


Spring 2009

# Interior Point Methods and Kernel Functions of a Linear Programming Problem

Latriece Y. Tanksley

# INTERIOR POINT METHODS AND KERNEL FUNCTIONS OF A LINEAR PROGRAMMING PROBLEM

by

LATRIECE Y. TANKSLEY

(Under the Direction of Goran Lesaja)

## ABSTRACT

In this thesis the Interior – Point Method (IPM) for Linear Programming problem (LP) that is based on the generic kernel function is considered.

The complexity (in terms of iteration bounds) of the algorithm is first analyzed for a class of kernel functions defined by (3-1). This class is fairly general; it includes classical logarithmic kernel function, prototype self-regular kernel function as well as non-self-regular functions, thus it serves as a unifying frame for the analysis of IPM. Historically, most results in the theory of IPM are based on logarithmic kernel functions while other two classes are more recent. They were considered with the intention to improve theoretical and practical performance of IPMs. The complexity results that are obtained match the best known complexity results for these methods.

Next, the analysis of the IPM was summarized and performed for three more kernel functions. For two of them we again matched the best known complexity results.

The theoretical concepts of IPM were illustrated by basic implementation for the classical logarithmic kernel function and for the parametric kernel function both described in (3-1). Even this basic implementation shows potential for a good performance. Better implementation and more numerical testing would be necessary to draw more definite

conclusions. However, that was not the goal of the thesis, the goal was to show that IPM with kernel functions different than classical logarithmic kernel function can have best known theoretical complexity.

INDEX WORDS: Interior Point Methods, Linear Programming, Kernel Functions, Log Barrier Functions

# INTERIOR POINT METHODS AND KERNEL FUNCTIONS OF A LINEAR PROGRAMMING PROBLEM

by

LATRIECE Y. TANKSLEY

B.S., Savannah State University, 2000

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial

Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

2009

INTERIOR POINT METHODS AND KERNEL FUNCTIONS

OF A LINEAR PROGRAMMING PROBLEM

Major Professor: Goran Lesaja

Committee: Billur Kaymakcalan

Scott Kersey

Yan Wu

Electronic Version Approved:

May 2009

# ACKNOWLEDGEMENTS

Mathematics has always been of great interest to me. I cannot begin to express the joy and pleasure that I've experienced studying at Georgia Southern University under the direction of the great faculty here. Dr. Goran Lesaja has played an intricate role in my graduate research. I could not have made it to this point without his advisement and knowledge. I would like to thank Dr. Yan Wu and Dr. Billur Kaymakcalan for serving on the graduate committee. I have had the pleasure of taking previous classes from both instructors and want to thank them for their encouragement throughout my academic career. I would also like to thank Dr. Scott Kearsey for serving on the graduate committee. I have not had the pleasure of knowing him as well as the others, but I appreciate his interest in my graduate work. I would also like to thank graduate student Jie Chen and Segio Valdrig for their generous help with the implementation of the algorithms in MatLab. I would like to thank my parents, Algene and Dorothy Tanksley, for their support throughout this long journey. They have been great role models in my life and this accomplishment is a direct result of their hard work and commitment towards my academic success. Finally, I would like to thank the math professors at Savannah State University who first encouraged me and guided me through undergraduate studies in math. I am eternally grateful to them for their leadership and interest in my studies.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS .....6

LIST OF FIGURES .....8

LIST OF TABLES .....9

NOMENCLATURE .....10

CHAPTER

1 INTRODUCTION .....12

2 INTERIOR POINT METHODS BASED ON KERNAL FUNCTIONS .....18

3 ANALYSIS OF THE ALGORITHM FOR A CLASS OF KERNEL FUNCTIONS . 27

4 ANALYSIS OF THE ALGORITHM FOR ADDITIONAL KERNAL FUNCTIONS50

5 NUMERICAL RESULTS .....68

6 CONCLUSION .....74

REFERENCES .....76

APPENDICES

APPENDIX A .....77

APPENDIX B .....82

# LIST OF FIGURES

Figure 1: Graphical Interpretation of IPM ..... 21

Figure 2: Generic Primal-Dual Interior-Point Algorithm for Linear Optimization ..... 26

Figure 3: IPM Based on generic kernel function ..... 70

# LIST OF TABLES

Table 1: Wyndor Glass Company Data ..... 14

Table 2: Complexity results for long-step methods ..... 64

Table 3: Complexity results for short-step methods ..... 65

Table 4: Numerical results for Example ..... 69

Table 5: Numerical results for randomly generated "small problems"
with dimension less than 10 ..... 70

Table 6: Numerical results for randomly generated problems
with dimension 200x300 ..... 71

Table 7: Numerical results for randomly generated problem
with dimension 400x700 ..... 71

Table 8: More Numerical Results ..... 72

Table 9: More Numerical Results ..... 72

# NOMENCLATURE

| | |
|---|---|
| $R^n$ | Euclidean n dim space |
| $R^n{}_+$ | All vectors of $R^n$ with nonnegative components |
| $x \in A$ | Element $x$ belongs to set $A$ |
| $\|x\|$ | A Euclidian norm of vector $x \in R^n$, $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^n}$ |
| $\mu \to 0,$ | $\mu$ converges to 0 |
| $f : R^n \to R$ | A function with n variables |
| $\nabla f$, $\nabla^2 f$ | A gradient and a Hessian of $f$ |
| $F : R^n \to R^m$ | A vector valued function |
| $\nabla F$ | A Jacobian of $F$ |
| $X = diag(x)$ | A diagonal matrix that has components of the vector $x$ on the main diagonal and zeros everywhere else |
| $xs$, $\frac{x}{s}$, $x^{-1}$ | Component wise operations (product, division, inverse) of vectors $x, s \in R^n$. For example, $xs = (x_1 s_1, \cdots, x_n s_n)$. |
| $o(f(n))$ | There exist a constant $C$ and a function $g(n)$ such that $f(n) \geq Cg(n)$ ("small o" notation). |

| | |
|---|---|
| $O(f(n))$ | There exist a constant $C$ and a function $g(n)$ such that $f(n) \leq Cg(n)$ |
| $\Omega(f(n))$ | There exist constants $C, K$ and a function $g(n)$ such that $Cg(n) \leq f(n) \leq Kg(n)$ ("big $\Omega$" notation). |

# CHAPTER 1

# INTRODUCTION

## Linear Programming Model

The mathematical model of linear programming is useful in solving a wide range of problems in industry, business, science and government. This is by far the most used optimization model.

These problems and their linear programming models can often be complex as they consist of a huge number of variables and constraints ranging up to the hundred thousands. A linear programming model consists of an objective function, that is a linear function, and the constraints on that function that are also linear. Linear programming involves the planning of activities to obtain a result that reaches a specified goal among all feasible alternatives.

A Linear Program (LP) is a problem that can be expressed in standard form as follows:

$$\begin{aligned} &\text{Minimum} && c^T x \\ &\text{subject to} && Ax = b \\ & && x \geq 0 \end{aligned} \qquad (1\text{-}1)$$

where $x \in R^n$ is the vector of variables to be solved for, and matrix $A \in R^{mxn}$ and vectors $c \in R^n$, $b \in R^m$ are input data. . The linear function $Z = c^T x$ is called the objective function, and the equations $Ax = b$ are called functional constraints, while $x \geq 0$ are called nonegativity constraints. The set $F = \{x \in R^n \mid Ax = b,\ x \geq 0\}$ is called a feasible set. Geometrically, the set represents a polyhedron in $R^n$.

Many practical problems can be modeled as LP models. To illustrate this fact we list the following simple example taken from the [Hillier, Lieberman, 2005].

**Example:** The Wyndor Glass Co. produces high-quality glass products, including windows and glass doors. It has three plants. Aluminum frames and hardware are made in Plant 1, wood frames are made in Plant 2, and Plant 3 produces the glass and assembles the products. Because of declining earnings, top management has decided to revamp the company's product line. Unprofitable products are being discontinued, releasing production capacity to launch two new products having large sales potential:

Product 1: An 8-foot glass door with aluminum framing

Product 2: A 4 x 6 foot double-hung wood-framed window

Product 1 requires some of the production capacity in Plants 1 and 3, but none in Plant 2. Product 2 needs only Plants 2 and 3. The marketing division has concluded that the company could sell as much of either product as could be produced by these plants. However, because both products would be competing for the same production capacity in Plant 3, it is not clear which mix of these two products would be most profitable. Each product will be produced in batches of 20, so the production rate is defined as the number of batches produced per week. Any combination of production rates that satisfies the restrictions is permitted, including producing none of one product and as much as possible of the other. Profit from selling one batch of Product 1 (glass doors) is $3000 and profit from selling one batch of Product 2 (windows) is $5000. We assume that all produced batches will be sold. Goal: To determine what the production rates should be for the two products in order to maximize the total profit, subject to the restrictions imposed by the limited production capacities available in the three plants.

Of course, this is a simplified real world situation but good enough to illustrate the usefulness of the model. Formulation of the Linear Programming (LP) Problem

Let $x_1$ = number of batches of product 1 produced per week

$x_2$ = number of batches of product 2 produced per week

$Z$ = total profit per week in thousands of dollars from producing these two products

The following table summarizes the data gathered:

Wyndor Glass Company Data

| Plant | Production Time Per Batch Hours | | Production Time Available per Week, Hours |
|---|---|---|---|
| | Product | | |
| | 1 | 2 | |
| 1 | 1 | 0 | 4 |
| 2 | 0 | 2 | 12 |
| 3 | 3 | 2 | 18 |
| Profit per Batch | 3000 | 5000 | |

Table 1

The objective function is $Z = 3x_1 + 5x_2$ and it represents a total profit measured in thousands of dollars. The objective function is subject to the restrictions imposed by the limited production capacities available in each of the plants, and they can be mathematically expressed by the following inequalities:

$$\begin{aligned} &\textit{Plant } 1 && x_1 \le 4 \\ &\textit{Plant } 2 && 2x_2 \le 12 \\ &\textit{Plant } 3 && 3x_1 + 2x_2 \le 18 \end{aligned}$$

Thus, the overall linear programming model illustrating the Wyndor Glass Company is

$$\begin{aligned} &\textit{Maximize} && Z = 3x_1 + 5x_2 \\ &\textit{subject to} && x_1 \le 4 \\ & && 2x_2 \le 12 \\ & && 3x_1 + 2x_2 \le 18 \\ & && x_1 \ge 0,\ x_2 \ge 0 \end{aligned}$$

By adding slack variables this problem can be transformed in the standard form (1-1).

$$\begin{aligned} &\textit{Maximize} && Z = 3x_1 + 5x_2 \\ &\textit{subject to} && x_1 + x_3 = 4 \\ & && 2x_2 + x_4 = 12 \\ & && 3x_1 + 2x_2 + x_5 = 18 \\ & && x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0,\ x_4 \ge 0, x_5 \ge 0 \end{aligned}$$

The similar procedure can be done for different inequality formulations of LP.

This example illustrates the applicability of the LP model. The number of problems that can be modeled as LP is huge and widespread to many areas of science, industry, business, finance, government, etc. For more examples see [HL] and other Operations Research textbooks. Therefore, the efficient methods to solve LP models are very important. In the following sections we will outline main methods that are used to solve LP models.

## Methods to Solve LP Models

The first successful general procedure, **Simplex Method**, for solving a LP problem was discovered by George Dantzig in 1947 although there were partial results discovered earlier. Theoretically the main idea of the simplex method (SM) is that it travels from vertex to vertex

on the boundary of the feasible region, repeatedly increasing or decreasing the objective function until either an optimal solution is found, or it is established that no solution exists. The number of iterations required in the worst case is an exponential function of the number of variables, as it was first discovered by Klee and Minti in 1972. However, the worst case behavior has not been observed in practice. On the contrary, the algorithm works very well in practice, typically requiring $O(n)$ iterations. Highly sophisticated implementations are available (CPLEX, MOSEK, LINDO, EXCEL SOLVER) and have excellent codes for simplex algorithms. These codes are capable of solving huge problems with millions of variables and thousands of constraints. This discrepancy between exponential worst case complexity and good practical behavior of simplex method prompted the research in two directions. One direction was a search for the algorithm with the polynomial worst case complexity and the other direction was the analysis of average complexity of simplex method.

In 1979, Leonid Khaciyan showed that the **Ellipsoid Method**, created by A. Nemirovski and D. Yudin for nonlinear programming problems, solves any linear program in a number of steps which is a polynomial function of the amount of data defining the linear program. Unfortunately, in practice, the simplex method turned out to be far superior to the ellipsoid method. However, theoretical importance is significant because it provided a basis to prove that polynomial methods exist for many combinatorial problems.

In 1982 K. Borgward provided the first probabilistic analysis of the simplex method, showing that the expected number of iterations is polynomially bounded. Soon afterwards, other authors provided similar analysis. A relatively simple and complete analysis was provided by Adler and Megiddo in 1985. Using the clever probability model, they showed that

upper and lower bounds on an average number of iterations is a function of $\Omega\left((\min\{m,n\})^2\right)$, where m is the number of constraints and n is the number of variables.

In 1984, Narendra Karmarkar introduced an **Interior-Point Method** (IPM) for linear programming, combining the desirable theoretical properties of the ellipsoid method and practical advantages of the simplex method. Its success initiated an explosion in the development of interior-point methods that continue to this day.

These methods do not pass from vertex to vertex along the edges of the feasible region, which is the main feature of the simplex algorithm; they follow the central path in the interior of the feasible region. Though this property is easy to state, the analysis of interior-point methods is a subtle subject which is much less easily understood than the behavior of the simplex method. Interior-point methods are now generally considered competitive with the simplex method in most, though not all, applications, and sophisticated software packages implementing them are now available (CPLEX, MOSEK, LINDO, EXCEL SOLVER).

# CHAPTER 2

## INTERIOR POINT METHODS BASED ON KERNEL FUNCTIONS

The linear optimization problem in standard form is

$$\text{(P)} \qquad \min\left\{c^T x : Ax = b,\ x \geq 0\right\},$$

where $A \in R^{m x n}$ $(rank(A) = m)$, $b \in R^m$, $c \in R^n$. The dual problem of (P) is

$$\text{(D)} \qquad \max\left\{b^T y : A^T y + s = c,\ s \geq 0\right\},$$

where $s \in R^n$ is a dual slack variable.

We can assume that the Interior Point Condition (IPC) is satisfied without loss of generality; that is, there exists a point $(x^0, s^0, y^0)$ such that $Ax^0 = b,\ x^0 > 0$ and $A^T y^0 + s^0 = c, s > 0$, which means that the interiors of the feasible regions of the primal(P) and dual(D), are not empty. If the problem doesn't satisfy the IPC, it can be modified so that it does and even in such a way that $x^0 = s^0 = e$, where $e$ denotes a vector of all ones. The details can be found in [Roos, C *et. al.*, 1997].

Optimality conditions for (P) and (D) yield the following system of equations:

$$\begin{aligned} Ax &= b, x \geq 0 \\ A^T y + s &= c,\ s \geq 0 \\ xs &= 0, \end{aligned} \qquad \text{(2-1)}$$

where the vector $xs$ denotes the component wise product of vectors x and s which is also called Hadamard product.

The theory of interior point methods (IPMs) that is based on the use of Newton's Method suggests that the third equation in (2-1) has to be perturbed. The third equation is often called the **complementarity condition** for the primal and dual, and is replaced by $xs = \mu e$, where $\mu$ is a positive parameter. The optimality conditions (2-1) are transformed to the following system:

$$\begin{aligned} Ax &= b, x > 0 \\ A^T + y + S &= c,\ s > 0 \\ xs &= \mu e. \end{aligned} \tag{2-2}$$

Since rank (A) = m, this system has unique solution for each $\mu$>0. We can write this solution as $\left(x(\mu), y(\mu), s(\mu)\right)$, calling $x(\mu)$ the $\mu$ – center of (P) and $(y(\mu), s(\mu))$ the $\mu$ – center of (D). The set of all $\mu$ – centers forms a homotopy path in the interior of the feasible region that is called the **central path**.

The main property of the central path can be summarized as follows: if $\mu \to 0,$ then the limit of the central path exists, the limit points satisfy the complementarity condition, and the limit yields optimal solutions for (P) and (D). This limiting property of the central path leads to the main idea of the iterative methods for solving (P) and (D): Trace the central path while reducing $\mu$ at each iteration. However, tracing the central path exactly would be too costly and inefficient. One of the main achievements of interior point methods was to show that tracing the central path approximately while still maintaining good properties of the algorithms is sufficient.

Tracing the central path means solving the system (2-2) using Newton Method on the function

$$F(x,y,s)=\begin{bmatrix} Ax-b \\ A^T y+s-c \\ xs-\mu e \end{bmatrix}=0. \tag{2-3}$$

Tracing the central path approximately means that only one, or at most, a couple of iterations of a modified (damped) Newton's Method will be performed for a particular $\mu$. One iteration of a Newton Method for the function (2-3) and particular $\mu$ is stated below.

$$\nabla F\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}=-F(x,y,s) \tag{2-4}$$

where

$$\nabla F=\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix}$$

denotes the Jacobian of $F$ and $[\Delta x,\Delta y,\Delta z]^T$ is a Newton's search direction that we want to calculate.

Solving (2-4) reduces to solving this system.

$$\begin{aligned} &A\Delta x=0 \\ &A^T\Delta y+\Delta s=0 \\ &s\Delta x+x\Delta s=\mu e-xs \end{aligned} \tag{2-5}$$

Next we update the current iterates $(x,y,s)$ by taking an appropriate step along the calculated direction.

$$x_+ = x + \alpha \Delta x,\ y_+ = y + \alpha \Delta y,\ s_+ = s + \alpha \Delta s$$

The step size α has to be chosen approximately, so that the new iterate is in a certain neighborhood of μ-center. The choice of $\alpha$ will be discussed later in the text.

The idea of the algorithm is illustrated in the Figure 1 blow.

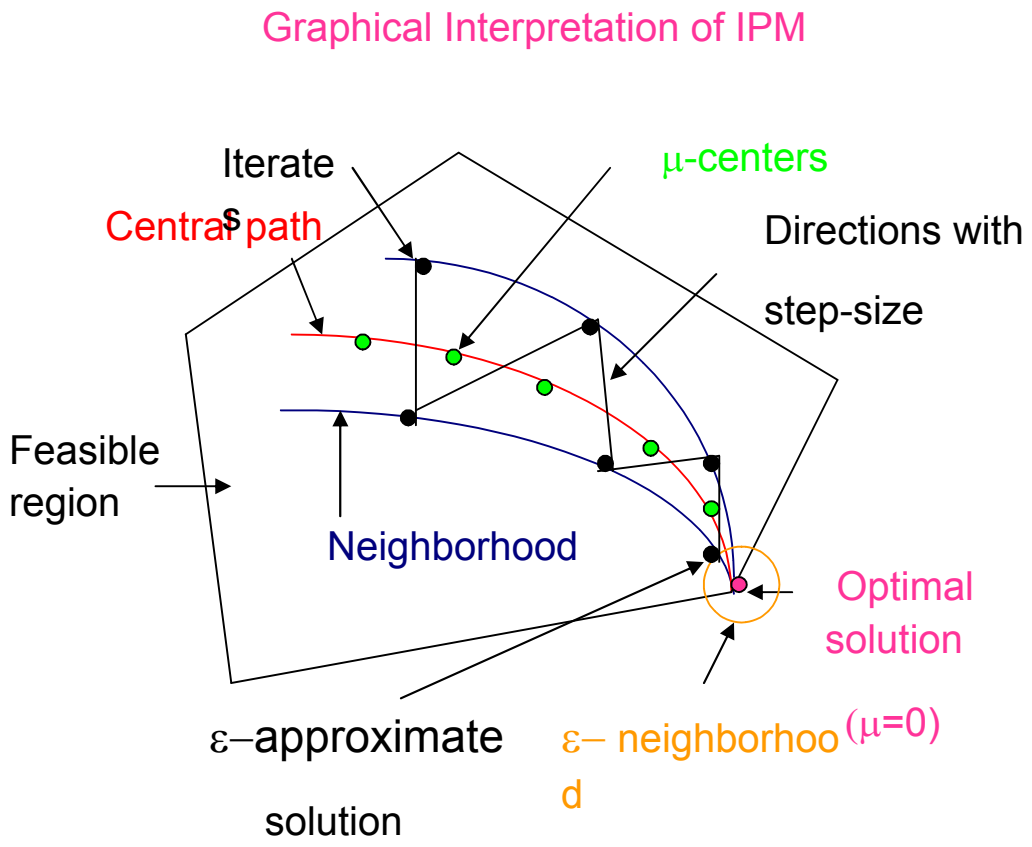



Figure 1

In order to generalize the algorithm outlined above, we introduce a new vector

$$\upsilon = \sqrt{\frac{xs}{\mu}} \tag{2-6}$$

which we use to define new scaled directions

$$d_x := \frac{\upsilon \Delta x}{x}\ ,\ d_s := \frac{\upsilon \Delta s}{s} \tag{2-7}$$

where the operations in (2-7) are component-wise product and division of vectors. Using the above definitions (2-6) and (2-7), the system (2-5) reduces to

$$\begin{aligned} &A\ d_x = 0 \\ &A\ \Delta y + d_s = 0 \\ &d_x + d_s = \upsilon^{-1} - \upsilon \end{aligned} \tag{2-8}$$

where $\overline{A} = \frac{1}{\mu} A V^{-1} X, \quad V := diag(v), \quad X = diag(x)$. Note that $X = diag(x)$ denotes a diagonal matrix that has components of the vector $x$ on the main diagonal and zeros everywhere else.

