Algorithms for the Solution of the Quadratic Programming Problem

Martina Vankova

Submitted in partial fulfilment
of the requirements for the degree of
MAGISTER SCIENTIAE
in the Faculty of Science at the
University of Port Elizabeth

January 2004

Supervisor: Prof. G. de Kock

Acknowledgments

I would like to thank my supervisor Professor Gideon de Kock for his support, the many hours spent in the reading of this dissertation and the many valuable suggestions for improvements that were made.

I would also like to thank the Department of Computer Science and Information Systems for enabling me to conduct this research.

Further, I wish to thank Professor John Gonsalves for his assistance, encouragement and guidance for the duration of this dissertation and for his time spent proofreading the dissertation.

Finally to Deon Viljoen, I would like to express my gratitude for his assistance with the implementation of the algorithms and proofreading of the dissertation.

Table of Contents

Summary

The purpose of this dissertation was to provide a review of the theory of Optimization, in particular quadratic programming, and the algorithms suitable for solving both convex and non-convex quadratic programming problems. Optimization problems arise in a wide variety of fields and many can be effectively modeled with linear equations. However, there are problems for which linear models are not sufficient thus creating a need for non-linear systems.

This dissertation includes a literature study of the formal theory necessary for understanding optimization and an investigation of the algorithms available for solving a special class of the non-linear programming problem, namely the quadratic programming problem. It was not the intention of this dissertation to discuss all possible algorithms for solving the quadratic programming problem, therefore certain algorithms for convex and non-convex quadratic programming problems were selected for a detailed discussion in the dissertation. Some of the algorithms were selected arbitrarily, because limited information was available comparing the efficiency of the various algorithms. Algorithms available for solving general non-linear programming problems were also included and briefly discussed as they can be used to solve quadratic programming problems. A number of algorithms were then selected for evaluation, depending on the frequency of use in practice and the availability of software implementing these algorithms. The evaluation included a theoretical and quantitative comparison of the algorithms. The quantitative results were analyzed and discussed and it was shown that the results supported the theoretical comparison. It was also shown that it is difficult to conclude that one algorithm is better than another as the efficiency of an algorithm greatly depends on the size of the problem, the complexity of an algorithm and many other implementation issues.

Optimization problems arise continuously in a wide range of fields and thus create the need for effective methods of solving them. This dissertation provides the fundamental theory necessary for the understanding of optimization problems, with particular reference to quadratic programming problems and the algorithms that solve such problems.

Keywords: Quadratic Programming, Quadratic Programming Algorithms, Optimization, Non-linear Programming, Convex, Non-convex.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Throughout human history, man has always strived to master his physical environment by making the best use of his available resources. These resources are however limited and the optimal use thereof poses potentially difficult problems. Problems of finding the best or worst situation arise constantly in daily life in a wide variety of fields that include science, engineering, economy and management. The theory of optimization attempts to find these solutions.

The theory and application of optimization is sometimes referred to as mathematical programming. Here the term programming does not refer to computer programming, but rather the allocation or composition of limited resources, although computers are extensively and in fact usually used to solve these problems. Programming problems deal with optimal allocation of limited resources such as equipment, raw materials and labor to the manufacture of one or more products. Therefore the aim is to allocate these resources in such a way that the products meet their given specifications, while at the same time maximizing some profit or minimizing cost.

To understand the concept of optimization problems, consider the case of a farmer who wants to plant a variety crops. His production is however limited by many factors. Amongst some of these are the availability of land, labor and water. After the crops have been harvested, they will be transported to different markets for sale. This give rise to the following questions: How much of each crop should he grow in order to make a maximum profit? What is the cheapest way of delivering the goods to a number of destinations?

Another example is to find the shape of a string, which is anchored at its ends along a given straight line and encloses the most area between itself and the line. This is often called Dido's problem, originating in the 18^{th} century BC, named after a Phoenician princess Dido of Tyre, who according to the legend founded the city of Carthage in North

Africa. She fled to Africa after her brother murdered her husband and appropriated her fortune. There she persuaded a local chief to sell her as much land as an ox hide could contain. She then cut the hide into thin strips and tied them together to make a cord 4km long. She laid the cord down in a semicircle with the ends touching the coast. This turns out to enclose the largest possible area and is considered as the original optimization problem.

There are many such problems arising in a wide range of fields. These can vary in size, from problems similar to the farmers crops involving few decision criteria to very large problems involving thousands of factors, for example, chemical reactions. Enormous efforts have been made to describe these complex human and social situations with mathematical expressions.

Initially only relatively simple problems, usually expressed using linear equations, were modeled. The evolution of mathematics and physics allowed for more complex, but also more accurately formulated problems to be modeled using non-linear equations. This together with the development of highly efficient algorithms and the vast increase in computational power [Sas1999] allows for large and complex problems to be solved more easily than in the past. In addition to solving large scale problems modern computers allow for the simulation and visualization of these problems. Some specific examples of optimization include [Win1995]:

Police patrol officer scheduling in San Francisco (linear programming), saved $11 million per year;

Reducing fuels costs in the electricity power industry (dynamic programming), saved over $125 million in costs;

Petrol blending (non-linear programming), saves $30 million per year;

Scheduling trucks (dynamic programming), reduced costs by $2.5 million per year;

Maximize the expected return of several investments while at the same time minimizing risk (quadratic programming);

Minimize the power loss in an electrical networks while satisfying the conservation of flow (quadratic programming); and

Optimization problems can also be found in modeling of digital circuits, chemical reactions, power flow, population ecology, heartbeat and nerve impulses, crop production, food mix problems etc.

1.2 Aims and Objectives

The aims of this dissertation are as follows:

Provide a concise review of the theory of Optimization with particular reference to quadratic programming;

Review the algorithms suitable for solving both the convex and non-convex quadratic programming problems;

Collect software applications for solving quadratic programming problems;

Select a number of the algorithms to compare and discuss their efficiency in solving a variety of test problems, using the collected software; and

Show how the fixed charge problem, often associated with integer programming can be expressed as a non-convex quadratic programming problem.

The theory of optimization provides a powerful framework for formulating the general optimization problem. This dissertation is however, concerned with algorithms for solving quadratic programming problems and comparison of these algorithms, rather than to show how quadratic programming problems are formulated.

1.3 Layout

Chapter 2 explores the key concepts which form part of the basic knowledge necessary for the rest of the dissertation. The general optimization problem and its classification are presented in mathematical context. The conditions for optimality, the identification of local and global optimum points, the convexity of the objective function and the Kuhn-Tucker conditions are described, with particular reference to the quadratic programming problem.

In Chapter 3 a selection of algorithms for solving the quadratic programming problem specifically concerned with a convex objective function are discussed. Examples are provided throughout the chapter to aid in the understanding of the algorithms.

Chapter 4 deals with algorithms for solving non-convex quadratic programming problems. It will be shown that these algorithms do not necessarily produce a global optimum. As in Chapter 3 examples are used in the explanation of the algorithms. Chapter 4 will also

introduce an important non-convex quadratic programming problem called the fixed charge problem.

Chapter 5 gives a brief overview of a selection algorithms for solving the general non-linear programing problem. These algorithms can be used to solve the more restrictive case of the non-linear problem, more specifically the quadratic programming problem which is the reason for introducing these algorithms.

Chapter 6 will provide a comparison of selected algorithms by providing some theoretic comparisons of algorithms and quantitative comparison using numerical examples to compare these algorithms for efficiency. In this chapter a brief description of software available for solving quadratic programming problems will also be provided.

Chapter 7 will present a summary of the research and the conclusions that have arisen during the research and provide some recommendations for future research.

Chapter 2

Optimization

2.1 Introduction

Optimization is the process whereby we seek to find the best or optimal value of a function, usually subject to certain constraints or restrictions. The function with it's restrictions is a mathematical optimization model that represents certain aspects of the physical environment [Ars2003]. Optimization models are used extensively in many areas of decision making and are the cornerstone of most optimization studies. There are many methods available with which optimization problems may be solved. However, not all optimization problems can be solved efficiently with all the methods. The methods are appropriate for only certain types of problems, each designed to account for specific mathematical properties of the model. It is thus important to be able to identify the characteristics of a problem in order to find the correct solution method.

The next section studies the criteria that facilitate the identification of optimal points and presents optimization in a mathematical context. It provides the background for understanding the remainder of this dissertation.

2.2 The General Optimization Problem

In order to provide a framework for discussing optimization problems, this section introduces the general definition of the problem. In general the optimization problem is formulated as a function f of n variables $\boldsymbol{x}=(x_1,\ldots,x_n)'$ written as

(2.2.1) $$f(\boldsymbol{x})=f(x_1,\ldots,x_n).$$

This function $f(x_1,\ldots,x_n)$ is referred to as the *objective function* and the variables $\boldsymbol{x}=(x_1,\ldots,x_n)'$ are called *decision variables* [Win1995]. To find the optimum of the objective function means to determine the values of the n variables, such that, the function f is either minimized or maximized.

In order to restrict the scope of a problem, the variables $x_1, \ldots, x_n$ are usually restricted in some way, for example they may be forced to be non-negative or may not exceed a given value. These restrictions are represented in the form of equations known as *constraint equations* or *constraints*. They are denoted by g and are functions of the n variables. The constraint equations can be defined as follows:

$$\begin{aligned} g_1(x_1, \ldots, x_n) \{\leq, =, \geq\} b_1 \\ g_2(x_1, \ldots, x_n) \{\leq, =, \geq\} b_2 \\ \\ g_p(x_1, \ldots, x_n) \{\leq, =, \geq\} b_p \end{aligned} \tag{2.2.2}$$

where the $b_i, i = 1, 2, \ldots, p$ are assumed to be known constants. Often the constraints are simple functions, for example the non-negativity constraints

$$x_j \geq 0, \quad j = 1, \ldots, n.$$

Optimization problems can be classified as either discrete or continuous. In discrete optimization problems, some or all of the variables have integral values at the optimal solution. Continuous optimization requires the functions in the problem to be continuous, with possibly continuous derivatives. This means that the solution value of each of the n variables can be any real number [Lue1973].

Continuous optimization can further be classified as constrained and unconstrained optimization. An unconstrained optimization problem is one whose objective is to maximize or minimize the function $f(\boldsymbol{x})$, without regard to any of the constraints (2.2.2). A constrained optimization problem attempts to minimize or maximize the objective function $f(\boldsymbol{x})$ and requires that the constraints (2.2.2) are satisfied in order to obtain the optimum, that the objective is to solve the following problem.

$$\begin{aligned} &\textit{Minimize / Maximize} && f(x_1, \ldots, x_n) \\ &\textit{Subject to} && g_1(x_1, \ldots, x_n) \{\leq, =, \geq\} b_1 \\ & && g_2(x_1, \ldots, x_n) \{\leq, =, \geq\} b_2 \\ & && \\ & && g_m(x_1, \ldots, x_n) \{\leq, =, \geq\} b_m \end{aligned} \tag{2.2.3}$$

The constrained problem (2.2.3) is referred to as the *Mathematical Programming Problem*.

An important concept associated with mathematical programming is the *feasible region*, which is defined as follows:

Definition 2.1 Feasible region:
The set of all points, denoted by F, satisfying all the constraints is called the feasible set or feasible region [Win1995].

$$F = \{ \boldsymbol{x} \mid g_i(\boldsymbol{x}) \{\leq, =, \geq\} b_i, \;\; i = 1, 2, \ldots, p \}$$

Given the above information, an optimal solution for the mathematical programming problem (2.2.3) is a point in the feasible region with the smallest objective function value for minimization and the largest objective function value for maximization.

The most commonly used approach for solving constrained problems is formulating (2.2.3) as a linear programming problem [CS1970]. Here the objective function and all the constraints are linear functions of the decision variables. A non-linear programming problem is a generalization of a linear programing problem, where the objective function and/or any of the constraints are non linear. Linear programming problems are usually relatively simple to solve, even if they are large. However, finding the solution for many non-linear problems is still very difficult in spite of the large amount of research being conducted in this field.

As with linear programming problems, non-linear programming problems require that the variables must be non-negative, that is, the non-negativity constraints must be included as part of the constraints in (2.2.3). The mathematical programming problem is then written with the non-negativity constraints written explicitly (with $p = m \quad n$) as

(2.2.4)

$$\begin{aligned} &\textit{Minimize / Maximize} && f(x_1, \ldots, x_n) \\ &\textit{Subject to} && g_1(x_1, \ldots, x_n) \{\leq, =, \geq\} b_1 \\ & && g_2(x_1, \ldots, x_n) \{\leq, =, \geq\} b_2 \\ & && \\ & && g_m(x_1, \ldots, x_n) \{\leq, =, \geq\} b_m \\ & && x_j \geq 0, \;\; j = 1, \ldots, n. \end{aligned}$$

A particularly well studied non-linear programming problem is Quadratic programming, in which the objective function is quadratic with all the constraints in terms of linear functions. In fact the main focus of this dissertation is the study of quadratic programming problems and the algorithms used for solving such problems. There are other types of constrained optimization problems that have not been mentioned here, however these will not be discussed in this dissertation. Figure 2.1 illustrates a graphical representation of the

classification of optimization problems with particular focus on linear and quadratic programming.

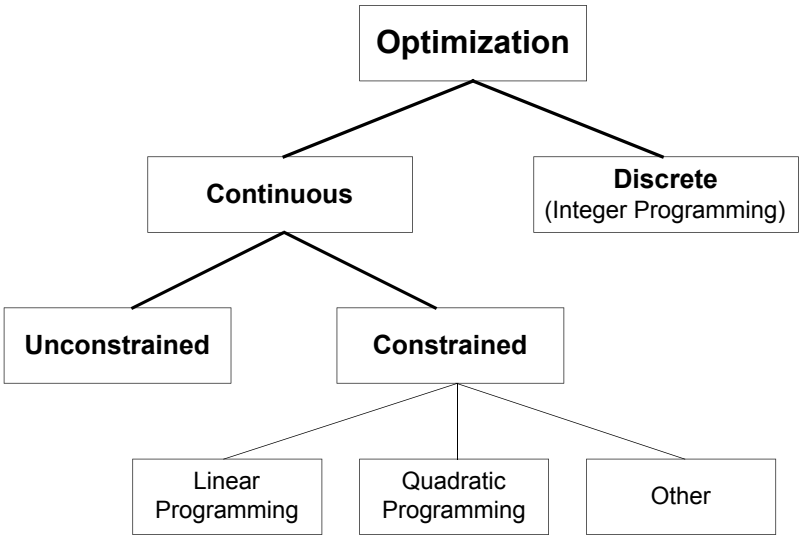

Figure 2.1 Classification of Optimization

Although this dissertation focuses on the quadratic programming problems, both linear programming problems as well as unconstrained optimization problems will be discussed as they provide a basis for the methods to be discussed in the later chapters. The next section will discuss some basic mathematical concepts that are required in order to solve such problems and serves as a reference to the rest of the dissertation.

2.3 Unconstrained Optimization

This section characterizes the conditions that must be satisfied in order to find a solution to the unconstrained optimization problem and introduces the important concepts of convex/concave functions, which are necessary for the development of the optimization theory.

2.3.1 Maximum and Minimum for Functions of One Variable

One can distinguish between two kinds of extreme points, namely local and global optimal point [Lue1973].

Definition 2.2 Local Minimum (Relative Minimum):
Let f be a real-valued function, then f has a local minimum at the point p if there exists a $\delta > 0$ such that $f(x) \geq f(p)$ for all x such that $|x - p| < \delta$.

Similarly p is local maximum if $f(x) \leq f(p)$ for all x sufficiently near p .

For differentiable functions in the following result is obtained

Theorem 2.1
If f is differentiable at p and is defined on containing the point p and if $f(p)$ is either a local minimum or a local maximum of f, then $f'(p) = 0$ [BF1993].

Thus the equation $f'(p) = 0$ is a necessary condition for $f(p)$ to be a local minimum or maximum, provided that f is differentiable. This is however not a sufficient condition as it is possible for $f'(p) = 0$ at a point which is neither a local minimum nor a local maximum. Such a point is called a *point of inflection* or a *saddle point* and is illustrated in Figure 2.2 as the point x_5. The points p for which $f'(p) = 0$ are called *stationary points*, that is, local and global minimum and maximum points and points of inflection are stationary points.

Definition 2.3 Global Minimum (Absolute Minimum):
Let f be a real-valued function with domain D, then $f(p)$ is called a global or absolute minimum value of f over D, provided that $f(p) \leq f(x)$ for all x D. Thus $f(p)$ is the smallest value of f on D [Mil2000].

The following theorem states that the global minimum and maximum values of a continuous function f is at some stationary point of f.

Theorem 2.2 Extreme Value Theorem
Suppose that $f(p)$ is the global minimum (or global maximum) value of the continuous function f, then p is a stationary point of f [BF1993].

To understand these ideas consider the Figure 2.2.

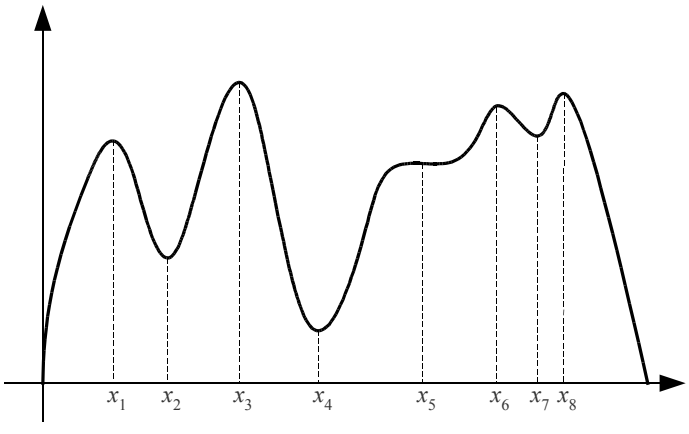

Figure 2.2 Local and global maximum/minimum and an inflection point

The function has a local minimum, a maximum or an inflection point at a point where the tangent to the function at that point is parallel to x-axis, which means that the derivative there is zero. Since the derivative is zero at x_i (i = 1,2,3,4,5,6,7,8 in Figure 2.2), all these points are stationary points. A zero-valued derivative is a necessary condition for a maximum or a minimum point in the range - ∞ to + ∞.

Figure 2.2 represents a function with more than one stationary point. Assuming it has no stationary points other than shown in the figure, then the function has a local maximum at x_1, x_3, x_6 and x_8. There is a local minimum at x_2, x_4 and x_7. The point x_5 represents a point of inflection.

Three methods can be used to distinguish maxima, minima and points of inflection. Consider two points $p_+ = p$ and $p_- = p$ for any small >0, that is, points that are slightly to the right and to the left of p respectively, as shown in Figure 2.3.

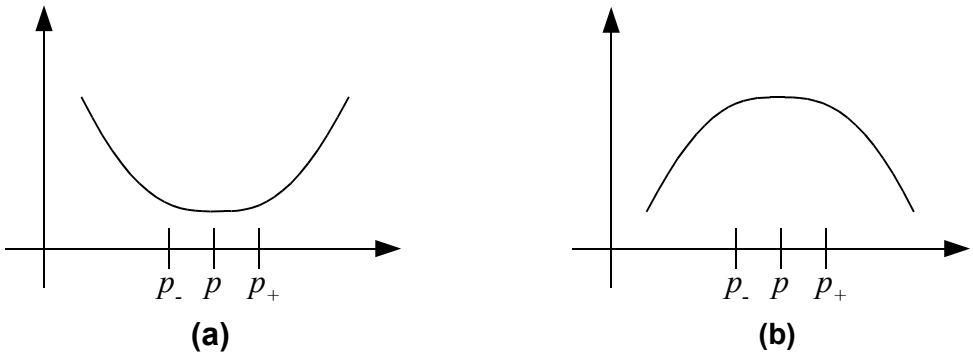

Figure 2.3 (a) A minimum point at p (b) A maximum point at p

By making reference to Figure 2.3 the three methods of distinguishing minimum, maximum and inflection points are:

1. Determine $f(p_+)$ and $f(p_-)$. If $f(p) \leq f(p\ )$ and $f(p) \leq f(p_-)$, then p represents a local minimum point. If $f(p) \geq f(p_+)$ and $f(p) \geq f(p_-)$, then p represents a local maximum point. Otherwise it is a point of inflection.
2. Determine $f'(p_+)$ and $f'(p_-)$. If $f'(p_+) > 0$ and $f'(p_-) < 0$ then p is a local minimum (or maximum if the signs are reversed). If the signs are both the same then p is an inflection point.
3. A third approach of determining a stationary point requires the evaluation of the second derivative. Determine $f''(p_+)$ and $f''(p_-)$. If $f''(p_+) > 0$ and $f''(p_-) > 0$ then p represents a local minimum. If $f''(p_+) < 0$ and $f''(p_-) < 0$ then p represents a local maximum. If the signs are different for $f''(p_+)$ and $f''(p_-)$, then p is an inflection point.

This result can be derived from Taylor's Theorem [Mil2000] and is provided in Appendix A.1. Table 2.1 summarizes the necessary and the sufficient conditions for maxima and minima for $f(x)$. The next section will introduce the concepts of convex and non-convex sets.

Table 2.1 Characterization of Local Maxima and Minima for $f(x)$ at p

	Local Maximum	**Local Minimum**
First order necessary condition	$f'(p) = 0$	$f'(p) = 0$
Second order necessary condition	$f''(p) \leq 0$	$f''(p) \geq 0$
Sufficient condition	$f'(p) = 0$ and $f''(p) < 0$	$f'(p) = 0$ and $f''(p) > 0$

2.3.2 Convexity/Concavity for Functions of One Variable

Convexity/concavity of $f(x)$ is a description of its shape. The knowledge of the convexity/concavity of functions is essential in solving many of the non-linear programming problems. The convexity/concavity of a function can be identified using secant or tangent lines. Consider the function $f(x)$ to be a function of one variable as shown in Figure 2.4. The function is convex if it lies below any secant line or above any tangent line as is the case with Figure 2.4. Conversely a function is concave if it lies above any secant line or below any tangent line [Mil2000].

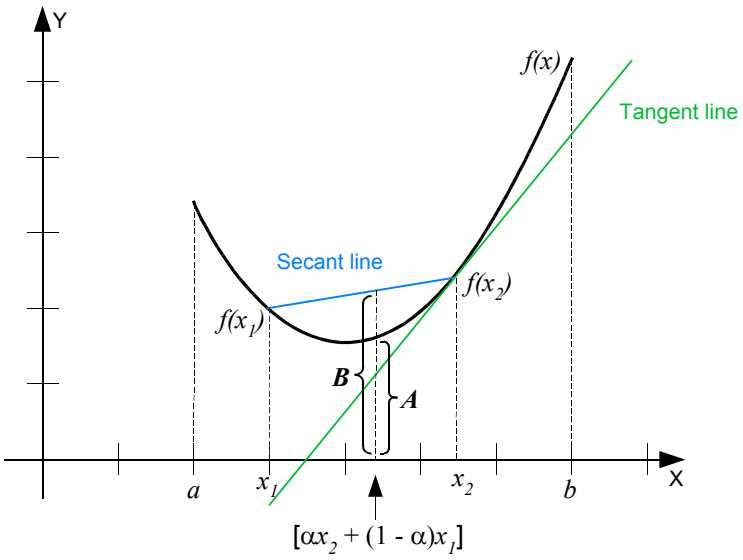

Figure 2.4 Convex function relative to secant and tangent lines

Figure 2.4 shows a convex function over the interval $[a, b]$ (the interval $[a, b]$ does not represent any restrictions on $f(x)$). Clearly the function lies above the tangent line in the interval $[a, b]$ and is therefore convex. At the points x_1 and x_2 the function takes on the values $f(x_1)$ and $f(x_2)$ respectively. The secant to the function between x_1 and x_2 is the line $\alpha f(x_2) \quad (1 \quad \alpha) f(x_1)$ for all α in $[0,1]$. As shown in the Figure $A = f(\alpha x_2 \quad (1 \quad \alpha) x_1)$ and $B = \alpha f(x_2) \quad (1 \quad \alpha) f(x_1)$. Clearly the function $f(x)$ is convex on the interval $[x_1, x_2]$ and on $[a, b]$ if $A \leq B$ for any $x_1, x_2 \quad [a, b]$.

Definition 2.4 Convex set:
A set of points S is a convex set if the line segment joining any pair of points in S is wholly contained in S [Win1995].

Definition 2.5 Convex function:
A function $f(x)$, defined on a convex set S, is said to be convex if for every α *such that* $0 \le \alpha \le 1$ *and for every pair of points* $x_1, x_2 \quad S$

$$f(\alpha x_2 \quad (1 \quad \alpha) x_1) \le \alpha f(x_2) \quad (1 \quad \alpha) f(x_1)$$ [Mil2000].

The definition of concavity follows similarly with the inequality reversed.

The second order derivative can also be used to determine if a function is convex/concave. The following theorem gives the second order sufficient condition for convexity/concavity.

Theorem 2.3 Test for convexity/concavity of a function $f(x)$: [Win1995]
Suppose that the function $f(x)$ is twice differentiable for all x in a convex set S, then
(a) If $f''(x) > 0$ on S, the function $f(x)$ is convex over S.
(b) If $f''(x) < 0$ on S, the function $f(x)$ is concave over S.

The following theorem states the importance of convex/concave functions.

Theorem 2.4
If $f(x)$ is a convex function over a convex set S, then any local minimum of $f(x)$ in S is also the global minimum of $f(x)$ over S [CS1970].

Similarly it follows that if $f(x)$ is a concave function then any local maximum of $f(x)$ is also the global maximum of $f(x)$ over S.

2.3.3 Maxima and Minima for Functions of *n* Independent Variables

The previous results can now be generalized for a real-valued function of n variables on the domain $U \quad ^n$. In future we shall use the standard notation where bold, lower case letter (e.g. $\boldsymbol{x}$) refers to a vector of n variables, a bold, uppercase letter (e.g. $\boldsymbol{A}$) refers to a matrix and the transpose of a matrix is indicated by prime (e.g. $\boldsymbol{A}'$).

Definition 2.6
Let $f: U \quad ^n \rightarrow$, then $\boldsymbol{p} \quad U$ is a local minimum of f in U, if there is a neighborhood $V^{(\varepsilon)} = \{\boldsymbol{x}: |\boldsymbol{x} \quad \boldsymbol{p}| < \varepsilon, \varepsilon > 0, \boldsymbol{x} \quad U\}$ of $\boldsymbol{p}$ such that $f(\boldsymbol{x}) \ge f(\boldsymbol{p})$ for all points $\boldsymbol{x} \quad V^{(\varepsilon)}$ for a sufficiently small ε [Mil2000].

Similarly $\boldsymbol{p}$ is a local maximum if $f(\boldsymbol{x}) \le f(\boldsymbol{p})$ for all points $\boldsymbol{x} \quad V^{(\varepsilon)}$. The following theorem can be used to identify the stationary points.

Theorem 2.5

Let $f: U \subseteq \mathbb{R}^n \to \mathbb{R}$ be differentiable. If $\boldsymbol{p} \in U$ is a local minimum, then $\frac{\partial f}{\partial x_j}(\boldsymbol{p}) = 0$, $j = 1, \ldots, n$, (that is, $\boldsymbol{p}$ is a stationary point of f) [Mil2000].

Once again this theorem has only identified necessary conditions for extreme points, but as these conditions are also satisfied by inflection or saddle points we refer to these points as stationary points.

Consider now the second order derivative for a function of n variables.

Definition 2.7 Hessian Matrix [Mil2000]

The Hessian ($\boldsymbol{H}$) of $f(\boldsymbol{x})$ is the n x n matrix

$$\boldsymbol{H} = \left[\frac{\partial^2 f}{\partial x_i \partial x_j} \right].$$

A concept that plays an important role in the theory of optimization is the *quadratic form*, more specifically the sign of the form. It plays an important role in distinguishing maxima from minima using classical derivative based techniques.

A quadratic form in the n scalar variables $x_1, x_2, \ldots, x_n$ is an expression of the form:

$$\begin{aligned} f(x_1, \ldots, x_n) = {} & q_{11}x_1^2 \;\; 2q_{12}x_1x_2 \;\; 2q_{13}x_1x_3 \;\; \ldots \;\; 2q_{1n}x_1x_n \ldots \\ & \ldots \;\; q_{22}x_2^2 \;\; 2q_{23}x_2x_3 \;\; \ldots \;\; 2q_{2n}x_2x_n \ldots \\ & \ldots \;\; q_{33}x_3^2 \;\; 2q_{34}x_3x_4 \;\; \ldots \;\; 2q_{3n}x_3x_n \ldots \\ & \ldots \;\; \ldots \;\; q_{nn}x_n^2 \end{aligned} \tag{2.3.1}$$

This expression can be made symmetric by setting $q_{ij} = q_{ji}$ for $i > j$ so that

$$\begin{aligned} f(x_1, \ldots, x_n) = {} & (q_{11}x_1 \;\; q_{12}x_2 \;\; \ldots \;\; q_{1n}x_n)x_1 \\ & (q_{21}x_1 \;\; q_{22}x_2 \;\; \ldots \;\; q_{2n}x_n)x_2 \\ & \\ & (q_{n1}x_1 \;\; q_{n2}x_2 \;\; \ldots \;\; q_{nn}x_n)x_n \end{aligned} \tag{2.3.2}$$

Defining a symmetric n x n matrix $\boldsymbol{Q} = (q_{ij})$ then the equation (2.3.2) can be expressed in matrix form as

$$f(\boldsymbol{x}) = \boldsymbol{x}' \boldsymbol{Q} \boldsymbol{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j. \tag{2.3.3}$$

The following definitions are associated with quadratic forms [KKO1966]

Definition 2.8 Positive Definite:

If $\boldsymbol{x}' \boldsymbol{Q} \boldsymbol{x} > 0$ for all non-zero vectors $\boldsymbol{x}$ in $\mathbb{R}^n$, then the matrix $\boldsymbol{Q}$ is called positive definite.

Definition 2.9 Negative Definite:

If $\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}<0$ *for all non-zero vectors* $\boldsymbol{x}$ *in* ^{n}, *then the matrix* $\boldsymbol{Q}$ *is called negative definite.*

Definition 2.10 Positive Semi-definite:

If $\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}\geq 0$ *for all vectors* $\boldsymbol{x}$ *in* ^{n}, *then the matrix* $\boldsymbol{Q}$ *is called positive semi-definite.*

Definition 2.11 Negative Semi-definite:

If $\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}\leq 0$ *for all vectors* $\boldsymbol{x}$ *in* ^{n}, *then the matrix* $\boldsymbol{Q}$ *is called negative semi-definite.*

If the matrix $\boldsymbol{Q}$ cannot be classified as either definite or semi-definite it is called *indefinite* [Mil2000].

The next theorem establishes the sufficiency conditions for $\boldsymbol{p}$ to be an extreme point.

Theorem 2.6 [Tah1997]

A sufficient condition for a stationary point $\boldsymbol{p}$ *to be a local extremum is for the Hessian matrix* $\boldsymbol{H}$, *evaluated at* $\boldsymbol{p}$, *to be*

(i) positive definite when $\boldsymbol{p}$ *is a local minimum point*

(ii) negative definite when $\boldsymbol{p}$ *is a local maximum point*

The rules for stationary points of a function on n variables are summarized in Table 2.2 with the following notation $f=\left(\frac{\partial f}{\partial x_j}\right), j=1,\ldots,n.$

Table 2.2 Characteristics of Local Maxima and Minima for $f(x_1,\ldots,x_n)$ at $\boldsymbol{p}$

	Local Maximum	**Local Minimum**
First order necessary condition	$f(\boldsymbol{p})=0$	$f(\boldsymbol{p})=0$
Second order necessary condition	$\boldsymbol{H}$ be negative semi-definite	$\boldsymbol{H}$ be positive semi-definite
Sufficient condition	$f(\boldsymbol{p})=0$ and $\boldsymbol{H}$ be negative definite	$f(\boldsymbol{p})=0$ and $\boldsymbol{H}$ be positive definite

2.3.4 Convex and Non-convex Sets

Definition 2.12 Convex Set:

The set S ^{n} *is said to be convex if and only if for all* $\boldsymbol{x}_1, \boldsymbol{x}_2$ S

$$\alpha \boldsymbol{x}_1 \quad (1 \quad \alpha)\boldsymbol{x}_2 \quad S$$

and α $[0,1]$ [Mil2000].

In other words, a set of points is a convex set if the line segment joining any pair of points in the set, is completely contained in the set. For example, ^{n} and $\{\boldsymbol{x}: \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$ are convex sets with $\boldsymbol{A}$ a n-element row vector. Figure 2.5 illustrates some examples of convex and non-convex sets of two variables.

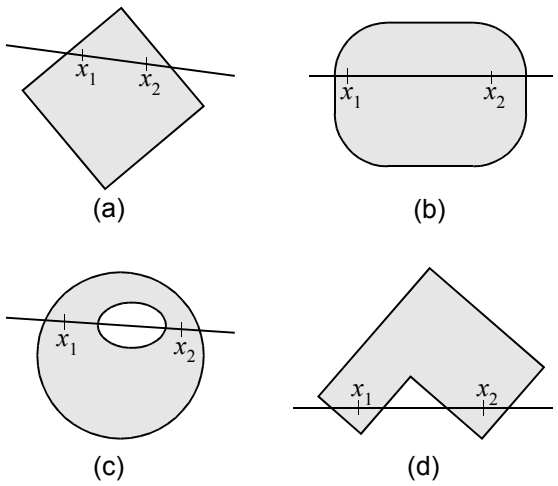

Figure 2.5 (a) and (b) Convex Sets and (c) and (d) Non-convex Sets

2.3.5 Convexity/Concavity for Functions of n Variables

The concept of convexity/concavity of a function of one variable $f(x)$ as described in Section 2.3.2 can be extended for a function $f(\boldsymbol{x})$ of n variables. Instead of secant lines and tangent lines the terminology of secant hyperplanes and tangent hyperplanes is used. A definition of convex function $f(\boldsymbol{x})$ now follows as

Definition 2.13 Convex function:

A function $f(\boldsymbol{x})$ *is a convex on a convex set* S *of points* $\boldsymbol{x}=(x_1,\ldots,x_n)$ *if for every* α *in* $[0,1]$ *and every pair of points* $\boldsymbol{x}_1, \boldsymbol{x}_2$ S

$$f(\alpha \boldsymbol{x}_2 \quad (1 \quad \alpha)\boldsymbol{x}_1) \leq \alpha f(\boldsymbol{x}_2) \quad (1 \quad \alpha) f(\boldsymbol{x}_1) \text{ [Mil2000].}$$

The definition of a concave function follows similarly with the inequality reversed.

It can be shown [BSS1993] that the Hessian matrix can be used to determine whether the function $f(\boldsymbol{x})$ is convex or concave. The following definitions are useful and are taken from Winston [Win1995]:

Definition 2.14

An i^{th} principal minor of an n x n matrix is the determinant of any i x i matrix obtained by deleting n i rows and the corresponding n i columns of the matrix.

Definition 2.15

An k^{th} leading principal minor of an n x n matrix is the determinant of the k x k matrix obtained by deleting the last n k rows and columns of the matrix.

The following theorem now shows that the Hessian matrix can be used to determine whether $f(\boldsymbol{x})$ is convex or concave.

