Abstract

CHEN, SHYH-HUEI. Solving Systems of Linear Inequalities. (Under the direction of Dr. Shu-Cherng Fang and Dr. Henry L. W. Nuttle.)

The problem of finding a feasible solution to a system of linear inequalities arises in numerous contexts. In this dissertation, we consider solving a system of linear inequalities in view of unconstrained convex programming problems. Solution methods for solving systems with either finitely or infinitely many linear inequalities are proposed. Convergence properties and implementation issues are discussed. Some computational results are also included.
SOLVING SYSTEMS OF LINEAR INEQUALITIES

by

SHYH-HUEI CHEN

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

OPERATIONS RESEARCH

Raleigh
2001

APPROVED BY:

Dr. Elmor L. Peterson

Dr. Shu-Cherng Fang
Chair of Advisory Committee

Dr. Hien T. Tran

Dr. Henry L. W. Nutter
Co-Chair of Advisory Committee
Dedicated to those I love, in particular,

my parents Chao-Hsiung Chen and Mei-Chih Chang,

my brother Shih-Ming and sister Yen-Ling,

and my beloved Fang-Chi Hsu.
Biography

Shyh-Huei Chen was born on December 14, 1968 in Taipei, Taiwan, R.O.C. He received a B.S. (1991) degree and an M.S. (1993) degree both in Applied Mathematics from National Chiao-Tung University (Taiwan) and National Chung-Cheng University (Taiwan) respectively. Then he served as a Second Lieutenant in the ROC army for two years. He received a Distinguished Service Award from the Army of Taiwan in 1995. Soon afterwards, he was employed as a research assistant in Institute of Statistical Science at Academia Sinica (Taiwan). In the fall of 1996, he came to the United States and started his Ph.D. study in the graduate program of Operations Research at North Carolina State University. While working on his Ph.D., he was employed as a teaching assistant (1997, 1999 and 2000) and he was appointed as a research assistant (1997-2000). His research interests include optimization theory, soft computing and network flows analysis. He is a member in the Omega Rho Honor Society.
Acknowledgments

I would like to express my sincere appreciation to my advisor, Dr. Shu-Cherng Fang, for his guidance, responsiveness, support and encouragement throughout my Ph.D. study. I thank Dr. Henry L. W. Nuttle for his support and valuable contributions as a co-advisor. Especially, I thank him for his extraordinary and enlightening teaching. I would also like to thank my advisory committee members: Dr. Elmor L. Peterson and Dr. Hien T. Tran, for their time and helpful suggestions.

I thank the students at the Fuzzy and Neural Group at North Carolina State University for their supports and friendships. Special thank to my “big brothers” Nanchieh Chiu and Ta-Wei Hung, for making my life easier. I thank Andrés Medaglia, who has been a good friend since the first day I joined the group, and Yi Laio, who cheerfully imparted his knowledge of the computer programming.

I thank my parents for their sacrifice, enthusiasm and incessant support throughout my study. Finally, I would like to thank my beloved, Fang-Chi Hsu, for her love, care, understanding, and enduring support.
# Contents

List of Tables vii

List of Figures ix

1 **Introduction** 1
   1.1 Systems of Finitely Many Linear Inequalities . . . . . . . . . . . . . . . . . . . . 1
   1.2 Systems of Infinitely Many Linear Inequalities . . . . . . . . . . . . . . . . . . . 3
   1.3 Organization of Dissertation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2 **Literature Review** 5
   2.1 Systems of Finitely Many Linear Inequalities . . . . . . . . . . . . . . . . . . . . 5
      2.1.1 Elimination Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
      2.1.2 Relaxation Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
         2.1.2.1 Projection Methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
         2.1.2.2 Cimmino-type Methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
      2.1.3 Optimization Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
         2.1.3.1 Unconstrained Formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
         2.1.3.2 Constrained Formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
   2.2 Systems of Infinitely Many Linear Inequalities . . . . . . . . . . . . . . . . . . . 12
      2.2.1 Relaxation Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
      2.2.2 Discretization Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

3 **Systems of Finitely Many Linear Inequalities** 16
   3.1 Entropic Perturbation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
      3.1.1 Properties of $g_{(p)}(w)$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
   3.2 Proposed Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
   3.3 Numerical Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
      3.3.1 Implementation Issues . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
      3.3.2 Computational Experience . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
| 3.3.2.1 | Effect of the Problem Size \((n, m, m/n)\) | 26 |
| 3.3.2.2 | Effect of the Distance between the Initial Solution and the Feasible Region | 28 |
| 3.4 | Remarks | 29 |

4 Systems of Infinitely Many Linear Inequalities | 39 |
| 4.1 | Cutting Plane Method Based on Centers | 40 |
| 4.1.1 | Cutting Plane Method | 40 |
| 4.1.2 | Analytic Center | 41 |
| 4.1.3 | Analytic Center Cutting Plane Method | 42 |
| 4.2 | Algorithms for Solving SLIS | 44 |
| 4.2.1 | ACCPM for Solving SLIS | 44 |
| 4.2.2 | EACCPM for Solving SLIS | 47 |
| 4.2.3 | Numerical Examples | 48 |
| 4.3 | Maximum Feasibility Problem | 52 |
| 4.3.1 | Proposed Approach for MFP | 53 |
| 4.3.2 | Numerical Examples | 57 |
| 4.4 | Remarks | 61 |

5 Entropic Regularization Method for Solving Linear Inequalities | 62 |
| 5.1 | Entropic Regularization Method | 63 |
| 5.1.1 | Finite Min-Max Problems | 64 |
| 5.1.2 | Infinite Min-Max Problems | 66 |
| 5.2 | Proposed Algorithm | 67 |
| 5.2.1 | Relation Between the Proposed Method and Other Methods | 70 |
| 5.3 | Numerical Examples | 71 |
| 5.3.1 | Finite Systems | 72 |
| 5.3.2 | Infinite Systems | 88 |
| 5.4 | Remarks | 92 |

6 Extensions and Related Applications | 94 |
| 6.1 | System of Linear Equations and Inequalities | 94 |
| 6.2 | ACCPM for Variational Inequality Problems | 95 |
| 6.2.1 | Entropic ACCPM for VI\((F, \Omega)\) and Examples | 97 |
| 6.3 | Remarks | 99 |

7 Summary and Future Research Directions | 106 |
| 7.1 | Summary of Dissertation | 106 |
| 7.2 | Future Research Directions | 108 |
List of Tables

3.1 small-size problems with under-determined systems ($\gamma = 1.7$) .... 30
3.2 small-size problems with well-determined systems ($\gamma = 1.7$) .... 30
3.3 small-size problems with over-determined systems ($\gamma = 1.7$) .... 30
3.4 large-size problems with under-determined systems ($\gamma = 1.7$) .... 31
3.5 large-size problems with well-determined systems ($\gamma = 1.7$) .... 31
3.6 large-size problems with over-determined systems ($\gamma = 1.7$) .... 31
3.7 small-size problems with under-determined systems ($\gamma = 1$) .... 32
3.8 small-size problems with well-determined systems ($\gamma = 1$) .... 32
3.9 small-size problems with over-determined systems ($\gamma = 1$) .... 32
3.10 large-size problems with under-determined systems ($\gamma = 1$) .... 33
3.11 large-size problems with well-determined systems ($\gamma = 1$) .... 33
3.12 large-size problems with over-determined systems ($\gamma = 1$) .... 33
3.13 EP method results with distance shift .................................. 34
3.14 SC method results with distance shift .................................. 34
3.15 SC method results with distance shift (continued) ................. 35
3.16 SC method results with distance shift (continued) ................. 35
4.1 ACCPM results for example 1 ............................................ 50
4.2 EACCPM results for example 1 ........................................... 50
4.3 ACCPM results for example 2 ............................................ 51
4.4 EACCPM results for example 2 ........................................... 51
4.5 results of MFP example 1 .................................................. 58
4.6 results of MFP example 2 .................................................. 60
5.1 steepest descent method with strategy (1) ........................... 76
5.2 conjugate gradient method with strategy (1) ......................... 77
5.3 modified Newton method with strategy (1) ........................ 78
5.4 quasi-Newton method with strategy (1) .............................. 79
5.5 steepest descent method with strategy (2) ............................. 80
5.6 conjugate gradient method with strategy (2) ......................... 81
5.7 modified Newton method with strategy (2) ........................ 82
5.8 quasi-Newton method with strategy (2) ............................... 83
5.9  steepest descent method with strategy (3)  .................  84
5.10  conjugate gradient method with strategy (3) .................  85
5.11  modified Newton method with strategy (3) ..................  86
5.12  quasi-Newton method with strategy (3) ......................  87
5.13  2-dimenssional with $r = 0.01$  ..........................  89
5.14  2-dimenssional with $r = 0.001$  ........................  89
5.15  3-dimenssional with $r = 0.01$  ..........................  89
5.16  3-dimenssional with $r = 0.001$  ........................  89
5.17  results for example 2 with $r = 0.05$ .......................  91
5.18  results for example 2 with $r = 0.01$ .......................  91
5.19  results for example 2 with $r = 0.001$ .......................  91
5.20  results for example 2 with $r = 0.001$ .......................  92

6.1  numerical results for ACCMP and EACCMP for $F_1$ in Example 1  100
6.2  numerical results for ACCMP and EACCMP for $F_2$ in Example 1  101
6.3  numerical results of ACs for $F$ in Example 2 .................  102
6.4  numerical results of EACs for $F$ in Example 2 ..............  102
List of Figures

3.1 total required iterations for under-determined systems \((n/m = 2)\) \ldots \quad 36
3.2 required CPU time for under-determined systems \((n/m = 2)\) \ldots \quad 36
3.3 total required iterations for well-determined systems \((n/m = 1)\) \ldots \quad 37
3.4 required CPU time for well-determined systems \((n/m = 1)\) \ldots \quad 37
3.5 total required iterations for over-determined systems \((n/m = 1/2)\) \ldots \quad 38
3.6 required CPU time for over-determined systems \((n/m = 1/2)\) \ldots \quad 38

5.1 blank area is the feasible region of example 2 with \(r = 0.05\) \ldots \quad 90

6.1 feasible set, ACs, and cuts for \(F_1(x)\) in Example 1 \ldots \quad 103
6.2 feasible set, entropic ACs, and cuts for \(F_1(x)\) in Example 1 \ldots \quad 103
6.3 feasible set, ACs, and cuts for \(F_2(x)\) in Example 1 \ldots \quad 104
6.4 feasible set, entropic ACs, and cuts for \(F_2(x)\) in Example 1 \ldots \quad 104
6.5 \(\|x^k - x^*\|\) vs \(k\) for ACCPM and EACCPM in Example 2 \ldots \quad 105
6.6 \(\|x^k - x^*\|\) vs \(k\) for ACCPM and EACCPM in Example 2 with binding constraint removed \ldots \quad 105
Chapter 1

Introduction

Consider the following system of linear inequalities:

$$
\sum_{i=1}^{n} a_i(t)x_i \leq b(t), \quad \forall t \in T
$$

(1.1)

where $x \in R^n$, $T$ is a compact set in $R^l$ with $l \in N$, and $a_1(t), a_2(t), \ldots, a_n(t)$ and $b(t)$, are real-valued functions on $T$. When $T$ has finite cardinality, say $|T| = m_1$, (1.1) becomes a system of finitely many linear inequalities:

$$
A_1x \leq b_1,
$$

(1.2)

where $A_1$ is an $m_1 \times n$ matrix and $b_1$ is an $m_1$-dimensional vector. If $|T| = \infty$, (1.1) becomes a system of infinitely many linear inequalities. In either case, our objective is to find a point $x \in R^n$ that satisfies (1.1).

1.1 Systems of Finitely Many Linear Inequalities

The problem of finding a feasible solution to system (1.2) arises in numerous contexts. It is omnipresent in optimization problems, such as the “Phase I” problem of the simplex method, or a preliminary step for the interior point methods [15]. Applications can be found in image reconstruction for computerized tomography [34] and pattern recognition [67].

For many years, a significant level of effort has been devoted to develop efficient and reliable numerical methods for solving (1.2). Basically, there are three approaches for solving system (1.2):
1. Elimination approach ([20][12]).

2. Relaxation approach ([52][1][5][8]).

3. Optimization approach ([11][36][67][31][42][40]).

The elimination approach introduced by Fourier could be the earliest method for solving linear inequalities [20]. However, this approach is not practical for computer implementation because of its complexity. The relaxation approach, also known as the projection or Cimmino-type method, can be easily implemented, but it exhibits slow convergence in practice. The optimization approach is perhaps the most commonly used one; however it may be computationally expensive when the matrix inversion is involved.

Recently, several new algorithms have been proposed for solving systems of linear inequalities. For the relaxation approach, Yang and Murty [69] proposed a new projection method called the “surrogate constraint method”. They reported some encouraging preliminary computational results. A variant of the surrogate method was proposed by Oko [54].

For the optimization approach, methods of finding solutions in a least-squares sense can be found in [21][33][4]. Moreover, by considering (1.2) as the dual problem of a standard form linear program, an “analytic center” method can be applied to find exact solutions [13][30][25]. Chen and Mangasarian [7] used a smooth approximation technique to convert linear inequalities into an unconstrained optimization problem. Pinar [58] and Pinar and Chen [59] provided a Newton-like algorithm based on a different approximation scheme. In addition, artificial neural networks have been applied to solve systems of linear inequalities [44].

In many practical applications, the decision variables $x_i$, for $i = 1, ..., n$, are restricted to be nonnegative. This leads us to consider the following system of linear inequalities:

$$
\begin{cases}
Ax & \leq b \\
x & \geq 0,
\end{cases}
\tag{1.3}
$$

where $A$ is an $m \times n$ matrix and $b$ is an $m$-dimensional vector. Notice that (1.3) can be converted to system (1.2) by letting $A_1 = \frac{A}{I_n}$ and $b_1 = \frac{b}{\alpha_1}$, where $I_n$ denotes the $n \times n$ identity matrix and $0_n$ the $n$-dimensional zero vector. However, this conversion increases the problem size.
In our study, we focus on the system (1.3) and extend the entropic perturbation method proposed by Fang [14] and Tsao and Fang [17] to find solutions. The proposed method can satisfy the nonnegativity constraints automatically without increasing the problem size.

1.2 Systems of Infinitely Many Linear Inequalities

The study of systems of infinitely many linear inequalities is a natural extension of many classical mathematical problems, including the functional approximation, matrix theory, game theory, optimal control and mathematical programming problems [35][23]. In particular, the first step of solving a linear semi-infinite programming problem usually requires a feasible solution to such a system. This problem also appears in a continuous model of image reconstruction for computerized tomography [34] and the inverse problem in radiation therapy treatment planning [9].

The study of the solution of systems of infinitely many linear inequalities started in 1920s [23]. However, no numerical method is reported until Jeroslav proposed a projection method in 1979 [39]. As the semi-infinite linear programming problems have become an exciting part of mathematical programming, the search for a feasible solution to (1.1) has attracted increasing interest [35][23].

As in the case of finite systems, in practice, for problems including systems like (1.2), \( x \) is often required to be nonnegative. This leads us to consider the following system:

\[
\begin{\cases}
    a(t)^T x \leq b(t), & \forall t \in T \\
    x \geq 0,
\end{cases}
\tag{1.4}
\]

where \( a : T \to \mathbb{R}^n \) and \( b : T \to \mathbb{R} \) are continuous function on \( T \).

Recently, a cutting plane method based on the concept of the “analytic center” was proposed for use in convex feasibility problems [24] [25]. In our research, we will further extend the idea of analytic center cutting plane method to solve system (1.4).

1.3 Organization of Dissertation

The remainder of this dissertation is organized as follow: In the next chapter, we review some existing methods for solving system (1.1). In Chapter 3, a new unconstrained convex optimization approach is proposed for solving systems of finitely
many linear inequalities, i.e., system (1.3). An entropic perturbation algorithm based on the Newton method together with some numerical results are presented in this chapter. Chapter 4 uses the analytic center cutting plane method (ACCPM) to solve systems of infinitely many linear inequalities, i.e., system (1.4). A new analytic center, called the entropic analytic center, is defined and used in the role of the analytic center in ACCPM. This chapter also treats the maximum feasibility issue of a system of infinitely many linear inequalities. In Chapter 5, we introduce the entropic regularization method for solving general min-max problems. A unified, flexible approach based on the entropic regularization method is proposed for solving systems involving both finitely and infinitely many linear inequalities like (1.1). In Chapter 6, the algorithms proposed for solving system of linear inequalities in Chapter 3 are extended to solve systems of linear equations and linear inequalities. The use of the entropic analytic center for solving variational inequality problems is also examined in this chapter. A summary and future research directions are given in Chapter 7.
Chapter 2

Literature Review

In this chapter, we give a concise review of the existing numerical methods for solving systems of linear inequalities.

2.1 Systems of Finitely Many Linear Inequalities

In this section, we review three approaches, namely, the elimination, relaxation, and optimization approaches, for solving the following system of linear inequalities:

\[ Ax \leq b, \quad (2.1) \]

where \( A \) is an \( m \times n \) matrix and \( b \) is an \( m \)-dimensional vector. Throughout this section, \( A_j \) denotes the \( j \)th row vector of matrix \( A \); \( \| \cdot \|_p \) the \( l_p \) norm; and \( (a)_+ \triangleq \max \{ a, 0 \} \), \( \forall a \in R \).

2.1.1 Elimination Approach

The classical variable elimination technique for solving linear inequalities was introduced by Fourier in 1826 [20]. It uses elementary row operations \( R \geq 0 \) to convert \( A \) into \( RA = 0 \) and \( b \) into \( \tilde{b} = Rb \). It can be shown that \( Ax \leq b \) has a solution if and only if \( \tilde{b} \geq 0 \). In this case, a feasible solution of \( Ax \leq b \) can be constructed by backtracking through the operations represented by the non-negative matrix \( R \). The following details the process of the elimination method.

Suppose \( x_1 \) is a variable that we wish to eliminate from the system (2.1). For \( x_1 \),
denote three index sets

\[
I^+ = \{ j \mid \text{sign}(a_{j1}) = +1, 1 \leq j \leq m \}, \quad (2.2)
\]

\[
I^- = \{ j \mid \text{sign}(a_{j1}) = -1, 1 \leq j \leq m \}, \quad (2.3)
\]

and

\[
I^0 = \{ j \mid a_{j1} = 0, 1 \leq j \leq m \}. \quad (2.4)
\]

If \( I^+ \) is empty then delete all the inequalities with indices in \( I^- \), since they can be satisfied by choosing sufficiently large \( x_1 \). Similarly, if \( I^- \) is empty, all the inequalities in \( I^+ \) can be discarded. If \( I^+ \neq \emptyset \) and \( I^- \neq \emptyset \), then, for each \( i \in I^+ \) and \( l \in I^- \), we create a new inequality by adding the following two inequalities together:

\[
(-a_{il1}) \times (A_i x) \leq (-a_{il}) \times b_l
\]

and

\[
a_{i1} \times A_i x \leq a_{i1} \times b_l. \quad (2.6)
\]

Notice that in these new inequalities, the coefficients of \( x_1 \) are wiped out. These new inequalities and the inequalities with indices belonging to \( I^0 \) form a new system of inequalities \( \tilde{A}(i)x \leq \tilde{b}(i) \).

We repeat this construction process to eliminate \( x_2 \) from \( \tilde{A}(1)x \leq \tilde{b}(1) \), \( x_3 \) from \( \tilde{A}(2)x \leq \tilde{b}(2) \), ..., and so on, until we have inequalities \( \tilde{A}(n-1)x \leq \tilde{b}(n-1) \) involving a single variable \( x_n \). Then we can solve these inequalities to obtain a range of the value for \( x_n \). If the range is empty there is no solution for \( Ax \leq b \). If the range of \( x_n \) is not empty, we pick a value \( x^*_n \) for \( x_n \) and backtrack to previous level to solve \( x_{n-1} \) by substituting \( x^*_n \) in \( \tilde{A}(n-2)x \leq \tilde{b}(n-2) \). Repeating the backtracking process, eventually, we construct a solution \( x^* = (x^*_1, ..., x^*_n)^T \) for the system (2.1).

The required operations for Fourier elimination are simple. However, the construction process quickly runs into an exponential complexity. For example, if half of the coefficients of \( x_1 \) appear with opposite sign, then the number of inequalities will grow from \( m \) to \( (m/2)^2 \) for the next step. This makes the elimination approach impractical for computer implementation. A recent survey on this method can be found in Chandru [6].
2.1.2 Relaxation Approach

The name “relaxation” refers to the fact that constraints are considered one at a time (all constraints except one are relaxed). The original research on the relaxation approach was directed at solving a system of equations. It was later extended for solving a system of inequalities [1].

This approach includes the so called “projection” and “Cimmino-type” methods. The major difference is that a projection method makes an orthogonal projection of an infeasible point onto a single hyperplane (defined by a violated constraint or a surrogate constraint), while a Cimmino-type method projects the point onto every violated constraints simultaneously.

One advantage of the relaxation method include is that it allows the solution set $S$ to be unbounded and contain no interior point. However, the relaxation method has the drawback of not being able to detect inconsistency in a system of inequalities. Hence, the results may become unpredictable when the method is applied to an inconsistent system.

2.1.2.1 Projection Methods

A projection method satisfies one inequality at each iteration by projecting orthogonally the current solution to the most violated inequality. The iteration proceeds according to:

$$x^0 \in R^n \text{ is arbitrary,}$$

$$x^{k+1} = x^k + \gamma (P_{H_i}(x^k) - x^k),$$  \hspace{1cm} (2.7)

where $\gamma \in (0, 2]$ and $P_{H_i}(x^k)$ is the orthogonal projection of $x^k$ onto the hyperplane $H_i \triangleq \{ x | A_i x - b_i = 0 \}$ ($i$ is the index of the most violated inequality). For $\gamma = 1$, the procedure is called the Method of Orthogonal Projection [1]. For $\gamma = 2$, it is called the Reflection Method [52]. It can also be modified to handle infinitely many linear constraints [39].

Projection methods are simple and easy to program, but they converge slowly. Moreover, they may only be able to return an $\varepsilon$-feasible solution ($Ax - b \leq \varepsilon$, where $\varepsilon$ is a small positive number) even when the system is consistent. This means that the methods are not guaranteed to converge to a feasible solution in a finite number of iterations.
Recently, one extension called the Surrogate Constraint Method was proposed by Yang and Murty [69]. A current solution is projected onto a surrogate hyperplane $H$ which is a linear combination of all violated constraints:

$$H = \{ x | \pi^T A_I x = \pi^T b_I \}, \quad (2.8)$$

where $I = \{ i | A_i x - b_i > 0, 1 \leq i \leq m \}$, $A_I$ is the submatrix of $A$ with the row vector $A_i$, $i \in I$, and $\pi = (\pi_i), i \in I$, is a pre-selected weighted vector with $\pi_i \geq 0$ and $\sum_{i \in I} \pi_i = 1$. The $k + 1$st step of the surrogate constraint method is defined as

$$x^{k+1} = x^k + \gamma (P_H(x^k) - x^k)$$

$$= x^k - \gamma (A_I^k x^k - b_I^k)^T \pi^k \frac{A_I^k \pi^k}{\|A_I^k \pi^k\|^2}.$$ \quad (2.9)

The computational results for this surrogate constraint method are quit encouraging and it can return an $\epsilon$-feasible solution for any specified $\epsilon > 0$ within finitely many iterations. However, there is no result on the rate of convergence. Another surrogate method for solving linear inequalities can be found in Oko [54].

### 2.1.2.2 Cimmino-type Methods

Cimmino [10] proposed an elegant iterative method for finding a solution to a system of linear equations. The method starts with an arbitrary point $x^0 \in \mathbb{R}^n$, and then calculates the centroid $x^1$ of a system of masses placed at the reflections of $x^0$ with respect to the hyperplane defined by the system of equations. The sequence of iterations can be formulated as

$$x^0 \text{ is arbitrary,}$$

$$x^{k+1} = \sum_{i=1}^{m} \lambda_i \left[ 2P_{H_i}(x^k) - x^k \right],$$ \quad (2.10)

where $\lambda_i > 0$, for $i = 1, \ldots, m$, are pre-selected masses with $\sum_{i=1}^{m} \lambda_i = 1$, and $H_i = \{ x | A_i x - b_i = 0 \}$, for $i = 1, \ldots, m$.

Censor and Elfving [8] generalized Cimmino’s method to solve linear inequalities. They considered only the reflections with respect to the violated constraints for the current solution. The $k + 1$st step of the Cimmino-type method for solving linear inequalities becomes

$$x^{k+1} = x^k + 2 \sum_{i \in I^k} \lambda_i x_i^k A_i^T,$$ \quad (2.11)
where $I^k = \{j | A_j x^k - b_j > 0, 1 \leq i \leq m\}$, and

$$c_i^k = \min \left\{ 0, \frac{b_i - A_i x^k}{\|A_i^T\|^2} \right\}, \text{ for } i = 1, ..., m. \tag{2.12}$$

Motivated by [8], Pierro and Iusem [57] considered a simultaneous projection algorithm:

- $x_0$ is arbitrary,
- $x^{k+1} = x^k + \gamma \sum_{i=1}^{m} \lambda_i P_i(x^k) - x^k$, \tag{2.13}

where $P_i(x^k) = x^k + c_i^k A_i^T$, for $i = 1, ..., m$, and $\gamma \in (0, 2)$.

Since the Cimmino-type methods make projections onto every violated constraint simultaneously, they are computationally expensive. Similar to the projection methods, Common-type methods also have no proven rate of convergence.

### 2.1.3 Optimization Approach

Any system of linear inequalities can be converted to a linear programming problem with zero objective function and solved directly by linear programming algorithms [11]. In this section, we review nonlinear optimization methods for solving (2.1).

#### 2.1.3.1 Unconstrained Formulation

Consider the following nonlinear unconstrained problem:

$$\min f(x) \equiv \frac{1}{2} \|s(x)_+\|_p^2, \tag{2.14}$$

where $s(x) = Ax - b$, and $p = 1$ or 2. Obviously, $x^*$ is a feasible solution of (2.1) if and only if $f(x^*) = 0$. Moreover, if system (2.1) is consistent, a minimizer of (2.14) is also a feasible solution of (2.1). When $p = 2$, the minimizer of (2.14) (regardless of whether it is feasible) is said to be a “least-squares solution” of (2.1).

Note that $f(x)$ is a convex, continuously differentiable, and piecewise quadratic function. The following properties of $f(x)$, with $p = 2$, are provided by Han [31]:

1. $x^*$ solves (2.1) if and only if $A^T s(x^*_+ = 0$.
2. $\nabla f(x) = A^T s(x)_+$.
Based on these two properties, Han [31] designed a steepest-descent type algorithm for solving system (2.14). A more efficient version of Han’s algorithm was given by Bramley and Winnicka [4]. The algorithm was shown to be globally convergent. However there was no proof of rate of convergence. Bingsheng He [33] proposed a globally convergent iterative algorithm with

\[
x^{k+1} = x^k - \gamma \frac{\|s^k(x^k)\|^2}{\|\nabla f(x^k)\|^2_2} \nabla f(x^k),
\]

where \(\gamma \in (0, 2)\). This algorithm has a linear rate of convergence.

Another approach for solving (2.14) is to approximate \(s(x)_+\) by using a smooth function. Then solve it by any unconstrained optimization method. Chen and Mangasarian [7] provided an approximation function:

\[
g(x, \alpha) \equiv \int_{-\infty}^{x} \frac{1}{1 + \exp(-\alpha x)} \, dx = x + \frac{1}{\alpha} \log(1 + \exp(-\alpha x)),
\]

where \(\alpha > 0\). Some useful properties of \(g(x, \alpha)\) are given below:

1. For any \(\alpha > 0\), \(g(x, \alpha)\) is strictly convex and strictly increasing with respect to \(x \in \mathbb{R}\).

2. For \(k \in \mathbb{N}\), \(g(x, \alpha)\) is \(k\)-times continuously differentiable function with respect to \(x\), with \(g'(x, \alpha) = 1/(1 + \exp(-\alpha x))\) and \(g''(x, \alpha) = \alpha \exp(-\alpha x)/(1 + \exp(-\alpha x))^2\).

3. For any \(\alpha > 0\), \(g(x, \alpha) > x_+, \forall x \in \mathbb{R}\).

4. \(\lim_{\alpha \to \infty} g(x, \alpha) = x_+, \forall x \in \mathbb{R}\).

5. \(\max_{x \in \mathbb{R}} \{g(x, \alpha) - x_+\} = g(0, \alpha) = \log 2/\alpha, \forall \alpha > 0\).