The new search direction $(\Delta x, \Delta y, \Delta s)$ is obtained by first solving the system (2-8).Once $d_x$ and $d_s$ are found we apply (2-6) to find $\Delta x$ and $\Delta s$. This direction can also be obtained directly by solving the following system:

$$\begin{aligned} &A\Delta x = 0 \\ &A^T \Delta y + \Delta s = 0 \\ &s\Delta x + x\Delta s = -\mu\upsilon\nabla\Psi(\upsilon). \end{aligned} \tag{2-9}$$

This system can be reduced to

$$M\Delta y = r \tag{2-10}$$

where

$$M = AS^{-1}XA^T$$
$$r = \mu AS^{-1}\upsilon\nabla\psi(\upsilon)$$

(2-11)

and $X = diag(x)$, $S = diag(s)$, $V = diag(\upsilon)$.

Once $\Delta y$ is found, $\Delta s$ and $\Delta x$ are found by back substitutions

$$\Delta s = -A^T\Delta y \tag{2-12}$$

$$\Delta x = -s^{-1}\left(x\Delta s - \mu\upsilon\nabla\Psi(\upsilon)\right) \tag{2-13}$$

where products denote component-wise products of vectors.

The following observation is crucial for the generalization of the method. Observe that $-\left(\upsilon^{-1} - \upsilon\right)$ is a gradient of the following function

$$\Psi_c(\upsilon) = \sum_{i=1}^{n}\left(\frac{\upsilon_i^2 - 1}{2} - \log\upsilon_i\right) = \left(\frac{\upsilon_1^2 - 1}{2} - \log\upsilon_2\right) + \cdots + \left(\frac{\upsilon_n^2 - 1}{2} - \log\upsilon_n\right). \tag{2-14}.$$

This function is called **log-barrier** function.

One may easily verify that the Hessian $\nabla^2\Psi_c(\upsilon) = diag(e + \upsilon^{-2})$. Since this matrix is positive definite, $\Psi_c(\upsilon)$ is strictly convex. Moreover, since $\nabla\Psi_c(e) = 0$, it follows that

$\Psi_c(\upsilon)$ attains its minimal value at $\upsilon = e$, with $\Psi_c(e) = 0$. Thus, it follows that $\Psi_c(\upsilon)$ is nonnegative everywhere and vanishes if and only if $\upsilon = e$, that is, if and only if $x = x(\mu)\, and\, s = s(\mu)$. Hence, we see that the $\mu - centers\, x(\mu)\, and\, s(\mu)$ can be characterized as the minimizers of the function $\Psi_c(\upsilon)$. Therefore, the function $\Psi_c(\upsilon)$ serves as a **proximity measure** to the $\mu$-centers (central path.. The norm based proximity measure that is derived from $\Psi_c(\upsilon)$ is defined as

$$\delta = \frac{1}{2}\left\|\nabla\Psi_c(\upsilon)\right\|. \tag{2-15}$$

Furthermore, the complimentary equation in (2-8) can be written as $d_x + d_s = -\nabla\Psi_c(\upsilon)$, which is also called the **scaled centering equation**. The importance of the equation arises from the fact that it essentially defines the search directions. Since $d_x$ and $d_s$ are orthogonal, we will still have $d_x d_s = 0$ if and only if $\upsilon = e$. The same is true for $\Delta x$ and $\Delta s$.

The main idea of the generalization of the method is to replace the log-barrier function (2-14) with some other barrier function that has the same properties as log barrier. The choice of this function will certainly affect the calculation of the search direction, the step size, and with that the rate of the convergence of the method. It is worth examining the classes of barrier functions that may lead to the improved behavior of the algorithm. In what follows, we will consider several such classes.

We will restrict ourselves to the case where the barrier function $\Psi(\upsilon)$ is separable with identical coordinate functions $\varphi(\upsilon_i)$. Thus,

$$\Psi(\upsilon) = \sum_{i=1}^{n} \varphi(\upsilon_i), \tag{2-16}$$

where $\varphi(t) : [0, \infty)$ is twice differentiable and attains its minimum at $t = 1$, with $\varphi(1) = 0$. The function $\varphi(t)$ is called a **kernel function**. The log-barrier function belongs to this class.

The algorithm based on generic kernel function that was outlined above is summarized in the Figure 2 below. In principle, each barrier function gives rise to a different primal-dual algorithm. The parameters $\tau, \theta$ and the step size $\alpha$ in the algorithm should be tuned in such a way that the number of iterations required by the algorithm is as small as possible. The resulting iteration bound will depend on the kernel function, and our main task becomes to find a kernel functions that give a good and possibly best known iteration bound. The question of finding the kernel function that minimizes the iteration bound is still an open question.

***Generic Primal-Dual Interior-Point Algorithm for Linear Optimization***

Input:

An input data A, b, c

A threshold parameter $\tau \geq 1$;

An accuracy parameter $\varepsilon > 0$;

A fixed barrier update parameter $\theta < 1$;

Iteration:

begin

$x := e;\ s := e;\ \mu := 1;$

while $n\mu > \varepsilon$ do

begin

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

calculate $\upsilon := \sqrt{\frac{xs}{\mu}};$

while $\Psi(\upsilon) > \tau$ do

begin

calculate the direction $(\Delta x, \Delta y, \Delta s)$

using (2-10) – (2-13) :

calculate step size $\alpha$ ;

update

$x := x + \alpha \Delta x,\ \ s := s + \alpha \Delta s,\ \ y := y + \alpha \Delta y;$

end

end

end

Figure 2 Generic IPM

# CHAPTER 3

## ANALYSIS OF THE ALGORITHM FOR A CLASS OF KERNEL FUNCTIONS

The following class of kernel functions will be used to analyze the algorithm, Generic IPM, discussed in the Chapter 2, Figure 2 .

$$\psi_{p,q}(t) = \begin{cases} \dfrac{t^{p+1}-1}{p+1} + \dfrac{t^{1-q}-1}{q-1}, & t > 0,\ p \in [0.1],\ q > 1 \\ \dfrac{t^{p+1}-1}{p+1} - \log t, & t > 0,\ p \in [0.1] \end{cases} \tag{3-1}$$

where $p$ is a growth parameter and $q$ is a barrier parameter.

Notice that $\dfrac{t^{1-q}-1}{q-1} \to -\log t$ when $q \to 1$. This class of kernel functions is fairly general. As we just explained, it includes log-kernel function as a special case. It also includes so-called self-regular kernel functions when $p = 1$. These functions have been extensively discussed in recent literature (Peng, J, *et. al*, 2002). Moreover, it also includes non self-regular functions when $0 \le p < 1$. This class of functions was first discussed in (Lesaja G., et.al., 2008). The results in this chapter follow the results presented in that paper. However, the proofs of several results that were omitted in the paper are outlined here.

### Properties of Kernel Functions

The derivatives of $\psi(t)$ in (3-1) play a crucial role in our analysis. Thus, we write down the first three derivatives:

$$\begin{aligned}\psi'(t) &= t^{p} - t^{-q} \\ \psi''(t) &= pt^{p-1} + qt^{-q-1} \\ \psi'''(t) &= p(p-1)t^{p-2} - q(q+1)t^{-q-2}.\end{aligned} \tag{3-2}$$

In the next several lemmas we will describe certain properties of kernel function and its derivatives and their relationships in terms of inequalities. These results will be used in the analysis of the Generic IPM.

The following lemma states the so-called exponential convexity of kernel function which is crucial in proving the polynomial complexity of the Generic IPM.

**Lemma 3.1** *If* $t_1 > 0$ *and* $t_2 > 0$, *then* $\psi\left(\sqrt{t_1 t_2}\right) \le \frac{1}{2}\left(\psi(t_1) + (\psi(t_2))\right)$.

Proof: It can be shown that the inequality in the lemma holds if and only if $t\psi''(t) + \psi'(t) \ge 0$ *for all* $t > 0$. This result is beyond the scope of the thesis and can be found (Peng, J., et. al., 2002). Using (3-1) one can easily verify that

$$t\psi''(t) + \psi'(t) = t\left(pt^{p-1} + \frac{q}{t^{q+1}}\right) + t^{p} - \frac{1}{t^{q}} = (p+1)t^{p} + \frac{(q-1)}{t^{q}} > 0,$$

which completes the proof.

**Lemma 3.2** If $t \ge 1$, then $\frac{\psi'(t)}{2}(t-1) \le \psi(t) \le \frac{p+q}{2}(t-1)^2$.

Proof: If $f(t) = 2\psi(t) - (t-1)\psi'(t)$, then $f'(t) = \psi'(t) - (t-1)\psi''(t)$ and $f''(t) = -(t-1)\psi'''(t)$. Also $f(1) = 0$ and $f'(1) = 0$. Since $\psi'''(t) < 0$ it follows that if

$t \geq 1$ then $f''(t) \geq 0$ whence $f'(t) \geq 0$ and $f(t) \geq 0$. This implies the first inequality. The second inequality follows from Taylor's theorem and the fact that $\psi''(1) = p + q$.

**Lemma 3.3** Suppose that $\psi(t_1) = \psi(t_2)$ with $t_1 \leq 1 \leq t_2$. The following statements hold:

i. One has $\psi'(t_1) \leq 0$, $\psi'(t_2) \geq 0$, and $-\psi'(t_1) \geq \psi'(t_2)$.

ii. If $\beta \geq 1$, then $\psi(\beta t_1) \leq \psi(\beta t_2)$; equality holds if and only if $\beta = 1$ or $t_1 = t_2 = 1$.

Proof: Proof of (i):

The statement is obvious if $t_1 = 1$ or $t_2$=1 because then $\psi'(t_1) = \psi'(t_2) = 0$ implies $t_1 = t_2 = 1$ Thus we may assume that $t_1 < 1 < t_2$. Suppose the opposite $-\psi'(t_1) < \psi'(t_2)$. By the mean value theorem we have

$$\psi'(t_1) = (1 - t_1)\psi''(\xi_2), \quad \text{for some } \xi_2 \in (1, t_2)$$

and

$$-\psi'(t_2) = (t_2 - 1)\psi''(\xi_1), \text{ for some } \xi_1 \in (t_1, 1).$$

Since $\psi''$(t) is monotonically decreasing one has $\psi''(\xi_1) \geq \psi''(\xi_2)$. Then we obtain

$$(t_2 - 1)\psi''(\xi_2) > (1 - t_1)\psi''(\xi_1) \geq (1 - t_1)\psi''(\xi_2),$$

Hence since $\psi''(\xi_2) > 0$ it follows that $t_2 - 1 > 1 - t_1$. Using this and the fact that $-\psi'(t)$ is convex, we may also write

$$
\begin{aligned}
&\psi(t_2) = \\
&= \int_1^{t_2} \psi'(\xi)d\xi && \text{Since } \psi'(t) \text{ is concave,} \\
&\geq \frac{1}{2}(t_2 - 1)\psi'(t_2) && \text{Since } t_2 - 1 > 1 - t_1 \text{ and } \psi'(t_2) > 0, \\
&> \frac{1}{2}(1 - t_1)\psi'(t_2) && \text{Since } \psi'(t_1) < \psi'(t_2), \\
&> -\frac{1}{2}(1 - t_1)\psi'( && \text{Since } \psi'(t) \text{ is concave.} \\
&\geq -\int_{t_1}^{1} \psi'(\xi)d\xi \\
&= \psi(t_1).
\end{aligned}
$$

This contradiction proves the first part of the lemma.

Proof of (ii):

Consider, $f(\beta) = \psi(\beta t_2) - \psi(\beta t_1)$.

One has $f(1) = 0$ and $f'(\beta) = t_2\psi'(\beta t_2) - t_1\psi'(\beta t_1)$.

Since $\psi''(t) \geq 0$ for all $t > 0$, $\psi'(t)$ is monotonically increasing. Hence, $\psi'(\beta t_1) \leq \psi'(\beta t_2)$.

Substitution gives

$$f'(\beta) = t_2\psi'(\beta t_2) - t_1\psi'(\beta t_1) \geq t_2\psi'(\beta t_2) - t_1\psi'(\beta t_2) = \psi'(\beta t_2)(t_2 - t_1) \geq 0$$

The last inequality holds since $t_2 \geq t_1$ *and* $\psi'(t) \geq 0$ *for* $t \geq 1$. This proves that $f(\beta) \geq 0$ for $\beta \geq 1$, and hence the inequality (ii) in the lemma follows.

If $\beta = 1$ obviously we have equality. Otherwise, if $\beta > 1$ and $f(\beta) = 0$, then the mean value theorem implies $f'(\xi) = 0$ for some $\xi \in (1, \beta)$. But this implies $\psi'(\xi t_2) = \psi'(\xi t_1)$. Since $\psi'(t)$

is strictly monotonic, this implies that $\xi t_2 = \xi t_1$, whence $t_1 = t_2$. Since also $t_1 \le 1 \le t_2$ we obtain $t_1 = t_2 = 1$. This completes the proof of the second part of the lemma.

**Lemma 3.4** If $t \ge 1$, then $\psi'(t)^2 \ge 2\psi(t)\psi''(t)$.

Proof : Defining $f(t) = \psi'(t)^2 - 2\psi(t)\psi''(t)$ one has $f(1) = 0$ and

$$f'(t) = 2\psi'(t)\psi''(t) - 2\psi'(t)\psi''(t) - 2\psi(t)\psi'''(t) = -2\psi(t)\psi'''(t) > 0 .$$

This proves the lemma.

**Lemma 3.5** Let $\rho(s) : [0,\infty) \to (0,1]$ be the inverse function of $-\frac{1}{2}\psi'(t)$ *for* $t \le 1$. The following inequality holds:

$$\rho(s) \ge \frac{1}{(1+2s)^{\frac{1}{q}}} . \tag{3-3}$$

Proof: Since $s = -\frac{1}{2}\psi'(t)$ , we have

$$-2s = t^p - t^{-q} \Rightarrow t^{-q} = t^p + 2s \le 1 + 2s .$$

Since $t = \rho(1)$, this implies the lemma.

**Lemma 3.6** If $t \ge 1$ and $q \ge 2 - p$ , then $t \le 1 + \sqrt{t\psi(t)}$ .

Proof: Defining $f(t) = t\psi(t) - (t-1)^2$ we have $f(1) = 0$ and

$$f'(t) = \psi(t) + t\psi'(t) - 2(t-1).$$

Moreover, it is clear that $f'(1)=0$ and

$$f''(t)=2\psi'(t)+t\psi''(t)-2=(2+p)t^{p}+(q-2)t^{-9}-2\geq pt^{p}+(q-2)t^{q}\geq p\left(t^{p}-t^{-q}\right)\geq 0.$$

The second inequality above is due to the fact that $q\geq 2-p$. Thus we obtain

$$t\psi(t)\geq (t-1)^{2},$$

which implies the lemma.

**Lemma 3.7** Let $\varphi:[0,\infty)\to[1,\infty)$ be the inverse function of $\psi(t)$ for $t\geq 1$. The following inequalities hold:

$$(1+(p+1)s)^{\frac{1}{p+1}}\leq \varphi(s)\leq 1+s+\sqrt{s^{2}+2s}\,. \tag{3-4}$$

If $q\geq 2-p$, then

$$\varphi(s)\leq 1+\sqrt{s+s^{2}+s\sqrt{s^{2}+2s}} \tag{3-5}$$

Proof: Since q>1 and $t\geq 1$, we have

$$s=\psi(t)=\frac{t^{p+1}-1}{p+1}+\frac{t^{1-q}-1}{q-1}\leq\frac{t^{p+1}-1}{p+1}$$

Hence, the first inequality in (3-4) follows.

The second inequality in (3-4) follows by using the first inequality of Lemma 3.2:

$$s=\psi(t)\geq\frac{1}{2}(t-1)\psi'(t)=\frac{1}{2}(t-1)\left(t^{p}-t^{-q}\right)\geq\frac{1}{2}(t-1)\left(1-\frac{1}{t}\right)=\frac{1}{2}\left(t+\frac{1}{t}-2\right).$$

Hence, solving the following inequality

$$t^2 - 2(1+s)^t + 1 \le 0$$

leads to

$$t = \varphi(s) \le 1 + s + \sqrt{s^2 + 2s}. \tag{3-6}$$

Finally, let $q \ge 2 - p$. By Lemma 3.6 one has $t \le 1 + \sqrt{t\psi(t)} \le 1 + \sqrt{ts}$.

Substitution of the upper bound for *t* given by (3-6) leads to

$$\varphi(s) \le 1 + \sqrt{s + s^2 + s\sqrt{s^2 + 2s}}.$$

This completes the proof of the lemma.

Now we will derive a very important bound for normed proximity measure $\delta(\upsilon) = \frac{1}{2}\|\nabla\Psi(\upsilon)\|$ in terms of the original proximity measure given by the barrier function $\Psi(\upsilon)$.

**Theorem 3.1** The following inequality holds:

$$\delta(\upsilon) \ge \frac{1}{2}\psi'(\varphi(\Psi(\upsilon))). \tag{3-7}$$

The proof is beyond the scope of thesis and can be found in (Peng, J., et. al., 2002)

**Corollary 3.1** If $\Psi(\upsilon) \ge \tau \ge 1$, then $\delta(\upsilon) \ge \frac{1}{6}(\Psi(\upsilon))^{\frac{p}{1+p}}$

Proof: Using Theorem 2.1, and the fact that $\Psi(\upsilon) \geq \tau \geq 1$, we have

$\delta(\upsilon) \geq \frac{1}{2}\psi'(\varphi(\Psi(\upsilon))) = \frac{1}{2}(\varphi(\Psi(\upsilon)))^p - \left((\varphi(\Psi(\upsilon)))^{-q}\right) \geq \frac{1}{2}(\varphi(\Psi(\upsilon)))^p - \left((\varphi(\Psi(\upsilon)))^{-1}\right)$. Note that

$t^p - \frac{1}{t}$ is monotonically increasing in $t$ . Thus, by using the first inequality in (3-4), we obtain

$$\begin{aligned}\delta(\upsilon) &\geq \frac{1}{2}\frac{(\varphi(\Psi(v)))^{p+1} - 1}{\varphi((\Psi(\upsilon))} \geq \frac{1}{2}\frac{(1+(p+1)\Psi(\upsilon))^{\frac{p+1}{p+1}} - 1}{(1+(p+1)\Psi(\upsilon))^{\frac{1}{p+1}}} \\ &= \frac{1}{2}\frac{(p+1)\Psi(\upsilon)}{(1+(p+1)\Psi(\upsilon))^{\frac{1}{p+1}}} \geq \frac{1}{2}\frac{\Psi(\upsilon)}{(3\Psi(\upsilon))^{\frac{1}{p+1}}} \\ &\geq \frac{1}{6}(\Psi(\upsilon))^{\frac{p}{p+1}}\end{aligned}$$

Which proves the corollary.

## Analysis of the algorithm

The outline of the analysis of the algorithm is as follows.

1. Outer iteration estimates:
    - Estimate of the increase of the barrier function after the μ update
2. Inner iteration estimates:
    - Estimator for default step size
    - Estimate of the decrease of the barrier function during the inner iteration with the default step size

## Outer Iteration Estimates

At the start of each outer iteration of the algorithm, just before the update of the parameter μ with the factor $1-\theta$, we have $\psi(\upsilon) \le \tau$. Since the μ vector is updated to $\mu^+ = (1-\theta)\mu$, with $0 < \theta < 1$, the vector $\upsilon$ is updated to $\upsilon^+ = \frac{\upsilon}{\sqrt{1-\theta}}$, which in general leads to an increase in the value of $\Psi(\upsilon)$. Then, during the subsequent inner iterations, $\Psi(\upsilon)$ decreases until it passes the threshold τ again. During the course of the algorithm the largest values of $\Psi(\upsilon)$ occur just after the updates of μ. That is why we need to derive an estimate for the effect of a μ-update on the value of $\Psi(\upsilon)$.

**Theorem 3.2** Let $\varphi : [0,\infty) \to [1,\infty)$ as defined in Lemma 3.7. Then for any positive vector $\upsilon$ and any $\beta \ge 1$ the following inequality holds:

$$\Psi(\beta\upsilon) \le n\psi\left(\beta\varphi\left(\frac{\Psi(\upsilon)}{n}\right)\right).$$

The proof is beyond the scope of this thesis and can be found in [Peng, J., et. al., 2002].