Theorem 2.7

(i) Suppose that $f(\boldsymbol{x})$ has continuous second order partial derivatives for each point $\boldsymbol{x}=(x_1,\ldots,x_n)$ S. Then $f(\boldsymbol{x})$ is a convex function on S if and only if for each $\boldsymbol{x}$ S, all principal minors of $\boldsymbol{H}$ are non-negative.

(ii) The function $f(\boldsymbol{x})$ is concave on S if and only if for each $\boldsymbol{x}$ S and $k=1,2,\ldots,n$, all non-zero principal minors have the same sign as $(-1)^k$.

The following paragraphs extend some of the definitions and theorems that were introduced earlier when discussing unconstrained optimization problems for constrained problems. Denote by S the set of all points $\boldsymbol{x}$ satisfying the constraints of problem (2.2.3), that is, points that lie in the feasible region F.

Definition 2.16 Local Minimum (Relative Minimum):

The function $f(\boldsymbol{x})$ is said to have a local minimum over a closed set S at the point $\boldsymbol{p}$, if $\boldsymbol{p}$ S and there exists an $\varepsilon>0$ such that for every $\boldsymbol{x}\neq\boldsymbol{p}$ in an ε neighborhood of $\boldsymbol{p}$ for which $\boldsymbol{x}$ S, $f(\boldsymbol{x})\geq f(\boldsymbol{p})$ [Lue1973].

Definition 2.17 Global Minimum (Absolute Minimum):

The function $f(\boldsymbol{x})$ is said to take on a global minimum over a closed set S ^{n}, at the point $\boldsymbol{p}$ S if for all $\boldsymbol{x}$ S, $f(\boldsymbol{x}) \geq f(\boldsymbol{p})$ [Lue1973].

Both of these definitions can be stated similarly for maximization with the inequality reversed, that is, $f(\boldsymbol{x}) \leq f(\boldsymbol{p})$.

Figure 2.6 extends Figure 2.2 over the closed interval.

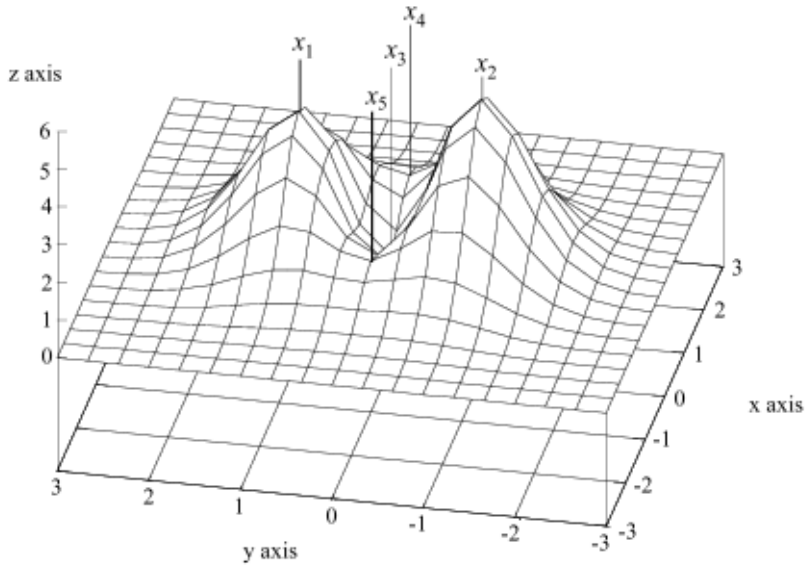

Figure 2.6 Constrained local and global maximum and minimum and an inflection point

In Figure 2.6 the function has a local maximum at x_1 and x_2, a local minimum at x_3 and the points x_4 and x_5 represent a point of inflection. If the interval is a closed interval, then the endpoints of the interval also represent local optimum points. However, the derivative of f at the endpoints is not necessarily zero [Mil2000].

Earlier the concept of convex/concave sets and functions was introduced. The following theorem illustrates the importance of these concepts.

Theorem 2.8

Consider the non-linear programming problem (2.2.3) expressed as a function of the decision variables $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)'$. *Suppose that the feasible region* F *is a convex set. If* $f(\boldsymbol{x})$ *is convex on* F, *then any local minimum is an optimal solution to the non-linear programming problem* [Win1995].

A similar conclusion can be stated about a maximization problem with a concave objective function $f(\boldsymbol{x})$.

The next section will briefly introduce some techniques necessary for solving the unconstrained optimization problems.

2.4 Solving Unconstrained Optimization Problems

An unconstrained problem is a problem of the form (2.2.1) which is maximized or minimized without any constraints imposed on the variables $\boldsymbol{x}$. Unconstrained problems seldom arise in practical applications, however they do provide a good introduction towards the understanding of constrained problems for two reasons. Firstly, the scope of a constrained problem may be widened so that the constraints imposed on the problem may vanish, thus making it an unconstrained problem. Secondly, constrained problems can in some cases be easily converted to unconstrained problems [Lue1973]. The underlying theory of unconstrained problems also provides the basis for devising many non-linear programming algorithms.

For a function of one variable the first order condition $f'(x)$ can be used to identify optimum points as $f'(x)=0$ identifies local optimum points. For a linear function of more than one variable the first order condition requires the solution of a system of equations $f(\boldsymbol{x})$.

In general however, these equations will not be linear. In this case second order conditions can be used to distinguish minimum points, maximum points or points of inflection. There are other methods for finding minima and maxima of functions, that may or may not use the

first and second order conditions. These methods are called *iterative methods* also known as *search methods* [Mil2000].

The iterative methods determine the solution through a series of estimates where each successive estimate is usually but not always better than the previous one. The iterative methods consist of three components:

1. A set of initial estimates for the unknown variables $\boldsymbol{x}_0$.
2. An algorithm for finding the next value, $\boldsymbol{x}_{k\ 1}$ from the current value $\boldsymbol{x}_k$, where $k = 0, 1, \ldots, n$.
3. A rule for deciding when the current estimate $\boldsymbol{x}_k$ is close enough to the exact (unknown) solution.

Some examples of iterative methods for finding the optimum for functions of a single variable include [Mil2000]:

Simultaneous methods: The first step is to select an interval over which to search, for example $[a, b]$. This interval is then subdivided into smaller intervals, which could either be equally spaced or randomly selected depending on the technique used. The value of the function is evaluated at each of the intermediate points, which then provides information about the shape of the function. On the basis of the information a new interval is selected and again subdivided into smaller intervals or the procedure stops;

Sequential methods: In these methods the choice of the evaluation points is made sequentially. Each new choice for a variable is based on the information provided from the previous step;

Curve Fitting algorithms: These methods approximate the function by a series of increasingly accurate curves like parabolas, cubic or higher order polynomials; and

Combined Techniques: Often the best approach to find an optimum is to use a combination of the techniques introduced above. For example, begin with sequential method and then, at some point when the search interval is small, switch to a curve fitting method, when it can be safely assumed that a function has a particular shape.

Some of the methods for finding the optimum for a function of multiple variables are as follows:

Gradient methods: The general idea is to generate successive points, starting from a given initial point, in the direction of the greatest decrease (minimization) of the function. Start with some initial point $\boldsymbol{x}^0$. Define $f(\boldsymbol{x}^k)$ as the gradient of f at the point $\boldsymbol{x}^k$ and $\boldsymbol{r}^k$ defines the step size which is determined such that $\boldsymbol{x}^{k+1}$ results in the largest improvement. The result is achieved by computing successive points $\boldsymbol{x}^{k+1}=\boldsymbol{x}^k - \boldsymbol{r}^k f(\boldsymbol{x}^k)$. The procedure is stopped when $f(\boldsymbol{x}^k)=0$ [CS1970]. This is however only a necessary condition unless it is known that the function is convex;

Newton's method: The method starts with some initial approximation $\boldsymbol{x}^0$. The next approximation $\boldsymbol{x}^n$ is the x-intercept of the tangent line to the function f at the point where $\boldsymbol{x}^{n+1}$ and the function f intersect. The procedure is terminated with $\boldsymbol{x}^p$ as the solution when $\boldsymbol{x}^p \approx \boldsymbol{x}^{p+1}$ [BF1993];

Conjugate Gradient Methods: The idea of the method is to obtain successive direction vectors by evaluating the current negative gradient vector and adding it to the linear combination of the previous direction vectors to obtain a new conjugate direction along which to move. Begin with some initial point $\boldsymbol{x}_0^0=(x_1^0,\ldots,x_n^0)'$ and an initial set of directions given by $\boldsymbol{D}^0=\{\boldsymbol{D}_1^0,\ldots,\boldsymbol{D}_n^0\}=\{\boldsymbol{I}_1,\ldots,\boldsymbol{I}_n\}$. Find the minimum along $\boldsymbol{I}_1$ direction to the next minimum point labeled $\boldsymbol{x}_1^0$. Then using the new value of x_1^0 and the old values of $x_2^0,\ldots,x_n^0$ find the minimum in the $\boldsymbol{I}_2$ direction and call the new point $\boldsymbol{x}_2^0$. Continue this way along each $\boldsymbol{I}_i$ direction until the point $\boldsymbol{x}_n^0$ is located. Calculate the direction $[\boldsymbol{I}_i \quad \boldsymbol{I}_0]$ then a search is done along this direction to get the next starting location at $\boldsymbol{x}_0^1$ for the next set of n line minimizations. Construct the new search direction $\boldsymbol{D}^1$ to calculate the next set of points. Continue in this way to construct direction $\boldsymbol{D}^k$ with new set of points $\boldsymbol{x}_0^k$. The procedure continues until the gradient at some point becomes sufficiently small [Mil2000];

Sequential Univariate Search: In these methods the arguments for which $f(\boldsymbol{x})$ is evaluated cannot be known in advance. Instead the sequence of argument values depends on the previously observed values of $f(\boldsymbol{x})$. The idea is to change one variable at a time so that the function is minimized in each direction. Begin by choosing an initial point $\boldsymbol{x}^0=(x_1^0,\ldots,x_n^0)'$. The next point $\boldsymbol{x}^1$ is obtained by performing minimization with respect to x_1^0, that is, $\boldsymbol{x}^1=\boldsymbol{x}^0 - \alpha_1 \boldsymbol{e}_1$ where $\boldsymbol{e}_1=(1,0,\ldots,0)'$ and α_1 is a scalar such that $f(\boldsymbol{x}^0 - \alpha_1 \boldsymbol{e}_1)$ is minimized. The general step to find the point $\boldsymbol{x}^{k+1}$

is completed by performing a minimization with respect to the variable x_k^0, that is, $\boldsymbol{x}^{k\ 1} = \boldsymbol{x}^k\ \ \alpha_{k\ 1}\boldsymbol{e}_{k\ 1}$ for $k = 0, 1, \ldots, n\ \ 1$ such that $f(\boldsymbol{x}^k\ \ \alpha_{k\ 1}\boldsymbol{e}_{k\ 1})$ is minimized. The procedure continues in this alternating sequence until a point $\boldsymbol{x}^p$ is found such that $\boldsymbol{x}^p\ \ \boldsymbol{x}^{p\ 1}$ [Mil2000].

The unconstrained optimization problem will not be discussed further, however some of the algorithms that will be discussed in Chapter 5 may also be used to solve unconstrained problems.

2.5 Constrained Optimization

This section deals with mathematical programming problems of the form (2.2.3). These problems are called constrained optimization problems because the objective is to find the optimum for $f(\boldsymbol{x})$ while satisfying the constraints (2.2.2). This is an important class of problems as most real world optimization problems are, in fact, constrained problems [CS1970]. The conditions for optimality for constrained problems will now be presented in some detail. It is important to note that for this section and the remainder of this dissertation only minimization problems will be considered. It is a simple procedure to convert a minimization problem to a maximization problem and vice versa. Simply multiply the objective function of a minimization problem by -1 converts it into a maximization problem and vice versa.

In this section the necessary conditions are derived, which $\boldsymbol{x}$ must satisfy if $f(\boldsymbol{x})$ takes on a local minimum at $\boldsymbol{x}$ subject to $g(\boldsymbol{x}) \le \boldsymbol{b}$ and $\boldsymbol{x} \ge 0$.

2.5.1 Equality Constraints – Lagrange Multipliers

Lagrangian multipliers can be used to solve non-linear programming in which all the constraints are equality constraints. Equality constraints are equations of the form

$$g_i(x_1, \ldots, x_n) = b_i, \qquad i = 1, \ldots, p.$$

The concept of Lagrangian multipliers is introduced here as it is used in numerous sections of this dissertation [Win1995].

Consider the problem

(2.5.1)
$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) \\ &\textit{Subject to} \quad g_i(\boldsymbol{x}) = b_i \quad i = 1, \ldots, m \end{aligned}$$

The method of Lagrange multipliers gives a set of necessary conditions to identify optimal points of equality constrained optimization problems. To solve (2.5.1) multipliers λ_i are associated with the i^{th} constraint in (2.5.1) and form the *Lagrangian function*

(2.5.2) $$L(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m) = f(x_1, \ldots, x_n) \quad \sum_{i=1}^{m} \lambda_i (b_i \quad g_i(x_1, \ldots, x_n)).$$

We now attempt to find a point $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ that minimizes $L(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m)$. Since (2.5.1) is a minimization problem, if $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ minimizes $L(\boldsymbol{x}, \boldsymbol{\lambda})$ then at the point $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$

$$\frac{\partial L}{\partial \lambda_i} = b_i \quad g_i(\tilde{\boldsymbol{x}}) = 0$$

this shows that $\tilde{\boldsymbol{x}} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ will satisfy the constraints in (2.5.1). To show that $(\tilde{x}_1, \ldots, \tilde{x}_n)$ solves (2.5.1), let $(\hat{x}_1, \ldots, \hat{x}_n)$ be any point that is in the feasible region of (2.5.1). Since $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ minimizes $L(\boldsymbol{x}, \boldsymbol{\lambda})$ for any numbers $(\hat{\lambda}_1, \ldots, \hat{\lambda}_n)$

(2.5.3) $$L(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m) \le L(\hat{x}_1, \ldots, \hat{x}_n, \hat{\lambda}_1, \ldots, \hat{\lambda}_n).$$

Since $(\tilde{x}_1, \ldots, \tilde{x}_n)$ and $(\hat{x}_1, \ldots, \hat{x}_n)$ are both feasible in (2.5.1), the terms in (2.5.2) involving λ's are all zero and (2.5.3) becomes $L(\tilde{x}_1, \ldots, \tilde{x}_n) \le L(\hat{x}_1, \ldots, \hat{x}_n)$. Thus, $(\tilde{x}_1, \ldots, \tilde{x}_n)$ solves (2.5.1). If$(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ solves the unconstrained minimization problem

(2.5.4) $$\textit{Minimize} \quad L(\boldsymbol{x}, \boldsymbol{\lambda})$$

then $(\tilde{x}_1, \ldots, \tilde{x}_n)$ solves (2.5.1). For $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ to solve (2.5.4) it is necessary that at $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$,

(2.5.5) $$\frac{\partial L}{\partial x_1} = \ldots = \frac{\partial L}{\partial x_n} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \lambda_1} = \ldots = \frac{\partial L}{\partial \lambda_m} = 0.$$

Thus if $f(\boldsymbol{x})$ is a convex function and each $g_i(\boldsymbol{x})$ is a linear function, then any point $(\tilde{x}_1, \ldots, \tilde{x}_n, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_m)$ that satisfies (2.5.5) will give an optimal solution $(\tilde{x}_1, \ldots, \tilde{x}_n)$ to (2.5.1).

Using the Lagrange multipliers allows systems of equations to be solved without having to rewrite some variables in terms of others and it is unnecessary to take account of the fact that the variables may not all be independent.

2.5.2 Inequality Constraints - Kuhn-Tucker Conditions

In this section Kuhn-Tucker conditions also known as the Karush-Kuhn-Tucker conditions will be investigated. The Kuhn-Tucker conditions, is a set of necessary and sufficient conditions for identifying stationary points of a non-linear programming problem subject to inequality constraints [CS1970]. The development of the Kuhn-Tucker conditions is based on the Lagrange multiplier method [Tah1997] described earlier.

Consider the minimization problem of the form

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) \\ &\textit{Subject to} \quad g_i(\boldsymbol{x}) \le b_i \quad i = 1, \ldots, m \\ &\qquad\qquad x_j \ge 0 \quad j = 1, \ldots, n \end{aligned} \tag{2.5.6}$$

Any inequalities can be converted to equalities by the addition of either non-negative *slack* or *surplus* variables as follows. The inequalities

$$g_i(\boldsymbol{x}) \le b_i \quad \text{for } i = 1, \ldots, q$$

can be written as

$$g_i(\boldsymbol{x}) \quad x_{n\ i} = b_i \qquad x_{n\ i} \ge 0 \quad \text{for } i = 1, \ldots, q$$

where $x_{n\ i}$ is referred to as a slack variable. Similarly the inequalities

$$g_i(\boldsymbol{x}) \ge b_i \quad \text{for } i = q+1, \ldots, r$$

can be written as

$$g_i(\boldsymbol{x}) \quad x_{n\ i} = b_i \qquad x_{n\ i} \ge 0 \quad \text{for } i = q+1, \ldots, r$$

where $x_{n\ i}$ is called a surplus variable [CS1970]. The addition of the slack or surplus variables does not in any way change the set of possible solutions to the original problem.

To obtain the Kuhn-Tucker conditions the inequality constraints in (2.5.6) are first converted to equalities by adding the slack variables $s_i^2 \ (\ge 0)$ and $t_j^2 \ (\ge 0)$ as follows

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) \\ &\textit{Subject to} \quad g_i(\boldsymbol{x}) \quad s_i^2 = b_i \quad i = 1, \ldots, m \\ &\qquad\qquad x_j \quad t_j^2 = 0 \quad j = 1, \ldots, n. \end{aligned} \tag{2.5.7}$$

Let $\boldsymbol{s} = (s_1, s_2, \ldots, s_m)'$ and $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)'$. Associating multipliers $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)'$ with the non-negativity constraints the Lagrangian function becomes

(2.5.8) $$L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{s},\boldsymbol{\mu},\boldsymbol{t}) = f(\boldsymbol{x}) \quad \sum_{i=1}^{m} \lambda_i (b_i \quad g_i(\boldsymbol{x}) \quad s_i^2) \quad \sum_{j=1}^{n} \mu_j (x_j \quad t_j^2).$$

The first order conditions require $L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{s},\boldsymbol{\mu},\boldsymbol{t}) = 0$. Taking the partial derivative of L with respect to $\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{s},\boldsymbol{\mu}$ and $\boldsymbol{t}$, the Kuhn-Tucker conditions necessary for $\boldsymbol{x},\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ to be a stationary point of the minimization problem can be summarized as follows.

(2.5.9) $$\frac{\partial L}{\partial \boldsymbol{x}} = \frac{\partial f(\boldsymbol{x})}{\partial x_j} \quad \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \quad \mu_j = 0 \quad j = 1,\ldots,n$$

(2.5.10) $$\frac{\partial L}{\partial \boldsymbol{\lambda}} = \quad (b_i \quad g_i(\boldsymbol{x}) \quad s_i^2) = 0 \quad i = 1,2,\ldots,m$$

(2.5.11) $$\frac{\partial L}{\partial \boldsymbol{s}} = \quad 2\lambda_i s_i = 0 \quad i = 1,2,\ldots,m$$

(2.5.12) $$\frac{\partial L}{\partial \boldsymbol{\mu}} = \quad (x_j \quad t_j^2) = 0 \quad j = 1,2,\ldots,n$$

(2.5.13) $$\frac{\partial L}{\partial \boldsymbol{t}} = \quad 2\mu_j t_j = 0 \quad j = 1,2,\ldots,n$$

From equation (2.5.11) $\lambda_i s_i = 0$, that is either $\lambda_i = 0$ or $s_i = 0$ or both. From equation (2.5.10) if $\lambda_i > 0$ then $s_i = 0$ and it follows that $g_i(\boldsymbol{x}) = b_i$. If $\lambda_i = 0$ then $|s_i| \geq 0$ and it follows that $g_i(\boldsymbol{x}) \leq b_i$. Therefore from (2.5.10) and (2.5.11) $\lambda_i (b_i \quad g_i(\boldsymbol{x})) = 0$ and $g_i(\boldsymbol{x}) \leq b_i$.

From equation (2.5.13) $\mu_j t_j = 0$, that is either $\mu_j = 0$ or $t_j = 0$ or both. From equation (2.5.12) if $\mu_j > 0$ then $t_j = 0$ and it follows that $x_j = 0$. If $\mu_j = 0$ then $|t_j| \geq 0$ and it follows that $x_j \geq 0$. Therefore from (2.5.12) and (2.5.13) $\mu_j x_j = 0$ and $x_j \geq 0$.

For a minimization problem the Kuhn-Tucker conditions can now be summarized as follows.

(2.5.14)
$$\begin{aligned} \frac{\partial f(\boldsymbol{x})}{\partial x_j} \quad \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \quad \mu_j = 0 \quad & j = 1,\ldots,n \\ \lambda_i [b_i \quad g_i(\boldsymbol{x})] = 0 \quad & i = 1,\ldots,m \\ g_i(\boldsymbol{x}) \leq b_i & \\ \mu_j x_j = 0 & \\ \boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu} \geq 0. & \end{aligned}$$

For a maximization problem the equation $\frac{\partial f(\boldsymbol{x})}{\partial x_j} \quad \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \quad \mu_j = 0$ becomes:

$$\frac{\partial f(\boldsymbol{x})}{\partial x_j} \quad \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \quad \mu_j = 0$$

The following is an example of The Kuhn-Tucker conditions applied to a quadratic minimization problem.

$$\textit{Minimize} \quad f(\boldsymbol{x}) = x_1^2 \quad x_2^2 \quad x_3^2$$

$$\textit{Subject to} \quad g_1(\boldsymbol{x}) = 2x_1 \quad x_2 \leq 5$$
$$g_2(\boldsymbol{x}) = x_1 \quad x_3 \leq 2$$
$$g_3(\boldsymbol{x}) = \quad x_1 \leq \quad 1$$
$$g_4(\boldsymbol{x}) = \quad x_2 \leq \quad 2$$
$$g_5(\boldsymbol{x}) = \quad x_3 \leq 0$$
$$\boldsymbol{x} \geq 0$$

Applying the Kuhn-Tucker conditions.

$$2x_1 \quad 2\lambda_1 \quad \lambda_2 \quad \lambda_3 \quad \mu_1 = 0$$
$$2x_2 \quad \lambda_1 \quad \lambda_4 \quad \mu_2 = 0$$
$$2x_3 \quad \lambda_2 \quad \lambda_5 \quad \mu_3 = 0$$
$$\lambda_1(5 \quad (2x_1 \quad x_2)) = 0$$
$$\lambda_2(2 \quad (x_1 \quad x_3)) = 0$$
$$\lambda_3(\quad 1 \quad x_1) = 0$$
$$\lambda_4(\quad 2 \quad x_2) = 0$$
$$\lambda_5(\quad x_3) = 0$$
$$2x_1 \quad x_2 \leq 5$$
$$x_1 \quad x_3 \leq 2$$
$$\mu_1 x_1 = 0$$
$$\mu_2 x_2 = 0$$
$$\mu_3 x_3 = 0$$
$$x_1, x_2, x_3 \geq 0$$
$$\mu_1, \mu_2, \mu_3 \geq 0$$
$$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5 \geq 0$$ [Tah1997].

2.5.3 Convexity and Concavity

As explained earlier (Sections 2.3.2 and 2.3.5) knowing if the function is convex/concave establishes the fact that there is only one global minimum/maximum point. For a constrained problem it needs to be determined whether the objective function is convex and if the set of constraints defines a convex set.

Figure 2.7 illustrates the concept of convexity for a minimization problem with linear constraints and a non-linear convex objective function. The objective function is represented using contour lines. The minimum of the objective function is at the point $\boldsymbol{x}_0$. At the point $\boldsymbol{x}_F$ the problem has a local minimum point while satisfying the constraints. It is obvious from the figure that $\boldsymbol{x}_F$ is also the global minimum point.

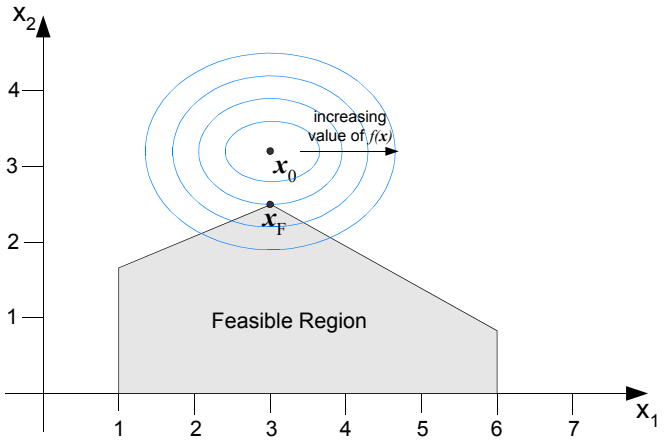

Figure 2.7 Convex objective function

Chapter 3 will discuss in detail some algorithms available for solving a convex optimization problem.

2.5.4 Non-convex Objective Function

In a problem which has a non-convex objective function the solution to the optimization problem may result in local optimal solutions which have a different value than the global optimal solution [BR2001]. In Figure 2.8, the points $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ are both unconstrained local minimum points of the objective function. Each local minimal solution has a different objective function value. The points $\boldsymbol{x}_{F1}$ and $\boldsymbol{x}_{F2}$ are both local minimum points that satisfy the constraints. It is now possible that either $\boldsymbol{x}_{F1}$ or $\boldsymbol{x}_{F2}$ or both of these points are global minimum points. In the figure $\boldsymbol{x}_{F1}$ represents the global minimum point.

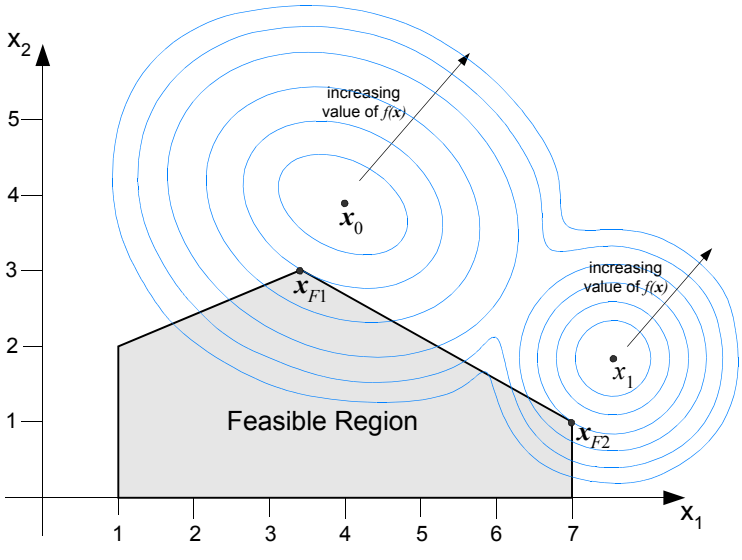

Figure 2.8 Non-convex objective function

Clearly minimization/maximization problems with a convex/concave objective function are considerably easier to solve than problems for which it is not known if the function is convex/concave.

Non-convex problems are difficult to solve and often a global minimum cannot be found, which makes convex problems much more desirable. Algorithm attempting to find the global minimum for non-convex problems must have the ability to distinguish between points which satisfy the minimality conditions, but are local minimum points are points which are the global minimal solutions [BR2001]. Often the algorithms can only obtain the local minimum point, however any improvement is usually better than none at all.

2.6 Classification of a Constrained Optimization Problem

The general mathematical programming problem has been discussed in Section 2.2. Various programming problems have been classified according to the type of the objective function and constraints. This section introduces a selection of optimization problems required in the discussion of the algorithms considered in the following chapters.

2.6.1 Linear Programming

A Linear Programming Problem is one where both the objective function and the constraints are linear functions of the decision variables. Linear programming is the most common way of formulating a vast number of mathematical programming problems. It was introduced during the 1940s in the United States in response to problems arising during World War II. In 1947 George Dantzig developed an efficient method for solving linear programming problems now known as the Simplex Method, which initiated widespread research into the development of new and more sophisticated algorithms. Due to the growth of more accurate and efficient algorithms, linear programming has proved valuable for modeling many and diverse types of problems in planning, routing, assignment, and design. Industries that make use of linear programming and its extensions include transportation, telecommunications, and manufacturing of many kinds [Fou2001].

Many real world problems can be represented very efficiently using linear programming. Formulating a problem using a linear model is often much easier than a non-linear model. Even if the objective function is not entirely linear, it is often possible to approximate it by a linear function. Thus, due to its simplicity linear programming if often selected above more complex forms. Mathematically a linear programming problem is represented by

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = c_1 x_1 \quad c_2 x_2 \quad \ldots \quad c_n x_n = \boldsymbol{c}' \boldsymbol{x} \\ &\textit{Subject to} && a_{11} x_1 \quad a_{12} x_2 \quad \ldots \quad a_{1n} x_n = b_1 \\ & && a_{21} x_1 \quad a_{22} x_2 \quad \ldots \quad a_{2n} x_n = b_2 \\ & && \qquad\qquad\qquad\qquad\quad = \\ & && a_{m1} x_1 \quad a_{m2} x_2 \quad \ldots \quad a_{mn} x_n = b_m \\ & && x_1, x_2, \ldots, x_n \geq 0 \end{aligned}$$

and can be written using matrix notation as

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = \boldsymbol{c}' \boldsymbol{x} \\ &\textit{Subject to} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & && \boldsymbol{x} \geq 0. \end{aligned}$$

where $\boldsymbol{A}$ is an m x n matrix, $\boldsymbol{b}$ an m element column vector and $\boldsymbol{c}$ and $\boldsymbol{x}$ being n element column vectors.

There are however, still many situations for which linear models are inadequate. Problems like these are modeled using non-linear equations as will be described in the Section 2.6.2 and 2.6.3.

2.6.2 Non-Linear Programming

A non-linear programming problem is one in which the objective function and/or its constraints are non-linear functions of the decision variables. Some examples of systems that cannot be described by linear models are as follows:

Systems with multiple operating points like chemical reactions and buckling beams;

Systems with periodic variations like a digital clock circuit and heart and nerve impulses;

Systems sensitive to initial conditions like the dynamics of population, climatic models and fluid flow models referred to as chaotic or complex dynamics [Sas1999];

Fixed charge problems, etc.

A particularly well studied non-linear programming problem is one where all the constraint equations are linear and the objective function is quadratic. This problem is called the quadratic programming (QP) problem. It is the quadratic programming problem and the algorithms for its solution that form the focus of this dissertation and will now be discussed in detail.

2.6.3 Quadratic Programming

A quadratic programming problem can be thought of as a generalization of a linear programming problem, or as a restricted case of a nonlinear problem. It is a problem in which all of the constraints of (2.2.3) are linear and therefore form a convex set, but the objective function is non-linear. In particular the objective function is a polynomial of the second degree.

The quadratic programming problem is formulated as follows:

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad\quad \boldsymbol{x} \ge 0 \end{aligned} \tag{2.6.1}$$

where $\boldsymbol{c}$ is a n element column vector, $\boldsymbol{b}$ is a m element column vector and $\boldsymbol{A}$ is a m x n matrix [Zan1969]. The expression $\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}$ is called the *quadratic form* and was discussed in Section 2.3.3. It was also shown, that $\boldsymbol{Q}$ is a symmetric n x n matrix in equation (2.3.2) in Section 2.3.3. The objective function is a sum of a linear term $\boldsymbol{c}'\boldsymbol{x}$ and a quadratic form

$x'Qx$. This problem is also referred to as the *primal problem*. The minimization problem is sometimes also written in the form

$$\textit{Minimize} \quad f(x) = c'x \quad \frac{1}{2}x'Qx$$

where the factor $\frac{1}{2}$ has simply been factored out of the matrix Q. The constraints remain unchanged. Any maximization problem can also be written as a minimization problem by multiplying the objective function by -1.

As already mentioned convexity plays an important role in determining the minimum point for convex problems as they are considerably easier to solve than non-convex problems. The following theorem states the condition under which the objective function for a quadratic programming problem is convex. The concepts of positive definiteness and semi-definiteness needed for this theorem are given in Section 2.3.3.

Theorem 2.9

The function $f(x) = c'x \quad x'Qx$ is convex if and only if Q is positive semi-definite [CS1970].

Clearly for a concave function the matrix Q must be negative semi-definite or negative definite. It is also important to note, that since the set of constraints are linear the set of constraints form a convex set [Mil2000].

Consider now the Kuhn-Tucker conditions applied to a quadratic programming problem (2.6.1). From (2.5.14) the new system of equations is as follows

$$\begin{gathered} 2Qx \quad A'\lambda \quad \mu = \quad c \\ \lambda'(b \quad Ax) = 0 \\ Ax \le b \\ \mu' x = 0 \\ x \ge 0, \ \mu \ge 0 \text{ and } \lambda \ge 0. \end{gathered}$$

Adding the slack variables y to the inequality $Ax \le b$ yields the following system

$$\begin{gathered} 2Qx \quad A'\lambda \quad \mu = \quad c \\ Ax \quad y = b \\ \mu' x = 0 \quad \text{and} \quad \lambda' y = 0 \\ x, y, \mu \ge 0 \text{ and } \lambda \ge 0. \end{gathered} \tag{2.6.2}$$

The above system can be written as

(2.6.3)
$$\begin{bmatrix} 2\boldsymbol{Q} & \mathbf{0}_{n\,x\,m} & \boldsymbol{A}' & \boldsymbol{I}_n \\ \boldsymbol{A} & \boldsymbol{I}_m & \mathbf{0}_{m\,x\,m} & \mathbf{0}_{m\,x\,n} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{b} \end{bmatrix}$$

with $\mu' \boldsymbol{x} = 0$, $\lambda' \boldsymbol{y} = 0$, $\boldsymbol{x}, \boldsymbol{y}, \mu \geq 0$ and $\lambda \geq 0$ [CS1970].

The Kuhn-Tucker conditions provide only a set of necessary conditions for a minimum. However, if $f(\boldsymbol{x})$ is a convex function and the constraints form a convex set, then any point $\boldsymbol{x}^*$ satisfying the conditions as defined in (2.6.3) is a global minimum solution [Win1995]. The Kuhn-Tucker conditions also provide another important result. Applying the conditions to the quadratic programming problem, reformulates the non-linear problem into a problem with the first two equations of (2.6.2) linear and non-linear equations $\mu' \boldsymbol{x} = 0$ and $\lambda' \boldsymbol{y} = 0$. This suggests the possibility of using algorithms for solving linear programming to solve the quadratic programming problem. Usually only minor modifications are necessary. Due to these results the the Kuhn-Tucker conditions are important for solving quadratic programming problems and are the basis for most of the algorithms that will be shown in Chapter 3 and 4.