Therefore, by choosing \(\alpha\) to be sufficiently large, \(s(x)_+\) can be approximated by \(g(Ax - b, \alpha)\) to any desired accuracy. Under the interior point assumption (i.e., there exists an \(x_0 \in \mathbb{R}^n\), such that \(Ax_0 < 0\)), using a large enough \(\alpha\) and replacing \(f(x)\) with \(f_1(x) = \frac{1}{2} \|g(Ax - b, \alpha)\|_1\) or \(f_2(x) = \frac{1}{2} \|g(Ax - b, \alpha)\|_2^2\), an exact solution of (2.14) can be attained by minimizing \(f_1(x)\) or \(f_2(x)\). Notice that \(f_1(x)\) and \(f_2(x)\) are smooth functions. Thus existing first or second order algorithms for unconstrained
minimization can be applied to get a linear, a super-linear or a local quadratic rate of convergence, depending on the algorithm. This approach can also be modified to solve convex inequalities.

Motivated by this smoothing method, Pinar [58] and Pinar and Chen [59] used the Huber function to approximate \( s(x)_+ \). The Huber function with smoothing parameter \( \beta > 0 \) is given by

\[
h(z, \beta) = \begin{cases} 
0 & \text{if } z \leq 0 \\
\frac{1}{2\beta} z^2 & \text{if } 0 < z < \beta \\
z - \frac{\beta}{2} & \text{if } z \geq \beta.
\end{cases}
\] (2.17)

Clearly, \( h(z, \beta) \) is once differentiable with respect to \( z \) and it approaches \( z_+ \) as \( \beta \) approaches 0. Thus a Newton-type algorithm can handle both consistent and inconsistent cases of (2.1) and terminate in a finite number of iterations. However, the authors presented no result on the rate of convergence.

### 2.1.3.2 Constrained Formulation

Consider the following quadratic programming problem:

\[
\begin{aligned}
\min & \quad \frac{1}{2} \|x\|_2^2 \\
\text{s.t.} & \quad Ax \leq b.
\end{aligned}
\] (2.18)

Obviously, any solution of (2.18) is a solution of (2.1). We can apply the Lagrange method [46] or the SOR (successive overrelaxation) method [51] to solve this quadratic programming problem. However, some serious numerical difficulty may occur while solving a quadratic programming problem with a singular objective function. Furthermore, most of the methods require an ingenious active-set strategy which is difficult to implement. Because of these difficulties, methods have been designed to solve the dual problem of (2.18).

Let \( w \in R^m \) denote the dual variables. The K-K-T conditions for (2.18) are

\[
\begin{aligned}
x + A^T w &= 0 \\
Ax &\leq b \\
w^T (Ax - b) &= 0 \\
w &\geq 0
\end{aligned}
\] (2.19)

Conditions (2.19) are also the K-K-T conditions of the dual problem of (2.18):

\[
\begin{aligned}
\min & \quad \frac{1}{2} w^T A A^T w + b^T w \\
\text{s.t.} & \quad w \geq 0.
\end{aligned}
\] (2.20)
Fukushima [21] has designed a conjugate gradient algorithm to solve (2.20). It can detect whether the problem in (2.18) is consistent and provide a least-squares solution in a finite number of iterations.

Another constrained nonlinear optimization formulation for (2.1) is

\[
\begin{align*}
\max & \quad \sum_{j=1}^{m} \ln(b_j - A_j^T x) \\
\text{s.t.} & \quad Ax \leq b.
\end{align*}
\]

Under the interior point assumption, we say that \( x^* \) is the “analytic center” (AC) of a convex polyhedron set \( \Omega = \{x \in R^n | Ax \leq b\} \) if the maximizer of system (2.21) exists and is equal to \( x^* \). Since the objective function of (2.21) is strictly convex, \( x^* \) is uniquely defined by the following optimality conditions:

\[
\begin{align*}
Ax + s & = b \\
A^T w & = 0 \\
Ws & = e \\
w & \geq 0, \ s \geq 0,
\end{align*}
\]

where \( w \in R^m \) are the dual variables, \( W = \text{diag}(w) \), \( s \in R^m \) are the slack variables and \( e = (1, 1, ..., 1)^T \). Several different methods designed to calculate the analytic center of \( \Omega \) by solving (2.22) can be found in [13][70][30][72]. Notice that the definition for the analytic center of \( \Omega \) requires that \( \Omega \) be bounded. Because of this restriction, formulation (2.21) is not popular.

### 2.2 Systems of Infinitely Many Linear Inequalities

In this section, we review approaches for solving the following system of linear inequalities:

\[
a(t)^Tx \leq b(t), \ \forall \ t \in T,
\]

where \( x \in R^n, T \) is a closed subset of \( R^l \), \( l \in N \), and \( a : T \rightarrow R^n \) and \( b : T \rightarrow R \). When \( |T| \) is finite, system (2.23) becomes a system of finitely many linear inequalities like (2.1). In this section, we focus only on the case when \( |T| \) is infinite.

There are only two numerical methods, both based on the relaxation method, that have been proposed for solving system (2.23) directly [39] [38]. Some projection methods for solving convex feasibility problems can be recast to solve (2.23). An
extensive survey on solving convex feasibility problem using the projection method can be found in [3].

Notice that system (2.23) can be converted to a linear semi-infinite program and numerical methods for the solution of the semi-infinite programming problem can be used to identified a solution of system (2.23). However, many methods for linear semi-infinite programming require one to find the most violated constraint at each iteration. This is computationally expensive.

In the next subsection, we review the relaxation method for solving (2.23) with denumerable inequalities, while in subsection 2.2.2 we discuss the discretization method for solving (2.23) with uncountable inequalities.

Throughout the remainder of this section, $\Omega$ denotes the solution set of (2.23) and $d_\Omega(x) = \min \{ \| x - y \|_2 : y \in \Omega \}$ is the Euclidean distance from $x$ to $\Omega$.

### 2.2.1 Relaxation Approach

The relaxation method for solving (2.23) is similar to those in [1] and [52] for solving (2.1). The basic idea of the relaxation method is simple. Suppose a current solution $x^k$ is not feasible for system (2.23) and let $x^{k+1}$ be the orthogonal projection of $x^k$ onto the boundary of a linear inequality near the inequality most violated by $x^k$.

Jerostow [39] proposed a projection method for solving a system of denumerably infinite linear inequalities

$$
a_t^T x \leq b_t, \forall t \in T \equiv N,
$$

where $a_t \in \mathbb{R}^n$ with $\| a_t \| = 1$ and $b_t \in \mathbb{R}$, $\forall t \in T$.

**Algorithm (Jerostow)**

Step 0: Pick an $x^0 \in \mathbb{R}^n$ and a sufficiently small positive number $\varepsilon$.

Set $k = 0$ and $j = 1$.

Step 1: Compute $\delta = \max_{i=1, \ldots, j} \{ a_i^T x^k - b_i \}$ and $\bar{\delta} = \min \{ \delta, \varepsilon \}$

Step 2: If no $t \in T$ exists with $a_t^T x^k - b_t \geq \bar{\delta}$, then stop.

Otherwise, find a $\hat{t} \in T$ such that $\delta_{\hat{t}} = a_{\hat{t}}^T x^k - b_{\hat{t}} \geq \bar{\delta}$.

Step 3: Let $x^{k+1} = x^k - \delta_{\hat{t}} a_{\hat{t}}$.

Set $k = k + 1$ and $j = j + 1$. Return to Step 1.
This algorithm does not require one to find the most violated constraint at every iteration, and allows one to work with only a finite subset of the constraints. Jeroslow also showed that for any given $\varepsilon > 0$ the algorithm terminates with $x^k$ for some finite $k \leq d_{\varepsilon}(x^0)$ and that $a_i^T x^{(k)} \leq b_t + \varepsilon$, $\forall t \in T$. Furthermore, if $\varepsilon = 0$ then the sequence $\{x^k\}_{k=0}^\infty$ has a limit $x^*$ which is feasible for (2.24). Another projection method for solving system (2.24) can be found in [38].

The relaxation method for solving system (2.24) has the same advantage as that for solving system (2.1). However, it also inherits the drawback of slow convergence and of not being able to detect inconsistency in system (2.24).

### 2.2.2 Discretization Approach

Consider the following system of linear inequalities:

$$a(t)^T x \leq b(t), \quad \forall t \in T,$$

where $x \in \mathbb{R}^n$, $T$ is a compact subset of $\mathbb{R}^l$, $l \in \mathbb{N}$, $|T| = \infty$, and $a : T \to \mathbb{R}^n$ and $b : T \to \mathbb{R}$ are continuous function on $T$. Because $|T| = \infty$, for a given $x^k \in \mathbb{R}^n$, in general, there is no efficient way to check its feasibility. Thus in most cases, the idea of discretizing $T$ becomes an acceptable approach for checking the feasibility of $x^k$.

A general discretization algorithm for solving (2.25) is as follows.

**Algorithm** (discretization method)

1. **Step 0:** Choose $T_0 \subset T$ with $|T_0| < \infty$. Set $k = 0$.

2. **Step 1:** Find a solution $x^k \in \Omega_k \triangleq \{x \in \mathbb{R}^n | a_i^T x \leq b_t, t \in T_k\}$.

3. **Step 2:** Find $\bar{T}_k \subset T$ with $|\bar{T}_k| < \infty$ such that $a_i^T x^k - b_t > 0$, $\forall t \in \bar{T}_k$.

4. **Step 3:** If $\bar{T}_k = \emptyset$ then stop. $x^k$ solves (2.25). Otherwise,

5. **Step 4:** Let $T_{k+1} = \bar{T}_k \cup T_k$. Set $k = k + 1$ and return to Step 1.

Notice that the discretization method can always provide a feasible solution if $\Omega \neq \emptyset$. However, at each iteration it requires a method for solving a subsystem of linear
inequalities and a procedure to identify violated constraints. This is computationally costly.

When the statement “let $T_{k+1} = \overline{T}_k \cup T_k$” in Step 4 is replaced to “choose $T_{k+1} \subset T$ such that $T_k \subset T_{k+1}$”, this algorithm is called the “exchange method”. The term “exchange” refers to the fact that at each iteration a number of new constraints are added and some old constraints may be deleted. In this case, a clever constraint-deleting strategy is required, and may well be difficult to implement. Notice that the discretization method can be considered as a special case of exchange method without a constraint deleting procedure.
Chapter 3

Systems of Finitely Many Linear Inequalities

Given an $m \times n$ matrix $A$ and an $m$-dimensional vector $b$, we consider the problem of finding an $n$-dimensional vector $x$ such that

\[
\text{LI: } \begin{cases} 
Ax \leq b \\
x \geq 0.
\end{cases} 
\]  

(3.1)

As assumed by most interior-point methods, we assume that system (3.1) has an interior feasible solution $x_0 > 0$ such that $Ax_0 < b$.

It is well-known that problem (3.1) can be treated as a linear programming problem. In Fang [14] and Fang and Tsao [17], an entropic perturbation scheme was proposed to derive an unconstrained convex dual for solving a linear programming problem. We will adopt this approach to solve the system (3.1). In Section 3.1, we review the entropic perturbation method for linear programming. Some elementary properties are discussed. The proposed approach is presented in Section 3.2 with numerical experiments reported in Section 3.3.

Throughout this chapter, we denote $e = (1, 1, ..., 1)^T$ and $\exp(w) = (\exp(w_1), \exp(w_2), ..., \exp(w_m))^T$, for $w \in \mathbb{R}^m$. 

16
3.1 Entropic Perturbation

Consider the following linearly constrained minimum entropy optimization problem:

$$\begin{align*}
\min & \quad \sum_{i=1}^{n} x_i \ln x_i + \sum_{j=1}^{m} s_j \ln s_j \\
\text{s.t.} & \quad Ax + s = b \\
& \quad x \geq 0, s \geq 0.
\end{align*}$$

(3.2)

Let

$$\hat{A} \triangleq \begin{bmatrix} A & I_m \end{bmatrix},$$

$$\hat{x} \triangleq \begin{bmatrix} x \\ s \end{bmatrix},$$

where $I_m$ is an $m \times m$ identity matrix. Then $\hat{A}$ becomes an $m \times l$ matrix and $\hat{x}$ an $l$-dimensional vector with $l = n + m$.

For any $p > 0$, problem (3.2) is equivalent to the following problem:

$$\begin{align*}
L_{(p)}: \quad \min & \quad f_{(p)}(\hat{x}) \equiv \frac{1}{p} \sum_{k=1}^{l} \hat{x}_k \ln \hat{x}_k \\
\text{s.t.} & \quad \hat{A} \hat{x} = b \\
& \quad \hat{x} \geq 0.
\end{align*}$$

(3.3)

Notice that $L_{(p)}$ is a standard form linear program with entropic perturbation [17].

We now derive an unconstrained dual program of (3.3) by using the following simple inequality

$$\ln z \leq z - 1, \quad \forall \ z > 0.$$  

(3.4)

Notice that inequality (3.4) becomes an equality if and only if $z = 1$.

For any $p > 0$, $w_j \in R$, $j = 1, ..., m$, and $\hat{x}_k > 0$, $k = 1, ..., l$, we define

$$z_k(p) = \frac{\exp \left\{ p \sum_{j=1}^{m} \hat{a}_{jk} w_j - 1 \right\}}{\hat{x}_k}, \quad k = 1, ..., l.$$  

(3.5)

Since $\hat{x}_k > 0$, $z_k(p)$ is also be positive. Using (3.4), we have

$$\left( p \sum_{j=1}^{m} \hat{a}_{jk} w_j - 1 \right) - \ln \hat{x}_k \leq \frac{\exp \left\{ p \sum_{j=1}^{m} \hat{a}_{jk} w_j - 1 \right\}}{\hat{x}_k} - 1.$$

(3.6)
Consequently,
\[ \hat{x}_k(p \sum_{j=1}^{m} \hat{a}_{jk}w_j) - \hat{x}_k \ln \hat{x}_k \leq \exp\{p \sum_{j=1}^{m} \hat{a}_{jk}w_j - 1\}. \quad (3.7) \]

Multiplying both sides of (3.7) by $1/p$ and then summing over $k$, we have
\[ \sum_{k=1}^{l} \hat{x}_k \left( \sum_{j=1}^{m} \hat{a}_{jk}w_j \right) - \frac{1}{p} \sum_{k=1}^{l} \exp\{p \sum_{j=1}^{m} \hat{a}_{jk}w_j - 1\} \leq \frac{1}{p} \sum_{k=1}^{l} \hat{x}_k \ln \hat{x}_k. \quad (3.8) \]

If $\hat{x}$ satisfies $\hat{A}\hat{x} = b$, then
\[ \sum_{k=1}^{l} \hat{x}_k \left( \sum_{j=1}^{m} \hat{a}_{jk}w_j \right) = \sum_{j=1}^{m} \sum_{k=1}^{l} (\hat{a}_{jk}\hat{x}_k)w_j = \sum_{j=1}^{m} b_jw_j. \quad (3.9) \]

Therefore, for any $\hat{x} > 0$ such that $\hat{A}\hat{x} = b$, we have
\[ \sum_{j=1}^{m} b_jw_j - \frac{1}{p} \sum_{k=1}^{l} \exp\{p \sum_{j=1}^{m} \hat{a}_{jk}w_j - 1\} \leq \frac{1}{p} \sum_{k=1}^{l} \hat{x}_k \ln \hat{x}_k. \quad (3.10) \]

Consequently, we can define an unconstrained dual problem of $L_{(p)}$ as follows:
\[ \text{DL}_{(p)}: \max_{w \in \mathbb{R}^m} \left\{ g_{(p)}(w) = \sum_{j=1}^{m} b_jw_j - \frac{1}{p} \sum_{k=1}^{l} \exp\{p \sum_{j=1}^{m} \hat{a}_{jk}w_j - 1\} \right\}. \quad (3.11) \]

Denote the minimum objective value for $L_{(p)}$ by $\min(L_{(p)})$ and the maximum objective value for $\text{DL}_{(p)}$ by $\max(\text{DL}_{(p)})$. An immediate result is the so-called weak duality theorem, which can be stated as follows:

**Theorem 1** If there exists an $\hat{x} > 0$ such that $\hat{A}\hat{x} = b$, then
\[ \min(L_{(p)}) \geq \max(\text{DL}_{(p)}). \quad (3.12) \]

### 3.1.1 Properties of $g_{(p)}(w)$

For $w \in \mathbb{R}^m$ and any $p > 0$, it is obvious that $g_{(p)}(w)$ is continuously differentiable with respect to $w$. By taking the first and second derivatives, we obtain its gradient
\[ \nabla g_{(p)}(w) = b - A \exp\{pA^T w - e\} = b - A \exp\{pA^T w - e\} - \exp\{pw - e\}, \quad (3.13) \]
and Hessian
\[ \nabla^2 g(p)(w) = -p\hat{A}D_p(w)\hat{A}^T, \] (3.14)
respectively, where
\[ D_p(w) = \text{diag}(d^1_p(w), \ldots, d^l_p(w)), \quad \text{and} \quad d^k_p(w) = \exp \left\{ p \sum_{j=1}^m \hat{a}_{jk} w_j - 1 \right\}, \quad \text{for} \quad k = 1, \ldots, l. \] (3.15)

**Lemma 1** For any \( p > 0 \), \( g(p)(w) \) is a strictly concave function.

*Proof.* Since \( d^k_p = \exp \left\{ p \sum_{j=1}^m \hat{a}_{jk} w_j - 1 \right\} > 0 \), for \( k = 1, \ldots, l \), \( D_p \) is a positive definite matrix. By the definition of \( \hat{A} \), we know that \( \hat{A} \) always has full row rank. Thus \( \hat{A}D_p\hat{A}^T \) is positive definite [37]. Hence, \( g(p)(w) \) is a strictly concave function. \( \square \)

We now state a strong duality theorem.

**Theorem 2** If (LI) has an interior feasible solution \( x_0 > 0 \) such that \( Ax_0 < b \), then there is a unique optimal solution pair \( (\hat{x}^*(p), w^*(p)) \) for \( L(p) \) and \( DL(p) \) with \( \hat{x}^*_k(p) = d^k_p(w^*) = \exp \left\{ p \sum_{j=1}^m \hat{a}_{jk} w^*_j(p) - 1 \right\} \), for \( k = 1, \ldots, l \). Moreover, in this case, \( \min(L(p)) = \max(DL(p)) \).

*Proof.* The proof is similar to that in [16] (p207 - p208). \( \square \)

This theorem ensures that, under the conditions stipulated, it suffices to maximize an unconstrained convex function \( g(p)(w) \) in order to solve system (3.1).

**Remark 1** We denote
\[ \hat{x}^*(p) = \exp \left\{ p\hat{A}^T w^*(p) - e \right\} \] (3.16)
as the dual-to-primal conversion formula. Consequently, we have
\[ x^*(p) = \exp \left\{ pA^T w^*(p) - e \right\}, \] (3.17)
and
\[ s^*(p) = \exp \left\{ pw^*(p) - e \right\} \] (3.18)
Ideally, a numerical algorithm for solving system (3.1) will not only be computationally efficient but will also have the ability to detect an inconsistency in the system. The following theorem establishes the equivalence between the inconsistency of system (3.1) and the unboundedness of \( g(p)(w) \) (for any \( p > 0 \)), and hence enables us to detect possible inconsistency of system (3.1) by checking possible unboundedness of \( g(p)(w) \) for any particular \( p > 0 \).

**Theorem 3** \( \{x \in \mathbb{R}^n | Ax \leq b, x \geq 0 \} = \emptyset \) if and only if \( g(p) \) is unbounded from above for any \( p > 0 \).

**Proof.** Suppose that \( \{x \in \mathbb{R}^n | Ax \leq b, x \geq 0 \} = \emptyset \). If \( g(p) \) is bounded from above for some \( p > 0 \), then there exists a positive number \( M_0 > 0 \) such that \( g(p)(w) \leq M_0, \forall w \in \mathbb{R}^m \). A variant of Farka’s Lemma says that \( \{x \in \mathbb{R}^n | Ax \leq b, x \geq 0 \} = \emptyset \) if and only if \( \Omega \triangleq \{w \in \mathbb{R}^n | A^Tw \leq 0, w \leq 0, b^Tw > 0 \} \neq \emptyset \). Therefore, there exists a \( w^0 \in \Omega \) with \( A^Tw^0 \leq 0, w^0 \leq 0 \). From equation (3.11) we have

\[
g(p_0)(w^0) = b^Tw^0 - \frac{1}{p_0} \left\{ \sum_{k=1}^n \exp\{p_0 \sum_{j=1}^m a_{jk}w^0_j - 1\} + \sum_{j=1}^m \exp\{p_0w^0_j - 1\} \right\}
\geq b^Tw^0 - \frac{(m+n)\exp(-1)}{p_0}
> b^Tw^0 - \frac{(m+n)}{2p_0}.
\]

Define \( \lambda = \frac{M_0 + \frac{(m+n)}{2p_0}}{b^Tw^0} \) and \( w^* = \lambda w_0 \). Since \( \lambda > 0 \), \( w^* \in \Omega \). Consequently,

\[
g(p_0)(w^*) > b^Tw^* - \frac{(m+n)}{2p_0} = \frac{M_0 + \frac{(m+n)}{2p_0}}{b^Tw^0} b^Tw^0 - \frac{(m+n)}{2p_0} = M_0.
\]

This contradiction completes the necessity part of the proof.

Now suppose that \( g(p_1) \) is unbounded from above for a particular \( p_1 > 0 \). Since \( g(p_1) \) is a continuous strictly concave function in \( \mathbb{R}^m \), there exists a \( w^{**} \in \mathbb{R}^m \) such that

\[
\lim_{\alpha \to +\infty} g(p_1)(\alpha w^{**}) = +\infty.
\]

We claim that \( w^{**} \) must belong to \( \Omega \).
First, we prove that $b^T w^{**} > 0$. Since $\lim_{\alpha \to +\infty} g_{(p_1)}(\alpha w^{**}) = +\infty$, for any given $M > 0$ there exists a sufficiently large $\alpha_M > 0$ such that

$$
0 < M \leq g_{(p_1)}(\alpha_M w^{**}) = \alpha_M b^T w^{**} - \frac{1}{p_1} \left\{ \sum_{k=1}^{n} \exp\{p_1 \alpha_M \sum_{j=1}^{m} a_{jk} w_j^{**} - 1\} \right\} < \alpha_M b^T w^{**}.
$$

It follows that $b^T w^{**} > 0$. Second, we prove that $w^{**} \leq 0$. Suppose that $w^{**}$ has a positive element $w_i^{**} > 0$, for some $i$, then

$$
g_{(p_1)}(\alpha w^{**}) = \alpha b^T w^{**} - \frac{1}{p_1} \left\{ \sum_{k=1}^{n} \exp\{p_1 \alpha \sum_{j=1}^{m} a_{jk} w_j^{**} - 1\} \right\} \leq \alpha \|b\| \|w^{**}\| - \frac{1}{p_1} \exp \{p_1 \alpha w_i^{**} - 1\}.
$$

Consequently, $\lim_{\alpha \to +\infty} g_{(p_1)}(\alpha w^{**}) = -\infty$. This contradiction proves that $w^{**} \leq 0$. Finally, we show that $A^T w^{**} \leq 0$. Suppose that $\sum_{j=1}^{m} a_{jk} w_j^{**} > 0$ for some $k$. Then,

$$
g_{(p_1)}(\alpha w^{**}) \leq \alpha \|b\| \|w^{**}\| - \frac{1}{p_1} \exp \left\{ p_1 \alpha \sum_{j=1}^{m} a_{jk} w_j^{**} - 1 \right\}.
$$

Again, we have $\lim_{\alpha \to +\infty} g_{(p_1)}(\alpha w^{**}) = -\infty$. This proves that $A^T w^{**} \leq 0$. Therefore, $w^{**} \in \Omega$, and hence $\Omega \neq \emptyset$. By Farka’s Lemma we have $\{x | Ax \leq b, x \geq 0\} = \emptyset$. This completes the sufficiency part of the proof.

\[ \Box \]

### 3.2 Proposed Approach

Since $p$ can be of any positive value, we set $p = 1$ to solve the system of inequalities (LI) by maximizing $g_{(1)}(w)$. Notice that, since $g_{(1)}(w)$ is a smooth function, we can apply unconstrained optimization algorithms to solve this problem. In this section, we outline an unconstrained convex programming approach to solve (LI) under the interior-point assumption that there exists an $x^0 > 0$ such that $Ax^0 < b$. 

21
**Algorithm 1:** Entropic Perturbation (EP) Algorithm

**Step 1:** Initialize an $m$-dimensional vector $w^0$, and set $k = 0$

**Step 2:** Calculate $x_i^0 = \exp\left\{\sum_{j=1}^{m} a_{ji}w_j^0 - 1\right\}$, for $i = 1, ..., n$.

**Step 3:** While $Ax^k - b > 0$

1. Calculate a movement direction vector $d^k_w$.
2. Update $w^{k+1}$ with $w^k + d^k_w$.
3. Calculate $x_i^{k+1} = \exp\left\{\sum_{j=1}^{m} a_{ji}w_j^{k+1} - 1\right\}$, for $i = 1, ..., n$.
4. Update $k$ with $k + 1$.

**Step 4:** Output $x^k$.

We may use any unconstrained optimization method for maximizing $g_1(w)$. The Newton method is very attractive in terms of its convergence rate near the solution, but it may not converge globally. However, combining a Newton direction with a line search results in global convergence. More precisely, if the direction vector $d^k_w$ is chosen to be

\[ d^k_w = \alpha_kd^k_g(w^k), \]  

where

\[ d^k_g(w^k) = -[\nabla^2 g_1(w^k)]^{-1}(\nabla g_1(w^k)) \]  

and

\[ \alpha_k = \arg \max_{\alpha \geq 0} g_1(w^k + \alpha d^k_g(w^k)), \]  

then, the entropic perturbation algorithm becomes global convergent with a quadratic rate of local convergence. This fact is established by the following theorem.

**Theorem 4** If $LI$ has an interior feasible solution, when the movement direction vector $d_w$ is given by Equation (3.19), then the sequence $\{w^k\}$ generated by the EP algorithm globally converges to a unique $w^*$, with $Ax^*-b < 0$, where $x^* = \exp\{A^Tw^*-e\}$. Furthermore, the rate of convergence is at least two.

*Proof.* By Theorem 2, $g_1(w)$ has a unique optimal solution. Let us denote the optimal solution by $w^*$. Since $-\nabla^2 g_1(w^k)$ is positive definite for all $w^k$, $d^k_w$ is an ascent direction. It follows that $\lim_{k \to \infty} w^k = w^*$.

22
Since \( \exp\{A^T w - e\} \) is a continuous function, we have
\[
\lim_{k \to \infty} x^k = \lim_{k \to \infty} \exp\{A^T w^k - e\} = \exp\{A^T w^* - e\} = x^*
\]

Note that the necessary and sufficient condition for \( w^* \) being a maximizer is
\[
\nabla g_{(1)}(w^*) = 0. \tag{3.22}
\]

Let \( s_j^* = \exp(w_j^* - 1) \), for \( j = 1, \ldots, m \). We have
\[
0 = \nabla g_{(1)}(w^*) = b - \hat{A} \exp\{\hat{A}^T w^* - e\} = b - |A| I_m \begin{bmatrix} x^* \\ s^* \end{bmatrix} = b - (Ax^* + s^*)
\tag{3.23}
\]

Since \( s_j^* > 0 \), for \( j = 1, \ldots, m \), we have \( Ax^* - b < 0 \).

The proof of rate of convergence can be found in [46] (p225). \( \square \)

**Remark 2** Theoretically, we can let \( p \) in DL\(_{(p)}\) defined by (3.11) be any positive constant, however the algorithm may encounter overflow problems when \( p \) is large.

**Remark 3** The main idea we used to develop the algorithm is based on the techniques of the entropic perturbation for linear programming. However, we can adopt another point of view from the dual problem DL\(_{(p)}\) by defining
\[
\tilde{g}_{(p)}(w) \triangleq \sum_{j=1}^{m} b_j w_j - \frac{1}{p} \sum_{k=1}^{l} \exp\{p \sum_{j=1}^{m} \hat{a}_{jk} w_j - c_k\}, \tag{3.24}
\]
as the dual objective function, where \( c = [c_1, c_2, \ldots, c_l]^T \) is any constant vector. Note that \( \tilde{g}_{(p)}(w) \) has the same smoothness property as \( g_{(p)}(w) \) with the gradient
\[
\nabla \tilde{g}_{(p)}(w) = b - \hat{A} \exp\{p\hat{A}^T w - c\}. \tag{3.25}
\]

If we rewrite the dual-to-primal conversion formula as \( \hat{x}^*(p) = \exp\{p\hat{A}^T \hat{w}^*(p) - c\} \), then the necessary and sufficient conditions for maximizing \( \tilde{g}_{(p)}(w) \) will result in
\( \hat{\alpha}^*(p) = b \) with \( Ax^*(p) < b \), where \( x^*(p) = \exp \{ pA^T \nu^*(p) - c \} \). Thus, we can state an alternative algorithm for solving (3.1).

**Algorithm 2: \( (p = 1, c = 0) \)**

Step 1: Initialize \( m\)-dimensional \( w^0 \), and set \( k = 0 \)

Step 2: Calculate \( x_i^0 = \exp \left\{ \sum_{j=1}^{m} a_{ji}w_j^0 \right\} \), \( i = 1, ..., n \).

