$\begin{aligned} s &= q + Mx \\ x \geq 0, s &\geq 0 \\ x^T s &= 0. \end{aligned} \tag{2.1}$$

A sufficient condition for existence and uniqueness of a solution to this problem is that M be symmetric positive definite. Since $(x, s) \geq 0$, the complementarity equation $x^T s = 0$ can be written equivalently as

$$xs = 0,$$

which represents component-wise product of vectors, as follows,

$$xs = (x_1 s_1, x_2 s_2, \ldots, x_n s_n)^T. \tag{2.2}$$

This product is often called Hadamard's product.

The feasible set of points (feasible region) of the LCP as defined in (2.1) is the

following set:

$$F = \left\{(x, s) \in \mathbb{R}^{2n} : s = Mx + q, x \geq 0, s \geq 0\right\}. \tag{2.3}$$

Furthermore, the set of strictly feasible points of the LCP is the following set:

$$F_0 = \left\{(x, s) \in F : x > 0, s > 0\right\}.$$

The solution set of the LCP is given by

$$F^* = \left\{(x^*, s^*) \in F : x^{*T} s^* = 0\right\}. \tag{2.4}$$

An important subset of the above solution set is a set of strict complementarity solutions

$$F_s^* = \left\{(x^*, s^*) \in F^* : x^* + s^* > 0\right\}. \tag{2.5}$$

We can now say that the main idea of the LCP is to find a certain vector x that is both feasible and complementary. This vector is called a solution of the LCP. The LCP is always solvable with the zero vector being a trivial solution, if $q \geq 0$.

2.2 Classes of LCP

In general LCP is NP-complete, which means that there exists no polynomial algorithms for solving it. Thus, the problem needs to be restricted to certain classes of matrices for which the polynomial algorithms exist. We now list several such classes of matrices M for LCP. They are as follows:

- Skew-symmetric matrices (SS):

$$(x \in \mathbb{R}^n)(x^T M x = 0). \tag{2.6}$$

- Positive semi-definite matrices (PSD):

$$(x \in \mathbb{R}^n)(x^T Mx \geq 0). \tag{2.7}$$

- P-matrices: Matrices with all principal minors positive or equivalently

$$(0 \neq x \in \mathbb{R}^n)(\exists i \in I)(x_i(Mx)_i > 0). \tag{2.8}$$

- P_0-matrices: Matrices with all principal minors nonnegative or equivalently

$$(0 \neq x \in \mathbb{R}^n)(\exists i \in I)(x_i \neq 0 \text{ and } x_i(Mx)_i \geq 0). \tag{2.9}$$

- Sufficient matrices (SU): Matrices which are column and row sufficient

  - Column sufficient matrices (CSU):

$$(\forall x \in \mathbb{R}^n)(\forall i \in I)(x_i(Mx)_i \leq 0 \Rightarrow x_i(Mx)_i = 0). \tag{2.10}$$

  - Row sufficient matrices (RSU): M is row sufficient if M^T is column sufficient.

- $P_*(\kappa)$: Matrices such that

$$(1 + 4\kappa) \sum_{i \in I^+(x)} x_i(Mx)_i + \sum_{i \in I^-(x)} x_i(Mx)_i \geq 0, \forall x \in \mathbb{R}^n,$$

where

$$I^+(x) = \{i : x_i(Mx)_i > 0\}, I^-(x) = \{i : x_i(Mx)_i < 0\},$$

or equivalently

$$x^T Mx \geq -4\kappa \sum_{i \in I^+(x)} x_i(Mx)_i, \forall x \in \mathbb{R}^n, \tag{2.11}$$

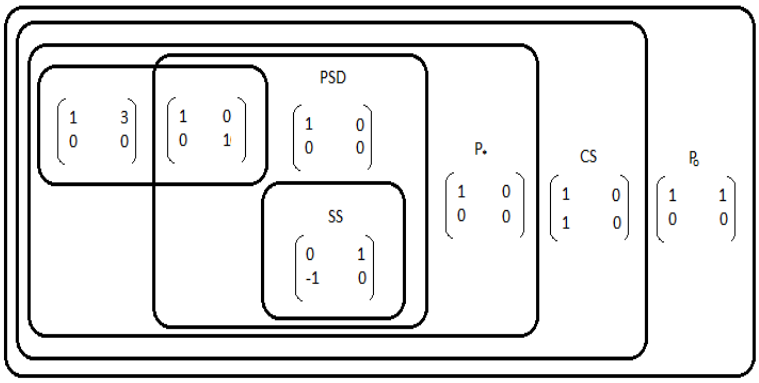

Figure 2.1: Relations and examples of the classes of matrices.

and

$$P_* = \bigcup_{\kappa \geq 0} P_*(\kappa). \tag{2.12}$$

Especially interesting, important (and nontrivial) is that the P_* matrices are just sufficient.

The relationship between some of the above classes is as follows:

$$SS \subset PSD \subset P_* = SU \subset CS \subset P_0, \quad P \subset P_*, \quad P \cap SS = \emptyset. \tag{2.13}$$

Some of these relations are obvious, like PSD $= P_*(0) \subset P_*$ or $P \subset P_*$, while others require proof. Refer to Figure 2.1, which was first published in [9], to see a visual flow of how these classes of matrices are related. Also, all of the above classes have the nice property that if matrix M belongs to one of these classes, then every principal sub-matrix of M also belongs to the class.

In this thesis, we will assume that matrix M is a positive semi-definite (PSD) matrix. This case is not the most general, but it is certainly most commonly used both in theory and practice. Hence, this is reason why we will focus on this class of matrices in the thesis. The LCP with a PSD matrix M is called *monotone* LCP.

2.3 Introductory Examples

LCP has many applications. Some examples of the LCP include but are by far not limited to: the bimatrix game, optimal invariant capital stock, optimal stopping, convex hulls in the plane, and the market equilibrium problems. Each one of the listed problems can be reformulated into the linear complementarity problem. In the sequel, we will describe several applications.

Example 1: Quadratic Programming

Quadratic programming is another application of the LCP. It is the problem of minimizing or maximizing a quadratic function of several variables subject to linear constraints on these variables. The quadratic program (QP) is defined as

$$\begin{aligned} \text{minimize} \quad & f(x) = c^T x + \tfrac{1}{2} x^T Q x \\ \text{subject to} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{2.14}$$

where $Q \in \mathbb{R}^{nxn}$ is symmetric, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{mxn}$ and $b \in \mathbb{R}^m$. Note that the case where $Q = 0$ gives rise to a linear program (LP). If x is a locally optimal solution of the quadratic program (2.14), then there exists a vector $y \in \mathbb{R}^m$ such that the pair

(x, y) satisfies the Karush-Kuhn-Tucker optimality conditions

$$\begin{aligned} u &= c + Qx - A^T y \geq 0, \quad x \geq 0, \quad x^T u = 0, \\ v &= -b + Ax \geq 0, \qquad\quad y \geq 0, \quad y^T v = 0. \end{aligned} \tag{2.15}$$

If Q is positive semi-definite (the objective function $f(x)$ is convex), then the conditions in (2.15) are sufficient for the vector x to be a globally optimal solution of (2.14).

The Karush-Kuhn-Tucker conditions in (2.14) define the LCP where

$$q = \begin{bmatrix} c \\ -b \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} Q & -A^T \\ A & 0 \end{bmatrix}. \tag{2.16}$$

Note that M is not symmetric, even though Q is symmetric. However, M does have a property known as *bisymmetry*. A square matrix A is *bisymmetric* if it can be brought to the form

$$A = \begin{bmatrix} G & -A^T \\ A & H \end{bmatrix},$$

where both G and H are symmetric. Also, if Q is positive semi-definite, then so is M. In general, a square matrix M is positive semi-definite if $z^T M z \geq 0$ for every vector z.

This convex quadratic programming model, in the form of (2.14), has a magnitude of practical applications in engineering, finance, and many other areas. The size of these practical problems can become very large. Thus, the LCP plays an important role in the numerical solution of these problems.

Example 2: Bimatrix games

Game theory analyzes strategic interactions in which the outcome of one's choices depends upon the choices of others. For a situation to be considered a game, there must be at least two rational players who take into account one another's actions when formulating their own strategies. We consider a game with two players called player I and player II and the game consists of large number of plays. Here at each play Player I picks one of m choices and Player II picks one of n choices. These choices are called pure strategies. If in a certain play, Player I choose pure strategy i and Player II chooses pure strategy j, then Player I loses A_{ij} and Player II loses B_{ij}. A positive value of A_{ij} represents a loss to Player I, while a negative value of A_{ij} represents a gain. Similarly for Player II and B_{ij}. The matrices A and B are called loss matrices, and the game is fully determined by the matrix pair (A, B).

If $A + B = 0$, the game is known as zero sum game and if $A + B \neq 0$ game is known as bimatrix game. Player I chooses to play strategy i with probability x_i such that $\sum x_i = 1$, and Player II chooses to play strategy j with probability y_j such that $\sum y_j = 1$, then expected loss of Player I is $x^T Ay$ and expected loss of Player II is $x^T By$.

A player is changing his own strategy while other player holds his strategy fixed to minimize the loss. i.e,

$$\begin{aligned} \overline{x}^T A\overline{y} &\leq x^T A\overline{y} \quad \forall x \geq 0 \quad e_m^T x = 1, \\ \overline{x}^T B\overline{y} &\leq \overline{x}^T By \quad \forall y \geq 0 \quad e_n^T y = 1. \end{aligned} \tag{2.17}$$

The objective is to find $(\overline{x}, \overline{y})$ that is called Nash equilibrium pair. Nash equilibrium can be find using LCP as described in the Lemma below.

Lemma 2.3.1. *Suppose $A, B \in \mathbb{R}^{m\times n}$ are positive loss matrices representing a game (A, B) and suppose that $(s, t) \in \mathbb{R}^{m\times n}$ solves LCP(M,q), where*

$$M = \begin{bmatrix} 0 & A \\ B^T & 0 \end{bmatrix}, q = -e_{m+n} \in \mathbb{R}^{m+n}.$$

Then $(\overline{x}, \overline{y})$ *such that,*

$$\overline{x} = \tfrac{s}{e_m^T s} \text{ and } \overline{y} = \tfrac{t}{e_m^T t},$$

is an equilibrium pair of $\Gamma(A, B)$.