Associated with each programming problem there is a corresponding dual problem [Mil2000]. Both of these problems are constructed from the same costs and coefficients but in such a way that if one of the problems is a minimization problem the other one is a maximization problem. Also the number of variables in the primal problem is equal to the number of constraints in the dual. When taking the dual of a programming problem, the original problem is referred to as a primal problem. The associated dual problem is defined as follows with $\boldsymbol{u}$ as unknown variables and φ as the objective function of the dual problem in terms of $\boldsymbol{x}$ and $\boldsymbol{u}$.

$$\begin{aligned} &\textit{Maximize} \quad \varphi(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{b}' \boldsymbol{u} \quad \boldsymbol{x}' \boldsymbol{Q} \boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}' \boldsymbol{u} \quad \boldsymbol{Q} \boldsymbol{x} \geq \boldsymbol{c} \\ &\qquad\qquad\qquad\quad \boldsymbol{u}, \boldsymbol{x} \geq 0. \end{aligned}$$

Calculating the solution to the dual problem may improve our understanding of some of the algorithms and can provide some computational advantage. However, due to the lack of symmetry, dual problems are not as important in quadratic programming as in linear programming [KKO1966] therefore, only the primal problem will be considered in the explanation of the algorithms.

2.7 Summary

This chapter introduced some of the basic mathematical concepts necessary for the remainder of this dissertation. The concept of identifying the maxima and minima of unconstrained and constrained problems was described. An important result was established with regard to convex and concave functions. If it is known that a function is convex/concave then any local minimum/maximum point obtained is the global minimum/maximum point. However, if it cannot be said that a function is convex or concave then the global optimum point may never be established.

Another important concept discussed in this chapter, were the Kuhn-Tucker conditions. These conditions provide a set of necessary conditions for optimality and sufficient conditions for optimality provided that the objective function is convex. These conditions also provide the basis for many of the algorithms to be discussed in the following chapters. The latter is due to the fact that by applying the Kuhn-Tucker conditions to the quadratic programming problem, produces a problem with a set of linear equations and the non-linear equations $\boldsymbol{\mu}' \boldsymbol{x} = 0$ and $\boldsymbol{\lambda}' \boldsymbol{y} = 0$. This allows quadratic programming problems to be solved by making use of modified linear programming algorithms.

This chapter also defined the general optimization problem and has classified the optimization problem according to the type of its objective function and constraints. Special cases have been identified for further discussion namely linear programming and non-linear programming. A further special case of the non-linear programming problems called quadratic programming problem, was discussed, which forms the focus of this dissertation.

In the following three chapters, an overview of selected algorithms for solving the quadratic programming problem under varying circumstances will be given. In Chapter 3 the algorithms for convex quadratic programming problems will be given. In Chapter 4 algorithms for the non-convex case will be discussed. Chapter 5 will give a brief overview of some algorithms for solving the general non-linear programming problems.

Chapter 3

Algorithms for the Convex Quadratic Programming Problem

3.1 Introduction

As discussed in Section 2.3.2, if a local minimum is found for a convex problem then it is also the global minimum. The same can be said of a maximization problem with a concave objective function. An example of a convex problem is illustrated in Figure 3.1. Figure 3.1 represents the function $z = 0.2(3x^2 \quad 3y^2)$.

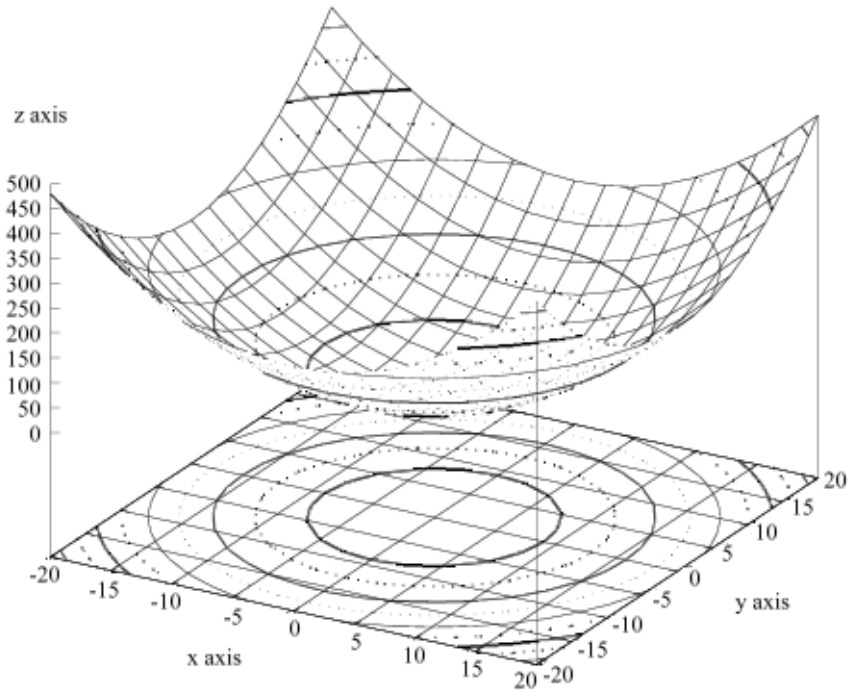

Figure 3.1 Contours of a convex objective function of a minimization problem.

Knowing that the function is convex or concave makes quadratic programming problems considerably easier to solve, as compared to problems for which the convexity of objective function is not known. In this chapter a selection of algorithms for solving quadratic programming problems with a convex objective function will be introduced. The factors that determined the selection of algorithms were its prevalence and the availability of research

material. These algorithms will be discussed in detail and an illustrative example will be presented in order to aid in their understanding. The same numerical example will be used throughout this chapter in order for the algorithms to be easily compared with one another.

Quadratic programming algorithms can be classified into three categories namely Active Set Methods, Ellipsoid Methods and Interior Point Methods, each of which will be described in more detail in this chapter.

Active Set methods keeps track of those constraints which are active, that is, constraints for which strict equality holds at a local minimum point. The idea of the method is to add active constraints to an active set provided certain conditions hold. If the conditions are not satisfied the corresponding constraint is dropped form the set. The simplex method mentioned in Section 2.6.1 can also be thought of as an active set method. It keeps track of active and inactive variables. The active set method searches for the solution on the boundary of the feasible region;

The Ellipsoid methods construct a sequence of ellipses and then checks if the center point is feasible. If it is not, a smaller ellipse is constructed. This process is repeated until an optimal solution has been found; and

Interior-point methods move through the interior of the feasible region towards the optimum rather than along the boundary as is the case with active set methods.

The following definitions will now be introduced and used throughout the remainder of the dissertation. A *basis* is a set of linearly independent variables. The variables that are in the basis are called *basic variables* and variables that are not in the basis are called *non-basic variables*. For a problem with m equations and n unknowns, a *basic solution* is obtained by setting $n \quad m$ variables equal to zero and solving for the remaining m variables. This will generally be only possible if the m equations are linearly independent.

3.2 Active Set Methods – The Simplex Method

The Simplex Method is a systematic procedure for solving mainly linear programming problems. The method moves from one extreme point to another, with a smaller (or at least not worse) value for the objective function $f(\boldsymbol{x})$. At each iteration, some of the inequality constraints are forced to be temporarily *binding*, which means that the left-hand side of the

constraint is equal to the right-hand side of the constraint. The indices of these constraints are referred to as the *active set*. To obtain the solution for a problem, the first step of the algorithm is to obtain an initial feasible solution and an active set containing the constraints satisfying that feasible solution. The feasible solution is then checked for optimality. If the feasible solution is not the optimal solution (in this case minimum), then an optimal solution can still be reached, or it can be shown that the objective function can be made arbitrarily small, by changing the constraints in the active set one at a time. The active set is changed by removing one of the constraints and selecting another constraint to come into the active set provided that the constraint will become binding [Had1962]. Since each change of a constraint in an active set decreases the value of the objective function, no active set can be repeated. If an active set should repeat, then the objective function would not decrease, which results in a contradiction. As no active set can ever be repeated, the solution will be obtained in a finite number of steps [BR2001].

It has been shown that the simplex method is an exponential time algorithm. An n-dimensional feasible region has 2^n vertices. In the worst case of the simplex method, all the vertices will be visited and thus the optimum will be found in exponential time [Win1995].

Although the simplex method was initially developed to solve linear programming problems, there are now many quadratic programming algorithms that make use of the simplex method. For reference purposes the simplex method is provided in Appendix B.1. The following sections introduce a selection of algorithms for solving quadratic programming problems that make use of the simplex method.

3.2.1 Wolfe's Algorithm

Although it is not the first method that was developed for solving quadratic programming problems, it is one of the most widely used methods and was developed by P. Wolfe in 1959 [Wol1959]. Wolfe's algorithm can be directly applied to solve any quadratic programming problem of the form

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\text{Subject to} && \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & && \boldsymbol{x} \ge 0. \end{aligned} \tag{3.2.1}$$

where $\boldsymbol{Q}$ is a $n \times n$ symmetric matrix, $\boldsymbol{A}$ a $m \times n$ matrix, $\boldsymbol{c}$ a n-vector and $\boldsymbol{b}$ a m-vector. If the objective function is convex, then any point satisfying the Kuhn-Tucker conditions (Section 2.6.3 on page 30)

$$\begin{aligned} 2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} &= \quad \boldsymbol{c} \\ \boldsymbol{A}\boldsymbol{x} \quad \boldsymbol{y} &= \boldsymbol{b} \\ \boldsymbol{x} \geq 0, \boldsymbol{\mu} \geq 0, \boldsymbol{y} \geq 0, \ \boldsymbol{\lambda} &\geq 0 \end{aligned} \tag{3.2.2}$$

and

$$\boldsymbol{\mu}'\boldsymbol{x} = 0 \quad \text{and} \quad \boldsymbol{\lambda}'\boldsymbol{y} = 0 \tag{3.2.3}$$

will solve the problem (3.2.1).

To find a point satisfying the Kuhn-Tucker conditions Wolfe's algorithm proposes a modified version of Phase One of the Two-phase simplex method, applied to the linear equations and the non-negativity constraints (3.2.2) (See Appendix B.2 for details regarding the Two-phase simplex method). Essentially the method adds artificial variables to all the constraints and then attempts to minimize the sum of the artificial variables. To ensure that the final solution also satisfies (3.2.3). Wolfe's method modifies the simplex method as follows:

No pivot is performed that would make μ_p and x_p both basic variables; and

No pivot is performed that would make λ_p and y_p both basic variables.

A number of implementations of these ideas have been presented by various authors ([KKO1966] [Sim1975] [Win1995]). The implementation proposed by Künzi et al. [KKO1966] will be discussed here. Their approach begins with the Kuhn-Tucker conditions (3.2.2) and (3.2.3). It is also assumed without loss of any generality that $b_i \geq 0$ for all i. If any of the b_i's should be negative then those equations can be written such that all $b_i \geq 0$, by multiplying the appropriate equations by -1, as shown in Appendix B.1. In order to determine an initial feasible solution, artificial variables are added to the system and then the simplex method is used to remove these variables subject to the conditions (3.2.3).

Two forms of Wolfe's algorithm are presented, namely a short form and a long form. While the long form works without any restrictions the short form requires that either $\boldsymbol{c} = \boldsymbol{0}$ or that the matrix $\boldsymbol{Q}$ is positive definite [Wol1959]. This restriction will be discussed at a later stage in this section.

Short Form

In order to obtain an initial basic feasible solution, $m \quad 2n$ artificial variables

$$\boldsymbol{s}=(s_1,\ldots,s_m)'$$
$$\boldsymbol{w}^1=(w_1^1,\ldots,w_n^1)'$$
$$\boldsymbol{w}^2=(w_1^2,\ldots,w_n^2)'$$

are added to the linear system (3.2.2) and the system (3.2.2) is expanded to

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} \quad \boldsymbol{w}^1 \quad \boldsymbol{w}^2 = \quad \boldsymbol{c}$$
(3.2.4)
$$\boldsymbol{A}\boldsymbol{x} \quad \boldsymbol{y} \quad \boldsymbol{s}=\boldsymbol{b}$$
$$\boldsymbol{x}\geq 0,\boldsymbol{\mu}\geq 0,\boldsymbol{\lambda}\geq 0,\boldsymbol{y}\geq 0,\boldsymbol{s}\geq 0,\boldsymbol{w}^1\geq 0,\boldsymbol{w}^2\geq 0.$$

Written in matrix notation

(3.2.5)
$$\begin{bmatrix} 2\boldsymbol{Q}_{(n\times n)} & \boldsymbol{0}_{(n\times m)} & \boldsymbol{A}'_{(n\times m)} & \boldsymbol{I}_n & \boldsymbol{I}_n & \boldsymbol{I}_n & \boldsymbol{0}_{(n\times m)} \\ \boldsymbol{A}_{(m\times n)} & \boldsymbol{I}_{(m\times m)} & \boldsymbol{0}_{(m\times m)} & \boldsymbol{0}_{(m\times n)} & \boldsymbol{0}_{(n\times n)} & \boldsymbol{0}_{(n\times n)} & \boldsymbol{I}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu} \\ \boldsymbol{w}^1 \\ \boldsymbol{w}^2 \\ \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{b} \end{bmatrix}$$

$$\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{y},\boldsymbol{w}^1,\boldsymbol{w}^2,\boldsymbol{s}\geq 0.$$

A basic feasible solution for problem (3.2.5) that satisfies (3.2.3) and contains at most $m \quad n$ possibly non-zero variable values can be given immediately by setting

$$\boldsymbol{x}=0,\boldsymbol{\lambda}=0,\boldsymbol{\mu}=0,\boldsymbol{y}=0$$

and for each j at least one of the two variables w_j^1 or w_j^2 is zero. Then the first basis contains the variables $s_i=b_i, i=1,\ldots,m$ and for each j, only one of the variables w_j^1 or w_j^2 is selected as follows

$$w_j^1= \quad c_j \quad \text{if} \quad c_j<0 \quad \text{and then} \quad w_j^2=0 \quad \text{or}$$
$$w_j^2=c_j \quad \text{if} \quad c_j\geq 0 \quad \text{and then} \quad w_j^1=0.$$

If $c_j=0$ then any one of w_j^1 or w_j^2 can be taken into the basis. As in Phase One of the simplex method (see Appendix B.1) the basic solution is represented in a tableau. The initial tableau is represented in Table 3.1 as follows:

Table 3.1 Initial Simplex Tableau

			α_1	.	.	.	.	.	.	.	.	α_j	.	.	.	.	.	.	.	α_N
Coeff. B.V.	Basic Variables	$\boldsymbol{x}_B$	x_1	...	x_n	y_1	...	y_m	λ_1	...	λ_m	μ_1	...	μ_n	w_1^t	...	w_n^t	s_1	...	s_m
$\tilde{c}_{B1}$	w_1^t	x_{B1}	α_{11}	.	.	.	.	.	.	.	.	α_{1j}	.	.	$(\ 1)^{t\ 1}$	0	0	0	.	$0=\alpha_{1N}$
			.									.			0		0			0
	w_n^t		.									.			0	0	$(\ 1)^{t\ 1}$	0	.	0
$\tilde{c}_{Bi}$	s_1		α_{i1}	.	.	.	.	.	.	.	.	α_{ij}	.	.	.	.	.	1	0	$0=\alpha_{iN}$
			.									.						0		0
$\tilde{c}_{BM}$	s_m	x_{BM}	α_{M1}	.	.	.	.	.	.	.	.	α_{Mj}	.	.	.	.	.	0	0	$1=\alpha_{MN}$
		z	$z_1\ \hat{c}_1$	.	.	.	.	.	.	.	.	$z_j\ \hat{c}_j$	.	.	.	.	.	.	.	$z_N\ \hat{c}_N$

In the tableau $M = n \quad m$ and $N = 3n \quad 3m$. The first column of the tableau gives the coefficients of the basic variables (Coeff. B.V.) corresponding to the variables in the basis (these are either 0 or 1 as explained in Appendix B.2). The second column gives the variables currently in the basis. The third column $\boldsymbol{x}_B$ gives the current value of the variables in the basis. The columns $\alpha_j, j = 1, \ldots, (2n \quad 2m)$ are vectors listing all real variables. The columns $\alpha_j, j = (2n \quad 2m \quad 1), \ldots, N$, list the artificial variables, where $t \quad \{1,2\}$. The entries α_{ij} are the coefficients of the variables in the left hand side of the linear equations in (3.2.4). The z in the bottom row of column three is calculated by $z = \sum_{i=1}^{M} (\tilde{c}_{Bi})' \quad x_{Bi}$ and for each column α_j, $z_j \quad \hat{c}_j = \sum_{i=1}^{M} (\tilde{c}_{Bi})' \quad \alpha_{ij} \quad \hat{c}_j, \; j = 1, \ldots, N$, where $\hat{c}_j = 1$ for the artificial variables and zero for the remaining variables.

The artificial variables will now be removed from the basis which will transform (3.2.5) back to (3.2.2). A new tableau is constructed at each iteration. This is achieved in two phases (not to be confused with the phases of the Two-phase simplex method). In the first phase (of the short form) the simplex method is used to minimize the objective function

$$f(\boldsymbol{s}) = \sum_{i=1}^{m} \tilde{c}_{Bi} s_i \quad \text{where } \tilde{c}_{Bi} = 1 \text{ for } i = 1, \ldots, m$$

subject to the constraints (3.2.4) with $\boldsymbol{\lambda} = \boldsymbol{0}$ and $\boldsymbol{\mu} = \boldsymbol{0}$, that is, λ_i and μ_j stay out of the basis ($\tilde{c}_{Bi} = 1$ for the artificial variables s_i in the basis and 0 for all other variables). It is assumed that there is no degeneracy, since if this is not the case the procedure continues until no s_i appear in the basis, even if $\sum_{i=1}^{m} s_i$ has a zero value.

The following table gives the modifications in the steps of the simplex method using the above tableau.

Step 1	Determine if there exists a $z_j \quad \hat{c}_j > 0$ in the simplex tableau. If all the $z_j \quad \hat{c}_j \leq 0$ then the current solution is minimal.
Step 2	If one or more $z_j \quad \hat{c}_j > 0$, then the solution is not minimal. Determine if every $z_j \quad \hat{c}_j$ has a positive number in its column α_j. If all $\alpha_{ij} \leq 0$ for those columns considered in a particular phase, then there is an unbounded solution.
Step 3	Select any p such that $z_p \quad \hat{c}_p = max(z_j \quad \hat{c}_j), \quad i \quad \alpha_{ij} > 0$. Thus the variable in column α_p can now enter the basis.

Step 4	In column α_p determine the row r of the smallest θ-ratio, such that $$\frac{x_{Br}}{\alpha_{rp}} = min\left\{\frac{x_{Bi}}{\alpha_{ip}}, \alpha_{ip} > 0\right\}.$$
Step 5	Pivot the tableau at the entry α_{rp} selected in the previous step. Thus column p replaces column r in the basis and the following transformation is performed: $$\hat{\alpha}_{ij} = \alpha_{ij} - \frac{\alpha_{rj}\alpha_{ip}}{\alpha_{rp}}, \quad i = 1,\ldots,I, \ j = 1,\ldots,J, \ i \neq r$$ $$\hat{\alpha}_{rj} = \frac{\alpha_{rj}}{\alpha_{rp}}$$ and the $\boldsymbol{x}_B$ can be similarly recalculated as $$\hat{x}_{Bi} = x_{Bi} - \frac{\alpha_{ip}x_{Br}}{\alpha_{rp}}, \quad i = 1,\ldots,I, \ j = 1,\ldots,J, \ i \neq r$$ $$\hat{x}_{Br} = \frac{x_{Br}}{\alpha_{rp}}$$
Step 6	When performing the transformation and one of the $x_{Bi} = s_i$ is replaced then $\tilde{c}_{Bi}$ becomes zero. Then recalculate the values of $z_j - \hat{c}_j$ and go to Step 1.

At the end of the first phase a basic solution has been reached, such that $\sum_{i=1}^{n} s_i = 0$. The basic solution contains $m - n$ of the variables x_j, y_i and w_j^t in the basis, that is, either w_j^1 or w_j^2, for each $i = 1,\ldots,m$ and $j = 1,\ldots,n$.

At this point the columns corresponding to s_i and to those w_j^t not in the basis can be removed from the tableau. Now an n-vector $\boldsymbol{w} = (w_1,\ldots,w_n)'$ is constructed from the remaining w_j^1 and w_j^2, where w_j is either w_j^1 or w_j^2 depending on which of the two is left in the basis at the end of the first phase. Now let $\boldsymbol{E}$ denote the coefficient matrix of the vector $\boldsymbol{w}$. $\boldsymbol{E}$ is a diagonal matrix with elements +1 or -1 depending on whether $w_j = w_j^1$ or $w_j = w_j^2$. Thus, at the end of fist phase we have a basic feasible solution to the tableau associated with the system

$$\begin{aligned} 2\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}'\boldsymbol{\lambda} - \boldsymbol{\mu} - \boldsymbol{E}\boldsymbol{w} &= \boldsymbol{c} \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} &= \boldsymbol{b} \\ \boldsymbol{x} \geq 0, \boldsymbol{\mu} \geq 0, \boldsymbol{w} \geq 0 \quad &\text{with} \quad \boldsymbol{\lambda} = \boldsymbol{0}, \boldsymbol{\mu} = \boldsymbol{0} \end{aligned} \tag{3.2.6}$$

satisfying the conditions (3.2.3).

In the second phase of the short form the simplex method (outlined in the above table) is used to minimize the objective function

$$f(\boldsymbol{w}) = \sum_{j=1}^{n} \tilde{c}_{Bj} w_j \quad \text{where } \tilde{c}_{Bj} = 1 \text{ for } j = 1, \ldots, n$$

subject to the constraints (3.2.6) and satisfying (3.2.3), that is, the following rules must be satisfied:

If the variable x_p is in the basis then in the transition to the next basic solution the corresponding μ_p cannot enter the basis. If the variable μ_p is already in the basis, then the corresponding x_p may not enter the basis (that is, $x_p \mu_p = 0$). However, if x_p is in the basis at zero level, μ_p may enter the basis only if x_p remains at zero level; and

If the variable λ_p should enter the basis then in the transition to the next basic solution the corresponding y_p cannot enter the basis. If the variable y_p should enter the basis, then the corresponding λ_p may not enter the basis (that is, $\lambda_p y_p = 0$). However, if y_p is in the basis at zero level, λ_p may enter the basis only if y_p remains at zero level.

If at the end of the second phase the minimum is found such that $\sum_{j=1}^{n} w_j = 0$, then a solution of (3.2.6) with $\boldsymbol{w} = \boldsymbol{0}$ has been found and therefore the solution for (3.2.2)-(3.2.3) and for the original problem (3.2.1) has been obtained.

It is possible that if no additional assumptions are made about $\boldsymbol{Q}$ and $\boldsymbol{c}$ then it may happen due to the condition (3.2.3), that no further iteration step can be made even though $\sum_{j=1}^{n} w_j > 0$. Further investigation into this situation showed that a solution would only be possible if either $\boldsymbol{c} = 0$ or the matrix $\boldsymbol{Q}$ is positive definite. The argument is based on the following theorem [KKO1966].

Let $\boldsymbol{x}_B$ denote the variables in the basis, the solution obtained at the end of the second phase with $\boldsymbol{x}_N$ corresponding to the zero components, that is $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ where $\boldsymbol{x}_N = 0$. Further, let $\boldsymbol{\mu}_B$ denote the components of $\boldsymbol{\mu}$ which are equal to zero and $\boldsymbol{\mu}_N$ denote those components of $\boldsymbol{\mu}$ which could be positive, that is $\boldsymbol{\mu} = (\boldsymbol{\mu}_B, \boldsymbol{\mu}_N)$ and $\boldsymbol{x}' \boldsymbol{\mu} = \boldsymbol{x}_B' \boldsymbol{\mu}_B \quad \boldsymbol{x}_N' \boldsymbol{\mu}_N = 0$. Let $\boldsymbol{k}$ be a n-vector, $\boldsymbol{q}$ and $\boldsymbol{d}$ both n-vectors having constant components and $\boldsymbol{R}$ an n x n matrix. All other symbols have the same meaning as before.

Theorem 3.1 [Wol1959]

Let $\hat{\boldsymbol{k}}$ represent the solution of the following linear problem:

Minimize $\boldsymbol{q}'\boldsymbol{k}$

subject to the constraints

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} \quad \boldsymbol{R}\boldsymbol{k} = \boldsymbol{d}$$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\boldsymbol{x} \geq 0, \boldsymbol{\mu} \geq 0, \boldsymbol{k} \geq 0, \boldsymbol{\mu}_B = 0, \boldsymbol{x}_N = 0.$$

Then there exists an n-vector $\boldsymbol{r}$ such that

$$\boldsymbol{Q}\boldsymbol{r} = \boldsymbol{0}, \quad \boldsymbol{A}\boldsymbol{r} = \boldsymbol{0} \text{ and } \boldsymbol{q}'\hat{\boldsymbol{k}} = \boldsymbol{d}'\boldsymbol{r}.$$

To apply Theorem 3.1 to the short form at the end of the second phase we set

$$\boldsymbol{k} = \boldsymbol{w}$$

$$\boldsymbol{q}' = (1, 1, \ldots, 1)$$

$$\boldsymbol{R} = \boldsymbol{E}, \quad \boldsymbol{d} = \quad \boldsymbol{c}.$$

The solution at the end of the second phase satisfies the hypothesis of Theorem 3.1 since due to the first rule $\boldsymbol{x}_N$ and $\boldsymbol{\mu}_B$ cannot be in the basis of the next simplex step and remain zero. Since no further decrease in $\sum w_j$ is possible, the final solution at the end of the second phase is also a solution of the linear programming problem formulated in Theorem 3.1. This implies that there exist an n-vector $\boldsymbol{r}$ with

$$\boldsymbol{Q}\boldsymbol{r} = \boldsymbol{0}$$

and

$$min \sum w_j = \quad \boldsymbol{c}'\boldsymbol{r}.$$

If $\boldsymbol{Q}$ is positive definite then

$$\boldsymbol{Q}\boldsymbol{r} = \boldsymbol{0} \qquad \boldsymbol{r} = \boldsymbol{0} \qquad min \sum w_j = 0.$$

If $\boldsymbol{c} = \boldsymbol{0}$ then it follows that

$$min \sum w_j = 0.$$

Thus in this special cases the short from will definitely lead to a solution of the original problem.

Long Form

The long form consists of three phases, the first two corresponding to the two phases of the short form. Thus the long form starts with the application of the short form with $\boldsymbol{c}$ being replaced by the zero vector, that is, the equation

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} \quad \boldsymbol{w}^1 \quad \boldsymbol{w}^2 = \quad \boldsymbol{c}$$

is replaced by

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} \quad \boldsymbol{w}^1 \quad \boldsymbol{w}^2 = \boldsymbol{0}.$$

Now the first and second phases are computed as before in the short form. At the end of the second phase the variable $\boldsymbol{w}$ will have been removed (in the case of degeneracy continue until all w_i are deleted). Thus we have a basic solution to the system

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} = \boldsymbol{0}$$
$$\boldsymbol{A}\boldsymbol{x} \quad \boldsymbol{y} = \boldsymbol{b}$$
$$\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{y} \geq 0.$$

which satisfies (3.2.3). This is equivalent to the system

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} \quad v\boldsymbol{c} = \boldsymbol{0}$$
$$\boldsymbol{A}\boldsymbol{x} \quad \boldsymbol{y} = \boldsymbol{b} \tag{3.2.7}$$
$$\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{y}, v \geq 0.$$

with $v = 0$. For the actual computation the new variable v is introduced in the first phase so that at the end of the second phase all the values necessary for the third phase are in the tableau. For this reason the equation

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} \quad \boldsymbol{w}^1 \quad \boldsymbol{w}^2 = \quad \boldsymbol{c}$$

is replaced by

$$2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\boldsymbol{\lambda} \quad \boldsymbol{\mu} \quad \boldsymbol{w}^1 \quad \boldsymbol{w}^2 \quad v\boldsymbol{c} = \boldsymbol{0},$$

with the additional rule that the variable v may never enter the basis during the first two phases, that is, $v = 0$. This amounts to $\boldsymbol{c}$ being $\boldsymbol{0}$. Therefore at the end of the second phase, we have a solution to (3.2.7) with $v = 0$ satisfying (3.2.3).

In order to obtain the solution for (3.2.2), v must equal to 1 and (3.2.3) must hold. To obtain this, the third phase of the long form proceeds by applying the simplex method to minimize the objective function

$$\textit{Minimize} \quad v$$

subject to the constraints (3.2.7) and the conditions (3.2.3).

Two cases may now arise:

1. v is unbounded from below.
2. The process ends with a finite value for v.

If case (2) occurs, Theorem 3.1 can be applied to the long form. $\boldsymbol{w} = \boldsymbol{0}$ and is maintained at zero during the third phase, the third phase minimizes v. Theorem 3.1 is applied and since $n = 1$ all vectors and matrices $\boldsymbol{k}, \boldsymbol{q}, \boldsymbol{d}$ and $\boldsymbol{R}$ employed in the theorem are scalar quantities

$$k = v, \quad q = \quad 1$$

$$q'k = \ v, \ R = c, \ d = 0$$

and it can be concluded that

$$min(\ v) = q'k = d'r = 0.$$

Therefore if v cannot be reduced to ∞ in the second phase then no steps of the simplex method can be carried out such that the conditions (3.2.3) are satisfied. In this case the objective functions $f(\boldsymbol{x})$ is not bounded from below and the problem has no solution.

In case (1) since only a finite number of basis are available, a finite sequence of basic solutions $(\boldsymbol{x}^j, \boldsymbol{\mu}^j, \boldsymbol{\lambda}^j, v^j)$ for $j = 1, 2, \ldots, g$ is obtained in the third phase. The solution $(\boldsymbol{x}^1, \boldsymbol{\mu}^1, \boldsymbol{\lambda}^1, v^1)$ is obtained from the first tableau at the beginning of the third phase with $v^1 = 0$, $\boldsymbol{x}^1 = (x_1, \ldots, x_n)$ where for all x_i in the basis the corresponding value is taken from the tableau and the remaining x_i are zero. This also applies for the variables $\boldsymbol{\mu}^1$ and $\boldsymbol{\lambda}^1$. The solution $(\boldsymbol{x}^2, \boldsymbol{\mu}^2, \boldsymbol{\lambda}^2, v^2)$ is taken from the second tableau in the third phase and so on. Assuming that the system is not degenerate suppose that the sequence of the v-values is $0 = v^0 < v^1 < v^2 < \ldots < v^g$ with the $\boldsymbol{x}$-parts $\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^g$ of their associated basic solutions, with $g = j$ when $v = 1$.

The solution with $v = 1$ is obtained as a linear combination of two solutions. Two cases are possible to force v to 1:

Case 1: For $v^g \geq 1$

Choose an index j with $1 \leq j \leq g \ \ 1$ such that $v^j < 1 \leq v^{j \ 1}$. Then

(3.2.8) $$(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}}, \hat{v}) = \frac{v^{j \ 1} \ 1}{v^{j \ 1} \ v^j}(\boldsymbol{x}^j, \boldsymbol{\mu}^j, \boldsymbol{\lambda}^j, v^j) \ \ \frac{1 \ v^j}{v^{j \ 1} \ v^j}(\boldsymbol{x}^{j \ 1}, \boldsymbol{\mu}^{j \ 1}, \boldsymbol{\lambda}^{j \ 1}, v^{j \ 1})$$

is the solution to the problem (3.2.2) since $\hat{v} = 1$.

Case 2: For $v^g < 1$

(3.2.9) $$(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}}, \hat{v}) = (\boldsymbol{x}^g, \boldsymbol{\mu}^g, \boldsymbol{\lambda}^g, v^g) \ \ \frac{1 \ v^g}{v^{g \ 1} \ v^g}(\boldsymbol{x}^{g \ 1}, \boldsymbol{\mu}^{g \ 1}, \boldsymbol{\lambda}^{g \ 1}, v^{g \ 1}).$$

Since $\hat{v} = 1$, $\hat{\boldsymbol{x}}$ solves the problem (3.2.2) [KTZ1971].

Apart from using the Long Form, there is another technique of dealing with the case of the matrix $\boldsymbol{Q}$ being semi-definite. This simple modification of the procedure was suggested by E. M. L. Beale [Wol1959]. This modification is done by adding some small value ε to the diagonal elements of the matrix $\boldsymbol{Q}$ (that is, $\boldsymbol{Q} \ \ \varepsilon \boldsymbol{I}$), which makes the positive semi-definite

matrix into a positive definite matrix. This technique is called the perturbation technique. Applying this technique to a problem with a semi-definite matrix allows for the problem to be solved using the short form of Wolfe's algorithm, which makes it more desirable to use as the short form of the algorithm is more efficient. A more detailed discussion on the perturbation technique is provided in Appendix B.3.

An example is now given, which illustrates the algorithm. The example given here is explained using the long form of Wolfe's algorithm as it can be used for both positive definite and semi-definite case. The example is also solved using the short form with the perturbation technique.

Example 1

Consider the following problem

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = x_1 \;\; 2x_2 \;\; x_3 \;\; \tfrac{1}{2}(x_1^2 \;\; 2x_2^2 \;\; 4x_3^2 \;\; 4x_2x_3) \\ &\textit{Subject to} && x_1 \;\; 2x_2 \;\; 4x_3 \le 12 \\ & && 2x_1 \;\; x_2 \;\; 3x_3 \le 9 \\ & && x_1, x_2, x_3 \ge 0 \end{aligned}$$

By introducing the slack variables y_1 and y_2 and the artificial variables s_1 and s_2, the constraints become $x_1 \;\; 2x_2 \;\; 4x_3 \;\; y_1 \;\; s_1 = 12$, $2x_1 \;\; x_2 \;\; 3x_3 \;\; y_2 \;\; s_2 = 9$ and $x_1, x_2, x_3 \ge 0$, $y_1, y_2 \ge 0$ and $s_1, s_2 \ge 0$. Then for this problem

$$\boldsymbol{c} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \; 2\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 4 \end{bmatrix}, \; \boldsymbol{A} = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 1 & 3 \end{bmatrix} \text{ and } \boldsymbol{b} = \begin{bmatrix} 12 \\ 9 \end{bmatrix}.$$

Here the matrix $\boldsymbol{Q}$ is positive definite. Adding the artificial variables $\boldsymbol{s}$, $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$ the problem is written as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 2 & 0 & 0 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 2 & 4 & 0 & 0 & 4 & 3 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 2 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y_1 \\ y_2 \\ \lambda_1 \\ \lambda_2 \\ \mu_1 \\ \mu_2 \\ \mu_3 \\ w_1^1 \\ w_2^1 \\ w_3^1 \\ w_1^2 \\ w_2^2 \\ w_3^2 \\ s_1 \\ s_2 \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 12 \\ 9 \end{bmatrix}.$$

The pivot elements will be indicated by a shaded cell and the abbreviation B.V. is used for basic variable due to the lack of space. The initial tableau then has the form

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2	v
0	w_1^1	0	1	0	0	0	0	1	2	-1	0	0	-1	0	0	0	0	-1
0	w_2^1	0	0	2	-2	0	0	2	1	0	-1	0	0	-1	0	0	0	-2
0	w_3^1	0	0	-2	4	0	0	4	3	0	0	-1	0	0	-1	0	0	-1
1	s_1	12	1	2	4	1	0	0	0	0	0	0	0	0	0	1	0	0
1	s_2	9	2	1	3	0	1	0	0	0	0	0	0	0	0	0	1	0
	z_j	12	3	3	7	1	1											

In the first phase all s_i are eliminated from the basis. In this case s_1 and s_2. From the first tableau it is clear that x_3 will enter the basis as its z_j is the largest positive. The variable w_3^1 will leave the basis as can be seen from the calculation after Step 4. During the first phase μ, λ and v do not enter the basis. After performing the transformations the next tableaus is as follows.