Step 3: While \( Ax^k - b > 0 \)

- Calculate a movement direction vector \( d^k_w \).
- Update \( w^k \) with \( w^k + d^k_w \).
- Calculate \( x_i^{k+1} = \exp \left\{ \sum_{j=1}^{m} a_{ji}w_j^{k+1} \right\} \), for \( i = 1, ..., n \)
- Update \( k \) with \( k + 1 \).

Step 4: Output \( x^k \).

### 3.3 Numerical Experiments

In this section, we report our computational experiments using the EP algorithm with Newton direction to solve system (3.1). Due to the robustness and popularity of the Quasi-Newton methods, our experiments also include the use of the BFGS algorithm [46] for calculating the movement direction. The Surrogate Constraint (SC) method proposed by Yang and Murty [69] is implemented for comparison purposes. All the algorithms have been coded in MATLAB with dense matrix structure.

In particular, we study the effect of three factors: (a) the “absolute size” of the problem, i.e., \( m \) and \( n \), (b) the “relative size” of the problem, i.e., \( n/m \), and (c) the distance between the initial solution and the feasible region.

We develop two groups of experiments with different sizes of the problem. In the first group, we vary both the absolute size and the relative size of the test problems. In the second, we vary the absolute size and the distance between the initial solution and the feasible region for a particular relative size.
3.3.1 Implementation Issues

Before reporting our computational experience, we discuss some implementation issues.

1. **Numerical overflow**: Evaluation of exponential functions is required in calculating $\nabla g_{(1)}(w)$ and $\nabla^2 g_{(1)}(w)$. By examining the formulas (3.13) and (3.14) for $\nabla g_{(1)}(w)$ and $\nabla^2 g_{(1)}(w)$, respectively, it is clear that arithmetic overflow may occur if matrix $A$ and the initial vector $w^0$ are not properly defined. The way we generate matrix $A$ is presented in Section 3.3.3. Since we always initialize $w^0 = 0$, the overflow problem is avoided in our computational experiments.

2. **Matrix computations**: Superficially, it would seem that converting the $m \times n$ matrix $A$ into an $m \times (m + n)$ matrix $\hat{A}$ should result in a heavier computation burden. In reality, the evaluation of $\nabla g_{(1)}(w)$ and $\nabla^2 g_{(1)}(w)$ can be accomplished without involving $\hat{A}$, and no extra storage for $I_n$ is needed.

Another major effort is required to find the Newton direction,

$$d_w = -[\nabla^2 g_{(1)}(w)]^{-1} \nabla g_{(1)}(w). \tag{3.26}$$

Direct inversion of the Hessian is computationally expensive. We use the “backslash” operator of Matlab to solve (3.26). This operator performs Gaussian Elimination on the following system:

$$-[\nabla^2 g_{(1)}(w)]d_w = \nabla g_{(1)}(w). \tag{3.27}$$

Since $-\nabla^2 g_{(1)}(w)$ is positive definite for any $w \in \mathbb{R}^m$, we can also apply Cholesky Factorization method to solve (3.27). The total flops for executing Gaussian Elimination and Cholesky Factorization are about $\frac{2}{3}m^3$ and $\frac{1}{3}m^3$, respectively [66].

3. **Step size**: In the EP algorithm the pure form of Newton method is used, i.e., the step size is set to 1. However, in general, a step size must be determined. Since $g_{(1)}(w)$ is strictly concave, a bisection line search is used to calculate the step size in an efficient manner.

25
4. Surrogate constraint method: The two parameters, $\gamma$ and $\pi$, in (2.9) for the surrogate constraint method are set as $\gamma = 1.7$ and

$$
\pi_i(x^k) = \begin{cases} 
0.2 \times \frac{(\bar{\alpha}_i x^k - \bar{b}_i)}{\sum_{j \in I^k} (\bar{\alpha}_j x^k - \bar{b}_j)} + 0.8 \times \frac{1}{|I^k|} & i \in I^k \\
0 & i \notin I^k,
\end{cases}
$$

(3.28)

where $\bar{\alpha} = [\frac{4}{-f_n}]$, $\bar{b} = [\frac{b}{6_n}]$ and $I^k = \{ i | \bar{\alpha}_i x^k - \bar{b}_i > 0, 1 \leq i \leq m + n \}$.

### 3.3.2 Computational Experience

Test problems were randomly generated such that the interior point assumption is satisfied. The problems fall into two sizes: “small” ($n, m \leq 100$) and “large” ($n, m \geq 100$). The nonnegativity constraints, $x \geq 0$, are not counted in the problem size. The test problems were run on a Pentium III 700MHZ computer with 256 MB memory.

Here, we briefly describe how the “$A$” and “$b$” are randomly generate to satisfy the interior point assumption. First we generate a solution vector $x = (x_1, x_2, ..., x_n)^T$ and a slack vector $s = (s_1, s_2, ..., s_m)^T$, where $x_i$, for $i = 1, ..., n$, and $s_j$, for $j = 1, ..., m$, are sampled from $Uniform(0, 1)$. Then, we randomly generate the coefficients $a_{ji}$, $j = 1, ..., m - 1$, $i = 1, ..., n$, using $Normal(0, 1)$. A bounding constraint is added to $A$ by setting $a_{mi} = \frac{1}{n}$, for $i = 1, ..., n$. All the elements of $A$, $x$ and $s$ are specified with four-digit precision. The right-hand side of the inequalities is defined by $b = Ax + 0.05 \times s$ to ensure the validity of the interior point assumption.

### 3.3.2.1 Effect of the Problem Size ($n, m, m/n$)

200 instances were generated for each small size problem and 50 instances for each large size problem. The results are summarized in four tables for each problem set (Table 3.1 - Table 3.3 and Table 3.4 - Table 3.6).

In the tables, $n$ and $m$ represent the number of variables and constraints of the system (3.1), respectively. We say system (3.1) is well-determined if $\frac{n}{m} \approx 1$, under-determined if $\frac{n}{m} \geq 1$, and over-determined if $\frac{n}{m} \leq 1$. Moreover, “Entropic” denotes the proposed EP method and “Surrogate” the SC method.

In some instances, the SC method was not able to converge to a feasible solution effectively. So we set an upper bound of 10,000 iterations for small-size problems and an upper bound of 25,000 iterations for large-size problems.

We executed three different implementations of the EP method:
• **NG**: Newton method with Gaussian elimination.

• **CF**: Newton method with the Cholesky Factorization.

• **QN**: Quasi-Newton method (BFGS) with line search.

Four statistics are reported in the tables.

• **IT**: the average number of iterations required for finding a feasible solution for those instances in which a feasible solution is actually found.

• **CPU**: the average CPU time (seconds) required for finding a feasible solution for those instances in which a feasible solution is actually found.

• **UF**: the number of instances in which the SC method did not converge to a feasible solution within the maximum allowed number of iterations.

• **AER**: the average violation ($l_2$-norm) associated with the violated constraints for the instances in which a feasible solution was not found by the SC method.

The relationship between the problem size and the total number of required iterations is shown in Figures 3.1, 3.3 and 3.5. The relationship between the problem size and the required CPU time is shown in Figures 3.2, 3.4 and 3.6.

Some important observations can be drawn from the computational statistics.

(i) According to our experience, the EP method always converges to an exact feasible solution as stated in Theorem 4. The SC method may only find an approximate solution.

(ii) The EP method (particularly **NG** and **CF**) required very few iterations ($\leq 10$ in our experiments) to find a feasible solution. This observation also confirms the quadratic rate of local convergence enjoyed by the EP method.

(iii) The total number of iterations required for the EP method grows very slowly with the size of the problem. Especially for the **NG** and **CF** versions, the total number of iterations varies little with respect to the problem size. This means that the EP method is very robust. In contrast, the number of iterations required by the SC method to solve over-determined system grows very fast.
(iv) The CPU time required for solving the over- and well-determined systems using the QN version grows very fast as the problem size increases. Therefore, it is not a good version. The CF version outperforms the NG version, although both the CPU times increase slowly as the problem size increases.

(v) For the under- and well-determined system, the SC method converges very fast. But as noted in (iii), it is unstable for solving over-determined systems. In particular, it failed to find solutions in some cases.

Notice that the numerical results reported by Yang and Murty [69] were also based on $\gamma = 1.7$. We also set $\gamma = 1.0$ and the upper bound of iteration as 10000 in implementing the SC method for the test problems, and found that the $\gamma = 1.7$ produces much better performance than $\gamma = 1.0$. The numerical results for $\gamma = 1.0$ are reported in Tables 3.7- Table 3.12.

3.3.2.2 Effect of the Distance between the Initial Solution and the Feasible Region

In some special circumstances, an initial solution that is near the feasible region may be found without much difficulty. However, in general, such a solution is not available. The sensitivity of the speed of convergence of an algorithm to the distance between an initial solution and the feasible region has to be studied. Most algorithms specify a default initial solution. Ideally an algorithm's performance will be insensitive to the selection of the initial solution.

In our experiments, we focused on the over-determined test problems considered in the first group of experiments (i.e., those with $n/m = 0.5$) and varied the distance between the initial solution and the feasible region. For the SC method, one way to vary the distance is to simply move the initial solution away from the origin (the default initial solution). However, with this approach, it is not clear what corresponding moves should be made for the EP method. To provide a fair comparison, we moved the feasible region away from the origin of the primal space but kept the default initial solutions of both algorithms.

Recall that for each randomly generated test problem, a feasible solution $x = (x_1, x_2, ..., x_n)^T$ was first selected. For each of the test problems generated in this way, we created 5 additional test problems by shifting the initial feasible region by
a deterministic amount of 0.5, 1, 2, 4 and 8, respectively. Specifically, a shift of 0.5 
means that we set $b = A(x + 0.5e) + 0.05s$ where $e = (1, 1, \ldots, 1)^T$.

An additional statistic was added to track performance:

- **WST**: the average time spent (or “wasted”) on those instances for which the 
  SC method did not produce a feasible solution in 25,000 iterations.

In this set of test problems, only the **CF** (with line search) version was used for 
the EP method. The numerical results are tabulated in Tables 3.13 - 3.16.

Two key observations can be made here:

(i) No matter the magnitude of the shift, the EP method always finds a feasible 
solution in few iterations. The CPU time and the total number of iterations only 
increase slightly as the size of the shift is increased. This shows the robustness 
of the EP method.

(ii) In contrast, the **UF** and **AER** statistics for the SC method deteriorate as the 
the shift is increased. This demonstrates that the SC method may become 
ineffective when initialized with a solution that is not well chosen.

### 3.4 Remarks

We have introduced an unconstrained convex programming approach for solving sys-
tems of finitely many linear inequalities which is able to find an exact feasible so-
lation. Specifically, we propose an entropic perturbation algorithm based on the 
Newton’s method. This algorithm can be easily implemented and no parameter set-
ing is needed. In theory, the algorithm converges globally with a quadratic rate. 
In practice, the computational experience is quite encouraging, especially for solving 
over-determined systems. The proposed method, unlike the relaxation methods, has 
the ability to detect inconsistency of a system of linear inequalities.

Some experiments were conducted to compare the performance of the proposed 
algorithm to that of the surrogate constraint algorithm developed by Yang and Murty. 
Our computational comparison indicates that the proposed method is particularly 
suitable for solving over-determined systems in which an initial solution near the 
feasible region is not given or cannot be found with ease.
### Table 3.1: small-size problems with under-determined systems ($\gamma = 1.7$)

<table>
<thead>
<tr>
<th>Size</th>
<th>NG CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.96</td>
<td>0.0041</td>
<td>1.96</td>
<td>0.0027</td>
<td>3.66</td>
<td>0.0052</td>
</tr>
<tr>
<td>40</td>
<td>2.21</td>
<td>0.0037</td>
<td>2.21</td>
<td>0.0065</td>
<td>5.46</td>
<td>0.0109</td>
</tr>
<tr>
<td>60</td>
<td>2.35</td>
<td>0.0101</td>
<td>2.35</td>
<td>0.0137</td>
<td>6.24</td>
<td>0.0205</td>
</tr>
<tr>
<td>80</td>
<td>2.56</td>
<td>0.0207</td>
<td>2.56</td>
<td>0.0244</td>
<td>7.07</td>
<td>0.0375</td>
</tr>
<tr>
<td>100</td>
<td>2.75</td>
<td>0.0401</td>
<td>2.75</td>
<td>0.0388</td>
<td>7.75</td>
<td>0.0587</td>
</tr>
</tbody>
</table>

### Table 3.2: small-size problems with well-determined systems ($\gamma = 1.7$)

<table>
<thead>
<tr>
<th>Size</th>
<th>NG CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>3.24</td>
<td>0.0033</td>
<td>3.24</td>
<td>0.0004</td>
<td>10.63</td>
<td>0.0198</td>
</tr>
<tr>
<td>40</td>
<td>3.48</td>
<td>0.0181</td>
<td>3.48</td>
<td>0.0249</td>
<td>17.79</td>
<td>0.0847</td>
</tr>
<tr>
<td>60</td>
<td>3.23</td>
<td>0.0436</td>
<td>3.23</td>
<td>0.0440</td>
<td>22.03</td>
<td>0.2071</td>
</tr>
<tr>
<td>80</td>
<td>3.20</td>
<td>0.0998</td>
<td>3.20</td>
<td>0.0740</td>
<td>26.52</td>
<td>0.4373</td>
</tr>
<tr>
<td>100</td>
<td>3.38</td>
<td>0.3854</td>
<td>3.37</td>
<td>0.1144</td>
<td>31.78</td>
<td>0.8375</td>
</tr>
</tbody>
</table>

### Table 3.3: small-size problems with over-determined systems ($\gamma = 1.7$)

<table>
<thead>
<tr>
<th>Size</th>
<th>NG CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
<th>IT CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5.41</td>
<td>0.0048</td>
<td>5.44</td>
<td>0.0146</td>
<td>23.55</td>
<td>0.0439</td>
</tr>
<tr>
<td>20</td>
<td>5.99</td>
<td>0.0233</td>
<td>6.00</td>
<td>0.0388</td>
<td>43.79</td>
<td>0.2031</td>
</tr>
<tr>
<td>30</td>
<td>6.07</td>
<td>0.0589</td>
<td>6.07</td>
<td>0.0708</td>
<td>58.30</td>
<td>0.5300</td>
</tr>
<tr>
<td>40</td>
<td>6.25</td>
<td>0.1231</td>
<td>6.25</td>
<td>0.1124</td>
<td>69.73</td>
<td>1.0833</td>
</tr>
<tr>
<td>50</td>
<td>6.24</td>
<td>0.2216</td>
<td>6.28</td>
<td>0.2102</td>
<td>82.79</td>
<td>2.1933</td>
</tr>
</tbody>
</table>

30
Table 3.4: large-size problems with under-determined systems ($\gamma = 1.7$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>NG</td>
<td>CF</td>
</tr>
<tr>
<td>200 100</td>
<td>3.00 0.7346</td>
<td>3.00 0.7198</td>
</tr>
<tr>
<td>300 150</td>
<td>3.00 2.7454</td>
<td>3.00 1.9730</td>
</tr>
<tr>
<td>400 200</td>
<td>3.00 6.3939</td>
<td>3.00 3.9978</td>
</tr>
<tr>
<td>500 250</td>
<td>3.00 12.511</td>
<td>3.00 7.1776</td>
</tr>
<tr>
<td>600 300</td>
<td>3.00 21.865</td>
<td>3.00 12.216</td>
</tr>
</tbody>
</table>

Table 3.5: large-size problems with well-determined systems ($\gamma = 1.7$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>NG</td>
<td>CF</td>
</tr>
<tr>
<td>200 200</td>
<td>3.74 4.1472</td>
<td>3.72 1.7865</td>
</tr>
<tr>
<td>300 300</td>
<td>3.42 13.422</td>
<td>3.42 5.0252</td>
</tr>
<tr>
<td>400 400</td>
<td>3.80 36.334</td>
<td>3.89 14.342</td>
</tr>
<tr>
<td>500 500</td>
<td>4.30 78.168</td>
<td>4.34 28.652</td>
</tr>
<tr>
<td>600 600</td>
<td>4.16 127.38</td>
<td>4.16 40.555</td>
</tr>
</tbody>
</table>

Table 3.6: large-size problems with over-determined systems ($\gamma = 1.7$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>NG</td>
<td>CF</td>
</tr>
<tr>
<td>100 200</td>
<td>6.90 5.0136</td>
<td>6.86 2.2192</td>
</tr>
<tr>
<td>150 300</td>
<td>7.10 18.196</td>
<td>7.12 5.9882</td>
</tr>
<tr>
<td>200 400</td>
<td>7.12 45.587</td>
<td>7.12 13.341</td>
</tr>
<tr>
<td>250 500</td>
<td>7.74 93.563</td>
<td>7.74 25.588</td>
</tr>
<tr>
<td>300 600</td>
<td>7.76 160.91</td>
<td>7.76 40.860</td>
</tr>
</tbody>
</table>
Table 3.7: small-size problems with under-determined systems ($\gamma = 1$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NG</td>
<td>CF</td>
</tr>
<tr>
<td>10 10</td>
<td>1.96 0.0014 1.96 0.0027 3.66 0.0052</td>
<td>43.73 0.0108 0 0</td>
</tr>
<tr>
<td>40 20</td>
<td>2.21 0.0037 2.21 0.0069 5.46 0.0109</td>
<td>39.80 0.0118 0 0</td>
</tr>
<tr>
<td>60 30</td>
<td>2.35 0.0101 2.35 0.0137 6.24 0.0205</td>
<td>49.46 0.0230 0 0</td>
</tr>
<tr>
<td>80 40</td>
<td>2.56 0.0267 2.56 0.0244 7.07 0.0375</td>
<td>49.25 0.0325 0 0</td>
</tr>
<tr>
<td>100 50</td>
<td>2.75 0.0461 2.75 0.0388 7.75 0.0557</td>
<td>52.91 0.0512 0 0</td>
</tr>
</tbody>
</table>

Table 3.8: small-size problems with well-determined systems ($\gamma = 1$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NG</td>
<td>CF</td>
</tr>
<tr>
<td>10 10</td>
<td>3.24 0.0033 3.24 0.0064 10.63 0.0198</td>
<td>106.0 0.0280 0 0</td>
</tr>
<tr>
<td>40 40</td>
<td>3.48 0.0181 3.48 0.0249 17.79 0.0847</td>
<td>73.31 0.0246 1 5.57E-17</td>
</tr>
<tr>
<td>60 60</td>
<td>3.23 0.0436 3.23 0.0440 22.03 0.2071</td>
<td>84.27 0.0433 0 0</td>
</tr>
<tr>
<td>80 80</td>
<td>3.29 0.0968 3.29 0.0740 26.92 0.4373</td>
<td>92.18 0.0740 0 0</td>
</tr>
<tr>
<td>100 100</td>
<td>3.38 0.3854 3.37 0.1144 31.78 0.8375</td>
<td>94.02 0.1148 0 0</td>
</tr>
</tbody>
</table>

Table 3.9: small-size problems with over-determined systems ($\gamma = 1$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NG</td>
<td>CF</td>
</tr>
<tr>
<td>10 20</td>
<td>5.41 0.0048 5.44 0.0146 23.55 0.0439</td>
<td>572.70 0.1443 36 1.96E-12</td>
</tr>
<tr>
<td>20 40</td>
<td>5.99 0.0233 6.00 0.0388 43.79 0.2031</td>
<td>1515.47 0.4552 51 2.71E-7</td>
</tr>
<tr>
<td>30 60</td>
<td>6.07 0.0569 6.07 0.0708 58.30 0.5300</td>
<td>1903.88 0.6916 43 3.48E-6</td>
</tr>
<tr>
<td>40 80</td>
<td>6.25 0.1231 6.25 0.1124 69.73 1.0833</td>
<td>2183.22 0.9822 45 5.92E-5</td>
</tr>
<tr>
<td>50 100</td>
<td>6.24 0.2216 6.28 0.2102 82.79 2.1933</td>
<td>2590.22 2.2085 42 4.65E-6</td>
</tr>
</tbody>
</table>
### Table 3.10: Large-size problems with under-determined systems ($\gamma = 1$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NG  CF</td>
<td>QN  CF</td>
</tr>
<tr>
<td></td>
<td>IT  CPU</td>
<td>IT  CPU</td>
</tr>
<tr>
<td>200 200</td>
<td>3.74 4.1472 3.72 1.7995 51.46 13.604</td>
<td>122.08 0.7610 0 0</td>
</tr>
<tr>
<td>300 300</td>
<td>3.42 13.422 3.42 5.6252 56.64 45.164</td>
<td>144.78 2.1223 0 0</td>
</tr>
<tr>
<td>400 400</td>
<td>3.89 36.334 3.89 14.342 68.48 121.21</td>
<td>164.94 4.1898 0 0</td>
</tr>
<tr>
<td>500 500</td>
<td>4.36 78.168 4.34 28.652 86.24 275.16</td>
<td>194.06 7.1843 0 0</td>
</tr>
<tr>
<td>600 600</td>
<td>4.16 127.38 4.16 45.555 90.74 481.04</td>
<td>210.48 10.943 0 0</td>
</tr>
</tbody>
</table>

### Table 3.11: Large-size problems with well-determined systems ($\gamma = 1$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NG  CF</td>
<td>QN  CF</td>
</tr>
<tr>
<td></td>
<td>IT  CPU</td>
<td>IT  CPU</td>
</tr>
<tr>
<td>200 200</td>
<td>3.74 4.1472 3.72 1.7995 51.46 13.604</td>
<td>45.26 0.3194 0 0</td>
</tr>
<tr>
<td>300 300</td>
<td>3.42 13.422 3.42 5.6252 56.64 45.164</td>
<td>59.38 1.0098 0 0</td>
</tr>
<tr>
<td>400 400</td>
<td>3.89 36.334 3.89 14.342 68.48 121.21</td>
<td>69.44 2.2716 0 0</td>
</tr>
<tr>
<td>500 500</td>
<td>4.36 78.168 4.34 28.652 86.24 275.16</td>
<td>82.56 4.2224 0 0</td>
</tr>
<tr>
<td>600 600</td>
<td>4.16 127.38 4.16 45.555 90.74 481.04</td>
<td>91.06 7.1202 0 0</td>
</tr>
</tbody>
</table>

### Table 3.12: Large-size problems with over-determined systems ($\gamma = 1$)

<table>
<thead>
<tr>
<th>Method</th>
<th>Entropic</th>
<th>Surrogate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NG  CF</td>
<td>QN  CF</td>
</tr>
<tr>
<td></td>
<td>IT  CPU</td>
<td>IT  CPU</td>
</tr>
<tr>
<td>100 200</td>
<td>6.60 5.0130 6.86 2.2192 137.38 37.361</td>
<td>446.02 19.442 14 3.06E-3</td>
</tr>
<tr>
<td>150 300</td>
<td>7.10 18.196 7.12 5.0082 191.88 149.76</td>
<td>5035.22 31.624 23 3.06E-3</td>
</tr>
<tr>
<td>200 400</td>
<td>7.12 45.587 7.12 13.341 244.56 416.89</td>
<td>4812.57 41.970 36 1.02E-2</td>
</tr>
<tr>
<td>250 500</td>
<td>7.74 93.563 7.74 25.588 361.44 962.17</td>
<td>5590.88 68.860 42 1.83E-2</td>
</tr>
<tr>
<td>300 600</td>
<td>7.76 160.91 7.76 40.860 348.32 1833.7</td>
<td>6613.33 104.89 44 3.53E-2</td>
</tr>
</tbody>
</table>
### Table 3.13: EP method results with distance shift

<table>
<thead>
<tr>
<th>size</th>
<th>shift</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>n m</td>
<td></td>
<td>IT</td>
<td>CPU</td>
<td>IT</td>
<td>CPU</td>
<td>IT</td>
</tr>
<tr>
<td>10 20</td>
<td>4.64</td>
<td>0.0209</td>
<td>4.93</td>
<td>0.0240</td>
<td>5.17</td>
<td>0.0207</td>
</tr>
<tr>
<td>20 40</td>
<td>5.35</td>
<td>0.0429</td>
<td>5.53</td>
<td>0.0514</td>
<td>5.69</td>
<td>0.0498</td>
</tr>
<tr>
<td>30 60</td>
<td>5.45</td>
<td>0.0762</td>
<td>5.62</td>
<td>0.0796</td>
<td>5.84</td>
<td>0.0824</td>
</tr>
<tr>
<td>40 80</td>
<td>5.49</td>
<td>0.1172</td>
<td>5.60</td>
<td>0.1203</td>
<td>5.82</td>
<td>0.1222</td>
</tr>
<tr>
<td>50 100</td>
<td>5.53</td>
<td>0.1687</td>
<td>5.68</td>
<td>0.1761</td>
<td>5.88</td>
<td>0.1820</td>
</tr>
<tr>
<td>100 200</td>
<td>5.82</td>
<td>0.9102</td>
<td>5.98</td>
<td>0.9248</td>
<td>6.02</td>
<td>0.9312</td>
</tr>
<tr>
<td>150 300</td>
<td>5.78</td>
<td>3.6926</td>
<td>6.00</td>
<td>3.8140</td>
<td>6.20</td>
<td>3.9578</td>
</tr>
<tr>
<td>300 600</td>
<td>6.04</td>
<td>30.419</td>
<td>6.28</td>
<td>31.966</td>
<td>6.46</td>
<td>32.510</td>
</tr>
</tbody>
</table>

### Table 3.14: SC method results with distance shift

<table>
<thead>
<tr>
<th>size</th>
<th>shift</th>
<th>0.5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>n m</td>
<td></td>
<td>IT</td>
<td>CPU</td>
</tr>
<tr>
<td>10 20</td>
<td></td>
<td>357.45</td>
<td>0.0976</td>
</tr>
<tr>
<td>20 40</td>
<td></td>
<td>838.51</td>
<td>0.2746</td>
</tr>
<tr>
<td>30 60</td>
<td></td>
<td>1153.4</td>
<td>0.4297</td>
</tr>
<tr>
<td>40 80</td>
<td></td>
<td>1421.0</td>
<td>0.7223</td>
</tr>
<tr>
<td>50 100</td>
<td></td>
<td>886.91</td>
<td>0.5642</td>
</tr>
<tr>
<td>100 200</td>
<td></td>
<td>2096.8</td>
<td>4.3044</td>
</tr>
<tr>
<td>150 300</td>
<td></td>
<td>1925.1</td>
<td>8.9505</td>
</tr>
<tr>
<td>200 400</td>
<td></td>
<td>3208.2</td>
<td>27.412</td>
</tr>
<tr>
<td>250 500</td>
<td></td>
<td>3314.8</td>
<td>41.064</td>
</tr>
<tr>
<td>300 600</td>
<td></td>
<td>5837.1</td>
<td>90.971</td>
</tr>
</tbody>
</table>

34
### Table 3.15: SC method results with distance shift (continued)

<table>
<thead>
<tr>
<th>size \ shift</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>n m</td>
<td>IT CPU UF AER WST</td>
<td>IT CPU UF AER WST</td>
</tr>
<tr>
<td>10 20</td>
<td>262.86 0.0790 15 3.065 6.0433</td>
<td>515.07 0.1390 16 0.007 5.9625</td>
</tr>
<tr>
<td>20 40</td>
<td>1472.4 0.4839 14 0.0004 7.4464</td>
<td>1108.4 0.3651 26 0.0023 7.5865</td>
</tr>
<tr>
<td>30 60</td>
<td>1244.0 0.4697 34 0.0106 8.4250</td>
<td>1069.4 0.4041 35 0.0331 8.8714</td>
</tr>
<tr>
<td>40 80</td>
<td>1514.0 0.7743 39 0.0015 11.686</td>
<td>1012.0 0.5476 49 0.0087 11.991</td>
</tr>
<tr>
<td>50 100</td>
<td>941.54 0.0274 31 0.0134 14.889</td>
<td>859.35 0.5092 34 0.0460 15.446</td>
</tr>
<tr>
<td>100 200</td>
<td>1646.3 2.9051 13 0.0229 48.744</td>
<td>1902.6 3.7658 14 0.0994 48.194</td>
</tr>
<tr>
<td>150 300</td>
<td>1675.2 8.4413 12 0.0612 13.410</td>
<td>1499.5 7.4416 13 0.1705 136.43</td>
</tr>
<tr>
<td>200 400</td>
<td>1783.2 14.132 18 0.0641 270.86</td>
<td>1820.0 13.598 19 0.1432 266.95</td>
</tr>
<tr>
<td>250 500</td>
<td>2988.1 38.646 17 0.1090 422.52</td>
<td>2984.5 40.993 20 0.2483 398.34</td>
</tr>
<tr>
<td>300 600</td>
<td>2876.2 46.020 21 0.1891 600.28</td>
<td>3233.6 51.005 22 0.4603 609.93</td>
</tr>
</tbody>
</table>

### Table 3.16: SC method results with distance shift (continued)

<table>
<thead>
<tr>
<th>size \ shift</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>n m</td>
<td>IT CPU UF AER WST</td>
</tr>
<tr>
<td>10 20</td>
<td>376.07 0.1023 23 0.0079 6.1791</td>
</tr>
<tr>
<td>20 40</td>
<td>912.63 0.3037 28 0.0226 7.6725</td>
</tr>
<tr>
<td>30 60</td>
<td>812.45 0.3128 41 0.0876 8.6583</td>
</tr>
<tr>
<td>40 80</td>
<td>1047.8 0.5708 47 0.0570 11.963</td>
</tr>
<tr>
<td>50 100</td>
<td>692.94 0.4593 40 0.1270 15.679</td>
</tr>
<tr>
<td>100 200</td>
<td>1455.4 2.9570 20 0.1588 47.922</td>
</tr>
<tr>
<td>150 300</td>
<td>1362.9 6.5644 14 0.3446 142.53</td>
</tr>
<tr>
<td>200 400</td>
<td>1327.6 11.842 20 0.3050 264.19</td>
</tr>
<tr>
<td>250 500</td>
<td>2430.5 31.519 21 0.5707 409.22</td>
</tr>
<tr>
<td>300 600</td>
<td>3889.8 59.237 22 1.1520 609.51</td>
</tr>
</tbody>
</table>
Figure 3.1: total required iterations for under-determined systems $(n/m = 2)$

Figure 3.2: required CPU time for under-determined systems $(n/m = 2)$
Figure 3.3: total required iterations for well-determined systems ($n/m = 1$)

Figure 3.4: required CPU time for well-determined systems ($n/m = 1$)
Figure 3.5: total required iterations for over-determined systems ($n/m = 1/2$)

Figure 3.6: required CPU time for over-determined systems ($n/m = 1/2$)
Chapter 4

Systems of Infinitely Many Linear Inequalities

The problem studied in this Chapter is that of finding a feasible solution for the following semi-infinite linear inequality system (SLIS):

\[
\begin{align*}
& f_1(t)^T x \leq g_1(t), \\
& \vdots \\
& f_m(t)^T x \leq g_m(t), \\
& x \geq 0,
\end{align*}
\]  

(4.1)

where \( x \in \mathbb{R}^n \), \( T \) is a convex compact set in \( \mathbb{R}^l \) with \( |T| = \infty \) and \( l \in \mathbb{N} \), \( f_j(t) = (f_{j1}(t), f_{j2}(t), \ldots, f_{jn}(t))^T \) for \( j = 1, \ldots, m \), and \( f_{j1}, f_{j2}, \ldots, f_{jn} \) and \( g_j \), for \( j = 1, \ldots, m \), are real-valued continuous functions on \( T \). We denote the solution set of (4.1) as \( \Omega \). As assumed by most interior-point methods, we assume that system (4.1) has an interior feasible solution \( x_0 \in int(\Omega) \), and without loss of generality, we also assume \( \|f_j(t)\| > 0, \forall t \in T, j = 1, \ldots, m \). From the assumptions, we know that the solution set \( \Omega \) is a nonempty closed convex set.