Proof. We write LCP conditions explicitly as

$$\begin{aligned} 0 \leq At - e_m \perp s \geq 0 \\ 0 \leq B^T s - e_n \perp t \geq 0 \end{aligned} \tag{2.18}$$

from the equation (2.1) we have $Mx + q = s \geq 0$ and $x \geq 0$. So we can write these inequalities as below,

$$\begin{bmatrix} 0 & A \\ B^T & 0 \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} + \begin{bmatrix} e_m \\ e_n \end{bmatrix} \geq 0, \quad \begin{bmatrix} At \\ B^T s \end{bmatrix} + \begin{bmatrix} e_m \\ e_n \end{bmatrix} \geq 0. \tag{2.19}$$

This implies $At - e_m \geq 0$ and $B^T s - e_n \geq 0$. Therefore $t \neq 0$ and $s \neq 0$. Then $\overline{x} = \frac{s}{e_m^T s}$ and $\overline{y} = \frac{t}{e_n^T t}$ well define. $\overline{x} \geq 0$, $\overline{y} \geq 0$, from the definition we have $e_m^T \overline{x} = 1$ and $e_n^T \overline{y} = 1$. Then $\overline{x}$ and $\overline{y}$ are mixed strategies. By complementarity we have,

$$\overline{x}^T (At - e_m) = \frac{s^T}{e_m^T s}(At - e_m) = 0. \tag{2.20}$$

Since $\overline{x}$ and $\overline{y}$ are mixed strategies, and from the Equation (2.20), we get the following property.

$$\overline{x}^T At = \overline{x}^T e_m = 1. \tag{2.21}$$

So we have,

$$\begin{aligned} A\overline{y} - (\overline{x}^T A\overline{y})e_m &= \tfrac{1}{e_n^T t}(At) - (\overline{x}^T A\overline{y})e_m \\ &= \tfrac{1}{e_n^T t}(At - (\overline{x}^T At)e_m) \\ &= \tfrac{1}{e_n^T t}(At - e_m) \ \text{ from } \ (2.21) \end{aligned} \tag{2.22}$$

Since $At - e_m \geq 0$ and $x \geq 0$, we have $x^T(A\overline{y} - (\overline{x}^T A\overline{y})e_m) \geq 0$. This implies,

$$x^T A\overline{y} \geq (x^T e_m)(\overline{x}^T A\overline{y}) = \overline{x}^T A\overline{y} \tag{2.23}$$

Similarly we can prove $\overline{x}^T By \geq \overline{x}^T B\overline{y}$. Hence $(\overline{x}, \overline{y})$ is a Nash equilibrium pair. □

Example 3: The Market Equilibrium Problem

The state of an economy where the supplies of producers and the demands of consumers are balanced at the resulting price level is called *market equilibrium* . We can use a linear programming model to describe the supply side that captures technological details of production activities for a particular market equilibrium problem. Econometric models with commodity prices as the primary independent variables generates the market demand function. Basically, we need to find vector x^* and subsequent vectors p^* and r^* such that the conditions below are satisfied for supply, demand, and equilibrium:

Supply conditions:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \geq b \\ & Bx \geq r^* \\ & x \geq 0 \end{aligned} \tag{2.24}$$

where c is the cost vector for the supply activities, x is the vector production activities. Technological constraints on production are represented by the first condition in (2.24)

and the demand requirement constraints are represented by the second condition in (2.24);

Demand conditions:

$$r^* = Q(p^*) = Dp^* + d \tag{2.25}$$

where $Q(\cdot)$ is the market demand function with p^* and r^* representing the vectors of demand prices and quantities, respectively. $Q(\cdot)$ is assumed to be an affine function;

Equilibrium condition:

$$p^* = \pi^* \tag{2.26}$$

where the (dual) vector of market supply prices corresponding to the second constraint in (2.24) is denoted by π^*.

Using Karush-Kuhn-Tucker conditions for problem (2.24), we see that a vector x^* is an optimal solution of problem (2.24) if and only if there exists vectors v^* and π^* such that:

$$\begin{array}{lll} y^* = c - A^T v^* - B^T \pi^* \geq 0, & x^* \geq 0, & (y^*)^T x^* = 0, \\ u^* = -b + Ax^* \geq 0, & v^* \geq 0, & (u^*)^T v^* = 0, \\ \delta^* = -r^* + Bx^* \geq 0, & \pi^* \geq 0, & (\delta^*)^T \pi^* = 0. \end{array} \tag{2.27}$$

If for r^*, we substitute the demand function (2.25) and we use condition (2.26), then we can see that the conditions in (2.27) gives us the linear complementarity problem where

$$q = \begin{bmatrix} c \\ -b \\ -d \end{bmatrix}, \; M = \begin{bmatrix} 0 & -A^T & -B^T \\ A & 0 & 0 \\ B & 0 & -D \end{bmatrix}. \tag{2.28}$$

Observe that the matrix M in (2.28) is bisymmetric and if the matrix D is symmetric, as it could have been seen, the Karush-Kuhn-Tucker optimization conditions of the market equilibrium problem, and in fact the linear problem in general, can be expressed in the LCP framework. This can also be extended to quadratic programming problems as discussed below.

$$\begin{aligned} \text{maximize} \quad & d^T x + \tfrac{1}{2} x^T D x + b^T y \\ \text{subject to} \quad & A^T y + B^T x \leq c \\ & x \geq 0, \quad y \geq 0 \end{aligned} \tag{2.29}$$

On the other hand, if D is asymmetric, Then M is not bisymmetric and the connection between the market equilibrium model and the quadratic program above fails to exist.

CHAPTER 3

LEMKE'S METHOD

In this chapter, we review a well known Lemke's algorithm to solve LCP. This is a pivoting algorithm introduced by Lemke [10] and it is a generalization of Dantzig's Simplex Method developed earlier for LP.

3.1 Basic Definition

We consider an LCP in the standard form as described in (2.1) Chapter 2.

$$\begin{aligned} s &= q + Mx \\ x \geq 0, s &\geq 0 \\ x^T s &= 0. \end{aligned} \tag{3.1}$$

We denote it LCP(M,q). We additionally assume that M is positive semidefinite matrix. To describe Lemke's method for solving LCP(M,q) we introduce some definitions.

Definition 3.1.1.

Consider the problem SLCP(M,q) (3.1).

1. A component s_i is called the complement of x_i, and vice versa, for $i = 1, 2, ..., n$.
2. Pair (x, s) is complementary if $x \geq 0, s \geq 0$, and $x^T s = 0$. (Note that a complementary pair must satisfy $x_i s_i = 0$ for $i = 1, 2, ..., n$.)
3. Pair (x, s) is almost complementary if $x \geq 0, s \geq 0$, and $x_i s_i = 0$ for i=1,2,...,n. except for a single index j, $1 \leq j \leq n$.

3.2 Lemke's Method

For positive semidefinite M matrix, Lemke's method generates a finite sequence of feasible, almost-complementary pairs that terminates at a complementary pair or an unbounded ray.

Similarly to the Simplex Method, an initial pair must first be obtained, usually via a Phase I scheme. There are different Phase I schemes depending on the particular structure of LCP. We will describe a commonly used Phase I scheme, which requires only one pivot.

Phase II generates a sequence of almost-complementary vector pairs. It performs a pivot at each iteration, selecting the pivot row by means of a ratio test like that of the Simplex Method, whose purpose is to ensure that the components of x and s remain nonnegative throughout the procedure. Phase II finishes when complementary pair is found or we end up on the unbounded ray.

This outline can be summarized as follows.

Lemke's Algorithm

Phase I: (Generates a Feasible Almost- Complementary Table).

1. If $q \geq 0$, STOP : $x = 0$ is a solution of LCP(M,q); that is, $(x, s) = (0, q)$ is a feasible complementary pair.

2. Otherwise, add the artificial variables x_0 and s_0 that satisfy the following relationships:

$$s = Mx + ex_0 + q, s_0 = x_0, \tag{3.2}$$

where e is the vector of ones in $\mathbb{R}^n$. Create the initial tableau,

	x	x_0	1
$s =$	M	e	q
$s_0 =$	0	1	0

3. Make this tableau feasible by carrying out a Jordan exchange on the x_0 column and the raw corresponding to the most negative q_i.

4. Without removing the artificial variables from the tableau, proceed to *Phase II*.

(**Phase II**: Generate a Feasible Complementary or Unbounded Tableau).

1. Start with a feasible almost-complementary pair (x, s) and the corresponding tableau in Jordan exchange form,

	s_{I_1}	x_{J_2}	1
$x_{J_1} =$	$H_{I_1J_1}$	$H_{I_1J_2}$	h_{I_1}
$s_{I_2} =$	$H_{I_2J_1}$	$H_{I_2J_2}$	h_{I_2}

   Record the variable that becames nonbasic (i.e., becames a column label) at the previous iteration. At the first step, this is simply the component of s that was exchanged with x_0 during Phase I.

2. Pivot column selection: Choose the column s corresponding to the complement of the variable that became nonbasic at the previous pivot.

3. Pivot row selection: Choose the row r such that,

$$-h_r/H_{rs} = min\,\{-h_i/H_{is} | H_{is} < 0\}.$$

If all $H_{is} \geq 0$, STOP: An unbounded ray has been found.

4. Carry out a Jordan exchange on element H_{rs}. If (x, s) is complementary, STOP: (x, s) is a solution. Otherwise, go to Step 2.

We continue with few remarks.

Remarks

1. Step 2 maintains almost-complementarity by moving a component into the basis as soon as its complement is moved out. By doing so, we ensure that for all except one of the components, exactly one of x_i and s_i is basic while the other is nonbasic. Since nonbasic variables are assigned the value 0, this fact ensures that $x_i s_i = 0$ for all except one component which is the almost complementary property. When the initial tableau of Phase II was derived from Phase I, it is the variables s_0 and x_0 that violate complementarity until an optimal tableau is found.

2. The ratio test in Step 3 follows from the same logic as in the Simplex Method. We wish to maintain non negativity of all the components in the last column, and so we allow the nonbasic variable in column s to increase away from zero only until it causes one of the basic variables to become zero. This basic variable is identified by the ratio test as the one to leave the basis in the current iteration.

3. In practice, it is not necessary to insert the s_0 row into the tableau, since s_0 remains in the basis throughout and is always equal to x_0.

The following important theorem assures that Lemke's algorithm terminates at the solution of the LCP(M,q) if M is positive semidifinite.

Theorem 3.2.1. *1. If $M \in \mathbb{R}^{nxn}$ is positive definite, then Lemke's algorithm terminates at the unique solution of LCP(M,q) for any $q \in \mathbb{R}^n$.*

2. If $M \in \mathbb{R}^{nxn}$ is positive semidefinite, then for each $q \in \mathbb{R}^n$, Lemke's algorithm terminates at a solution of LCP(M,q) or at an unbounded ray. In the latter case, the set $\{x|Mx + s \geq 0, x \geq 0\}$ is empty; that is, there is no feasible pair.

The proof can be found in [4].