Coeff. B.V.	B.V.	x_B	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2	v
0	w_1^1	0	1	0	0	0	0	1	2	-1	0	0	-1	0	0	0	0	-1
0	w_2^1	0	0	1	0	0	0	4	2 2/5	0	-1	-1/2	0	-1	-1/2	0	0	-2 1/2
0	x_3	0	0	-1/2	1	0	0	1	3/4	0	0	-1/4	0	0	-1/4	0	0	-1/4
1	s_1	12	1	4	0	1	0	-4	-3	0	0	1	0	0	1	1	0	1
1	s_2	9	2	2 2/5	0	0	1	-3	-2 1/4	0	0	3/4	0	0	3/4	0	1	3/4
	z_j	12	3	6 1/2		1	1								1 3/4			

After the first iteration s_1 nor s_2 has not been removed from the basis. Again the largest positive is selected corresponding to the row z_j. In this case x_2 will enter the basis and w_2^1 will leave the basis.

Coeff. B.V.	B.V.	x_B	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2	v
0	w_1^1	0	1	0	0	0	0	1	2	-1	0	0	-1	0	0	0	0	-1
0	x_2	0	0	1	0	0	0	4	2 1/2	0	-1	-1/2	0	-1	-1/2	0	0	-2 1/2
0	x_3	0	0	0	1	0	0	3	2	0	-1/2	-1/4	0	-1/2	-1/4	0	0	-1 1/2
1	s_1	12	1	0	0	1	0	-20	-13	0	4	3	0	4	3	1	0	11
1	s_2	9	2	0	0	0	1	-13	-8 1/2	0	2 1/2	2	0	2 1/2	2	0	1	7
	z_j	12	3			1	1							6 1/2	5			

At this stage the variable w_2^2 will enter the basis and s_1 will be removed from the basis resulting in the following tableau.

Coeff. B.V.	B.V.	x_B	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2	v
0	w_1^1	0	1	0	0	0	0	1	2	-1	0	0	-1	0	0	0	0	-1
0	x_2	3	1/4	1	0	1/4	0	-1	-3/4	0	0	1/4	0	0	1/4	1/4	0	1/4
0	x_3	1 1/2	1/8	0	1	1/8	0	1/2	3/8	0	0	-1/8	0	0	-1/8	1/8	0	-1/8
0	w_2^2	3	1/4	0	0	1/4	0	-5	-3 1/4	0	1	3/4	0	1	3/4	1/4	0	2 3/4
1	s_2	1 1/2	1 3/8	0	0	-5/8	1	-1/2	-3/8	0	0	1/8	0	0	1/8	-5/8	1	1/8
	z_j	0	1 3/8												1/8			

At this point x_1 enters the basis and w_1^1 is removed from the basis.

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2	v
0	x_1	0	1	0	0	0	0	1	2	-1	0	0	-1	0	0	0	0	-1
0	x_2	3	0	1	0	1/4	0	-1 1/4	-1 1/4	1/4	0	1/4	1/4	0	1/4	1/4	0	1/2
0	x_3	1 1/2	0	0	1	1/8	0	3/8	1/8	1/8	0	-1/8	1/8	0	-1/8	1/8	0	0
0	w_2^2	3	0	0	0	1/4	0	-5 1/4	-3 3/4	1/4	1	3/4	1/4	1	3/4	1/4	0	3
1	s_2	1 1/2	0	0	0	-5/8	1	-1 7/8	-3 1/8	1 3/8	0	1/8	1 3/8	0	1/8	-5/8	1	1 1/2
	z_j	0											1 3/8					

At this point w_1^2 enters the basis and replaces s_2 .

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2	v
0	x_1	1	1	0	0	-4/9	5/7	-1/3	-2/7	0	0	0	0	0	0	-4/9	5/7	0
0	x_2	2 3/4	0	1	0	1/3	-1/5	-1	-2/3	0	0	2/9	0	0	2/9	1/3	-1/5	2/9
0	x_3	1 3/8	0	0	1	1/5	0	1/2	2/5	0	0	-1/7	0	0	-1/7	1/5	0	-1/7
0	w_2^2	2 3/4	0	0	0	1/3	-1/5	-5	-3 1/6	0	1	5/7	0	1	5/7	1/3	-1/5	2 3/4
0	w_1^2	1	0	0	0	-4/9	5/7	-1 1/3	-2 2/7	1	0	0	1	0	0	-4/9	5/7	1
	z_j	0																

Since all the s_i have been removed from the basis the first phase is finished. To proceed with the second phase the columns corresponding to w_i^1 and w_i^2 not in the basis can be removed from the tableau. An n-vector $\boldsymbol{w}=(w_1,\ldots,w_n)'$ is constructed from the remaining w_i^1 and w_i^2 , where w_i is either w_i^1 or w_i^2 depending on which of the two is left in the basis at the end of the first phase. The new tableau then looks as follows.

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	s_1	s_2	v
0	x_1	1	1	0	0	-4/9	5/7	-1/3	-2/7	0	0	0	-4/9	5/7	0
0	x_2	2 3/4	0	1	0	1/3	-1/5	-1	-2/3	0	0	2/9	1/3	-1/5	2/9
0	x_3	1 3/8	0	0	1	1/5	0	1/2	2/5	0	0	-1/7	1/5	0	-1/7
1	w_1	2 3/4	0	0	0	1/3	-1/5	-5	-3 1/6	0	1	5/7	1/3	-1/5	2 3/4
1	w_2	1	0	0	0	-4/9	5/7	-1 1/3	-2 2/7	1	0	0	-4/9	5/7	1
	z_j	0					1/2			1	1	5/7			

In the second phase the procedure for eliminating variables from the basis and entering variables into the basis follows as in the first phase, however in the second phase $\sum w_i$ is to be minimized, that is, all w_i are eliminated from the basis while satisfying the condition $\boldsymbol{x}'\boldsymbol{\mu}=0$. From this tableau μ_3 is next to enter the basis, however x_3 is already in the basis and therefore in order to satisfy the condition $\boldsymbol{x}'\boldsymbol{\mu}=0$, μ_3 cannot enter the basis. Therefore y_2 is the next variable to enter the basis and w_2 is removed from the basis.

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	s_1	s_2	v
0	x_1	0	1	0	0	0	0	1	2	-1	0	0	1/3	-1/5	-1
0	x_2	3	0	1	0	1/4	0	-1 1/4	-1 1/4	1/4	0	1/4	1/5	0	1/2
0	x_3	0	0	0	1	0	0	3	2	0	-1/2	-1/2	3/5	-1/2	-1 1/2
1	w_1	3	0	0	0	1/4	0	-5 1/4	-3 3/4	1/4	1	3/4	-4/9	5/7	3
0	y_2	1 1/2	0	0	0	-5/8	1	-1 7/8	-3 1/8	1 3/8	0	1/8	0	0	1 1/2
	z_j	3				1/4				1/4	1				

Neither μ_1 or μ_2 can enter the basis since x_1 and x_2 are already in the basis, therefore y_1 enters that basis.

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	s_1	s_2	v
0	x_1	0	1	0	0	0	0	1	2	-1	0	0	1/3	-1/5	-1
0	x_2	0	0	1	0	0	0	4	2 1/2	0	-1	-1/2	2/3	-5/6	-2 1/2
0	x_3	0	0	0	1	0	0	3	2	0	-1/2	-1/2	3/5	-1/2	-1 1/2
0	y_1	12	0	0	0	1	0	-21	-15	1	4	3	-1 4/5	3	12
0	y_2	9	0	0	0	0	1	-15	-12 1/2	2	2 1/2	2	-1 1/7	1 5/6	9
	z_j	0													

Since all w_i have been removed from the basis, the second phase is completed. In the third phase, the simplex method is applied to the basic solution obtained at the end of the second phase to minimize v. Again the simplex method is applied, in which case v will enter the basis and y_1 is removed from the basis.

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	s_1	s_2	v
0	x_1	1	1	0	0	0	0	-3/4	3/4	-1	1/3	1/4	1/5	0	0
0	x_2	2 1/2	0	1	0	1/5	0	-3/8	-5/8	1/5	-1/6	1/8	1/4	-1/5	0
0	x_3	1 1/2	0	0	1	1/8	0	3/8	1/8	1/8	0	-1/8	1/3	-1/6	0
1	v	1	0	0	0	0	0	-1 3/4	-1 1/4	0	1/3	1/4	-1/7	1/4	1
0	y_2	9	0	-18	0	-3/4	1	3/4	-1 1/4	1 1/4	-1/2	-1/4	2/9	-1/3	0
	z_j														

The previous tableau is the first tableau of the third phase. At this point the value of $v=0$, $\boldsymbol{x}^1=(0,0,0,12,9)'$, $\boldsymbol{\mu}^1=(0,0,0,0,0)'$ and $\boldsymbol{\lambda}^1=(0,0)$. After the first transformation the above tableau results in $v^2=1\geq 1$, $\boldsymbol{x}^2=(1,2\frac{1}{2},1\frac{1}{2},0,0)'$, $\boldsymbol{\mu}^2=(0,0,0,0,0)'$ and $\boldsymbol{\lambda}^2=(0)$. At this point $v^g=v^2\geq 1$ and $v^j<1\leq v^{j-1}$, that is, $v^j=0=v^1<1\leq v^2=1=v^{j-1}$.

$$(\hat{\boldsymbol{x}},\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\lambda}},\hat{v}) = \frac{v^2 \quad 1}{v^2 \quad v^1}(\boldsymbol{x}^1,\boldsymbol{\mu}^1,\boldsymbol{\lambda}^1,v^1) \quad \frac{1 \quad v^1}{v^2 \quad v^1}(\boldsymbol{x}^2,\boldsymbol{\mu}^2,\boldsymbol{\lambda}^2,v^2)$$
$$= 0(0,0,0,12,9,0,0,0,0,0,0) \quad 1(1,2\tfrac{1}{2},1\tfrac{1}{2},0,0,0,0,0,0,0,0,1)$$
$$= (1,2\tfrac{1}{2},1\tfrac{1}{2},0,0,0,0,0,0,0,0,1)$$

The corresponding linear combination of $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ solves the problem since $0\,v^1 \quad 1\,v^2 = 1 = v$ with $\hat{\boldsymbol{x}} = (1,2\tfrac{1}{2},1\tfrac{1}{2},0)'$. Therefore the solution point is $x_1 = 1$, $x_2 = 2\tfrac{1}{2}$ and $x_3 = 1\tfrac{1}{2}$. Since this point satisfies the Kuhn-Tucker conditions it is the solution for the original quadratic programming problem (3.2.1). Substituting the solution to the objective function of (3.2.1) the final answer is

$$f(\boldsymbol{x}) = \quad x_1 \quad 2x_2 \quad x_3 \quad \frac{1}{2}(x_1^2 \quad 2x_2^2 \quad 4x_3^2 \quad 4x_2x_3) = \quad 3\tfrac{3}{4}.$$

Example 2

The same example is now solved by using the perturbation technique.

$$\textit{Minimize} \quad f(\boldsymbol{x}) = \quad x_1 \quad 2x_2 \quad x_3 \quad \frac{1}{2}(x_1^2 \quad 2x_2^2 \quad 4x_3^2 \quad 4x_2x_3)$$
$$\textit{Subject to} \quad x_1 \quad 2x_2 \quad 4x_3 \le 12$$
$$2x_1 \quad x_2 \quad 3x_3 \le 9$$
$$x_1, x_2, x_3 \ge 0$$

As mentioned earlier, the perturbation technique involves adding a small value to the diagonal elements of $\boldsymbol{Q}$. For this problem the value $\varepsilon = 0.001$ will be added and the new matrix $\boldsymbol{Q}$ is as follows

$$2\boldsymbol{Q} = \begin{bmatrix} 1.001 & 0 & 0 \\ 0 & 2.001 & 2 \\ 0 & 2 & 4.001 \end{bmatrix}.$$

To solve this problem the long form is not necessary as the problem can be solved using the short form of the algorithm. Applying the Kuhn-Tucker conditions and adding the artificial variables s_i, w_i^1 and w_i^2 the initial tableau is given as follows:

Coeff. B.V.	B.V.	$\boldsymbol{x}_B$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2
0	w_1^1	1	1.001	0	0	0	0	1	2	-1	0	0	-1	0	0	0	0
0	w_2^1	2	0	2.001	-2	0	0	2	1	0	-1	0	0	-1	0	0	0
0	w_3^1	1	0	-2	4.001	0	0	4	3	0	0	-1	0	0	-1	0	0
1	s_1	12	1	2	4	1	0	0	0	0	0	0	0	0	0	1	0
1	s_2	9	2	1	3	0	1	0	0	0	0	0	0	0	0	0	1
	z_j	12	3	3	7	1	1										

The first step is to execute the first phase of the algorithm. As in the previous example s_i now needs to be eliminated from the tableau. After performing a series of iterations, in this case 5, the tableau looks as follows:

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	w_1^2	w_2^2	w_3^2	s_1	s_2
0	x_1	1	1	0	0	-0.45	0.73	-0.36	-0.27	0	0	0.09	0	0	0.09	-0.45	0.73
0	x_2	2.5	0	1	0	0.36	-0.18	-0.91	-0.68	0	0	0.23	0	0	0.23	0.36	-0.18
0	x_3	1.5	0	0	1	0.18	-0.09	0.55	0.41	0	0	-0.14	0	0	-0.14	0.18	-0.09
0	w_2^2	0.004	0	0	0	0.36	-0.18	-4.91	-3.18	0	1	0.73	0	1	0.73	0.36	-0.18
0	w_1^2	0.001	0	0	0	-0.45	0.73	-1.36	-2.27	1	0	0.09	1	0	0.09	-0.45	0.73
	z_j																

All s_i have been removed from the basis which ends the first phase. At the beginning of the second phase all w_i^1 and w_i^2 not in the basis can be removed from the tableau. A new vector w_i is constructed from the remaining w_i^1 and w_i^2 . The tableau at the beginning of the second phase is as follows:

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	s_1	s_2
0	x_1	1	1	0	0	-0.45	0.73	-0.36	-0.27	0	0	0.09	-0.45	0.73
0	x_2	2.5	0	1	0	0.36	-0.18	-0.91	-0.68	0	0	0.23	0.36	-0.18
0	x_3	1.5	0	0	1	0.18	-0.09	0.55	0.41	0	0	-0.14	0.18	-0.09
1	w_1	0.004	0	0	0	0.36	-0.18	-4.91	-3.18	0	1	0.73	0.36	-0.18
1	w_2	0.001	0	0	0	-0.45	0.73	-1.36	-2.27	1	0	0.09	-0.45	0.73
	z_j													

In the second phase all w_i are eliminated from the basis. The procedure is the same as in Example 1 and after 2 iterations the tableau is given as follows:

Coeff. B.V.	B.V.	$\boldsymbol{x_B}$	x_1	x_2	x_3	y_1	y_2	λ_1	λ_2	μ_1	μ_2	μ_3	s_1	s_2
0	x_1	0.999	1	0	0	0	0	0.999	1.998	-0.999	0	0	0	0
0	x_2	2.497	0	1	0	0	0	3.995	2.497	0	-0.999	-0.499	0	0
0	x_3	1.498	0	0	1	0	0	2.997	1.998	0	-0.499	-0.499	0	0
0	y_1	0.015	0	0	0	1	0	-20.97	-14.98	0.999	3.995	2.997	1	0
0	y_2	0.011	0	0	0	0	1	-14.98	-12.49	1.998	2.496	1.998	0	1
	z_j													

Since all w_i have been eliminated from the basis, the second phase is finished and the solution has been found. The final solution given in the final tableau rounding to one

decimal place is $x_1 = 1, x_2 = 2\frac{1}{2}$ and $x_3 = 1\frac{1}{2}$, which is the same as using the long form to solve the problem. Substituting into the original problem, the final answer is

$$f(\boldsymbol{x}) = \; x_1 \; 2x_2 \; x_3 \; \frac{1}{2}(x_1^2 \; 2x_2^2 \; 4x_3^2 \; 4x_2x_3) = \; 3\tfrac{3}{4}.$$

Although it is relatively easy to determine if a matrix is definite or semi-definite, this operation needs a large amount of computational power and time when dealing with real world problems in which the matrices become considerably larger than in the problem above. Therefore it is not necessary to do this operation and instead simply use the long form of the algorithm regardless of the matrix being definite or semi-definite or alternatively always perturb the diagonal elements and then the short form can be used.

3.2.1.1 Dantzig's Algorithm

The next variation of the Simplex Method was introduced by G. B. Dantzig in 1961 [Boo1964]. This procedure was also found by Van de Panne and Winston in 1964 [VW1964], however independently from Dantzig. The reason for selecting to explain Dantzig's algorithm above the other variations of Wolfe's method is attributed to the fact that it was Dantzig who first introduced the simplex method for linear programming. This algorithm is not restricted to problems where $\boldsymbol{Q}$ is positive definite, that is, it can be used to solve quadratic programming problems for which the matrix $\boldsymbol{Q}$ is semi-definite.

Consider the following minimization problem:

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \;\; \boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} && \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & && \boldsymbol{x} \ge 0. \end{aligned} \tag{3.2.10}$$

It is assumed that an initial basic feasible solution for (3.2.10) exists. This can be obtained by Phase One of the Two-phase simplex method if necessary.

The Kuhn-Tucker conditions for (3.2.10) are give as

$$\begin{aligned} 2\boldsymbol{Q}\boldsymbol{x} \;\; \boldsymbol{A}'\boldsymbol{\lambda} \;\; \boldsymbol{\mu} &= \;\; \boldsymbol{c} \\ \boldsymbol{A}\boldsymbol{x} \;\; \boldsymbol{y} &= \boldsymbol{b} \\ \boldsymbol{\mu}'\boldsymbol{x} = 0, \;\; \boldsymbol{\lambda}'\boldsymbol{y} &= 0 \\ \boldsymbol{x} \ge 0, \boldsymbol{\mu} \ge 0, \boldsymbol{y} \ge 0, \;\; \boldsymbol{\lambda} &\ge 0. \end{aligned} \tag{3.2.11}$$

The system (3.2.11) can be written as

(3.2.12)
$$\begin{gathered} 2\boldsymbol{Q}\boldsymbol{x} \quad \boldsymbol{A}'\lambda \quad \mu = \boldsymbol{c} \\ \boldsymbol{A}\boldsymbol{x} \quad \boldsymbol{y} = \boldsymbol{b} \\ \mu'\boldsymbol{x} = 0\,,\ \lambda'\boldsymbol{y} = 0 \\ \boldsymbol{x} \geq 0, \mu \geq 0, \boldsymbol{y} \geq 0,\ \lambda \geq 0. \end{gathered}$$

This procedure is similar to the simplex method in that it also uses a tableau to obtain a set of feasible solutions to obtain the minimum. The tableau is based on the Kuhn-Tucker conditions (3.2.12) and is given as follows:

Basic Variable	$\boldsymbol{x}_B$	$\boldsymbol{x}$	$\boldsymbol{y}$	μ	λ
μ	$\boldsymbol{c}$	$2\boldsymbol{Q}$	$\boldsymbol{0}$	$\boldsymbol{I}$	$\boldsymbol{A}$
$\boldsymbol{y}$	$\boldsymbol{b}$	$\boldsymbol{A}$	$\boldsymbol{I}$	$\boldsymbol{0}$	$\boldsymbol{0}$

The solution corresponding to the tableau is $\mu = \boldsymbol{c}$ and $\boldsymbol{y} = \boldsymbol{b}$ for the basic variables and $\boldsymbol{x} = 0,\ \boldsymbol{y} = 0,\ \lambda = 0$ for the non-basic variables. If $\boldsymbol{b} \geq 0$, the solution $\boldsymbol{y} = \boldsymbol{b}$ and $\boldsymbol{x} = 0$ is a feasible solution to the constraint of the quadratic programming problem. For convenience, during the computational procedure the tableau is written using the following substitution: $\xi_j = x_j,\ j = 1, \ldots, n$; $\xi_{n\ i} = y_i,\ i = 1, \ldots, m$; $\omega_j = \mu_j,\ j = 1, \ldots, n$ and $\omega_{n\ i} = \lambda_i,\ i = 1, \ldots, m$.

Two types of basic feasible solutions are possible:

1. A basic feasible solution is in standard form if of each pair (ξ_p, ω_p) exactly one element is non-basic, for $p = j \quad i$ where $j = 1, \ldots, n$ and $i = 1, \ldots, m$.
2. The basic feasible solution is in non-standard form if of each pair (ξ_p, ω_p) there is exactly one pair which is basic and therefore exactly one pair which is non-basic.

Denote by $\boldsymbol{u} = (\boldsymbol{x}'\ \boldsymbol{y}')' \geq 0$ and $\boldsymbol{v} = (\mu'\ \lambda')' \geq 0$. From the Kunh-Tucker conditions it can be stated that $\boldsymbol{u}$ is a solution to (3.2.11) if $\boldsymbol{v}'\boldsymbol{u} = 0$. From the Kuhn-Tucker conditions if a basic feasible solution in standard form satisfies $\boldsymbol{v} \geq 0$ then the optimal solution has been found. This is the aim of the procedure.

The basic elements of $\boldsymbol{u}$ must be non-negative, however, there is no such restriction on the elements of $\boldsymbol{v}$, although the vector $\boldsymbol{v}$ will be non-negative in the optimal solution. The next step is to determine the rules for selecting which variable will enter the basis and which variable is removed from the basis. Once the variables that leave the basis and that enter the basis have been determined, the transformation continues as in the simplex method. The pivot element is in the column associated with the non-basic variable which becomes

basic and the row associated with the basic variable which becomes non-basic. Let ρ_i be the values taken by the basic variables (these are the variables corresponding to the second column of the tableau). The rules are given in the following table.

Step 1	If tableau is in standard form and all basic variables $\boldsymbol{v} \geq 0$, a minimal solution has been found and the procedure stop. Otherwise go to Step 2.
Step 2	(a) If the tableau is in standard form then the non-basic $\boldsymbol{u}$ -variables ξ_h should enter the basis, whose ω_h has the largest negative ρ_h (e.g. in absolute values -2.5 is larger than -2), since it follows from equation (2.5.9) $\partial L / \partial \xi_h = \mu_h$ that it is profitable to make ξ_h basic. Go to Step 3.
	(b) If the tableau is non-standard, that is one pair (ξ_h, ω_h) is basic and one pair (ξ_k, ω_k) non-basic, then for the non-basic pair (ξ_k, ω_k), ω_k should enter the basis, that is, the Lagrangian of the non-basic pair. This can be done since Increasing ω_k will never increase the value of L (where L is the Lagrangian equation (2.5.8)). Go to Step 4.
Step 3	Let τ_i denote the values in the column of the variable that is to enter the basis, determined in Step 2 (a), that is, ξ_h. As in the Simplex Method (Appendix B.1), determine the ratio ρ_i / τ_i for all basic $\boldsymbol{u}$-variables and for ω_h. Delete from the basis the variable corresponding to the smallest positive ratio. Go to Step 5.
Step 4	Let τ_i denote the values of the column of the variable that is to enter the basis determined in Step 2 (b), that is, ω_k. As in the Simplex Method (Appendix B.1), determine the ratio ρ_i / τ_i for all basic $\boldsymbol{u}$ -variables and ω_k. Delete the variable corresponding to the smallest positive ratio. This rule is the same as for the standard case and changes the tableau from non-standard form back to standard form. Go to Step 5.
Step 5	Once the variable to enter the basis and to leave the basis has been determined the transformations for each row are executed as in equations (B.1.8) and (B.1.9) of Step 5 of the simplex method given in Appendix B.1. Then go to Step 1.

After determining the variable that will enter the basis in Step 2 (a) it has to be determined by how much the value of ξ_h will increase. The value of ω_h changes with the increasing value of ξ_h, thus ω_h might become zero at a point where further additions to ξ_h are still feasible. Further increases however, do no improve the value of the objective function. It is

therefore important to establish how far should the variable that is to become basic increase. As long as ω_h remains negative, the ξ_h is increased. When ω_h becomes zero any increases in ξ_h increase the value of L. If during the increase from zero to a positive value, $\partial L/\partial \xi_h = \mu_h$ becomes zero while all basic $\boldsymbol{u}$-variables are still negative, then ω_h is deleted from the basis. There are now two possibilities. First, the value of the objective function has decreased, ξ_h replaces ω_h as a basic variable and the new solution is in standard form. Or the second, while increasing ξ_h some basic variable ξ_k becomes zero before ω_h becomes zero. Any further increase in ξ_h will cause the solution to become infeasible, therefore ξ_r has to become non-basic. This will lead to a non-standard tableau containing one pair (ξ_h, ω_h) of basic variables and one pair (ξ_k, ω_k) of non-basic variables.

It is possible for a non-standard tableau (in Step 2 (b)) that while ω_k increases, some basic $\boldsymbol{u}$-variable ξ_l becomes zero. Then ω_k replaces ξ_l. In this case the pair (ξ_h, ω_h) is still basic and the pair (ξ_l, ω_l) is non-basic and the solution is still non-standard. At this point ω_l enters the basis. It is also possible that ω_h is deleted as ω_k is increased before any basic $\boldsymbol{u}$-variable becomes zero. At this point the tableau is in standard form and the procedure continues as in the table outlined above.

The value of the objective function increases monotonically, therefore no solution can be repeated and the solution is obtained in a finite number of steps. Once all basic variables $\boldsymbol{v} \geq 0$, the minimal solution has been found. The values for $\xi_j = x_j, j = 1, \ldots, n$ obtained in the final tableau are substituted into the original objective function of problem (3.2.10) to obtain the minimal solution for the problem.

The following example (used in Wolfe's algorithm) is used to illustrate the procedure outlined above. In this example the matrix $\boldsymbol{Q}$ is positive definite.

Example

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = x_1 \;\; 2x_2 \;\; x_3 \;\; \tfrac{1}{2}(x_1^2 \;\; 2x_2^2 \;\; 4x_3^2 \;\; 4x_2x_3) \\ &\textit{Subject to} && x_1 \;\; 2x_2 \;\; 4x_3 \leq 12 \\ & && 2x_1 \;\; x_2 \;\; 3x_3 \leq 9 \\ & && x_1, x_2, x_3 \geq 0 \end{aligned}$$

The initial tableau is give as follows.

Basic Variable	x_B	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ω_1	ω_2	ω_3	ω_4	ω_5
ω_1	-1	-1	0	0	0	0	1	0	0	-1	-2
ω_2	-2	0	-2	2	0	0	0	1	0	-2	-1
ω_3	-1	0	2	-4	0	0	0	0	1	-4	-3
ξ_4	12	1	2	4	1	0	0	0	0	0	0
ξ_5	9	2	1	3	0	1	0	0	0	0	0

This tableau is standard and the objective value of the function is zero. As ω_2 is the most negative ξ_2 is added to the basis and ω_2 leaves the basis since $\frac{2}{2}<\left\{\frac{12}{2},\frac{9}{1}\right\}$. The next tableau thus looks as follows.

Basic Variable	x_B	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ω_1	ω_2	ω_3	ω_4	ω_5
ω_1	-1	-1	0	0	0	0	1	0	0	-1	-2
ξ_2	1	0	1	-1	0	0	0	-1/2	0	1	1/2
ω_3	-3	0	0	-2	0	0	0	1	1	-6	-4
ξ_4	10	1	0	6	1	0	0	1	0	-2	-1
ξ_5	8	2	0	4	0	1	0	1/2	0	-1	-1/2

The tableau is standard form. Since ω_3 is the most negative ξ_3 enters the basis. Calculating the ratios ω_3 leaves the basis.

Basic Variable	x_B	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ω_1	ω_2	ω_3	ω_4	ω_5
ω_1	-1	-1	0	0	0	0	1	0	0	-1	-2
ξ_2	2 1/2	0	1	0	0	0	0	-1	-1/2	4	2 1/2
ξ_3	1 1/2	0	0	1	0	0	0	-1/2	-1/2	3	2
ξ_4	5 1/2	1	0	0	1	0	0	4	3	-20	-13
ξ_5	2	2	0	0	0	1	0	2 1/2	2	-13	-8 1/2

Again the tableau is standard. Since ω_1 is the most negative ξ_1 enters the basis.

Basic Variable	x_B	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ω_1	ω_2	ω_3	ω_4	ω_5
ξ_1	1	1	0	0	0	0	-1	0	0	1	2
ξ_2	2 1/2	0	1	0	0	0	0	-1	-1/2	4	2 1/2
ξ_3	1 1/2	0	0	1	0	0	0	-1/2	-1/2	3	2
ξ_4	0	0	0	0	1	0	1	4	3	-21	-15
ξ_5	0	0	0	0	0	1	2	2 1/2	2	-15	-12 1/2

Since all the x_B are positive the final solution has been found. The solution is $\xi_1 = x_1 = 1$, $\xi_2 = x_2 = 2\frac{1}{2}$ and $\xi_3 = x_3 = 1\frac{1}{2}$ and the final solution of the objective function is $f(x) = 3\frac{3}{4}$.

For the purpose of consistency the example used in Wolfe's algorithm was also used for Dantzig's algorithm. However, this example does not fully show all the aspects of the algorithm – in particular a non-standard basis – therefore a more complex example is provided in Appendix B.4.

3.2.1.2 Other Simplex Method Variants

Although Wolfe's algorithm is not the first adaptation of the simplex method for quadratic programming, it is the most widely used, which is the reason the method was selected rather than other methods. Earlier variants of the simplex method are the Frank-Wolfe algorithm introduced by M. Frank and P. Wolfe in 1956 [FW1956], Hildreth's method developed by C. G. Hildreth in 1957 and the Barankin-Dorfman algorithm formulated by Barankin E. W. and R. Dorfman in 1958 [Boo1964]. Another variant of the simplex method is Lemke's algorithm introduced in 1962 [Sim1975].

All of these algorithms make use of the Kuhn-Tucker conditions and the row transformations of the simplex method as in Wolfe's algorithm. The difference is only in the selection criteria, for which variable is to enter the basis and which basic variable is to become non-basic.

3.2.2 Beale's Algorithm

Before describing this algorithm consider the following linear programming problem.

Minimize $f(x) = c'x$

(3.2.13) $$\text{Subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq 0$$

If there should be any inequalities in the original problem, these can be converted to equalities by adding appropriate slack variables as described in Appendix B.1.

Suppose that there exists an initial basic feasible solution $\boldsymbol{x} \equiv (\boldsymbol{x}_B, \boldsymbol{x}_R)$ with $\boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b}$ and $\boldsymbol{x}_R = \boldsymbol{0}$ ($\boldsymbol{x}_B$ is the basic variable and $\boldsymbol{x}_R$ the non-basic variable). It is possible to partition $\boldsymbol{A}$ such that $\boldsymbol{A} = [\boldsymbol{B} | \boldsymbol{R}]$ where $\boldsymbol{B}$ is an m x m matrix and $\boldsymbol{R}$ a m x $(n - m)$ matrix. Now

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x}_B + \boldsymbol{R}\boldsymbol{x}_R$$

so that

(3.2.14) $$\boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b} - \boldsymbol{B}^{-1}\boldsymbol{R}\boldsymbol{x}_R .$$

Further since

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{c}'\boldsymbol{x} \\ &= \boldsymbol{c}_B'\boldsymbol{x}_B + \boldsymbol{c}_R'\boldsymbol{x}_R \\ &= \boldsymbol{c}_B'(\boldsymbol{B}^{-1}\boldsymbol{b} - \boldsymbol{B}^{-1}\boldsymbol{R}\boldsymbol{x}_R) + \boldsymbol{c}_R'\boldsymbol{x}_R \\ &= \boldsymbol{c}_B'\boldsymbol{B}^{-1}\boldsymbol{b} - (\boldsymbol{c}_B'\boldsymbol{B}^{-1}\boldsymbol{R} - \boldsymbol{c}_R')\boldsymbol{x}_R \\ &= z_0 - \sum_{j \in P} (z_j - c_j) x_j \end{aligned} \tag{3.2.15}$$

using the fact that $\boldsymbol{B}^{-1}\boldsymbol{R}\boldsymbol{x}_R = \sum_{j \in P} \boldsymbol{B}^{-1} a_j x_j$ where P is the current set of indices of the non-basic variables.

Equations (3.2.14) and (3.2.15) express the basic variables and the objective function in terms of the non-basic variables. The simplex method considers equation (3.2.15) to determine which of the non-basic variables will provide the largest decrease in the value of the objective function. If $z_j - c_j \leq 0$ for all $j \in P$, then the current solution is optimal. Once the particular non-basic variable has been identified, equation (3.2.14) will be used to determine the largest possible value to which the non-basic variable can increase without violating the non-negativity restrictions.

In 1955, E. M. Beale [Bea1959] used these ideas in solving quadratic programming problems of the form

(3.2.16) $$\begin{aligned} &\text{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} + \boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\text{Subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad\qquad\quad \boldsymbol{x} \geq 0 \end{aligned}$$

where $\boldsymbol{A}$ is a m x n matrix, $\boldsymbol{b}$ an m-vector and $\boldsymbol{x}$ a n-vector. Without loss of any generality it is again assumed that $\boldsymbol{b} \geq 0$ and an initial basic feasible solution exists (this can be obtained

by Phase One of the simplex method if necessary). As before, if there should be any inequalities in the original problem, these can be converted to equalities by adding appropriate slack variables.