Notice that (4.1) can be converted to a semi-infinite linear programming problem with zero objective function and solved by semi-infinite programming algorithms, such as exchange method and discretization method [35][62]. However, those algorithms are not usually efficient for large size problems.

Recently, a cutting plane method based on the “analytic center” concept was proposed by Goffin et al. [24][25] to study convex feasibility problems. In this chapter,
we extend the idea of the analytic center cutting plane method (ACCMP) to solve (4.1).

In section 4.1, we give a brief review for the theoretical foundation of analytic center cutting plane method. The proposed approach is presented in Section 4.2 with some numerical examples. We discuss the feasibility issue of a special class of SLIS, namely, maximum feasibility problem, in Section 4.3. An algorithm is proposed and illustrated with some numerical examples.

4.1 Cutting Plane Method Based on Centers

In this section, the theoretical foundation of the analytic center cutting plane method is briefly presented. Throughout this section, $A$ is a $m \times n$ real matrix, $a$ is an $n$-dimensional vector and $b$ is an $m$-dimensional vector.

4.1.1 Cutting Plane Method

Cutting plane methods usually require the knowledge of a compact set (a polytope) which encloses the feasible region ($\Omega$) of a given problem. One often refers to the computing mechanism of a cutting plane method as the oracle. Given a sequence of test points, \{${x^k}$\}, the oracle produces cutting planes that provide a polyhedral outer approximation to $\Omega$. To be more precise, for any given $x^k \in R^n$, the oracle either responds that $x^k \in \Omega$ or returns the coefficient vector $a$ for a hyperplane \{$x \in R^n | a^T x = a^T x^k$\} such that $\Omega \subset \{x \in R^n | a^T x \leq a^T x^k$\}. In the latter case, a separating plane will be placed through $x^k$ in order to shrink the containing polytope.

The oracle can return either one cut or multiple cuts for each “query”. Notice that the cuts generated above are linear cuts. Research on using nonlinear cuts (or cutting surfaces) can be found in [47][48]. However, the issue of type of cut is beyond the scope of this dissertation. We only consider linear cuts for our linear problems.

A cut at the query point $\hat{x}$ can be taken in the form of

$$a^T x \leq a^T \hat{x} - \gamma.$$ 

If $\gamma = 0$, the cut is called a “central cut”, i.e., the cut passes through the query point $\hat{x}$; if $\gamma > 0$, the cut is “deep”, i.e. it cuts off the query point; if $\gamma < 0$, the cut is “shallow".
The main issue in using cutting plane methods is how to define the sequence of test points, \( \{x^k\} \). Ideally, placing a cutting plane through the “center” of a polytope, as opposed to through a boundary point, can divide the polytope into two approximately “equal” parts, such that we can have a shrinking rate about 1/2.

A variety of centering methods have been proposed to generate the test points. This list includes the center of gravity, the center of the maximum volume ellipsoid and the volumetric center [28]. A recent addition to the list is the analytic center cutting plane method (ACCMP).

### 4.1.2 Analytic Center

**Definition 1** \( x^* \) is the “analytic center” (AC) of a bounded convex polyhedral set \( \bar{\Omega} = \{x \in R^n | Ax \leq b\} \) with \( \text{int}(\bar{\Omega}) \neq \emptyset \) if it is the unique maximizer of the following optimization problem [72]:

\[
\begin{align*}
\max & \quad \sum_{j=1}^{m} \ln s_j \\
\text{s.t.} & \quad Ax + s = b \\
& \quad s \geq 0.
\end{align*}
\] (4.2)

The analytic center is a function of the data describing the polytope. The function depends on the analytic representation of \( \bar{\Omega} \), not just the set. Thus its sensitivity to changes in the data (added/deleted cuts) can be predicted. Since the objective function of (4.2) is strictly concave, \( x^* \) is uniquely defined by the following optimality conditions:

\[
\begin{align*}
Ax + s &= b \\
A^T w &= 0 \\
Ws &= e \\
w \geq 0, \quad s \geq 0,
\end{align*}
\] (4.3)

where \( w \in R^m \) are the dual variables and \( W = \text{diag}(w) \). Several algorithms have been proposed for computing analytic centers. These include the dual (or primal or primal-dual) Newton procedure and dual (or primal or primal-dual) potential algorithm [72].

It can be computationally expensive to calculate a new analytic center. Some update procedures have been proposed to construct a new analytic center from the current analytic center when cuts are added into \( \bar{\Omega} \). Efficient update procedures can be found in [70] (for single shallow cut), [25] (for single central cut), and [29] (for multiple central cuts).
4.1.3 Analytic Center Cutting Plane Method

The ACCPM combines a highly efficient interior point method with a classical cutting plane scheme. The theory of ACCPM has been studied in depth including the convergence properties and complexity estimates. A single shallow cut scheme for solving linear feasibility problems was proposed by Ye [70], while Goffin et al. considered a single central cut method for solving convex feasibility problems [25]. The ACCPM has also been implemented successfully for solving a wide variety of problems in addition to feasibility problems [28].

In order to estimate the convergence and complexity analysis of ACCPM for solving convex feasibility problems, without loss of generality, three assumptions are commonly required [25]:

Assumption 1. \( \Omega \subset \Omega^0 \triangleq \{ x \in \mathbb{R}^n | 0 \leq x \leq e \} \).

Assumption 2. \( \text{int}(\Omega) \neq \emptyset \) and there is a full dimensional closed ball, \( B_e \), with radius \( \epsilon < \frac{1}{2} \) such that \( B_e \subset \text{int}(\Omega) \).

Assumption 3. There exists an oracle which for each \( \bar{x} \in \Omega^0 \) either returns that \( \bar{x} \in \Omega \) or generates a cut \( \{ x \in \mathbb{R}^n | a^T x = a^T \bar{x} \} \) with \( \| a \| = 1 \) such that \( \Omega \subset \{ x \in \mathbb{R}^n | a^T x \leq a^T \bar{x} \} \).

The ACCPM for solving convex feasibility problems can be stated as follows.

**Initialization**

Let \( \Omega^0 = \{ x \in \mathbb{R}^n | 0 \leq x \leq e \} \), \( x^0 = \frac{e}{2} \) be the center of \( \Omega^0 \) and \( k = 0 \).

\[
(A^0 = \begin{bmatrix} I \\ -I \end{bmatrix} \text{ and } b^0 = \begin{bmatrix} e \\ 0 \end{bmatrix})
\]

**Basic Step**

Step 1: Query the oracle to see if \( x^k \in \Omega \).

Step 2: If yes, stop. Otherwise, the oracle returns \( a_k \).

Step 3: Update \( \Omega^{k+1} = \Omega^k \cap \{ x \in \mathbb{R}^n | a_k^T x \leq a_k^T x^k \} \).

\[
(A^{k+1} = \begin{bmatrix} A^k \\ a_k^T \\ a_k^T x^k \end{bmatrix} \text{ and } b^{k+1} = \begin{bmatrix} b^k \\ a_k^T x^k \end{bmatrix})
\]

Step 4: Compute the analytic center of \( \Omega^{k+1} \).

Step 5: Set \( k = k + 1 \) and return to Step 1.
Under the above three assumptions, defining the potential function at analytic center \( x^k \) as
\[
P(\Omega^k) = \sum_{j=1}^{2n+k} \ln(b^j_k - A^k_j x^k) = \sum_{j=1}^{2n+k} \ln s^j_k,
\]
where \( A^k_j \) is the \( j^{\text{th}} \) row vector of \( A^k \), Ye [70], and Goffin et al. [25] showed the following properties for the sequence of \( \{\Omega^k\} \):

**Property 1** \( \Omega \subset \Omega^k \), \( k = 1, 2, \ldots \).

**Property 2** For \( x \in \Omega^k \), if \( \bar{s}_j = b^j_k - A^k_j x \), \( j = 1, \ldots, 2n+k \), then \( 0 \leq \bar{s}_j \leq 1 \), \( j = 1, \ldots, 2n \) and \( 0 \leq \bar{s}_j \leq \sqrt{n} \), \( j = 2n+1, \ldots, 2n+k \).

**Property 3** \( P(\Omega^k) \geq (2n + k) \ln \epsilon \), \( k = 1, 2, \ldots \).

**Property 4** \( P(\Omega^{k+1}) \leq P(\Omega^k) + \frac{1}{2} \ln \left[ a^T_{k+1}((A^j)^T(S^j)^{-2}(A^j))^{-1}a_{k+1} \right] - (1.5 - \ln 4) \), where \( S^k = \text{diag}(s^k) \).

It follows that
\[
\begin{align*}
\ln \epsilon & \leq (2n + k + 1)^{-1} P(\Omega^{k+1}) \\
& \leq (2n + k + 1)^{-1} \left\{ P(\Omega^0) + \frac{1}{2} \sum_{j=0}^{k} \ln \left[ a^T_{j+1}((A^j)^T(S^j)^{-2}(A^j))^{-1}a_{j+1} \right] \right\} \\
& \leq (2n + k + 1)^{-1} \left\{ 2n \ln \frac{1}{2} + \frac{1}{2} \sum_{j=0}^{k} \ln \left[ a^T_{j+1}((A^j)^T(S^j)^{-2}(A^j))^{-1}a_{j+1} \right] \right\} \quad (4.4)
\end{align*}
\]

Nesterov [53] gave a constructive proof to show the following bound:
\[
\sum_{j=0}^{k} a^T_{k+1}((A^j)^T(S^j)^{-2}(A^j))^{-1}a_{k+1} \leq 2n^2 \ln(1 + \frac{k+1}{8n^2}). \quad (4.5)
\]

By using the concavity of \( \ln \) and combining (4.4) and (4.5), we have
\[
\frac{2n + k}{\frac{1}{2} + 2n \ln(1 + \frac{k}{8n^2})} \leq n \frac{1}{\epsilon^2}, \quad \text{for} \ k = 1, 2, \ldots \quad (4.6)
\]

This implies that the ACCPM stops as soon as \( k \) satisfies
\[
\frac{2n + k}{\frac{1}{2} + 2n \ln(1 + \frac{k}{8n^2})} \geq n \frac{1}{\epsilon^2}.
\]
This result not only demonstrates the convergence of ACCPM, but also allows us to determine its computational complexity. By considering the $O(1)$ steps needed for the center update procedure [70][25] as well as the bound on the number of calls to the oracle, we see that ACCPM for solving convex feasibility problems is a polynomial-time algorithm.

4.2 Algorithms for Solving SLIS

In this section, a cutting plane algorithm based on the analytic center is proposed for solving the semi-infinite linear inequality system (4.1).

4.2.1 ACCPM for Solving SLIS

Notice that, in general, the solution set $\Omega$ of (4.1) is a closed convex set, but it might be unbounded. In this case, we assume that there is a sufficiently large $M$ such that $\text{int}(\Omega) \cap \{x \in R^n|0 \leq x \leq Me\} \neq \emptyset$, where $e$ is the vector of all ones. Then the ACCPM can be applied to solve SLIS (4.1).

In order to apply ACCPM to solve (4.1) and to establish the convergence and the complexity properties, two assumptions are required [25]:

Assumption 1. $\Omega \subset \{x \in R^n|0 \leq x \leq e\}$, where $e$ is the vector of all ones.

Assumption 2. $\text{int}(\Omega) \neq \emptyset$ and there is a full dimensional closed ball, $B_\epsilon$, with radius $\epsilon < \frac{1}{2}$ such that $B_\epsilon \subset \text{int}(\Omega)$.

We outline an algorithm for solving (4.1).
Algorithm 3: ACCPM for solving SLIS

Step 0: Set $k = 0$.

Let $A^k = \begin{bmatrix} I \\ -I \end{bmatrix}$, $b^k = \begin{bmatrix} e \\ 0 \end{bmatrix}$ and $x^k = \frac{1}{2}e$.

Step 1: For $j = 1, ..., m$, compute $\delta_j = \max_{t \in T} \{f_j(t)^T x^k - g_j(t)\}$ and $t_j^k = \arg \max_{t \in T} \{f_j(t)^T x^k - g_j(t)\}$.

Step 2: If no $\delta_j > 0$, $j = 1, ..., m$, then stop. $x^k$ is a feasible solution to SLIS.

Otherwise,

Step 3: Let $\bar{A}^k = \begin{bmatrix} f_j(t_j^k)^T \\ \|f_j(t_j^k)\| \end{bmatrix}$ and $\bar{b}^k = \begin{bmatrix} f_j(t_j^k)^T x^k \\ \|f_j(t_j^k)\| \end{bmatrix}$ for $\delta_j > 0$, $j = 1, ..., m$.

Set $A^{k+1} = \begin{bmatrix} A^k \\ \bar{A}^k \end{bmatrix}$, $b^{k+1} = \begin{bmatrix} b^k \\ \bar{b}^k \end{bmatrix}$ and $k = k + 1$.

Step 4: Find the analytic center $x^k$ of $\Omega^k = \{x \in R^n | A^k x \leq b^k\}$.

Return to Step 1.

Notice that Algorithm 3 is a multiple cutting plane (Step 3) algorithm based on the analytic center. In order to make sure that the algorithm works, we need to show that the oracle (Steps 1 - 3) is well-defined. Obviously, from Algorithm 3, the cuts returned by the oracle are defined by the most violated constraints. It is necessary to show that $\Omega$ always lies on the correct side of the cutting planes.

Lemma 2 $\Omega \subset \{x \in R^n | \bar{A}^k x \leq \bar{b}^k\}$ for all $k$.

Proof. For any $x \in \Omega$, we have $a_j(t_j^k)^T x - b_j(t_j^k) \leq 0$, for $j = 1, ..., m$. From Step 3 of Algorithm 3, we know that $a_j(t_j^k)^T x^k - b_j(t_j^k) \geq 0$, for $j = 1, ..., m$. Consequently $a_j(t_j^k)^T x \leq a_j(t_j^k)^T x^k$, for $j = 1, ..., m$, i.e., $\Omega \subset \{x \in R^n | \bar{A}^k x \leq \bar{b}^k\}$.

Notice that, at each iteration the algorithm provides multiple central cuts (at most $m$ cuts) simultaneously. Under assumptions 1 and 2, Ye provides a complexity estimate for Algorithm 3 (Theorem 10 of [71]), namely,
Theorem 5 Algorithm 3 terminates in at most \( k \) iterations, where \( k \) is the smallest integer satisfying the following inequality

\[
\frac{\epsilon^2}{(m+1)^2} \geq \frac{n}{2} + \frac{18n^2}{15} \ln\left(1 + \frac{m(k+1)}{8n^2}\right).
\]

While Algorithm 3 will converge in a finite number of steps, in practice there are two issues which need to be considered relative to speed of convergence: (1) the cutting scheme, i.e., central cuts vs. deep cuts; and (2) the optimizer in Step 1.

In practice, the deeper the cut, the more the polytope \( \Omega^k \) shrinks, and the faster the cutting plane method converges. The cuts generated by Algorithm 3 are central cuts. However they can be replaced with deeper cuts. Instead of using a central cut in the form of \( f_j(t^k_j)^T x \leq f_j(t^k_j)^T x^k \), the most violated constraint \( f_j(t^k_j)^T x - g_j(t^k_j) \leq 0 \) can be added into \( \Omega^k \) as the cut, \( f_j(t^k_j)^T x \leq f_j(t^k_j)^T x^k - \gamma_j^k \) where \( \gamma_j^k = f_j(t^k_j)^T x - g_j(t^k_j)^T x^k > 0 \). In this case, the algorithm is similar (in the dual sense) to the column-generation method for solving LP. However, in this multiple deep cut scheme, it is impossible to have an efficient procedure for updating centers, and the complexity analysis breaks down.

Notice that, in Step 1 we need a global optimization procedure for identifying the most violated constraint(s) for a current solution \( x^k \). It is not always possible to have an efficient algorithm for this purpose. In implementation, usually, the set \( T \) is replaced by a finite (grid) subset \( \tilde{T} \) of \( T \). The finer partition \( \tilde{T} \) is, the better optimizer we have but at the cost of more computation. In order to avoid excessive computational burden, we may also adopt the idea, proposed by Wu et al. [68], of relaxing the requirement of finding the cuts. In their procedure, a new cut is found at \( \tilde{t} \in T \) such that \( f_j(\tilde{t})^T x - g_j(\tilde{t}) > \delta \), where \( \delta > 0 \) is a sufficiently small number.

Our relaxed cut scheme is a search procedure built on a sequence of subsets \( T_i \subset \tilde{T} \), \( i = 1, \ldots, K \), with \( T_i \subset T_{i+1} \) and \( T_K = \tilde{T} \). We also require the relative size \( \frac{|T_i|}{|T_{i+1}|} << 1 \). For example, in our implementation, \( \frac{|T_1|}{|T_{i+1}|} = 0.1 \). A new cut is identified by the most violated constraint for all \( t \in T_1 \). If there is a violated constraint in set \( T_1 \), then the search procedure stops. If there is no violated constraint in \( T_1 \), then we search for the most violated constraint in \( T_2 \). If there is no violated constraint in \( T_2 \), then we search in \( T_3 \). This search procedure continues until \( T_i = T_K = \tilde{T} \).

At first glance, the requirement for finding the cuts may seem to be over-relaxed. However, notice that, the contribution of any violated constraint to the centering
process is roughly the same, i.e. the most violated constraint is not necessarily needed in ACCPM. This is the main advantage of ACCPM over the projection method [3] for solving SLIS.

### 4.2.2 EACCPM for Solving SLIS

Based on the ideas embodied in the algorithm for solving systems of linear inequalities proposed in Chapter 3 (EP Algorithm), we define an “entropic analytic center” (EAC) for a convex polyhedral set.

**Definition 2** Let \( \Omega_0 = \{ x \in \mathbb{R}^n | Ax \leq b, x \geq 0 \} \) with the assumption that \( \text{int}(\Omega_0) \neq \emptyset \). If \( x^* \) is the unique minimizer of the linearly constrained minimum entropy optimization problem,

\[
\min \sum_{i=1}^{n} x_i \ln x_i + \sum_{j=1}^{m} s_j \ln s_j \\
\text{s.t.} \quad Ax + s = b \\
\quad x \geq 0, s \geq 0,
\]

then \( x^* \) is the entropic analytic center of \( \Omega_0 \).

Unlike the analytic center, the entropic analytic center does not require \( \Omega_0 \) to be bounded. Since the objective function of (4.7) is an entropic function, we refer to the minimizer of (4.7) as the entropic analytic center. Solving (4.7) is equivalent to solving the following unconstrained (Dual) optimization problem:

\[
\max_{w \in \mathbb{R}^m} \left\{ g(w) = b^T w - \sum_{k=1}^{r} \exp \left\{ \hat{A}^T_k w - 1 \right\} \right\},
\]

where \( \hat{A} = [A|I_m] \) and \( \hat{A}_k \) denotes the \( k^{th} \) column vector of matrix \( \hat{A} \). The EP Algorithm proposed in Chapter 3 can be modified to obtain the entropic analytic center.

We now propose to replace the analytic center with the entropic analytic center in Algorithm 3 to create an entropic analytic center cutting plane method (EACCPM) in Algorithm 4. The procedure for computing the entropic analytic center can also detect inconsistency of a linear inequality system. This allows Algorithm 4 to detect the possible inconsistency in SLIS. However, to date we have not obtained a convergence proof nor determined the computational complexity of the algorithm.
Algorithm 4: EACC2PM for solving SLIS

Step 0: Set $k = 0$. Let $A^0 = \emptyset$, $b^0 = \emptyset$ and $x^k = \exp(-1)$.

Step 1: For $j = 1, \ldots, m$, compute $\delta_j = \max_{t \in T} \{f_j(t)^T x^k - g_j(t)\}$ and

$$t^k_j = \arg \max_{t \in T} \{f_j(t)^T x^k - g_j(t)\}.$$

Step 2: If no $\delta_j > 0$, $j = 1, \ldots, m$, then stop. $x^k$ is a solution. Otherwise,

Step 3: Let $\bar{A}^k = \left[ \frac{f_j(t^k_j)^T}{\|f_j(t^k_j)\|} \right]$ and $\bar{b}^k = \left[ \frac{f_j(t^k_j)^T x^k}{\|f_j(t^k_j)\|} \right]$ for $\delta_j > 0$, $j = 1, \ldots, m$.

Set $A^{k+1} = \left[ \begin{array}{c} A^k \\ \bar{A}^k \end{array} \right]$, $b^{k+1} = \left[ \begin{array}{c} t^k \\ \bar{b}^k \end{array} \right]$ and $k = k + 1$.

Step 4: Find the entropic analytic center $x^k$ of $\Omega^k = \{ x \mid A^k x \leq b^k \}$.

Step 5: If there is no entropic analytic center then stop. There is no solution for this system. Otherwise, return to Step 1.

The possible computational refinements discussed in the previous subsection are addressed in the next subsection.

4.2.3 Numerical Examples

In this subsection, two semi-infinite linear inequality systems are used to illustrate the computational behavior of the proposed algorithms. We use MATLAB Version 6.0 running on a Pentium III 700MHZ computer with 256 MB memory.
Example 1.

\[
\begin{align*}
-x_1 - 4(t + 1)x_2 - t^2 + 1 & \leq 0, \quad \forall \ t \in [-2, 2] \\
2(t + 1)x_1 - x_2 - t^2 + 0.5 & \leq 0 \\
0 & \leq x \leq 1.
\end{align*}
\]

(4.9)

Basically we test both ACCPM and EACCPM. The “dual Newton procedure” [72] is adopted for the ACCPM implementation but a center update procedure is not used. Rather, for each iteration, the analytic center is calculated directly. In each algorithm, we implement two kinds of cutting schemes, central cut and deep cut. Within each scheme, three different approaches, namely, the global optimization method, the discretization method and the proposed relaxed cut method, are used for identifying cuts in Step 1, providing 6 alternative algorithms.

For the global optimization method, the FMINBND subroutine of MATLAB is used for finding the global maximizer in Step 1. For the discretization method, the interval \((-2 \leq t \leq 2\) is discretized using 1001 evenly spaced points. If for the current solution \(x^k\) there is no violated constraint at any of these 1001 points, we take \(x^k\) as a solution of (4.9).

We briefly describe a recursive procedure used to identify the cuts for the proposed relaxed cut method. To be more precise, we associate two functions, \(f_1(t)\) and \(f_2(t)\), with the first and second inequalities of (4.9), respectively. Initially, the interval \([-2, 2]\) is evenly partitioned into 10 subintervals, i.e., there are 11 points in \(T_1 \triangleq \{t|t = -2 + 0.4 \times k, k = 0, \ldots, 10\}\). For the current \(x^k\), we test each point in \(T_1\) to find the most violated constraints for inequalities \(f_1\) and \(f_2\), respectively. A new cut is found at \(t^k_i = \arg \max_{t \in T_1} \{f_i(t)\}\) provide if \(f_i(t^k_i) > 0, \ i = 1\) or \(2\). If there is a violated constraint then the procedure stops and returns cut(s). If in set \(T_1\) there is no violated constraint for both inequalities \(f_1\) and \(f_2\), we refine the partition into 100 equal-length subintervals and \(T_2 \triangleq \{t|t = -2 + 0.04 \times k, k = 0, \ldots, 100\}\). If there is no violated constraint in set \(T_2\), then we refined the partition again and \(T_3 \triangleq \{t|t = -2 + 0.004 \times k, k = 0, \ldots, 1000\}\). The algorithm stops if there is no violated constraint in \(T_3\). Notice that \(T_3\) is the same setting as in the discretization method.

The computational results are summarized in Tables 4.1 and 4.2. In the table, \textbf{GO} denotes the global optimization method; \textbf{DM}, the discretization method; and \textbf{RM}, the proposed relaxed cut method. \textbf{CPU} is the time (seconds) required for the
algorithm to terminate; IT, the number of iterations required; and CUT, the total number of cuts generated.

### Table 4.1: ACCPM results for example 1

<table>
<thead>
<tr>
<th></th>
<th>Central Cut</th>
<th>Deep Cut</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GO</td>
<td>DM</td>
<td>RM</td>
<td>GO</td>
<td>DM</td>
</tr>
<tr>
<td>CPU</td>
<td>0.1800</td>
<td>0.1300</td>
<td>0.0700</td>
<td>0.1410</td>
<td>0.0700</td>
</tr>
<tr>
<td>IT</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CUT</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Table 4.2: EACCPM results for example 1

<table>
<thead>
<tr>
<th></th>
<th>Central Cut</th>
<th>Deep Cut</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GO</td>
<td>DM</td>
<td>RM</td>
<td>GO</td>
<td>DM</td>
</tr>
<tr>
<td>CPU</td>
<td>0.1900</td>
<td>0.1310</td>
<td>0.1500</td>
<td>0.0800</td>
<td>0.0500</td>
</tr>
<tr>
<td>IT</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CUT</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Example 2.

\[
\begin{align*}
\sin(t) \cos(s)x_1 + \sin(t) \sin(s)x_2 + \cos(t)x_3 &< 0.1 \\
-0.9 \sin(t) \cos(s) - 0.7 \sin(t) \sin(s) - 0.3 \cos(t) &\leq 0, \\
\sin(t) \cos(s)x_1 + \sin(t) \sin(s)x_2 + \cos(t)x_3 &< 0.06 \\
-0.8 \sin(t) \cos(s) - 0.7 \sin(t) \sin(s) - 0.4 \cos(t) &\leq 0, \\
\forall s \in [0, 2\pi], \forall t \in [0, \pi], &\ 0 \leq x \leq 1.
\end{align*}
\]

(4.10)

In this example, the feasible set for each of the inequalities (4.10) forms a 3-dimensional ball, \( B_r(\bar{x}) \), with radius \( r \) and center \( \bar{x} \), i.e., the solution set of (4.10) is \( \Omega = B_{0.1}((0.9, 0.7, 0.3)^T) \cap B_{0.06}((0.8, 0.7, 0.4)^T) \).

We did not implement the global optimization method due to the amount of computation that could have been required.