**Corollary 3.2** Let $0 \le \theta \le 1$ and $\upsilon_+ = \frac{\upsilon}{\sqrt{1-\theta}}$. If $\Psi(\upsilon) \le \tau$, then

$$\Psi(\upsilon_+) \le n\psi\left(\frac{\varphi\left(\frac{\tau}{n}\right)}{\sqrt{1-\theta}}\right) \le \frac{(p+q)n}{2}\left(\frac{\varphi\left(\frac{\tau}{n}\right)}{\sqrt{1-\theta}} - 1\right)^2. \qquad (3\text{-}8)$$

Proof: With $\beta \geq 1$ and $\Psi(\upsilon) \leq 1$ the first inequality follows from Theorem 3.2. The second inequality follows by using Lemma 3.2 and $\psi''(1) = \frac{p+q}{q}$.

The following upper bounds on the value of $\psi(\upsilon_+)$ after the μ-update follow immediately

$$\psi(\upsilon_+) \leq L_1 := n\varphi\left(\frac{1+\frac{\tau}{n}+\sqrt{\left(\frac{\tau}{n}\right)^2+\frac{2\tau}{n}}}{\sqrt{(1-\theta)}}\right), \qquad q > 1; \tag{3-9}$$

and

$$\psi(\upsilon_+) \leq L_2 := n\varphi\left(\frac{1+\sqrt{\frac{\tau}{n}+\frac{\tau^2}{n^2}+\frac{\tau}{n}\sqrt{\frac{\tau^2}{n^2}+\frac{2\tau}{n}}}}{\sqrt{1-\theta}}\right), \qquad q \geq 2-p. \tag{3-10}$$

**Default Step-size**

In this subsection, we will determine a default step size which not only keeps the iterations feasible but also gives rise to sufficiently large decrease of $\Psi(\upsilon)$ in each inner iteration. During an inner iteration, the parameter μ is fixed. After the step in the direction $(\Delta x, \Delta y, \Delta s)$ with step size α, the new iterate is

$$x_+ = x + \alpha\,\Delta x,\ s_+ = s + \alpha\,\Delta x,\ y_+ = y + \alpha\,\Delta y \tag{3-11}$$

And a new $\upsilon$-vector is given by

$$\upsilon_+ = \sqrt{\frac{x_+ s_+}{\mu}}. \tag{3-12}$$

Since

$$x_+ = x\left(e + \alpha\frac{\Delta x}{x}\right) = x\left(e + \alpha\frac{d_x}{\upsilon}\right) = \frac{x}{\upsilon}(u + \alpha d_x),$$
$$s_+ = s\left(e + \alpha\frac{\Delta s}{s}\right) = s\left(e + \alpha\frac{d_s}{\upsilon}\right) = \frac{s}{\upsilon}(u + \alpha d_s),$$
$$xs = \mu\upsilon^2$$

we obtain

$$\upsilon_+ = \sqrt{(\upsilon + \alpha d_x)(\upsilon + \alpha d_s)}.$$

Next, we consider the decrease in $\Psi$ as a function of $\alpha$. We define two functions

$$f(\alpha) = \Psi(\upsilon_+) - \Psi(\upsilon), \tag{3-13}$$

and

$$f_1(\alpha) := \frac{1}{2}\left(\Psi(\upsilon + \alpha d_\alpha) + \Psi(\upsilon + \alpha ds)\right) - \Psi(\upsilon). \tag{3-14}$$

Lemma 3.1 implies that

$$\Psi(\upsilon_+) = \Psi\left(\sqrt{(\upsilon + \alpha d_x)(\upsilon + \alpha d_s)}\right) \le \frac{1}{2}\left(\Psi(\upsilon + \alpha d_x) + \Psi(\upsilon + \alpha d_s)\right).$$

The above inequality shows that $f_1(\alpha)$ is an upper bound of $f(\alpha)$. Obviously,

$f(0) = f_1(0) = 0$. Taking the derivative with respect to $\alpha$, we get

$$f'_1(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\left(\psi'(\upsilon_i + \alpha\, d_{xi})d_{xi} + \psi'(\upsilon_i + \alpha\, d_{si})d_{si}\right).$$

From the above equation and using that $d_x + d_s = -\nabla\Psi(\upsilon)$ we obtain

$$f'_1(0) = \frac{1}{2}\nabla\Psi(v)^T(d_x + d_s) = -\frac{1}{2}\nabla\Psi(v)^T\nabla\Psi(v) = -2\delta(\upsilon)^2. \qquad (3\text{-}15)$$

Differentiating once again, we get

$$f''_1(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\left(\psi''(\upsilon_i + \alpha\, d_{xi})\right)d_{xi}^2 + \psi''\left((\upsilon_i + \alpha\, d_{si})\right)d_{si}^2 > 0, \text{ unless } d_x = d_s = 0.$$

(3-16)

It is worthwhile to point out that during an inner iteration $x$ *and* $s$ are not both at the $\mu$ center since $\Psi(v) \geq \tau \geq 0$, so we may conclude that $f_1(\alpha)$ is strictly convex in $\alpha$.

**Lemma 3.8** The following inequality holds:

$$f_1''(\alpha) \leq 2\delta^2\psi''(\upsilon_{\min} - 2\alpha\delta). \qquad (3\text{-}17)$$

Proof: Since $d_x \perp d_s$, and $d_x + d_s = -\nabla\Psi(\upsilon)$ it is easy to see that $\|(d_x, d_s)\| = 2\delta.$ where $\delta = \frac{1}{2}\|\nabla\Psi(\upsilon)\|$ Therefore, we have $\|d_x\| \leq 2\delta$ and $\|d_s\| \leq 2\delta.$ Hence,

$$\upsilon_i + ad_x \geq \upsilon_{\min} - 2a\delta, \quad \upsilon_i + ad_s \geq \upsilon_{\min} - 2a\delta, \quad 1 \leq i \leq n.$$

Using (3-16) and definition of $\delta$, we get

$$f_1''(a) \leq \frac{1}{2}\psi''(\upsilon_{\min} - 2a\delta)\sum_{i=1}^{n}\left(d_{xi}^2 + d_{si}^2\right) = 2\delta^2\psi''(\upsilon_{\min} - 2a\delta).$$

This proves the lemma.

**Lemma 3.9** If the step-size $\alpha$ satisfies

$$-\psi'(\upsilon_{\min} - 2\alpha\delta) + \psi'(\upsilon_{\min}) \le 2\delta, \tag{3-18}$$

then $f_1'(\alpha) \le 0$.

Proof: Using the Lemma 3.8,(3-15) and (3-17), we write:

$$\begin{aligned} f_1'(a) &= f_1'(0) + \int_o^a f_1''(\varsigma)d\varsigma \\ &\le -2\delta^2 + 2\delta^2 \int_o^a \psi''(\upsilon_{\min} - 2\varsigma\delta)d\varsigma \\ &= -2\delta^2 - \delta \int_o^a \psi''(\upsilon_{\min} - 2\varsigma\delta)\, d(\upsilon_{\min} - 2\varsigma\delta) \\ &= -2\delta^2 - \delta\left(\psi'(\upsilon_{\min} - 2\alpha\delta) - \psi'(\upsilon_{\min})\right) \\ &\le -2\delta^2 + 2\delta^2 = 0. \end{aligned}$$

This proves the lemma.

**Lemma 3.10** The largest possible value of the step-size satisfying the condition of Lemma 3.9 is given by

$$\bar{\alpha} := \frac{1}{2\delta}\left(\rho(\delta) - \rho(2\delta)\right) \tag{3-19}$$

Proof: We want $\alpha$ such that (3-18) holds with $\alpha$ as large as possible. Let us denote $\upsilon_{\min}$ as $\upsilon_1$. Since $\psi''(t)$ is decreasing the derivative with respect to $\upsilon_1$. of the expression which is to the left side of the inequality (3-18) (i.e. $\psi''(\upsilon_1 - 2\alpha\delta) + \psi''(\upsilon_1)$) is negative. Hence, fixing $\delta$, the smaller $\upsilon_1$ is, the smaller $\alpha$ will be. We have

$$\delta = \frac{1}{2} \| \nabla \Psi(\upsilon) \| \geq \frac{1}{2} | \psi'(\upsilon_1) | \geq -\frac{1}{2} \psi'(\upsilon_1).$$

Equality holds if and only if $\upsilon_1$. is the only coordinate in $\upsilon$ that differs from 1 and $\upsilon_1 \leq 1$ (in case $\psi'(\upsilon_1) \leq 0$). Hence the worst situation for the step size occurs when $\upsilon_1$ satisfies

$$-\frac{1}{2} \psi'(\upsilon_1) = \delta. \tag{3-20}$$

The derivative with respect to $\alpha$ of the expression that is the left side of the inequality (3-18) equals $2\delta\psi''(\upsilon_1 - 2\alpha\delta) \geq 0$ and hence this expression is increasing in $\alpha$ Thus, the largest possible value of $\alpha$ satisfying (3-18), satisfies

$$-\frac{1}{2} \psi'(\upsilon_1 - 2\alpha\delta) = 2\delta. \tag{3-21}$$

Due to the definition of $\rho$, (3-20) and (3-21) can be written as

$$\upsilon_1 = \rho(\delta), \qquad \upsilon_1 - 2\alpha\delta = \rho(2\delta).$$

This implies

$$\alpha = \frac{1}{2\delta}(\upsilon_1 - \rho(2\delta)) = \frac{1}{2\delta}(\rho(\delta) - \rho(2\delta))$$

and the lemma is proved.

**Lemma 3.11** Let $\bar{\alpha}$ be defined by (3-19) . The following inequality holds:

$$\bar{\alpha} \geq \frac{1}{\psi''(\rho(2\delta))}. \tag{3-22}$$

Proof: By definition of $\rho$,

$$-\psi'(\rho(\delta)) = 2\delta$$

Taking the derivative with respect to $\delta$, we find

$$-\psi''(\rho(\delta))\rho'(\delta) = 2$$

which leads to

$$\rho'(\delta) = -\frac{2}{\psi''(\rho(\delta))} < 0. \tag{3-23}$$

Hence, $\rho$ is monotonically decreasing. An immediate consequence of (3-19)) is

$$\overline{\alpha} = \frac{1}{2\delta}\int_{2\delta}^{\delta} \rho'(\sigma)d\sigma = \frac{1}{\delta}\int_{\delta}^{2\delta} \frac{d\sigma}{\psi''(\rho(\sigma))} \tag{3-24}$$

where we also used (3-23). To obtain a lower bound for $\overline{\alpha}$, we want to replace the argument of the last integral by its minimal value. We would like to know when $\psi''(\rho(\sigma))$ is a maximal for $\sigma \in [\delta, 2\delta]$. We know that $\psi''$ is monotonically decreasing. Thus $\psi''(\rho(\sigma))$ is maximal for $\sigma \in [\delta, 2\delta]$ when $\rho(\sigma)$ is minimal. Since $\rho$ is monotonically decreasing this occurs when $\sigma = 2\delta$. Therefore

$$\overline{\alpha} = \frac{1}{\delta}\int_{\delta}^{2\delta} \frac{d\sigma}{\psi''(\rho(\sigma))} \geq \frac{1}{\delta}\frac{\delta}{\psi''(\rho(2\delta))} = \frac{1}{\psi''(\rho(2\delta))},$$

and the lemma is proved.

**Theorem 3.3** We have $\bar{\alpha} \geq \tilde{\alpha} := \dfrac{1}{(p+q)(1+4\delta)^{\frac{1+q}{q}}}$ (3-25)

Proof: Using Lemma 3.11, and the fact that $\psi''(t)$ is monotonically decreasing for $t \in (0,+\infty)$ we have

$$\bar{\alpha} \geq \frac{1}{\psi''(p(2\delta))} \geq \frac{1}{p(1+4\delta)^{\frac{1+q}{q}} + q(1+4\delta)^{\frac{1+q}{q}}} \geq \frac{1}{(p+q)(1+4\delta)^{\frac{q+1}{q}}} = \tilde{\alpha}$$

and the theorem is proved.

Thus, we can define the following default step-size

$$\tilde{\alpha} := \frac{1}{(p+q)(1+4\delta)^{\frac{q+1}{q}}}, \tag{3-26}$$

## Inner Iteration Estimates

Using the lower bound on the step size obtained in (3-25), we can obtain results on the decrease of the barrier function during inner iteration.

**Lemma 3.12** If the step size is such that $\alpha \leq \bar{\alpha}$, then $f(\alpha) \leq -\alpha\delta^2$. (3-27)

Proof**:** Let the univariate function *h* be such that

$$h(0) = f_1(0) = 0,\ h'(0) = f_1^{'} = 2\delta^2,\ h^{"}(a) = 2\delta^2\psi''(\upsilon_{\min} - 2a\delta)$$

According to (3-17) we have $f''(\alpha) \le h''(\alpha)$ and that implies $f'(\alpha) \le h'(\alpha)$ and $f(\alpha) \le h(\alpha)$. Taking $\alpha \le \widetilde{\alpha}$, with $\widetilde{\alpha}$ defined in (3-26)), we have

$$h'(\alpha) = -2\delta^2 + 2\delta^2 \int_0^{\alpha} \psi''(\upsilon_{\min} - 2\xi\delta)d\xi = -2\delta^2 - \delta(\psi'(\upsilon_{\min} - 2\alpha\delta) - \psi'(\upsilon_{\min})) \le 0.$$

Since $\psi'''(t)$ decreasing in $t$, $h''(\alpha)$ is increasing $\alpha$. Using Lemma 3.13 below, we get:

$$f_1(a) \le h(a) \le \frac{1}{2} a h'(0) = -\alpha\delta^2 .$$

As we mentioned before, $f_1(\alpha)$ is an upper bound of $f(a)$, hence, the lemma is proved.

**Theorem 3.4** The following inequality holds:

$$f(\widetilde{\alpha}) \le -\frac{1}{60(p+q)} \Psi(\upsilon)^{\frac{p(q-1)}{q(p+1)}} . \tag{3-28}$$

Poof: According to Lemma 3.12, if the step-size is such that $\alpha \le \overline{\alpha}$, then $f(\alpha) \le -\alpha\delta^2$. By (3-25) the default step-size $\widetilde{\alpha}$ satisfies $\widetilde{\alpha} \le \alpha$, hence, the following upper bound for $f(\alpha)$ is obtained $f(\widetilde{\alpha}) \le -\widetilde{\alpha}\delta^2$. Using Corollary 3.2 and the fact that $f(\widetilde{\alpha})$ is monotonically decreasing in $\delta$, we obtain

$$f(\widetilde{\alpha}) \le \frac{\delta^2}{(p+q)(1+4\delta)^{\frac{q+1}{q}}}$$

$$\leq \frac{\Psi(\upsilon)^{\frac{2p}{p+1}}}{36(p+q)\left(1+\frac{2}{3}\Psi(\upsilon)^{\frac{p}{p+1}}\right)^{\frac{q+1}{q}}}$$

$$\leq -\frac{1}{60(p+q)}\Psi(\upsilon)^{\frac{p(q-1)}{q(p+1)}}$$

Thus, the proof is complete.

**Estimate of the total number of iterations**

As we've already mentioned, there are two types of algorithms.

- The **Short-step Algorithms**, where the barrier update parameter θ depends on the size of the problem; that is, $\theta = O\left(\frac{1}{\sqrt{n}}\right)$.

- The **Long-step Algorithms**, where the barrier update parameter θ is fixed; that is, $\theta \in (0,1)$.

We will give the estimate of the total number of iterations needed for both types of algorithms.

We will need following technical results. The proofs can be found in (Peng, J., et. al., 2002)]

**Lemma 3.13** If $\alpha \in [0,1]$ and $t \geq -1$ then $(1+t)^{\alpha} \leq 1+\alpha t$.

**Lemma 3.14** Let $h(t)$ be twice differentiable convex function with $h(0) = 0,\ h'(0) < 0,$ and let $h(t)$ attain its (global) minimum at $t^* > 0.$ . If $h''(t)$ is monotonically increasing for $t \in \{0, t^*\}$ then $h(t) \le \frac{th'(0)}{2},\ 0 \le t \le t^*$.

**Lemma 3.15** Let $t_0, t_1, ...t_k$ be a sequence of positive numbers such that

$t_{k+1} \le t_k - \beta_k^{1-\gamma}, \quad k = 0,1...K-1$ where $\beta > 0$ *and* $0 < \gamma \le 1$. *Then* $K \le \left\lceil \frac{t_0^{\gamma}}{\beta\gamma} \right\rceil$.

**Long-step Algorithms**

**Lemma 3.16** The total number of outer iterations in both cases is the same.

$$\frac{1}{\theta} \log\frac{n}{\varepsilon} \tag{3-29}$$

Proof: The number of outer iterations is the number of iterations *K* necessary to obtain $n\mu \le \varepsilon$ . Previous and new $\mu$ are related as follows $\mu := (1-\theta)\mu$ . Thus, $n\mu \le \varepsilon$ can be written as $\mu_0(1-\theta)^K \le \frac{\varepsilon}{n}$. We can assume that $\mu_0 = 1$ and by taking the logarithm of both sides of the inequality we obtain $K\log(1-\theta) \le \log\frac{\varepsilon}{n}$. Using the Taylor theorem for $\log(1-\theta)$ we obtain $K \le \frac{1}{\theta} \log\frac{n}{\varepsilon}$ proving the lemma.

Now we need to estimate the upper bound on the total number of inner iterations per one outer iteration for the large-step methods. That number is equal to the number of iterations necessary to return to the situation $\Psi(\upsilon) \le \tau$ . We denote the value of $\Psi(\upsilon)$ after the $\mu$ update

as $\Psi_0$. The subsequent values in the same outer iteration are denoted as $\Psi_k$, $k = 1,2,\cdots,K$ where $K$ denotes the total number of inner iterations in the outer iteration. By using (3-9), we have

$$\Psi_0 \le n\psi\left(\frac{1+\frac{\tau}{n}+\sqrt{\left(\frac{\tau}{n}\right)^2+\frac{2\tau}{n}}}{\sqrt{1-\Theta}}\right).$$

Since $\psi(t) \le \frac{t^{p+1}-1}{p+1}$ when $t \ge 1$ and $1-(1-\theta)^{\frac{p+1}{2}} \le \theta$, after some elementary reductions, we obtain:

$$\Psi_0 \le \frac{n\theta+(p+1)\tau+n(p+1)\sqrt{\left(\frac{\tau}{n}\right)^2+\frac{2\tau}{n}}}{(p+1)(1-\theta)^{\frac{p+1}{2}}}. \tag{3-30}$$

Now Theorem 3.4 leads to

$$\Psi_{k+1} \le \Psi_k - \beta(\Psi_k)^{1-\gamma}, \qquad k = 0,1,....,K-1, \tag{3-31}$$

where $\beta = \frac{1}{60(p+q)}$ and $\gamma = \frac{p+q}{q(p+1).}$. Using Lemma 3.15 and (3-30) and (3-31) we obtain the following upper bound on the number K of inner iterations.

$$K \le 60q(p+1)(\Psi_0)^{\frac{p+q}{q(p+1)}} \le 60q(p+1)\left(\frac{n\theta+(p+1)\tau+n(p+1)\sqrt{\left(\frac{\tau}{n}\right)^2+\frac{2\tau}{n}}}{(p+1)(1-\theta)^{\frac{p+1}{2}}}\right)^{\frac{p+q}{q(p+1)}} \quad (3\text{-}32)$$

Now we can derive an upper bound on the total number iterations needed by the large-update version of the Generic IPM in Figure 2.1. According to Lemma 3.16 the number of outer iterations is bounded above by

$$\frac{1}{\theta}\log\frac{n}{\varepsilon}$$

By multiplying the number of outer iterations and the number of inner iterations obtained in (3-32) in the lemma above we get an upper bound for the total number of iterations

$$\frac{60q(p+1)}{\theta}\left(\frac{n\theta+(p+1)\tau+n(p+1)\sqrt{\left(\frac{\tau}{n}\right)^2+\frac{2\tau}{n}}}{(p+1)(1-\theta)^{\frac{p+1}{2}}}\right)^{\frac{p+q}{q(p+1)}}\log\frac{n}{\varepsilon}.$$

For large update methods we know that $\theta = \Theta(1)$, and $\tau = O(n)$. After some elementary transformations the iteration bound reduces to the following bound

$$O\left(q\,n^{\frac{p+q}{q(p+q)}}\log\frac{n}{\varepsilon}\right) \quad (3\text{-}33)$$

This result is summarized in the theorem below

**Theorem 3.5**: Given that $\theta = \Theta(1)$ and $\tau = O(n)$ which are characteristics of the large-update methods the Generic IPM described in the Figure 2.1 will obtain $\varepsilon$ - appropriate solutions of (P) and (D) in at most $O\left( q\, n^{\frac{p+q}{q(p+q)}} \log\frac{n}{\varepsilon} \right)$ iterations.

The obtained complexity result contains several previously known complexity results as special cases.