By reordering the columns of $\boldsymbol{A}$, the elements of $\boldsymbol{c}$ and the rows and columns of $\boldsymbol{Q}$, the vector $\boldsymbol{x}$ can be partitioned as before into two parts $\boldsymbol{x} \equiv (\boldsymbol{x}_B, \boldsymbol{x}_R)$, where $\boldsymbol{x}_R$ is an $(n \quad m)$ vector containing variables not in the basis and $\boldsymbol{x}_B$ is a vector containing basic variables. The matrix $\boldsymbol{A}$ can be partitioned into $[\boldsymbol{B} \quad \boldsymbol{R}]$, where the matrix $\boldsymbol{B}$ contains the variables $\boldsymbol{x}_B$ associated with the columns of $\boldsymbol{A}$ and $\boldsymbol{R}$ is the m x $(n \quad m)$ matrix containing the columns of $\boldsymbol{A}$ not in $\boldsymbol{B}$ associated with $\boldsymbol{x}_R$. Therefore the constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can be rewritten as

(3.2.17) $$\boldsymbol{B}\boldsymbol{x}_B \quad \boldsymbol{R}\boldsymbol{x}_R = \boldsymbol{b}.$$

As the equation (3.2.14) in the linear program above (3.2.17) can then be written as

(3.2.18) $$\boldsymbol{x}_B = \boldsymbol{B}^{\ 1}\boldsymbol{b} \quad \boldsymbol{B}^{\ 1}\boldsymbol{R}\boldsymbol{x}_R,$$

where the initial basic solution $\boldsymbol{x}_B = \boldsymbol{B}^{\ 1}\boldsymbol{b}$ is obtained by setting $\boldsymbol{x}_R = 0$. The initial basic solution will be feasible if $\boldsymbol{B}^{\ 1}\boldsymbol{b} \geq 0$. Finally $\boldsymbol{c}$ is partitioned as $\boldsymbol{c}' \equiv [\boldsymbol{c}'_{\boldsymbol{B}}, \boldsymbol{c}'_{\boldsymbol{R}}]$ and $\boldsymbol{Q}$ as

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{BB} & \boldsymbol{Q}_{BR} \\ \boldsymbol{Q}_{RB} & \boldsymbol{Q}_{RR} \end{bmatrix}$$

where $\boldsymbol{Q}_{BB}$ is a m x m matrix containing the rows and columns of $\boldsymbol{Q}$ associated with the basic variables $\boldsymbol{x}_B$. $\boldsymbol{Q}_{BR}$ is a m x $(n \quad m)$ matrix containing the rows of $\boldsymbol{Q}$ associated with $\boldsymbol{x}_B$ and the columns of $\boldsymbol{Q}$ associated with $\boldsymbol{x}_R$. Similarly $\boldsymbol{Q}_{RB}$ is a $(n \quad m)$ x m matrix such that $\boldsymbol{Q}_{RB} = \boldsymbol{Q}_{BR}'$. $\boldsymbol{Q}_{RR}$ is a $(n \quad m)$ x $(n \quad m)$ matrix. Since the objective function is quadratic, the equation (3.2.15) becomes

$$\begin{aligned} z &= \boldsymbol{c}'_{\boldsymbol{B}}\boldsymbol{x}_B \quad \boldsymbol{c}'_{\boldsymbol{R}}\boldsymbol{x}_R \quad [\boldsymbol{x}'_{\boldsymbol{B}}\boldsymbol{x}'_{\boldsymbol{R}}] \begin{bmatrix} \boldsymbol{Q}_{BB} & \boldsymbol{Q}_{BR} \\ \boldsymbol{Q}_{RB} & \boldsymbol{Q}_{RR} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_R \end{bmatrix} \\ z &= \{(\boldsymbol{c}'_{\boldsymbol{B}} \quad (\boldsymbol{B}^{\ 1}\boldsymbol{b})'\boldsymbol{Q}_{BB})\boldsymbol{B}^{\ 1}\boldsymbol{b}\} \\ &\quad \{2(\boldsymbol{B}^{\ 1}\boldsymbol{b})'\boldsymbol{Q}_{BR} \quad 2(\boldsymbol{B}^{\ 1}\boldsymbol{b})'\boldsymbol{Q}_{BB}\boldsymbol{B}^{\ 1}\boldsymbol{R} \quad \boldsymbol{c}'_{\boldsymbol{R}} \quad \boldsymbol{c}'_{\boldsymbol{B}}\boldsymbol{B}^{\ 1}\boldsymbol{R}\}\boldsymbol{x}_R \\ &\quad \boldsymbol{x}'_{\boldsymbol{R}}\{\boldsymbol{Q}_{RR} \quad 2\boldsymbol{Q}_{RB}\boldsymbol{B}^{\ 1}\boldsymbol{R} \quad (\boldsymbol{B}^{\ 1}\boldsymbol{R})'\boldsymbol{Q}_{BB}\boldsymbol{B}^{\ 1}\boldsymbol{R}\}\boldsymbol{x}_R \end{aligned} \tag{3.2.19}$$

after substituting (3.2.18) into the partitioned objective function. The expressions in the braces by in equation (3.2.19) are denoted by

$$\begin{aligned} z_0 &= \{(\boldsymbol{c}'_{\boldsymbol{B}} \quad (\boldsymbol{B}^{\ 1}\boldsymbol{b})'\boldsymbol{Q}_{BB})\boldsymbol{B}^{\ 1}\boldsymbol{b}\} \\ \boldsymbol{p} &= \{2(\boldsymbol{B}^{\ 1}\boldsymbol{b})'\boldsymbol{Q}_{BR} \quad 2(\boldsymbol{B}^{\ 1}\boldsymbol{b})'\boldsymbol{Q}_{BB}\boldsymbol{B}^{\ 1}\boldsymbol{R} \quad \boldsymbol{c}'_{\boldsymbol{R}} \quad \boldsymbol{c}'_{\boldsymbol{B}}\boldsymbol{B}^{\ 1}\boldsymbol{R}\} \\ \boldsymbol{D} &= \{\boldsymbol{Q}_{RR} \quad 2\boldsymbol{Q}_{RB}\boldsymbol{B}^{\ 1}\boldsymbol{R} \quad (\boldsymbol{B}^{\ 1}\boldsymbol{R})'\boldsymbol{Q}_{BB}\boldsymbol{B}^{\ 1}\boldsymbol{R}\} \end{aligned} \tag{3.2.20}$$

where $\boldsymbol{D}$ if necessary is adjusted to make it a symmetric matrix without changing the quadratic form. The equation (3.2.15) for quadratic problems then becomes

(3.2.21) $$z = z_0 \quad \boldsymbol{p}\boldsymbol{x}_R \quad \boldsymbol{x}'_R \boldsymbol{D}\boldsymbol{x}_R \ .$$

Once again both the basic feasible solution and the objective function are expressed in terms of the non-basic variables. Also note that no restrictions have been placed on the quadratic form, that is, Beale's algorithm is applicable to any quadratic programming problem without the objective function having to be convex (or concave in the case of maximization).

The function z is a function of the variables $\boldsymbol{x}_R \equiv (x_{R_1}, \ldots, x_{R_{n\ m}})$. For the initial solution $\boldsymbol{x}_R = 0$ and therefore $z = z_0$. Consider now the effects on z when changing the value of each of the non-basic variables in $\boldsymbol{x}_R$. The first partial derivative of (3.2.21) with respect to the non-basic variables x_{R_j} is

(3.2.22) $$\frac{\partial z}{\partial x_{R_j}} = p_j \quad 2\sum_{k=1}^{n\ m} d_{jk} x_{R_k}, \qquad j = 1, \ldots, n \quad m.$$

Initially every non-basic variable is equal to zero therefore

$$\frac{\partial z}{\partial x_{R_j}} = p_j, \qquad j = 1, \ldots, n \quad m.$$

The objective function can be decreased by increasing any one of the non-basic variables x_{R_j} for which p_j is negative. As with the simplex method, the aim is to obtain the optimal solution as fast as possible, therefore we bring into the basis the variable x_{R_h}, for which

$$p_h = \min_j (p_j).$$

As x_{R_h} begins to increase, the values of the variables $x_{B_1}, \ldots, x_{B_m}$ change in accordance with (3.2.18). In the case of the simplex method x_{R_h} is increased until one of the basic variables reaches zero, thereby completing the pivot. However, with a quadratic objective function it is possible that the partial derivative $\partial z / \partial x_{R_h}$ becomes zero during the pivot while all the basic variables x_{B_i} are still positive. In this case any further increase in x_{R_h} would worsen the value of the objective function rather that improve it. Now if $\partial z / \partial x_{R_h} > 0$ then any increase in x_{R_h} will not reduce the value of the objective function z. If $\partial z / \partial x_{R_h} < 0$ an increase in x_{R_h} will reduce z. Therefore the value of x_{R_h} cannot increase indefinitely, thus it is profitable to increase the variable x_{R_h} until either:

1. One has to stop to avoid making some basic variable negative, that is one of the basic variables becomes zero; or
2. $\partial z / \partial x_{R_h}$ becomes zero and is about to become positive.

Once the variable which is to become basic has been determined, the values of x_{R_h} at which conditions 1 and 2 occur are computed. These values are called *critical values*.

Before computing the critical values it is useful to represent the basic and non-basic variables in a tableau similar to the simplex tableau.

Basic Variable	$\boldsymbol{x}_B$	x_1	…	x_n
$x_{B_1} =$		y_{11}		y_{1n}
$x_{B_m} =$		y_{m1}		y_{mn}

The column $\boldsymbol{x}_B$ denotes the value of the basic variables and y_{ij} for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ denotes the coefficients of the variables for the constraints.

The critical values are now computed as follows. In condition 1 the value of x_{R_h} (denoted by $x_{R_h}^{(1)}$) at which x_{B_r} becomes zero is determined by the usual simplex condition as in (B.1.7), namely

$$x_{R_h}^{(1)} = x_{R_h} = \frac{x_{B_r}}{y_{rh}} = \min_i \left(\frac{x_{B_i}}{y_{ih}}, y_{ih} > 0 \right),$$

in order to reduce z. Here h denotes the column of the entering variable and r denotes the row of the smallest ratio.

In condition 2 because all the other non-basic variables $x_{R_j}, j \neq h$ remain zero, the equation (3.2.22) reduces to

$$\frac{\partial z}{\partial x_{R_h}} = p_h \quad 2 d_{hh} x_{Rk}.$$

Since $p_h < 0$, case 2 can only arise when $d_{hh} > 0$, in which case the partial derivative becomes zero when x_{R_h} (denoted by $x_{R_h}^{(2)}$ for case 2) reaches the value

$$x_{R_h}^{(2)} = x_{R_h} = \frac{p_h}{2 d_{hh}} .$$

Thus x_{R_h} will increase to $\min(x_{R_h}^{(1)}, x_{R_h}^{(2)})$ where $x_{R_h}^{(1)} = \infty$ if all $y_{ih} \leq 0$, and $x_{R_h}^{(2)} = \infty$ if $d_{hh} \leq 0$. If x_{R_h} can be increased indefinitely, that is $x_{R_h}^{(1)} = \infty$ and $x_{R_h}^{(2)} = \infty$, without causing either of the two cases to occur, then the problem has an unbound solution.

Suppose that case 1 occurs, that is x_{R_h} increases until $x_{R_h} = x_{R_h}^{(1)}$, then the θ-ratio is computed from the column $\boldsymbol{y}_h$ and the standard simplex pivot procedure is followed with x_{R_h} entering the basis and x_{B_r} leaving the basis. After the transformation has been completed the values of $\boldsymbol{x}_B$ and y_{ij} have been updated. However, $z_j \quad c_j$ can no longer be calculated as in linear programming. Instead $\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{A}$ and $\boldsymbol{Q}$ are repartitioned according to the basic and non-basic variables and the new values of $z_0, \boldsymbol{p}$ and $\boldsymbol{D}$ are recomputed. At this point the computational procedure is repeated.

Suppose that case 2 occurs, that is, the value of $x_{R_h}^{(2)} < x_{R_h}^{(1)}$, then $x_{R_h} = x_{R_h}^{(2)}$. At this point there are $m \quad 1$ variables positive, $x_{B_1}, \ldots, x_{B_m}$ and x_{R_h}. Clearly a basic solution to the constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ does not exist since the basis contains more then m variables. However $\partial z / \partial x_{R_h} = 0$, so the $m \quad 1$ variables can be considered as a basic solution to the $m \quad 1$ constraints

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

and

(3.2.23) $$u_g \equiv \frac{\partial z}{\partial x_{R_h}} = p_h \quad 2 \sum_{k=1}^{n \quad m} d_{hk} x_{R_k}.$$

The new variable u_g is introduced only for computational purposes and it is not required to be non-negative. For this reason it is called a *free variable*. The subscript g indicates the number of free variables introduced so far. The new constraint (3.2.23) can be rewritten in standard form as

(3.2.24) $$u_g \quad 2 \sum_{k=1}^{n \quad m} d_{hk} x_{R_k} = p_h.$$

The addition of this constraint means that an extra row is added to the tableau and it is added prior to the pivoting process. Thus with the introduction of u_g along with the variables x_{B_i} already in the basis, represents a basic feasible solution to the $m \quad 1$ constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\partial z / \partial x_{R_h} = 0$.

This new constraint $u_g = \partial z / \partial \tilde{x}_{R_h}$ is introduced to the tableau as follows.

Basic Variable	$\boldsymbol{x}_B$	x_1	…	x_n	u_g
$x_{B_1} =$		y_{11}		y_{1n}	0
					0
$x_{B_m} =$		y_{m1}		y_{mn}	0
$u_g =$	p_h	$y_{(m \quad 1)1}$		$y_{(m \quad 1)n}$	1

Before the transformation begins, u_g has the negative value p_h and is therefore the $m\ \ 1^{\text{th}}$ basic variable. The entries $y_{(m\ \ 1)1}$ to $y_{(m\ \ 1)n}$ are the coefficients of x_{R_k} in equation (3.2.24). Because none of the current basic variables appear in the new constraint, the new basis matrix $\boldsymbol{B}$ denoted by $\hat{\boldsymbol{B}}$ takes the form

$$\hat{\boldsymbol{B}}=\begin{bmatrix}\boldsymbol{B} & \mathbf{0}\\ \mathbf{0} & 1\end{bmatrix}.$$

The matrix $\boldsymbol{Q}$ is then also adjusted accordingly. The new value of the basic variables are

$$\hat{\boldsymbol{x}}_{\boldsymbol{B}}=\hat{\boldsymbol{B}}^{\ 1}\hat{\boldsymbol{b}}=\begin{bmatrix}\boldsymbol{B}^{\ 1} & \mathbf{0}\\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{b}\\ p_h\end{bmatrix}$$

therefore the constraint (3.2.24) can be added to the tableau without altering any of the values already present and without needing to do any calculations.

The non-basic variable x_{R_h} can now be brought into the basis. Because $x_{R_h}^{(2)}<x_{R_h}^{(1)}$ the variable selected by the simplex exit criteria must be the free variable u_g, that is, x_{R_h} will replace u_g in the basis. Pivoting now continues in the usual manner. After the transformation $x_{B_1},\ldots,x_{B_m},x_{R_h}$ are the basic variables and the next iteration begins. It is possible that at a later stage u_g may be brought back into the basis if $\partial z/\partial u_g$ is large enough in magnitude, either positive or negative. At this point the $p_h=\max_j(|p_j|)$ excluding those $p_j>0$ for which x_{R_j} is one of the variables $x_1,\ldots,x_n$ or a slack variable. In cases where $\partial z/\partial u_g$ is positive u_g will be increased from its non-basic value of zero to a positive level thereby causing z to decrease as desired. If u_g enters the basis, either negative or positive, it can be deleted together with its associated row from the tableau. This follows from the fact that it is a free variable and as such will never be selected to leave the basis or will in any way figure in the selection of the exiting variable because there is no need to prevent it from becoming positive.

The pivoting continues as described above with the size of the basis increasing or decreasing until a point is reached from which it is impossible to improve the value of the objective function. Therefore, the swapping of variables into and out of the basis continues until the following condition is satisfied.

$$\frac{\partial z}{\partial x_{R_j}}\begin{cases}\geq 0 & \text{if } x_{R_j} \text{ is one of the variables } x_1,\ldots x_n, \text{ or a slack variable,}\\ =0 & \text{if } x_{R_j} \text{ is a free variable.}\end{cases}$$

Beale proved that this method leads to the desired solution in finitely many steps provided that the following additional rule is used.

Supplementary Rule: If there are any non-basic free variables u for which $\partial z/\partial u_g \neq 0$, then the free variable must be removed from the non-basic set, that is the free variable is introduced into the basis at the next pivot. Only if the derivative of z with respect to all free variables is zero and hence if in this way a decrease in f is not possible, shall one of the original variables be introduced into the basis.

Degeneracy in Beale's algorithm

In case 1, when computing the value of $x_{R_h}^{(1)}$ it is possible that two or more critical values may be the same. Thus the introduction of the variable x_{R_h} into the basis will cause two or more of the basic variables to become zero and therefore give rise to degeneracy. Then, when some new variable x_{R_k} is chosen to enter the basis at the next iteration, it might happen that $x_{R_k}^{(1)}=0$. In this case it would not be possible to increase x_{R_k} at all. This could repeat over the next iterations and could occur through an endlessly repeating sequence of degenerate bases, which means the procedure will no longer be finite. This situation is however remote and can be disregarded in practice [Sim1975]. The technique of perturbation is discussed in detail in Appendix B.3 and can be used here to solve the problem of degeneracy [KTZ1971].

Example

Consider again the example as used in Wolfe's algorithm.

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = \; x_1 \;\; 2x_2 \;\; x_3 \;\; (\tfrac{1}{2}x_1^2 \;\; x_2^2 \;\; 2x_3^2 \;\; 2x_2x_3) \\ &\textit{Subject to} && x_1 \;\; 2x_2 \;\; 4x_3 \le 12 \\ & && 2x_1 \;\; x_2 \;\; 3x_3 \le 9 \\ & && x_1, x_2, x_3 \ge 0 \end{aligned}$$

Adding an additional slack variable the constraint becomes

$$\begin{aligned} &\textit{Subject to} && x_1 \;\; 2x_2 \;\; 4x_3 \;\; x_4 = 12 \\ & && 2x_1 \;\; x_2 \;\; 3x_3 \;\; x_5 = 9 \\ & && x_1, x_2, x_3, x_4, x_5 \ge 0 \end{aligned}$$

The initial basis solution is $\boldsymbol{x}_B = (x_4, x_5) = (12, 9)$ and can be represented in a tableau. The initial value of the objective function is zero.

	x_1	x_2	x_3	x_4	x_5
$x_{B_1}=x_4=12$	1	2	4	1	0
$x_{B_2}=x_5=9$	2	1	3	0	1

The problem is now partitioned according to the variables belonging to the basis and variables not in the basis.

$$\boldsymbol{x}=(\boldsymbol{x_B},\boldsymbol{x_R})=(x_4,x_5|x_1,x_2,x_3)=(12,9|0,0,0)$$

$$\boldsymbol{c}'=[\boldsymbol{c}'_{\boldsymbol{B}}|\boldsymbol{c}'_{\boldsymbol{R}}]=[0\ 0|\ \ 1\ \ 2\ \ 1]$$

$$\boldsymbol{A}=[\boldsymbol{B}|\boldsymbol{R}]=\begin{bmatrix}1 & 0 & 1 & 2 & 4\\ 0 & 1 & 2 & 1 & 3\end{bmatrix}$$

$$\boldsymbol{Q}=\begin{bmatrix}\boldsymbol{Q}_{BB} & \boldsymbol{Q}_{BR}\\ \boldsymbol{Q}_{RB} & \boldsymbol{Q}_{RR}\end{bmatrix}=\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & \frac{1}{2} & 0 & 0\\ 0 & 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1 & 2\end{bmatrix}$$

This can now be substituted into (3.2.20) to compute the values of z_0, $\boldsymbol{p}$ and $\boldsymbol{D}$

$$z_0=0\ ,\ \boldsymbol{p}=[\ \ 1\ \ \ \ 2\ \ \ \ 1]\ \text{and}\ \boldsymbol{D}=\begin{bmatrix}\frac{1}{2} & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 2\end{bmatrix}.$$

The matrix $\boldsymbol{D}$ is already symmetric so it does not need to be adjusted. Substituting the above results into (3.2.21)

$$\begin{aligned} z &= z_0\ \ \boldsymbol{p}\boldsymbol{x_R}\ \ \boldsymbol{x}'_{\boldsymbol{R}}\boldsymbol{D}\boldsymbol{x_R}\\ &= \ x_1\ \ 2x_2\ \ x_3\ \ (\tfrac{1}{2}x_1^2\ \ x_2^2\ \ 2x_3^2\ \ 2x_2x_3). \end{aligned}$$

To determine the entering variable the values of p_j are computed.

$$\frac{\partial z}{\partial x_{R_1}}=\frac{\partial z}{\partial x_1}=p_1=\ \ 1\ \ x_1=\ \ 1\ ,\ \frac{\partial z}{\partial x_{R_2}}=\frac{\partial z}{\partial x_2}=p_2=\ \ 2\ \ 2x_2\ \ 2x_3=\ \ 2$$

$$\text{and}\ \frac{\partial z}{\partial x_{R_3}}=\frac{\partial z}{\partial x_3}=p_3=\ \ 1\ \ 4x_3\ \ 2x_2=\ \ 1$$

Select $p_h=min(p_j)$ to enter the basis. In this case $h=2$, that is, x_2 is selected to become basic. The two critical points are now computed. The first critical point is given by

$$x_{R_2}^{(1)}=min\left\{\frac{x_{Bi}}{y_{i2}},\ y_{i2}<0\right\}=min\left\{\frac{12}{2},\frac{9}{1}\right\}=6.$$

However, the second critical value is

$$x_{R_2}^{(2)}=\frac{p_2}{2d_{22}}=\frac{2}{2(1)}=1$$

therefore x_2 will be increased to only 1 and neither of the current basic variables will be driven to zero. Since $x^{(2)}_{R_h} < x^{(1)}_{R_h}$ a new free variable u_1 is introduced to the problem.

$$u_1 = \frac{\partial z}{\partial x_{R_2}} = \quad 2 \quad 2x_2 \quad 2x_3 \text{ which can be written as } u_1 \quad 2x_2 \quad 2x_3 = \quad 2.$$

Before x_2 is increased u_1 must be added to the basis, where $u_1 = \quad 2$ as $x_2 = 0$ and $x_3 = 0$. The current tableau now looks as follows:

	x_1	x_2	x_3	x_4	x_5	u_1
$x_{B_1} = x_4 = 12$	1	2	4	1	0	0
$x_{B_2} = x_5 = 9$	2	1	3	0	1	0
$x_{B_3} = u_1 = \quad 2$	0	-2	2	0	0	1

Since the new free variable u_1 was introduced it must leave the basis and therefore x_2 replaces u_1 in the basis and the new tableau becomes:

	x_1	x_2	x_3	x_4	x_5	u_1
$x_{B_1} = x_4 = 10$	1	0	6	1	0	1
$x_{B_2} = x_5 = 8$	2	0	4	0	1	1/2
$x_{B_3} = x_2 = 1$	0	1	-1	0	0	-1/2

The new basic solution has been generated. The problem is again repartitioned.

$$\boldsymbol{x} = (\boldsymbol{x_B}, \boldsymbol{x_R}) = (x_4, x_5, x_2 | x_1, x_3, u_1) = (10, 8, 1 | 0, 0, 0)$$

$$\boldsymbol{c}' = [\boldsymbol{c}'_{\boldsymbol{B}} | \boldsymbol{c}'_{\boldsymbol{R}}] = [0 \quad 0 \quad 2 | \quad 1 \quad 1 \quad 0]$$

$$\boldsymbol{A} = [\boldsymbol{B} | \boldsymbol{R}] = \begin{bmatrix} 1 & 0 & 2 & 1 & 4 & 0 \\ 0 & 1 & 1 & 2 & 3 & 0 \\ 0 & 0 & 2 & 0 & 2 & 1 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{BB} & \boldsymbol{Q}_{BR} \\ \boldsymbol{Q}_{RB} & \boldsymbol{Q}_{RR} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The values of z_0, $\boldsymbol{p}$ and $\boldsymbol{D}$ are now computed for the second iteration to give

$$z_0 = \quad 1 \quad , \quad \boldsymbol{p} = [\quad 1 \quad 3 \quad 0] \text{ and } \boldsymbol{D} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{4} \end{bmatrix}.$$

As $\boldsymbol{D}$ is not symmetric it must be adjusted to

$$\boldsymbol{D}=\begin{bmatrix}\frac{1}{2} & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & \frac{1}{4}\end{bmatrix}.$$

The objective function is expressed as a function of non-basic variables

$$z=z_0 \quad \boldsymbol{p}\boldsymbol{x}_{\boldsymbol{R}} \quad \boldsymbol{x}'_{\boldsymbol{R}}\boldsymbol{D}\boldsymbol{x}_{\boldsymbol{R}}$$
$$= \quad 1 \quad x_1 \quad 3x_3 \quad \frac{1}{2}x_1^2 \quad x_3^2 \quad \frac{1}{4}u_1^2.$$

To determine the entering variable the values of p_j are calculated

$$\frac{\partial z}{\partial x_{R_1}}=\frac{\partial z}{\partial x_1}=p_1= \quad 1 \quad x_1= \quad 1 \text{ and } \frac{\partial z}{\partial x_{R_2}}=\frac{\partial z}{\partial x_3}=p_2= \quad 3 \quad 2x_3= \quad 3.$$

Chose x_3 to enter the basis as it is the $max\{p_j\}$. The critical points are

$$x_{R_2}^{(1)}=min\left\{\frac{x_{Bi}}{y_{i2}}, y_{i2}>0\right\}=min\left\{\frac{10}{6},\frac{8}{4}\right\}=\frac{5}{3}$$

and

$$x_{R_2}^{(2)}=\frac{p_2}{2q_{22}}=\frac{3}{2(1)}=\frac{3}{2}.$$

Therefore x_3 can be increased up to $\frac{3}{2}$. Since $x_{R_2}^{(2)}<x_{R_2}^{(1)}$ a new free variable u_2 is introduced.

$u_2=\frac{\partial z}{\partial x_{R_2}}=\frac{\partial z}{\partial x_3}= \quad 3 \quad 2x_3$ which can be rewritten as $u_2 \quad 2x_3= \quad 3.$

Before x_3 is increased u_2 must be added to the basis, where $u_2= \quad 3$ as $x_3=0$.

	x_1	x_2	x_3	x_4	x_5	u_1	u_2
$x_{B_1}=x_4=10$	1	0	6	1	0	1	0
$x_{B_2}=x_5=8$	2	0	4	0	1	1/2	0
$x_{B_3}=x_2=1$	0	1	-1	0	0	-1/2	0
$x_{B_4}=u_2= \quad 3$	0	0	-2	0	0	0	1

The free variable u_2 must leave the basis therefore bringing x_3 into the basis the new tableau becomes

	x_1	x_2	x_3	x_4	x_5	u_1	u_2
$x_{B_1}=x_4=1$	1	0	0	1	0	1	3
$x_{B_2}=x_5=2$	2	0	0	0	1	1/2	2
$x_{B_3}=x_2=\frac{5}{2}$	0	1	0	0	0	-1/2	-1/2
$x_{B_4}=x_3=\frac{3}{2}$	0	0	1	0	0	0	-1/2

Repartitioning the problem again

$$\boldsymbol{x}=(\boldsymbol{x}_{\boldsymbol{B}},\boldsymbol{x}_{\boldsymbol{R}})=(x_4,x_5,x_2,x_3|x_1,u_1,u_2)=(1,2,\tfrac{5}{2},\tfrac{3}{2}|0,0,0)$$
$$\boldsymbol{c}'=[\boldsymbol{c}'_{\boldsymbol{B}}|\boldsymbol{c}'_{\boldsymbol{R}}]=[0\ 0 \quad 2 \quad 1|\ \ 1\ 0\ 0]$$

$$\boldsymbol{A}=[\boldsymbol{B}\,|\,\boldsymbol{R}]=\begin{bmatrix}1&0&0&6&1&0&0\\0&1&0&4&2&\frac{1}{2}&0\\0&0&1&1&0&\frac{1}{2}&0\\0&0&0&2&0&0&1\end{bmatrix}$$

$$\boldsymbol{Q}=\begin{bmatrix}\boldsymbol{Q}_{BB}&\boldsymbol{Q}_{BR}\\\boldsymbol{Q}_{RB}&\boldsymbol{Q}_{RR}\end{bmatrix}=\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&1&0&0&0\\0&0&1&2&0&0&0\\0&0&0&0&\frac{1}{2}&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}.$$

Which after substitution gives

$$z_0=\frac{9}{4},\quad \boldsymbol{p}=[\,1\;0\;0\,]\text{ and }\boldsymbol{D}=\begin{bmatrix}\frac{1}{2}&0&0\\0&\frac{1}{4}&0\\0&0&\frac{1}{4}\end{bmatrix}.$$

As p_1 is negative the current solution is still not optimal and the objective function becomes

$$\begin{aligned}z&=z_0\quad \boldsymbol{p}\boldsymbol{x}_{\boldsymbol{R}}\quad \boldsymbol{x}'_{\boldsymbol{R}}\boldsymbol{D}\boldsymbol{x}_{\boldsymbol{R}}\\&=\frac{9}{4}\quad x_1\quad \frac{1}{2}x_1^2\quad \frac{1}{4}u_1^2\quad \frac{1}{4}u_2^2.\end{aligned}$$

Again determining p_j

$$\frac{\partial z}{\partial x_{R_1}}=\frac{\partial z}{\partial x_1}=p_1=\quad 1\quad x_1=\quad 1.$$

Clearly x_1 will enter the basis as p_1 is the only negative value negative. Calculating the critical points

$$x_{R_1}^{(1)}=min\left\{\frac{x_{Bi}}{y_{i1}},\,y_{i1}>0\right\}=min\left\{\frac{1}{1},\frac{2}{2}\right\}=1$$

and

$$x_{R_1}^{(2)}=\frac{p_1}{2q_{11}}=\frac{1}{2\left(\frac{1}{2}\right)}=1.$$

Bring x_1 into the basis

	x_1	x_2	x_3	x_4	x_5	u_1	u_2
$x_{B_1}=x_1=1$	1	0	0	1	0	1	3
$x_{B_2}=x_5=0$	0	0	0	-2	1	-1 1/2	-4
$x_{B_3}=x_2=\frac{5}{2}$	0	1	0	0	0	-1/2	-1/2
$x_{B_4}=x_3=\frac{3}{2}$	0	0	1	0	0	0	-1/2

Again repartition the problem into

$$\boldsymbol{x}=(\boldsymbol{x_B},\boldsymbol{x_R})=(x_1,x_5,x_2,x_3|x_4,u_1,u_2)=(1,0,\tfrac{5}{2},\tfrac{3}{2}|0,0,0)$$

$$\boldsymbol{c}'=[\boldsymbol{c}'_{\boldsymbol{B}}|\boldsymbol{c}'_{\boldsymbol{R}}]=[\ 1\ 0\quad 2\quad 1|0\ 0\ 0]$$

$$\boldsymbol{A}=[\boldsymbol{B}|\boldsymbol{R}]=\begin{bmatrix}1&0&0&0&1&1&3\\2&1&0&0&0&\frac{1}{2}&2\\0&0&1&0&0&\frac{1}{2}&\frac{1}{2}\\0&0&0&1&0&0&\frac{1}{2}\end{bmatrix}$$

$$\boldsymbol{Q}=\begin{bmatrix}\boldsymbol{Q}_{BB}&\boldsymbol{Q}_{BR}\\\boldsymbol{Q}_{RB}&\boldsymbol{Q}_{RR}\end{bmatrix}=\begin{bmatrix}\frac{1}{2}&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&1&0&0&0\\0&0&1&2&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}.$$

which gives the values

$$z_0=\ 3\tfrac{3}{4}\ ,\ \boldsymbol{p}=[0\ 0\ 0]\ \text{and}\ \boldsymbol{D}=\begin{bmatrix}\frac{1}{2}&\frac{1}{2}&\frac{3}{2}\\\frac{1}{2}&\frac{3}{4}&\frac{3}{2}\\\frac{3}{2}&\frac{3}{2}&\frac{19}{4}\end{bmatrix}.$$

Examining the components of $\boldsymbol{p}$ and the termination criteria of Beale's algorithm has been satisfied since

$$\frac{\partial z}{\partial x_{R_1}}=\frac{\partial z}{\partial x_4}=p_1=0\ ,\ \frac{\partial z}{\partial x_{R_2}}=\frac{\partial z}{\partial u_1}=p_2=0\ \text{and}\ \frac{\partial z}{\partial x_{R_3}}=\frac{\partial z}{\partial u_2}=p_2=0\ .$$

Therefore the optimal solution has been found at $x_1=1, x_2=2\frac{1}{2}, x_3=1\frac{1}{2}$ and the value of the objective function is $z=f(\boldsymbol{x})=\ 3\frac{3}{4}$.

3.2.3 Theil-Van de Panne Procedure

This procedure was first introduced by H. Theil and C. Van de Panne in 1960 [TV1960]. The procedure deals with strictly convex (for minimization) and concave (for maximization) quadratic programming problems, where the matrices $\boldsymbol{c, Q, A}$ and $\boldsymbol{b}$ are given and $\boldsymbol{Q}$ is a positive definite symmetric matrix, that is

(3.2.25)
$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\text{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad \boldsymbol{x} \ge 0 \end{aligned}$$

In this algorithm the non-negativity constraints are considered to be part of the constraints and therefore the problem can be reformulated in the following form.

(3.2.26) $\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}$

(3.2.27) $\text{Subject to} \quad \hat{\boldsymbol{A}}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{x} \le \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} = \hat{\boldsymbol{b}}$

The idea of this procedure is to minimize (3.2.26) without taking into account any of the constraints and then using the result as a basis for further computation. In order to compute the vector that minimizes (3.2.26) without regard to any constraints, differentiate $f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \frac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}$ with respect to $\boldsymbol{x}$ and set the result equal to zero. This yields

(3.2.28) $$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{c} \quad \boldsymbol{Q}\boldsymbol{x} = 0.$$

Since $\boldsymbol{Q}$ is positive definite, its inverse exists, thus

$$\boldsymbol{x} = \boldsymbol{x} \;=\; \boldsymbol{Q}^{\;1}\boldsymbol{c}.$$

Let $S \quad \{1,\ldots,m\}$ and let $\boldsymbol{x}^S$ be a solution to the following reduced problem

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} \quad \hat{\boldsymbol{A}}_S\boldsymbol{x} = \hat{\boldsymbol{b}}_S \end{aligned}$$

Consider the Lagrangian function $L(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x}) \quad (\boldsymbol{\lambda})'g(\boldsymbol{x})$ and $g(\boldsymbol{x}) = \hat{\boldsymbol{A}}_S\boldsymbol{x} \quad \hat{\boldsymbol{b}}_S$, then

(3.2.29) $$L(\boldsymbol{x},\boldsymbol{\lambda}) = \boldsymbol{c}'\boldsymbol{x} \quad \frac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \quad (\boldsymbol{\lambda}^S)'(\hat{\boldsymbol{A}}_S\boldsymbol{x} \quad \hat{\boldsymbol{b}}_S).$$

Differentiate with respect to $\boldsymbol{x}$ and with respect to $\boldsymbol{\lambda}$ and equate to zero to get

(3.2.30) $$\boldsymbol{c} \quad \boldsymbol{Q}\boldsymbol{x}^S \quad \hat{\boldsymbol{A}}'_S\boldsymbol{\lambda}^S = 0 \quad \text{and}$$

(3.2.31) $$\hat{\boldsymbol{A}}_S\boldsymbol{x}^S = \hat{\boldsymbol{b}}_S.$$

From (3.2.30) we get

(3.2.32) $$\boldsymbol{x}^S = \quad \boldsymbol{Q}^{\;1}\boldsymbol{c} \quad \boldsymbol{Q}^{\;1}\hat{\boldsymbol{A}}'_S\boldsymbol{\lambda}^S = \boldsymbol{x} \quad \boldsymbol{Q}^{\;1}\hat{\boldsymbol{A}}'_S\boldsymbol{\lambda}^S.$$

To find $\boldsymbol{\lambda}^S$ multiply (3.2.32) by $\hat{\boldsymbol{A}}_S$, which gives using (3.2.31)

(3.2.33) $$\hat{\boldsymbol{b}}_S = \hat{\boldsymbol{A}}_S\boldsymbol{x}^S = \hat{\boldsymbol{A}}_S\boldsymbol{x} \quad \hat{\boldsymbol{A}}_S\boldsymbol{Q}^{\;1}\hat{\boldsymbol{A}}'_S\boldsymbol{\lambda}^S.$$

The matrix $\hat{\boldsymbol{A}}_S\boldsymbol{Q}^{\;1}\hat{\boldsymbol{A}}'_S$ appears often in the later expressions and therefore the following notation is adopted

(3.2.34) $$\boldsymbol{P}_S = \hat{\boldsymbol{A}}_S \boldsymbol{Q}^{-1} \hat{\boldsymbol{A}}'_S .$$

If $\hat{\boldsymbol{A}}_S$ has a full row rank, then the matrix $\boldsymbol{P}_S$ is a symmetric and positive definite matrix, and therefore has an inverse. By this it is meant that the number of independent rows of $\boldsymbol{P}_S$ is k if $\boldsymbol{P}_S$ is an k x k matrix. From (3.2.33)

(3.2.35) $$\boldsymbol{\lambda}^S = \boldsymbol{P}_S^{-1}(\hat{\boldsymbol{A}}_S \boldsymbol{Q}^{-1} \boldsymbol{c} \quad \hat{\boldsymbol{b}}_S) = \boldsymbol{P}_S^{-1}(\hat{\boldsymbol{A}}_S \boldsymbol{x} \quad \hat{\boldsymbol{b}}_S).$$

By substituting (3.2.35) into (3.2.32) an expression for $\boldsymbol{x}^S$ is obtained in terms of $\boldsymbol{c}, \boldsymbol{Q}, \hat{\boldsymbol{A}}_S$ and $\hat{\boldsymbol{b}}_S$,

(3.2.36) $$\boldsymbol{x}^S = \boldsymbol{x} \quad \boldsymbol{Q}^{-1} \hat{\boldsymbol{A}}'_S (\boldsymbol{P}_S^{-1})(\hat{\boldsymbol{A}}_S \boldsymbol{x} \quad \hat{\boldsymbol{b}}_S).$$

The computational procedure now proceeds as follows. First, begin by considering an empty set in which case the set $S =$, and the vector $\boldsymbol{x} = \boldsymbol{Q}^{-1} \boldsymbol{c}$, which is a vector that minimizes (3.2.26) without regard to any constraints. If at this point $\boldsymbol{x}$ does not violate any of the constraints, the optimal solution has been found. Usually however, the problem arises that $\boldsymbol{x}$ is not feasible, that is $\hat{\boldsymbol{A}}\boldsymbol{x} = \hat{\boldsymbol{A}}\boldsymbol{Q}^{-1}\boldsymbol{c} \not\leq \hat{\boldsymbol{b}}$.