3.3 Example

We consider a quadratic programming problem

$$\begin{aligned} min \;\; & \tfrac{1}{2}x_1^2 - x_1x_2 + \tfrac{1}{2}x_2^2 + 4x_1 - x_2 \\ s.t \;\; & x_1 + x_2 - 2 \geq 0 \\ & x_1, x_2 \geq 0. \end{aligned} \tag{3.3}$$

The KKT condition of this problem are described in Example 1, Chapter 2, (2.15) and (2.16). In this case we have

$$Q = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \; A = \begin{bmatrix} 1 & 1 \end{bmatrix}, \; p = \begin{bmatrix} 4 \\ -1 \end{bmatrix}, \; b = \begin{bmatrix} 2 \end{bmatrix},$$

which leads to the following LCP

$$M = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & 0 \end{bmatrix}, \; q = \begin{bmatrix} 4 \\ -1 \\ -2 \end{bmatrix}, \; \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ u_1 \end{bmatrix}.$$

Below are the steps of the Lemke's algorithm applied to this problem.

Phase I

Step 1: According to the Phase I of the Lemke's Algorithm, here we add the artificial variable x_0 that satisfy the following relationship, $s = Mx + ex_0 + q$. so the initial table is as follows.

	x_1	x_2	x_3	x_0	1
$s_1 =$	1	-1	-1	1	4
$s_2 =$	-1	1	-1	1	-1
$s_3 =$	1	1	0	1	-2

We make this table feasible by carrying out a Jordan elimination on the x_0 column (pivot column, s=4) and the row corresponding to the most negative entry in the last column (pivot row, r=3). Here $s = 4$ and $r = 3$. Since $B_{rs} = \frac{1}{A_{rs}}$ and $B_{rj} = \frac{-A_{rj}}{A_{rs}}$, $B_{is} = \frac{A_{is}}{A_{rs}}$ and $B_{rj} = A_{ij} - B_{is}A_{rj}$ we find the entries of the second table below.

	x_1	x_2	x_3	s_3	1
$s_1 =$	0	-2	-1	1	6
$s_2 =$	-2	0	-1	1	1
$x_0 =$	-1	-1	0	1	2

This table yields almost complementary solution $x_0 = 2$, $x_1 = 0$, $x_2 = 0$, $x_3 = 0$ and $s_1 = 0$, $s_2 = 1$, $s_3 = 0$.

Phase II

Step 2: In Phase I we obtained the following table.

	x_1	x_2	x_3	s_3	1
$s_1 =$	0	-2	-1	1	6
$s_2 =$	-2	0	-1	1	1
$x_0 =$	-1	-1	0	1	2

Since s_3 became non basic at last pivot, here we choose x_3 as pivot column.

Minimum ratio test gives $min\left\{\frac{-6}{-1} = 6, \frac{-1}{-1} = 1\right\} = 1$.

Thus pivot row is $r = 2$ (from minimum ratio test). When $s = 3$ and $r = 2$ we find the entries in the third table by using the Jordan elimination. Using formulas indicated in Step 1 we obtain the following table:

	x_1	x_2	s_2	s_3	1
$s_1 =$	2	-2	1	0	5
$x_3 =$	-2	0	-1	1	1
$x_0 =$	-1	-1	0	1	2

This table yields almost complementary solution $x_0 = 2$, $x_1 = 0$, $x_2 = 0$, $x_3 = 1$ and $s_1 = 5$, $s_2 = 0$, $s_3 = 0$.

Step 3: By continuing the same process as in Step 2 we get $s = 2$ and $r = 3$. After performing Jordan elimination we obtain the following table.

	x_1	x_0	s_2	s_3	1
$s_1 =$	4	2	1	-2	1
$x_3 =$	-2	0	-1	1	1
$x_2 =$	-1	-1	0	1	2

This is a final table, because it contains a solution that is fully complementary, $x_0 = 0$, $x_1 = 0$, $x_2 = 2$, $x_3 = 1$ and $s_1 = 1$, $s_2 = 0$, $s_3 = 0$. Hence, the solution of the original problem (3.3) is $x_1 = 0$ and $x_2 = 2$.

CHAPTER 4

FULL NEWTON-STEP INTERIOR-POINT METHOD

In this chapter, we will discuss the IPM method for solving a monotone LCP. The advantage of the method is that it uses full-Newton-steps, thus, avoids the calculation of the step size at each iteration. First, we will explain the concept of the IPM with full Newton-steps, then we will analyze its convergence and finally we will give an estimate on the number of iterations needed to find an ϵ - approximate solution of LCP.

4.1 Interior Point Condition for LCP

We consider the monotone LCP in the standard form. Find a point $(x, s) \in \mathbb{R}^{2n}$ that satisfies the following conditions

$$\begin{aligned} Mx + q - s = 0, \quad (x, s) \geq 0, \\ xs = 0, \end{aligned} \tag{4.1}$$

where $M \in \mathbb{R}^{n \text{x} n}$ is positive semidefinite matrix, $q \in \mathbb{R}^n$ is a vector. Also recall that the xs in the last equation represents the component-wise (Hadamard) product of the vectors x and s as defined in (2.2).

We assume that problem (4.1) has a strictly feasible point or equivalently, that it satisfies Interior Point Condition (IPC).

IPC Assumption

There exists $(x^0, s^0) > 0$ such that $Mx^0 + q - s^0 = 0$,

This is an essential assumption needed in the future development of the IPM for LCP.

Recall that we denote feasible region of (4.1) as

$$F = \left\{(x, s) \in \mathbb{R}^{2n} : s = Mx + q, x \geq 0, s \geq 0\right\}, \tag{4.2}$$

and the interior of the feasible region, which we also call strictly feasible region, as

$$F_0 = \left\{(x, s) \in \mathbb{R}^{2n} : s = Mx + q, x > 0, s > 0\right\}. \tag{4.3}$$

In other words IPC can be equivalently stated as: F_0 is not empty, i.e., there is a point $(x^0, s^0) \in F_0$.

4.2 Main Idea of the Method

The general idea is to solve (4.1) using Newton's method. However, Newton's method can "get stuck" at the complementarity equation $xs = 0$. Therefore, the main idea of primal-dual interior-point methods is to replace the last equation in (4.1), the so called *complementarity equation*, with the parameterized equation $xs = \mu e$, with parameter $\mu > 0$. So we consider the following system

$$\begin{aligned} Mx + q - s &= 0, \\ xs &= \mu e, \end{aligned} \tag{4.4}$$

where e is defined as a vector of all ones of size n. By the last equation, any solution (x, s) of (4.4) will satisfy $x > 0$ and $s > 0$. In this thesis we will always assume that the interior point condition (IPC) is satisfied.

It can be shown that for certain classes of matrices, if M has a full rank, i.e. $\text{rank}(M) = n$ and IPC holds, then the parameterized system (4.4) has a unique solution, for each $\mu > 0$. This is particularly true for positive semi-definite matrices that we are considering in this thesis. This solution is denoted as $(x(\mu), s(\mu))$ and we

call $(x(\mu), s(\mu))$ the μ-center of (4.1). The set of μ-centers (with μ running through all positive real numbers) gives a homotopy path, which is called *the central path* of (4.1). The importance of the central path for the LP was discovered first by Sonnevend [21] and Megiddo [16] for Linear optimization(LO) and later generalized to LCP by Kojima et al. [9]. The main property of the central path is that if $\mu \to 0$, then the limit of the central path exists and since the limit points satisfy the complementarity condition, the limit yields the optimal solutions for (4.1).

This limiting property of the central path leads to the main idea of the iterative methods for solving (4.1): Trace the central path while reducing μ at each iteration. Theoretically, an exact trace is wanted, but practically it is too inefficient. However, it has been shown that it is only sufficient to trace the central path approximately in order to achieve global convergence and maintain favorable convergence properties of the given algorithms.

We rewrite the system (4.4) as follows,

$$F(x, S) = \begin{bmatrix} Mx + q - S \\ Xs - \mu e \end{bmatrix} = 0 \tag{4.5}$$

As discussed previously, the IPMs trace the central path approximately. The general outline of the generic interior-point primal-dual method is discussed below. Without loss of generality, we assume that a point (x, s) is "close" to the μ-center, $(x(\mu), s(\mu))$ for some parameter $\mu > 0$. Then, μ is decreased to $\mu^+ := (1 - \theta)\mu$, for some $\theta \in (0, 1)$. Next, we redefine $\mu = \mu^+$, and we solve one step of Newton method

applied to function $F(x, s)$ in (4.5). This leads to,

$$\nabla F \begin{bmatrix} \Delta x \\ \Delta s \end{bmatrix} = -F(x, s) \tag{4.6}$$

which is equivalent to the following system.

$$\begin{aligned} -M\Delta x + \Delta s &= 0, \\ s\Delta x + x\Delta s &= \mu e - xs. \end{aligned} \tag{4.7}$$

Since M has full row rank, the system (4.7) has a unique solution for any $(x, s) > 0$. The solution $(\Delta x, \Delta s)$ is known as the Newton direction. By taking a step along this search direction, we construct a new iterate (x^+, s^+)

$$x^+ = x + \Delta x, \quad s^+ = s + \Delta s. \tag{4.8}$$

Here we choose to have a full Newton step, i.e, we choose a step-size to be equal to one. We repeat the procedure until we find iterates that are in a certain neighborhood of the μ-center $(x(\mu), s(\mu))$. Then, again, μ is reduced by the factor $1 - \theta$ and Newton's method is applied again targeting a new μ-center, and so on. We repeat this process until μ is small enough, i.e. $n\mu \leq \epsilon$, where ϵ is a small positive number. At this stage in the algorithm, we have found ϵ-approximate solutions of (4.1).

This is exactly the classical Newton method. The key feature of the IPM is to keep iterates close to the central path at all times. This is also a main difference of IPM and classical Newton Method. The convergence in this method is guaranteed by appropriate choice of parameters and by using two types of steps at each iterations; the Feasibility step and the Centering step. After one feasibility step an iterates may be "too far" from the central path (μ-center). After doing a couple of centering steps approximate solution will again be sufficiently close to the μ-center.

Scaling

Before formally stating the algorithm, we introduce important scaling that allows generalization and introduction of kernel-based barrier functions. For any triple (x, s, μ) with $x > 0$, $s > 0$ and $\mu > 0$, we introduce the so called *variance vector*:

$$v := \sqrt{\frac{xs}{\mu}}. \tag{4.9}$$

Note that the pair (x, s) coincides with the μ-center $(x(\mu), s(\mu))$ if and only if $v = e$.

The *scaled search directions* d_x and d_s are then defined as

$$d_x := \frac{v\Delta x}{x}, \quad d_s := \frac{v\Delta s}{s}, \tag{4.10}$$

where each of the operations are component-wise product and division.