1. When $p = 1$ and $q > 1$, the kernel function $\psi(t)$ becomes the prototype self-regular function. If in addition, $q = \log n$ the iteration bound reduces to the best known bound for self-regular function, which is $O\left( \sqrt{n} \log n \log\frac{n}{\varepsilon} \right)$.

2. Letting $p = 1$ and $q = 1$, the iteration bound becomes $O\left( n \log\frac{n}{\varepsilon} \right)$ and $\psi(t)$ represents the classical logarithmic kernel function.

3. For $q = 2$ and $p = 0$, $\psi(t)$ represents the simple kernel function $\psi(t) = t - \frac{1}{t} - 2$ which is not self-regular. The iteration bound is the same as the one obtained for the logarithmic kernel function.

## Short-Step Algorithms

To get the best possible bound for short-step methods we need to use the bound described in (3-10).

$$\Psi_0 \le n\psi\left(\frac{1+\sqrt{\frac{\tau}{n}+\frac{\tau^2}{n^2}+\frac{\tau}{n}\sqrt{\frac{\tau^2}{n^2}+\frac{2\tau}{n}}}}{\sqrt{1-\theta}}\right) \le \frac{(p+q)n}{2}\left(\frac{1+\sqrt{\frac{\tau}{n}+\frac{\tau^2}{n^2}+\frac{\tau}{n}\sqrt{\frac{\tau^2}{n^2}+\frac{2\tau}{n}}}}{\sqrt{1-\theta}}-1\right)^2$$

Using $1-\sqrt{1-\theta}=\frac{\theta}{1+\sqrt{1-\theta}} \le \theta$ the above inequality can be simplified to

$$\Psi_0 \le \frac{p+q}{2(1-\theta)}\left(\theta\sqrt{n+\sqrt{\tau+\frac{\tau^2}{n}+\tau\sqrt{\frac{\tau^2}{n^2}+\frac{2\tau}{n}}}}\right)^2 \tag{3-34}$$

Following the same line of arguments as in the above subsection 3.3.1 we conclude that the total number $K$ of inner iterations is bounded above by

$$K \le 60q(p+1)(\Psi_0)^{\frac{p+q}{q(p+1)}} \le \frac{p+q}{(1-\theta)}\left(\theta\sqrt{n+\sqrt{\tau+\frac{\tau^2}{n}+\tau\sqrt{\frac{\tau^2}{n^2}+\frac{2\tau}{n}}}}\right)^{\frac{2(p+q)}{q(p+1)}} \tag{3-35}$$

Given the upper bound on the number of the outer iterations as mentioned in the previous subsection 3.3.1 the upper bound on the total number of iterations is

$$\frac{60q(p+q)}{\theta(1-\theta)}\left(\theta\sqrt{n}+\sqrt{\tau\frac{\tau^2}{n}+\tau\sqrt{\frac{\tau^2}{n^2}+\frac{2\tau}{n}}}\right)^{\frac{2(p+q)}{q(p+1)}}\log\frac{n}{\varepsilon}. \tag{3-36}$$

For small update methods it is well known that $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$ and $\tau = O(1)$. After some elementary reductions one easily obtains that the iteration bound is $O\left(q^2\sqrt{n}\log\frac{n}{\varepsilon}\right)$. We summarize this result in the theorem below.

**Theorem 3.6**: Given that $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$ and $\tau = O(1)$ which are characteristics of the small update methods the Generic IPM described in the Figure 2.1 we will obtain $\varepsilon$ - appropriate solutions of (P) and (D) in at most $O\left(q^2\sqrt{n}\log\frac{n}{\varepsilon}\right)$ iterations.

# CHAPTER 4

# ANALYSIS OF THE ALGORITHM FOR ADDITIONAL KERNEL FUNCTIONS

## Summary of the algorithm analysis

Looking carefully at the analysis of the Generic IPM described in Chapter 3 the procedure can be summarized in the following way.

Step 0: Input a kernel function $\psi$; an update parameter $\theta,\ 0<\theta<1$; a threshold parameter $\tau$; and an accuracy parameter $\varepsilon$.

Step 1: Solve the equation $-\frac{1}{2}\psi'=s$ to get $\rho(s)$ the inverse function of $-\frac{1}{2}\psi'(t), t\in(0,1]$. If the equation is hard to solve, derive a lower bound for $\rho(s)$.

Step 2: Calculate the decrease of $\Psi(t)$ in terms of $\delta$ for the default step-size $\tilde{\alpha}$ from $f(\tilde{\alpha})\le-\frac{\delta^2}{\psi''(\rho(2\delta))}$.

Step 3: Solve the equation $\psi(t)=s$ to get $\varphi(s)$, the inverse function of $\psi(t), t\ge 1$. If the equation is hard to solve, derive lower and upper bounds for $\varphi(s)$.

Step 4: Derive a lower bound for $\delta$ in terms of $\Psi(\upsilon)$ by using $\delta(v)\ge\frac{1}{2}\psi'(\varphi(\Psi(\upsilon)))$.

Step 5: Using the results of Step 3 and Step 4 find a valid inequality of the form $f(\tilde{\alpha})\le-\kappa\Psi(\upsilon)^{1-\gamma}$ for some positive constants $\kappa$ and $\gamma\in(0,1]$.

Step 6: Calculate the upper bound of $\Psi_0$ from

$$\Psi_0 \le L_\psi(n,\theta,\tau) = n\psi\left(\frac{\varphi\left(\frac{\tau}{n}\right)}{\sqrt{1-\theta}}\right) \le \frac{n}{2}\psi"(1)\left(\frac{\varphi\left(\frac{\tau}{n}\right)}{\sqrt{1-\theta}}-1\right)^2.$$

Step 7: Derive an upper bound for the total number of iterations from

$$\frac{\Psi_0^\gamma}{\theta\kappa\gamma}\log\frac{n}{\varepsilon}.$$

Step 8: Set $\tau = O(n)\, and\, \theta = \Theta(1)$ so as to calculate complexity bound for large-update methods, or set $\tau = O(1)$ and $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$ so as to calculate the complexity bound for small update methods.

## Additional Kernel Functions

We will consider the following additional kernel functions.

$$\psi_1(t) = \frac{t^2-1}{2} + \frac{t^{1-q}-1}{q(q-1)} - \frac{q-1}{q}(t-1), \qquad q>1.$$

(4-1)

$$\psi_2(t) = \frac{t^2-1}{2} + \frac{e^{\frac{1}{t}}-e}{e} \qquad (4\text{-}2)$$

$$\psi_3(t) = \frac{t^2 - 1}{2} - \int_1^t e^{\frac{1}{\zeta} - 1} d\zeta \qquad (4\text{-}3)$$

The growth term in all of them is the same $\psi_g(t) = \frac{t^2 - 1}{2}$ while the barrier term varies $\psi_b(t)$. The reason for considering this growth term is that according to the analysis above it seems to give the best complexity results and, thus, will give a more consistent view of the complexity analysis.

The following lemmas are useful for the above kernel functions. They are variations of the similar lemmas in the previous chapter and actually they are also valid for the class of kernel functions (3-1) used in that chapter. Their main purpose is to help facilitate the summary analysis described in the previous subsection.

**Lemma 4.1** When $\psi(t) = \psi_i(t)$ for $1 \le i \le 3$, then $\sqrt{1+2s} \le \varphi(s) \le 1 + \sqrt{2s}$.

Proof: The inverse function of $\psi(t)$ for $t \in [1, \infty)$ is obtained by solving for $t$ from the equation $\psi(t) = s$, for $t \ge 1$. In almost all cases it is hard to solve this equation explicitly. However, we can easily find a lower and an upper bound for $t$ and this suffices for our goal. First one has

$$s = \psi(t) = \frac{t^2 - 1}{2} + \psi_b(t) \le \frac{t^2 - 1}{2},$$

where $\psi_b(t)$ denotes the barrier term. The inequality is due to the fact that $\psi_b(1) = 0$ and $\psi_b(t)$ is monotonically decreasing. It follows that

$$t = \varphi(s) \geq \sqrt{1+2s}$$

For the second inequality we use the fact that $\psi_i''(t) \geq 1 \text{ for } 1 \leq i \leq 3$. Note that $\psi_i(t)$ are nonnegative strictly convex functions such that $\psi_i(1) = 0$. This implies that $\psi_i(t)$ is twice differentiable and therefore is completely determined by its second derivative

$$\psi_i(t) = \int_1^t \int_1^\xi \psi_i''(\varsigma)\, d\varsigma\, d\xi \tag{4-4}$$

Thus

$$s = \psi_i(t) = \int_1^t \int_1^\xi \psi_i''(\varsigma)\, d\varsigma\, d\xi \geq \int_1^t \int_1^\xi d\varsigma\, d\xi = \frac{1}{2}(t-1)^2$$

which implies

$$t = \varphi(s) \leq 1 + \sqrt{2s}\ .$$

This completes the proof.

**Lemma 4.2** Let $1 \leq i \leq 3$. Then one has $L_\psi(n,\theta,\tau) \leq \frac{\psi''(1)}{2} \frac{\left(\sqrt{2\tau} + \theta\sqrt{n}\right)^2}{1-\theta}$. Hence, if $\tau = O(1)$ and $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$, *then* $\Psi_0 = O(\psi''(1))$.

Prof: By Lemma 4.1 we have $\varphi(s) \leq 1 + \sqrt{2s}$. Hence, by using Theorem 3.2 and first inequality in (3-8) we have

$$\Psi_0 \le L_\psi(n,\theta,\tau) = n\psi\left(\frac{\varphi\left(\frac{\tau}{n}\right)}{\sqrt{1-\theta}}\right) \le n\psi\left(\frac{1+\sqrt{\frac{2\tau}{n}}}{\sqrt{1-\theta}}\right) \tag{4-5}$$

By using Taylor theorem and the fact that $\psi(1) = \psi'(1) = 0$ we obtain

$$\psi(t) = \frac{1}{2}\psi''(1)(t-1)^2 + \frac{1}{3!}\psi''(\xi)(\xi-1)^3$$

Given the fact that $\psi'''(\xi) < 0$ we obtain for $t > 1$ and $1 < \xi < t$

$$\psi(t) \le \frac{1}{2}\psi''(1)(t-1)^2 \tag{4-6}$$

Applying (4-6) to (4-5)with $t = \dfrac{1+\sqrt{\frac{2\tau}{n}}}{\sqrt{1-\theta}}$ we obtain

$$\psi_0 \le \frac{n\psi''(1)}{2}\left(\frac{1+\sqrt{\frac{2\tau}{n}}}{\sqrt{1-\theta}} - 1\right)^2 \le \frac{n\psi''(1)}{2}\left(\frac{\theta+\sqrt{\frac{2\tau}{n}}}{\sqrt{1-\theta}}\right)^2 = \frac{\psi''(1)}{2}\frac{\left(\sqrt{2\tau}+\theta\sqrt{n}\right)^2}{1-\theta}$$

where we also used the fact that $1-\sqrt{1-\theta} = \dfrac{\theta}{1+\sqrt{1-\theta}} \le \theta$. This proves the lemma.

**Lemma 4.3** Let $\eta : [0,\infty) \to (0,1]$ be the inverse function of the restriction of $-\psi_b'(t)$ to the interval $(0,1]$ where $\psi_b(t)$ is the barrier term in the kernel functions $\psi_i(t), 1 \le i \le 3$. Then

$$\rho(s) \ge \eta(1+2s).$$

Proof: Let $t = \rho(s)$. Due to the definition of $\rho$ as the inverse function of

$-\frac{1}{2}\psi'(t)$ *for* $t \le 1$ this means that

$$-2s = \psi'(t) = t + \psi_b'(t),\ t \le 1.$$

Since $t \le 1$ this implies

$$-\psi_b'(t) = t + 2s \le 1 + 2s$$

Since $-\psi_b'(t)$ is monotonically decreasing in all three cases, it follows that

$$t = \rho(s) \ge \eta(1 + 2s),$$

proving the lemma.

**Analysis of the Generic IPM with Additional Kernel Functions**

In this subsection we will provide the analysis of the Generic IPM using additional kernel functions stated in the subsection. We will follow the steps described earlier.

**Example**

Consider the function

$$\psi_1(t) = \psi(t) = \frac{t^2 - 1}{2} + \frac{t^{1-q} - 1}{q(q-1)} - \frac{q-1}{q}(t-1), \qquad q > 1.$$

Step 1: The inverse function of $-\psi_b'(t) = 1 + \frac{t - q - 1}{q}$ is given by $\eta(s) = -\frac{1}{(1 + q(s-1))^{\frac{1}{q}}}$.

Hence, by Lemma 6.3, $\rho(s) = \dfrac{1}{(1+2qs)^{\frac{1}{q}}}$.

Step 2: It follows that

$$f(\alpha) \le -\frac{\delta^2}{\psi''(\eta(2\delta))} = -\frac{\delta^2}{1+\dfrac{1}{\rho(2\delta)^{q+1}}} \le -\frac{\delta^2}{1+(1+4q\delta)^{\frac{q+1}{q}}}. \tag{4-7}$$

Step 3: By Lemma 6.1 the inverse function of $\psi(t)$ *for* $t \in [1,\infty)$ satisfies

$$\sqrt{1+2s} \le \varphi(s) \le 1+\sqrt{2s}$$

Omitting the argument $\upsilon$, we therefore have

$$\varphi(\Psi(\upsilon)) \ge \sqrt{1+2\Psi}\ .$$

Step 4: Now using the fact that $\delta(\upsilon) \ge \dfrac{1}{2}\psi'(\varphi(\Psi(\upsilon)))$, and assuming $\Psi \ge \tau \ge 1$, we obtain

$$\delta \ge \frac{1}{2}(\sqrt{1+2\Psi} - 1 + \frac{1}{q}\left(1 - \frac{1}{(1+2\Psi)^q}\right) \ge \frac{1}{2}\left(\sqrt{1+2\Psi} - 1\right) = \frac{\Psi}{1+\sqrt{1+2\Psi}}. \tag{4-8}$$

Step 5: Combining (4-7) and (4-8) after some elementary reductions, we obtain

$$f(\widetilde{\alpha}) \le -\frac{\delta^2}{1+(1+4q\delta)^{\frac{q+1}{q}}} \le -\frac{1}{53}\Psi^{\frac{q-1}{2q}}. \tag{4-9}$$

Thus, it follows that

$$\Psi_{k+1} \le \Psi_k - \kappa\left(\Psi_k\right)^{1-\gamma}, \quad k = 0,1,...,K-1$$

with $\kappa = \frac{1}{53q}$ and $\gamma = \frac{q+1}{2q}$, and K denotes the number of inner iterations. Hence, by Lemma 3.15 the number K of inner iterations is bounded above by

$$K \leq \frac{\Psi_0}{k\gamma} = \frac{106q^2}{q+1}\Psi_0 \leq 106q\Psi_0^{\frac{q+1}{2q}}. \tag{4-10}$$

Step 6: To estimate $\Psi_0$ we use Lemma 6.2, with $\psi''(1) = 2$. Thus, we obtain

$$\Psi_0 \leq \frac{\left(\theta\sqrt{n} + \sqrt{2\tau}\right)^2}{1-\theta}.$$

Step 7: Thus, using Lemma 3.16 the total number of iterations is bounded above by

$$\frac{K}{\theta}\log\frac{n}{\varepsilon} \leq \frac{106q}{\theta}\left(\frac{\left(\theta\sqrt{n} + \sqrt{2\tau}\right)^2}{1-\theta}\right)^{\frac{q+1}{2q}}\log\frac{n}{\varepsilon}. \tag{4-11}$$

Step8: For large update methods ( with $\tau = O(n)\, and\, \theta = \Theta(1)$ ) the right hand side expression becomes

$$O\left(qn^{\frac{q+1}{2q}}\log\frac{n}{\varepsilon}\right). \tag{4-12}$$

For small update methods ( with $\tau = O(1)$ and $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$ ) the right hand side expression becomes

$$O\left(q\sqrt{n}\log\frac{n}{\varepsilon}\right). \tag{4-13}$$

**Example**

Consider the kernel function

$$\psi_2(t) = \psi(t) = \frac{t^2 - 1}{2} + \frac{e^{\frac{1}{t}} - e}{e}.$$

Step 1: The inverse function of $-\psi_b'(t) = \frac{e^{\frac{1}{t}} - 1}{t^2}$ is such that $\eta(s) = t \Leftrightarrow \frac{e^{\frac{1}{t}} - 1}{t^2} = s,\ t \le 1$. It follows that $\frac{e^{\frac{1}{t}} - 1}{t^2} = st^2 \le s$ whence we obtain $\eta(s) = t \ge \frac{1}{1 + \log s}$. Hence, by Lemma4.3, $\rho(s) \ge \eta(1 + 2s)$.

Step 2: Since $\psi''(t)$ is monotonically decreasing we have

$$f(\tilde{\alpha}) \le -\frac{\delta^2}{\psi''(\rho(2\delta))} \le \frac{\delta^2}{\psi''\left(\eta(1 + 4q\delta)\right)}.$$

Now putting $t = \eta(1 + 4q\delta)$ we have $t \le 1$ and we may write

$$f(\alpha) \le -\frac{\delta^2}{\psi''(t)} = -\frac{\delta^2}{1 + \frac{1 + 2t}{t^4} e^{\frac{1}{t} - 1}} \le \frac{\delta^2}{1 + \frac{3}{t^4} e^{\frac{1}{t} - 1}} = \frac{\delta^2}{1 + \frac{3(1 + 4\delta)}{t^2}} \le \frac{\delta^2}{1 + \frac{15\delta}{t^2}}.$$

Since $\frac{1}{t^2} = \frac{1}{\left(\eta(1 + 4\delta)\right)^2} \le (1 + \log(1 + 4\delta))^2$ we finally get

$$f(\alpha) = \frac{\delta^2}{1 + 15\delta(1 + \log(1 + 4\delta))^2}. \qquad (4\text{-}14)$$

Step 3: By Lemma 6.1 the inverse function of $\psi(t)$ for $t \in [1,\infty)$ satisfies

$\sqrt{1+2s} \le \varphi(s) \le 1+\sqrt{2s}$. Omitting the argument $\upsilon$, we thus have $\varphi(\Psi(\upsilon)) \ge \sqrt{1+2\Psi}$.

Step 4: Now using that $\delta(\upsilon) \ge \frac{1}{2}\psi'(\varphi(\Psi(\upsilon)))$, we obtain

$$\delta \ge \frac{1}{2}\left(\sqrt{1+2\Psi} - \frac{e^{\frac{1}{\sqrt{1+2\Psi}}-1}}{1+2\Psi}\right) \ge \frac{1}{2}\left(\sqrt{1+2\Psi}-1\right) = \frac{\Psi}{1+\sqrt{1+2\Psi}}. \tag{4-15}$$

Step 5: Substitution of ((4-15) into the (4-14) gives, after some elementary reductions, while assuming $\Psi_0 \ge \Psi \ge \tau \ge 1$

$$f(\alpha) \le -\frac{\Psi^{\frac{1}{2}}}{44(1+\log(1+\sqrt{2\Psi}))^2} \le -\frac{\Psi^{\frac{1}{2}}}{44(1+\log(1+\sqrt{2\Psi_0}))^2}.$$

Thus, it follows that

$$\Psi_{k+1} \le \Psi_k - \kappa(\Psi_k)^{1-\gamma}, \quad k = 0,1,...,K-1$$

with $\kappa = \frac{1}{44(1+\log(1+\sqrt{2\Psi_0}))^2}$ and $\gamma = \frac{1}{2}$, and K denotes the number of inner iterations.