Consider all one-element sets S consisting of constraints violated by $\boldsymbol{x}$. Say for example constraint h is violated by $\boldsymbol{x}$, then the set S has one element, that is $S = \{h\}$. Now use the equation (3.2.36) to calculate the value for $\boldsymbol{x}^S$. Suppose that none of the resulting vectors $\boldsymbol{x}^S$ are feasible, then consider $\boldsymbol{x}^S$ for which the set S contains two elements say for example $S = \{h, i\}$. Then constraint h is violated by $\boldsymbol{x}$ and constraint i is violated by $\boldsymbol{x}^h$ where $\boldsymbol{x}^h = \boldsymbol{x}^S$ for $S = \{h\}$. If none of the resulting vectors are feasible, consider the vector $\boldsymbol{x}^S$ for which the set S has three elements say for example $S = \{h, i, j\}$ where j is a constraint violated by $\boldsymbol{x}^{hi}$ where $\boldsymbol{x}^{hi} = \boldsymbol{x}^S$ for $S = \{h, i\}$. Proceed in this way until a feasible vector has been found. This means that for the particular $\boldsymbol{x}^S$ none of the constraints are violated.

Now suppose that the set S has been found, for which the vector $\boldsymbol{x}^S$ does not violate any constraints, that is, it is feasible, thus it remains to show if the vector $\boldsymbol{x}^S$ solves the problem (3.2.26)-(3.2.27) or not. The feasible vector $\boldsymbol{x}^S$ is the solution vector if and only if for all $h \quad S$ the vector $\boldsymbol{x}^{S \quad h}$ violates the constraint h (where the notation $S \quad h$ indicates a set S with the h^{th} constraint removed). This result follows from Theorem 3.2. If the vector $\boldsymbol{x}^{S \quad h}$ does not violate the constraint h, this solution point is not the solution vector. This means that there exist another feasible point which may be the solution point. In this case the sets S for the two points will be different.

Theorem 3.2

Let $\hat{S}$ be a set of indices of the constraints that $\hat{\boldsymbol{x}}$ satisfies as equalities, that is

$$\boldsymbol{A}\hat{\boldsymbol{x}} = b_j \quad \text{for } j \quad \hat{S}$$
$$\text{and } \boldsymbol{A}\hat{\boldsymbol{x}} < b_j \quad \text{for } j \quad \hat{S}$$
$$\text{then } \hat{\boldsymbol{x}} = \boldsymbol{x}^{\hat{S}}$$

If $\hat{\boldsymbol{x}} = \boldsymbol{x}^{\hat{S}}$ and $\hat{S} \neq$, then for all proper subsets S $\hat{S}$ including $S =$ we have h as one of the constraints violated by $\boldsymbol{x}^{S}$ for at least one h $(\hat{S} \quad S)$ [KKO1966].

There is another way to check if a solution has been found at each iteration. The criteria on which the decision is based is not whether the constraints are violated or not, but rather consists of considering the signs of the Lagrangians associated with the binding constraints [Boo1964]. This procedure has computational advantages as we do not need to check for violated constraints at each iteration and will now be briefly discussed.

Write Z for the subset of constraints complementary to S. Consider the set S of constraints that are satisfied exactly. Add a constraint r from Z to this set to get the set $S \quad r$ with $\boldsymbol{x}^{S \quad r}$ as the associated minimizing vector, $\boldsymbol{\lambda}^{S \quad r}$ as the Lagrangian vector and $\boldsymbol{\lambda}_r^{S \quad r}$ as the Lagrangian associated with the r^{th} constraint. Conversely, consider the set S and delete a constraint t from this set. Thus we get a new set $S \quad t$, with $\boldsymbol{x}^{S \quad t}$ as the associated minimizing vector and $\boldsymbol{\lambda}^{S \quad t}$ as the Lagrangian vector. The constraints r and t are written as $\hat{\boldsymbol{A}}_r \boldsymbol{x} \{<,=,>\} \hat{\boldsymbol{b}}_r$ and $\hat{\boldsymbol{A}}_t \boldsymbol{x} \{<,=,>\} \hat{\boldsymbol{b}}_t$, depending whether the constraints are sufficiently satisfied by $\boldsymbol{x}$, exactly satisfied or violated by $\boldsymbol{x}$. The following theorem shows the connection between the signs of the Lagrangians and the violation of constraints.

Theorem 3.3 [Boo1964]

Consider complementary sets S and Z of constraints, a constraint $r \quad Z$, and a constraint $t \quad S$, such that $(\hat{\boldsymbol{A}}'_{S,}\hat{\boldsymbol{A}}'_{r})'$ has a full row rank. Then

(1) $\hat{\boldsymbol{A}}_r \boldsymbol{x}^S > \hat{\boldsymbol{b}}_r$ *if and only if* $\boldsymbol{\lambda}_r^{S\ r} > 0$

(2) $\hat{\boldsymbol{A}}_r \boldsymbol{x}^S = \hat{\boldsymbol{b}}_r$ *if and only if* $\boldsymbol{\lambda}_r^{S\ r} = 0$

(3) $\hat{\boldsymbol{A}}_r \boldsymbol{x}^S < \hat{\boldsymbol{b}}_r$ *if and only if* $\boldsymbol{\lambda}_r^{S\ r} < 0$

and

(1') $\boldsymbol{\lambda}_t^S > 0$ *if and only if* $\hat{\boldsymbol{A}}_t \boldsymbol{x}^{S\ t} > \hat{\boldsymbol{b}}_t$

(2') $\boldsymbol{\lambda}_t^S = 0$ *if and only if* $\hat{\boldsymbol{A}}_t \boldsymbol{x}^{S\ t} = \hat{\boldsymbol{b}}_t$

(3') $\boldsymbol{\lambda}_t^S < 0$ *if and only if* $\hat{\boldsymbol{A}}_t \boldsymbol{x}^{S\ t} < \hat{\boldsymbol{b}}_t$.

The statements above can be explained as follows. For example, statement (1) reads: If the vector $\boldsymbol{x}^S$ violates the r^{th} constraint, then the Lagrangian associated with the r^{th} constraint is positive and vice versa. Statement (3') reads: If the Lagrangian attached to constraint t is negative, then it can be deleted from S, to get $\boldsymbol{x}^{S\ t}$, which will then sufficiently satisfy the deleted constraint and vice versa. The proof of the statements is given in Appendix B.5.

The following results are used to calculate the values of the Lagrangians and are obtained from the proof of the Theorem 3.3 [Boo1964] given in Appending B.5.

(3.2.37) $$\boldsymbol{\lambda}_r^{S\ r} = \frac{1}{\gamma \quad \boldsymbol{p}'\boldsymbol{P}_S^{\ 1}\boldsymbol{p}}(\hat{\boldsymbol{A}}_r \boldsymbol{x}^S \quad \hat{\boldsymbol{b}}_r)$$

where $\boldsymbol{p}' = \hat{\boldsymbol{A}}_r \boldsymbol{Q}^{\ 1}\hat{\boldsymbol{A}}'_S$, $\boldsymbol{P}_S = \hat{\boldsymbol{A}}_S \boldsymbol{Q}^{\ 1}\hat{\boldsymbol{A}}'_S$ and $\gamma = \hat{\boldsymbol{A}}_r \boldsymbol{Q}^{\ 1}\hat{\boldsymbol{A}}'_r$

(3.2.38) $$\hat{\boldsymbol{A}}_r \boldsymbol{x}^S = \hat{\boldsymbol{A}}_r \boldsymbol{x} \quad \hat{\boldsymbol{A}}_r \boldsymbol{Q}^{\ 1}\hat{\boldsymbol{A}}'_S \boldsymbol{\lambda}^S = \hat{\boldsymbol{A}}_r \boldsymbol{x} \quad \boldsymbol{p}'\boldsymbol{\lambda}^S$$

and

(3.2.39) $$\boldsymbol{\lambda}_t^S = \delta(\hat{\boldsymbol{A}}_t \boldsymbol{x}^{S\ t} \quad \hat{\boldsymbol{b}}_t)$$

where δ is the last diagonal element of $\boldsymbol{P}_S^{\ 1}$.

Thus to compute the solution using the signs of the Lagrangians, the procedure begins with calculating $\boldsymbol{x}$. If at this point none of the constraints are violated, the optimal solution has been found. If some constraint is violated, say for example constraint h then the value of the Lagrangian $\boldsymbol{\lambda}_h^h$ is calculated. If the Lagrangian is positive the optimal solution has been found. If not, assume the constraint i has been violated. Calculate the Lagrangians $\boldsymbol{\lambda}_h^{h,i}$ and $\boldsymbol{\lambda}_i^{h,i}$. If both Lagrangians are positive the solution has been found, otherwise calculate the

Lagrangians for the next violated constraint. Thus at every solution point found the Lagrangians are computed. Quite generally, if any vector obtained is feasible and has all associated Lagrangians positive we have a solution to the problem.

Theil-Van de Panne in absence of degeneracy

Degeneracy occurs when the solution to a programming problem is the same, whether some specific equality constraint is imposed or not. In the absence of degeneracy, the cases (2) and (2') will not occur in the solution. Let the solution vector $\hat{\boldsymbol{x}}$ minimizing $f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \frac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}$ subject to $\hat{\boldsymbol{A}}\boldsymbol{x} \le \hat{\boldsymbol{b}}$ be found by minimizing $f(\boldsymbol{x})$ subject to certain subset $\hat{S} \quad \{1, \ldots, m\}$ of constraints. The theorems on which the Theil-Van de Panne procedure is based on are

Theorem 3.4 Sufficiency Theorem:
If $\boldsymbol{x}^S$ *is feasible and* $\boldsymbol{\lambda}^S > 0$ *then* $\boldsymbol{x}^S = \hat{\boldsymbol{x}}$ [Boo1964].

and

Theorem 3.5 Necessity Theorem:
If $\boldsymbol{x}^S$ *is feasible but* $\boldsymbol{\lambda}_t^S < 0$ *for some* $t \quad S$ *then* $\boldsymbol{x}^S \neq \hat{\boldsymbol{x}}$ [Boo1964].

Both of these theorems are valid only in the absence of degeneracy and will not be proved in the text. See [Boo1964] for details and the proof of the theorems. These theorems together prove that $\boldsymbol{x}^S = \hat{\boldsymbol{x}}$, and $S = \hat{S}$, if and only if the set S is such that $\boldsymbol{x}^S$ is feasible and $\boldsymbol{\lambda}^S > 0$. The set S is generated using the procedure explained earlier in calculating the solution points and then finding the violated constraints at each solution point.

Theil-Van de Panne with degeneracy

Degeneracy occurs when a constraint $r \quad Z$ happens to be exactly satisfied $\hat{\boldsymbol{A}}_r \boldsymbol{x}^S = \hat{\boldsymbol{b}}_r$ or an element $\boldsymbol{\lambda}_t^S$ of $\boldsymbol{\lambda}^S$ happens to be zero. These are cases (2) and (2'). In the case of degeneracy the following theorem holds

Theorem 3.6 Degeneracy Theorem: [Boo1964]

Consider complementary sets S and Z of constraints, as defined before, with a constraint $r \in Z$ and a constraint $t \in S$ such that $(\hat{\boldsymbol{A}}'_S \hat{\boldsymbol{A}}'_r)'$ has full row rank. Then

i) if $\hat{\boldsymbol{A}}_r \boldsymbol{x}_S = \hat{\boldsymbol{b}}_r$ then $\boldsymbol{x}^S = \boldsymbol{x}^{S \cup r}$

ii) if $\boldsymbol{\lambda}_t^S = 0$ then $\boldsymbol{x}^S = \boldsymbol{x}^{S \setminus t}$.

In the case of degeneracy the procedure of finding the solution point is the same as explained earlier. That is, calculate a solution point and determine any violated constraints until for some set S, say $\hat{S}$ a feasible vector $\boldsymbol{x}^{\hat{S}}$ with all Lagrangians $\boldsymbol{\lambda}^{\hat{S}}$ positive has been found. In the case of degeneracy, $\boldsymbol{x}^{\hat{S}}$ will bind not only all the constraints belonging to $\hat{S}$, but some other constraints as well. The degeneracy will not be noticed because an element of $\boldsymbol{\lambda}^{\hat{S}}$ is zero. If and element $\boldsymbol{\lambda}_h^{\hat{S}} = 0$ then $\boldsymbol{x}^{\hat{S}} = \boldsymbol{x}^{\hat{S} \setminus h}$, where the notation $\hat{S} \setminus h$ indicates a set $\hat{S}$ with the h^{th} constraint removed. Since the set $\hat{S}$ goes from a smaller set to a larger set we would have come across $\boldsymbol{x}^{\hat{S} \setminus h}$ before getting to $\boldsymbol{x}^{\hat{S}}$. Thus by Theorem 3.6 all the constraints for which the associated Lagrangians equal zero can be deleted from $\hat{S}$ and then the smallest set is unique.

The perturbation technique mentioned in the previous algorithms and proved in Appendix B.3, can also be used to deal with problems in which degeneracy occurs.

Example

(3.2.40) *Minimize* $f(\boldsymbol{x}) = x_1 \quad 2x_2 \quad x_3 \quad \frac{1}{2}(x_1^2 \quad 2x_2^2 \quad 4x_3^2 \quad 4x_2x_3)$

Subject to $x_1 \quad 2x_2 \quad 4x_3 \le 12$ (1)

$2x_1 \quad x_2 \quad 3x_3 \le 9$ (2)

$x_1 \le 0$ (3)

$x_2 \le 0$ (4)

$x_3 \le 0$ (5)

For the problem (3.2.40)

$$\boldsymbol{c} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 4 \end{bmatrix} \quad \text{and} \quad \boldsymbol{Q}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

The first step is to calculate $\boldsymbol{x}$.

(3.2.41) $$\boldsymbol{x} = \boldsymbol{Q}^{-1}\boldsymbol{c} = \begin{bmatrix} 1 \\ \frac{5}{2} \\ \frac{3}{2} \end{bmatrix}$$

Using the result of (3.2.41) the vector $\boldsymbol{x}$ does not violate any of the constraints, thus the optimal solution has been found in the first iteration. This solution corresponds with the solutions obtained in the previous algorithms for the same example. Thus the optimal solution is $f(\boldsymbol{x}) = 3\frac{3}{4}$. It is purely coincidental that the solution was reached in the initial step. For a more detailed and illustrative example of the Theil-Van de Panne procedure refer to Appendix B.6.

3.2.4 Other Active Set Methods

There are many more active set algorithms for solving the quadratic programming problem. It is not possible to describe all the algorithms available and thus only some of the algorithms have been selected for a detailed explanation. Some of the other algorithms include amongst others: Houthakker's algorithm introduced by H. S. Houthakker in 1960 [Hou1960], Fletcher's algorithm proposed in 1971 [BSS1993]; Goldfab's algorithm in 1983 [Mil2000]; The Weighted Gram-Schmidt method proposed in 1984 by P.E. Gill, N. I. M. Gould, W. Murray, M. A. Saunders and M. H. Wright [GGM+1984];

3.3 Ellipsoid Methods

This method was proposed in 1972 by A.S. Nemirovsky, D. B. Yudin and N. Z. Shor [VBW2002]. The ellipsoid method was the first known method that could solve linear programming problems in polynomial time. However it was shown that it is much slower than the simplex method [VB1994] [CR1999].

The idea of the ellipsoid method is to start with some ellipsoid E_0 large enough to contain the feasible region F. At each iteration of the algorithm and new ellipsoid E_k is created that is smaller than the previous but still contains the feasible region. For each iteration k of the algorithm

Determine c_k which is the center of E_k;

The method checks if the center c_k of the current ellipsoid is feasible. If this is the case the procedure stops with c_k as the solution. If not, then at least one of the constraints is violated;

Determine a_k, the constraint most violated by c_k;

Use the a_k hyperplane to split E_k into two regions R_1 and R_2. One of the regions, say R_1 contains c_k and the other, R_2 contains the feasible region; and

A new ellipsoid $E_{k\ 1}$ with a minimum volume is constructed such that is contains all the points in $E_k \cap R_2$ but that is smaller that E_k.

Figure 3.2 illustrates these steps

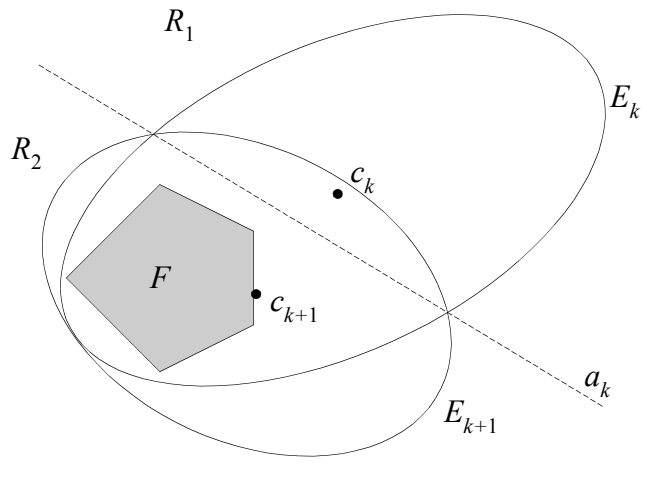

Figure 3.2 Ellipsoid method

It has been proved that the ellipsoid methods are slower than simplex methods and do not work very well in practice [Gou2002] and will therefore not be discussed further in this dissertation.

3.4 Interior Point Methods

The simplex method is based on the observation that an optimal solution occurs at one or more corners of the feasibility region of an n-dimensional solution space, where n is the number of decision variables. The interior point method gets its name from the fact that the solution points move through the interior of the feasible region towards the optimum rather

than along the boundary, as in the simplex method [And1998]. The first interior point like method was developed by I. Dikini in 1967, when he introduced a method called the affine scaling method for solving linear programming problems. This method however, did not receive much attention and it is no longer used for practical implementations [Van2001]. The first interior point method to receive interest was introduced in 1984 by N. K. Karmarkar [Kar1984]. Karmarkar's paper motivated many researchers to develop better interior point methods for linear programming. It was reported by D. den Hertog [Her1994] that there was about 2,000 papers dealing with interior point methods 8 years after Karmarkar's initial publication in 1984 [Mil2000].

As mentioned in Section 3.2, the simplex method is an exponential time algorithm. The ellipsoid method was the first polynomial time algorithm introduced, however, this proved not to be the case in practice. The interior point method attempts to overcome the potential weakness of the simplex method and it has been shown that the interior point methods can be solved in polynomial time [Mil2000]. This implies that if a problem of n variables is solved using the interior point method, there exist positive numbers a and b such that for any n , the problem can be solved in a time of at most $a n^b$.

The interior point methods can be classified as either affine scaling methods or path following methods. The affine scaling method rescales the components of the solution vector at each iteration and then moves in the appropriate direction towards the optimum point. The affine scaling algorithms are believed not to be polynomial, however this has not been proved conclusively [Gor1999]. The path following method falls under the category of the so called *barrier methods*, which incorporate the constraints into the objective function as logarithmic terms to remove the inequalities from the problem. The resulting function becomes a function of a so called *barrier parameter*. This parameter is decreased at each iteration. The solution points form a so called *central path* through the interior of the region towards the optimum point. Both the affine scaling methods and path following methods result in a "path" from the initial point to the optimal solution and therefore virtually all approaches using the interior point method could be called path following methods and therefore path following methods will be discussed in detail [Mil2000].

3.4.1 Path Following Algorithm

This section describes a path following interior point method [Van1994] [BJR+1997] [Van2001] [GW2001]. Consider the quadratic programming problem.

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad \boldsymbol{x} \ge 0 \end{aligned}$$

The inequality constraints are changed to equality constraints by adding slack variables $\boldsymbol{y}$ to $\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$. The problem now becomes

(3.4.1) $$\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x}$$

(3.4.2) $$\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \quad \boldsymbol{y} = \boldsymbol{b}$$

(3.4.3) $$\boldsymbol{x}, \boldsymbol{y} \ge 0.$$

To solve the quadratic programming problem, the constrained problem is transformed into a sequence of unconstrained problems by incorporating the constraints (3.4.2) and (3.4.3) as logarithmic terms.

The concept of the barrier approach is to start from a point $\boldsymbol{x}$ in the strict interior of the inequalities, that is, $\boldsymbol{x} > 0$ and $\boldsymbol{y} > 0$. Assuming such a point has been found that satisfies the constraints, to solve (3.4.1)-(3.4.3), a new point is obtained that hopefully decreases the objective function value while ensuring that the boundary $x_j = 0$ and $y_i = 0$ of the feasible region is never crossed. One technique to prevent an optimization algorithm from crossing the boundary is to assign a penalty to approaching it. The most popular method is to augment the objective function by a logarithmic barrier term.

Subtracting the terms $\log(x_j)$ from the objective function, will cause the objective to increase without bound as $x_j \to 0$, that is, $\log(\boldsymbol{x}) \to \infty$ as $\boldsymbol{x} \to 0$. This creates the problem that if the constrained optimal solution is on the boundary, that is one or more $x_j^* = 0$, then the barrier will prevent us from reaching it. To get around this difficulty, a barrier parameter $\mu \ge 0$ is introduced to control the magnitude of the barrier term. As the solution to the problem is likely to lie on the boundary of the feasible region, the barrier must be removed gradually by decreasing μ toward zero. How this parameter is calculated will be discussed at a later stage. Because the function $\log(\boldsymbol{x}) = \infty$ when $\boldsymbol{x} < 0$, it is required that $\boldsymbol{x} > 0$. As long as $\boldsymbol{x}$ remains positive the optimal solution to the barrier problems will be in the interior of the non-negativity constraint $\boldsymbol{x} > 0$. The barrier terms are introduced as follows

(3.4.4)
$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{c}'\boldsymbol{x} \;\; \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \;\; \mu\sum_j \log x_j \;\; \mu\sum_i \log y_i \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \;\; \boldsymbol{y} = \boldsymbol{b}. \end{aligned}$$

The new objective function is called a *barrier function* or a *logarithmic barrier function*.

The new formulation (3.4.4) has a nonlinear objective function with linear equality constraints, and can therefore be solved using the Lagrange multipliers, resulting in the following objective function

(3.4.5)
$$\begin{aligned} \textit{Minimize}\, f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda}) = \boldsymbol{c}'\boldsymbol{x} \;\; \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \;\; \mu\sum_{j=1}^{n} \log x_j \;\; \mu\sum_{i=1}^{m} \log y_i \\ \boldsymbol{\lambda}'(\boldsymbol{b} \;\; \boldsymbol{A}\boldsymbol{x} \;\; \boldsymbol{y}). \end{aligned}$$

Differentiating (3.4.5) with respect to $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{\lambda}$ and setting it to zero, yields the following set of equations.

(3.4.6)
$$\begin{aligned} \boldsymbol{c} \;\; \boldsymbol{Q}\boldsymbol{x} \;\; \boldsymbol{A}'\boldsymbol{\lambda} \;\; \mu\boldsymbol{x}^{-1} = 0 \\ \mu\boldsymbol{y}^{-1} \;\; \boldsymbol{\lambda} = 0 \\ \boldsymbol{b} \;\; \boldsymbol{A}\boldsymbol{x} \;\; \boldsymbol{y} = 0 \\ \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda} \geq 0 \end{aligned}$$

where $\boldsymbol{x}^{-1} = (x_1^{-1}, \ldots, x_n^{-1})'$ and $\boldsymbol{y}^{-1} = (y_1^{-1}, \ldots, y_n^{-1})'$.

We now introduce the following non-standard notation. This notation, has been extensively used in explaining the interior point method and therefore will also be used in this dissertation.

Given a vector $\boldsymbol{x} = (x_1, \ldots, x_n)'$, then capital $\boldsymbol{X}$ will denote the diagonal matrix with $\boldsymbol{x}$ on the main diagonal, that is

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} x_1 & 0 & & 0 \\ 0 & x_2 & & \\ & & & 0 \\ 0 & & 0 & x_n \end{bmatrix},$$

with $\boldsymbol{X}^{-1}$ and $\boldsymbol{Y}^{-1}$ the inverses of $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively. Further let $\boldsymbol{e} = (1, 1, \ldots, 1)'$, then using this notation, the set of equations (3.4.6) can be written as

(3.4.7) $$\boldsymbol{c} \;\; \boldsymbol{Q}\boldsymbol{x} \;\; \boldsymbol{A}'\boldsymbol{\lambda} \;\; \mu\boldsymbol{X}^{-1}\boldsymbol{e} = 0$$

(3.4.8) $$\mu\boldsymbol{Y}^{-1}\boldsymbol{e} \;\; \boldsymbol{\lambda} = 0$$

(3.4.9) $$\boldsymbol{b} \;\; \boldsymbol{A}\boldsymbol{x} \;\; \boldsymbol{y} = 0$$

$$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda} \geq 0.$$

Let

(3.4.10) $$w = \mu X^{-1} e \quad \text{with} \quad w \geq 0 .$$

Substituting (3.4.10) into (3.4.7) and multiplying equations (3.4.8) and (3.4.10) by Y and X respectively, the system (3.4.7)-(3.4.9) is rewritten as

(3.4.11) $$\begin{aligned} Qx - A'\lambda - w &= c \\ Ax - y &= b \end{aligned}$$

(3.4.12) $$\Lambda Y e = \mu e$$

(3.4.13) $$XW e = \mu e .$$

It can be seen that equations (3.4.12) and (3.4.13) are similar to the non-linear equations produced by applying the Kuhn-Tucker conditions to the quadratic programming problem.

In general the path following method proceeds as follows. The method begins by identifying an initial solution $(x^0, y^0, \lambda^0, w^0)$ that satisfies the equations (3.4.11) and the non-negativity constraints strictly, that is, $(x, y, \lambda, w) > 0$. However, it does not necessarily satisfy equations (3.4.12) and (3.4.13). The method then iteratively updates the values of x, y, λ and w, such that, the new solution satisfies the same conditions and gets progressively closer to satisfying conditions (3.4.12) and (3.4.13). This is done as follows:

1. Estimate the appropriate value of μ.
2. Compute the step direction $(\Delta x, \Delta y, \Delta\lambda, \Delta w)$, that is, the direction along which to move to the next point.
3. Calculate the distance or step length parameter by which to move to the next point.
4. Replace the current solution $(x^k, y^k, \lambda^k, w^k)$ with the new solution $(x^{k+1}, y^{k+1}, \lambda^{k+1}, w^{k+1})$ where k defines the k^{th} iteration.

The computations required for calculating the step direction will now be discussed. To compute the direction for the next step, (x, y, λ, w) in equations (3.4.11)-(3.4.13) is replaced with $(x + \Delta x, y + \Delta y, \lambda + \Delta\lambda, w + \Delta w)$, where $\Delta x = x^{k+1} - x^k$. Similarly for $\Delta y, \Delta\lambda$ and Δw.

(3.4.14) $$\begin{aligned} Q(x + \Delta x) - A'(\lambda + \Delta\lambda) - (w + \Delta x) &= c \\ A(x + \Delta x) - (y + \Delta y) &= b \\ (\Lambda + \Delta\Lambda)(Y + \Delta Y) e &= \mu e \\ (X + \Delta X)(W + \Delta W) e &= \mu e \end{aligned}$$

The system (3.4.14) can now be rewritten with all the terms containing Δ on the left hand side of the equal sign and the remaining terms on the right hand side of the equal sign to obtain the following non-linear system

(3.4.15) $$A'\Delta\lambda \quad \Delta w \quad Q\Delta x = c \quad A'\lambda \quad w \quad Qx$$

(3.4.16) $$A\Delta x \quad \Delta y = b \quad Ax \quad y$$

(3.4.17) $$Y\Delta\lambda \quad \Lambda\Delta y \quad \Delta\Lambda\Delta Ye = \mu e \quad \Lambda Ye$$

(3.4.18) $$W\Delta x \quad X\Delta w \quad \Delta X\Delta We = \mu e \quad XWe.$$

Denote by σ and ρ the right hand side of equation (3.4.15) and (3.4.16) respectively. In order to obtain a linear system of equations the non-linear terms are dropped, which gives the following linear system of $2n \quad 2m$ equations in $2n \quad 2m$ unknowns.

(3.4.19) $$A'\Delta\lambda \quad \Delta w \quad Q\Delta x = \sigma$$

(3.4.20) $$A\Delta x \quad \Delta y = \rho$$

(3.4.21) $$Y\Delta\lambda \quad \Lambda\Delta y = \mu e \quad \Lambda Ye$$

(3.4.22) $$W\Delta x \quad X\Delta w = \mu e \quad XWe$$

Provided that A has a full row rank, the above system is nonsingular and has a unique solution that defines the step directions for the path following method [Van2001]. Dropping the non-linear term from equations (3.4.17) and (3.4.18) is a common approach of solving non-linear system of equations.

From equations (3.4.21) and (3.4.22), we solve for Δy and Δw

(3.4.23) $$\Delta y = \Lambda^{-1}(\mu e \quad \Lambda Ye \quad Y\Delta\lambda)$$

(3.4.24) $$\Delta w = X^{-1}(\mu e \quad XWe \quad W\Delta x).$$

Substituting (3.4.23) and (3.4.24) into (3.4.19) and (3.4.20) to eliminate Δy and Δw gives the reduced system of equations

(3.4.25) $$\begin{aligned} A'\Delta\lambda \quad (X^{-1}W \quad Q)\Delta x &= \sigma \quad \mu X^{-1}e \quad w \\ A\Delta x \quad \Lambda^{-1}Y\Delta\lambda &= \rho \quad \mu\Lambda^{-1}e \quad y. \end{aligned}$$

Replacing σ and ρ with the terms on the right hand side of (3.4.15) and (3.4.16), (3.4.25) in matrix notation is written as

(3.4.26) $$\begin{bmatrix} (X^{-1}W \quad Q) & A' \\ A & \Lambda^{-1}Y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} c \quad A'\lambda \quad \mu X^{-1}e \quad Qx \\ b \quad Ax \quad \mu\Lambda^{-1}e \end{bmatrix}.$$

To progress to the next iteration the system of equations (3.4.26) must be solved for Δx and $\Delta\lambda$. The values obtained for Δx and $\Delta\lambda$ are substituted into (3.4.23) and (3.4.24) to calculate Δy and Δw. At this point we have obtained the necessary values for the step direction.

The next step is to compute the step length parameter, denoted by ϑ. The step direction was calculated under the assumption that $\vartheta = 1$, that is, $x^{k \quad 1} = x^k \quad \Delta x^k$. This however

may cause the new solution to violate the non-negativity constraints. In order to avoid this violation we want to find $\vartheta \in (0,1]$ such that

$$x_j + \vartheta \Delta x_j > 0 \quad \text{for } j = 1, \ldots, n.$$

Multiplying this by $1/x_j$

$$\frac{1}{\vartheta} + \frac{\Delta x_j}{x_j} > 0 \qquad \text{for } j = 1, \ldots, n,$$

$$\frac{1}{\vartheta} > -\frac{\Delta x_j}{x_j} \quad \text{for } j = 1, \ldots, n.$$

Similarly this inequality must also apply for $\boldsymbol{y}, \boldsymbol{\lambda}$ and $\boldsymbol{w}$. Therefore the upper limit of $1/\vartheta$ is given by

$$\max_{i\,j} \left\{ -\frac{\Delta x_j}{x_j}, -\frac{\Delta y_i}{y_i}, -\frac{\Delta \lambda_i}{\lambda_i}, -\frac{\Delta w_j}{w_j} \right\}.$$

This choice of ϑ will however not guarantee strict inequality and a step size factor r is introduced in order prevent the next point from touching the boundary as follows

(3.4.27) $$\vartheta = \min \left\{ r \left(\max_{i\,j} \left\{ -\frac{\Delta x_j}{x_j}, -\frac{\Delta y_i}{y_i}, -\frac{\Delta \lambda_i}{\lambda_i}, -\frac{\Delta w_j}{w_j} \right\} \right)^{-1}, 1 \right\}.$$

The factor r is selected to be less than but very close to one. Clearly the value of ϑ will be between zero and one.

We now look at the formulation of μ. As mentioned earlier a central path is a trajectory in the relative interior of the feasible region

$$F^0 = \{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w} \mid \boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}'\boldsymbol{\lambda} + \boldsymbol{w} = \boldsymbol{c}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} = \boldsymbol{b}; \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w} > 0)\}.$$

The algorithm begins in the interior of this feasible region and at each iteration estimates a value of μ representing a point on the central path that is hopefully closer to the optimal solution than the current point. The algorithm then attempts to step towards this point, making sure that the new point remains strictly in the interior of the region. The parameter μ serves as a measure of optimality for feasible points, as $\mu \to 0$, the central path converges to the optimal solution of (3.4.11)-(3.4.13) and therefore also to the solution of the original problem. However, we cannot just set μ to zero as that would destroy the convergence properties of the algorithm [JB2003]. Figure 3.3 illustrates an example of a central path. The different values of μ are illustrated as μ approaches zero.

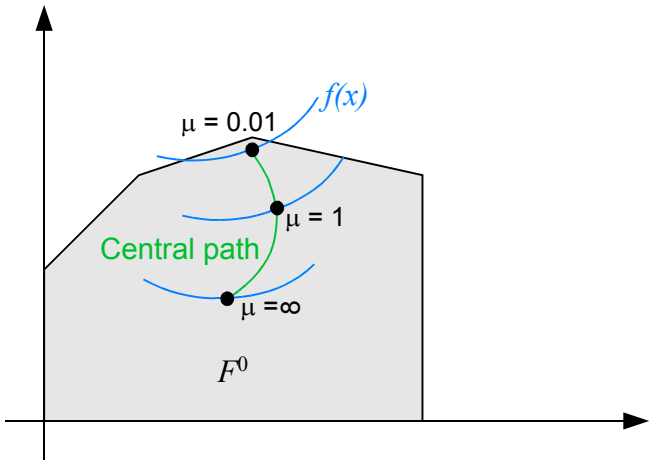

Figure 3.3 Central path of the solution for the path following algorithm.