Lemma 4.2.1. *If v is defined, as in (4.9) and the search directions d_x, d_s are defined as in (4.10), then the Newton system from (4.7) can be transformed into the following system:*

$$\begin{aligned} -\widetilde{M}d_x + d_s &= 0, \\ d_x + d_s &= v^{-1} - v, \end{aligned} \tag{4.11}$$

where

$$\widetilde{M} := DMD, \quad D := X^{\frac{1}{2}}S^{-\frac{1}{2}}, \quad S := diag(s), \quad and \quad X := diag(x).$$

Proof. Recall the Newton system given in (4.7)

$$-M\Delta x + \Delta s = 0, \tag{4.12}$$

$$s\Delta x + x\Delta s = \mu e - xs. \tag{4.13}$$

The scaled search directions d_x,d_s as defined in (4.10), can be rewritten as

$$\Delta x = \frac{xd_x}{v}, \quad \Delta s = \frac{sd_s}{v}, \tag{4.14}$$

where v is defined in (4.9).

By applying (4.14) to (4.13), we obtain

$$\begin{aligned} s\left(\tfrac{xd_x}{v}\right) + x\left(\tfrac{sd_s}{v}\right) &= \mu e - xs \\ \left(\tfrac{sx}{v}\right) d_x + \left(\tfrac{xs}{v}\right) d_s &= \mu e - xs \\ d_x + d_s &= \tfrac{v}{sx}\left(\mu e - xs\right) \\ d_x + d_s &= v^{-1} - v. \end{aligned}$$

We have shown the transformation for (4.13) using (4.14). Next we will focus our attention on transforming (4.12). If we apply (4.14) to (4.12), we see

$$-M\left(\frac{xd_x}{v}\right) + \left(\frac{sd_s}{v}\right) = 0. \tag{4.15}$$

The above equation can be transformed in the following way. First, observe that any vector $a \in \mathbb{R}^n$ can be written as

$$a = [\mathrm{diag}(a)]\, e,$$

where

$$\mathrm{diag}(a) = \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_n \end{bmatrix} \tag{4.16}$$

and e is a vector of all ones.

Therefore, vector $\frac{xd_x}{v}$ can be written as

$$\begin{aligned} \frac{xd_x}{v} &= \left(XV^{-1}D_x\right) e \\ &= XV^{-1}\left(D_x e\right) \\ &= XV^{-1}d_x \end{aligned} \tag{4.17}$$

where

$$X = \text{diag}(x), \quad V^{-1} = \text{diag}(v^{-1}), \quad D_x = \text{diag}(d_x).$$

Similarly, vector $\frac{sd_s}{v}$ can be written as

$$\begin{aligned} \frac{sd_s}{v} &= \left(SV^{-1}D_x\right)e \\ &= SV^{-1}\left(D_s e\right) \\ &= SV^{-1}d_s \end{aligned} \tag{4.18}$$

where

$$S = \text{diag}(s), \quad V^{-1} = \text{diag}(v^{-1}), \quad D_s = \text{diag}(d_s).$$

Substitution of (4.17) and (4.18) into (4.15) leads to

$$\begin{aligned} S^{-1}V\left(-MXV^{-1}d_x + SV^{-1}d_s\right) = 0 \\ -S^{-1}VMXV^{-1}d_x + d_s = 0 \end{aligned} \tag{4.19}$$

The matrix $S^{-1}VMXV^{-1}$ can be simplified by observing that

$$VS^{-1} = \text{diag}\left(\frac{v}{s}\right) = \text{diag}\left(\sqrt{\frac{x}{\mu s}}\right) = \frac{1}{\sqrt{\mu}}X^{\frac{1}{2}}S^{-\frac{1}{2}} = \frac{1}{\sqrt{\mu}}D. \tag{4.20}$$

and

$$XV^{-1} = \text{diag}\left(\frac{x}{v}\right) = \text{diag}\left(\sqrt{\frac{\mu x}{\mu s}}\right) = \sqrt{\mu}X^{\frac{1}{2}}S^{-\frac{1}{2}} = \sqrt{\mu}D. \tag{4.21}$$

where $D := X^{\frac{1}{2}}S^{-\frac{1}{2}}$, $S := \text{diag}(s)$, and X:= diag(x).

Next, by applying (4.20) and (4.21) to (4.19), we get

$$-\left[DMD\right]d_x + d_s = 0.$$

If we denote $\widetilde{M} := DMD$, we obtain

$$-\widetilde{M}d_x + d_s = 0.$$

Hence, the lemma is proved. □

Lemma 4.2.2. *If matrix M is positive semi-definite, then $\widetilde{M}$ is also positive semi-definite.*

Proof. Let $a \in \mathbb{R}^n$ and $\widetilde{M}$ be as defined above, then

$$\begin{aligned} a^T \widetilde{M} a &= a^T (DMD) a \\ &= \left(a^T D\right) M (Da) \\ &= (Da)^T M (Da) \\ &\geq 0. \end{aligned}$$

By assumption, we know M is positive semi-definite. Hence, by definition, $\widetilde{M}$ is positive semi-definite. □

Proximity measure

We need proximity measure to check how close we are to the μ- center. Note that

$$v^{-1} - v = 0 \quad \Leftrightarrow \quad v = e.$$

Therefore, we see that $v = e$ if and only if the pair (x, s) coincides with the μ-center $(x(\mu), s(\mu))$.

A very important observation is that the right hand side $v^{-1} - v$ in the last equation of (4.11) equals the negative gradient of the function

$$\Psi(v) := \sum_{i=1}^{n} \left(\frac{v_i^2 - 1}{2} - \log v_i \right), \tag{4.22}$$

which can be written as,

$$d_x + d_s = -\nabla \Psi(v). \tag{4.23}$$

This equation is known as the *scaled centering equation.* The scaled centering equation basically defines the search directions. An easy verification is that $\nabla^2 \Psi(v) = \text{diag}(e +$

v^{-2}). Since this matrix is positive definite, $\Psi(v)$ is strictly convex. We can see that $\nabla\Psi(e) = 0$, hence $\Psi(v)$ attains its minimal value at $v = e$, with $\Psi(e) = 0$. So, it follows that $\Psi(v)$ is non-negative everywhere and vanishes at $v = e$, which means it vanishes at the μ-center $(x(\mu), s(\mu))$. Therefore, we can conclude that the μ-center $(x(\mu)s(\mu))$ can be characterized as the minimizer of the function $\Psi(v)$. Thus, $\Psi(v)$ serves as a measure of how close (x, s) is to the μ-center.

From the discussion above we see that $\|\nabla\Psi(v)\|$ can also serve as a proximity measure. Therefore we define

$$\delta(x, s, \mu) = \delta(v) = \frac{1}{2}\left\|v - v^{-1}\right|\| = \frac{1}{2}\|\nabla\Psi(v)\|. \tag{4.24}$$

4.3 Full Newton-step Interior-Point Algorithm for LCP

We can now formally describe the Full Newton-step Interior Point algorithm. As we mentioned, this algorithm follows the central path approximately. Suppose we start with (x, s) close to μ-center. We start with outer iteration by first reducing μ to $\mu^+ = (1 - \theta)\mu$. Therefore, new v becomes $v^+ = \frac{v}{\sqrt{1-\theta}}$. Then we find the directions dx and ds by solving the system (4.11) and calculate the original directions Δx and Δs by solving the Newton system (4.14). These are called feasibility directions. We update x and s using the recently found search directions Δx and Δs and by solving the system (4.8).

As a consequence, $\Psi(v)$ changes to $\Psi(v^+)$. The inequality, $\Psi(v) \leq \tau$, means that (x, s) is in a τ-neighborhood of the μ-center $(x(\mu), s(\mu))$, where $\tau > 0$ represents a certain threshold value. Recall that, we measure the closeness of (x, s) to μ-center

$(x(\mu), s(\mu))$ by the value of $\Psi(v)$. However, after the θ-update of μ, the updated $\Psi(v^+)$ may be greater than τ, if so, we need to perform further steps to reduce $\Psi(v^+)$ to get closer to the new μ-center, i.e, to get back to the τ-neighborhood of a new μ-center.

To accomplish this, we perform inner iteration where we find the directions dx and ds by solving the system (4.11). Since x and s have changed in the outer iteration, v differs at every iteration. Then we calculate the original direction Δx and Δs by solving the Newton system (4.14). We update x and s using the recently found search directions Δx and Δs, by solving the system (4.8). This process is repeated until $\Psi(v) \leq \tau$, upon which the process begins again. We begin again by reducing μ and updating v, and so on until we have an iterate that is ϵ-close to the actual solution. The Full Newton-step form of the Algorithm is shown in Table 4.1. In the sequel, we will refer to it as simply the *Algorithm*.

The graphical representation of the Algorithm is given in Figure 4.1

Full Newton-step Interior-Point Algorithm for LCP

Input:

Determine input parameters:

threshold parameter $\tau > 0$,

fixed barrier update parameter θ, $0 < \theta < 1$,

accuracy parameter $\epsilon > 0$.

begin

Set $(x_0, s_0, \mu_0) > 0$ so that the IPC is satisfied;

while $n\mu \geq \epsilon$ **do**

$\mu := (1 - \theta)\mu$;

$v := \sqrt{\frac{xs}{\mu}}$;

Calculate direction (dx, ds) by solving (4.11);

Calculate original direction $(\Delta x, \Delta s)$ by solving (4.14);

Update $x := x + \Delta x$ and $s := s + \Delta s$;

Update $v := \sqrt{\frac{xs}{\mu}}$;

while $\Psi(v) > \tau$ **do**

Calculate direction (dx, ds) by solving (4.11);

Calculate original direction $(\Delta x, \Delta s)$ by solving (4.14);

Update $x := x + \Delta x$ and $s := s + \Delta s$;