Hence, by Lemma 3.15 the number K of inner iterations is bounded above by

$$K \le \frac{\Psi_0}{k\gamma} = 88(1+\log(1+\sqrt{2\Psi_0}))^2 \Psi_0^{\frac{1}{2}}.$$

Step 6: We use Lemma 4.2 with $\psi''(1) = 4,$ to estimate $\Psi_0$.

$$\Psi_0 \le 2\frac{\left(\theta\sqrt{n}+\sqrt{2\tau}\right)^2}{1-\theta}. \tag{4-16}$$

Substitution of (4-16) in the expression for $K$ gives

$$K \le 88\sqrt{2}\left(1+\log\left(1+2\frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}\right)\right)^2 \frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}. \tag{4-17}$$

Step 7: Thus the total number of iterations is bounded above by

$$\frac{K}{\theta}\log\frac{n}{\varepsilon} \le 88\sqrt{2}\left(1+\log\left(1+2\frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}\right)\right)^2 \frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}\log\frac{n}{\varepsilon}. \tag{4-18}$$

Step 8: For large update methods, when $\tau = O(n)$ *and* $\theta = \Theta(1)$ the right hand side expression (4-18) becomes

$$O\left(\sqrt{n}(\log n)^2 \log\frac{n}{\varepsilon}\right). \tag{4-19}$$

For small update methods, when $\tau = O(1)$ and $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$, the right hand side expression (4-18) becomes

$$O\left(\sqrt{n}\log\frac{n}{\varepsilon}\right) \tag{4-20}$$

**Example**

Consider the kernel function

$$\psi_3(t) = \psi(t) = \frac{t^2 - 1}{2} - \int_1^t e^{\frac{1}{\zeta}-1} d\zeta \, .$$

Step 1: The inverse function of $-\psi'_b(t) = e^{\frac{1}{\varepsilon}-1}$ is given by $\eta(s) = \frac{1}{1+\log s}$. Hence, by Lemma 6.3,

$$\rho(s) \geq \frac{1}{1+\log(1+2s)} \, .$$

Step 2: It follows that

$$f(\alpha) \leq -\frac{\delta^2}{\psi''(\rho(2\delta))} = -\frac{\delta^2}{1+\frac{e^{\frac{1}{\rho(2\delta)}-1}}{\rho(2\delta)^{q+1}}} \leq -\frac{\delta^2}{1+(1+4q\delta)(1+\log(1+4\delta))^2} \, .$$

(4-21)

Step 3: By Lemma 4.1 the inverse function of $\psi(t)$ for $t \in [1,\infty)$ satisfies

$$\sqrt{1+2s} \leq \varphi(s) \leq 1+\sqrt{2s} \, .$$

Thus we have, omitting the argument $\upsilon$,

$$\varphi(\Psi(\upsilon)) \geq \sqrt{1+2\Psi} \, .$$

Step 4: Now using that $\delta \geq \frac{1}{2}\psi'(\varphi(\Psi(\upsilon)))$ we obtain

$$\delta \geq \frac{1}{2}\left(\sqrt{1+2\Psi} - e^{\frac{1}{\sqrt{1+2\Psi}}-1}\right) \geq \frac{1}{2}\left(\sqrt{1+2\Psi}-1\right) = \frac{\Psi}{1+\sqrt{1+2\Psi}} \, . \qquad (4\text{-}22)$$

Step 5: Substitution of (4-22) into the (4-21) gives, after some elementary reductions, while assuming $\Psi_0 \geq \Psi \geq \tau \geq 1$

$$f(\alpha) \leq -\frac{\Psi^{\frac{1}{2}}}{21(1+\log(1+\sqrt{\Psi}))^2} \leq -\frac{\Psi^{\frac{1}{2}}}{21(1+\log(1+\sqrt{\Psi_0}))^2}.$$

Thus, it follows that

$$\Psi_{k+1} \leq \Psi_k - \kappa(\Psi_k)^{1-\gamma}, \quad k = 0,1,...,K-1$$

with $\kappa = \frac{1}{21(1+\log(1+\sqrt{2\Psi_0}))^2}$ and $\gamma = \frac{1}{2}$, and K denotes the number of inner iterations.

Hence, by Lemma 3.15 the number K of inner iterations is bounded above by

$$K \leq \frac{\Psi_0}{k\gamma} = 42(1+\log(1+\sqrt{2\Psi_0}))^2 \Psi_0^{\frac{1}{2}}. \tag{4-23}$$

We use Lemma 4.2, with $\psi''(1) = 2$ to estimate $\Psi_0$. This gives

$$\Psi_0 \leq \frac{\left(\theta\sqrt{n} + \sqrt{2\tau}\right)^2}{1-\theta}. \tag{4-24}$$

Substitution of (4-24) into (4-24) leads to the following estimate for number K of inner iterations

$$K \leq 42\left(1+\log\left(1+\frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}\right)\right)^2 \frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}. \tag{4-25}$$

Step 7: By Lemma 3.16 the total number of iterations is bounded above by

$$\frac{K}{\theta}\log\frac{n}{\varepsilon} \le 42\left(1+\log\left(1+\frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}\right)\right)^2 \frac{\theta\sqrt{n}+\sqrt{2\tau}}{\sqrt{1-\theta}}\log\frac{n}{\varepsilon} \tag{4-26}$$

Step 8: For large-update methods when $\tau = O(n)$ *and* $\theta = \Theta(1)$ the right hand side expression (4-26) becomes

$$O\left(\sqrt{n}(\log n)^2 \log\frac{n}{\varepsilon}\right). \tag{4-27}$$

For small update methods, when $\tau = O(1)$ and $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$, the right hand side expression (4-26) becomes

$$O\left(\sqrt{n}\log\frac{n}{\varepsilon}\right). \tag{4-28}$$

**Summary of complexity results**

The complexity results for Generic IPM with kernel functions defined in (3-1) and in (4-1)-(4-3) are summarized in the Table for large-step methods and in the table for small-step methods. For the class of kernel functions (3-1) we consider three special cases,

- the logarithmic kernel function: $\psi(t) = \frac{t^2-1}{2} - \log t$
- the classical self-regular function when $p = 1$: $\psi(t) = \frac{t^2-1}{2} + \frac{t^{1-q}-1}{q-1}$

- the linear non-self-regular function when $p = 0 : \psi(t) = t - 1 + \frac{t^{1-q} - 1}{q - 1}$

**Complexity of large-update methods**

The complexity results for large-step methods are summarized below. They are obtained by taking into the account that $\tau = O(n)$ *and* $\theta = \Theta(1)$.

Complexity results for long-step methods

| i | Kernel Function $\psi_i(t)$ | Iteration Bound |
|---|---|---|
| 1 | $\frac{t^2 - 1}{2} - \log t$ | $O(n)\log\frac{n}{\varepsilon}$ |
| 2 | $\frac{t^2 - 1}{2} + \frac{t^{1-q} - 1}{q - 1}$ | $O(qn^{\frac{q+1}{2q}})\log\frac{n}{\varepsilon}$ |
| 3 | $t - 1 + \frac{t^{1-q} - 1}{q - 1}$ | $O(qn)\log\frac{n}{\varepsilon}$ |
| 4 | $\frac{t^2 - 1}{2} + \frac{t^{1-q} - 1}{q(q-1)} - \frac{q-1}{q}(t-1)$ | $O(qn^{\frac{q+1}{2q}})\log\frac{n}{\varepsilon}$ |
| 5 | $\frac{t^2 - 1}{2} + \frac{e^{\frac{1}{t}} - e}{e}$ | $O(\sqrt{n}\log^2 n)\log\frac{n}{\varepsilon}$ |
| 6 | $\frac{t^2 - 1}{2} - \int_1^t e^{\frac{1}{\varepsilon}-1} d\xi$ | $O(\sqrt{n}\log^2 n)\log\frac{n}{\varepsilon}$ |

Table 2

Notice that the best bound is obtained in case of 3 and 4 by taking $q = \frac{1}{2}\log n$ which gives the iteration bound of

$$O\left(\sqrt{n}\,\log n \log\frac{n}{\varepsilon}\right), \tag{4-29}$$

which is currently the best known bound for large-update methods.

**Complexity of small update methods**

The complexity results for small-update methods are summarized below. They are obtained by taking into the account that $\theta = \Theta\left(\frac{1}{\sqrt{n}}\right)$ and $\tau = O(1)$.

Complexity results for short-step methods

| i | Kernel Function $\psi_i(t)$ | Iteration Bound |
|---|---|---|
| 1 | $\frac{t^2-1}{2} - \log t$ | $O(n)\log\frac{n}{\varepsilon}$ |
| 2 | $\frac{t^2-1}{2} + \frac{t^{1-q}-1}{q-1}$ | $O(q^2\sqrt{n})\log\frac{n}{\varepsilon}$ |
| 3 | $t-1+\frac{t^{1-q-1}}{q-1}$ | $O(q^2\sqrt{n})\log\frac{n}{\varepsilon}$ |
| 4 | $\frac{t^2-1}{2} + \frac{t^{1-q}-1}{q(q-1)} - \frac{q-1}{q}(t-1)$ | $O(q\sqrt{n})\log\frac{n}{\varepsilon}$ |
| 5 | $\frac{t^2-1}{2} + \frac{e^{\frac{1}{t}}-e}{e}$ | $O(q\sqrt{n})\log\frac{n}{\varepsilon}$ |
| 6 | $\frac{t^2-1}{2} - \int_1^t e^{\frac{1}{\varepsilon}-1} d\xi$ | $O(q\sqrt{n})\log\frac{n}{\varepsilon}$ |

Table 3

The above table shows that the small-update methods based on listed kernel functions all have the same complexity, namely

$$O\left(\sqrt{n}\log\frac{n}{\varepsilon}\right). \tag{4-30}$$

This is up till now the best iteration bound for IPMs solving LP problems.

Historically most of the IPMs were based on the logarithmic kernel function. Notice that the gap between theoretical complexity of short-step methods and large-step methods is significant; the short step methods have much better theoretical complexity. However, in practical implementations large-step methods work better. This discrepancy was one of the motivations to consider other kernel functions in hopes to find the kernel function which would not have a gap or the gap would be smaller. As we can see, this goal has been achieved; for cases 2 and 4 the gap is much smaller because for these kernel functions large step method has much better complexity than for the classical logarithmic kernel function. This is one of the main achievements of considering different classes of kernel functions.

# CHAPTER 5

## NUMERICAL RESULTS

The Generic IPM described in the Figure 1 was implemented in MATLAB 7.6.0 with the class of kernel functions described by formula (3-1)

$$\psi_{p,q}(t) = \begin{cases} \dfrac{t^{p+1}-1}{p+1} + \dfrac{t^{1-q}-1}{q-1}, & t > 0,\ p \in [0,1],\ q > 1 \\ \dfrac{t^{p+1}-1}{p+1} - \log t, & t > 0,\ p \in [0,1] \end{cases}$$

This imply that there are two implementations of the algorithm, one for the classical logarithmic kernel function

$$\psi(t) = \frac{t^2-1}{2} - \log t \tag{5-1}$$

and one for the kernel function with parameters $p$ and $q$

$$\psi_{p,q}(t) = \frac{t^{p+1}-1}{p+1} + \frac{t^{1-q}-1}{q-1}, \quad t > 0,\ p \in [0,1],\ q > 1 \tag{5-2}$$

We call the first implementation “Classical Method” and the second implementation “New Method”. Both codes are listed in the Appendix A.

The algorithm was tested on several examples with different sizes ranging from very small to moderate size problems. The data was entered in some cases “by hand” and for the others they were generated randomly.

**Example:** Consider the following simple LP model.

$$\begin{aligned} \max \quad & 2x_1 + 2x_2 \\ s.t. \quad & x_1 + x_2 \le 3 \\ & x_1 \ge 0,\ x_2 \ge 0. \end{aligned}$$

It is easy to see that this problem has infinitely many optimal solutions, they are all the points on the segment $[(1,0);(0,1)]$ and the optimal value is $Z = 6$. The problem was solved by New Method with $p = 0.8,\ q = 1.2$ with accuracy parameter $\varepsilon = 0.001$. It took unusually many iterations (57), however algorithm steadily converged to the expected result $(x_1, x_2) = (1.5,1,5)$.

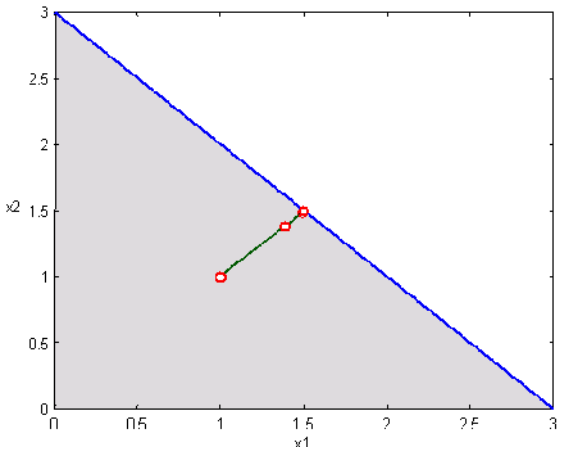
Numerical results for Example

| x | | | y | s | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1.383 | 1.383 | 0.234 | -0.8234 | 0.1 | 0.1 | 1.2489 |
| 1.4883 | 1.4883 | 0.0234 | -1.6023 | 0.0656 | 0.0656 | 1.7567 |
| .... | …. | ….. | ……. | …… | …… | …… |
| 1.4999 | 1.4999 | 0.0002 | -2.0002 | 0.0002 | 0.0002 | 2.0002 |

Table 4

Objective function value Z = -5.9997

This example also illustrates an important feature of the interior-point methods that distinguish them from simplex-type methods; that in the case of infinitely many optimal solutions they converge to the center of the optimal set rather than to the vertex. The graphical illustration of the above example with several first iterations is given below.




IPM Based on generic kernel function

Figure 3

Next, the algorithm was examined on the set of 200 randomly generated "small" problems for with sizes less than 10. The average number of iteration and CPU time is given in the table below.

| Classical method | | New method | |
|---|---|---|---|
| Average Number Of Iterations | Time | Average Number Of Iterations | Time |
| 24.6 | 0.0362 | 40.7 | 0.0448 |

Numerical results for randomly generated "small problems" with dimension less than 10

Table 5

Next, the algorithm was examined on the set of 200 randomly generated "moderate size" problems of the size 200x300. The average number of iteration and CPU time is given in the table below.

| Classical method | | New method | |
|---|---|---|---|
| Average Number Of Iterations | Time | Average Number Of Iterations | Time |
| 35.81 | 2.6626 | 40 | 2.8812 |

Numerical results for randomly generated problems with dimension 200x300

Table 6

The algorithm was then applied to the randomly generated problems of the bigger size 400x700. The result is given in the table below

| Classical method | | New method | |
|---|---|---|---|
| Average Number Of Iterations | Time | Average Number Of Iterations | Time |
| 57 | 35.8048 | 41 | 25.4030 |

Numerical results for randomly generated problem with dimension 400x700

Table 7

Results summarized in tables seem to suggest that Classical Methods works slightly better for the problems of the smaller size while, as the dimension of the problem increases, the New Method becomes better. Another feature of the IPM is also visible from these examples and

that is that the number of iterations does not increase significantly with the increase in the size of the problem.

In the sequel the New Method was examined on the set of 200 randomly generated "small" problems with sizes less than 10 and for different values of parameters p and q. The results are given in the Table below.

| p=0.7,q=1.3 | | p=0.6,q=1.4 | |
|---|---|---|---|
| Iteration | Time | Iteration | Time |
| 40.6 | 0.0673 | 41.6 | 0.0648 |

| p=0.9,q=1.1 | | p=0.8,q=1.2 | |
|---|---|---|---|
| Iteration | Time | Iteration | Time |
| 35.9 | 0.0526 | 37.2 | 0.0603 |

Table 8: More Numerical Results

Next, the New Method was examined on the set of 20 randomly generated problems of the size 200x300 and for different values of parameters p and q. The results are given in the Table below.

Table 9: More Numerical Results

The table seems to suggest that for the problems of the smaller size the New Method works the best when $p = 0.9,\ q = 1.1$, which is in line with theoretical expectation . However, for the problems of the higher dimension it is hard to make conclusion which combination of parameters works the best. Theory suggests that $p = 1,\ q = \log n$ where $n$ is the number of variables gives the best complexity. However for the particular set of problems in the previous table, it seems that combination $p = 0.7,\ q = 1.3$ works the best .

Better implementation and more testing is necessary for more definite conclusions. However, that was not the intention of the thesis. The goal was to make the basic implementations that ilustrates theoretical concepts discussed in the thesis. Even on this basic level the implementation of the New Method shows the potential to work well. Of coure, with more sophisticated implementation the performance can be further improved.

# CHAPTER 6

# CONCLUSION

In this thesis the Interior – Point Method (IPM) for Linear Programming problem (LP) that is based on the generic kernel function is considered. The algorithm is described in Chapter 2.

In Chapter 3 the complexity (in terms of iteration bounds) of the algorithm is analyzed for a class of kernel functions defined by (3-1). This class is fairly general; it includes classical logarithmic kernel function, prototype self-regular kernel function as well as non-self-regular functions, thus it serves as a unifying frame for the analysis of IPMs. Two versions of the IPMs are considered, the short-step algorithms where barrier parameter $\theta$ depends on the size of the problem and long-step algorithms where barrier parameter is a fixed constant $\theta \in (0,1)$.

Historically most of the IPMs were based on the logarithmic kernel function. Notice that the gap between theoretical complexity of short-step methods and large-step methods is significant; the short step methods have much better theoretical complexity. However, in practical implementations large-step methods work better. This discrepancy was one of the motivations to consider other kernel functions in hopes to find the kernel function which would not have a gap or the gap would be smaller. As we can see this goal has been achieved; for kernel functions 2 and 4, the gap is much smaller than for the classical logarithmic kernel function. In addition, the complexity results that are obtained match the best known complexity results for these methods. This chapter is mostly based on the paper (Bai, Y., et al., 2008) with the addition of most of the proofs that were omitted in the paper.

The main contribution of the thesis is contained in Chapter 4. The detailed complexity analysis of the IPM that was provided in Chapter3 for kernel function (3-1) is summarized and the analysis of the algorithm was performed for three additional kernel functions (4-1) – (4-3). For one of them we again matched the best known complexity results for the large-step methods and for the other two the complexity is slightly weaker, however still significantly improved in comparison with classical logarithmic kernel function.

The IPM that is theoretically analyzed in Chapter 3 is implemented in Chapter 5 for the classical logarithmic kernel function (Classical Method) and for the parametric kernel function (New Method) both described in (3-1). Although the implementation is on the basic level, it shows potential for a good performance of IPM based on kernel function different than classical logarithmic kernel function on which most of the commercial codes are based. The preliminary calculations seem to indicate that IPM with classical kernel logarithmic function perform better on problems of the smaller size while for larger problems the New Methods seems to work slightly better. Also, based on the preliminary numerical tests it is hard to make conclusion which combination of parameters p and q in (3-1) works the best. Better implementation and more numerical testing would be necessary to draw more definite conclusions.

However, that was not the goal of the thesis, the goal was to show that IPM with kernel functions different than classical logarithmic kernel function can have best known theoretical complexity and to show that they have potential for practical implementations.

## REFERENCES

[1] Y.Q Bai, C.Roos. A polynomial-time algorithm for linear optimization based on a new simple kernel function. Optimization Methods Software, 18 (6):631-646, 2003.

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

[3] G. Lesaja. Introducing Interior-Point Methods for Introductionary Operations Research Courses and/or Linear Programming Courses, Open Operational Research Journal, accepted, 2008.

[4] F. Hillier and G. Lieberman. Introduction to Operations Research. Seventh Edition. McGraw Hill Publishing, 2001.

[5] J. Peng, C.Roos, and T. Telarky. Self- regularity: A New Paradigm for Primal-Dual Interior Point Algorithms. Princeton University Press, 2002.

[6] C. Roos, T. Telarky, and J.-Ph.Vial. Theory and Algorithms for Linear Optimization. An Interior-Point Approach. John Wiley &Sons, Chichester, UK, 1997.

[7] S. Wright. Primal-Dual Interior Point Methods. SIAM Publishing, 1997.

[7] Y. Bai, M. El ghami, and C. Roos. A Comparative Study of New Barrier Functions for Primal – Dual Interior – Point Algorithms in Linear Optimization.