To calculate the value of the barrier parameter μ the following must be considered. If the value of μ is too large, the sequence of solution could converge to the center of the feasible set, which is obviously not desired. If the value is too small, the sequence could stray too far from the central path and the algorithm could run into the boundary of the feasible set at a point that is not optimal. Therefore, the aim is to find a reasonable compromise between the two extremes. The following formulae is used to estimate the value of the barrier parameter. From equation (3.4.10) $\boldsymbol{w}=\mu \boldsymbol{X}^{-1}\boldsymbol{e}$. Now it can be shown that

$$\boldsymbol{w}=\mu \boldsymbol{X}^{-1}\boldsymbol{e}$$

$$\boldsymbol{w}=\mu \boldsymbol{e}'\boldsymbol{x}^{-1}\boldsymbol{e} \qquad \text{multiplying by } \boldsymbol{x}$$

$$\boldsymbol{x}'\boldsymbol{w}=\mu n \qquad \text{since } \boldsymbol{e} \text{ has } n \text{ elements.}$$

Similarly it follows that $\boldsymbol{\lambda}'\boldsymbol{y}=\mu m$. Thus the value of μ is taken as the average of these two values to give

(3.4.28) $$\mu=\frac{\boldsymbol{w}'\boldsymbol{x}}{n}\frac{\boldsymbol{\lambda}'\boldsymbol{y}}{m}.$$

This formula gives the value of μ whenever it is known that $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{w})$ lies on the central path. We want to use the formula (3.4.28) to produce an estimate for μ even when the current point is not on the central path. Since we want to step from the current point towards a point on the central path with a better approximation to the optimal solution, the value of μ is reduced by a certain fraction. Let δ denote this fraction value, where δ is a number between zero and one. Thus the value of the barrier parameter is given by

(3.4.29) $$\mu = \delta \frac{\boldsymbol{w}' \boldsymbol{x} \quad \boldsymbol{\lambda}' \boldsymbol{y}}{n \quad m}.$$

It has been shown that setting $\delta = \frac{1}{10}$ works well in practice [Van2001].

At this point all the necessary values have been calculated, that is, the next step direction, the step length parameter and μ. The last step is to replace the current solution with the new solution point as follows

(3.4.30) $$\begin{aligned} \boldsymbol{x}^{k \quad 1} &\leftarrow \boldsymbol{x}^{k} \quad \vartheta \Delta \boldsymbol{x}^{k}, \qquad \boldsymbol{w}^{k \quad 1} \leftarrow \boldsymbol{w}^{k} \quad \vartheta \Delta \boldsymbol{w}^{k}, \\ \boldsymbol{\lambda}^{k \quad 1} &\leftarrow \boldsymbol{\lambda}^{k} \quad \vartheta \Delta \boldsymbol{\lambda}^{k}, \qquad \boldsymbol{y}^{k \quad 1} \leftarrow \boldsymbol{y}^{k} \quad \vartheta \Delta \boldsymbol{y}^{k}. \end{aligned}$$

The process of replacing the current solution with the next continues until some stopping criteria is reached. Let $\varepsilon > 0$ be a small positive tolerance. There are a number of stopping rules that can be used:

1. The relative change in the value of the objective function from one iteration to the next
$$\left| f(\boldsymbol{x}^{k \quad 1}) \right| \quad \left| f(\boldsymbol{x}^{k}) \right| < \varepsilon.$$
2. The relative change in the vector $\boldsymbol{x}$ is such that
$$\boldsymbol{x}^{k \quad 1} \quad \boldsymbol{x}^{k} < \varepsilon.$$
3. When μ is small enough, that is, $(n \quad m)\mu \le \varepsilon$, that is, when the difference $\boldsymbol{w}' \boldsymbol{x} \quad \boldsymbol{\lambda}' \boldsymbol{y}$ is sufficiently small.

Before the algorithm begins the variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}$ and $\boldsymbol{w}$ have to be initialized. The algorithm begins at any point in the interior of the feasible region F^0. Usually it is convenient to set all variables to one, provided that the constraints are satisfied. However, if this is not the case, to obtain an initial solution that satisfies at least some of the equations can be done by finding a solution to the following system:

$$\begin{bmatrix} (\boldsymbol{Q} \quad \boldsymbol{I}) & \boldsymbol{A}' \\ \boldsymbol{A} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{b} \end{bmatrix}$$

and then setting the remaining variables to 1.

The steps of the interior point method can now be summarized in the follow table.

Step 1	Initialize the variables $\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{\lambda}^0, \boldsymbol{w}^0$.
Step 2	Calculate the value of the barrier parameter μ using equation (3.4.29).

Step 3	Calculate the step directions $(\Delta \boldsymbol{x}, \Delta \boldsymbol{y}, \Delta \boldsymbol{\lambda}, \Delta \boldsymbol{w})$ using (3.4.23), (3.4.24) and (3.4.26).
Step 4	Compute a step length parameter ϑ such that the new point has strictly positive components.
Step 5	Replace the old solution with the new solution such that $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \vartheta \Delta \boldsymbol{x}^k$, $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \vartheta \Delta \boldsymbol{w}^k$, $\boldsymbol{\lambda}^{k+1} \leftarrow \boldsymbol{\lambda}^k + \vartheta \Delta \boldsymbol{\lambda}^k$, $\boldsymbol{y}^{k+1} \leftarrow \boldsymbol{y}^k + \vartheta \Delta \boldsymbol{y}^k$. Test the stopping criteria. If the criteria is satisfied then the solution is optimal or it is found to be unbounded, otherwise go to Step 2 and continue this process until the optimal solution has been reached.

An example is now presented in order to illustrate the procedure.

Example

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = x_1 + 2x_2 + x_3 + \tfrac{1}{2}(x_1^2 + 2x_2^2 + 4x_3^2 + 4x_2x_3) \\ &\textit{Subject to} && x_1 + 2x_2 + 4x_3 \le 12 \\ & && 2x_1 + x_2 + 3x_3 \le 9 \\ & && x_1, x_2, x_3 \ge 0 \end{aligned}$$

As before

$$\boldsymbol{c} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 4 \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 1 & 3 \end{bmatrix} \text{ and } \boldsymbol{b} = \begin{bmatrix} 12 \\ 9 \end{bmatrix}.$$

Initializing the variables x, y, λ and w to

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ and } \boldsymbol{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Since $n = 3$, $m = 2$ and $\delta = \frac{1}{10}$, substituting into (3.4.29) gives

$$\mu = \frac{1}{10}.$$

Substituting into (3.4.26) gives the following set of equations

$$\boldsymbol{x} = \begin{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 2 & 5 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 4 & 3 \end{bmatrix} \\ \begin{bmatrix} 1 & 2 & 4 \\ 2 & 1 & 3 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta \lambda_1 \\ \Delta \lambda_2 \end{bmatrix} = \begin{bmatrix} 1.1 \\ 5.1 \\ 6.1 \\ 5.1 \\ 3.1 \end{bmatrix}.$$

Solving this system of equations yields the following set of values

$$\Delta x_1^0 = 1.024$$

$$\Delta x_2^0 = 1.589$$
$$\Delta x_3^0 = 0.795$$
$$\Delta \lambda_1^0 = \ 0.233$$
$$\Delta \lambda_2^0 = \ 1.457.$$

Calculating $\Delta \boldsymbol{y}$ and $\Delta \boldsymbol{w}$ by substituting into (3.4.23) and (3.4.24) yields

$$\Delta y_1^0 = \ 0.667$$
$$\Delta y_2^0 = 0.557$$
$$\Delta w_1^0 = 0.124$$
$$\Delta w_2^0 = \ 2.489$$
$$\Delta w_3^0 = \ 1.695.$$

Now calculating the value of ϑ by using the equation (3.4.27)

$$max\left\{\frac{1.024}{1};\ \frac{1.589}{1};\ \frac{0.795}{1};\frac{0.667}{1};\ \frac{0.557}{1};\frac{0.233}{1};\frac{1.457}{1};\ \frac{0.124}{1};\frac{2.489}{1};\frac{1.695}{1}\right\}.$$

Clearly the largest value is w_2 and selecting $r = 0.9 < 1$ therefore

$$\vartheta = 0.9\left(\frac{1}{2.489}\right) = 0.3616.$$

The next solution point is calculated using the equations (3.4.30) as follows

$$x_1^1 = x_1^0 \quad \vartheta \Delta x_1^0 = 1 \quad 0.3616(\ \ 1.024) = 0.630$$
$$x_2^1 = x_2^0 \quad \vartheta \Delta x_2^0 = 1 \quad 0.3616(1.589) = 1.575$$
$$x_3^1 = x_3^0 \quad \vartheta \Delta x_3^0 = 1 \quad 0.3616(0.795) = 1.287$$
$$\lambda_1^1 = \lambda_1^0 \quad \vartheta \Delta \lambda_1^0 = 1 \quad 0.3616(\ \ 0.233) = 0.916$$
$$\lambda_2^1 = \lambda_2^0 \quad \vartheta \Delta \lambda_2^0 = 1 \quad 0.3616(\ \ 1.475) = 0.473.$$

The values for $\boldsymbol{w}$ and $\boldsymbol{y}$ are computed similarly. At this point the value of the objective function is computed for the points $\boldsymbol{x}^0$ and $\boldsymbol{x}^1$ and tested for optimality.

$$f(\boldsymbol{x}^0) = \ \ 2.5$$
$$f(\boldsymbol{x}^1) = \ \ 3.13$$

Let $\varepsilon = 0.0001$. At this point the solution is not optimal since $\left|f(\boldsymbol{x}^1)\right| \ \ \left|f(\boldsymbol{x}^0)\right| = 0.63$ is not less then or equal to ε , therefore the procedure continues to find the next solution point. The following table shows the solution of $\boldsymbol{x}$ at each iteration.

Iteration	x_1	x_2	x_3
1	0.630	1.575	1.287
2	0.862	2.099	1.290
3	0.912	2.328	1.367
4	0.986	2.438	1.452
5	0.989	2.477	1.482
6	0.996	2.492	1.494
7	0.998	2.497	1.498
8	0.9998	2.4993	1.4995
9	0.9999	2.4997	1.4998

After nine iterations the approximate solution is obtained. Rounding the approximate solution gives the results $x_1 = 1, x_2 = 2\frac{1}{2}$ and $x_3 = 1\frac{1}{2}$. Thus the value of the objective function is $f(\boldsymbol{x}) = \ 3\frac{3}{4}$, which is the optimal solution as seen before.

Like the simplex method, many variations of the interior point method have been and still are being presented. The basic concept remains the same, however each variation attempts to improve the method in terms of efficiency, accuracy an the speed to find the optimal solution.

3.5 Summary

This chapter gave an overview of the algorithms available for solving convex quadratic programming problems. The quadratic problems have been classified into active set methods, ellipsoid methods and interior point methods. One of the most widely used active set methods, the Wolfe's algorithm was described in detail. This method is based on the simplex method for solving linear programming problems, which aids in the understanding of the method provided the reader is familiar with linear programming. There are many other methods based on the simplex method some of which have been introduced in this chapter.

Other active set methods that are not variations of the simplex method have also been described. These were, Beale's algorithm and Theil-Van de Panne procedure. Beale's algorithm is similar to Wolfe's method as it also uses simplex transformation, however, it does not begin with the Kuhn-Tucker conditions. The procedure of Theil-Van de Panne solves the problem by first solving the objective function without regarding any of the

constraints and then progressively incorporates the violated constraints into the next solution point.

The next type of method discussed in this chapter are the Ellipsoid methods. These methods however are not widely used and do not work very well in practice, therefore these methods are not discussed in detail in this dissertation and are left for future research.

Another type of method that has received much attention are the interior point methods first proposed by N. K. Karmarkar [Mil2000]. Many papers are available explaining the interior point method. These methods search for the optimum on the interior of the feasible region rather than the exterior as do the simplex methods. Virtually all approaches using the interior method are the so called path following methods and therefore the path following method was discussed in detail.

In the next chapter algorithms for solving quadratic problems with a non-convex objective function will be discussed.

Chapter 4

General Quadratic Minimization Algorithms

4.1 Introduction

General quadratic programming problems with no convexity assumptions, that is, problems where $\boldsymbol{Q}$ cannot be classified as either definite of semi-definite. This chapter introduces a selection of algorithms for finding the local minimizing points for general quadratic minimization problems. Figure 4.1 illustrates a problem with a non-convex objective function and represents the function $z = 2x \quad 2y \quad 2x^2 \quad 2y^2$.

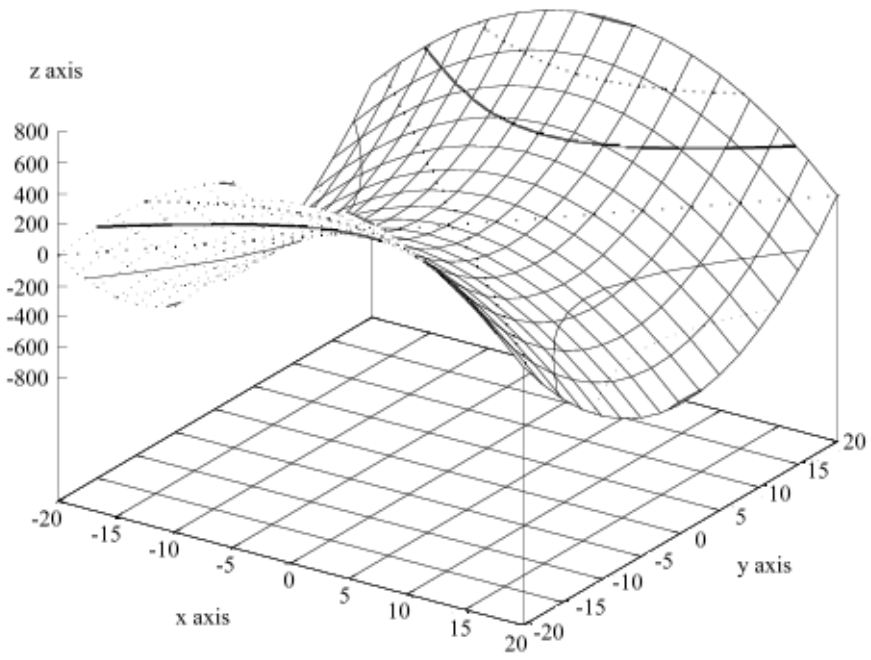

Figure 4.1 Contours of a non-convex objective function.

As discussed in the previous chapter, if it is not known if a function is convex or concave, the global minimum may never be found. In fact, it is possible that an algorithm finds a stationary point that is not even a local minimum. However, in many cases a local optimum is an improvement on the currently available solution or no solution at all.

The algorithms that will be discussed in the chapter enable one to find the local optimum points provided certain conditions are satisfied or to conclude that the problem has no finite optimal solution. An example is also provided to aid in the understanding of these algorithms.

4.2 Fixed Charge Problem

A fixed charge problems arises frequently in optimization. It is a problem in which there is a cost associated with performing an activity at a non-zero level. This cost however, does not depend on the level of the activity. For example, a manufacturer needs to rent a machine to produce tables. Regardless of how many tables are manufactured (1 or 1000) the cost or the fixed charge of the rent must be paid.

It has been shown by G. de Kock [DK1975] that a fixed charge problem can be formulated as a non-convex quadratic programming problem. This will allow these types of problems to be solved by using some of the algorithms discussed in the dissertation. Consider the following integer programming problem

$$\text{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \alpha' h(\boldsymbol{x})$$
$$x \quad D$$
$$\text{where} \quad D = \{x : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, 0 \le \boldsymbol{x} \le \boldsymbol{d}\}$$

where $\boldsymbol{A}$ is a m x n matrix, $\boldsymbol{c}, \boldsymbol{x}$ and $\boldsymbol{d}$ E^n, $\boldsymbol{b}$ E^m and $h : E^n \to \{0,1\}^n$ is a vector function such that

$$h(\boldsymbol{x}) = (h(x_j)) \quad \text{where} \quad h(x_j) \begin{cases} 0 & \text{if } x_j = 0 \\ 1 & \text{if } x_j > 0. \end{cases}$$

where E^n is a n dimensional Euclidean space. If $\alpha \ge 0$ then according to Hirsch and Dantzig [HD1968] the above problem is equivalent to

$$\begin{aligned} &\text{Minimize} && f(\boldsymbol{x}, \boldsymbol{h}) = \boldsymbol{c}'\boldsymbol{x} \quad \alpha'\boldsymbol{h} \\ &\text{Subject to} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{d}, \\ &&& 0 \le x_j \le h_j d_j \le d_j \quad \text{and} \\ &&& \sum_{j=1}^{n} x_j (1 \quad h_j) = 0. \end{aligned} \tag{4.2.1}$$

The problem (4.2.1) is linear, except for one of the quadratic functions in the constraints. By taking this quadratic function in to the objective function, we obtain the following

(4.2.2)
$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x},\boldsymbol{h}) = \boldsymbol{c}'\boldsymbol{x} \quad \alpha'\boldsymbol{h} \quad \delta\sum_{j=1}^{n} x_j(1 \quad h_j) \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{d} \\ &\qquad\qquad 0 \le x_j \le h_j d_j \le d_j \quad \text{with} \quad 0 \le h_j \le 1. \end{aligned}$$

According to De Kock (4.2.2) is equivalent to (4.2.1) if $\delta > 0$ is chosen sufficiently large. De Kock further proves that the objective functions of (4.2.2) is neither convex nor concave by the following theorem.

Theorem 4.1

The objective function of (4.2.2)*, namely*

$$f(\boldsymbol{x},\boldsymbol{h}) = \boldsymbol{c}'\boldsymbol{x} \quad \alpha'\boldsymbol{h} \quad \delta\sum_{j=1}^{n} x_j(1 \quad h_j)$$

is neither convex nor concave.

Proof:

The quadratic function $f(\boldsymbol{x},\boldsymbol{h})$ *can be written as follows*

$$g(\boldsymbol{y}) = \boldsymbol{a}'\boldsymbol{y} \quad \boldsymbol{y}'\boldsymbol{Q}\boldsymbol{y}$$

where $\boldsymbol{y} = (\boldsymbol{x},\boldsymbol{h})', \boldsymbol{a}' = (\boldsymbol{c}' \quad \delta\boldsymbol{e}', \alpha')$ *with* $\boldsymbol{e}' = (1,1,\ldots,1)$ *and*

$$\boldsymbol{Q} = \quad \delta\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I}_n \\ \boldsymbol{I}_n & \boldsymbol{0} \end{bmatrix}.$$

$g(\boldsymbol{y})$ *is convex if and only if the eigenvalues of* $\boldsymbol{Q}$ *are positive, that is,* $\boldsymbol{Q}$ *is positive semi-definite. Conversely* $\boldsymbol{Q}$ *is concave if it is negative semi-definite. Therefore it is only necessary to prove that* $\boldsymbol{Q}$ *has both positive and negative eigenvalues. Consider now the quadratic function of* $\boldsymbol{Q}$*, namely*

$$z_n(\lambda) = |\lambda\boldsymbol{I}_{2n} \quad \boldsymbol{Q}| = \begin{vmatrix} \lambda\boldsymbol{I}_n & \delta\boldsymbol{I}_n \\ \delta\boldsymbol{I}_n & \lambda\boldsymbol{I}_n \end{vmatrix}.$$

If $n = 1$ *it follows that* $z_1(\lambda) = (\lambda^2 \quad \delta^2)$*. Suppose that*

$$z_n(\lambda) = (\lambda^2 \quad \delta^2)^n = (\lambda \quad \delta)^n(\lambda \quad \delta)^n$$

Let the above be true for $n = k$*. Then by moving the* $(n \quad 1)^{th}$ *row and column of the determinant representation of* $z_n(\lambda)$ *to the second row and column, it follows directly that*

$$z_n(\lambda) = \begin{vmatrix} \lambda & \delta \\ \delta & \lambda \end{vmatrix} f_{n \quad 1}(\lambda).$$

Therefore the statement holds for $n = k \quad 1$*,* $\boldsymbol{Q}$ *therefore has eigenvalues* δ *and* δ*, both with multiplicity* n*. This proves the theorem* [DK1975].

4.3 Beale's Algorithm

Beale's algorithm has been discussed in great detail in Section 3.2.2, for solving convex quadratic programming problems. This algorithm can be used for problems for which the convexity of the objective function is not known, like for example, the fixed charge problem discussed above. However, as already mentioned it may not produce a global minimum nor a local minimum.

Beale's algorithm expresses the basic variables and the objective function in terms of the non-basic variables. It then determines which of the non-basic variables will provide the largest decrease in the value of the objective function. Once the particular non-basic variable has been identified, the largest possible value is determined to which the non-basic variable can increase without violating the non-negativity restrictions. At this point the non-basic variable becomes basic and the solution is tested for optimality. If the solution is not optimal the process is repeated.

There are two additional rules that improve the chances for the algorithm to obtain local minimum points for non-convex problems [Bea1959]:

1. A free variable should be removed from the non-basic set if z in (3.2.19) contains any off-diagonal terms in this variable, even if the linear terms are zero.
2. A free variable should be removed from the non-basic set if z in (3.2.19) contains no off-diagonal terms in this variable, but the diagonal terms are non-positive. In this case the variable may be either increased or decreased until some basic variable (not a free variable) becomes equal to zero.

Provided these two rules are applied, the algorithm will produce a local minimum point unless there is a restricted variable in the linear term in the final expression z which happens to vanish. In this case, that point can be made a local minimum by an arbitrarily small change in the coefficients of the objective function.

Apart from these two additional rules, Beale's algorithm proceeds in the same manner as in Section 3.2.2.

Example

Consider the following quadratic maximization problem, which was taken from lecture notes on quadratic programming written by N. Gould [Gou2000]. Note that this quadratic problem is a non-convex problem.

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = x_1 \;\; 2x_2 \;\; 2x_1^2 \;\; 2x_2^2 \\ &\textit{Subject to} \quad x_1 \;\; x_2 \le 1 \\ &\qquad\qquad 3x_1 \;\; x_2 \le 1.5 \\ &\qquad\qquad x_1, x_2 \ge 0 \end{aligned}$$

The two constraints need to be rewritten as equalities by adding slack variables x_3 and x_4 such that

$$\begin{aligned} &x_1 \;\; x_2 \;\; x_3 = 1 \quad \text{and} \\ &3x_1 \;\; x_2 \;\; x_4 = 1.5. \end{aligned}$$

An initial feasible solution is immediately obvious as $x_B = (x_3, x_4) = (1, 1.5)$. The initial tableau is now as follows:

	x_1	x_2	x_3	x_4
$x_{B_1} = x_3 = 1$	1	1	1	0
$x_{B_2} = x_4 = 1.5$	3	1	0	1

The problem is now partitioned according to the variables that belong to the basis and variables which are not in the basis.

$$\boldsymbol{x} = (\boldsymbol{x_B}, \boldsymbol{x_R}) = (x_3, x_4 | x_1, x_2) = (1, 1.5 | 0, 0)$$

$$\boldsymbol{c}' = [\boldsymbol{c}'_{\boldsymbol{B}} | \boldsymbol{c}'_{\boldsymbol{R}}] = [0 \;\; 0 | 1 \;\; 2]$$

$$\boldsymbol{A} = [\boldsymbol{B} | \boldsymbol{R}] = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 3 & 1 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{BB} & \boldsymbol{Q}_{BR} \\ \boldsymbol{Q}_{RB} & \boldsymbol{Q}_{RR} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

This can now be substituted into (3.2.20) to compute the values of z_0, $\boldsymbol{p}$ and $\boldsymbol{D}$

$$z_0 = 0 \;\; , \;\; \boldsymbol{p} = [1 \;\; 2] \text{ and } \boldsymbol{D} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

The matrix $\boldsymbol{D}$ is already symmetric so it does not need to be adjusted. Substituting the above results into (3.2.21)

$$\begin{aligned} z &= z_0 \;\; \boldsymbol{p}\boldsymbol{x_R} \;\; \boldsymbol{x}'_{\boldsymbol{R}}\boldsymbol{D}\boldsymbol{x_R} \\ &= x_1 \;\; 2x_2 \;\; 2x_1^2 \;\; 2x_2^2. \end{aligned}$$

To determine the entering variable the values of p_j are computed.

$$\frac{\partial z}{\partial x_{R_1}} = \frac{\partial z}{\partial x_1} = p_1 = 1 \quad 4x_1 = 1 \ , \ \frac{\partial z}{\partial x_{R_2}} = \frac{\partial z}{\partial x_2} = p_2 = \quad 2 \quad 4x_2 = \quad 2 \ .$$

Select $min(p_j)$ where p_j is negative, to enter the basis. In this case x_2 will enter the basis. Finding the critical points

$$x_{R_2}^{(1)} = min\left\{\frac{x_{Bi}}{y_{i2}}, y_{i2} > 0\right\} = \left\{\frac{1}{1}, \frac{3}{2}\right\} = 1 \ .$$

However, the next critical value is

$$x_{R_2}^{(2)} = \frac{p_2}{2d_{22}} = \frac{2}{2(2)} = \frac{1}{2},$$

therefore x_2 will be increased to only $\frac{1}{2}$. Since $x_{R_2}^{(2)} < x_{R_2}^{(1)}$ a new free variable u_1 is introduced to the problem such that

$$u_1 = \frac{\partial z}{\partial x_{R_2}} = \quad 2 \quad 4x_2 \text{ which can be rewritten as } \ u_1 \quad 4x_2 = \quad 2 \ .$$

Before x_2 is increased u_1 must be added to the basis, where $u_1 = \quad 2$ as $x_2 = 0$ and $x_3 = 0$. The current tableau is now as follows:

	x_1	x_2	x_3	x_4	u_1
$x_{B_1} = x_3 = 1$	1	1	1	0	0
$x_{B_2} = x_4 = 1.5$	3	1	0	1	0
$x_{B_3} = u_1 = \quad 2$	0	-4	0	0	1

Bringing x_2 into the basis, the variable that leaves the basis needs to be calculated. The free variable u_1 must leave the basis and is replaced by x_2. The new tableau becomes

	x_1	x_2	x_3	x_4	u_1
$x_{B_1} = x_3 = 0.5$	1	0	1	0	1/4
$x_{B_2} = x_4 = 1$	3	0	0	1	1/4
$x_{B_3} = x_2 = 0.5$	0	1	0	0	-1/4

The new basic solution has been generated. The problem is again repartitioned.

$$\boldsymbol{x} = (\boldsymbol{x_B}, \boldsymbol{x_R}) = (x_3, x_4, x_2 | x_1, u_1)$$

$$\boldsymbol{c}' = [\boldsymbol{c}'_{\boldsymbol{B}} | \boldsymbol{c}'_{\boldsymbol{R}}] = [0 \ 0 \quad 2 | 1 \ 0]$$

$$\boldsymbol{A} = [\boldsymbol{B} | \boldsymbol{R}] = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 3 & 0 \\ 0 & 0 & 4 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Q}=\begin{bmatrix}\boldsymbol{Q}_{BB} & \boldsymbol{Q}_{BR}\\ \boldsymbol{Q}_{RB} & \boldsymbol{Q}_{RR}\end{bmatrix}=\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 2 & 0 & 0\\ 0 & 0 & 0 & 2 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

which gives the values

$$z_0=\frac{1}{2},\quad \boldsymbol{p}=[1\ \ 0] \text{ and } \boldsymbol{D}=\begin{bmatrix}2 & 0\\ 0 & \frac{1}{8}\end{bmatrix}.$$

Substituting the above results into (3.2.21)

$$\begin{aligned} z &= z_0 \quad \boldsymbol{p}\boldsymbol{x}_{\boldsymbol{R}} \quad \boldsymbol{x}'_{\boldsymbol{R}}\boldsymbol{D}\boldsymbol{x}_{\boldsymbol{R}} \\ &= \frac{1}{2} \quad x_1 \quad u_1 \quad 2x_1^2 \quad \frac{1}{8}u_1^2 . \end{aligned}$$

To determine the entering variable the values of p_j are computed.

$$\frac{\partial z}{\partial x_{R_1}}=\frac{\partial z}{\partial x_1}=p_1=1 \quad 4x_1=1,\ \frac{\partial z}{\partial x_{R_2}}=\frac{\partial z}{\partial u_1}=p_2= \quad 1 \quad \frac{1}{4}u_1=0.$$

Since the partial derivative with respect to the actual variable is positive and with respect to the free variable is zero, the local minimum has been found at $x_1=0$ and $x_2=\frac{1}{2}$. Thus the final value if the objective function is $f(\boldsymbol{x})=\frac{1}{2}$. This solution may however, not be the global minimum.

4.4 Keller's Algorithm

This algorithm was developed by E. L. Keller in 1972 [Kel1973] and it is a generalization of the algorithm developed by Dantzig. It allows for finding local minimum points for general quadratic minimization problems and allows for finding the optimal solution for convex quadratic programming problems [CC1978].

Consider the quadratic minimization problem

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x})=\boldsymbol{c}'\boldsymbol{x} \quad \frac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x}\le\boldsymbol{b} \\ &\qquad\qquad\ \ \boldsymbol{x}\ge 0. \end{aligned} \tag{4.4.1}$$

The Kuhn-Tucker conditions are necessary conditions for the solution of problem (4.4.1). Applying the Kuhn-Tucker conditions to (4.4.1) as in Section 2.6.3, the problem is reformulated as follows

(4.4.2)
$$\begin{gathered} Qx \quad A'\lambda \quad c=\mu \\ Ax \quad b=y \\ x\geq 0,\lambda\geq 0,\mu\geq 0,y\geq 0 \\ \text{and } x'\mu=0 \text{ and } \lambda' y=0. \end{gathered}$$

Now let

$$\begin{bmatrix} A' & Q \\ 0 & A \end{bmatrix} = M = \begin{bmatrix} M'_{IJ} & M_{JJ} \\ M_{II} & M_{IJ} \end{bmatrix} \text{ and } \begin{bmatrix} c \\ b \end{bmatrix} = q = \begin{bmatrix} q_J \\ q_I \end{bmatrix},$$

where $I=\{1,\ldots,m\}$, $J=\{m \quad 1,\ldots,m \quad n\}$. Also let

$$u_I=y, u_J=x, v_I=\lambda \text{ and } v_J=\mu,$$

then (4.4.1) becomes

(4.4.3)
$$\begin{aligned} &\textit{Minimize } f(u_J)=q_J'u_J \quad \frac{1}{2}u_J'M_{JJ}u_J \\ &\textit{Subject to } M_{IJ}u_J\leq q_I \\ &\qquad\qquad u_J\geq 0. \end{aligned}$$

The system (4.4.2) now becomes

(4.4.4)
$$\begin{gathered} M_{JJ}u_J \quad M'_{IJ}v_I \quad q_J=v_J \\ M_{IJ}u_J \quad q_I=u_I \\ u\geq 0, v\geq 0 \\ \text{and } u'v=0 \end{gathered}$$

where $u'=[u'_I,u'_J]$ and $v'=[v'_I,v'_J]$. If (u,v) satisfies the first two equations of (4.4.4) then

$$f(u_J)=\frac{1}{2}(\quad q'_I v_I \quad q'_J u_J \quad u'v).$$

Let

$$F(u,v)=\quad q'_I v_I \quad q'_J u_J \quad u'v,$$

then the system (4.4.4) can be represented by the schema

v'_I	u'_J	1	
M'_{IJ}	M_{JJ}	q_J	$=v_J$
0	M_{IJ}	q_I	$=u_I$
q'_I	q'_J	0	$=\vartheta$

where $\vartheta=F(u,v) \quad u'v$. If $q_I\geq 0$ and $q_J\geq 0$, it is clear that $u_I=q_I, u_J=0, v_I=0$ and $v_J=q_J$ is the solution to (4.4.4). If however, there exists a component of q that is negative, a new schema can be constructed from which it is possible to obtain a solution or to conclude that the problem is unbounded. The new schema is constructed using pivoting, that is, permuting rows and then applying the same permutation to all the other rows. There

are three types of pivots carried out in the schema, which will now be represented in tabular form. Consider a more general schema for the pivots.

(4.4.5)

$\boldsymbol{v}'_I$	$\boldsymbol{u}'_J$	1	
$\boldsymbol{M}'_{IJ}$	$\boldsymbol{M}_{JJ}$	$\boldsymbol{q}_J$	$=\boldsymbol{v}_J$
$\boldsymbol{M}_{II}$	$\boldsymbol{M}_{IJ}$	$\boldsymbol{q}_I$	$=\boldsymbol{u}_I$
$\boldsymbol{q}'_I$	$\boldsymbol{q}'_J$	0	$=\vartheta$

E. Keller shows that in this schema $\boldsymbol{M}_{II}$ and $\boldsymbol{M}_{JJ}$ are symmetrical [Kel1973]. In the initial schema $\boldsymbol{M}_{II}$ was the zero matrix. The variables on the right hand side of schema (4.4.5) are called basic variables and the variables along the top of the schema are called non-basic variables. The variables also require to hold the property that for each i, one member of the pair u_i, v_i is basic and one non-basic. Thus the transformations are as follows:

i) A 1x1 pivot on t by which non-basic u_t and basic v_t are interchanged.