For the discretization method, at each iteration the \( s \)-interval \([0, 2\pi]\) is discretized using 1001 evenly spaced points and the \( t \)-interval \([0, \pi]\) using 501 evenly spaced points, i.e., there are total 501501 grid points in the \((s, t)\) domain.
For the relaxed cut method, we use a recursive procedure similar to the one presented in example 1 to identify the violated constraints. To be more precise, we define \( T_1 = \{(s, t) | (s, t) = \frac{\pi}{5}(j, k); j = 0, \ldots, 10; k = 0, \ldots, 5\}, T_2 = \{(s, t) | (s, t) = \frac{\pi}{50}(j, k); j = 0, \ldots, 100; k = 0, \ldots, 50\} \) and \( T_3 = \{(s, t) | (s, t) = \frac{\pi}{500}(j, k); j = 0, \ldots, 1000; k = 0, \ldots, 500\} \). The results are summarized in Tables 4.3 and 4.4.

<table>
<thead>
<tr>
<th></th>
<th>Central Cut</th>
<th>Deep Cut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DM</td>
<td>RM</td>
</tr>
<tr>
<td>CPU</td>
<td>25.146</td>
<td>2.884</td>
</tr>
<tr>
<td>IT</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>CUT</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 4.3: ACCPM results for example 2

<table>
<thead>
<tr>
<th></th>
<th>Central Cut</th>
<th>Deep Cut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DM</td>
<td>RM</td>
</tr>
<tr>
<td>CPU</td>
<td>27.604</td>
<td>2.884</td>
</tr>
<tr>
<td>IT</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>CUT</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 4.4: EACCPM results for example 2

Some observations can be made here:

(i) There is no big difference in computational effort required between using ACCPM and EACCPM to solve these two examples.

(ii) As we expected, the deep cut scheme performs better than central cut scheme, but the difference is not big.

(iii) The impact of the proposed relaxed cut method in example 1 is not obvious. However, in the second example the CPU time reduction is very significant while the IT and CUT figures do not increase very much.
4.3 Maximum Feasibility Problem

Consider a linear semi-infinite system:

\[
\begin{cases}
  f(t)^T x \leq g(t), \quad \forall t \in [0, 1] \\
  x \geq 0,
\end{cases}
\]  

(4.11)

where \( x \in \mathbb{R}^n \) and, \( f(t) = (f_1(t), f_2(t), \ldots, f_n(t))^T \) and \( g(t) \) are real-valued continuous functions on \([0, 1]\). We denote \( \Omega \) as the solution set of (4.11). Notice that (4.11) can be converted to a linear semi-infinite programming problem with zero objective function and solved by semi-infinite programming algorithms. Most solution procedures for (4.11) (or linear semi-infinite programs) assume that \( \Omega \neq \emptyset \) (or \( \text{int}(\Omega) \neq \emptyset \)).

In practice, unfortunately, one usually doesn’t know in advance whether \( \Omega \) is empty. If \( \Omega \) is empty, it is natural to consider the question of what is the largest portion of \( t \)-interval \(([0, 1])\) for which system (4.11) is consistent. Specifically, we define \( \Omega(\alpha) \triangleq \{ x \geq 0 | f(t)^T x \leq g(t), \forall t \in [\alpha, 1] \} \), for \( \alpha \in [0, 1] \), and consider the so-called “maximum feasibility problem” (MFP) for finding the smallest \( \alpha \in [0, 1] \) such that \( \Omega(\alpha) \neq \emptyset \).

The maximum feasibility problem can be cast as the following linear semi-infinite programming problem:

\[
\begin{align*}
\text{MFP-LSIP} \quad & \max \quad 1 - \alpha \\
\text{s.t.} \quad & f(t)^T x \leq g(t), \forall t \in [\alpha, 1] \\
& 0 \leq \alpha \leq 1 \\
& x \geq 0.
\end{align*}
\]  

(4.12)

Obviously, the bisection method (with respect to \( \alpha \)) combined with a linear semi-infinite programming algorithm can be used to solve (4.12). However, it is computationally expensive to execute a linear semi-infinite programming algorithm in each iteration of the bisection method. Moreover, most of the numerical algorithms for linear semi-infinite programming do not have the ability to detect the inconsistency of subsystems of (4.12) and require a search procedure for global optimization.

In this section, we extend the idea of ACCPM to solve MFP-LSIP (4.12). The proposed approach can also be used to solve one class of linear semi-infinite systems as well as a fuzzy linear inequality.
4.3.1 Proposed Approach for MFP

We consider a cutting plane method for solving MFP-LSIP. In order to be able to estimate the complexity of the method, we make two assumptions akin to Goffin et al. [25].

Assumption 1. \( \Omega(1) \cap \{ x \in \mathbb{R}^n | 0 \leq x \leq e \} \neq \emptyset \), where \( e \) is the vector of all ones.

Assumption 2. There exists one \( \alpha \in [0, 1) \) such that \( \Omega(\alpha) \neq \emptyset \) and \( \Omega(\alpha) \subset \{ x \in \mathbb{R}^n | 0 \leq x \leq e \} \).

In order to define the oracle, we need another assumption, namely,

Assumption 3. \( \| f(t) \| \neq 0 \), \( \forall t \in [0, 1] \) and for any given \( \bar{x} \in \mathbb{R}^n \), the cardinality of \( \{ t \in [0, 1] | f(t)^T \bar{x} - g(t) = 0 \} \) is finite.

The problem here is to find the smallest \( \alpha \in [0, 1] \) such that \( \Omega(\alpha) \neq \emptyset \). In order to define the oracle, we introduce another formulation for the MFP. Let the function \( \beta(x) \) be defined as

\[
\beta(x) \triangleq \begin{cases} 
+\infty, & \text{if } f(1)^T x > g(1); \\
\min \{ \alpha \in [0, 1] | x \in \Omega(\alpha) \neq \emptyset \}, & \text{if } f(1)^T x \leq g(1).
\end{cases} \tag{4.13}
\]

Consequently, solving MFP is equivalent to solving the following optimization problem

\[
\min_{x \in \mathbb{R}^n} \beta(x). \tag{4.14}
\]

**Lemma 3** Under the above mentioned three assumptions, \( \Omega^* \), the solution set of (4.12), is a nonempty convex set.

**Proof.** Under the assumptions, we know that the feasible set of MFP-LSIP is nonempty. It follows that the solution set of MFP-LSIP is nonempty. Consequently, \( \Omega^* \) is nonempty.

Suppose there are two \( x_1^* \) and \( x_2^* \) in \( \Omega^* \). Then, we have \( \alpha^* = \beta(x_1^*) = \beta(x_2^*) \),

\[
\begin{align*}
\left\{ f(t)^T x_1^* \leq g(t), \quad & \forall t \in [\alpha^*, 1], \quad \text{and} \quad f(t)^T x_2^* \leq g(t), \quad \forall t \in [\alpha^*, 1] \right. \\
x_1^* \geq 0 & \quad \left. \text{and} \quad x_2^* \geq 0 \right.
\end{align*}
\]

53
For any $\gamma \in [0, 1]$, let $\bar{x} = \gamma x_1^* + (1 - \gamma) x_2^*$. Then, we have $\bar{x} \geq 0$ and

$$f(t)^T \bar{x} - g(t) = \gamma(f(t)^T x_1^* - g(t)) + (1 - \gamma)(f(t)^T x_2^* - g(t)) \leq 0, \forall t \in [\alpha^*, 1].$$

It follows that $\bar{x} \in \Omega^*$, i.e., $\Omega^*$ is convex. \hfill \Box

Notice that, for a given $x^0$, if $f(1)^T x^0 < g(1)$ then $\beta(x^0)$ can be obtained by finding the zeros of $f(t)^T x^0 - g(t) = 0$. From the definition of $\Omega(\alpha)$, we know that $\beta(x^0)$ should be the largest zero, say $t^0$, $0 \leq t^0 < 1$, such that there exists an interval $B_{x^0}(t^0)$ with $\delta^0 > 0$ and

$$(f(t^0 - \delta)^T x^0 - g(t^0 - \delta))(f(t^0 + \delta)^T x^0 - g(t^0 + \delta)) < 0, \forall \delta \in [0, \delta^0]. \quad (4.15)$$

We now outline an algorithm for solving MFP-LSIP.

**Algorithm 5: ACCPM for solving MFP-LSIP**

**Step 0:** Pick a sufficiently small $\varepsilon > 0$. Set $k = 0$.

Let $A^k = \begin{bmatrix} I & -I \\ f(1)^T & 0 \end{bmatrix}$ and $b^k = \begin{bmatrix} e \\ 0 \end{bmatrix}$.

Find the analytic center $x^k$ of $\Omega^k = \{x \in \mathbb{R}^n | A^k x \leq b^k \}$.

**Step 1:** Find the largest zero $t_k$ of $f(t)^T x^k - g(t) = 0$ such that $t_k < 1$.

**Step 2:** If $t_k \leq 0$ then output $\alpha^* = 0$ and stop. Otherwise,

**Step 3:** Set $k = k + 1$.

Let $A^k = \begin{bmatrix} A^{k-1} \\ f(t^{k-1})^T \end{bmatrix}$ and $b^k = \begin{bmatrix} b^{k-1} \\ g(t^{k-1}) \end{bmatrix}$.

Find the analytic center $x^k$ of $\Omega^k = \{x \in \mathbb{R}^n | A^k x \leq b^k \}$.

**Step 4:** If $\|x^k - x^{k-1}\| < \varepsilon$ then output $\alpha^* = t^{k-1}$ and stop.

Otherwise, return to Step 1.

54
Since $f(t^k)^T x^k = g(t^k)$, the cuts generated by the proposed algorithm are central cuts. Notice that in the proposed algorithm, the root-finding procedure has been relaxed to just finding the largest root smaller than 1 instead of also satisfying condition (4.15).

We now show that $\Omega^*$, the solution set of (4.12), always lies on the correct side of the cutting plane generate by the proposed algorithm.

**Lemma 4** For any cut generated by the proposed algorithm, $\Omega^* \subset \{ x \in \mathbb{R}^n | f(t^k)^T x \leq g(t^k) \}$.

**Proof.** Let $x^*$ be a solution of (4.12). We have $\Omega^* = \{ x \in \mathbb{R}^n | f(t)^T x \leq g(t), \forall t \in [\beta(x^*), 1] \}$ and $\beta(x^*) \leq \beta(x), \forall x \in \mathbb{R}^n$. For each analytic center $x^k$ generated by the algorithm, we have $t^k \geq \beta(x^k) \geq \beta(x^*)$. Consequently, we have $t^k \in [\beta(x^*), 1]$, i.e., $\Omega^* \subset \{ x \in \mathbb{R}^n | f(t^k)^T x \leq g(t^k) \}$. \hfill \square

Theoretically, ACCPM relies on the important interior assumption: $\text{int}(\Omega^k) \neq \emptyset$, for all $k$ [25]. We know that $\Omega^0$ is a full-dimensional convex polyhedron and $x^0 \in \text{int}(\Omega^0)$. It follows that $\Omega^1$ is a full-dimensional convex polyhedron. Consequently, for all $k$, $\Omega^k$ is a full-dimensional convex polyhedron, i.e., $\text{int}(\Omega^k) \neq \emptyset, \forall k$. In addition, if the cuts generated by the algorithm satisfy $\| f(t^k) \| = 1$ for each $k$ then the complexity analysis of ACCPM presented in [70] and [25] can be applied to the proposed algorithm. Specifically,

**Theorem 6** Provided Assumptions 1 - 3 are satisfied and $\| f(t^k) \| = 1$ for each $k$, the proposed algorithm terminates in at most $k$ iterations, where $k$ is the smallest integer satisfying the inequality

$$
\frac{2n + k}{2 + 2n \ln(1 + \frac{k}{8n^2})} \geq \frac{n}{4 \epsilon^2}.
$$

In practice, the computation of the zero, $t^k$, might yield $\Omega^{k+1}$ such that $\text{int}(\Omega^{k+1}) \approx \emptyset$ or $\Omega^{k+1} \approx \emptyset$. In these cases, the ACCPM does not work. One way to avoid this computational difficulty is to relax the cutting plane by replacing $t^k$ with $t^k + \delta$ ($\delta$ is a small positive number) in Step 1 of the proposed algorithm.

Another approach to avoid the computational difficulty is to replace the analytic center with the entropic analytic center (EAC). Recall that in computing the EAC
we have the ability to detect inconsistency of a system. However, there is no reported complexity analysis for the EAC cutting plane method.

We now outline the EACCPM algorithm.

**Algorithm 6: EACCPM for Solving MFP-LSIP**

Step 0: Pick a sufficiently small $\varepsilon > 0$. Set $k = 0$.

Let $A^k = [f(1)^T]$ and $b^k = [g(1)]$.

Find the entropic analytic center $x^k$ of $\Omega^k = \{x \geq 0|A^k x \leq b^k\}$.

Step 1: Find the largest zero $t^k$ of $f(t)^T x^k - g(t) = 0$ such that $t^k < 1$.

Step 2: If $t^k \leq 0$ then output $\alpha^* = 0$ and stop.

Step 3: Set $k = k + 1$.

Let $A^k = \begin{bmatrix} A^{k-1} \\ f(t^{k-1})^T \end{bmatrix}$ and $b^k = \begin{bmatrix} b^{k-1} \\ g(t^{k-1}) \end{bmatrix}$.

Find the entropic analytic center $x^k$ of $\Omega^k = \{x \geq 0|A^k x \leq b^k\}$.

Step 4: If no $x^k$ exists or $\|x^k - x^{k-1}\| < \varepsilon$ then output $\alpha^* = t^{k-1}$ and stop.

Otherwise, return to Step 1.

Notice that, the proposed approach can be modified to solve the following class of linear semi-infinite system:

\[
\begin{cases} 
  f(t)^T x \leq g(t), \quad \forall t \in T \\
  x \geq 0,
\end{cases}
\]

where $T$ is a closed interval in $\mathbb{R}^1$. Other methods for solving linear semi-infinite system usually require a search procedure for global optimization on $T$ (to identify the most violated inequality). This can be costly. With the proposed algorithms, global optimization is replaced with a root-finding procedure.

In general, there is no guarantee that the sequence $\{t^k\}$ generated by either AC-CPM or EACCPM is a monotone decreasing sequence. However under some conditions, $\{t^k\}$ can be shown to be monotone decreasing. In particular,
Theorem 7 Let $x^k$ be the ACs (or EACs) generated by the proposed algorithm at $k^{th}$ iteration. If each $f_i(t), i = 1, ..., n$, is a convex function on $[0, 1]$ and $g(t)$ is a concave function on $[0, 1]$, then $t^k < t^{k-1}$, for all $k$.

Proof. Let $f(t, x) \triangleq f(t)^T x - g(t)$. For any give $x \geq 0$, we know that $f(t, x)$ is a convex function with respect to $t \in [0, 1]$. Without loss of generality, we assume that there is always one root of $f(t, x^k) = 0$ with $t \in [0, 1]$. From the property of analytic center, we know $x^0 \in \text{int} \{x \geq 0|f(1, x) \leq 0\}$, i.e. $f(1, x^0) < 0$. Since $f(t, x^0)$ is convex function on $t$, we can have $t^0 < 1$ in the first iteration. Suppose that as of the $(k - 1)^{st}$ iteration, we have $t^{k-1} < ... < t^0 < 1$. Now, in $k^{th}$ iteration, we have $x^k \in \text{int} \{x \geq 0|f(1, x) \leq 0, f(t^0, x) \leq 0, ..., f(t^{k-1}, x) \leq 0\}$, i.e., we have $f(1, x^k) < 0, f(t^0, x^k) < 0, ..., f(t^{k-1}, x^k) < 0$. Since $f(t, x^k)$ is convex on $t$, it follows that $t^k < t^{k-1}$. □

4.3.2 Numerical Examples

We implemented both the ACCPM and the EACPM algorithms on 2 problems and compared their performance. We didn’t use $\|x^k - x^{k-1}\| < \varepsilon$ as the stopping criterion to terminate the algorithm. Instead, we executed the algorithms until they compute an $\Omega^{k+1}$ such that $\text{int}(\Omega^{k+1}) \approx \emptyset$ or $\Omega^{k+1} = \emptyset$. In our computational experiments, we used the “dual Newton procedure” to find the analytic center.

The first test problem is as follows.

\[
\begin{align*}
x_1 + (t + 3)x_2 + (t + 3)^2x_3 & \leq t, \quad \forall t \in [0, 1] \\
x_1 & \geq 0.
\end{align*}
\]

Clearly, for any $t \in [0, 1]$, $x^* = (0, 0, 0)$ is the one and only feasible solution of (4.17), so the optimal solution of MFP for (4.17) is $\alpha^* = 0$. The results are summarized in Table 4.5. In the table, $k$ denotes the iteration number, $x^k = (x_1, x_2, x_3)$, $\alpha = t^k$. N/A means that the algorithm determines $\text{int}(\Omega^{k+1}) \approx \emptyset$ and NO means that the algorithm determines $\Omega^{k+1} = \emptyset$. 

57
Table 4.5: results of MFP example 1

<table>
<thead>
<tr>
<th>k</th>
<th>ACCPM</th>
<th>EACCPM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>0</td>
<td>0.2035</td>
<td>0.0637</td>
</tr>
<tr>
<td>1</td>
<td>0.1347</td>
<td>0.0406</td>
</tr>
<tr>
<td>2</td>
<td>0.0839</td>
<td>0.0254</td>
</tr>
<tr>
<td>3</td>
<td>0.0513</td>
<td>0.0158</td>
</tr>
<tr>
<td>4</td>
<td>0.0312</td>
<td>0.0098</td>
</tr>
<tr>
<td>5</td>
<td>0.0229</td>
<td>0.0073</td>
</tr>
<tr>
<td>6</td>
<td>0.0121</td>
<td>0.0039</td>
</tr>
<tr>
<td>7</td>
<td>0.0066</td>
<td>0.0021</td>
</tr>
<tr>
<td>8</td>
<td>0.0035</td>
<td>0.0011</td>
</tr>
<tr>
<td>9</td>
<td>0.0019</td>
<td>0.0006</td>
</tr>
<tr>
<td>10</td>
<td>0.0038</td>
<td>5.6e-08</td>
</tr>
<tr>
<td>11</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>12</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

In the next example, we demonstrate how to apply the proposed method to determine the best $\alpha$-preference for a fuzzy linear inequality [73].

Consider the following fuzzy linear inequality:

\[
\begin{aligned}
\tilde{1}x_1 + \tilde{2}x_2 + \tilde{1}x_3 & \leq_{\alpha} \frac{1}{3} \\
\end{aligned}
\]

(4.18)

where $\leq_{\alpha}$ is a relation between two fuzzy numbers, and $\tilde{1}$, $\tilde{2}$ and $\frac{1}{3}$ are fuzzy numbers with membership functions:

\[
\mu_1(x) = \begin{cases}
\frac{x+3}{4}, & \text{if } 0 \leq x \leq 1; \\
\sqrt{2 - x}, & \text{if } 1 \leq x \leq 2; \\
0, & \text{otherwise},
\end{cases}
\]

(4.19)

\[
\mu_2(x) = \begin{cases}
\frac{\sqrt{1x}}{2}, & \text{if } 0 \leq x \leq 2; \\
2 - \frac{x}{2}, & \text{if } 2 \leq x \leq 4; \\
0, & \text{otherwise},
\end{cases}
\]

(4.20)
and

\[
\mu_M(x) = \begin{cases} 
\frac{3x+15}{16}, & \text{if } -5 \leq x \leq \frac{1}{3}; \\
\frac{3-3x}{2}, & \text{if } \frac{1}{3} \leq x \leq 1; \\
0, & \text{otherwise.}
\end{cases} \tag{4.21}
\]

A convex fuzzy number \( \tilde{M} \), \( M \in R \), is a fuzzy set defined on the real line \( R \) with a membership function \( \mu_{\tilde{M}}(\cdot) \) such that its upper \( \alpha \)-level set \( \tilde{M}_\alpha = \{ x \in R | \mu_{\tilde{M}}(x) \geq \alpha \} \) forms an interval \([L_{\tilde{M}}(\alpha), R_{\tilde{M}}(\alpha)]\) where

\[
L_{\tilde{M}}(\alpha) = \min \{ x \in R | \mu_{\tilde{M}}(x) \geq \alpha \} \tag{4.22}
\]

and

\[
R_{\tilde{M}}(\alpha) = \max \{ x \in R | \mu_{\tilde{M}}(x) \geq \alpha \} \tag{4.23}
\]

are real-valued continuous functions in \( \alpha \in [0, 1] \). Let \( \tilde{M}_1 \) and \( \tilde{M}_2 \) be convex fuzzy numbers. For any given \( \alpha \in [0, 1] \), we say \( \tilde{M}_1 \leq_\alpha \tilde{M}_2 \), if

\[
\begin{align*}
L_{\tilde{M}_1}(t) &\leq L_{\tilde{M}_2}(t), \quad \forall t \in [\alpha, 1], \\
R_{\tilde{M}_1}(t) &\leq R_{\tilde{M}_2}(t),
\end{align*}
\]

The problem of the best \( \alpha \)-preference for the fuzzy linear inequality (4.18) is that of finding the smallest value of \( \alpha \) such that (4.18) has a solution.

After some calculations, for a given \( \alpha \in [0, 1] \), (4.18) can be transformed to the following system of continuous linear inequalities:

\[
\begin{align*}
(4t - 3)x_1 + (2t^3)x_2 + (4t - 3)x_3 &\leq \frac{16}{3}t - 5, \quad \forall t \in [\alpha, 1] \\
(2 - t^2)x_1 + (4 - 2t)x_2 + (2 - t^2)x_3 &\leq 1 - \frac{2}{3}t, \\
x_1, x_2, x_3 &\geq 0
\end{align*}
\tag{4.24}
\]

The associated maximum feasibility problem is to find the smallest \( \alpha \) such that (4.24) has a feasible solution. The proposed method can be modified to solve this problem. For this example, at each iteration the oracle can generate one cut or two cuts, depending on the zeros of each equation. To be more precise, we associate \( t_{k_1}^i \) and \( t_{k_2}^i \) with the first and second inequalities of (4.24), respectively. If either \( t_{k_1}^i \) or \( t_{k_2}^i \) is 0, the oracle will generate only one cut corresponding to the non-zero \( t_{k_1}^i \) or \( t_{k_2}^i \). If both \( t_{k_1}^i \) and \( t_{k_2}^i \) are not 0, the oracle will generate a cut for each inequality based on the value \( t^k \triangleq \max\{t_{k_1}^i, t_{k_2}^i\} \). In this manner, we won't over-estimate \( t^k \).

The optimal solution for this example is \( \alpha^* = \frac{15}{16} \). The computational results are summarized in Table 4.6.
The computational results for the two examples demonstrate that EACCPM has some promising advantages over ACCPM. However, the convergence and complexity properties remain to be studied.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$\alpha$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0652</td>
<td>0.0333</td>
<td>0.0652</td>
<td>0.9691</td>
<td>0.0716</td>
<td>0.0139</td>
<td>0.0716</td>
<td>0.9634</td>
</tr>
<tr>
<td>1</td>
<td>0.0374</td>
<td>0.0186</td>
<td>0.0374</td>
<td>0.9539</td>
<td>0.0344</td>
<td>0.0029</td>
<td>0.0344</td>
<td>0.9475</td>
</tr>
<tr>
<td>2</td>
<td>0.0214</td>
<td>0.0104</td>
<td>0.0214</td>
<td>0.9464</td>
<td>0.0121</td>
<td>0.0003</td>
<td>0.0121</td>
<td>0.9406</td>
</tr>
<tr>
<td>3</td>
<td>0.0124</td>
<td>0.0059</td>
<td>0.0124</td>
<td>0.9425</td>
<td>0.0020</td>
<td>5.0e-06</td>
<td>0.0020</td>
<td>0.9380</td>
</tr>
<tr>
<td>4</td>
<td>0.0072</td>
<td>0.0034</td>
<td>0.0072</td>
<td>0.9403</td>
<td>6.5e-05</td>
<td>2.6e-09</td>
<td>6.5e-05</td>
<td>0.9375</td>
</tr>
<tr>
<td>5</td>
<td>0.0042</td>
<td>0.0019</td>
<td>0.0042</td>
<td>0.9391</td>
<td>1.3e-07</td>
<td>2.9e-15</td>
<td>1.3e-07</td>
<td>0.9375</td>
</tr>
<tr>
<td>6</td>
<td>0.0029</td>
<td>5.0e-07</td>
<td>0.0029</td>
<td>0.9382</td>
<td>2.3e-12</td>
<td>1.1e-25</td>
<td>2.3e-12</td>
<td>0.9375</td>
</tr>
<tr>
<td>7</td>
<td>0.0014</td>
<td>1.1e-08</td>
<td>0.0014</td>
<td>0.9379</td>
<td>3.1e-18</td>
<td>8.6e-41</td>
<td>3.1e-18</td>
<td>0.9375</td>
</tr>
<tr>
<td>8</td>
<td>0.0007</td>
<td>3.0e-10</td>
<td>0.0007</td>
<td>0.9377</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9375</td>
</tr>
<tr>
<td>9</td>
<td>4.6e-08</td>
<td>6.2e-17</td>
<td>4.6e-08</td>
<td>0.9375</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9375</td>
</tr>
<tr>
<td>10</td>
<td>6.0e-11</td>
<td>3.3e-23</td>
<td>6.0e-11</td>
<td>0.9375</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9375</td>
</tr>
<tr>
<td>11</td>
<td>6.5e-15</td>
<td>7.0e-32</td>
<td>6.5e-15</td>
<td>0.9375</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9375</td>
</tr>
<tr>
<td>12</td>
<td>2.9e-19</td>
<td>1.7e-41</td>
<td>2.9e-19</td>
<td>0.9375</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9375</td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9375</td>
</tr>
</tbody>
</table>
4.4 Remarks

In this chapter we have studied the analytic center cutting plane method for solving linear semi-infinite inequality systems. Two multiple-cut schemes, one based on central cuts and the other based on deep cuts, were presented. Instead of using a global optimization to identify the most violated constraints, we used a relaxed cutting plane to form cuts. In theory, the algorithm based on the central cuts stops after $O(\frac{m^2n^2}{\epsilon^2})$ cutting planes have been generated.

We have also defined a new analytic center, the entropic analytic center, to be the unique minimizer of the minimum entropy optimization problem (4.7). The entropic analytic center can be obtained by a modification of the algorithm proposed in Chapter 3. We examined the use of the entropic analytic center as a replacement for the analytic center in ACCPM to solve SLIS. A relaxed cut method was proposed for helping define the cuts. The computational results did not show any much difference between ACCPM and EACCPM. However, it showed the potential advantage of the proposed relaxed cut method in solving a SLIS.

The feasibility issue of a SLIS whose coefficients are continuous functions of a one-dimensional parameter ($t$) has been defined as the maximum feasibility problem. We proposed an algorithm based on ACCPM to solve the maximum feasibility problem. The proposed approach employs a root-finding procedure instead of the commonly used global optimization procedure for identifying violated constraints. The EACCPM algorithm can be further extended to solve a class of SLIS (4.16) with the ability to detect inconsistency.
Chapter 5

Entropic Regularization Method
for Solving Linear Inequalities

The problem studied in this chapter is to find a feasible solution to the following linear inequality system:

\[ \sum_{i=1}^{n} a_i(t)x_i \leq b(t), \quad \forall t \in T, \quad (5.1) \]

where \( x \in \mathbb{R}^n \), \( T \) is a compact set in \( \mathbb{R}^l \) with \( l \in \mathbb{N} \), and \( a_1(t), a_2(t), \ldots, a_n(t) \) and \( b(t) \), are real-valued functions on \( T \).

When \( T \) is finite, say \( T = \{t_1, t_2, \ldots, t_m\} \), (5.1) becomes a system of finitely many linear inequalities:

\[ Ax \leq b, \quad (5.2) \]

where \( A \) is an \( m \times n \) matrix with \( A_{ji} = a_i(t_j) \) and \( b \) is an \( m \)-dimensional vector with \( b_j = b(t_j) \).

When \( T \) is convex compact subset of \( \mathbb{R}^l \) with \( |T| = \infty \), and \( a_1(t), a_2(t), \ldots, a_n(t) \) and \( b(t) \) are continuous functions on \( T \), (5.1) becomes a system of infinitely many linear inequalities.

While, unlike the problems we studied in Chapters 3 and 4, \( x \) is not required to be nonnegative, the algorithms proposed in Chapters 3 and 4 still can be applied to solve (5.1) by replacing \( x = \bar{x} - \tilde{x} \), where \( \bar{x} \) and \( \tilde{x} \) are nonnegative. However this transformation increases the problem dimension.

Besides the direct methods mentioned in Chapter 2 for solving (5.1), one can
convert it to an equivalent min-max problem, namely

\[
\min_{x \in \mathbb{R}^n} \max_{t \in T} \left\{ \sum_{i=1}^{n} a_i(t)x_i - b(t) \right\} \triangleq \min_{x \in \mathbb{R}^n} F(x). \tag{5.3}
\]

We know that system (5.1) has a solution if and only if \( F(x) \) has a minimizer \( x^* \) with \( F(x^*) \leq 0 \). The major difficulty in solving (5.3) lies in the non-differentiability of the max-function

\[
F(x) = \max_{t \in T} \left\{ \sum_{i=1}^{n} a_i(t)x_i - b(t) \right\}.
\]

Several numerical methods for solving (5.3) can be found in [22], [61], [60], [2] and [63].