Update $v := \sqrt{\frac{xs}{\mu}}$;

end do

end do

end

Table 4.1: Full Newton-step Interior-Point Algorithm for LCP

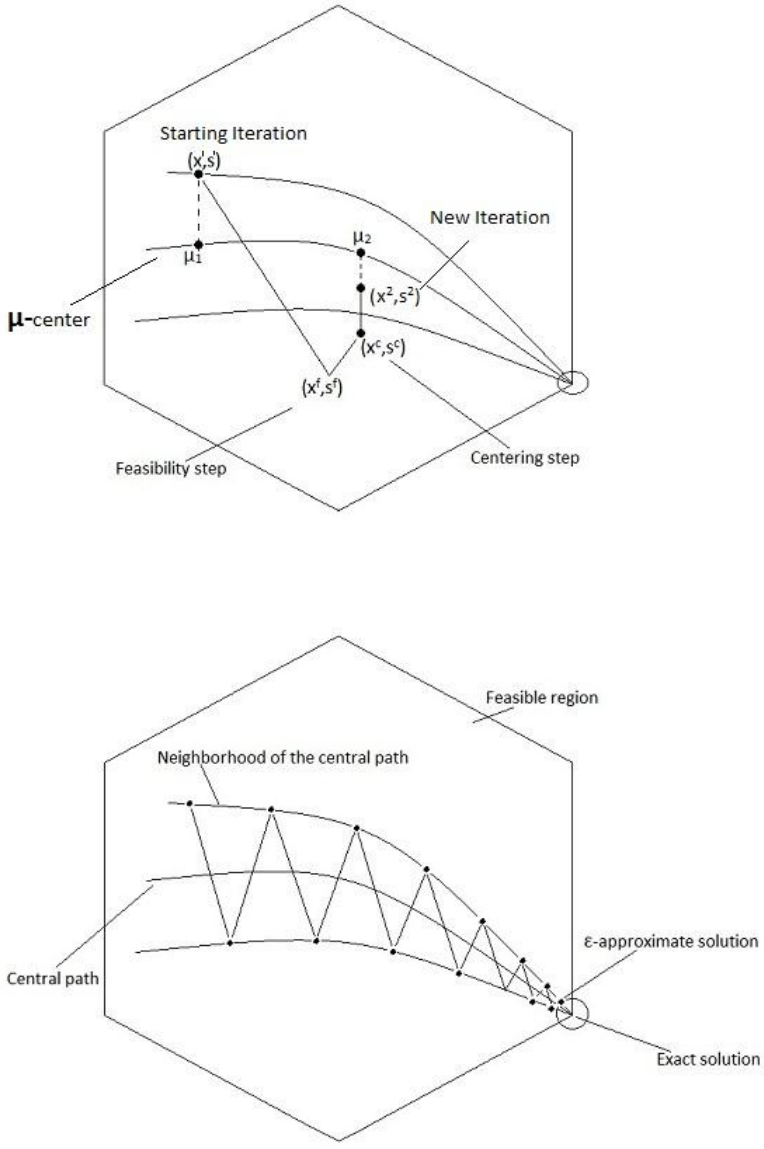

Figure 4.1: Graphical representation of the Algorithm.

CHAPTER 5

ANALYSIS OF THE ALGORITHM

5.1 Some Useful Inequalities

Before we start the analysis of the algorithm, we derive some equations and inequalities that will be frequently used in the sequel.

From the equations (4.9) and (4.10) in Chapter 4, we have following equation.

$$\begin{aligned} dxds &= \frac{v\Delta x}{x}\frac{v\Delta s}{s} \\ &= \frac{\frac{\sqrt{xs}}{\mu}\frac{\sqrt{xs}}{\mu}}{xs}\Delta x\Delta s \\ &= \frac{\Delta x\Delta s}{\mu}. \end{aligned} \tag{5.1}$$

From the equation (4.11) in Chapter 4, $dx+ds = v^{-1}-v$ and $\delta(x,s,\mu) = \delta(v) = \frac{1}{2}\left\|v^{-1}-v\right\|$, therefore $\left\|dx+ds\right\|^2 = \left\|v^{-1}-v\right\|^2 = 4\delta^2$. Furthermore,

$$\begin{aligned} \left\|dx+ds\right\|^2 &= (dx+ds)^T(dx+ds) \\ &= dx^Tdx + ds^Tds + 2dx^Tds \\ &= \left\|dx\right\|^2 + \left\|ds\right\|^2 + 2dx^Tds. \end{aligned} \tag{5.2}$$

From the equation (4.11) in Chapter 4, $\widetilde{M}dx = ds$. Since $\widetilde{M}$ is positive semi-definite, we can write $dx^Tds = dx^T\widetilde{M}dx \geq 0$ so $dx^Tds \geq 0$.

Let $p_v = dx+ds$ and $q_v = dx-ds$ then

$$\begin{aligned} \left\|p_v\right\|^2 - \left\|q_v\right\|^2 &= p_v^Tp_v - q_v^Tq_v \\ &= (dx+ds)^T(dx+ds) - (dx-ds)^T(dx-ds) \\ &= 4dx^Tds. \end{aligned} \tag{5.3}$$

Since $dx^T ds \geq 0$, we conclude that $\|p_v\|^2 \geq \|q_v\|^2$. Furthermore, we can write

$$\begin{aligned} dx^T ds &\leq \tfrac{1}{4} \|p_v\|^2 \\ &= \tfrac{1}{4} \|dx + ds\|^2 \\ &= \delta^2. \end{aligned} \tag{5.4}$$

Therefore we have

$$0 \leq dx^T ds \leq \delta^2. \tag{5.5}$$

From the equation (5.3) we get

$$\begin{aligned} \|p_v\|^2 - \|q_v\|^2 &= 4dx^T ds \\ \|q_v\|^2 &= \|p_v\|^2 - 4dx^T ds \\ &\leq \|p_v\|^2, \qquad \text{since} \;\; dx^T ds \geq 0 \\ &= \|dx + ds\|^2 \\ &= 4\delta^2. \end{aligned} \tag{5.6}$$

Now we consider $p_v^2 - q_v^2$:

$$\begin{aligned} p_v^2 - q_v^2 &= (dx - ds)^2 - (dx + ds)^2 \\ &= 4dxds \\ dxds &= \tfrac{1}{4}(p_v^2 - q_v^2) \\ |dxds| &= \tfrac{1}{4} \left|(p_v^2 - q_v^2)\right|. \end{aligned} \tag{5.7}$$

Case I : $p_v^2 - q_v^2 \geq 0$. Given that $p_v^2 - q_v^2 \leq p_v^2$ it follows that

$$\begin{aligned} |dxds| &= \tfrac{1}{4} |p_v^2 - q_v^2| \\ &= \tfrac{1}{4}(p_v^2 - q_v^2) \\ &\leq \tfrac{1}{4} p_v^2 \\ &= \tfrac{1}{4} |p_v|^2. \end{aligned} \tag{5.8}$$

Case II : $p_v^2 - q_v^2 \leq 0$. This implies $q_v^2 - p_v^2 \geq 0$, i.e. $|p_v^2 - q_v^2| \leq q_v^2$.

$$\begin{aligned} |dxds| &= \tfrac{1}{4} |p_v^2 - q_v^2| \\ &\leq \tfrac{1}{4} |q_v|^2 . \end{aligned} \tag{5.9}$$

Thus,

$$max_i \, |dx_i ds_i| \leq \tfrac{1}{4} max \left\{ |p_v|^2 , |q_v|^2 \right\},$$

which leads to

$$max_i \, |dx_i ds_i| = \|dxds\|_\infty \leq \tfrac{1}{4} max \left\{ \|p_v\|^2 , \|q_v\|^2 \right\}$$

Therefore, from equation (5.6) we have,

$$\|dxds\|_\infty \leq \delta^2. \tag{5.10}$$

Next, we have

$$\begin{aligned} \|dxds\|^2 &= (dxds)^T (dxds) \\ &= (dx_1 ds_1)^2 + (dx_2 ds_2)^2 + ... + (dx_n ds_n)^2 \\ &\leq (dx_1 ds_1 + dx_2 ds_2 + ... + dx_n ds_n)^2 \\ &= (dx^T ds)^2 \\ &\leq \delta^4. \end{aligned} \tag{5.11}$$

Hence, $\|dxds\| \leq \delta^2$.

Now, we can easily obtain similar inequalities for Δx and Δs:

$$\begin{aligned} \Delta x^T \Delta s &= (xv^{-1} dx)^T (sv^{-1} ds) \\ &= (dx \sqrt{\tfrac{x}{s}} \sqrt{\mu})^T (ds \sqrt{\tfrac{s}{x}} \sqrt{\mu}) \\ &= \mu (dx \sqrt{\tfrac{x}{s}})^T (ds \sqrt{\tfrac{s}{x}}) \\ &= \mu dx^T ds \\ &\leq \mu \delta^2, \end{aligned} \tag{5.12}$$

$$\begin{aligned} \|\Delta x \Delta s\|_\infty &= max_i \, |\Delta x_i \Delta s_i| \\ &= max_i \, |\mu dx_i ds_i| \\ &= \mu max_i \, |dx_i ds_i| \\ &= \mu \, \|dxds\|_\infty \\ &\leq \mu \delta^2, \end{aligned} \tag{5.13}$$

$$\begin{aligned} \|\Delta x \Delta s\|^2 &= \textstyle\sum_{i=1}^n (\Delta x_i \Delta s_i)^2 \\ &= \textstyle\sum_{i=1}^n \mu^2 (dx_i ds_i)^2 \\ &= \mu^2 \textstyle\sum_{i=1}^n (dx_i ds_i)^2 \\ &= \mu^2 \, \|dxds\|^2 , \\ \|\Delta x \Delta s\| &= \mu \, \|dxds\| \\ &\leq \mu \delta^2. \end{aligned} \tag{5.14}$$

5.2 Analysis of a Full Newton-step

Keep μ fixed.

Let $x^+ = x + \Delta x$ and $s^+ = s + \Delta s$. Then,

$$\begin{aligned} x^+ s^+ &= (x + \Delta x)(s + \Delta s) \\ &= xs + x\Delta s + s\Delta x + \Delta x \Delta s \\ &= xs + (\mu e - xs) + \Delta x \Delta s \\ &= \mu e + \Delta x \Delta s. \end{aligned} \tag{5.15}$$

From the equation (5.1) we have,

$$\begin{aligned} x^+ s^+ &= \mu e + \Delta x \Delta s \\ &= \mu e + \mu dxds \\ &= \mu(e + dxds). \end{aligned} \tag{5.16}$$

Lemma 5.2.1. *The full Newton step is feasible iff*

$$e + dxds \geq 0, \tag{5.17}$$

and strictly feasible iff

$$e + dxds > 0. \tag{5.18}$$

Proof. a) If full-Newton step is feasible then $x^+ \geq 0$ and $s^+ \geq 0$. Hence, $x^+s^+ \geq 0$ and from (5.16) it follows $e + dxds \geq 0$.

If Newton step is strictly feasible then $x^+ > 0$ and $s^+ > 0$. Hence, $x^+s^+ > 0$ and from (5.16) it follows $e + dxds > 0$.

b) If $(e + dxds) \geq 0$ then from the equation (5.15) $\mu e + \Delta x \Delta s \geq 0$. Let step length α be defined in the interval $0 \leq \alpha \leq 1$. Then $x^\alpha = x + \alpha\Delta x$ and $s^\alpha = s + \alpha\Delta s$. When $\alpha = 0$, $x^0 = x$ and $s^0 = s$. When $\alpha = 1$, then $x^1 = x + \Delta x$, i.e $x^1 = x^+$ and $s^1 = s + \Delta s$, i.e $s^1 = s^+$.