# APPENDICES

## APPENDIX A

### MatLab codes for the Classical Method

```
function [i,xx,yy,ss,z,d]=IpmClassical(A, b, c, epsilon)
%
% input tau, epsilon, theta
%
% sizes: A--m*n, b--m, s--n, x--n, y--m, c--n
%
% To call this function, please set up the problem by defining A, b and
% c. Or load the example problem.

    [m,n]=size(A);
    x=1*ones(n,1); s=1*ones(n,1); y=zeros(m,1); mu=x'*s/n;
    rd=c-A'*y-s;
    rp=b-A*x;

    i=1;
    xx(i,:)=x;
    yy(i,:)=y;
    ss(i,:)=s;

    while norm(rd)>epsilon||norm(rp)>epsilon||n*mu > epsilon
        i=i+1;
        [dx,ds,dy]=SolvesystemClassical(A,b,c,x,y,s,mu);
        inds=find(ds<0);indx=find(dx<0);
        alpha=0.9*min(abs([1;s(inds)./ds(inds);x(indx)./dx(indx)]));
        x=x+alpha*dx;
        y=y+alpha*dy;
        s=s+alpha*ds;
        mu=min(x'*s/n,0.9*mu);
        rd=c-A'*y-s;
        rp=b-A*x;

        xx(i,:)=x;
        yy(i,:)=y;
        ss(i,:)=s;
        d(i,:)=dx';
        if i>200 || alpha<1e-11
            i;
            break;
        end
    end
    z=x'*c;
    %[i,z]
end
```

```
function [dx,ds,dy]=SolvesystemClassical(A,b,c,x,y,s,mu)
% This function solves the following system
%        A*dx = 0
% A^T*dy + ds = 0
% S*dx + x*ds = - mu*v.*grad(Psi(v))
%
% Psi(v)=sum((v.^(p+1)-1)/(p+1)+(v.^(1-q)-1)/(q-1));
% grad(v)=v.^p-v.^q;

    gama=.1;
    X=diag(x);
    S=diag(s);
    S_inv=diag(1./s);
    rd=c-A'*y-s;
    rp=b-A*x;

    M=A*S_inv*X*A';
    r=b+A*S_inv*(X*rd-gama*mu*ones(size(A,2),1));

    dy=M\r;
    ds=rd-A'*dy;
    dx=-x+S_inv*(gama*mu*ones(size(A,2),1)-X*ds);

end
```

## MatLab codes for the New Method

```
function [i,xx,yy,ss,z,d]=IpmNew(A, b, c, epsilon)
%
%
% input tau, epsilon, theta
% v>0, 0<=p<=1, q>1, tau>1
% sizes: A--m*n, b--m, s--n, x--n, y--m, c--n
%
% To call this function, please set up the problem by defining A, b and
% c. Or load the example problem.

p=1-0.2;
q=1+0.2;
theta=0.1;
%tau=1.5;
    [m,n]=size(A);
    x=ones(n,1); s=ones(n,1); y=zeros(m,1); mu=x'*s/n;
    rd=c-A'*y-s;
    rp=b-A*x;
    i=1;
    xx(i,:)=x;
    yy(i,:)=y;
    ss(i,:)=s;
    while norm(rd)>epsilon||norm(rp)>epsilon||n*mu > epsilon
        i=i+1;
        v=sqrt(x.*s./mu);
        [dx,ds,dy]=SolvesystemNew(A,b,c,x,y,s,mu,v,p,q);
```

```
        delta=1/2*sqrt(sum((v.^p-1./v.^q).^2));
        alpha=1/((p+q)*(1-4*abs(delta))^(1+1/q));
        alpha=abs(alpha);
        inds=find(ds<0);indx=find(dx<0);
        alpha=0.9*min(abs([1;s(inds)./ds(inds);x(indx)./dx(indx)]));
        x=x+alpha*dx;
        y=y+alpha*dy;
        s=s+alpha*ds;

        rd=c-A'*y-s;
        rp=b-A*x;
        mu=(1-theta)*mu;
        xx(i,:)=x';
        yy(i,:)=y';
        ss(i,:)=s';
        d(i,:)=dx';
        if i>200 || alpha<1e-10
            i;
            break;
        end
    end
    z=x'*c;
    %[i,z]
end


function [dx,ds,dy]=SolvesystemNew(A,b,c,x,y,s,mu,v,p,q)
% This function solves the following system
%        A*dx = 0
% A^T*dy + ds = 0
% S*dx + x*ds = - mu*v.*grad(Psi(v))
%
% Psi(v)=sum((v.^(p+1)-1)/(p+1)+(v.^(1-q)-1)/(q-1));
% grad(v)=v.^p-v.^q;
gama=.1;
    X=diag(x);
    S=diag(s);
    S_inv=diag(1./s);
    rd=c-A'*y-s;
    rp=b-A*x;

    M=A*S_inv*X*A';
    r=A*S_inv*(X*rd+mu*v.*(v.^p-gama*v.^(-q)))+rp;

    dy=M\r;
    ds=rd-A'*dy;
    dx=-S_inv*(X*ds+mu*v.*(v.^p-gama*v.^(-q)));
end
```

## Problem Generator

```
function R=mytest(NO,m,n)
% This function solve the random systems of dimention m*n
```

```
% input m,n are the dimention of A
% input NO is the number of the iterations
%
% MYTEST();   will apply both ipm methods to a random matrix with random
% dimension m*n, where m,n are positive integers less than 10
%
% MYTEST(N)  will iterate 'MYTEST()' N times.
%
% MYTEST(N,m,n)  will iterately apply the methods to N random problems
% with dimension m*n.

    if ~exist('NO')
        NO=1;
    end
    m_ne=0;n_ne=0;
    if ~exist('m')
        m_ne=1;
    end
    if ~exist('n')
        n_ne=1;
    end

    dim=zeros(NO,2);
    k=1;
    while k<=NO
        if m_ne
            m=fix(rand(1)*10);
        end
        if n_ne
            n=fix(rand(1)*10);
        end
        A=rand(m,n);b=rand(m,1);c=rand(n,1);
        if rank(A)<min(m,n) || min(m,n)==0
            %fprintf('rank(A)<min(m,)');
            if m==0 && ~m_ne
                display('STOP, m=0');
                return;
            elseif n==0 && ~n_ne
                display('STOP, n=0');
                return;
            end
            continue;
        end
        tic;
        [i1(k),x1,yy1,ss1,z1(k)]=IpmNew(A, b, c, 0.01);
        t1(k)=toc;
        tic;
        [i2(k),xx2,yy2,ss2,z2(k)]=IpmClassical(A, b, c, 0.01);
        t2(k)=toc;
        dim(k,1)=m;dim(k,2)=n;
        k=k+1;
    end
    R = [dim(:,1),dim(:,2),i1',z1',t1',i2',z2',t2'];
    myfun®;
end
```

## Organization of output

```
function myfun®

    fprintf('       New method              Classical method \n');
    fprintf('  idx dim(m*n)  iteration Opt     time     iteration Opt
time\n');
    fprintf('-------------------------------------------------------------
--\n');
    for k=1:size(R,1)
        fprintf('   %3d %3d%3d  %3d   %6.4f   %4.4f     %3d   %6.4f
%4.4f\n', ...
                k,R(k,:));
    end
    R_ave=sum(R)/size(R,1);
    fprintf('-------------------------------------------------------------
--\n');
    fprintf('                     New method        Classical method \n');
    fprintf('         dim(m*n)   iteration time     iteration  time\n');
    fprintf(' average %3.1f %3.1f     %3.1f    %4.4f      %3.1f
%4.4f\n', ...
                R_ave([1,2,3,5,6,8]));
    fprintf('-------------------------------------------------------------
--\n');

end
```

# APENDIX B

## MatLab code Example 5.1

max 3 * x_1 + 5 * x_2

s.t.

x_1 + <=4

2 * x_2 <=12

3 * x_1 + 2 * x_2 <=18

x_1 , x_2 >= 0

A =

1 0 1 0 0

0 2 0 1 0

3 1 0 0 1

(change to min-problem)

c =

-3

-5

0

0

0

b =

4

12

18

```
>> [i,x,y,s,z]=IpmClassical(A, b, c, epsilon)
i =
     7
z =
  -41.9805

>> [i,x,y,s,z]=IpmNew(A, b, c, epsilon)

i =
    60
z =
  -41.9993

>> [i,x,y,s,z]=IpmClassical(A, b, c, epsilon)

i =
     7
x =
    1.0000    1.0000    1.0000    1.0000    1.0000
    1.4437    1.7530    0.8831    0.9645    1.0398
    2.0334    3.0422    0.7370    0.3825    0.8654
    2.9815    4.6039    0.4101    0.0542    0.4968
    3.8206    5.7689    0.0779    0.0054    0.1095
    3.9803    5.9762    0.0095    0.0020    0.0169
    3.9977    5.9975    0.0013    0.0005    0.0028
```

```
y =

       0       0       0

  -0.1333  -0.0519   0.0235

  -0.2783  -0.5925  -0.1361

  -0.5626  -1.3849  -0.4117

  -0.8317  -2.0210  -0.6562

  -0.9001  -2.1384  -0.6935

  -0.9363  -2.1549  -0.6873


s =

   1.0000   1.0000   1.0000   1.0000   1.0000

   0.4093   0.1000   0.9699   0.8885   0.8131

   0.1458   0.0100   0.8931   1.2073   0.7509

   0.0146   0.0068   0.8669   1.6891   0.7159

   0.0032   0.0026   0.8825   2.0717   0.7069

   0.0011   0.0007   0.9052   2.1434   0.6986

   0.0003   0.0002   0.9368   2.1554   0.6878


z =

  -41.9805


>> [i,x,y,s,z]=IpmNew(A, b, c, epsilon)

i =

   60

z =

  -41.9993
```

x =

| | | | | |
|---|---|---|---|---|
| 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 1.4437 | 1.7530 | 0.8831 | 0.9645 | 1.0398 |
| 2.0409 | 3.0601 | 0.7365 | 0.3780 | 0.8702 |
| 3.0192 | 4.6730 | 0.4016 | 0.0474 | 0.5044 |
| 3.8797 | 5.8542 | 0.0624 | 0.0308 | 0.1302 |
| 3.9559 | 5.9723 | 0.0383 | 0.0293 | 0.1223 |
| 3.9623 | 5.9859 | 0.0372 | 0.0256 | 0.1236 |
| 3.9669 | 5.9883 | 0.0330 | 0.0232 | 0.1105 |
| 3.9702 | 5.9896 | 0.0297 | 0.0208 | 0.0997 |
| 3.9732 | 5.9906 | 0.0268 | 0.0188 | 0.0897 |
| 3.9759 | 5.9916 | 0.0241 | 0.0169 | 0.0808 |
| 3.9783 | 5.9924 | 0.0217 | 0.0152 | 0.0727 |
| 3.9805 | 5.9932 | 0.0195 | 0.0137 | 0.0655 |
| 3.9824 | 5.9938 | 0.0176 | 0.0123 | 0.0589 |
| 3.9842 | 5.9945 | 0.0158 | 0.0111 | 0.0530 |
| 3.9857 | 5.9950 | 0.0143 | 0.0100 | 0.0478 |
| 3.9872 | 5.9955 | 0.0128 | 0.0090 | 0.0430 |
| 3.9884 | 5.9960 | 0.0116 | 0.0081 | 0.0387 |
| 3.9896 | 5.9964 | 0.0104 | 0.0073 | 0.0348 |
| 3.9906 | 5.9967 | 0.0094 | 0.0065 | 0.0314 |
| 3.9916 | 5.9971 | 0.0084 | 0.0059 | 0.0282 |
| 3.9924 | 5.9973 | 0.0076 | 0.0053 | 0.0254 |
| 3.9932 | 5.9976 | 0.0068 | 0.0048 | 0.0229 |
| 3.9939 | 5.9979 | 0.0061 | 0.0043 | 0.0206 |
| 3.9945 | 5.9981 | 0.0055 | 0.0039 | 0.0185 |
| 3.9950 | 5.9983 | 0.0050 | 0.0035 | 0.0167 |

| | | | | |
|---|---|---|---|---|
| 3.9955 | 5.9984 | 0.0045 | 0.0031 | 0.0150 |
| 3.9960 | 5.9986 | 0.0040 | 0.0028 | 0.0135 |
| 3.9964 | 5.9987 | 0.0036 | 0.0025 | 0.0122 |
| 3.9967 | 5.9989 | 0.0033 | 0.0023 | 0.0109 |
| 3.9971 | 5.9990 | 0.0029 | 0.0021 | 0.0098 |
| 3.9974 | 5.9991 | 0.0026 | 0.0018 | 0.0089 |
| 3.9976 | 5.9992 | 0.0024 | 0.0017 | 0.0080 |
| 3.9979 | 5.9993 | 0.0021 | 0.0015 | 0.0072 |
| 3.9981 | 5.9993 | 0.0019 | 0.0013 | 0.0065 |
| 3.9983 | 5.9994 | 0.0017 | 0.0012 | 0.0058 |
| 3.9984 | 5.9995 | 0.0016 | 0.0011 | 0.0052 |
| 3.9986 | 5.9995 | 0.0014 | 0.0010 | 0.0047 |
| 3.9987 | 5.9996 | 0.0013 | 0.0009 | 0.0042 |
| 3.9989 | 5.9996 | 0.0011 | 0.0008 | 0.0038 |
| 3.9990 | 5.9996 | 0.0010 | 0.0007 | 0.0034 |
| 3.9991 | 5.9997 | 0.0009 | 0.0006 | 0.0031 |
| 3.9992 | 5.9997 | 0.0008 | 0.0006 | 0.0028 |
| 3.9993 | 5.9997 | 0.0007 | 0.0005 | 0.0025 |
| 3.9993 | 5.9998 | 0.0007 | 0.0005 | 0.0023 |
| 3.9994 | 5.9998 | 0.0006 | 0.0004 | 0.0020 |
| 3.9995 | 5.9998 | 0.0005 | 0.0004 | 0.0018 |
| 3.9995 | 5.9998 | 0.0005 | 0.0003 | 0.0016 |
| 3.9996 | 5.9998 | 0.0004 | 0.0003 | 0.0015 |
| 3.9996 | 5.9999 | 0.0004 | 0.0003 | 0.0013 |
| 3.9996 | 5.9999 | 0.0004 | 0.0002 | 0.0012 |
| 3.9997 | 5.9999 | 0.0003 | 0.0002 | 0.0011 |
| 3.9997 | 5.9999 | 0.0003 | 0.0002 | 0.0010 |

| | | | | |
|---|---|---|---|---|
| 3.9997 | 5.9999 | 0.0003 | 0.0002 | 0.0009 |
| 3.9998 | 5.9999 | 0.0002 | 0.0002 | 0.0008 |
| 3.9998 | 5.9999 | 0.0002 | 0.0001 | 0.0007 |
| 3.9998 | 5.9999 | 0.0002 | 0.0001 | 0.0006 |
| 3.9998 | 5.9999 | 0.0002 | 0.0001 | 0.0006 |
| 3.9998 | 5.9999 | 0.0002 | 0.0001 | 0.0005 |
| 3.9999 | 6.0000 | 0.0001 | 0.0001 | 0.0005 |

y =

| | | |
|---|---|---|
| 0 | 0 | 0 |
| -0.1333 | -0.0519 | 0.0235 |
| -0.2846 | -0.6023 | -0.1375 |
| -0.5916 | -1.4288 | -0.4214 |
| -0.9196 | -2.0876 | -0.6622 |
| -1.4299 | -2.2344 | -0.5248 |
| -1.5993 | -2.2683 | -0.4714 |
| -1.5888 | -2.2669 | -0.4748 |
| -1.5890 | -2.2668 | -0.4743 |
| -1.5884 | -2.2665 | -0.4741 |
| -1.5878 | -2.2662 | -0.4739 |
| -1.5873 | -2.2660 | -0.4738 |
| -1.5868 | -2.2658 | -0.4737 |
| -1.5864 | -2.2656 | -0.4735 |
| -1.5860 | -2.2654 | -0.4734 |
| -1.5857 | -2.2652 | -0.4733 |
| -1.5854 | -2.2651 | -0.4732 |
| -1.5851 | -2.2649 | -0.4732 |

-1.5848 -2.2648 -0.4731

-1.5846 -2.2647 -0.4730

-1.5844 -2.2646 -0.4730

-1.5842 -2.2645 -0.4729

-1.5841 -2.2645 -0.4729

-1.5839 -2.2644 -0.4728

-1.5838 -2.2643 -0.4728

-1.5837 -2.2643 -0.4728

-1.5835 -2.2642 -0.4727

-1.5834 -2.2642 -0.4727

-1.5834 -2.2641 -0.4727

-1.5833 -2.2641 -0.4727

-1.5832 -2.2641 -0.4727

-1.5831 -2.2640 -0.4726

-1.5831 -2.2640 -0.4726

-1.5830 -2.2640 -0.4726

-1.5830 -2.2640 -0.4726

-1.5830 -2.2639 -0.4726

-1.5829 -2.2639 -0.4726

-1.5829 -2.2639 -0.4726

-1.5828 -2.2639 -0.4726

-1.5828 -2.2639 -0.4725

-1.5828 -2.2639 -0.4725

-1.5828 -2.2639 -0.4725

-1.5828 -2.2638 -0.4725

-1.5827 -2.2638 -0.4725

-1.5827 -2.2638 -0.4725

| | | |
|---|---|---|
| -1.5827 | -2.2638 | -0.4725 |
| -1.5827 | -2.2638 | -0.4725 |
| -1.5827 | -2.2638 | -0.4725 |
| -1.5827 | -2.2638 | -0.4725 |
| -1.5827 | -2.2638 | -0.4725 |
| -1.5827 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |
| -1.5826 | -2.2638 | -0.4725 |

s =

| | | | | |
|---|---|---|---|---|
| 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 0.4093 | 0.1000 | 0.9699 | 0.8885 | 0.8131 |
| 0.1425 | 0.0100 | 0.8960 | 1.2136 | 0.7489 |
| 0.0143 | 0.0168 | 0.8812 | 1.7184 | 0.7110 |
| 0.0220 | 0.0112 | 0.9486 | 2.1166 | 0.6911 |
| 0.0160 | 0.0110 | 1.4328 | 2.2373 | 0.5277 |
| 0.0148 | 0.0097 | 1.5996 | 2.2686 | 0.4717 |
| 0.0132 | 0.0088 | 1.5888 | 2.2669 | 0.4748 |
| 0.0119 | 0.0079 | 1.5890 | 2.2668 | 0.4743 |
| 0.0107 | 0.0071 | 1.5884 | 2.2665 | 0.4741 |

| | | | | |
|---|---|---|---|---|
| 0.0096 | 0.0064 | 1.5878 | 2.2662 | 0.4739 |
| 0.0087 | 0.0057 | 1.5873 | 2.2660 | 0.4738 |
| 0.0078 | 0.0052 | 1.5868 | 2.2658 | 0.4737 |
| 0.0070 | 0.0047 | 1.5864 | 2.2656 | 0.4735 |
| 0.0063 | 0.0042 | 1.5860 | 2.2654 | 0.4734 |
| 0.0057 | 0.0038 | 1.5857 | 2.2652 | 0.4733 |
| 0.0051 | 0.0034 | 1.5854 | 2.2651 | 0.4732 |
| 0.0046 | 0.0031 | 1.5851 | 2.2649 | 0.4732 |
| 0.0041 | 0.0027 | 1.5848 | 2.2648 | 0.4731 |
| 0.0037 | 0.0025 | 1.5846 | 2.2647 | 0.4730 |
| 0.0033 | 0.0022 | 1.5844 | 2.2646 | 0.4730 |
| 0.0030 | 0.0020 | 1.5842 | 2.2645 | 0.4729 |
| 0.0027 | 0.0018 | 1.5841 | 2.2645 | 0.4729 |
| 0.0024 | 0.0016 | 1.5839 | 2.2644 | 0.4728 |
| 0.0022 | 0.0015 | 1.5838 | 2.2643 | 0.4728 |
| 0.0020 | 0.0013 | 1.5837 | 2.2643 | 0.4728 |
| 0.0018 | 0.0012 | 1.5835 | 2.2642 | 0.4727 |
| 0.0016 | 0.0011 | 1.5834 | 2.2642 | 0.4727 |
| 0.0014 | 0.0010 | 1.5834 | 2.2641 | 0.4727 |
| 0.0013 | 0.0009 | 1.5833 | 2.2641 | 0.4727 |
| 0.0012 | 0.0008 | 1.5832 | 2.2641 | 0.4727 |
| 0.0010 | 0.0007 | 1.5831 | 2.2640 | 0.4726 |
| 0.0009 | 0.0006 | 1.5831 | 2.2640 | 0.4726 |
| 0.0008 | 0.0006 | 1.5830 | 2.2640 | 0.4726 |
| 0.0008 | 0.0005 | 1.5830 | 2.2640 | 0.4726 |
| 0.0007 | 0.0005 | 1.5830 | 2.2639 | 0.4726 |
| 0.0006 | 0.0004 | 1.5829 | 2.2639 | 0.4726 |

```
   0.0006   0.0004   1.5829   2.2639   0.4726
   0.0005   0.0003   1.5828   2.2639   0.4726
   0.0005   0.0003   1.5828   2.2639   0.4725
   0.0004   0.0003   1.5828   2.2639   0.4725
   0.0004   0.0002   1.5828   2.2639   0.4725
   0.0003   0.0002   1.5828   2.2638   0.4725
   0.0003   0.0002   1.5827   2.2638   0.4725
   0.0003   0.0002   1.5827   2.2638   0.4725
   0.0002   0.0002   1.5827   2.2638   0.4725
   0.0002   0.0001   1.5827   2.2638   0.4725
   0.0002   0.0001   1.5827   2.2638   0.4725
   0.0002   0.0001   1.5827   2.2638   0.4725
   0.0002   0.0001   1.5827   2.2638   0.4725
   0.0001   0.0001   1.5827   2.2638   0.4725
   0.0001   0.0001   1.5826   2.2638   0.4725
   0.0001   0.0001   1.5826   2.2638   0.4725
   0.0001   0.0001   1.5826   2.2638   0.4725
   0.0001   0.0001   1.5826   2.2638   0.4725
   0.0001   0.0001   1.5826   2.2638   0.4725
   0.0001   0.0001   1.5826   2.2638   0.4725
   0.0001   0.0000   1.5826   2.2638   0.4725
   0.0001   0.0000   1.5826   2.2638   0.4725

z =

  -41.9993

>>
```