Recently, Li and Fang [45] and Fang and Wu [18] proposed the “entropic regularization” approach for solving (5.3). The entropic regularization provides a smooth function \( F_p(x) \) in an explicit form to approximate the max-function \( F(x) \). A solution for (5.3) can be obtained by minimizing \( F_p(x) \).

In this chapter, we apply the idea of the entropic regularization method to solve (5.1). In Section 5.1, we review the entropic regularization method for solving the min-max problems. Some elementary properties are discussed. The proposed approach is presented in Section 5.2 and numerical experiments are reported in Section 5.3.

### 5.1 Entropic Regularization Method

In this section, we study the entropic regularization method for solving the following min-max problem:

\[
\min_{x \in X} \max_{t \in T} \{ f(x,t) \} \triangleq \min_{x \in X} F(x) \tag{5.4}
\]

where \( X \subset \mathbb{R}^n \), a compact subset \( T \subset \mathbb{R}^l \), \( l \in \mathbb{N} \) and \( f(x,t) \in C^1(X), \forall t \in T \).

When \( T \) is finite, we refer to (5.4) as a “finite min-max problem”, while “infinite min-max problem” means \( |T| = \infty \). We first discuss the entropic regularization method for solving finite min-max problems, and then extend it to handle infinite min-max problems.
5.1.1 Finite Min-Max Problems

In this subsection, let \( T = \{ t_1, t_2, ..., t_m \} \) and \( X = \mathbb{R}^n \). For \( x \in \mathbb{R}^n \), we define \( f_j(x) \triangleq f(x, t_j), \ j = 1, ..., m \). Notice that (5.4) has the following Lagrange function:

\[
L(x, \lambda) \triangleq \sum_{j=1}^{m} \lambda_j f_j(x),
\]

where \( \lambda = (\lambda_1, ..., \lambda_m)^T \) and satisfies

\[
\lambda \in \Lambda \triangleq \{ \lambda \in \mathbb{R}^m | \sum_{j=1}^{m} \lambda_j = 1, \lambda \geq 0 \}.
\]

Under some regularity assumptions, it can be shown that

\[
F(x) = \max_{\lambda \in \Lambda} L(x, \lambda), \ \forall x \in \mathbb{R}^n.
\]

However, maximization of \( L(x, \lambda) \) over \( \Lambda \) usually yields a function \( \lambda(x) \) which is not smooth and has no explicit expression. Use of a smoothing technique, or regularization method, has been proposed to overcome this issue [22][61].

A general regularization method employs the following perturbed Lagrange function:

\[
L_p(x, \lambda) \triangleq \sum_{j=1}^{m} \lambda_j f_j(x) + \frac{1}{p} R(\lambda),
\]

where \( p > 0 \) is a control parameter and \( R \) is the regularization function. Through proper selection of the regularization function \( R \), maximizing (5.8) with respect to \( \lambda \) results in a smooth approximation function

\[
F_p(x) = \max_{\lambda \in \Lambda} L_p(x, \lambda).
\]

Some regularization functions can be found in [22][61][2]; however, no explicit expression for \( F_p(x), \ \forall x \in \mathbb{R}^n \) is given.

Li and Fang [45] considered Shannon’s entropy function as the regularization function, i.e.,

\[
R(\lambda) = - \sum_{j=1}^{m} \lambda_j \ln \lambda_j,
\]

where \( \lambda \in \Lambda \). Substituting in (5.8), we have

\[
L_p(x, \lambda) = \sum_{j=1}^{m} \lambda_j f_j(x) - \frac{1}{p} \sum_{j=1}^{m} \lambda_j \ln \lambda_j,
\]
where $\lambda \in \Lambda$.

Notice that $L_p(x, \lambda)$ is a strictly concave function in $\lambda$. Maximizing (5.11) over $\Lambda$ gives a unique optimal solution

$$
\lambda_j^*(x, p) = \frac{\exp[p f_j(x)]}{\sum_{k=1}^m \exp[p f_k(x)]}, \quad 1 \leq j \leq m. \tag{5.12}
$$

Substituting $\lambda_j^*$'s into $F_p(x)$ results in a smooth function,

$$
F_p(x) = \frac{1}{p} \ln \left\{ \sum_{j=1}^m \exp[p f_j(x)] \right\}. \tag{5.13}
$$

Li and Fang [45] showed the following properties of $F_p(x)$:

1. For any $x \in \mathbb{R}^n$, $F_r(x) \leq F_s(x)$ if $s \leq r$.

2. $F_p(x)$ converges to $F(x)$ uniformly on $\mathbb{R}^n$, as $p$ tends to infinity.

3. For any $p > 0$, $F(x) \leq F_p(x) \leq F(x) + \frac{\ln m}{p}$, for all $x \in \mathbb{R}^n$.

4. $F_p(x)$ is convex, if each $f_j(x)$, $j = 1, ..., m$, is a convex on $\mathbb{R}^n$.

Now consider $f_j(x) = \sum_{i=1}^n a_i(t_j)x_i - b(t_j), j = 1, ..., m$, for all $x \in \mathbb{R}^n$. Letting $A_j = (a_1(t_j), a_2(t_j), ..., a_n(t_j))$ and $b_j = b(t_j)$, (5.13) becomes

$$
F_p(x) = \frac{1}{p} \ln \left\{ \sum_{j=1}^m \exp[p(A_jx - b_j)] \right\}, \tag{5.14}
$$

with gradient

$$
\nabla F_p(x) = A^T \lambda, \tag{5.15}
$$

and Hessian

$$
\nabla^2 F_p(x) = p A^T (\Gamma - \lambda \lambda^T) A, \tag{5.16}
$$

where $\lambda = (\lambda_1, ..., \lambda_m)^T$ with

$$
\lambda_j = \frac{\exp[p(A_jx - b_j)]}{\sum_{i=1}^m \exp[p(A_i x - b_i)]}, \quad j = 1, ..., m, \tag{5.17}
$$

and $\Gamma = \text{diag}(\lambda)$. Notice that $F_p(x)$ is a smooth convex function on $\mathbb{R}^n$. With a sufficiently large $p$, we can apply unconstrained optimization algorithms to minimize $F_p(x)$ to produce a very accurate solution for the min-max problem (5.3).
5.1.2 Infinite Min-Max Problems

In this subsection, we assume that $X \subset \mathbb{R}^n$ and $T \subset \mathbb{R}^d$ ($|T| = \infty$) are convex compact sets and $f(x, t) \in C^1(X), \forall t \in T$. Similar to the previous approach for the finite min-max problem, the Lagrange function of (5.4) can be defined as

$$L(x, \lambda) \triangleq \int_T f_t(x)\lambda(t)dt,$$

for each $x \in X$ and

$$\lambda \in \Lambda \triangleq \left\{ \lambda(t) \in L^1(T) | \int_T \lambda(t)dt = 1 \text{ and } \lambda(t) \geq 0, \text{ a.e.} \right\}.$$

Using Shannon’s entropy function,

$$R(\lambda) = -\int_T \lambda(t)\ln \lambda(t)dt,$$

as the regularization function, we have

$$L_p(x, \lambda) = \int_T f_t(x)\lambda(t)dt - \frac{1}{p} \int_T \lambda(t)\ln \lambda(t)dt,$$

for all $x \in X$ and $\lambda \in \Lambda$.

Maximizing (5.21) with respect $\lambda$ gives

$$\lambda^*(t) = \frac{\exp[pf_t(x)]}{\int_T \exp[pf_t(x)]dt}.$$

Replacing $\lambda$ in (5.21) with $\lambda^*$ results in

$$F_p(x) = \frac{1}{p} \ln \{\int_T \exp[pf_t(x)]dt\}.$$

Fang and Wu [18] showed that $F_p(x)$ converges to $F(x)$ uniformly on $X$ as $p$ tends to infinity under the following two conditions:

**Condition 1.** $f_t(x)$ is uniformly bounded, i.e., there exists an $M$ such that $|f_t(x)| \leq M$, $\forall x \in X$ and $\forall t \in T$.

**Condition 2.** $f_t(x)$ is super-uniformly continuous, i.e., $\forall \epsilon > 0$ there exists a $\delta > 0$ such that if for any $x, y \in X$ and $|x - y| < \delta$, then $|f_t(x) - f_t(y)| < \epsilon, \forall t \in T$. 

66
Now, letting \( f_i(x) = \sum_{i=1}^{n} a_i(t)x_i - b(t) \) with \( a_1(t), a_2(t), ..., a_n(t) \) and \( b(t), j = 1, ..., m \), being continuous valued functions on \( T \), we obtain a convex function

\[
F_p(x) = \frac{1}{p} \ln \{ \int_T \exp [p(\sum_{i=1}^{n} a_i(t)x_i - b(t))] dt \}.
\]  

(5.24)

Furthermore \( F_p(x) \) converges uniformly to \( F(x) \) on \( X \), as \( p \) tends to infinity. In addition we have

\[
\nabla F_p(x) = \begin{bmatrix}
\int_T a_1(t)\lambda^*(t)dt \\
\vdots \\
\int_T a_n(t)\lambda^*(t)dt
\end{bmatrix},
\]

(5.25)

and

\[
\frac{\partial^2 F_p(x)}{\partial x_i \partial x_j} = p \left\{ \int_T a_i(t)a_j(t)\lambda^*(t)dt - \int_T a_i(t)\lambda^*(t)dt \int_T a_j(t)\lambda^*(t)dt \right\},
\]

(5.26)

where

\[
\lambda^*(t) = \frac{\exp [p(\sum_{i=1}^{n} a_i(t)x_i - b(t))]}{\int_T \exp [p(\sum_{i=1}^{n} a_i(t)x_i - b(t))] dt}.
\]

(5.27)

Note that \( F_p(x) \) is a smooth function which can be treated by unconstrained optimization algorithms. However, this approach could encounter a computational bottleneck in the multidimensional integration over \( T \).

### 5.2 Proposed Algorithm

Notice that the optimization problem \( \min F_p(x) \), where \( F_p(x) \) is defined in (5.14) or (5.24), is an unconstrained convex program. We now present a “model algorithm” for solving (5.1). The term “model algorithm” means that the algorithm is presented in a very simple format such that the convergence of the algorithm is clear.
Algorithm 7: model algorithm

Initialization:
Initialize $x^0$. Set $p$ equal to a sufficiently large number and set $k = 0$.

Basic Step:

Step 1: If $\sum_{i=1}^{n} a_i(t)x^k_i \leq b(t), \forall t \in T$, then output $x^k$ as a solution and stop. Otherwise,

Step 2: If $x^k$ is the minimizer of $F_p(x)$, then output $x^k$ as an approximate solution and stop. Otherwise,

Step 3: Execute one iteration of an unconstrained minimization algorithm to generate an updated $x^{k+1}$.

Step 4: Set $k = k + 1$ and return to Step 1.

Theoretically speaking, the model algorithm should be able to find an exact feasible solution or to provide a approximate solution for system (5.1). However, in practice, several computational issues need to be considered: (i) the initialization of $x^0$ and $p$; (ii) the stopping conditions in Step 1 and Step 2; and (iii) the unconstrained optimization algorithm used in Step 3.

1. Numerical overflow: Evaluation of exponential functions is required in calculating $\nabla F_p(x^k)$ and $\nabla^2 F_p(x^k)$. The crucial point is the computation of $\sum_{j=1}^{m} \exp[p(A_jx^k - b_j)]$ or $\int_T \exp[p(\sum_{i=1}^{n} a_i(t)x^k_i - b(t))] dt$. Notice that, it is easy to encounter an overflow problem, if the parameters $x^0$ and $p$ are not well-chosen. In theory, the larger $p$ is, the better approximation $F_p(x)$ we can have. However, in general, there is no simple way to predict which $p$ and $x^0$ will not encounter an overflow problem. Our strategy is to start with an $x^0$ and a small $p^0$ and then at each iteration $k$ apply a descent algorithm to generate $x^k$ and an adaptive procedure to adjust $p^k$ according to the value of $\|\nabla F_p(x^k)\|$ or $k$ (iteration counter). The procedure will be discussed in the subsection on numerical examples.
2. **Stopping condition:** The stopping conditions in Step 1 and Step 2 are used to check the feasibility of the system (5.1). In Step 1, when $T$ is finite, there is no problem in detecting whether the current solution $x^k$ is feasible. However, when $T$ is infinite, there is no efficient method to detect the feasibility of $x^k$. In this case, either we use a very fine discretization $\bar{T}$ of $T$ at each iteration to check the feasibility of $x^k$, or skip Step 1 and only use the minimizer of $F_p(x)$ to check the feasibility.

3. **Optimization algorithm:** Ideally, any unconstrained optimization algorithm can be implemented to minimize $F_p(x)$. However, the following lemma implies that the use of the inverse of Hessian is prohibited when $T$ is finite.

**Lemma 4** If $\lambda$ is a positive $n$-dimensional vector and $\Gamma = \text{diag}(\lambda)$ then $\Gamma - \lambda\lambda^T$ is a singular matrix.

**Proof.**

\[
\begin{align*}
\text{rank}(\Gamma - \lambda\lambda^T) &= \text{rank} \left( \begin{bmatrix}
\lambda_1 & 0 & \cdots & 0 \\
0 & \lambda_2 & \cdots & 0 \\
\cdots & \cdots & \cdots & 0 \\
0 & \cdots & 0 & \lambda_n
\end{bmatrix} - \begin{bmatrix}
\lambda_1\lambda_1 & \lambda_1\lambda_2 & \cdots & \lambda_1\lambda_n \\
\lambda_2\lambda_1 & \lambda_2\lambda_2 & \cdots & \lambda_2\lambda_n \\
\cdots & \cdots & \cdots & \cdots \\
\lambda_n\lambda_1 & \cdots & \lambda_n\lambda_{n-1} & \lambda_n\lambda_n
\end{bmatrix} \right) \\
&= \text{rank} \left( \begin{bmatrix}
\lambda_1\lambda_1 - \lambda_1 & \lambda_1\lambda_2 & \cdots & \lambda_1\lambda_n \\
\lambda_2\lambda_1 & \lambda_2\lambda_2 - \lambda_2 & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\lambda_n\lambda_1 & \cdots & \lambda_n\lambda_{n-1} & \lambda_n\lambda_n - \lambda_n
\end{bmatrix} \right) \\
&= \text{rank} \left( \begin{bmatrix}
\lambda_1 - 1 & \lambda_2 & \cdots & \lambda_n \\
\lambda_1 & \lambda_2 - 1 & \cdots & \cdots \\
\cdots & \cdots & \cdots & \lambda_n \\
\lambda_1 & \cdots & \lambda_{n-1} & \lambda_n - 1
\end{bmatrix} \right) \\
&= \text{rank} \left( \begin{bmatrix}
\lambda_1 - 1 & \lambda_2 & \cdots & \sum_{i=1}^n \lambda_i - 1 \\
\lambda_1 & \lambda_2 - 1 & \cdots & \sum_{i=1}^n \lambda_i - 1 \\
\cdots & \cdots & \cdots & \cdots \\
\lambda_1 & \cdots & \lambda_{n-1} & \sum_{i=1}^n \lambda_i - 1
\end{bmatrix} \right) \leq n - 1.
\end{align*}
\]
Consequently, $\Gamma - \lambda \lambda^T$ is singular.

When $T$ is finite, for any given $p > 0$, the Hessian (5.16) of $F_p(x)$ is singular for all $x \in \mathbb{R}^n$. This suggests that the variants of Newton method are not applicable for the entropic regularization method.

When $T$ is infinite, singularity of $\nabla^2 F_p(x)$ occurs when $a_i(t)$, for some $i = 1, \ldots, n$, is linear function in $t$. This will be illustrated in Section 5.3.

5.2.1 Relation Between the Proposed Method and Other Methods

Notice that, if system (5.1) is inconsistent or $p$ is not well-chosen, Step 2 in the model algorithm may provide only an approximate solution for (5.1). However, this approximate solution is not one of the commonly seen $l_1$, $l_2$ or $l_{\infty}$ sense solutions. In this section, we discuss the relationship between the entropic regularization method and other methods. Here, we only consider the case in which $T$ is finite.

First, we study the relation between

$$\min_{x \in \mathbb{R}^n} F_p(x) = \frac{1}{p} \ln \left\{ \sum_{j=1}^{m} \exp[p(A_j x - b_j)] \right\}$$

and linear programming.

Consider system (5.1) as the following linear programming problems:

$$\text{LP}_0 \left\{ \begin{array}{l}
\min & 0 \\
\text{s.t.} & Ax \leq b.
\end{array} \right. \quad (5.28)$$

We can convert above program to the following unconstrained optimization with a homotopy parameter $p > 0$:

$$\text{UP}_{(p)} \left\{ \begin{array}{l}
\min & \sum_{j=1}^{m} \exp[p(A_j x - b_j)] \\
\text{s.t.} & x \in \mathbb{R}^n.
\end{array} \right. \quad (5.29)$$

Obviously, when $p \to \infty$, solving $\text{UP}_{(p)}$ is equivalent to solving $\text{LP}_0$. In this case, we can think $\sum_{j=1}^{m} \exp[p(A_j x - b_j)]$ as a penalty function for solving $\text{LP}_0$. Because of concavity of $\ln(\cdot)$ function and the fact that $\exp(\cdot) > 0$, minimizing $F_p(x) = \frac{1}{p} \ln \left\{ \sum_{j=1}^{m} \exp[p(A_j x - b_j)] \right\}$ is equivalent to minimizing $\sum_{j=1}^{m} \exp[p(A_j x - b_j)]$. However,
\( F_p(x) \) can not be regarded as a traditional penalty function or barrier function for solving program \( \text{LP}_0 \).

Now, recall Yang and Murty’s surrogate constraint method [69] mentioned in Chapter 2:

\[
x^{k+1} = x^k + \gamma (P_H(x^k) - x^k) = x^k - \frac{\gamma (A_j x^k - b_j)^T \pi}{\|A_j^T \pi\|^2} A_j^T \pi,
\]

(5.30)

where \( \gamma \in (0, 2] \) and \( P_H(x^k) \) is the orthogonal projection of \( x^k \) onto the hyperplane \( H = \{ x | \pi^T A_j x = \pi^T b_j \} \) with \( \sum_{j \in J} \pi_j = 1 \) and \( J = \{ j | A_j x^k - b_j > 0, 1 \leq j \leq m \} \).

When a general descent method is used for minimizing \( F_p(x) \), we have

\[
x^{k+1} = x^k - \alpha \nabla F_p(x^k) = x^k - \alpha A^T \lambda,
\]

(5.31)

where \( \alpha > 0 \) and \( \lambda_j = \exp[p(A_j x - b_j)]/\sum_{i=1}^{m} \exp[p(A_i x - b_i)], \ j = 1, ..., m. \)

Notice that \( \lambda_j \)'s in (5.31) are normalized weights and can be used as a particular weight vector \( \pi \) in (5.30) (except that Yang and Murty’s method only uses those weights associated with the violated constraints). Especially, when \( p \) is large enough, the value of \( \lambda_j \)'s associated with those non-violated constraints are much smaller than those associated with the violated constraints. Therefore, the gradient of the hyperplane \( H \) in Yang and Murty’s method is close to the gradient of \( F_p(x^k) \). In other words, the projection direction of Yang and Murty’s method is close to the decent direction of entropic regularization method. Furthermore, if we let \( p \to \infty \), then \( \lambda_j = 1 \) for the most violated constraint \( j \). Therefore, the orthogonal projection method [1] can be viewed as a special case of the entropic regularization method. Hence, it is natural to suspect that the entropic regularization method will have similar computational behavior as the successive orthogonal projection method.

### 5.3 Numerical Examples

In this section, our computational experience with some numerical examples with randomly generated data is reported. Since the entropic regularization method has computational limitations, our goal is not to demonstrate any superiority of the method.
We only intend to illustrate its computational behavior. All the test problems were run on a Pentium III 700MHZ computer with 256 MB memory using MATLAB. The initial solution $x^0$ is set at the origin for all test problems.

### 5.3.1 Finite Systems

Recalling equation (5.2), in this subsection the model algorithm and its variants are used to solve the following linear inequality system:

$$Ax \leq b,$$

(5.32)

where $A$ is an $m \times n$ matrix and $b$ is an $m$-dimensional vector.

For our examples “$A$” and “$b$” are randomly generated. We start with a solution vector $x = (x_1, x_2, ..., x_n)^T$ and a slack vector $s = (s_1, s_2, ..., s_m)^T$, where $x_i$, for $i = 1, ..., n$, and $s_j$, for $j = 1, ..., m$, are sampled from $Uniform(0, 1)$. Then, we randomly generate the coefficients $a_{ji}$, $j = 1, ..., m-1$, $i = 1, ..., n$, using $Normal(0, 1)$. A bounding constraint is added to $A$ by setting $a_{mi} = \frac{-1}{m}$, for $i = 1, ..., n$. All the elements of $A$, $x$ and $s$ are specified with four-digit precision. The right-hand side of the inequalities is defined by $b = Ax + 0.05 \times s$ to ensure that there is a nonempty feasible region. In our computational experiments, $(n, m)$ are selected from \{(20, 10), (40, 20), (20, 20), (10, 20), (20, 40)\} and 100 instances are generated for each $(n, m)$ combination. Notice that (20, 10) and (40, 20) are under-determined systems ($\frac{n}{m} > 1$), (20, 20) is well-determined system ($\frac{n}{m} = 1$), and (10, 20) and (20, 40) are over-determined systems ($\frac{n}{m} < 1$).

Four basic unconstrained optimization algorithms are implemented for searching for the minimizer of $F_p(x)$, namely, the steepest descent method, the conjugate gradient method, the modified Newton method and the quasi-Newton method [46]. The Fletcher-Reeves method is implemented for the conjugate gradient approach. The Newton direction of the modified Newton approach is defined by

$$d_x = -[\xi I + \nabla^2 F_p(x)]^{-1} \nabla F_p(x),$$

(5.33)

where $I$ is a $n \times n$ identity matrix and $\xi$ is small positive number. In the implementation, we set $\xi = 0.05$. The BFGS method is adopted for implementing quasi-Newton approach.

As we mentioned, the setting of $p$ is crucial. Three strategies for setting $p$ are used:
(1) Fixed \( p \): The value of \( p \) is a pre-selected value and it is not changed during the whole computation. In our computational experiments \( p \) is taken to be 10, 20, 40 and 80.

(2) Increasing \( p \) with respect to \( k \): In the initialization step, \( p \) is set to be 0 then \( p \) is increased by 1 in every \( \tau \) iterations. In the implementation, the following statement is added before Step 1 in the model algorithm:

\[
\text{if } \mod(k, \tau) = 0 \text{ then } p \leftarrow p + 1, \tag{5.34}
\]

where \( \tau \in \{1, 2, 5, 10\} \).

(3) Increasing \( p \) with respect to \( \|\nabla F_p(x^k)\| \): In the initialization step, we set \( p = 1 \) and select two parameters, \( \delta > 0 \) and \( \mu > 1 \). Then the following statement is added into the model algorithm before Step 1:

\[
\text{if } \|\nabla F_p(x^k)\| < \delta \text{ then } p \leftarrow \mu p.
\]

In the implementation, we set \( \delta = 0.001 \) and \( \mu \in \{2, 4, 6, 8\} \).

In some instances, the proposed method was not able to converge to a feasible solution efficiently, so we set an upper bound of 1000 iterations. The results are summarized in three sets of tables distinguished by the strategy used for setting \( p \): Table 5.1 - Table 5.4 for strategy (1), Table 5.5 - Table 5.8 for strategy (2) and Table 5.9 - Table 5.12 for strategy (3).

Several statistics are reported in the tables.

- \( p^* \): the average \( p \) value as of the final iteration (for those instances in which the method converges to a feasible solution within 1000 iterations).
- \( FD \): the number of instances in which the method converges to a feasible solution within 1000 iterations.
- \( IT \): the average number of iterations required for finding a feasible solution for those instances in which a feasible solution is actually found.
- \( CPU \): the average CPU time (seconds) required for finding a feasible solution for those instances in which a feasible solution is actually found.
• **UF**: the number of instances in which the method does not converge to a feasible solution in 1000 iterations.

• **WST**: the average time (seconds) spent (or “wasted”) on those instances for which the method does not produce a feasible solution in 1000 iterations.

• **AER**: the average violation \(l_2\)-norm associated with the violated constraints for the instances in which a feasible solution is not found by the method.

• **OV**: the number of instances in which the method encounters computational difficulty (overflow problem).

Some observations can be drawn from the computational results.

(i) There is no rule for deciding how large a \( p \) is sufficient to obtain a feasible solution. For under- or well-determined system \( p \) can be set as small as 1 (see Tables 5.9, 5.10 and 5.11). In order to solve an over-determined system, \( p \) must be set as large as possible. However, using a large \( p \) could result in an overflow problem.

(ii) The entropic regularization method can be tailored to solve under- and well-determined systems of linear inequalities without much difficulty. However, there are some computational difficulties in solving over-determined systems. In contrast to the computational results reported in Chapter 3, the entropic regularization and the surrogate constraint method have very similar computational behavior.

(iii) The implementation of quasi-Newton method (BFGS) can provide a feasible solution in fewer iterations than other implementations if no overflow problem occurs. However, BFGS is very likely to run into the overflow problem when applied to over-determined systems.