Since we have $x^0 s^0 = xs > 0$, the proof uses a continuity argument, namely that x^1 and s^1 are nonnegative if $x^\alpha s^\alpha$ is positive for all α in the open interval (0,1).

The points x^1 and s^1 are feasible iff the open segment connecting x^0 and x^1 lies in the interior of the feasible region, and the s^0 and s^1 lies in the interior of the feasible region. We have

$$\begin{aligned} x^\alpha s^\alpha &= (x + \alpha\Delta x)(s + \alpha\Delta s) \\ &= xs + \alpha(x\Delta s + s\Delta x) + \alpha^2 \Delta x \Delta s \\ &= xs + \alpha(\mu e - xs) + \alpha^2 \Delta x \Delta s \\ &\geq xs + \alpha(\mu e - xs) - \alpha^2 \mu e \ \text{ since } \ (\mu e + \Delta x \Delta s \geq 0) \\ &= (1 - \alpha)(xs + \alpha\mu e). \end{aligned} \tag{5.19}$$

Since xs, e and μ are positive, $x^\alpha s^\alpha \geq 0$ for $0 \leq \alpha < 1$. Therefore by continuity, the vectors x^1 and s^1 can not have negative entries. □

Lemma 5.2.2. *If $\delta(x^+, s^+, \mu) \leq 1$, then the full-Newton step is feasible, i.e, x^+ and s^+ are nonnegative. Moreover, if $\delta < 1$, then x^+ and s^+ are positive i.e. they are strictly feasible and*

$$\delta(x^+, s^+, \mu) \leq \tfrac{\delta^2}{2\sqrt{1-\delta^2}}.$$

Proof. Let $\delta^+ = \delta(x^+, s^+, \mu)$ and $v^+ = \sqrt{\frac{x^+ s^+}{\mu}}$. Since $\delta(v) = \frac{1}{2}\left\|v^{-1} - v\right\|$, we have

$$\begin{aligned} \delta^+ &= \tfrac{1}{2}\left\|(v^+)^{-1} - v^+\right\| \\ &= \tfrac{1}{2}\left\|(v^+)^{-1}(e - (v^+)^2)\right\|. \end{aligned} \tag{5.20}$$

From (5.16) we have $x^+ s^+ = \mu(e + dxds)$, and v^+ becomes $v^+ = \sqrt{e + dxds}$. By substituting this value into the equation (5.20) we get

$$\begin{aligned} 2\delta^+ &= \left\|(\sqrt{e + dxds})^{-1} - (e - e - dxds)\right\| \\ &= \left\|\tfrac{dxds}{\sqrt{e+dxds}}\right\| \\ &\leq \tfrac{\|dxds\|}{\left\|\sqrt{e+dxds}\right\|} \\ &\leq \tfrac{\|dxds\|}{\sqrt{1-\|dxds\|_\infty}} \\ &\leq \tfrac{\delta^2}{\sqrt{1-\delta^2}}. \end{aligned} \tag{5.21}$$

Thus,

$$\delta^+ \leq \frac{\delta^2}{2\sqrt{1-\delta^2}}$$

and the lemma is proved. □

Corollary 5.2.3. *If $\delta(x, s, \mu) \leq \frac{1}{\sqrt{2}}$ then $\delta(x^+, s^+, \mu) \leq \delta^2(x, s, \mu)$.*

Proof. From Lemma 5.2.2 we have

$$\begin{aligned}\delta(x^+, s^+, \mu) &\leq \frac{\delta^2(x,s,\mu)}{2\sqrt{1-\delta^2(x,s,\mu)}} \\ &\leq \frac{\delta^2(x,s,\mu)}{2\sqrt{1-\frac{1}{2}}} \\ &= \frac{\delta^2(x,s,\mu)}{\sqrt{2}} \\ &\leq \delta^2(x,s,\mu),\end{aligned} \tag{5.22}$$

which proves the corollary. □

Corollary 5.2.3 actually indicates that we have quadratic convergence if the iterates are sufficient close to the μ center.

We also have the following lemma

Lemma 5.2.4. $(x^+)^T s^+ \leq \mu(n + \delta^2)$

Proof. We have

$$\begin{aligned}(x^+)s^+ &= e^T(x^+ s^+) \\ &= e^T(x + \Delta x)(s + \Delta s) \\ &= e^T(\mu e + \Delta x \Delta s) \\ &= \mu e^T e + e^T \Delta x \Delta s \\ &= \mu n + \Delta x^T \Delta s \\ &\leq \mu + \mu\delta^2 \\ &= \mu(n + \delta^2).\end{aligned} \tag{5.23}$$

□

The immediate consequence is the following corollary.

Corollary 5.2.5.

$$\|v\|^2 \leq n + \delta^2. \tag{5.24}$$

Proof. We have

$$\begin{aligned} \|v\|^2 &= v^T v \\ &= (\sqrt{\tfrac{xs}{\mu}})^T(\sqrt{\tfrac{xs}{\mu}}) \\ &= \tfrac{1}{\mu}(x_1 s_1 + x_2 s_2 + ... + x_n s_n) \\ &\leq \tfrac{1}{\mu}(\mu(n + \delta^2)) \\ &= n + \delta^2. \end{aligned} \tag{5.25}$$

□

$\underline{\mu \textbf{ update, } \mu \to \mu^+}$.

Lemma 5.2.6. *Let $(x, s) > 0$ be a current iterate and $\mu > 0$ such that $\delta = \delta(x, s, \mu)$ and $\mu^+ = (1 - \theta)\mu$. Then*

$$\delta(x, s, \mu^+)^2 \leq \frac{3}{4}\delta^2(1 - \theta) + \frac{\delta^2}{4(1 - \theta)} + \frac{n\theta^2}{4(1 - \theta)}. \tag{5.26}$$

Proof. since $\delta(x, s, \mu)^2 = \frac{1}{4}\left\|v^{-1} - v\right\|^2$, or equivalently $4\delta(x, s, \mu^+)^2 = \left\|(v^+)^{-1} - v^+\right\|^2$, we have

$$\begin{aligned} 4\delta(x, s, \mu^+)^2 &= \left\| \sqrt{\tfrac{\mu^+}{xs}} - \sqrt{\tfrac{xs}{\mu^+}} \right\|^2 \\ &= \left\| \sqrt{1 - \theta} v^{-1} - \tfrac{v}{\sqrt{1-\theta}} \right\|^2 \\ &= \left\| \sqrt{1 - \theta}(v^{-1} - v) + \sqrt{1 - \theta} v - \tfrac{v}{\sqrt{1-\theta}} \right\|^2 \\ &= \left\| \sqrt{1 - \theta}(v^{-1} - v) - \tfrac{\theta v}{\sqrt{1-\theta}} \right\|^2 \\ &= (\sqrt{1 - \theta}(v^{-1} - v) - \tfrac{\theta v}{\sqrt{1-\theta}})^T(\sqrt{1 - \theta}(v^{-1} - v) - \tfrac{\theta v}{\sqrt{1-\theta}}) \end{aligned}$$

$$\begin{aligned}
4\delta(x,s,\mu^+)^2 &= (1-\theta)\left\|(v)^{-1}-v\right\|^2 + \tfrac{\theta^2\|v\|^2-(2\theta-2\theta^2)(v^Tv^{-1}-\theta^2\|v\|^2)}{(1-\theta)} \\
&= (1-\theta)4\delta^2 - \tfrac{\|v\|^2\theta(2-\theta)-n(2\theta-2\theta^2)}{(1-\theta)} \\
&\leq (1-\theta)4\delta^2 - \tfrac{(n+\delta^2)\theta(2-\theta)-n(2\theta-2\theta^2)}{(1-\theta)} \\
&= (1-\theta)4\delta^2 + \tfrac{\delta^2\theta(2-\theta)}{(1-\theta)} + \tfrac{n\theta^2}{(1-\theta)} \\
&= 3\delta^2(1-\theta) + \tfrac{\delta^2}{(1-\theta)} + \tfrac{n\theta^2}{(1-\theta)}.
\end{aligned}$$

The inequality above is due to (5.24). □

When $\delta(x,s,\mu) \leq \frac{1}{2}$, the goal is to find θ value which will satisfy $\delta(x,s,\mu^+) \leq \frac{1}{\sqrt{2}}$. From the Lemma 5.2.6. we have

$$\begin{aligned}
\delta(x,s,\mu^+)^2 &\leq \tfrac{3}{4}\delta^2(1-\theta) + \tfrac{\delta^2}{4(1-\theta)} + \tfrac{n\theta^2}{4(1-\theta)} \\
&\leq \tfrac{3}{16}(1-\theta) + \tfrac{1}{16(1-\theta)} + \tfrac{n\theta^2}{4(1-\theta)}.
\end{aligned} \tag{5.27}$$

By substituting $\theta = \frac{1}{\sqrt{kn}}$ into the above inequality, we have

$$\begin{aligned}
\delta(x,s,\mu^+)^2 &\leq \tfrac{1}{16(1-\theta)}[3(1-\theta)^2 + 1 + 4\theta^2 n] \\
&\leq \tfrac{1}{16(1-\frac{1}{\sqrt{kn}})}[4 + \tfrac{3}{kn} - \tfrac{6}{\sqrt{kn}} + \tfrac{4n}{kn}] \leq \tfrac{1}{2}
\end{aligned} \tag{5.28}$$

So we have $-4kn + 2\sqrt{kn} + 4n + 3 \leq 0$. Then $k = \frac{1 \mp \sqrt{1+4(3+4n)^2}}{16n}$ since we need the biggest θ we have to choose the smallest k; thus, we choose $k = \frac{1-\sqrt{1+4(3+4n)^2}}{16n}$. After doing several simplifications we get $k \leq 2$ for $n \geq 3$. We can summarize this result in the following corollary.

Corollary 5.2.7. *If $\delta(x,s,\mu) \leq \frac{1}{2}$ and $\theta = \frac{1}{\sqrt{2n}}$, then $\delta(x,s,\mu^+) \leq \frac{1}{\sqrt{2}}$.*

Combining Lemma 5.2.6 and Corollary 5.2.3 we get the following theorem.

Theorem 5.2.8. *If $\delta(x,s,\mu) \leq \frac{1}{2}$ then $\delta(x^+,s^+,\mu^+) \leq \frac{1}{2}$.*

Proof. Let $\delta(x,s,\mu) \leq \frac{1}{2}$. Then, by Corollary 5.2.7 we have $\delta(x,s,\mu^+) \leq \frac{1}{\sqrt{2}}$. Next, by Corollary 5.2.3 we get

$$\delta(x^+,s^+,\mu^+) \leq \delta^2(x,s,\mu^+) \leq \tfrac{1}{2}.$$

□

Thus, all the iterates of the algorithm are guaranteed to be in the same neighborhood ($\tau = \frac{1}{2}$) of the central path. This leads to the following estimate on the number of iterations to obtain ϵ- approximate solution of the LCP.