## MatLab code for Table 5.2

random problems with random dimentions less that 10

| | | New method | | | Classical method | | |
|---|---|---|---|---|---|---|---|
| idx | dim(m*n) | iteration | Opt | time | iteration | Opt | time |
| 36 | 6 5 | 12 | 1.4119 | 0.0056 | 13 | 1.4164 | 0.0396 |
| 37 | 6 7 | 15 | 0.9627 | 0.0065 | 16 | 0.9627 | 0.0046 |
| 38 | 6 8 | 17 | 0.7974 | 0.0072 | 24 | 0.7974 | 0.0073 |
| 39 | 4 8 | 17 | 0.7864 | 0.0336 | 21 | 0.7864 | 0.0061 |
| 40 | 8 3 | 15 | 0.4944 | 0.0070 | 38 | 0.3653 | 0.0140 |
| 41 | 4 9 | 72 | 1.0654 | 0.0757 | 27 | 1.0668 | 0.0086 |
| 42 | 5 8 | 18 | 0.4801 | 0.0076 | 24 | 0.4801 | 0.0072 |
| 43 | 8 9 | 14 | 1.3682 | 0.0070 | 19 | 1.3682 | 0.0064 |
| 44 | 2 2 | 13 | 0.7773 | 0.0042 | 14 | 0.7773 | 0.0033 |
| 45 | 6 5 | 13 | 0.2225 | 0.0059 | 14 | 0.2229 | 0.0049 |
| 46 | 8 6 | 14 | 1.6662 | 0.0068 | 20 | 1.6114 | 0.0268 |
| 47 | 3 6 | 17 | 0.3840 | 0.0069 | 30 | NaN | 0.0101 |
| 48 | 3 7 | 64 | 0.5853 | 0.0260 | 7 | 0.5886 | 0.0018 |
| 49 | 3 3 | 12 | 0.8750 | 0.0044 | 14 | 0.8750 | 0.0038 |
| 50 | 2 7 | 64 | 0.6310 | 0.0458 | 7 | 0.6367 | 0.0017 |
| 51 | 7 7 | 12 | 1.7618 | 0.0047 | 13 | 1.7618 | 0.0036 |
| 52 | 2 7 | 64 | 0.1590 | 0.0244 | 7 | 0.1647 | 0.0018 |

| | New method | | Classical method | |
|---|---|---|---|---|
| | dim(m*n) | iteration time | iteration | time |
| average | 4.9 5.4 | 40.7 0.0448 | 24.6 | 0.0362 |

| | | | New method | | | Classical method | | |
|---|---|---|---|---|---|---|---|---|
| idx | dim(m*n) | | iteration | Opt | time | iteration | Opt | time |
| 1 | 9 | 1 | 45 | NaN | 0.0681 | 30 | NaN | 0.0326 |
| 2 | 9 | 1 | 45 | NaN | 0.0674 | 45 | NaN | 0.7326 |
| 3 | 7 | 7 | 12 | 3.1727 | 0.0053 | 14 | 3.1727 | 0.0042 |
| 4 | 7 | 7 | 12 | 2.0364 | 0.0050 | 13 | 2.0364 | 0.0037 |
| 5 | 5 | 4 | 14 | 0.7954 | 0.0675 | 16 | 0.7257 | 0.0059 |
| 6 | 6 | 3 | 49 | 0.4286 | 0.0243 | 92 | 0.4560 | 0.1107 |
| 7 | 6 | 1 | 45 | NaN | 0.1613 | 45 | NaN | 0.0612 |
| 8 | 3 | 9 | 66 | 0.2055 | 0.0284 | 9 | 0.1994 | 0.0024 |
| 9 | 3 | 1 | 45 | NaN | 0.1071 | 36 | NaN | 0.0133 |
| 10 | 7 | 8 | 13 | 2.3593 | 0.0057 | 14 | 2.3593 | 0.0042 |
| 11 | 2 | 7 | 64 | 0.3550 | 0.0246 | 8 | 0.3596 | 0.0019 |
| 12 | 6 | 7 | 14 | 1.7947 | 0.0055 | 15 | 1.7947 | 0.0043 |
| 13 | 2 | 6 | 62 | NaN | 0.0637 | 27 | NaN | 0.0096 |
| 14 | 3 | 2 | 201 | 0.4458 | 0.3814 | 71 | 0.1206 | 0.1475 |
| 15 | 7 | 8 | 14 | 2.3068 | 0.0059 | 16 | 2.3068 | 0.0258 |
| 16 | 7 | 5 | 28 | 0.4972 | 0.0138 | 19 | 0.5465 | 0.0067 |
| 17 | 4 | 9 | 17 | 1.2697 | 0.0076 | 18 | 1.2697 | 0.0053 |
| 18 | 2 | 7 | 64 | NaN | 0.0814 | 17 | NaN | 0.0044 |
| 19 | 1 | 4 | 58 | 0.0069 | 0.0650 | 6 | 0.0106 | 0.0015 |
| 20 | 5 | 5 | 12 | 2.0070 | 0.0045 | 13 | 2.0070 | 0.0036 |
| 21 | 1 | 5 | 60 | 0.0321 | 0.0245 | 6 | 0.0387 | 0.0026 |
| 22 | 1 | 2 | 52 | 0.2947 | 0.0739 | 5 | 0.2976 | 0.0012 |
| 23 | 9 | 6 | 23 | 0.7775 | 0.0118 | 52 | 0.5957 | 0.0617 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 24 | 5 | 3 | 18 | 1.7080 | 0.0088 | 33 | 1.2630 | 0.0476 |
| 25 | 4 | 3 | 31 | 0.3020 | 0.0528 | 28 | 0.3146 | 0.0426 |
| 26 | 5 | 8 | 16 | 2.4185 | 0.0069 | 16 | 2.4185 | 0.0046 |
| 27 | 4 | 7 | 16 | 0.2408 | 0.0064 | 20 | 0.2408 | 0.0059 |
| 28 | 9 | 8 | 17 | 1.8185 | 0.0091 | 24 | 1.2417 | 0.0096 |
| 29 | 1 | 2 | 52 | 0.2448 | 0.0208 | 5 | 0.2478 | 0.0012 |
| 30 | 3 | 4 | 14 | 1.0822 | 0.0051 | 17 | 1.0822 | 0.0046 |
| 31 | 4 | 5 | 14 | 1.4672 | 0.0054 | 16 | 1.4672 | 0.0044 |
| 32 | 4 | 2 | 201 | 0.0760 | 0.3930 | 59 | 0.0470 | 0.1529 |
| 33 | 5 | 8 | 25 | 0.5901 | 0.0115 | 28 | 0.5901 | 0.0087 |
| 34 | 1 | 9 | 66 | 0.0042 | 0.0293 | 7 | 0.0082 | 0.0018 |
| 35 | 3 | 7 | 15 | 0.8637 | 0.0060 | 201 | 0.2481 | 0.3810 |
| 36 | 6 | 5 | 12 | 1.4119 | 0.0056 | 13 | 1.4164 | 0.0396 |
| 37 | 6 | 7 | 15 | 0.9627 | 0.0065 | 16 | 0.9627 | 0.0046 |
| 38 | 6 | 8 | 17 | 0.7974 | 0.0072 | 24 | 0.7974 | 0.0073 |
| 39 | 4 | 8 | 17 | 0.7864 | 0.0336 | 21 | 0.7864 | 0.0061 |
| 40 | 8 | 3 | 15 | 0.4944 | 0.0070 | 38 | 0.3653 | 0.0140 |
| 41 | 4 | 9 | 72 | 1.0654 | 0.0757 | 27 | 1.0668 | 0.0086 |
| 42 | 5 | 8 | 18 | 0.4801 | 0.0076 | 24 | 0.4801 | 0.0072 |
| 43 | 8 | 9 | 14 | 1.3682 | 0.0070 | 19 | 1.3682 | 0.0064 |
| 44 | 2 | 2 | 13 | 0.7773 | 0.0042 | 14 | 0.7773 | 0.0033 |
| 45 | 6 | 5 | 13 | 0.2225 | 0.0059 | 14 | 0.2229 | 0.0049 |
| 46 | 8 | 6 | 14 | 1.6662 | 0.0068 | 20 | 1.6114 | 0.0268 |
| 47 | 3 | 6 | 17 | 0.3840 | 0.0069 | 30 | NaN | 0.0101 |
| 48 | 3 | 7 | 64 | 0.5853 | 0.0260 | 7 | 0.5886 | 0.0018 |
| 49 | 3 | 3 | 12 | 0.8750 | 0.0044 | 14 | 0.8750 | 0.0038 |
| 50 | 2 | 7 | 64 | 0.6310 | 0.0458 | 7 | 0.6367 | 0.0017 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 51 | 7 | 7 | 12 | 1.7618 | 0.0047 | 13 | 1.7618 | 0.0036 |
| 52 | 2 | 7 | 64 | 0.1590 | 0.0244 | 7 | 0.1647 | 0.0018 |
| 53 | 4 | 5 | 15 | 3.6371 | 0.0059 | 16 | 3.6371 | 0.0455 |
| 54 | 9 | 3 | 42 | 0.4919 | 0.0216 | 41 | 0.7683 | 0.0158 |
| 55 | 7 | 1 | 45 | NaN | 0.0833 | 28 | NaN | 0.0224 |
| 56 | 5 | 4 | 20 | 1.4544 | 0.0331 | 14 | 1.6696 | 0.0063 |
| 57 | 2 | 1 | 45 | NaN | 0.0891 | 45 | NaN | 0.0296 |
| 58 | 7 | 5 | 41 | 1.6545 | 0.0388 | 33 | 0.8089 | 0.0497 |
| 59 | 1 | 7 | 64 | 0.1189 | 0.0272 | 8 | 0.1203 | 0.0020 |
| 60 | 9 | 9 | 12 | 3.7354 | 0.0071 | 15 | 3.7354 | 0.0049 |
| 61 | 6 | 8 | 16 | 1.3473 | 0.0067 | 17 | 1.3473 | 0.0051 |
| 62 | 2 | 8 | 65 | 0.1011 | 0.0256 | 7 | 0.1045 | 0.0018 |
| 63 | 2 | 6 | 16 | 0.5000 | 0.0055 | 19 | 0.5000 | 0.0048 |
| 64 | 2 | 6 | 62 | 0.4073 | 0.0233 | 6 | 0.4123 | 0.0014 |
| 65 | 4 | 6 | 15 | 1.0340 | 0.0374 | 19 | 1.0340 | 0.0054 |
| 66 | 3 | 9 | 66 | 0.9751 | 0.0292 | 8 | 0.9772 | 0.0021 |
| 67 | 5 | 5 | 13 | 0.9247 | 0.0050 | 14 | 0.9247 | 0.0039 |
| 68 | 6 | 6 | 13 | 1.6887 | 0.0051 | 14 | 1.6887 | 0.0452 |
| 69 | 2 | 2 | 13 | 0.1492 | 0.0045 | 15 | 0.1492 | 0.0036 |
| 70 | 3 | 2 | 171 | NaN | 0.1243 | 54 | 2.9584 | 0.0724 |
| 71 | 7 | 2 | 201 | 0.2807 | 0.2557 | 66 | 0.0886 | 0.0771 |
| 72 | 5 | 2 | 201 | 0.6778 | 0.4181 | 75 | 1.0557 | 0.1723 |
| 73 | 2 | 9 | 66 | 0.0872 | 0.0264 | 7 | 0.0909 | 0.0018 |
| 74 | 5 | 7 | 14 | 1.4273 | 0.0056 | 15 | 1.4273 | 0.0040 |
| 75 | 9 | 9 | 12 | 4.2331 | 0.0055 | 14 | 4.2331 | 0.0048 |
| 76 | 4 | 4 | 13 | 0.7073 | 0.0047 | 14 | 0.7073 | 0.0040 |
| 77 | 2 | 9 | 66 | 0.4177 | 0.0272 | 7 | 0.4248 | 0.0019 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 78 | 7 | 9 | 15 | 2.3273 | 0.0066 | 16 | 2.3273 | 0.0050 |
| 79 | 7 | 5 | 15 | 0.6883 | 0.0382 | 13 | 0.8011 | 0.0524 |
| 80 | 3 | 5 | 17 | 0.7023 | 0.0066 | 19 | 0.7023 | 0.0053 |
| 81 | 8 | 5 | 17 | 1.6375 | 0.0082 | 87 | 0.5440 | 0.1686 |
| 82 | 5 | 6 | 17 | 0.5266 | 0.0070 | 22 | 0.5266 | 0.0065 |
| 83 | 6 | 9 | 16 | 1.1541 | 0.0073 | 19 | 1.1541 | 0.0059 |
| 84 | 1 | 2 | 52 | 0.9460 | 0.0199 | 5 | 0.9489 | 0.0012 |
| 85 | 3 | 9 | 20 | 1.4204 | 0.0087 | 39 | NaN | 0.0153 |
| 86 | 9 | 4 | 29 | 0.1871 | 0.0378 | 32 | 0.6971 | 0.0615 |
| 87 | 9 | 5 | 37 | 0.9494 | 0.0764 | 32 | 1.3586 | 0.0558 |
| 88 | 4 | 9 | 18 | 0.5192 | 0.0078 | 23 | 0.5192 | 0.0411 |
| 89 | 9 | 5 | 41 | 0.5489 | 0.0475 | 18 | 1.5730 | 0.0528 |
| 90 | 6 | 4 | 58 | NaN | 0.1482 | 58 | NaN | 0.1727 |
| 91 | 3 | 5 | 15 | 0.6959 | 0.0058 | 20 | 0.6959 | 0.0055 |
| 92 | 7 | 5 | 14 | 2.9338 | 0.0065 | 14 | 2.3208 | 0.0053 |
| 93 | 3 | 5 | 60 | 0.4861 | 0.0749 | 7 | 0.4866 | 0.0020 |
| 94 | 3 | 2 | 196 | 4.5525 | 0.3504 | 146 | 1.8557 | 0.3399 |
| 95 | 2 | 8 | 65 | 0.6134 | 0.0258 | 7 | 0.6152 | 0.0018 |
| 96 | 6 | 8 | 15 | 1.6926 | 0.0062 | 19 | 1.6926 | 0.0058 |
| 97 | 3 | 6 | 15 | 0.8336 | 0.0057 | 17 | 0.8336 | 0.0047 |
| 98 | 4 | 3 | 56 | NaN | 0.0830 | 14 | 1.4102 | 0.0045 |
| 99 | 7 | 2 | 201 | 0.0365 | 0.4461 | 83 | 1.0238 | 0.2068 |
| 100 | 6 | 1 | 45 | NaN | 0.1986 | 45 | NaN | 0.0328 |
| 101 | 1 | 4 | 58 | 0.6234 | 0.0448 | 6 | 0.6307 | 0.0014 |
| 102 | 7 | 5 | 14 | 1.1884 | 0.0070 | 38 | 0.4592 | 0.1011 |
| 103 | 2 | 5 | 60 | 0.5702 | 0.0634 | 6 | 0.5733 | 0.0014 |
| 104 | 8 | 5 | 38 | 1.2510 | 0.0198 | 19 | 1.3811 | 0.0390 |

105 5 7 16 0.7932 0.0068 19 0.7932 0.0267

106 2 3 56 0.6463 0.0198 5 0.6492 0.0012

107 7 9 18 1.2119 0.0081 24 1.2119 0.0078

108 4 2 116 0.4112 0.1974 116 0.3480 0.2755

109 2 4 58 0.3450 0.0323 6 0.3478 0.0014

110 8 6 47 0.8233 0.0480 25 0.8203 0.0931

111 1 5 60 0.0042 0.0245 7 0.0052 0.0017

112 5 7 19 2.2536 0.0077 25 2.2536 0.0074

113 7 7 13 1.4697 0.0053 14 1.4697 0.0044

114 9 2 120 0.4624 0.2295 201 0.8555 0.4208

115 9 8 12 1.9521 0.0063 13 1.9450 0.0872

116 1 1 45 0.4377 0.0147 5 0.4378 0.0011

117 6 3 40 1.3974 0.0493 60 0.0616 0.1541

118 2 5 60 0.0523 0.0223 6 0.0577 0.0014

119 4 5 16 1.3757 0.0060 17 1.3757 0.0046

120 9 8 13 2.8652 0.0067 15 3.0675 0.0062

121 8 8 11 5.0975 0.0048 15 5.0975 0.0047

122 2 2 13 0.3116 0.0042 39 NaN 0.0116

123 9 5 18 0.4133 0.0092 21 0.5460 0.0381

124 5 4 17 1.3629 0.0079 18 1.3382 0.0401

125 9 5 17 1.4464 0.0085 16 1.9442 0.0707

126 7 6 15 2.1528 0.0073 13 1.5332 0.0388

127 1 2 52 0.4030 0.0202 5 0.4060 0.0012

128 9 8 17 1.5577 0.0091 16 1.3553 0.0158

129 7 1 45 NaN 0.1823 36 NaN 0.0295

130 8 9 17 2.5650 0.0080 21 2.5650 0.0070

131 3 4 15 0.3709 0.0055 17 0.3709 0.0062

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 132 | 8 | 7 | 13 | 1.5864 | 0.0063 | 16 | 2.2049 | 0.0061 |
| 133 | 9 | 9 | 12 | 1.4834 | 0.0053 | 14 | 1.4834 | 0.0045 |
| 134 | 8 | 4 | 88 | 0.1456 | 0.1307 | 58 | 0.4504 | 0.1475 |
| 135 | 6 | 2 | 27 | 0.0678 | 0.0600 | 16 | NaN | 0.0094 |
| 136 | 2 | 9 | 66 | 0.4524 | 0.0270 | 8 | 0.4538 | 0.0019 |
| 137 | 1 | 7 | 64 | 0.1021 | 0.0279 | 7 | 0.1043 | 0.0017 |
| 138 | 1 | 6 | 62 | 0.0028 | 0.0263 | 7 | 0.0051 | 0.0017 |
| 139 | 7 | 7 | 12 | 3.4885 | 0.0052 | 13 | 3.4885 | 0.0037 |
| 140 | 3 | 4 | 15 | 0.2784 | 0.0056 | 17 | 0.2784 | 0.0048 |
| 141 | 7 | 9 | 14 | 3.1083 | 0.0062 | 15 | 3.1083 | 0.0047 |
| 142 | 1 | 9 | 66 | 0.0150 | 0.0301 | 7 | 0.0228 | 0.0018 |
| 143 | 8 | 8 | 12 | 3.1648 | 0.0050 | 13 | 3.1648 | 0.0041 |
| 144 | 3 | 8 | 16 | 0.6641 | 0.0651 | 17 | 0.6641 | 0.0048 |
| 145 | 6 | 2 | 127 | NaN | 0.1905 | 29 | 0.0231 | 0.0666 |
| 146 | 7 | 7 | 11 | 3.2350 | 0.0046 | 13 | 3.2350 | 0.0043 |
| 147 | 2 | 4 | 58 | 0.7658 | 0.0200 | 6 | 0.7670 | 0.0013 |
| 148 | 8 | 8 | 12 | 2.1261 | 0.0050 | 13 | 2.1261 | 0.0038 |
| 149 | 2 | 5 | 60 | 0.2748 | 0.0218 | 6 | 0.2799 | 0.0526 |
| 150 | 1 | 3 | 56 | 0.1017 | 0.0223 | 6 | 0.1030 | 0.0014 |
| 151 | 8 | 9 | 15 | 2.2372 | 0.0071 | 17 | 2.2372 | 0.0058 |
| 152 | 7 | 3 | 195 | 0.4054 | 0.3624 | 48 | 0.3538 | 0.1150 |
| 153 | 3 | 9 | 17 | 0.7720 | 0.0073 | 20 | 0.7720 | 0.0058 |
| 154 | 1 | 5 | 60 | 0.0085 | 0.0624 | 7 | 0.0096 | 0.0017 |
| 155 | 5 | 7 | 16 | 0.6305 | 0.0068 | 16 | 0.6305 | 0.0046 |
| 156 | 7 | 8 | 16 | 1.9168 | 0.0070 | 25 | 1.9168 | 0.0083 |
| 157 | 4 | 5 | 14 | 0.7501 | 0.0052 | 15 | 0.7501 | 0.0045 |
| 158 | 7 | 1 | 45 | NaN | 0.1248 | 16 | NaN | 0.0114 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 159 | 3 | 3 | 11 | 0.9672 | 0.0264 | 12 | 0.9672 | 0.0031 |
| 160 | 1 | 1 | 45 | 0.4010 | 0.0133 | 4 | 0.4008 | 0.0008 |
| 161 | 4 | 8 | 18 | 0.6044 | 0.0076 | 19 | 0.6044 | 0.0053 |
| 162 | 6 | 5 | 14 | 1.2959 | 0.0066 | 16 | 1.2533 | 0.0058 |
| 163 | 9 | 6 | 15 | 1.5190 | 0.0076 | 23 | 1.2027 | 0.0529 |
| 164 | 9 | 9 | 12 | 2.4810 | 0.0054 | 13 | 2.4810 | 0.0041 |
| 165 | 7 | 2 | 54 | 3.0844 | 0.0905 | 26 | 1.1829 | 0.0092 |
| 166 | 9 | 5 | 11 | 0.6774 | 0.0054 | 20 | 0.8400 | 0.0390 |
| 167 | 3 | 9 | 66 | 0.1178 | 0.0305 | 7 | 0.1240 | 0.0019 |
| 168 | 2 | 3 | 56 | 1.2827 | 0.0201 | 5 | 1.2850 | 0.0012 |
| 169 | 8 | 5 | 14 | 3.1274 | 0.0066 | 21 | 2.3811 | 0.0387 |
| 170 | 2 | 4 | 58 | 0.2706 | 0.0206 | 6 | 0.2715 | 0.0014 |
| 171 | 4 | 3 | 79 | 1.8914 | 0.1462 | 159 | 0.6958 | 0.3458 |
| 172 | 1 | 5 | 60 | 0.0769 | 0.0245 | 6 | 0.0821 | 0.0015 |
| 173 | 1 | 3 | 56 | 0.1561 | 0.0220 | 5 | 0.1636 | 0.0012 |
| 174 | 7 | 4 | 55 | 0.3958 | 0.0958 | 31 | 0.8488 | 0.0474 |
| 175 | 7 | 6 | 22 | 0.4917 | 0.0108 | 23 | 0.5037 | 0.0401 |
| 176 | 1 | 4 | 58 | 0.0114 | 0.0541 | 6 | 0.0139 | 0.0014 |
| 177 | 6 | 7 | 15 | 1.2088 | 0.0063 | 19 | 1.2088 | 0.0057 |
| 178 | 8 | 7 | 22 | 2.3937 | 0.0109 | 31 | 3.0084 | 0.1104 |
| 179 | 9 | 8 | 12 | 2.3249 | 0.0063 | 23 | 2.0587 | 0.0430 |
| 180 | 1 | 8 | 65 | 0.0225 | 0.0277 | 8 | 0.0236 | 0.0329 |
| 181 | 8 | 3 | 21 | 0.2064 | 0.0111 | 52 | 0.1950 | 0.0961 |
| 182 | 8 | 1 | 45 | NaN | 0.1603 | 45 | NaN | 0.0903 |
| 183 | 3 | 1 | 45 | NaN | 0.0892 | 45 | NaN | 0.0170 |
| 184 | 4 | 9 | 16 | 0.7594 | 0.0070 | 25 | 0.7594 | 0.0080 |
| 185 | 7 | 6 | 12 | 1.7165 | 0.0058 | 16 | 1.2626 | 0.0795 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 186 | 9 | 9 | 12 | 4.2814 | 0.0057 | 13 | 4.2814 | 0.0042 |
| 187 | 7 | 9 | 15 | 1.0139 | 0.0067 | 17 | 1.0139 | 0.0051 |
| 188 | 2 | 3 | 56 | 0.0452 | 0.0192 | 5 | 0.0469 | 0.0012 |
| 189 | 8 | 4 | 48 | 0.4489 | 0.0365 | 14 | 1.0610 | 0.0049 |
| 190 | 1 | 2 | 52 | 0.0018 | 0.0198 | 5 | 0.0047 | 0.0012 |
| 191 | 4 | 9 | 18 | 1.5071 | 0.0082 | 23 | 1.5071 | 0.0282 |
| 192 | 4 | 3 | 15 | 0.5991 | 0.0068 | 16 | 0.6997 | 0.0056 |
| 193 | 6 | 3 | 123 | 0.0110 | 0.2378 | 67 | 0.0438 | 0.1039 |
| 194 | 1 | 6 | 62 | 0.1704 | 0.0566 | 7 | 0.1724 | 0.0017 |
| 195 | 3 | 2 | 53 | NaN | 0.0634 | 17 | NaN | 0.0057 |
| 196 | 2 | 4 | 58 | 0.0721 | 0.0200 | 6 | 0.0768 | 0.0015 |
| 197 | 3 | 8 | 16 | 2.0542 | 0.0065 | 20 | 2.0542 | 0.0056 |
| 198 | 5 | 1 | 45 | NaN | 0.1366 | 45 | NaN | 0.0839 |
| 199 | 4 | 5 | 13 | 1.3755 | 0.0054 | 15 | 1.3755 | 0.0041 |
| 200 | 3 | 3 | 14 | 1.8529 | 0.0051 | 15 | 1.8529 | 0.0039 |