(iv) The performance of the steepest descent method and the conjugate gradient method are similar no matter which \( p \) setting strategy is applied. As expected, both need many more iterations than the modified Newton and quasi-Newton methods to converge to a feasible solution.
(v) The implementation of the modified Newton method outperforms the other three implementations. Specifically, the two strategies for increasing $p$ worked very well with the modified Newton method. Based on our results, the modified Newton method with strategy (3) is the more robust. However, the modified Newton method requires the tuning of 3 parameters ($\xi$, $\delta$ and $\mu$). Furthermore, in some problems the quality of the approximate solutions produced by the modified Newton method are not within the acceptable tolerance (see Tables 5.3 and 5.11).
Table 5.1: steepest descent method with strategy (1)

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>p</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>14.91</td>
<td>0.0261</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>20</td>
<td>100</td>
<td>22.12</td>
<td>0.0383</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>40</td>
<td>100</td>
<td>29.15</td>
<td>0.0506</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>80</td>
<td>100</td>
<td>45.11</td>
<td>0.0772</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>33.16</td>
<td>0.0673</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>49.33</td>
<td>0.0976</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>79.38</td>
<td>0.1567</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>80</td>
<td>88</td>
<td>131.4</td>
<td>0.2588</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>50.62</td>
<td>0.0916</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>75.67</td>
<td>0.1368</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>117.04</td>
<td>0.2120</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>190.62</td>
<td>0.3466</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>43</td>
<td>89.09</td>
<td>0.1768</td>
<td>57</td>
<td>1.6053</td>
<td>0.0553</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>20</td>
<td>54</td>
<td>103.28</td>
<td>0.1823</td>
<td>46</td>
<td>1.6350</td>
<td>0.0173</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>40</td>
<td>83</td>
<td>159.20</td>
<td>0.2807</td>
<td>17</td>
<td>1.7049</td>
<td>0.0057</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>80</td>
<td>96</td>
<td>257.23</td>
<td>0.4662</td>
<td>4</td>
<td>1.8753</td>
<td>0.0165</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>35</td>
<td>173.71</td>
<td>0.3794</td>
<td>65</td>
<td>2.0654</td>
<td>0.0745</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>41</td>
<td>226.80</td>
<td>0.5100</td>
<td>59</td>
<td>2.1015</td>
<td>0.0216</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>40</td>
<td>79</td>
<td>380.39</td>
<td>0.8289</td>
<td>21</td>
<td>2.1488</td>
<td>0.0184</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>80</td>
<td>80</td>
<td>554.96</td>
<td>1.2235</td>
<td>18</td>
<td>2.2176</td>
<td>0.0851</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 5.2: conjugate gradient method with strategy (1)

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>p</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>42.90</td>
<td>0.0789</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>20</td>
<td>100</td>
<td>48.09</td>
<td>0.0873</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>40</td>
<td>100</td>
<td>59.70</td>
<td>0.1082</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>80</td>
<td>100</td>
<td>68.16</td>
<td>0.1233</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>158.42</td>
<td>0.3245</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>209.53</td>
<td>0.4279</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>269.84</td>
<td>0.5510</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>80</td>
<td>88</td>
<td>333.02</td>
<td>0.6807</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>117.35</td>
<td>0.2227</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>150.82</td>
<td>0.2852</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>195.90</td>
<td>0.3706</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>267.69</td>
<td>0.5067</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>49</td>
<td>150.45</td>
<td>0.2809</td>
<td>51</td>
<td>1.8351</td>
<td>0.0594</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>20</td>
<td>57</td>
<td>130.37</td>
<td>0.2422</td>
<td>43</td>
<td>1.8369</td>
<td>0.0176</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>40</td>
<td>84</td>
<td>149.11</td>
<td>0.2774</td>
<td>16</td>
<td>1.8402</td>
<td>0.0048</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>80</td>
<td>99</td>
<td>212.08</td>
<td>0.3950</td>
<td>1</td>
<td>1.8430</td>
<td>0.0087</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>39</td>
<td>254.23</td>
<td>0.5729</td>
<td>61</td>
<td>2.2329</td>
<td>0.0785</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>44</td>
<td>326.59</td>
<td>0.7376</td>
<td>56</td>
<td>2.2383</td>
<td>0.0220</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>40</td>
<td>82</td>
<td>519.23</td>
<td>1.1730</td>
<td>18</td>
<td>2.2510</td>
<td>0.0084</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>80</td>
<td>75</td>
<td>668.01</td>
<td>1.5141</td>
<td>23</td>
<td>2.2610</td>
<td>0.0383</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 5.3: modified Newton method with strategy (1)

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>p</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>3.25</td>
<td>0.0075</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>20</td>
<td>100</td>
<td>3.92</td>
<td>0.0093</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>40</td>
<td>100</td>
<td>4.47</td>
<td>0.0101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>80</td>
<td>100</td>
<td>4.81</td>
<td>0.0109</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>5.37</td>
<td>0.0202</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>6.81</td>
<td>0.0258</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>8.30</td>
<td>0.0317</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>80</td>
<td>81</td>
<td>10.01</td>
<td>0.0412</td>
<td>7</td>
<td>4.1320</td>
<td>8.6988</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>4.99</td>
<td>0.0122</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>6.37</td>
<td>0.0156</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>7.79</td>
<td>0.0193</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>9.37</td>
<td>0.0229</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>50</td>
<td>8.56</td>
<td>0.0196</td>
<td>50</td>
<td>2.1193</td>
<td>0.0595</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>20</td>
<td>60</td>
<td>6.77</td>
<td>0.0149</td>
<td>40</td>
<td>2.0732</td>
<td>0.0175</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>40</td>
<td>90</td>
<td>7.20</td>
<td>0.0157</td>
<td>10</td>
<td>2.0790</td>
<td>0.0050</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>8.50</td>
<td>0.0180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>43</td>
<td>29.23</td>
<td>0.1150</td>
<td>57</td>
<td>3.4848</td>
<td>0.0833</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>46</td>
<td>14.96</td>
<td>0.0555</td>
<td>54</td>
<td>3.4844</td>
<td>0.0229</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>40</td>
<td>84</td>
<td>12.08</td>
<td>0.0425</td>
<td>16</td>
<td>3.4674</td>
<td>0.0055</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>80</td>
<td>97</td>
<td>15.40</td>
<td>0.0552</td>
<td>1</td>
<td>3.4650</td>
<td>0.0030</td>
<td>2</td>
</tr>
<tr>
<td>$n$</td>
<td>$m$</td>
<td>$p$</td>
<td>FD</td>
<td>IT</td>
<td>CPU</td>
<td>UF</td>
<td>WST</td>
<td>AER</td>
<td>OV</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>5.20</td>
<td>0.0117</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>20</td>
<td>100</td>
<td>5.26</td>
<td>0.0126</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>40</td>
<td>100</td>
<td>2.42</td>
<td>0.0126</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>80</td>
<td>100</td>
<td>5.48</td>
<td>0.0130</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>9.95</td>
<td>0.0338</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>10.66</td>
<td>0.0365</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>10.73</td>
<td>0.0358</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>11.09</td>
<td>0.0371</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>100</td>
<td>10.88</td>
<td>0.0246</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>100</td>
<td>11.42</td>
<td>0.0255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>40</td>
<td>100</td>
<td>11.75</td>
<td>0.0264</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>11.93</td>
<td>0.0265</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>50</td>
<td>12.76</td>
<td>0.0275</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>20</td>
<td>58</td>
<td>13.74</td>
<td>0.0295</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>40</td>
<td>86</td>
<td>14.45</td>
<td>0.0299</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>15.39</td>
<td>0.0318</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>43</td>
<td>25.16</td>
<td>0.0706</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>45</td>
<td>26.67</td>
<td>0.0703</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>40</td>
<td>86</td>
<td>28.90</td>
<td>0.0747</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>80</td>
<td>95</td>
<td>29.35</td>
<td>0.0762</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
Table 5.5: steepest descent method with strategy (2)

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>τ</th>
<th>( p^* )</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>1</td>
<td>9.36</td>
<td>100</td>
<td>9.36</td>
<td>0.0185</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>3.89</td>
<td>100</td>
<td>6.98</td>
<td>0.0145</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>5</td>
<td>1.92</td>
<td>100</td>
<td>6.99</td>
<td>0.0137</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>1.27</td>
<td>100</td>
<td>7.87</td>
<td>0.0162</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>1</td>
<td>31.34</td>
<td>100</td>
<td>31.34</td>
<td>0.0718</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>7.35</td>
<td>100</td>
<td>13.82</td>
<td>0.0327</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>5</td>
<td>2.67</td>
<td>100</td>
<td>10.96</td>
<td>0.0263</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>1.79</td>
<td>100</td>
<td>11.75</td>
<td>0.0273</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1</td>
<td>58.94</td>
<td>100</td>
<td>58.94</td>
<td>0.1191</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>10.68</td>
<td>100</td>
<td>20.69</td>
<td>0.0418</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>5</td>
<td>3.98</td>
<td>100</td>
<td>17.43</td>
<td>0.0355</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>2.31</td>
<td>100</td>
<td>17.88</td>
<td>0.0362</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>1</td>
<td>135.99</td>
<td>99</td>
<td>135.99</td>
<td>0.2658</td>
<td>1</td>
<td>1.9720</td>
<td>0.0735</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>58.89</td>
<td>99</td>
<td>117.22</td>
<td>0.2284</td>
<td>1</td>
<td>1.9530</td>
<td>0.0012</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>5</td>
<td>32.93</td>
<td>96</td>
<td>161.78</td>
<td>0.3140</td>
<td>4</td>
<td>1.9505</td>
<td>0.0042</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>24.64</td>
<td>92</td>
<td>240.45</td>
<td>0.4669</td>
<td>8</td>
<td>1.9636</td>
<td>0.0111</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>1</td>
<td>432.34</td>
<td>79</td>
<td>432.34</td>
<td>1.0626</td>
<td>21</td>
<td>2.4211</td>
<td>0.1033</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>93.24</td>
<td>92</td>
<td>185.83</td>
<td>0.4543</td>
<td>8</td>
<td>2.4031</td>
<td>0.0212</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>5</td>
<td>46.13</td>
<td>91</td>
<td>227.84</td>
<td>0.5533</td>
<td>9</td>
<td>2.3746</td>
<td>0.0147</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>26.57</td>
<td>83</td>
<td>259.17</td>
<td>0.6274</td>
<td>17</td>
<td>2.4152</td>
<td>0.0156</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.6: conjugate gradient method with strategy (2)

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>$\tau$</th>
<th>$p^*$</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>1</td>
<td>42.27</td>
<td>100</td>
<td>39.49</td>
<td>0.0817</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>16.76</td>
<td>100</td>
<td>31.22</td>
<td>0.0651</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>5</td>
<td>5.56</td>
<td>100</td>
<td>25.15</td>
<td>0.0531</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>2.78</td>
<td>100</td>
<td>22.55</td>
<td>0.0480</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>1</td>
<td>276.43</td>
<td>65</td>
<td>268.78</td>
<td>0.6152</td>
<td>22</td>
<td>2.2810</td>
<td>0.5324</td>
<td>13</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>110.44</td>
<td>95</td>
<td>214.79</td>
<td>0.4906</td>
<td>5</td>
<td>2.2794</td>
<td>0.7498</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>5</td>
<td>24.87</td>
<td>99</td>
<td>120.22</td>
<td>0.2750</td>
<td>1</td>
<td>2.2840</td>
<td>0.1051</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>9.10</td>
<td>100</td>
<td>85.23</td>
<td>0.1966</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1</td>
<td>285.64</td>
<td>90</td>
<td>270.77</td>
<td>0.5774</td>
<td>10</td>
<td>2.1320</td>
<td>0.1750</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>100.77</td>
<td>99</td>
<td>190.79</td>
<td>0.4057</td>
<td>1</td>
<td>2.1230</td>
<td>0.0895</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>5</td>
<td>19.14</td>
<td>100</td>
<td>89.92</td>
<td>0.1920</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>6.90</td>
<td>100</td>
<td>63.30</td>
<td>0.1354</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>1</td>
<td>253.24</td>
<td>97</td>
<td>227.36</td>
<td>0.4796</td>
<td>3</td>
<td>2.0930</td>
<td>0.0097</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>89.39</td>
<td>100</td>
<td>160.23</td>
<td>0.3365</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>5</td>
<td>34.99</td>
<td>100</td>
<td>156.62</td>
<td>0.3290</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>24.58</td>
<td>98</td>
<td>220.23</td>
<td>0.4622</td>
<td>2</td>
<td>2.0980</td>
<td>0.0022</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>1</td>
<td>545.00</td>
<td>7</td>
<td>517.14</td>
<td>1.3120</td>
<td>93</td>
<td>2.5336</td>
<td>0.7209</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>284.14</td>
<td>43</td>
<td>539.53</td>
<td>1.3617</td>
<td>57</td>
<td>2.5194</td>
<td>0.2195</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>5</td>
<td>76.96</td>
<td>94</td>
<td>364.60</td>
<td>0.9140</td>
<td>6</td>
<td>2.5068</td>
<td>0.0543</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>32.57</td>
<td>94</td>
<td>309.53</td>
<td>0.7749</td>
<td>6</td>
<td>2.5050</td>
<td>0.0044</td>
<td>0</td>
</tr>
<tr>
<td>$n$</td>
<td>$m$</td>
<td>$\tau$</td>
<td>$p^*$</td>
<td>FD</td>
<td>IT</td>
<td>CPU</td>
<td>UF</td>
<td>WST</td>
<td>AER</td>
<td>OV</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>----</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>1</td>
<td>1.34</td>
<td>100</td>
<td>1.34</td>
<td>0.0043</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1.00</td>
<td>100</td>
<td>1.32</td>
<td>0.0044</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>5</td>
<td>1.00</td>
<td>100</td>
<td>1.32</td>
<td>0.0044</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>1.00</td>
<td>100</td>
<td>1.32</td>
<td>0.0044</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>1</td>
<td>2.19</td>
<td>100</td>
<td>2.19</td>
<td>0.0093</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>1.01</td>
<td>100</td>
<td>1.96</td>
<td>0.0084</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>5</td>
<td>1.00</td>
<td>100</td>
<td>1.96</td>
<td>0.0089</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>1.00</td>
<td>100</td>
<td>1.96</td>
<td>0.0085</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1</td>
<td>2.33</td>
<td>100</td>
<td>2.33</td>
<td>0.0066</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>1.13</td>
<td>100</td>
<td>2.06</td>
<td>0.0063</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>5</td>
<td>1.00</td>
<td>100</td>
<td>2.05</td>
<td>0.0060</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>1.00</td>
<td>100</td>
<td>2.05</td>
<td>0.0065</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>1</td>
<td>20.88</td>
<td>100</td>
<td>20.88</td>
<td>0.0492</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>20.02</td>
<td>100</td>
<td>39.28</td>
<td>0.0933</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>5</td>
<td>19.43</td>
<td>100</td>
<td>94.19</td>
<td>0.2275</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>19.29</td>
<td>100</td>
<td>185.22</td>
<td>0.4434</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>1</td>
<td>26.19</td>
<td>100</td>
<td>26.19</td>
<td>0.1005</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>25.12</td>
<td>100</td>
<td>49.39</td>
<td>0.1928</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>5</td>
<td>24.52</td>
<td>100</td>
<td>119.31</td>
<td>0.4721</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>22.80</td>
<td>99</td>
<td>220.44</td>
<td>0.8719</td>
<td>1</td>
<td>3.9460</td>
<td>0.0022</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.8: quasi-Newton method with strategy (2)

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>τ</th>
<th>$p^*$</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>1</td>
<td>2.25</td>
<td>99</td>
<td>2.25</td>
<td>0.0103</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1.43</td>
<td>99</td>
<td>2.29</td>
<td>0.0109</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>5</td>
<td>1.05</td>
<td>99</td>
<td>2.31</td>
<td>0.0087</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>1.00</td>
<td>99</td>
<td>2.33</td>
<td>0.0098</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>1</td>
<td>3.12</td>
<td>98</td>
<td>3.12</td>
<td>0.0144</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>1.77</td>
<td>98</td>
<td>3.07</td>
<td>0.0139</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>5</td>
<td>1.06</td>
<td>98</td>
<td>3.09</td>
<td>0.0142</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>10</td>
<td>1.00</td>
<td>98</td>
<td>3.10</td>
<td>0.0158</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1</td>
<td>6.40</td>
<td>93</td>
<td>6.40</td>
<td>0.0172</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>3.25</td>
<td>93</td>
<td>6.01</td>
<td>0.0159</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>5</td>
<td>1.59</td>
<td>93</td>
<td>6.02</td>
<td>0.0167</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>1.04</td>
<td>92</td>
<td>5.84</td>
<td>0.0161</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>1</td>
<td>21.04</td>
<td>84</td>
<td>21.04</td>
<td>0.0482</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>20.11</td>
<td>89</td>
<td>39.76</td>
<td>0.0933</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>5</td>
<td>2.81</td>
<td>47</td>
<td>11.53</td>
<td>0.0275</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>10</td>
<td>1.59</td>
<td>46</td>
<td>11.00</td>
<td>0.0278</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>54</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>1</td>
<td>34.83</td>
<td>90</td>
<td>34.83</td>
<td>0.1031</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>25.58</td>
<td>90</td>
<td>50.84</td>
<td>0.1460</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>5</td>
<td>4.78</td>
<td>41</td>
<td>21.80</td>
<td>0.0645</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>59</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>10</td>
<td>2.25</td>
<td>40</td>
<td>17.80</td>
<td>0.0528</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
</tr>
</tbody>
</table>
Table 5.9: steepest descent method with strategy (3)

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>$\mu$</th>
<th>$p^*$</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>8.55</td>
<td>0.0156</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>8.55</td>
<td>0.0168</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>8.55</td>
<td>0.0160</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>8.55</td>
<td>0.0162</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>13.85</td>
<td>0.0294</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>13.85</td>
<td>0.0298</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>13.85</td>
<td>0.0287</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>13.85</td>
<td>0.0291</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>25.86</td>
<td>0.0488</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>25.86</td>
<td>0.0484</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>25.86</td>
<td>0.0486</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>25.86</td>
<td>0.0480</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>17.18</td>
<td>62</td>
<td>305.79</td>
<td>0.5379</td>
<td>38</td>
<td>1.7828</td>
<td>0.4847</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>4</td>
<td>27.87</td>
<td>68</td>
<td>297.06</td>
<td>0.5222</td>
<td>32</td>
<td>1.7920</td>
<td>0.5200</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>6</td>
<td>45.18</td>
<td>67</td>
<td>281.48</td>
<td>0.4979</td>
<td>33</td>
<td>1.7935</td>
<td>0.4913</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>8</td>
<td>27.33</td>
<td>67</td>
<td>269.34</td>
<td>0.4762</td>
<td>33</td>
<td>1.7953</td>
<td>0.4882</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>10.67</td>
<td>42</td>
<td>310.19</td>
<td>0.6683</td>
<td>58</td>
<td>2.1438</td>
<td>0.6127</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>4</td>
<td>28.24</td>
<td>51</td>
<td>373.67</td>
<td>0.8025</td>
<td>49</td>
<td>2.1464</td>
<td>0.6700</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>6</td>
<td>45.80</td>
<td>51</td>
<td>366.63</td>
<td>0.7902</td>
<td>49</td>
<td>2.1488</td>
<td>0.6554</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>8</td>
<td>33.25</td>
<td>51</td>
<td>345.67</td>
<td>0.7448</td>
<td>49</td>
<td>2.1523</td>
<td>0.6576</td>
<td>0</td>
</tr>
</tbody>
</table>

84
<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>$\mu$</th>
<th>$p^*$</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>23.14</td>
<td>0.0446</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>23.14</td>
<td>0.0442</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>23.14</td>
<td>0.0443</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>23.14</td>
<td>0.0445</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>57.50</td>
<td>0.1208</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>57.50</td>
<td>0.1205</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>57.50</td>
<td>0.1201</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>57.50</td>
<td>0.1213</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>43.38</td>
<td>0.0852</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>43.38</td>
<td>0.0844</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>43.38</td>
<td>0.0849</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>43.38</td>
<td>0.0854</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>18.71</td>
<td>76</td>
<td>282.59</td>
<td>0.5321</td>
<td>24</td>
<td>1.89</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>27.32</td>
<td>79</td>
<td>247.13</td>
<td>0.4658</td>
<td>21</td>
<td>1.89</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>45.01</td>
<td>81</td>
<td>251.16</td>
<td>0.4740</td>
<td>19</td>
<td>1.89</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>28.66</td>
<td>82</td>
<td>247.67</td>
<td>1.4691</td>
<td>18</td>
<td>1.89</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>10.55</td>
<td>49</td>
<td>310.28</td>
<td>0.6957</td>
<td>51</td>
<td>2.24</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>28.93</td>
<td>61</td>
<td>379.85</td>
<td>0.8519</td>
<td>39</td>
<td>2.25</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>38.42</td>
<td>60</td>
<td>356.38</td>
<td>0.7994</td>
<td>40</td>
<td>2.25</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>25.79</td>
<td>61</td>
<td>363.92</td>
<td>0.8144</td>
<td>39</td>
<td>2.25</td>
</tr>
</tbody>
</table>
Table 5.11: modified Newton method with strategy (3)

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>( \mu )</th>
<th>( p^* )</th>
<th>FD</th>
<th>IT</th>
<th>CPU</th>
<th>UF</th>
<th>WST</th>
<th>AER</th>
<th>OV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>1.32</td>
<td>0.0045</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>1.32</td>
<td>0.0049</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>1.32</td>
<td>0.0048</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>1.32</td>
<td>0.0044</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>1.96</td>
<td>0.0122</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>1.96</td>
<td>0.0095</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>1.96</td>
<td>0.0107</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>1.96</td>
<td>0.0112</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>100</td>
<td>2.05</td>
<td>0.0062</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>2.05</td>
<td>0.0060</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>6</td>
<td>1</td>
<td>100</td>
<td>2.05</td>
<td>0.0068</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>100</td>
<td>2.05</td>
<td>0.0065</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>25.06</td>
<td>99</td>
<td>30.12</td>
<td>0.0472</td>
<td>1</td>
<td>2.5040</td>
<td>0.1540</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>4</td>
<td>34.76</td>
<td>99</td>
<td>22.83</td>
<td>0.0474</td>
<td>1</td>
<td>2.4930</td>
<td>0.1540</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>6</td>
<td>60.49</td>
<td>99</td>
<td>21.39</td>
<td>0.0469</td>
<td>1</td>
<td>2.4930</td>
<td>0.1540</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>8</td>
<td>37.34</td>
<td>99</td>
<td>20.01</td>
<td>0.0475</td>
<td>1</td>
<td>2.5030</td>
<td>0.1540</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>33.54</td>
<td>100</td>
<td>52.96</td>
<td>0.1236</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>4</td>
<td>44.74</td>
<td>100</td>
<td>35.94</td>
<td>0.1259</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>6</td>
<td>71.70</td>
<td>100</td>
<td>34.71</td>
<td>0.1251</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>8</td>
<td>55.46</td>
<td>100</td>
<td>33.57</td>
<td>0.1251</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td>m</td>
<td>$\mu$</td>
<td>$p^*$</td>
<td>FD</td>
<td>IT</td>
<td>CPU</td>
<td>UF</td>
<td>WST</td>
<td>AER</td>
<td>OV</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>------</td>
<td>------</td>
<td>----</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>2</td>
<td>1.28</td>
<td>100</td>
<td>3.74</td>
<td>0.0139</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>4</td>
<td>1.81</td>
<td>100</td>
<td>3.74</td>
<td>0.0141</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>6</td>
<td>2.35</td>
<td>100</td>
<td>3.74</td>
<td>0.0140</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>8</td>
<td>2.89</td>
<td>100</td>
<td>3.74</td>
<td>0.0169</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>2</td>
<td>1.10</td>
<td>100</td>
<td>5.90</td>
<td>0.0289</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>4</td>
<td>1.30</td>
<td>100</td>
<td>5.90</td>
<td>0.0292</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>6</td>
<td>1.50</td>
<td>100</td>
<td>5.90</td>
<td>0.0294</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>8</td>
<td>1.70</td>
<td>100</td>
<td>5.90</td>
<td>0.0291</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>1.07</td>
<td>100</td>
<td>6.32</td>
<td>0.0171</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>4</td>
<td>1.21</td>
<td>100</td>
<td>6.32</td>
<td>0.0168</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>6</td>
<td>1.35</td>
<td>100</td>
<td>6.32</td>
<td>0.0165</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>8</td>
<td>1.49</td>
<td>100</td>
<td>6.32</td>
<td>0.0169</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>2</td>
<td>1.04</td>
<td>50</td>
<td>10.34</td>
<td>0.0263</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>4</td>
<td>6.84</td>
<td>55</td>
<td>14.36</td>
<td>0.0339</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>6</td>
<td>22.67</td>
<td>63</td>
<td>18.95</td>
<td>0.0434</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>8</td>
<td>14.22</td>
<td>63</td>
<td>18.32</td>
<td>0.0427</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>2</td>
<td>2.48</td>
<td>44</td>
<td>18.23</td>
<td>0.0535</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>4</td>
<td>9.94</td>
<td>50</td>
<td>25.32</td>
<td>0.0724</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>6</td>
<td>28.72</td>
<td>57</td>
<td>32.77</td>
<td>0.0907</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>43</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>8</td>
<td>18.32</td>
<td>59</td>
<td>35.19</td>
<td>0.0984</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>41</td>
</tr>
</tbody>
</table>
5.3.2 Infinite Systems

In this subsection, three numerical examples are reported to demonstrate the computational behavior of the entropic regularization method for solving systems of infinitely many linear inequalities. Two of the examples have bounded feasible regions while the other one has an unbounded feasible region. Depending on the properties of the Hessian of $F_p(x)$, either the Newton method or the steepest descent method is used. Simpson’s rule is implemented to do the multidimensional integration over $T$. All initial solutions are set to be the origin.

Example 1.

In this example, the feasible set of the inequalities in (5.1) forms an $n$-dimensional ball $B_r(e)$ with radius $r$ and center $e$ (vector of all ones). We consider the cases when $n = 2$ and when $n = 3$. For $n = 2$, the system of linear inequalities is

$$\cos(t)x_1 + \sin(t)x_2 \leq r + \cos(t) + \sin(t), \quad \forall \ t \in [0, 2\pi].$$

For $n = 3$, the system of linear inequalities is

$$\begin{cases} 
\cos(t)\sin(s)x_1 + \sin(t)\sin(s)x_2 + \cos(s)x_3 \\
\leq r + \cos(t)\sin(s) + \sin(t)\sin(s) + \cos(s), \\
\quad \forall \ t \in [0, 2\pi] \text{ and } \forall \ s \in [0, \pi].
\end{cases}$$

The Newton method with linear search is applied in this example and the stopping condition is $\|\nabla F_p(x)\| < 10^{-6}$. For $n = 2$, the $t$-interval, $[0, 2\pi]$, is divided into 1000 subintervals for application of Simpson’s rule; while for $n = 3$, the $(t,s)$-region, $[0, 2\pi] \times [0, \pi]$, is partitioned into $100 \times 100$ subregions.

The results are summarized in Tables 5.13 - 5.16. In the tables, $x^*$ is the approximate minimizer; $\text{IT}$, the required iterations for reaching $x^*$; and $\text{CPU}$, the CPU time for obtaining $x^*$. In this example, the proposed method can find a feasible solution within two iterations regardless of the values of $n, p$ and $r$ (the size of feasible region).
<table>
<thead>
<tr>
<th>$p$</th>
<th>$x^*$</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.999999, 1.000000)</td>
<td>2</td>
<td>0.1910</td>
</tr>
<tr>
<td>10</td>
<td>(1.000000, 0.999999)</td>
<td>2</td>
<td>0.2200</td>
</tr>
<tr>
<td>100</td>
<td>(0.999999, 0.999999)</td>
<td>2</td>
<td>0.2210</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p$</th>
<th>$x^*$</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.999939, 0.999939)</td>
<td>1</td>
<td>0.1100</td>
</tr>
<tr>
<td>10</td>
<td>(1.000000, 0.999999)</td>
<td>2</td>
<td>0.1500</td>
</tr>
<tr>
<td>100</td>
<td>(0.999999, 0.999999)</td>
<td>2</td>
<td>0.1510</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p$</th>
<th>$x^*$</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1.000000, 1.000000, 0.999999)</td>
<td>2</td>
<td>4.3460</td>
</tr>
<tr>
<td>10</td>
<td>(0.999999, 0.999999, 1.000000)</td>
<td>2</td>
<td>3.8750</td>
</tr>
<tr>
<td>100</td>
<td>(0.999999, 0.999999, 0.999999)</td>
<td>2</td>
<td>5.8090</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p$</th>
<th>$x^*$</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.999939, 0.999939, 0.999939)</td>
<td>1</td>
<td>1.9730</td>
</tr>
<tr>
<td>10</td>
<td>(0.999999, 0.999999, 0.999999)</td>
<td>2</td>
<td>3.8860</td>
</tr>
<tr>
<td>100</td>
<td>(0.999999, 0.999999, 1.000000)</td>
<td>2</td>
<td>3.8750</td>
</tr>
</tbody>
</table>
Example 2.
Consider the following system of inequalities

\[ a_1(t)x_1 + a_2(t)x_2 \leq b(t), \quad \forall \ t \in [0, 1.5\pi], \]

where

\[ a_1(t) = r(1.5\pi - t)\cos(t), \]

\[ a_2(t) = 0.5r(1.5\pi - t)\sin(t), \]

and

\[ b(t) = r(1.5\pi - t)\cos(t)(0.7 + r(1.5\pi - t)\cos(t)) + \]

\[ 0.5r(1.5\pi - t)\sin(t)(0.8 + 0.5r(1.5\pi - t)\sin(t)). \]

Figure 5.1 shows the feasible region of above system when \( r = 0.05 \).

![Figure 5.1: blank area is the feasible region of example 2 with \( r = 0.05 \)](image)

In this example Simpson’s rule uses the same settings (\( n = 2 \)) as in example 1. The results are summarized in Tables 5.17 - 5.19. In the tables, N/A indicates that the overflow problem occurred and \( \bar{x}^* \) means that \( \bar{x} \) is a feasible solution for (5.35). In this example, the proposed method is sensitive to the value of \( p \) and the size of the feasible region (\( r \)). Specifically, the smaller the feasible region of the problem, the larger the \( p \) needed by the algorithm in order to obtain a feasible solution. However, when \( r = 0.001 \), no matter how large we set \( p \), the proposed method fails to find a feasible solution.
Table 5.17: results for example 2 with \( r = 0.05 \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( x^* )</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3.663164, −39.0254)</td>
<td>3</td>
<td>0.3810</td>
</tr>
<tr>
<td>10</td>
<td>(1.128588, −3.06877)</td>
<td>5</td>
<td>0.4310</td>
</tr>
<tr>
<td>100</td>
<td>(0.875910, 0.480138)</td>
<td>6</td>
<td>0.5000</td>
</tr>
<tr>
<td>1000</td>
<td>(0.811923, 0.809212)*</td>
<td>6</td>
<td>0.6410</td>
</tr>
<tr>
<td>10000</td>
<td>(0.737843, 0.869941)*</td>
<td>11</td>
<td>1.0520</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( p )</th>
<th>( x^* )</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>50000</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>100000</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 5.18: results for example 2 with \( r = 0.01 \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( x^* )</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(14.81064, −198.966)</td>
<td>3</td>
<td>0.2800</td>
</tr>
<tr>
<td>10</td>
<td>(2.137507, −19.1526)</td>
<td>4</td>
<td>0.3700</td>
</tr>
<tr>
<td>100</td>
<td>(0.870198, −1.17170)</td>
<td>4</td>
<td>0.3800</td>
</tr>
<tr>
<td>1000</td>
<td>(0.743519, 0.621907)</td>
<td>6</td>
<td>0.5300</td>
</tr>
<tr>
<td>10000</td>
<td>(0.729420, 0.787680)</td>
<td>5</td>
<td>0.4600</td>
</tr>
<tr>
<td>50000</td>
<td>(0.716974, 0.807987)*</td>
<td>9</td>
<td>0.8020</td>
</tr>
<tr>
<td>100000</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 5.19: results for example 2 with \( r = 0.001 \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( x^* )</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(141.5155, −1997.12)</td>
<td>3</td>
<td>0.4100</td>
</tr>
<tr>
<td>10</td>
<td>(14.78419, −198.990)</td>
<td>3</td>
<td>0.4600</td>
</tr>
<tr>
<td>100</td>
<td>(2.111064, −19.1766)</td>
<td>3</td>
<td>0.4910</td>
</tr>
<tr>
<td>1000</td>
<td>(0.843750, −1.19526)</td>
<td>4</td>
<td>0.5610</td>
</tr>
<tr>
<td>10000</td>
<td>(0.717019, 0.602829)</td>
<td>5</td>
<td>0.4610</td>
</tr>
<tr>
<td>100000</td>
<td>(0.704351, 0.782190)</td>
<td>5</td>
<td>0.8210</td>
</tr>
<tr>
<td>500000</td>
<td>(0.703238, 0.797045)</td>
<td>6</td>
<td>0.6310</td>
</tr>
</tbody>
</table>

91
Example 3.

Consider the following system of inequalities:

\[-x_1 - tx_2 \leq \frac{-1}{2-t}, \quad \forall t \in [0, 1]. \tag{5.39}\]

Notice that the feasible region is unbounded, and that $\nabla^2 F_p(x)$ is singular for $x \in \mathbb{R}^2$. We applied the steepest descent method to this example. With $p = 1$, the proposed method returns a feasible solution $(739.6398, 399.7550)$ in one iteration and encounters the overflow problem $(F_p(x) = -\infty)$ at the next iteration. When $p \geq 2$, the algorithm returns a feasible solution in one iteration and no improvement is made at subsequent iterations, i.e., the algorithm returns the same solution as at the first iteration. The results are summarized in Table 5.20.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$x^*$</th>
<th>IT</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(739.6398, 399.7550)$</td>
<td>1</td>
<td>0.0800</td>
</tr>
<tr>
<td>10</td>
<td>$(74.3698, 62.4638)$</td>
<td>1</td>
<td>0.0700</td>
</tr>
<tr>
<td>100</td>
<td>$(7.8850, 7.8028)$</td>
<td>1</td>
<td>0.0500</td>
</tr>
<tr>
<td>500</td>
<td>$(1.9767, 1.9728)$</td>
<td>1</td>
<td>0.0400</td>
</tr>
<tr>
<td>10000</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

5.4 Remarks

In this chapter we have studied the entropic regularization method for solving linear inequality systems. In theory, the proposed method is a unified and flexible method for solving systems of linear inequalities, i.e., it can be modified easily to solve finitely or infinitely many linear inequalities using any commercial or public, unconstrained nonlinear optimizer.