Theorem 5.2.9. *If $\theta = \frac{1}{\sqrt{2n}}$, $\mu_0 = \frac{1}{\sqrt{2}}$, $\delta(x,s,\mu) \leq \frac{1}{2}$ and $n \geq 3$, then the Full Newton-step IPM requires at most $\sqrt{2n}\ log\frac{n}{\epsilon}$ iterations to obtain ϵ-approximate solution of LCP(M,q).*

Proof. At the start of the Algorithm duality gap has a certain value and in each iteration duality gap is reduced by the factor $1-\theta$. The duality gap can be transformed as follows

$$\begin{aligned} x_k^T s_k &\leq \mu_k(n+\delta^2) \\ &\leq \mu_k(n+\tfrac{1}{4}) \\ &\leq (1-\theta)^k\mu_0(n+\tfrac{1}{4}) \\ &\leq (1-\theta)^k\tfrac{1}{\sqrt{2}}\sqrt{2}n \\ &\leq (1-\theta)^k n. \end{aligned}$$

Let $(1-\theta)^k n \leq \epsilon$. Then

$$\begin{aligned} log(1-\theta)^k n &\leq log\epsilon \\ klog(1-\theta) + logn &\leq log\epsilon \\ -klog(1-\theta) &\geq logn - log\epsilon \\ -klog(1-\theta) &\geq log\tfrac{n}{\epsilon}, \end{aligned}$$

Since $-log(1-\theta) \geq \theta$, we have

$$\begin{aligned} -k\theta &\geq log\tfrac{n}{\epsilon} \\ k &\geq \tfrac{1}{\theta} log\tfrac{n}{\epsilon} \\ k &\geq \sqrt{2n} log\tfrac{n}{\epsilon}. \end{aligned}$$

which completes the proof.

□

CHAPTER 6

NUMERICAL RESULTS

In this chapter, The Full Newton-step interior-Point Algorithm for LCP, as given in Table 4.1, is implemented in MATLAB. We performed numerical tests of our implementation of the algorithm for the set of problems of various dimensions. Some problems were generated "by hand" and others were randomly generated. The summary of results is given in tables below.

6.1 Generating Sample Problems

In what follows we briefly describe how the test problems were generated.

Generating Matrix M

The first group of problem was generated manually. Actually problem $EH1$ is the same problem as the one in Example 3.3 in Chapter 3 (Lemke's Method). The PSD matrices of the problems were generated as either diagonal matrices with positive elements or tridiagonal matrices with positive elements on diagonal.

In the second group matrices of the sample problem were randomly generated by using "rand" function as described below.

$$\begin{aligned} A &= rand(n) \\ M &= A^T A. \end{aligned}$$

PSD was checked by evaluating eigenvalues, i.e.,

$$eig(M) \geq 0.$$

Starting points and initial conditions

In creating the appropriate starting point we first choose x^0 as a vector of ones. Then s^0 is calculated as $Mx^0 + q = s^0$. However, we also need to make sure that initial conditions for the application of the algorithm are satisfied, that is, we need to check if $(x^0, s^0) > 0$ and $\delta \leq \frac{1}{2}$ are satisfied, where δ is defined by (4.24) in Chapter 4.

Parameters

We also tested several sets of parameters. First, we take the set of parameters $\tau = \frac{1}{3}$ and $\theta = \frac{1}{\sqrt{2n}}$ as required by the algorithm in order to guarantee convergence.

Next, we try a wider τ-neighborhoods ($\tau = \frac{1}{2}$, $\tau = 0.9$) and more aggressive reduction of μ-parameter at each iteration, by taking barrier parameter to be a fixed value independent of the size of the problem ($\theta = \frac{1}{\sqrt{6}}$, $\theta = 0.5$). In these cases we can not guarantee convergence, however, in most instances the algorithm still converges.

6.2 Summary of Numerical Results

We generated 12 test problems. Five of them were generated manually (denoted as EH) with dimensions up to n=10 and seven were randomly generated with dimensions up to n=300. We solved this set of test problems with the following set of parameters.

1. $\tau = \frac{1}{3}$ and $\theta = \frac{1}{\sqrt{2n}}$
2. $\tau = \frac{1}{3}$ and $\theta = \frac{1}{\sqrt{6}}$
3. $\tau = \frac{1}{2}$ and $\theta = \frac{1}{\sqrt{6}}$
4. $\tau = 0.9$ and $\theta = 0.5$

5. $\tau = 0.9$ and $\theta = 0.9$

The number of inner and outer iterations as well as CPU time for each case are listed in the tables below.

1. $\tau = \frac{1}{3}$ and $\theta = \frac{1}{\sqrt{2n}}$.

$Problem$	$Size$	$CPUtime$	$InnerIter$	$OuterIter$
$EH1$	3×3	0.366277	2	71
$EH2$	4×4	$6.7580e-02$	1	86
$EH3$	5×5	$3.9294e-02$	1	99
$EH4$	6×6	$6.6948e-02$	2	111
$EH5$	10×10	$7.2536e-02$	2	151
$ER1$	3×3	$2.8160e-02$	1	71
$ER2$	5×5	$4.9308e-02$	2	99
$ER3$	10×10	$5.1607e-02$	3	151
$ER4$	50×50	4.409072	5	374
$ER5$	100×100	6.02802	13	541
$ER6$	200×200	25.572454	14	800
$ER7$	300×300	84.213142	16	994

Tab.1, $\theta = \frac{1}{\sqrt{2n}}$, $\tau = \frac{1}{3}$

2. In the Table 2 below we fixed the barrier update parameter θ for all the examples and we did not change threshold parameter. So $\tau = \frac{1}{3}$ and $\theta = \frac{1}{\sqrt{6}}$.

Problem	*Size*	*CPUtime*	*InnerIter*	*OuterIter*
$EH1$	3×3	$6.7547e-02$	2	71
$EH2$	4×4	$4.5001e-02$	1	71
$EH3$	5×5	$5.8841e-02$	1	72
$EH4$	6×6	$5.8032e-02$	2	72
$EH5$	10×10	$5.1062e-02$	2	73
$ER1$	3×3	$7.8325e-02$	8	71
$ER2$	5×5	$4.9308e-02$	8	72
$ER3$	10×10	$5.9844e-02$	10	73
$ER4$	50×50	1.488650	12	76
$ER5$	100×100	1.901632	13	77
$ER6$	200×200	3.572454	14	79
$ER7$	300×300	9.233684	15	79

Tab.2, $\theta = \frac{1}{\sqrt{6}}, \tau = \frac{1}{3}$

We can observe that in all instances the number of outer iterations is significantly reduced in comparing with Table 1 and is almost independent of the dimension of the problem. That, of course, reflects on the significant reduction of CPU time as well.

Although the convergence is not guaranteed any more we see that the algorithm still converges for all test problems.

3. In the Table 3 below we increase the threshold parameter and keep the barrier update parameter as in Table 2. So $\tau = \frac{1}{2}$ and $\theta = \frac{1}{\sqrt{6}}$.

$Problem$	$Size$	$CPUtime$	$InnerIter$	$OuterIter$
$EH1$	3×3	0.018107	2	71
$EH2$	4×4	$3.4474e-02$	1	71
$EH3$	5×5	$3.9648e-02$	1	72
$EH4$	6×6	$4.6360e-02$	1	72
$EH5$	10×10	$2.6462e-02$	2	73
$ER1$	3×3	$2.9832e-02$	1	71
$ER2$	5×5	$2.9494e-02$	1	72
$ER3$	10×10	$3.3830e-02$	3	73
$ER4$	50×50	0.95765	6	76
$ER5$	100×100	1.301632	7	77
$ER6$	200×200	3.282454	8	79
$ER7$	300×300	9.023684	8	79

Tab.3, $\theta = \frac{1}{\sqrt{6}}, \tau = \frac{1}{2}$

Increasing threshold parameter means that we increase the neighborhood. By comparing Table 2 and Table 3 we can see that the number of outer iterations seems to be almost the same for both. But number of inner iterations in Table 3 reduced significantly. The reason is that number of outer iteration depends mostly on θ while number of inner iteration depend mostly on τ.

4. In the Table 4 below we increase both parameter values, $\tau = 0.9$ and $\theta = 0.5$.

$Problem$	$Size$	$CPUtime$	$InnerIter$	$OuterIter$
$EH1$	3×3	0.035081	2	54
$EH2$	4×4	$1.4631e-02$	1	54
$EH3$	5×5	$4.0955e-02$	1	54
$EH4$	6×6	$1.6545e-02$	1	55
$EH5$	10×10	$2.1862e-02$	2	55
$ER1$	3×3	$1.6553e-02$	1	54
$ER2$	5×5	$2.7308e-02$	1	54
$ER3$	10×10	$2.8403e-02$	2	55
$ER4$	50×50	0.58698	5	57
$ER5$	100×100	1.038574	7	59
$ER6$	200×200	2.57954	8	60
$ER7$	300×300	6.97584	8	60

Tab.4, $\theta = 0.5, \tau = 0.9$

From Table 4 we can conclude that by increasing θ, number of outer iterations reduces significantly. However, increasing τ value does not affect to the number of inner iterations almost at all. The reason may be that the increase in value of $\psi(v)$ is either moderate or significant and number of these cases does not change much when τ is increased beyond certain value ($\tau = \frac{1}{2}$ seems to be that value).

5. In the Table 5 below we increase parameter θ to $\theta = 0.9$.

$Problem$	$Size$	$CPUtime$	$InnerIter$	$OuterIter$
$EH1$	3×3	0.60936	2	17
$EH2$	4×4	$1.9631e-02$	1	17
$EH3$	5×5	$4.0355e-02$	2	17
$EH4$	6×6	$4.0545e-02$	2	17
$EH5$	10×10	$4.1862e-02$	2	17
$ER1$	3×3	$4.0053e-02$	1	16
$ER2$	5×5	$4.0308e-02$	3	17
$ER3$	10×10	$1.8403e-02$	4	17
$ER4$	50×50	0.09698	7	13
$ER5$	100×100	0.238574	8	18
$ER6$	200×200	1.57954	9	18
$ER7$	300×300	3.56584	10	18

Tab.5, $\theta = 0.9$,$\tau = 0.9$

The result we obtained from the Table 5 above shows that by increasing the θ value we further continue to reduce the number of outer iterations significantly. However, the number of inner iterations increases slightly. The reason is that targeting μ-centers that are further appart will most likely result in the higher number of cases when $\psi(v) > \tau$, which triggers inner iteration calculations. The above analysis shows clearly advantage of IPM versus Lemke's algorithm for large dimensional LCPs.