---

| | New method | | Classical method | |
|---|---|---|---|---|
| dim(m*n) | iteration | time | iteration | time |
| average 4.9 5.4 | 40.7 | 0.0448 | 24.6 | 0.0362 |

---

## MatLab code forTable 5.3

solve the random system with dimention 200*300

```
m=200;n=300;
A=rand(m,n);
b=rand(m,1);
c=rand(n,1);
epsilon=0.01;
```

in new mehtod, p=0.8,q=1.2

| | New method | | | Classical method | | |
|---|---|---|---|---|---|---|
| idx | iteration | Opt | time | iteration | Opt | time |
| 34 | 31 | 13.3783 | 2.1870 | 34 | 13.3783 | 2.4084 |
| 35 | 33 | 13.5477 | 2.3660 | 38 | 13.5477 | 2.7975 |
| 36 | 35 | 12.2685 | 2.4543 | 47 | 12.2685 | 3.6504 |
| 37 | 34 | 12.1387 | 2.4273 | 42 | 12.1387 | 3.1664 |
| 38 | 31 | 14.3202 | 2.1804 | 42 | 14.3202 | 3.1886 |
| 39 | 33 | 11.7495 | 2.4815 | 30 | 11.7495 | 2.0452 |
| 40 | 30 | 15.0938 | 2.1312 | 39 | 15.0938 | 2.8654 |
| 41 | 40 | 15.4534 | 3.0469 | 44 | 15.4534 | 3.3165 |
| 42 | 38 | 12.4233 | 2.8703 | 44 | 12.4233 | 3.2338 |
| 43 | 66 | 12.7004 | 5.6847 | 40 | 12.7022 | 2.8478 |
| 44 | 39 | 12.4386 | 2.9631 | 40 | 12.4386 | 2.8733 |
| 45 | 33 | 13.5479 | 2.3851 | 42 | 13.5479 | 3.0937 |
| 46 | 36 | 13.7716 | 2.6777 | 43 | 13.7716 | 3.1809 |
| 47 | 33 | 12.7877 | 2.5047 | 35 | 12.7877 | 2.5006 |
| 48 | 36 | 16.7568 | 2.7086 | 36 | 16.7568 | 2.5545 |
| 49 | 38 | 15.3569 | 2.9702 | 34 | 15.3569 | 2.4201 |

average

| New method | | Classical method | |
|---|---|---|---|
| iteration | time | iteration Opt | time |
| 35.8100 | 2.6626 | 40.0000 | 2.8812 |

| | New method | | | Classical method | | |
|---|---|---|---|---|---|---|
| idx | iteration | Opt | time | iteration | Opt | time |
| 1 | 33 | 14.1325 | 2.2903 | 36 | 14.1325 | 2.4028 |
| 2 | 34 | 11.4602 | 2.3497 | 40 | 11.4602 | 2.6396 |
| 3 | 34 | 14.8578 | 2.3979 | 43 | 14.8578 | 3.4298 |
| 4 | 32 | 14.0615 | 2.3182 | 42 | 14.0615 | 2.9494 |
| 5 | 37 | 13.8361 | 2.6571 | 40 | 13.8361 | 2.7531 |
| 6 | 27 | 12.5830 | 1.8774 | 31 | 12.5830 | 2.0912 |
| 7 | 35 | 15.0891 | 2.4443 | 46 | 15.0891 | 3.1579 |
| 8 | 42 | 15.4758 | 3.0902 | 51 | 15.4758 | 3.5747 |
| 9 | 33 | 13.9757 | 2.3580 | 36 | 13.9755 | 2.3991 |
| 10 | 33 | 14.4968 | 2.3421 | 34 | 14.4968 | 2.3027 |
| 11 | 31 | 11.4472 | 2.1374 | 65 | 11.4295 | 4.6683 |
| 12 | 29 | 13.3044 | 1.9933 | 36 | 13.3044 | 2.4292 |
| 13 | 34 | 13.6009 | 2.3862 | 50 | 13.6009 | 3.4991 |
| 14 | 34 | 11.4376 | 2.4010 | 41 | 11.4376 | 2.8593 |
| 15 | 31 | 14.8721 | 2.1535 | 35 | 14.8721 | 2.3473 |
| 16 | 38 | 14.0051 | 2.7457 | 35 | 14.0051 | 2.3498 |
| 17 | 29 | 13.8349 | 2.0148 | 35 | 13.8349 | 2.3490 |
| 18 | 35 | 12.8241 | 2.4992 | 42 | 12.8241 | 2.9333 |
| 19 | 33 | 13.0498 | 2.3489 | 42 | 13.0499 | 2.9874 |
| 20 | 33 | 14.5358 | 2.3046 | 35 | 14.5358 | 2.3323 |
| 21 | 33 | 13.7607 | 2.3485 | 38 | 13.7607 | 2.6101 |
| 22 | 31 | 13.3368 | 2.1436 | 36 | 13.3368 | 2.4539 |
| 23 | 31 | 12.6427 | 2.1378 | 44 | 12.6427 | 3.0703 |
| 24 | 33 | 13.1727 | 2.3204 | 47 | 13.1727 | 3.3550 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 25 | 38 | 12.4437 | 2.7154 | 45 | 12.4437 | 3.1244 |
| 26 | 30 | 16.4101 | 2.0944 | 39 | 16.4101 | 2.6710 |
| 27 | 43 | 14.5286 | 3.1413 | 39 | 14.5286 | 2.6813 |
| 28 | 32 | 14.2258 | 2.2594 | 31 | 14.2258 | 2.1112 |
| 29 | 34 | 11.9038 | 2.4134 | 51 | 11.9038 | 3.6221 |
| 30 | 29 | 12.9792 | 1.9981 | 33 | 12.9792 | 2.1890 |
| 31 | 30 | 13.5566 | 2.0404 | 39 | 13.5566 | 2.6850 |
| 32 | 30 | 13.9748 | 2.1090 | 31 | 13.9748 | 2.0360 |
| 33 | 38 | 13.9658 | 2.6906 | 41 | 13.9658 | 2.7484 |
| 34 | 31 | 13.3783 | 2.1870 | 34 | 13.3783 | 2.4084 |
| 35 | 33 | 13.5477 | 2.3660 | 38 | 13.5477 | 2.7975 |
| 36 | 35 | 12.2685 | 2.4543 | 47 | 12.2685 | 3.6504 |
| 37 | 34 | 12.1387 | 2.4273 | 42 | 12.1387 | 3.1664 |
| 38 | 31 | 14.3202 | 2.1804 | 42 | 14.3202 | 3.1886 |
| 39 | 33 | 11.7495 | 2.4815 | 30 | 11.7495 | 2.0452 |
| 40 | 30 | 15.0938 | 2.1312 | 39 | 15.0938 | 2.8654 |
| 41 | 40 | 15.4534 | 3.0469 | 44 | 15.4534 | 3.3165 |
| 42 | 38 | 12.4233 | 2.8703 | 44 | 12.4233 | 3.2338 |
| 43 | 66 | 12.7004 | 5.6847 | 40 | 12.7022 | 2.8478 |
| 44 | 39 | 12.4386 | 2.9631 | 40 | 12.4386 | 2.8733 |
| 45 | 33 | 13.5479 | 2.3851 | 42 | 13.5479 | 3.0937 |
| 46 | 36 | 13.7716 | 2.6777 | 43 | 13.7716 | 3.1809 |
| 47 | 33 | 12.7877 | 2.5047 | 35 | 12.7877 | 2.5006 |
| 48 | 36 | 16.7568 | 2.7086 | 36 | 16.7568 | 2.5545 |
| 49 | 38 | 15.3569 | 2.9702 | 34 | 15.3569 | 2.4201 |
| 50 | 35 | 10.1950 | 2.5621 | 41 | 10.1950 | 2.9928 |
| 51 | 39 | 14.4130 | 3.0114 | 47 | 14.4130 | 3.6163 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 52 | 35 | 14.5251 | 2.6399 | 40 | 14.5251 | 2.9441 |
| 53 | 72 | 14.9484 | 6.2311 | 45 | 14.9489 | 3.2947 |
| 54 | 35 | 16.3867 | 2.5885 | 40 | 16.3867 | 2.8813 |
| 55 | 38 | 12.7772 | 2.9128 | 42 | 12.7772 | 3.1070 |
| 56 | 34 | 13.0523 | 2.5302 | 38 | 13.0523 | 2.8084 |
| 57 | 35 | 13.0530 | 2.5921 | 43 | 13.0530 | 3.2378 |
| 58 | 34 | 13.0528 | 2.5316 | 35 | 13.0528 | 2.4582 |
| 59 | 31 | 15.8548 | 2.2426 | 35 | 15.8548 | 2.5212 |
| 60 | 29 | 14.2370 | 2.0127 | 39 | 14.2370 | 2.8138 |
| 61 | 43 | 13.5499 | 3.3413 | 38 | 13.5499 | 2.7139 |
| 62 | 35 | 12.8535 | 2.5790 | 35 | 12.8535 | 2.4133 |
| 63 | 33 | 12.9314 | 2.4403 | 37 | 12.9314 | 2.6749 |
| 64 | 95 | 13.7239 | 9.0412 | 44 | 13.7796 | 3.2587 |
| 65 | 31 | 15.8651 | 2.2830 | 34 | 15.8651 | 2.3814 |
| 66 | 37 | 13.8844 | 2.7863 | 38 | 13.8844 | 2.6984 |
| 67 | 38 | 14.4416 | 2.9258 | 46 | 14.4415 | 3.5383 |
| 68 | 35 | 13.3879 | 2.5361 | 44 | 13.3879 | 3.1893 |
| 69 | 37 | 14.7301 | 2.6624 | 47 | 14.7301 | 3.5063 |
| 70 | 34 | 14.0756 | 2.5296 | 40 | 14.0755 | 2.8794 |
| 71 | 37 | 13.8434 | 2.7015 | 41 | 13.8434 | 2.9964 |
| 72 | 32 | 14.9581 | 2.3169 | 32 | 14.9581 | 2.1915 |
| 73 | 31 | 13.0970 | 2.2517 | 40 | 13.0970 | 2.9671 |
| 74 | 32 | 12.1322 | 2.3486 | 42 | 12.1322 | 3.1600 |
| 75 | 33 | 14.5811 | 2.4676 | 36 | 14.5811 | 2.5803 |
| 76 | 36 | 15.1297 | 2.7490 | 40 | 15.1297 | 2.9157 |
| 77 | 28 | 16.3501 | 2.0110 | 37 | 16.3501 | 2.6951 |
| 78 | 37 | 11.7357 | 2.7682 | 38 | 11.7357 | 2.6857 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 79 | 47 | 13.4661 | 3.8289 | 38 | 13.4661 | 2.7929 |
| 80 | 30 | 14.1476 | 2.1423 | 38 | 14.1476 | 2.7840 |
| 81 | 33 | 14.0924 | 2.4596 | 38 | 14.0924 | 2.7350 |
| 82 | 37 | 14.7447 | 2.7495 | 39 | 14.7447 | 2.7690 |
| 83 | 40 | 13.4680 | 2.9901 | 46 | 13.4680 | 3.4051 |
| 84 | 37 | 11.5136 | 2.7620 | 42 | 11.5136 | 3.0184 |
| 85 | 36 | 14.5218 | 2.7568 | 36 | 14.5218 | 2.6255 |
| 86 | 54 | 13.8599 | 3.9193 | 36 | 13.8574 | 2.6522 |
| 87 | 32 | 12.6705 | 2.4086 | 41 | 12.6705 | 3.1666 |
| 88 | 34 | 12.5076 | 2.6738 | 38 | 12.5076 | 2.8091 |
| 89 | 40 | 14.6875 | 3.2020 | 34 | 14.6875 | 2.5030 |
| 90 | 30 | 11.3641 | 2.2120 | 33 | 11.3641 | 2.3333 |
| 91 | 34 | 14.8216 | 2.5382 | 40 | 14.8216 | 2.9601 |
| 92 | 34 | 16.0217 | 2.5202 | 48 | 16.0217 | 3.8445 |
| 93 | 33 | 14.0469 | 2.4813 | 41 | 14.0469 | 3.1175 |
| 94 | 40 | 15.8152 | 3.0568 | 41 | 15.8152 | 2.9229 |
| 95 | 30 | 14.3876 | 2.2207 | 34 | 14.3876 | 2.4590 |
| 96 | 30 | 15.8316 | 2.2985 | 40 | 15.8316 | 3.0683 |
| 97 | 33 | 14.2434 | 2.4493 | 38 | 14.2434 | 2.8031 |
| 98 | 37 | 13.7590 | 2.8397 | 46 | 13.7590 | 3.5587 |
| 99 | 39 | 15.5669 | 2.8991 | 52 | 15.5669 | 3.9323 |
| 100 | 35 | 15.0051 | 2.6977 | 48 | 15.0051 | 3.7811 |

---

average

| New method | | Classical method | |
|---|---|---|---|
| iteration | time | iteration Opt | time |
| 35.8100 | 2.6626 | 40.0000 | 2.8812 |

## MatLab code for Table 5.4

solve the random system with dimention 400*700

```
m=400;n=700;
A=rand(m,n);
b=rand(m,1);
c=rand(n,1);
epsilon=0.01;
```

in Dr. Lesaja's mehtod, p=0.8,q=1.2

| | New method | | Classical method | |
|---|---|---|---|---|
| idx | iteration | Opt time | iteration | Opt time |
| 1 | 41 | 16.8015 25.4030 | 57 | 16.8018 35.8048 |

## MatLab code for Table 5.4

impNew

```
p=1-lambda;
q=1+lambda;
```

average of 200 iterations

---

| | 0.1 | | 0.2 | |
|---|---|---|---|---|
| dim(m*n) | iteration | time | iteration | time |
| average 5.0 5.1 | 35.9 | 0.0526 | 37.2 | 0.0603 |

---

---

| | 0.3 | | 0.4 | |
|---|---|---|---|---|
| dim(m*n) | iteration | time | iteration | time |
| average 4.8 5.1 | 40.6 | 0.0673 | 41.6 | 0.0648 |

---

## MatLab code for Table 5.6

20 iterations

---

| | 0.3 | | 0.4 | |
|---|---|---|---|---|
| dim(m*n) | iteration | time | iteration | time |
| average 200.0 300.0 | 33.8 | 2.5982 | 37.8 | 2.9957 |

---

---

| | 0.1 | | 0.2 | |
|---|---|---|---|---|
| dim(m*n) | iteration | time | iteration | time |
| average 200.0 300.0 | 36.6 | 2.8611 | 34.7 | 2.6867 |

---

| | | 0.1 | | | 0.2 | | |
|---|---|---|---|---|---|---|---|
| idx | dim(m*n) | iteration | Opt | time | iteration | Opt | time |
| 1 | 200300 | 36 | 15.3765 | 2.7472 | 35 | 15.3765 | 2.6816 |
| 2 | 200300 | 33 | 13.6721 | 2.5727 | 33 | 13.6721 | 2.5643 |
| 3 | 200300 | 39 | 13.1033 | 3.0115 | 41 | 13.1033 | 3.2237 |
| 4 | 200300 | 38 | 14.9661 | 3.0094 | 40 | 14.9661 | 3.2188 |
| 5 | 200300 | 32 | 12.9225 | 2.4362 | 28 | 12.9225 | 2.0493 |
| 6 | 200300 | 32 | 13.3005 | 2.3987 | 34 | 13.3005 | 2.5932 |
| 7 | 200300 | 38 | 11.4939 | 2.9336 | 36 | 11.4939 | 2.8237 |
| 8 | 200300 | 32 | 13.6853 | 2.4423 | 37 | 13.6853 | 2.9811 |
| 9 | 200300 | 33 | 12.6659 | 2.5796 | 28 | 12.6659 | 2.0762 |
| 10 | 200300 | 79 | 13.1793 | 6.9372 | 43 | 13.1802 | 3.4659 |
| 11 | 200300 | 33 | 12.4495 | 2.4903 | 33 | 12.4495 | 2.4935 |
| 12 | 200300 | 33 | 14.7143 | 2.5612 | 31 | 14.7143 | 2.3323 |
| 13 | 200300 | 32 | 13.0591 | 2.3790 | 34 | 13.0591 | 2.6057 |
| 14 | 200300 | 41 | 14.7339 | 3.2123 | 33 | 14.7339 | 2.4766 |
| 15 | 200300 | 34 | 15.4971 | 2.6664 | 33 | 15.4971 | 2.6206 |
| 16 | 200300 | 37 | 12.8786 | 2.9451 | 40 | 12.8786 | 3.0927 |
| 17 | 200300 | 36 | 12.6252 | 2.7991 | 37 | 12.6252 | 2.9537 |
| 18 | 200300 | 33 | 14.2547 | 2.5594 | 32 | 14.2547 | 2.4313 |
| 19 | 200300 | 32 | 15.6560 | 2.4247 | 32 | 15.6560 | 2.4553 |
| 20 | 200300 | 29 | 14.5694 | 2.1154 | 34 | 14.5694 | 2.5937 |