When we applied the method to solve systems of finitely many linear inequalities, the experiments showed that the entropic regularization method could provide promising results for under- and well-determined systems. However, for general use a clever procedure for adapting $p$ is required in order to avoid the overflow problem. Furthermore, there is no guarantee for the performance of the proposed method.
The proposed method also provides an alternative for solving systems of infinitely many linear inequalities. It does not require a global optimization procedure. Instead, a multi-dimensional integration is required. The computational results indicate that it offers promise as a solution method. However, the overflow problem may occur.
Chapter 6

Extensions and Related Applications

6.1 System of Linear Equations and Inequalities

The EP algorithm presented in Chapter 3 can be modified easily to directly solve a system of linear equations and linear inequalities. Consider the following system

$$
\begin{align*}
A_1x &= b_1 \\
A_2x &\leq b_2 \\
x &\geq 0,
\end{align*}
$$

where $A_1$ is an $m_1 \times n$ matrix, $A_2$ is an $m_2 \times n$ matrix, $b_1$ is an $m_1$-dimensional vector, and $b_2$ is an $m_2$-dimensional vector.

Let

$$
\hat{A} = \begin{bmatrix} A_1 & 0 \\ A_2 & I_{m_2} \end{bmatrix},
$$

and change Step 3 of Algorithm 1 as follows:

**Step 3:** While $A_1x^k - b_1 \neq 0$ or $A_2x^k - b_2 > 0$

- Calculate a movement direction vector $d_w^k$.
- Update $w^{k+1}$ with $w^k + d_w^k$.
- Calculate $x_i^{k+1} = \exp\{\sum_{j=1}^n a_{ji}w_j^{k+1} - 1\}$, for $i = 1, ..., n$.
- Update $k$ with $k + 1$. 

94
Then the modified algorithm can be directly applied to solve system (6.1).

In numerical computation, it is usually impossible to achieve \( A_1 x^k - b_1 = 0 \). In practice, we typically settle for getting an approximate solution within some tolerance. For a given tolerance \( \varepsilon > 0 \), we can replace \( A_1 x^k - b_1 \neq 0 \) with \( \| A_1 x^k - b_1 \| < \varepsilon \). Furthermore, if \( A_1 \) has full row-rank and, \( A_1 \) and \( A_2 \) both satisfy the interior point assumption (there exists an \( x_0 \in R^n \) such that \( A_1 x_0 = b_1 , A_2 x_0 < b_2 \) and \( x_0 > 0 \)), then the conclusion of Theorem 4 is still valid, because \( g(\rho)(w) \) is a strictly concave function.

This method for solving systems of linear equations and inequalities offers some advantages over other methods. Many of the classical methods, for example, the Gauss-Seidel, SOR and GMRES methods \cite{41\cite{66}} may not converge for any given \( A_1 \) and are not able to handle inequalities. In contrast to some other recently proposed methods, such as the surrogate constraint method \cite{69}\cite{54} and Newton-like method \cite{58}, we do not need to represent the equations, \( A_1 x = b_1 \), by two systems of inequalities, i.e.,

\[
\begin{align*}
A_1 x & \leq b_1 \\
-A_1 x & \leq -b_1,
\end{align*}
\]  

(6.2)

Notice that, the entropic regularization approach proposed in Chapter 5 can also be used to solve system (6.1). However, this approach requires us to convert \( A_1 x = b_1 \) into system (6.2) which will increase the problem dimension.

### 6.2 ACCPM for Variational Inequality Problems

Recently, the analytic center cutting plane method has been applied to solve variational inequality problems \cite{26}. An extensive survey on the variational inequality problem can be found in the paper of Harker and Pang \cite{32}. In this subsection, we briefly introduce the variational inequality problem with some of its properties. Then a scheme for using the analytic center cutting plane method to solve variational inequalities is described. Two numerical examples are included to illustrate the potential of the proposed entropic analytic center cutting plane method.

**Definition 3** The Variational Inequality Problem, VI\((F, \Omega)\).

Let \( \Omega \) be a convex compact subset of \( R^n \) and let \( F \) be a continuous function from \( \Omega \) into \( R^n \). A vector \( x^* \in R^n \) is a solution of VI\((F, \Omega)\) if and only if it satisfies the
system of nonlinear inequalities

\[ F(x^*)^T (x - x^*) \geq 0, \forall x \in \Omega. \] (6.3)

It is well-known that under the assumptions stated in Definition 3 the solution set \( \Omega^* \subset \Omega \) of VI(\( F, \Omega \)) is a nonempty and compact set [43].

**Definition 4** \( F \) is monotone on \( \Omega \) if and only if, \( \forall x, y \in \Omega, \)

\[ (F(x) - F(y))^T (x - y) \geq 0; \]

pseudo-monotone on \( \Omega \) if and only if, \( \forall x, y \in \Omega, \)

\[ F(y)^T (x - y) \geq 0 \Rightarrow F(x)^T (x - y) \geq 0; \]

and strongly monotone on \( \Omega \) if and only if, \( \exists \alpha > 0 \) s.t. \( \forall x, y \in \Omega, \)

\[ (F(x) - F(y))^T (x - y) \geq \alpha \|x - y\|. \]

It follows from the definitions that strongly monotone functions are monotone, and monotone functions are pseudo-monotone. In addition, if \( F \) is strongly monotone on \( \Omega \) then there exists a unique solution for VI(\( F, \Omega \)) [43].

The idea of adopting the cutting plane method for solving variational inequalities can be found in [49]. From [49], we have

**Proposition 1** If \( F \) is pseudo-monotone on \( \Omega \), then \( x^* \in \Omega \) solves VI(\( F, \Omega \)) if and only if \( F(x)^T (x - x^*) \geq 0, \forall x \in \Omega. \)

The solution set \( \Omega^* \) can be characterized as the intersection of all valid cutting planes for finding \( \Omega^* \). We next outline an algorithm proposed by Goffin et al. [26] for identifying an \( \varepsilon \)-solution of VI(\( F, \Omega \)) when \( \Omega = \{ x \in \mathbb{R}^n | Ax \leq b \} \) and \( F \) is pseudo-monotone.
Algorithm 8: ACCPM for solving VI

Step 0: Set $k = 0$, $A^k = A$ and $b^k = b$.

Step 1: Calculate the analytic center $x^k$ of $\Omega^k = \{x \in R^n | A^k x \leq b^k\}$.

Step 2: Compute $g(x^k) = \min_{x \in \Omega^k} F(x^k)^T (x - x^k)$.

Step 3: If $g(x^k) \geq -\varepsilon$ then stop. Otherwise,

Step 4: Let $A^{k+1} = \left[ \begin{array}{c} A^k \\ F(x^k)^T \end{array} \right]$, $b^{k+1} = \left[ \begin{array}{c} b^k \\ F(x^k)^T x^k \end{array} \right]$.

Set $k = k + 1$ and return to Step 1.

Notice that, at each iteration the new cutting plane generated by the algorithm is $H^k = \{x \in R^n | F(x^k)^T x = F(x^k)^T x^k\}$. Furthermore, if $F$ is strongly monotone, then the Steps 2 and 3 can be replaced by the following stopping criterion:

$$\|x^k - x^{k-1}\| \leq \varepsilon,$$

where $\varepsilon$ is a sufficiently small positive number.

6.2.1 Entropic ACCPM for VI($F, \Omega$) and Examples

In this subsection we propose to replace the analytic center with the entropic analytic center in Algorithm 8 to create an entropic analytic center cutting plane method (EACCPM) for solving variational inequality problems. Two examples with $F(x)$ being continuous and strongly monotone are used to illustrate the potential of the proposed algorithm. The first example is a two-dimensional problem. In this case we can draw pictures to show the entropic analytic centers and the cuts generated by the algorithm. The second example is based on one of examples given in [65]. In order to compare ACCPM and EACCPM, we implement the Dual Newton Procedure for calculating the analytic centers in ACCPM [72].

Example 1 (Linear Symmetric Mapping)

In this example, we consider two functions, $F_1$ and $F_2$, such that one of the solutions is an interior point and the other one is on the boundary of the convex polyhedral set $\Omega$ defined by

$$A = \left[ \begin{array}{cc} \frac{4}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{4}{3} \end{array} \right], \quad b = \left[ \begin{array}{c} 1 \\ 1 \end{array} \right].$$
Let
\[
F_1(x) = \begin{bmatrix} 2 & -2 \\ -2 & 5 \end{bmatrix} x + \begin{bmatrix} \frac{1}{2} \\ \frac{1}{4} \end{bmatrix},
\]
and
\[
F_2(x) = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} x + \begin{bmatrix} -2 \\ -2 \end{bmatrix}.
\]

The solutions of VI($F_1, \Omega$) and VI($F_2, \Omega$) are $x_1^* = (0.5, 0.25)$ and $x_2^* = (0.857143, 0.821429)$, respectively. The iterative results obtained with ACCPM and EAACPM are listed in Table 6.1, for $F_1$ and Table 6.2, for $F_2$. The first few iterations are also depicted in Figure 6.1 and 6.2 for $F_1$ and Figure 6.3 and 6.4 for $F_2$. Because of computational difficulties which occurred in matrix inversion in the dual Newton procedure, ACCPM stopped after the 12th iteration for $F_1$ with $x_1 = (0.516501, 0.252374)$, and after the 13th iteration for $F_2$ with $x_2 = (0.846124, 0.826938)$. However, no computational difficulty occurred with EAACPM, and it continued and returned $x_1^{18} = (0.500000, 0.250000)$ with $\|x^{18} - x^*\| < 10^{-6}$ for $F_1$, and $x_2^{19} = (0.857151, 0.821424)$ with $\|x^{19} - x^*\| < 10^{-5}$ for $F_2$. The results show that, for both functions, the EAACPM Algorithm returns an approximate solution with $\|x^k - x^*\| < 10^{-3}$ in 12 iterations, while the ACCPM can achieve an approximate solution to only $10^{-1}$ and $10^{-2}$ for $F_1$ and $F_2$, respectively.

**Example 2 (Nonlinear Asymmetric Mapping)**

In this example we consider $F(x) = Mx + D(x) + q$, where

\[
M = \begin{bmatrix} 0.726 & -0.949 & 0.266 & -1.193 & -0.504 \\ 1.645 & 0.678 & 0.333 & -0.217 & -1.443 \\ -1.016 & -0.225 & 0.769 & 0.934 & 1.007 \\ 1.063 & 0.567 & -1.144 & 0.550 & -0.548 \\ -0.259 & 1.453 & -1.073 & 0.509 & 1.026 \end{bmatrix},
\]

\[
D(x) = 3 \times \begin{bmatrix} \arctan(x_1 - 0.2) \\ \arctan(x_2 - 0.2) \\ \arctan(x_3 - 0.2) \\ \arctan(x_4 - 0.2) \\ \arctan(x_5 - 0.2) \end{bmatrix}, \quad \text{and } q = \begin{bmatrix} 0.3308 \\ -0.1992 \\ -0.2938 \\ -0.0976 \\ -0.3312 \end{bmatrix},
\]

and the convex polyhedron $\Omega$ is defined by

\[
A = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \text{ and } b = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.
\]
Here, $M$ is a asymmetric positive definite matrix. The arctan function in $D(x)$ provides the nonlinear mapping for $F(x)$. The solution of $\text{VI}(F, \Omega)$ is $x^* = (0.2, 0.2, 0.2, 0.2, 0.2, 0.2)^T$. The iterative results are shown in Table 6.3 for ACCPM, and in Table 6.4 for EACCPM, respectively. Encountering the same computational difficulties as with in example 1, the ACCPM stopped at the 26$\text{th}$ iteration. But the EACCPM continued to return a solution for a prespecified 50 iterations. In this example, the EACCPM and ACCPM algorithm both return an approximate solution with $\|x^k - x^*\| \leq 10^{-2}$ in 26 iterations. The gaps between $x^*$, the solution of $\text{VI}(F, \Omega)$, and $x^k$ returned by ACCPM, and between $x^*$ and $x^k$ returned by EACCPM are depicted in Figure 6.5. Notice that, in particular, the performance of EACCPM is much more stable than that of ACCPM. We also note that by using the algorithm in [65], a solution for the similar problem was obtained with $\|x^k - x^*\| \leq 10^{-6}$ in about 6 iterations. However, in each iteration, the computationally expensive LCP-Algorithm of Lemke was used to solve the linearized variational inequality problem.

Finally we note that relaxing the binding constraint in $\Omega$, $\sum_{i=1}^{5} -x_i \leq -1$, to define a new feasible set $\tilde{\Omega} = \{x \in \mathbb{R}^5| \sum_{i=1}^{5} 0.5x_i \leq 1, x \geq 0\}$ does not change the solution, i.e., $x^* = (0.2, 0.2, 0.2, 0.2, 0.2)^T$ solves both $\text{VI}(F, \Omega)$ and $\text{VI}(F, \tilde{\Omega})$. However, in this case, $x^*$ is a interior point of $\tilde{\Omega}$. Figure 6.6 is analogous to Figure 6.5. The performance of EACCPM is still more stable than that of ACCPM.

### 6.3 Remarks

In this chapter, the EP Algorithm proposed in Chapter 3 has been extended to solve systems of linear equations and inequalities. The proposed method can treat equations and inequalities simultaneously without increasing the problem size.

We also examined the use of the entropic analytic center for solving variational inequality problems with the EACCPM algorithm. The results from the two numerical examples we used to test EACCPM suggest that there is potential for the use of entropic analytic centers in solving variational inequalities.
### Table 6.1: numerical results for ACCMP and EACCMP for $F_1$ in Example 1

<table>
<thead>
<tr>
<th>Iteration</th>
<th>ACCMP</th>
<th>EACCMP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>0</td>
<td>0.447606</td>
<td>0.394007</td>
</tr>
<tr>
<td>1</td>
<td>0.544245</td>
<td>0.190651</td>
</tr>
<tr>
<td>2</td>
<td>0.454663</td>
<td>0.287856</td>
</tr>
<tr>
<td>3</td>
<td>0.556150</td>
<td>0.262473</td>
</tr>
<tr>
<td>4</td>
<td>0.339111</td>
<td>0.153121</td>
</tr>
<tr>
<td>5</td>
<td>0.466136</td>
<td>0.239916</td>
</tr>
<tr>
<td>6</td>
<td>0.533006</td>
<td>0.280086</td>
</tr>
<tr>
<td>7</td>
<td>0.496865</td>
<td>0.220787</td>
</tr>
<tr>
<td>8</td>
<td>0.527047</td>
<td>0.280498</td>
</tr>
<tr>
<td>9</td>
<td>0.497149</td>
<td>0.240852</td>
</tr>
<tr>
<td>10</td>
<td>0.498315</td>
<td>0.256402</td>
</tr>
<tr>
<td>11</td>
<td>0.498355</td>
<td>0.256421</td>
</tr>
<tr>
<td>12</td>
<td>0.516501</td>
<td>0.252374</td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>16</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>17</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>18</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table 6.2: numerical results for ACCMP and EACCMP for $F_2$ in Example 1

<table>
<thead>
<tr>
<th>Iteration</th>
<th>ACCMP $x_1$</th>
<th>ACCMP $x_2$</th>
<th>EACCMP $x_1$</th>
<th>EACCMP $x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.447606</td>
<td>0.394007</td>
<td>0.494521</td>
<td>0.553360</td>
</tr>
<tr>
<td>1</td>
<td>0.637710</td>
<td>0.616977</td>
<td>0.616457</td>
<td>0.641522</td>
</tr>
<tr>
<td>2</td>
<td>0.763908</td>
<td>0.722649</td>
<td>0.691452</td>
<td>0.752581</td>
</tr>
<tr>
<td>3</td>
<td>0.815039</td>
<td>0.780789</td>
<td>0.746801</td>
<td>0.752581</td>
</tr>
<tr>
<td>4</td>
<td>0.861789</td>
<td>0.792711</td>
<td>0.788142</td>
<td>0.786130</td>
</tr>
<tr>
<td>5</td>
<td>0.731506</td>
<td>0.867834</td>
<td>0.819260</td>
<td>0.808108</td>
</tr>
<tr>
<td>6</td>
<td>0.829978</td>
<td>0.822135</td>
<td>0.841257</td>
<td>0.819168</td>
</tr>
<tr>
<td>7</td>
<td>0.884816</td>
<td>0.800459</td>
<td>0.854420</td>
<td>0.821218</td>
</tr>
<tr>
<td>8</td>
<td>0.848535</td>
<td>0.820669</td>
<td>0.857463</td>
<td>0.821204</td>
</tr>
<tr>
<td>9</td>
<td>0.866549</td>
<td>0.814353</td>
<td>0.856439</td>
<td>0.821780</td>
</tr>
<tr>
<td>10</td>
<td>0.844118</td>
<td>0.826513</td>
<td>0.858631</td>
<td>0.820684</td>
</tr>
<tr>
<td>11</td>
<td>0.859836</td>
<td>0.819105</td>
<td>0.856396</td>
<td>0.821802</td>
</tr>
<tr>
<td>12</td>
<td>0.852398</td>
<td>0.823324</td>
<td>0.856396</td>
<td>0.821802</td>
</tr>
<tr>
<td>13</td>
<td>0.846124</td>
<td>0.826938</td>
<td>0.857311</td>
<td>0.821345</td>
</tr>
<tr>
<td>14</td>
<td>N/A</td>
<td>N/A</td>
<td>0.857018</td>
<td>0.821491</td>
</tr>
<tr>
<td>15</td>
<td>N/A</td>
<td>N/A</td>
<td>0.857207</td>
<td>0.821396</td>
</tr>
<tr>
<td>16</td>
<td>N/A</td>
<td>N/A</td>
<td>0.857097</td>
<td>0.821451</td>
</tr>
<tr>
<td>17</td>
<td>N/A</td>
<td>N/A</td>
<td>0.857170</td>
<td>0.821415</td>
</tr>
<tr>
<td>18</td>
<td>N/A</td>
<td>N/A</td>
<td>0.857127</td>
<td>0.821436</td>
</tr>
<tr>
<td>19</td>
<td>N/A</td>
<td>N/A</td>
<td>0.857151</td>
<td>0.821424</td>
</tr>
</tbody>
</table>
Table 6.3: numerical results of ACs for $F$ in Example 2.

| Iteration $k$ | $x_1$      | $x_2$      | $x_3$      | $x_4$      | $x_5$      | $|x^k - x^*|$ |
|-------------|------------|------------|------------|------------|------------|-------------|
| 0           | 0.351903   | 0.351903   | 0.351903   | 0.351903   | 0.351903   | 0.339667    |
| 5           | 0.335359   | 0.166612   | 0.256299   | 0.223073   | 0.193145   | 0.152269    |
| 10          | 0.259024   | 0.156769   | 0.229542   | 0.225268   | 0.202341   | 0.082882    |
| 15          | 0.227064   | 0.183176   | 0.177888   | 0.216348   | 0.227239   | 0.050136    |
| 20          | 0.189808   | 0.199131   | 0.218255   | 0.206822   | 0.202227   | 0.022122    |
| 26          | 0.203218   | 0.200770   | 0.203305   | 0.200844   | 0.203116   | 0.005683    |

Table 6.4: numerical results of EACs for $F$ in Example 2.

| Iteration $k$ | $x_1$      | $x_2$      | $x_3$      | $x_4$      | $x_5$      | $|x^k - x^*|$ |
|-------------|------------|------------|------------|------------|------------|-------------|
| 0           | 0.284373   | 0.284373   | 0.284373   | 0.284373   | 0.284373   | 0.118663    |
| 10          | 0.228635   | 0.173526   | 0.208915   | 0.225958   | 0.218247   | 0.051097    |
| 20          | 0.207348   | 0.202364   | 0.206002   | 0.198443   | 0.203222   | 0.010412    |
| 30          | 0.202830   | 0.201515   | 0.202679   | 0.201354   | 0.200257   | 0.004402    |
| 40          | 0.202463   | 0.200367   | 0.200880   | 0.200822   | 0.200879   | 0.002902    |
| 50          | 0.201347   | 0.200745   | 0.200873   | 0.200294   | 0.200588   | 0.001906    |
Figure 6.1: feasible set, ACs, and cuts for $F_1(x)$ in Example 1

Figure 6.2: feasible set, entropic ACs, and cuts for $F_1(x)$ in Example 1
Figure 6.3: feasible set, ACs, and cuts for $F_2(x)$ in Example 1

Figure 6.4: feasible set, entropic ACs, and cuts for $F_2(x)$ in Example 1

104
Figure 6.5: $\|x^k - x^*\|$ vs $k$ for ACCPM and EACCPM in Example 2

Figure 6.6: $\|x^k - x^*\|$ vs $k$ for ACCPM and EACCPM in Example 2 with binding constraint removed
Chapter 7

Summary and Future Research Directions

7.1 Summary of Dissertation

In this dissertation, we have developed new approaches for solving the following system of linear inequalities:

\[
\begin{align*}
\{ & f_1(t)^T x \leq g_1(t), \\
& \quad \vdots \quad \forall t \in T, \\
& f_m(t)^T x \leq g_m(t),
\end{align*}
\]

where \( x \in \mathbb{R}^n \), \( T \) is a compact subset of \( \mathbb{R}^l \) with \( l \in \mathbb{N} \), and \( f_j : T \to \mathbb{R}^n \) and \( g_j : T \to \mathbb{R} \), for \( j = 1, \ldots, m \), are real-valued functions on \( T \).

In Chapter 3, system (7.1) is specified to be a system of finitely many linear inequalities with nonnegative decision variables which can be formulated as

\[
\begin{align*}
\{ & Ax \leq b \\
& x \geq 0,
\end{align*}
\]

where \( A \) is an \( m \times n \) matrix and \( b \) is an \( m \)-dimensional vector. In this case, \( |T| = 1 \), and \( b_j = g_j(t) \) and \( A_j = f_j(t)^T \) (the \( j \)th row vector of \( A \)), for \( j = 1, \ldots, m \). We introduce an unconstrained convex programming approach for solving (7.2) which yields an exact feasible solution if one exists and has the ability to detect inconsistency when it is present. Specifically, we propose an entropic perturbation algorithm (the EP
Algorithm) based on the Newton method. This algorithm can be easily implemented and no parameter setting is needed. In theory, the algorithm converges globally with a quadratic rate. In practice, the computational experience is quite encouraging, especially for solving over-determined systems.

In Chapter 4, the focus is on the following type of linear inequalities:

\[
\begin{aligned}
  & f_1(t)^T x \leq g_1(t), \\
  & \quad \vdots \quad \forall \ t \in T \\
  & f_m(t)^T x \leq g_m(t), \\
  & x \geq 0,
\end{aligned}
\]  

(7.3)

where \( f_j \) and \( g_j \), for \( j = 1, ..., m \), are continuous functions on a convex compact set \( T \) with \(|T| = \infty\). We study the analytic center cutting plane method (ACCPM) for solving system (7.3). Instead of using a global optimizer to identify the most violated constraints, we propose a relaxed cutting plane scheme to form cuts. The potential advantage of the relaxed cut scheme is clearly illustrated by examples.

Motivated by the definition of analytic center and the characteristics of the solution returned by the algorithm proposed in Chapter 3 (EP Algorithm), we define a new analytic center, namely, the entropic analytic center. We examine the use of the entropic analytic center as a replacement for the analytic centers in ACCPM to solve (7.3). The computational results show its potential.

Also in Chapter 4, we study one feasibility issue for system (7.3), namely, the maximum feasibility problem. This problem is to determine the largest portion of \( T \), which is restricted to be \([0,1]\), such that system (7.3) is consistent. We propose an algorithm based on ACCPM to solve the maximum feasibility problem. This approach employs a root-finding procedure instead of the commonly used global optimization procedure for identifying violated constraints.

In Chapter 5, we study the entropic regularization method for solving system (7.1). Specifically, we consider the following linear inequality system:

\[
  f(t)^T x \leq g(t), \ \forall \ t \in T,
\]  

(7.4)

where \( f(t) = (f_1(t), ..., f_n(t))^T \), and \( g(t) \) and \( f_i(t), \ i = 1, ..., n \), are continuously differentiable functions on a compact set \( T \). We propose an optimization approach
based on the entropic regularization method for solving system (7.4). Unlike the approaches presented in Chapters 3 and 4, this approach provides an approximate solution regardless of the consistency of (7.4).

The computational results for the entropic regularization method look promising for under- and well-determined systems ($|T| < \infty$). However, in general, a clever procedure for adapting the parameter $p$ is required to avoid an overflow problem. The proposed approach also provides an alternative for solving systems of infinitely many linear inequalities ($T$ is convex and $|T| = \infty$). This approach does not require a global optimization procedure for identifying the most violated constraint over $T$. Instead, a multi-dimensional integration over $T$ is needed.

In Chapter 6, the EP Algorithm proposed in Chapter 3 is extended to solve a system of linear equations and linear inequalities. This method can treat equations and inequalities simultaneously without increasing the problem size.

Also in Chapter 6, we further examine of the use of the entropic analytic centers. The entropic analytic center cutting plane method (EACCPM) is used to solve strongly monotone variational inequality problems. The computational results show that the EACCPM is a promising approach.

### 7.2 Future Research Directions

Future directions of this research can be described as follows:

1. We have demonstrated the computational behavior of the EP Algorithm for solving mid-size dense problems. However, it is necessary, in practice, to have a special implementation for solving a large-size, sparse, and highly over-determined system of linear inequalities. Notice that most of the run time of the EP Algorithm is spent on calculating the inverse of the Hessian matrix whose dimension is equal to the number inequalities. However, it is unnecessary to consider all inequalities at the same time, especially for solving a highly over-determined system. A possible efficient numerical method is to use the entropic analytic center cutting plane method together with sparse Cholesky Factorization.
2. Notice that it is not efficient to compute a new entropic analytic center from a “cold start” initial solution ($w^0$ is the origin in our case) after new cuts are introduced at each iteration of EACCPM. In order to reduce the computational effort of the EACCPM, a good “warm start” procedure for computing the entropic analytic center should be studied.

3. We have illustrated the potential of using the entropic analytic centers in solving linear inequality systems and variational inequality problems. However, a convergence result is still missing for the EACCPM algorithm. In the hope of being able to obtain a convergence result, further investigations of the analytic and geometric properties of entropic analytic centers should be conducted.

4. The computational results show that the entropic regularization method does not seem to be a very promising approach for solving system (7.4). However, unlike the EP Algorithm or the ACCPM, this approach can provide an approximate solution regardless of whether the system is feasible. In practice, we are also interested in getting an approximate solution within some tolerance in least computational effort. Hence, the quality of the approximate solution produced by the entropic regularization approach should be studied. Research on a more efficient optimizer that has the ability to ensure the quality of the approximate solution could be a worthy direction.
Bibliography


113


114