Note: We would also like to point out that the solution of EH 1 obtained using our IPM matches the solution obtained using classical Lemke's algorithm. That is a strong indicator of the correctness of our implementation of IPM.

CHAPTER 7

CONCLUSION

In this thesis, we consider the Linear Complementarity Problem (LCP) defined by (2.1) with positive semidefinite matrix (PSD), which is also known as monotone LCP . Although LCP is not an optimization problem, it is closely related to many important optimization problems and it has many important applications.

THe LCP problem can be solved using classical simplex-type (pivoting) Lemke's algorithm that is described in Chapter 3. However in the last two decades a new class of Newton-type IPM have been developed and successfully applied to solve LCP.

We propose a new IPM to solve monotone LCP. The algorithm is given in Table 4.1 in Chapter 4. Main feature of the IPM is that there is no calculation of the step size, i.e, we use full Newton step at each iteration; thus, we call the algorithm Full-Newton-Step IPM. The convergence of the algorithm is guaranteed by the appropriate choice of parameters. Furthermore, we have proved that iteration bound is $O(\sqrt{n}log\frac{n}{\epsilon})$ which matches the best known iteration bound for these type of algorithms.

If θ depend on n such as $\theta = O(\frac{1}{\sqrt{n}})$, then the algorithm is called a short-step algorithm. If θ is independent of n such as $\theta = O(1)$, then the algorithm is called a long-step algorithm. In our method, in order to prove convergence result, parameter θ depends on n, therefore the method is a short-step algorithm.

The main emphasis of the thesis is to provide the convergence analysis of the Algorithm described in Table 4.1 and to prove that iteration bound matches the best known iteration bound for these types of methods. This is done in Chapter 5.

However, in Chapter 6 we also provided an initial implementation of the method and tested it on a small set of test problems of various dimensions. We tested our algorithm on several set of parameters. First we set the parameter to be $\tau = \frac{1}{3}$ and $\theta = \frac{1}{\sqrt{2n}}$ as required by the algorithm in order to guarantee convergence. Next, we tried a wider τ-neighborhoods. ($\tau = \frac{1}{2}$, $\tau = 0.9$) and more aggressive reductions of μ-parameter at each iteration, by taking barrier parameter to be a fixed value independent of the size of the problem ($\theta = \frac{1}{\sqrt{6}}$, $\theta = 0.5$, $\theta = 0.9$).

The results we obtained show that the method converges for all test problems even in the case when choice of parameters does not theoretically guarantee convergence. In addition, we observe that increase in parameters τ and specially θ reduces number of inner and outer iterations significantly. Even more importantly, the number of outer iterations increased minimally with the increase in dimension, it is almost constant irrespective of the dimension.This indicates the robustness and stability of the approach.

Thus, even this very preliminary implementation shows not only theoretical but also practical validity of the proposed approach.

APPENDIX A

MATLAB Codes

The following is the listing of the main program. In it input data are generated either manually or randomly. Also a strictly feasible starting point is generated. Then the problem with these input data is solved using subroutine IPM2.m that implements Algorithm.

A.1 Main Program : *prog3.m*

```
%Main Program
% For the Matrices generating by hand(First set of examples)

%M-file prog3.m
%A -- Matrix A
%q -- Matrix q
clear;
%load matrix
n=4;

%define parameters
eps=10^-4;
taw=1/3;

%define matrices
load M.txt
load q.txt

x=ones(n,1);
s=M*x+q;
mu=1/sqrt(2);

v=sqrt(x.*s./mu);

[x s] = IPM4(M,x,s,mu,v)

% For the Matrices generated by using random generater (for the second set of examples)
%Using n, we generate a diagonal matrix of size n, with random entries.
A = rand([n,1]);

%We create a positive definite matrix M from matrix A
M = A'A;
eig(M);
```

A.2 IPM Algorithm : *IPM2.m*

```
function [x,s]= IPM2(M,x,s,mu,v);
n=3;
%x=2*ones(n,1);
%s=2*ones(n,1);
%mu=1/3;
%v=sqrt(x.*s./mu);
x=[1;3;.1];
s=M*x+b;
mu=1/3;
%taw=1/sqrt(2);
taw=1/3;
%outer loop
theta=1/sqrt(6);
while x'*s>=eps

    mu=(1-theta)*mu;
    v=sqrt(x.*s./mu);

    [dx,ds]=SolveSystem2(M,x,s,mu,v);
    Dx = x.*dx./v;
    Ds = s.*ds./v;
    x=x+Dx;
    s=s+Ds;
    v=sqrt(x.*s./mu);
    delta=0.5*norm(v.^-1-v);
    %si=.5*(v.^2-1)-log(v);
    %Si=sum(si,1)

%inner loop
while delta>= taw


    V = diag(v);
    X = diag(x);
    S = diag(s);
    D = X^(.5)*S^(-.5);
    Mtilda = D*M*D;

    %[dx,ds]=SolveSystem(M,x,s,mu,v);
    [dx,ds]=SolveSystem2(M,x,s,mu,v);
    Dx = x.*dx./v;
    Ds = s.*ds./v;
    x=x+Dx;
    s=s+Ds ;
    v=sqrt(x.*s./mu);
    delta=0.5*norm(v.^-1-v);
   %si=.5*(v.^2-1)-log(v);
   %Si=sum(si,1);
```

```
end
%x=x';
%s=s';

end
end
```

A.3 Newton System Solver : *SolveSystem2.m*

This subroutine solves the linear system at each iteration of IPM2.m

```
function[dx,ds]=SolveSystem2(M,x,s,~,v)
% This is the function solving system

% Mtilda*dx-ds = 0
% dx+ds = 0
n=3;
X=diag(x);
S=diag(s);
    D = X^(.5)*S^(-.5);
    Mtilda = D*M*D;
%Mtilda=D*M*D;

dx=(eye(n)+ Mtilda)\(v.^-1 - v);
ds=Mtilda*dx;
end
```

APPENDIX B

MATLAB Output

Below we provide the entire output for Example 3.3.

B.1 *Output of EH1*

```
 % Mathlab out put of the example describe in both Lemke's method and IPM
>> count1 =

   71


count2 =

    2


x =

    0.0000
    2.0000
    1.0000


s =

    1.0000
    0.0000
    0.0000

Elapsed time is 0.019958 seconds.
```

BIBLIOGRAPHY

[1] Bai Y., Roos C., *A New Class of Polynomial Interior-Point Algorithms for Linear Complementarity Problems*, Pacific Journal of Optimization, Vol. 4, No. 1, (2008).

[2] Bai Y., Lesaja G., Roos C., Wang G., Ghami M. El, *A Class of Large and Small Update Primal-Dual Interior-Point Algorithms for Linear Optimization*, Journal of Optimization Theory and Applications, Vol. 138, No. 3, pp. 341-359, (2008).

[3] Cottle R., Pang J., Stone R., *The Linear Complementarity Problem*, Academic Press, Inc., Boston, (1992).

[4] Cottle and Dantzig (1968), *Complementary pivot theory of mathematical programming*, Linear Algebra and Its Applications 1.

[5] Facchinei F., Pang J.S., *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Springer, New York, (2003)

[6] Karmarkar N., *A New Polynomial Time Algorithm for Linear Programming*, Combinatorica, Vol 4, No. 4, (1984).

[7] Khachiyan L.G., *A Polynomial Algorithm in Linear Programming*, Soviet Mathematics Doklady, 20, pp. 373-395 (1984).

[8] Klee V. and Minty G.J., *How Good is the Simplex Algorithm?* Inequalities, III, pp. 159-175, Academic Press, New York, NY, (1972).

[9] Kojima M., Megiddo N., Noma T., Yoshise A., *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*, Springer-Verlag, Berlin, Germany (1991).

[10] Lemke C.E., *Bimatrix Equilibrium Points and Mathematical Programming*, Management Science II, pp. 681-689, (1965).

[11] Lesaja G., *Introducing Interior-Point Methods for Introductory Operations Research Courses and/or Linear Programming Courses*, Open Operational Research Journal, Vol. 3, pp. 1-12, (2009).

[12] G. Lesaja and C. Roos. Unified analysis of kernel-based interior-point methods for $P_*(\kappa)$-LCP. *SIAM Journal on Optimization,* 20(6): 3014-3039, 2010.

[13] Lesaja G., Mansouri H., Roos C.,Zangiabadi M., *Full-Newton-Step Interior-point Methods for Linear Optimization Based on Locally Self-Concordant Barrier Functions*, Mathematical Programming, accepted, (2010).

[14] Mansouri H., *Full-Newton-Step Interior-point Methods for conic Optimization*, Ph.D. Thesis, TU Delft, Netherland, (2008).

[15] Nesterov Y., Nemirovski A., *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Studies in Applied Mathematics, Philadelphia, PA, (1994).

[16] Megiddo N., *Pathways to the Optimal Set in Linear Programming*, Progress in Mathematical Programming: Interior Point and Related Methods, pp. 131-158, Springer, New York, (1989).

[17] Peng J.S., Roos C., Terlak T., *Self-Regularity: A New Paradigm for Primal-Dual Interior-Point Algorithms*, Princeton University Press, (2002)

[18] Roos C., *A Full-Newton Step $O(n)$ Infeasible Interior-Point Algorithm for Linear Optimization*, *SIAM Journal on Optimization,* 16(6): 1110-1136, 2006.

[19] Roos C., Terlaky T., Vial J.P., *Theory and Algorithms for Linear Optimization*, John Wiley and Sons, Chichester, UK (1997).

[20] Shor N.Z., *Cut-off Method with Space Extension in Convex Programming Problems*, Cybernetics 13, pp. 94-96, (1977).

[21] Sonnevend G., *An "analytic center" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming*, System Modeling and Optimization. Proceedings of the 12th IFIP-Conference, Budapest, Hungary, September 1985. Lecture Notes in Control and Information Sciences, vol. 84, pp. 866-876. Springer, Berlin (1986).

[22] Wright S., *Primal-Dual Interior-Point Methods*, SIAM Publishing, Philadelphia, PA (1997).

[23] Wright S., Ferris M., Mangasarian O., *Linear Programming with MATLAB*, SIAM, Series on Optimization, (2007).

[24] Nemirovski A., Yudin D.B., *Informational Complexity and Effective Methods of Solution for Convex Extremal Problems*, Ekonomika i Matematicheshie Metody 12 (in Russian), pp. 357-369, (1976).