(4.4.6)

v_i	u_t	u_j	
m_{it}	m_{tt}	m_{tj}	q_t $=v_t$
m_{ij}	m_{tj}	m_{jj}	q_j $=v_j$
m_{ii}	m_{it}	m_{ij}	q_i $=u_i$
q_i	q_t	q_j	0 $=\vartheta$

$\downarrow$ pivot on t

v_i	v_t	u_j		
m_{ij} $m_{it}m_{tj}/m_{tt}$	m_{tj}/m_{tt}	m_{jj} m_{tj}^2/m_{tt}	q_j $q_t m_{tj}/m_{tt}$	$=v_j$
m_{ii} m_{it}^2/m_{tt}	m_{it}/m_{tt}	m_{ij} $m_{it}m_{tj}/m_{tt}$	q_i $q_t m_{it}/m_{tt}$	$=u_i$
m_{it}/m_{tt}	$1/m_{tt}$	m_{tj}/m_{tt}	q_t/m_{tt}	$=u_t$
q_i $m_{it}q_t/m_{tt}$	q_t/m_{tt}	q_j $m_{tj}q_t/m_{tt}$	q_t^2/m_{tt}	$=\vartheta$

ii) A 1x1 pivot on s by which non-basic v_s and basic u_s are interchanged.

v_i	v_s	u_t	
m_{it}	m_{st}	m_{tt}	q_t $=v_t$
m_{ii}	m_{is}	m_{it}	q_i $=u_i$
m_{is}	m_{ss}	m_{st}	q_s $=u_s$
q_i	q_s	q_t	0 $=\vartheta$

(4.4.7) ↓ pivot on s

v_i	u_s	u_t		
m_{is}/m_{ss}	$1/m_{ss}$	m_{st}/m_{ss}	q_s/m_{ss}	$=v_s$
$m_{it} \quad m_{is}m_{st}/m_{ss}$	m_{st}/m_{ss}	$m_{tt} \quad m_{st}^2/m_{ss}$	$q_t \quad q_s m_{st}/m_{ss}$	$=v_t$
$m_{ii} \quad m_{is}^2/m_{ss}$	m_{is}/m_{ss}	$m_{it} \quad m_{is}m_{st}/m_{ss}$	$q_i \quad q_s m_{is}/m_{ss}$	$=u_i$
$q_i \quad m_{is}q_s/m_{ss}$	q_s/m_{ss}	$q_t \quad m_{st}q_s/m_{ss}$	q_s^2/m_{ss}	$=\vartheta$

iii) A 2x2 pivot on s,t by which basic u_s and non-basic u_t are interchanged and basic v_t and non-basic v_s are interchanged.

v_s	v_i	u_t	u_j		
m_{st}	m_{it}	m_{tt}	m_{jt}	q_t	$=v_t$
m_{sj}	m_{ij}	m_{jt}	m_{jj}	q_j	$=v_j$
0	0	m_{st}	m_{sj}	q_s	$=u_s$
0	m_{ii}	m_{it}	m_{ij}	q_i	$=u_i$
q_s	q_i	q_t	q_j	0	$=\vartheta$

(4.4.8) ↓ pivot on s,t

v_t	v_i	u_s	u_j		
$\frac{1}{m_{st}}$	$\frac{m_{it}}{m_{st}}$	$\frac{m_{tt}}{m_{st}^2}$	$\frac{(m_{st}m_{jt} \quad m_{sj}m_{tt})}{m_{st}^2}$	$\frac{(q_t m_{st} \quad q_s m_{tt})}{m_{st}^2}$	$=v_s$
$\frac{m_{sj}}{m_{st}}$	$\frac{m_{ij} \quad m_{it}m_{sj}}{m_{st}}$	$\frac{(m_{st}m_{jt} \quad m_{sj}m_{tt})}{m_{st}^2}$	ζ	ξ	$=v_j$
0	0	$\frac{1}{m_{st}}$	$\frac{m_{sj}}{m_{st}}$	$\frac{q_s}{m_{st}}$	$=u_t$
0	m_{ii}	$\frac{m_{it}}{m_{st}}$	$\frac{m_{ij} \quad m_{it}m_{sj}}{m_{st}}$	$\frac{q_i \quad q_s m_{it}}{m_{st}}$	$=u_i$
$\frac{q_s}{m_{st}}$	$\frac{q_i \quad q_s m_{it}}{m_{st}}$	$\frac{(q_t m_{st} \quad q_s m_{tt})}{m_{st}^2}$	ξ	$\frac{(2q_s q_t m_{st} \quad m_{tt}q_s^2)}{m_{st}^2}$	$=\vartheta$

where $\xi = q_j \ \frac{(q_s m_{st}m_{jt} \quad q_t m_{sj}m_{st} \quad q_s m_{tt}m_{sj})}{m_{st}^2}$ and $\zeta = m_{jj} \ \frac{(2m_{st}m_{sj}m_{jt} \quad m_{sj}^2 m_{tt})}{m_{st}^2}$.

The following theorem states that, given a schema arising from a general quadratic programing problem, a sequence of pivots can be generated which terminates in a schema

from which it can be said that either the local optimum has been found, or for a convex problem the global optimum has been found or that the objective function is unbounded.

Theorem 4.2 [Kel1973]

Let (4.4.5) *be the schema, where* $\boldsymbol{M}_{I\,I}$ *and* $\boldsymbol{M}_{J\,J}$ *are symmetric. Then, by a finite sequence of transformations and pivots of types (i), (ii) and (iii), it is possible to obtain a schema in one of the following forms (where symbolizes a non-negative entry, + a positive entry, a non-positive entry, - a negative entry and ± an entry of arbitrary sign):*

Case (a)

$\boldsymbol{v}'_I$	$\boldsymbol{u}'_J$		
			$=\boldsymbol{v}_J$
			$=\boldsymbol{u}_I$

This schema shows the solution to the Kuhn-Tucker conditions.

Case (b)

$\boldsymbol{v}'_I$	.	u_t	.		
		.			
0 0 - -	.		.	-	$=v_t$
		.			.
		0 0 + +		 ± ±	$=\boldsymbol{u}_I$
± ±		-			

This schema shows an unbound solution. In this case there is a $t \quad J$ *such that* $q_t<0$, $m_{tt}\le 0$. *Also,* $m_{it}\ge 0$ *for all* $i \quad I$. *If* $m_{it}=0 (i \quad I)$, *then* $q_i\ge 0$; *if* $m_{it}>0 (i \quad I)$, *then* q_i *can have an arbitrary sign.*

Case (c)

.	v_s	.	$\boldsymbol{u}_J$		
					$=\boldsymbol{v}_J$
					.
	0				
0	0	0		-	$=u_s$
	0				
	+				

This schema shows infeasibility. In this case there is an $s \quad I$ *such that* $q_s<0$, $m_{sj}\le 0$ *for all* $j \quad J$ *and* $m_{si}=0$ *for all* $i \quad I$.

In all the cases $\boldsymbol{M}_{I\,I}$ *for the last schema, is positive semi definite.*

Considering the problem as given by (4.4.3) and the Kuhn-Tucker conditions given by (4.4.4), according to Theorem 4.2, a finite sequence of transformations will lead to a schema in one of the three terminal cases (a), (b) or (c).

In case (a), $\boldsymbol{q}$ is non-negative. If we take $\boldsymbol{u}_I = \boldsymbol{q}_I, \boldsymbol{u}_J = 0, \boldsymbol{v}_I = 0$ and $\boldsymbol{v}_J = \boldsymbol{q}_J$, then conditions (4.4.4) are satisfied and $\boldsymbol{u}_J$ is a solution point for the problem. If f is convex, then $\boldsymbol{u}_J$ is also a global minimum point. However, for a general quadratic minimization problem this need not be the case, but we can conclude that $\boldsymbol{u}_J$ is a local minimum point. By Theorem 4.2 M_{II} is positive semi-definite and thus $\boldsymbol{v}'_I M_{II} \boldsymbol{v}_I \geq 0$ for all $\boldsymbol{v}_I$. Also if $j \quad J$ any small increase in u_j will increase the value of F if $q_j > 0$. Thus $\boldsymbol{q}_J > 0$ is a sufficient condition for a point to be a local minimum point.

In case (b), $u_i = q_i \quad m_{it} u_t$ for $i \quad I$; $v_j = q_j \quad m_{jt} u_t$ for $j \quad J$ and $F(\boldsymbol{u}, \boldsymbol{v}) = u_t v_t \quad q_t u_t$. Since $v_t = q_t \quad m_{tt} u_t$, the value of F can be expressed in terms of u_t as $F(\boldsymbol{u}, \boldsymbol{v}) = 2 q_t u_t \quad m_{tt} u_t^2$. Since $q_t < 0$ and $m_{tt} \leq 0, F(\boldsymbol{u}, \boldsymbol{v}) \to \infty$ as $u_t \to \infty$. If $m_{it} > 0, i \quad I$ then $u_i \to \infty$ as $u_t \to \infty$. If $m_{it} = 0, \ i \quad I$ then $u_i = q_i$ is negative and will remain constant as u_t increases. It can then be concluded that F is unbound from below over the feasibility region.

In case (c), it is impossible to find a non-negative solution to the s th row equation $u_s = q_s \quad \sum_{j \ J} m_{sj} u_j$ and it can be concluded that the feasible region is empty. The proof of the theorem will not be provided in the dissertation and can be found in [Kel1973].

Using the transformations (4.4.6)-(4.4.8) the steps of the algorithm can now be summarized in the following table. It is assumed, that at this point the initial schema has already been constructed.

Step 1	If $\boldsymbol{q}_J \geq 0$ the procedure stops as $\boldsymbol{u}_I = \boldsymbol{q}_I, \boldsymbol{u}_J = 0, \boldsymbol{v}_I = 0$ and $\boldsymbol{v}_J = \boldsymbol{q}_J$ is a solution to (4.4.4) (Case (a)). Otherwise, go to step 2.
Step 2	Choose $t \quad J$ such that $q_t < 0$. Go to step 3.
Step 3	If $m_{tt} \leq 0$, go to step 4. If $m_{tt} > 0$, find s such that $q_s / m_{st} = max\{q_t / m_{it}, m_{it} < 0, i \quad I\}$.
(a)	If $q_t / m_{tt} \geq q_s / m_{st}$ or if $\{m_{it} < 0, i \quad I\} =$ make a 1x1 pivot on t and return to step 1.
(b)	If $q_t / m_{tt} < q_s / m_{st}$ and if $m_{ss} \neq 0$, make a 1x1 pivot on s and return to step 3.

(c)	If $q_t/m_{tt} < q_s/m_{st}$ and if $m_{ss}=0$, make a 2x2 pivot on s,t and return to step 1.
Step 4	If $m_{tt} \le 0$, find s such that $q_s/m_{st} = max\{q_t/m_{it}, m_{it}<0, i \quad I\}.$
(a)	If $\{m_{it}<0, i \quad I\}=$ the procedure stops as the terminal case occurs (Case (b)).
(b)	If $m_{ss} \neq 0$, make a 1x1 pivot on s and return to step 3.
(c)	If $m_{ss}=0$, make a 2x2 pivot on s,t and return to step 1.

Example

Consider the following example as in the previous section.

$$\textit{Minimize} \quad f(\boldsymbol{x}) = x_1 \quad 2x_2 \quad \frac{1}{2}(\quad 4x_1^2 \quad 4x_2^2)$$

$$\textit{Subject to} \quad x_1 \quad x_2 \le 1$$
$$3x_1 \quad x_2 \le 1.5$$
$$x_1, x_2 \ge 0$$

Thus from this it can be said that

$$\boldsymbol{Q} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}, \boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 3 & 1 \end{bmatrix} \boldsymbol{b} = \begin{bmatrix} 1 \\ 1.5 \end{bmatrix} \text{ and } \boldsymbol{c} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

The first schema of Keller's algorithms is thus as follows:

v_i	v_i	u_j	u_t		
v_3	v_4	u_1	u_2		
1	3	-4	0	1	$= v_1 \quad v_j$
1	1	0	4	-2	$= v_2 \quad v_t$
0	0	-1	-1	1	$= u_3 \quad u_i$
0	0	-3	-1	3/2	$= u_4 \quad u_i$
-1	-3/2	1	-2	0	$= \vartheta$

It is clear that u_2 will be the variable exchanged by v_2 as it is the only one for which $q_t < 0$. Also $m_{tt} > 0$ and $q_t/m_{tt} \ge q_s/m_{st}$ therefore a 1x1 pivot on t is executed. The next schema thus gives the following results.

v_3	v_4	u_1	v_2		
1	3	-4	0	1	$= v_1$
1/4	1/4	0	1/4	1/2	$= u_2$
1/4	1/4	-1	-1/4	3/2	$= u_3$
1/4	1/4	-3	-1/4	2	$= u_4$
-3/2	-2	1	-1/2	-1	$= \vartheta$

Going back to Step 1 there are no values for which $q_J \leq 0$. Therefore a terminal case has been reached and the procedure stops. Thus the result is $u_1 = 0$ and $u_2 = \frac{1}{2}$ and the value of the objective function is $f(x) = \frac{1}{2}$.

4.5 Interior Point Method

The interior point method has been discussed in detail in Section 3.4 of the previous chapter, for solving convex quadratic programming problems. The interior point method is one of the most commonly used algorithms for solving quadratic programming problems and has been extended for solving non-convex quadratic programming problems [VS1991].

Denote the matrix ($X^{-1}W \quad Q$) in (3.4.26) by $H = (X^{-1}W \quad Q)$. Let

$$N = H \quad A' \Lambda^{-1} Y A.$$

R. J. Vanderbei and D. F. Shanno [VS1991] showed that N is positive definite for a convex quadratic programming problem. However, for non-convex problems the matrix N may fail to be positive semi-definite. If N is indefinite, the algorithm may converge, but it may not converge to even a local minimum.

In order to solve this problem the diagonal elements of H are perturbed as follows:

$$\hat{H} = H \quad \eta I, \quad \eta \geq 0,$$

with the value of η chosen such that N is positive definite. To compute the value of η, the system

$$\begin{bmatrix} H & A' \\ A & \Lambda^{-1} Y \end{bmatrix}$$

is factored into LDL' notation, also called Cholesky factorization. In the LDL' factorization factors a matrix such that L is a lower triangular with the elements on the diagonal equal to one, D is a diagonal matrix with the elements on the diagonal being the elements of the input matrix and L' is the transpose of L. Only the diagonal and the lower triangular matrices are used for computation [Mat2004]. If H is positive definite, then after factoring, all diagonal elements of D associated with an original diagonal element of H must be positive and every element associated with an original diagonal element of $\Lambda^{-1} Y$ must be negative. Thus after factoring, the diagonal elements of D are checked if the sign of the diagonal elements associated with H are positive and those associated with

the diagonal element of $\Lambda^{1} Y$ are negative. If this is the case, H is positive definite and no perturbation is required. Otherwise, let η_0 denote the value of the element of D with the largest magnitude, then an initial perturbation is applied with $\eta = 1.2\eta_0$. For an 1x1 matrix, one perturbation will produce a positive definite matrix, however for a larger matrix this may not be the case. If the matrix is not positive definite after one perturbation, the perturbed matrix is factored again as before and perturbed again. If the perturbation proves to be insufficient the matrix is perturbed again until a positive definite matrix is found.

This perturbation does not guarantees that the algorithm converges to a local minimum although it has been shown to work well in practice [VS1991]. The algorithm then continues as discussed in Section 3.4.

4.6 Summary

This chapter discussed algorithms for solving quadratic programming problems for which the objective function is not necessarily convex, although it is probable that only a local minimum point will be found.

One of the algorithms discussed was Beale's algorithms which was introduced in Section 3.2.2, in the discussion of convex case algorithms. This algorithm can be used to obtain local minimum points for non-convex quadratic programming problems however cannot guarantee a global minimum point unless the objective function is known to be convex. The algorithm was not discussed again in detail, however the necessary rules that improve this algorithm for non-convex quadratic programming problems were described.

Another algorithm discussed in this chapter was Keller's algorithm. This algorithm is a generalization of the algorithm introduced by Dantzig and later by Van de Panne and Winston. It is based on the Kuhn-Tucker conditions like many of the convex case algorithms. It constructs a schema, similar to the simplex tableau for which there is a set of transformations.

The Interior Point method is a commonly used approach for solving optimization problems. It has been discussed in detail in Chapter 3. In this chapter, modifications of the interior

point method have been described that improve the algorithm when solving non-convex quadratic programming problems.

This chapter also briefly discussed the fixed charge problem, which can be formulated as a non-convex quadratic programming problem, which enables this problem to be solved by the algorithms discussed in the dissertation.

The next chapter will briefly introduce a selection of algorithms for solving non-linear programming problems as these algorithms can be used to solve the quadratic programming problems, which itself is a special case of the non-linear programming problem.

Chapter 5

General Non-linear Programming Algorithms

5.1 Introduction

The general non-linear programming problem is a problem for which the objective function and/or the constraints are non-linear. The algorithms discussed in the previous two section were specifically designed to solve quadratic programming problems. There are however, many general non-linear programming algorithms which can be used to solve quadratic programming problems. Figure 5.1 illustrates a simple two-dimensional example of a general non-linear programming problem.

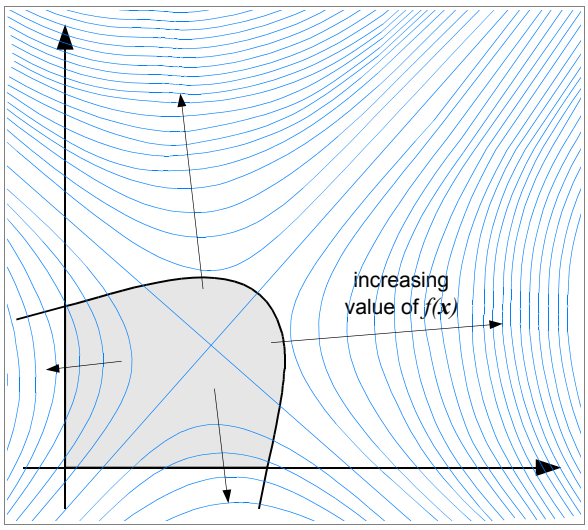

Figure 5.1 Non-linear programming problem

In this chapter a selection of algorithms for solving the general non-linear programming problem will be briefly introduced. The methods that will be discussed will not be described in as much detail as has been done in Chapter 3 and 4 as they do not form the primary focus of the dissertation. The methods will be briefly introduce outlining the basic concept of the algorithm.

Some of the methods for solving the general non-linear programming problem include the gradient-like methods, linear approximation methods, non-linear but unconstrained approximation methods. The gradient methods and the linear approximation method called the cutting plane method will now be introduced in some more detail.

5.2 Gradient Methods

Some of the most widely used computational techniques for solving non-linear programming problems are the gradient methods [Mil2000]. The gradient methods are well known for being used to solve unconstrained optimization problems. The idea of gradient methods has been used to attempt to find local optimum point for constrained optimization problem. Only if the convexity or concavity properties are known the gradient methods will converge to the global optimum.

The idea of these methods is to move toward the optimum along the best possible direction until a constraint forces a change in this direction. In general the gradient-based method can be outlined as follows:

1. Begin with an initial feasible solution $\boldsymbol{x}^0$.
2. Move along the direction of the negative gradient, $f(\boldsymbol{x}^0)$, for a minimization problem ($f(\boldsymbol{x}^0)$for a maximization problem) with a step size α^0, to a feasible point $\boldsymbol{x}^1$ for which $f(\boldsymbol{x}^1) < f(\boldsymbol{x}^0)$. In general for a feasible point $\boldsymbol{x}^{k-1}$, $f(\boldsymbol{x}^{k-1}) < f(\boldsymbol{x}^k)$.
3. The procedure stops when $f(\boldsymbol{x}^{k-1}) \approx f(\boldsymbol{x}^k)$.

As a more detailed example, consider the following gradient method. Consider the non-linear programming problem of the form

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad \boldsymbol{x} \ge 0 \end{aligned}$$

Assume that an initial feasible solution $\boldsymbol{x}^0$ exist. This can be obtained by Phase One of the simplex method if necessary. Then compute the initial direction $\boldsymbol{d}^{0\prime} = f(\boldsymbol{x}^0)$, where the vector $\boldsymbol{d}^0$ is a tangent to the function at the point $\boldsymbol{x}^0$. An iterative procedure is now developed that will generate a sequence of solutions $\boldsymbol{x}^k$ such that $f(\boldsymbol{x}^{k-1}) < f(\boldsymbol{x}^k)$. The next solution is then obtain by

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

Two cases are possible with reference to the initial solution x^0 :

1. An $\varepsilon > 0$ exists such that $x^1 = x^0 + \alpha d^0$, $d^{0\prime} = -f(x^0)$ is a feasible solution for all α where $0 \le \alpha \le \varepsilon$.
2. There exists no $\alpha > 0$ such that $x^1 = x^0 + \alpha d^0$ is a feasible solution.

Case 1 will be discussed first. Moving from x^0 in the direction $-f(x^0)$ produces the maximum rate of decrease in $f(x)$. It will now be shown that there exists a $\alpha > 0$, such that if

$$x^1 = x^0 + \alpha d^0, \quad d^{0\prime} = -f(x^0),$$

then x^1 is a feasible solution, and $f(x^1) < f(x^0)$. This can be shown by Taylor's theorem. For any α

$$f(x^0 + \alpha d^0) = f(x^0) + \alpha(-f(\xi)) d^0,$$
$$\xi = x^0 + \vartheta \alpha d^0, \quad 0 \le \vartheta \le 1 .$$

But, each $\partial f / \partial x_j$ is a continuous function of x , therefore $-f(x) d^0$ is a continuous function of x . Since $d^{0\prime} d^0 > 0$, there exists a $\delta > 0$ such that $-f(x) d^0 < 0$ for all x , $|x - x^0 < \delta|$, therefore for any α , $0 < \alpha < min(\varepsilon, \delta)$ it follows that $x = x^0 + \alpha d^0$ is a feasible solution and $f(x) < f(x^0)$.

The next step is to determine the value of α such that a new feasible solution x is obtained which produces the greatest possible decrease in $f(x)$. We first determine the largest value of α for which $x = x^0 + \alpha d^0$ is feasible. To do this it must be true that

(5.2.1) $$A(x^0 + \alpha d^0) \le b$$

(5.2.2) $$x^0 + \alpha d^0 \ge 0 .$$

Denote by M the subset of constraints (5.2.1) which are active at x^0 and N the subset of constraints (5.2.2) for which $x^0 = 0$. Since it was assumed that a move in the direction d^0 of finite distance can be made without violating the constraints, it must be true for any $\alpha > 0$ that

(5.2.3) $$a_r(x^0 + \alpha d^0) \le b_r, \quad r \in M$$

(5.2.4) $$x_p^0 + \alpha d_p^0 \ge 0, \quad p \in N ,$$

where a_r denotes the rows of the matrix A for which the corresponding constraints are active at x^0 , that is, $a_r x^0 = b_r$.

The value of α is limited by the fact that when α is made sufficiently large, some constraints that are not active at $\boldsymbol{x}^0$ usually become active or some components $x_p,\ p \quad N$ of $\boldsymbol{x}$ which were positive at $\boldsymbol{x}^0$ become zero. To determined the largest value of α for which $\boldsymbol{x} = \boldsymbol{x}^0 \quad \alpha \boldsymbol{d}^0$ is feasible, the following values are computed:

$$\rho = \begin{cases} \min\limits_{p} \left[\dfrac{x_p^0}{d_p^0} \right], d_p^0 < 0, p = 1,2,\ldots,n, \quad p \quad N \\ \infty \quad \text{if no} \quad d_p^0 < 0, p = 1,2,\ldots,n, \quad p \quad N; \end{cases}$$

and

$$\gamma = \begin{cases} \min\limits_{r} \left[\dfrac{b_r \quad \boldsymbol{a}_r \boldsymbol{x}^0}{\boldsymbol{a}_r \boldsymbol{d}^0} \right], \boldsymbol{a}_r \boldsymbol{d}^0 > 0, r = 1,2,\ldots,m, \quad r \quad M, \\ \infty \quad \text{if no} \quad \boldsymbol{a}_r \boldsymbol{d}^0 > 0, r = 1,2,\ldots,m, \quad r \quad M. \end{cases}$$

If ρ is finite, then some components of $\boldsymbol{x}$ will be negative when $\alpha > \rho$ and if γ is finite then some constraints will be violated when $\alpha > \gamma$.

Let $\varepsilon = min(\rho, \gamma)$. Then α must satisfy $0 < \alpha \le \varepsilon$ in order to satisfy the feasibility of $\boldsymbol{x}$. However, α cannot just be set to ε because $f(\boldsymbol{x})$ changes with $\boldsymbol{x}$. Thus even though $f(\boldsymbol{x}^0 \quad \alpha \boldsymbol{d}^0) < f(\boldsymbol{x}^0)$ if α is small enough, this may not be the case if α is too large, since $f(\boldsymbol{\xi}) \boldsymbol{d}^0$ may become negative. Thus the value of α has to be selected such that $0 < \alpha \le \varepsilon$ which minimizes $f(\boldsymbol{x}^0 \quad \alpha \boldsymbol{d}^0)$. This can be achieved in a number of ways:

1. Set $\frac{df}{d\alpha} = 0$ and solve for α. This can however only be done in simple cases.
2. Usually this kind of problem is solved using a computer and the value of α is obtained by using some numerical search procedure, for example, a simultaneous method mentioned. Here the interval $0 \le \lambda \le \varepsilon$ is subdivided by a number of equally spaced points $\alpha_v = v \Delta \alpha$ and $f(\boldsymbol{x}^0 \quad \alpha_v \boldsymbol{d}^0)$ is evaluated for each v. Then the α_v yielding the smallest value of $f(\boldsymbol{x}^0 \quad \alpha_v \boldsymbol{d}^0)$ is selected for α.

At this point the new direction and the step length have been calculated, therefore the new solution is computed by $\boldsymbol{x}^1 = \boldsymbol{x}^0 \quad \alpha \boldsymbol{d}^0$. Once the value of $\boldsymbol{x}^1$ has been found, the new solution point is obtained and the procedure is applied again as described above. The value $\boldsymbol{x}^0$ is replaced by $\boldsymbol{x}^1$ and the values of $\boldsymbol{d}^{1\prime}$, ε and α are determined. The new solution point $\boldsymbol{x}^2$ can then be computed. The procedure continues in this manner from a feasible solution $\boldsymbol{x}^k$ to a new feasible solution $\boldsymbol{x}^{k \ 1}$ such that $f(\boldsymbol{x}^{k \ 1}) < f(\boldsymbol{x}^k)$ until a termination criteria is reached.

The iterative procedure stops either when a feasible solution $\boldsymbol{x}^k = \hat{\boldsymbol{x}}$ has the property that $f(\hat{\boldsymbol{x}})\hat{\boldsymbol{d}} \geq 0$, or a feasible solution $\boldsymbol{x}^k$ is reached for which $\boldsymbol{x} = \boldsymbol{x}^k \quad \alpha \boldsymbol{d}^k$ is feasible for any $\alpha > 0$ and $f(\boldsymbol{x}^k \quad \alpha \boldsymbol{d}^k) < f(\boldsymbol{x}^k)$ for all $\alpha > 0$. At this point it is desirable to increase α to infinity, so that $|\boldsymbol{x}| \to \infty$. This means that the solution of the problem is unbounded. It is possible that neither of these two termination cases will be reached in a finite number of steps [Had1964].

Consider now case 2 where it is possible that there exists no $\alpha > 0$ such that $\boldsymbol{x}^{k \quad 1} = \boldsymbol{x}^k \quad \alpha \boldsymbol{d}^k$ is a feasible solution. If this happens then the current feasible solution is on the boundary of the set of feasible solution and either one or more constraints and/or the non-negativity restrictions will be violated if a move is made in the direction of the gradient vector. In this case the principle for selecting the direction in which to move is modified so that no constraint or non-negativity restriction is violated. The modification of the principle will not be discussed here.

5.2.1 Zoutendijk's Method of Feasible Directions

This is another method that uses the gradient approach and was introduced by G. Zoutendijk in 1960 [KKO1966] [BSS1993]. This method is used to solve non-linear programming problems with linear constraints.

Consider the non-linear programming problem of the form

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) \\ &\textit{Subject to} && \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & && \boldsymbol{x} \geq 0. \end{aligned}$$

The procedure starts from an initial feasible solution $\boldsymbol{x}^0$ such that $\boldsymbol{A}\boldsymbol{x}^0 \leq \boldsymbol{b}$ and attempts to find a new solution that gives a better value of $f(\boldsymbol{x})$. To obtain the best possible direction to the next solution, the method solves small linear or non-linear subprograms, for example, by the simplex method.

Given a feasible point $\boldsymbol{x}^k$, let M be the set of indices that represent the active constraints, that is, $\boldsymbol{a}'_r \boldsymbol{x}^k = \boldsymbol{b}_r$ for $r \quad M$. The direction vector $\boldsymbol{d}^k$ is determined by the solution to the linear programming problem

$$\textit{Minimize} \qquad f(\boldsymbol{x}^k)\boldsymbol{d}$$

$$\textit{Subject to } \boldsymbol{a}'_r \boldsymbol{d} \le 0 \quad r \quad M$$
$$\sum_{r=1}^{n} |d_r| = 1$$

where $\boldsymbol{d} = (d_1, \ldots, d_n)$. The equation $\sum_{r=1}^{n} |d_r| = 1$ is a normalizing equation that ensures a bounded solution. The other constraints assure that the vector of the form $\boldsymbol{x}^k \quad \alpha \boldsymbol{d}^k$ is feasible for a sufficiently small $\alpha > 0$. Subject to these conditions $\boldsymbol{d}$ is chosen to line up as closely as possible with the negative gradient of f.

The overall procedure progresses by generating feasible directions and moving along them to decrease the objective function. The transition from the k^{th} iteration point $\boldsymbol{x}^k$ to $\boldsymbol{x}^{k\ 1}$ is given by $\boldsymbol{x}^{k\ 1} = \boldsymbol{x}^k \quad \alpha \boldsymbol{d}^k$. The step length is determined by computing the following line search problem

$$\textit{Minimize} \quad f(\boldsymbol{x}^k \quad \alpha \boldsymbol{d}^k)$$
$$\textit{Subject to} \quad 0 \le \alpha \le \alpha_{max},$$

where the value of α_{max} is determined by

$$\alpha_{max} = \begin{cases} minimum\{\hat{b}_i / \hat{d}_i : \hat{d}_i > 0\} & \text{if } \hat{\boldsymbol{d}} \not\le \boldsymbol{0} \\ \infty & \text{if } \hat{\boldsymbol{d}} \le 0 \end{cases}$$
$$\hat{\boldsymbol{b}} = \boldsymbol{b}_r \quad \boldsymbol{A}_r \boldsymbol{x}^k, \quad r \quad M$$
$$\hat{\boldsymbol{d}} = \boldsymbol{A}_r \boldsymbol{d}^k, \quad r \quad M.$$

At this point the new direction and the value of α have been obtained and therefore the new point can be computed according to $\boldsymbol{x}^{k\ 1} = \boldsymbol{x}^k \quad \alpha \boldsymbol{d}^k$. The process continues again until $f(\boldsymbol{x}^k)\boldsymbol{d} = 0$.

5.2.2 Gradient Method Variations

Many variations of the gradient based method have been developed. Very much like in the variations of the simplex method, the basic idea of all gradient methods remains the same with small variation on different selection criteria of the best direction in which to move from the current feasible solution, or the calculation of the step length. These variations will not be discussed here. Some of the variations of the gradient based method are: the Lagrangian gradient method, Rosen's gradient projection method introduced in 1960 by J. B. Rosen [KKO1966] and the method of feasible directions of Topkis and Veinott introduced in 1967 [BSS1993].

5.3 Cutting Plane Methods

Another class of algorithms for solving non-linear programming problems use linear approximation to find the solution. One of these methods is known as the Cutting Plane method. These methods solve the non-linear programming problem by approximating the non-linear function and the constraints with straight line segments. The linear problem is then solved by using linear programming algorithms, for example, the simplex method. The solutions to the linear problems should then converge to the solution of the original problem [Lue1973]. Figure 5.2 illustrates a minimization problem of two variables with a non-linear objective function and one non-linear constraint.

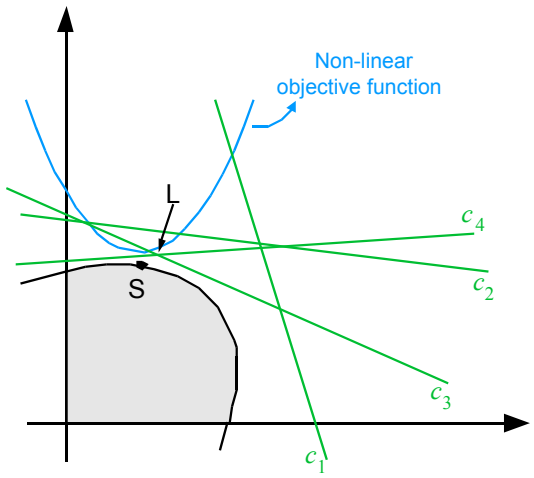

Figure 5.2 Linear inequalities containing a non-linear feasible region

The lines c_1, c_2, c_3 and c_4 represent four artificially created boundaries (cutting lines). The point S represents a true optimum and L represents the optimum to the linear problem with the same objective function and the four linear constraints c_1 to c_4. The idea is to create a sequence of cutting planes, one at a time, so that L approaches S. The solution to the problem is an approximation and will in most cases not be exact thus the procedure terminates when L lies close to the optimal solution, that is, let L^k be the k^{th} solution then the process terminates when $L^k - L^{k-1} \leq \varepsilon$. The value of ε is determined by a small positive number that is specified before the start if the procedure.

The Cutting Plane methods are only applicable to convex problems and are not very efficient. Usually however, they are easy to implement. The cutting plane procedure will now be introduced in more detail.

Consider a non-linear programming problem of the form

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) \\ &\textit{Subject to} \quad g(\boldsymbol{x}) \le \boldsymbol{0} \\ &\qquad\qquad \boldsymbol{x} \ge 0 \end{aligned} \tag{5.3.1}$$

Provided that the objective function is non-linear, the non-linear problem must be converted to a problem with a linear objective function and non-linear constraints. In order to do this conversion a new variable x_{n+1} is defined so that $x_{n+1} \le f(\boldsymbol{x})$, that is $f(\boldsymbol{x}) - x_{n+1} \ge 0$. This new constraint is then added to the set of constraints and the objective function is replaced with x_{n+1}. Thus we have the same problem with a linear objective function and $m+1$ non-linear constraints as follows

$$\begin{aligned} &\textit{Minimize} \quad x_{n+1} \\ &\textit{Subject to} \quad g(\boldsymbol{x}) \le \boldsymbol{0} \\ &\qquad\qquad f(\boldsymbol{x}) - x_{n+1} \le 0 \\ &\qquad\qquad \boldsymbol{x} \ge 0 \end{aligned} \tag{5.3.2}$$

Provided that the objective function was already linear, this conversion is omitted. It can be shown that the objective function and the Kuhn-Tucker conditions to the two problems (5.3.1) and (5.3.2) are the same and therefore points that satisfy (5.3.1) are the same as those that satisfy (5.3.2). Therefore the two problems have the same optimal solution. The proof of this statement will not be given here [Mil2000].

The next step of this method is to construct the cutting planes. This is done by linearization of the constraints expressed by using the first order Taylor's series approximation at the current point. To illustrate this concept, consider one constraint $g(\boldsymbol{x}) \le 0$ that depends on n variables. Now consider a feasible point $\boldsymbol{x}^k$. Drop this point down to the surface of the feasible region below the $x_1, x_2, \ldots, x_n$ plane and construct a plane that is tangent to the surface at that point. This tangent plane is the first order Taylor's series linearization of $g(\boldsymbol{x})$ at the point $\boldsymbol{x}^k$. The constraint function $g(\boldsymbol{x}^k)$ is approximated for $\boldsymbol{x}$ in the neighborhood of $\boldsymbol{x}^k$ by

$$g(\boldsymbol{x}) = g(\boldsymbol{x}^k) + [\nabla g(\boldsymbol{x}^k)]'(\boldsymbol{x} - \boldsymbol{x}^k)$$

and therefore the linearization of the inequality constraint around $\boldsymbol{x}^k$ is

$$g(\boldsymbol{x}^k) \quad [\quad g(\boldsymbol{x}^k)]'(\boldsymbol{x} \quad \boldsymbol{x}^k) \leq 0.$$

The cutting plane method now proceeds as follows:

Step 1	The first step is to find some initial point $\boldsymbol{x}^0$. This point does not need to be feasible.
Step 2	Solve the linear problem in which each constraint is replaced by it's first order Taylor series approximation. *Minimize* $f(\boldsymbol{x})$ *Subject to* $g_i(\boldsymbol{x}^0) \quad [\quad g_i(\boldsymbol{x}^0)]'(\boldsymbol{x} \quad \boldsymbol{x}^0) \leq 0 \quad$ for $i = 1, \ldots, m$ $\boldsymbol{x} \geq 0$ This can be done by using the simplex method. Denote this solution by $\boldsymbol{x}^1$.
Step 3	The constraint that is most violated by $\boldsymbol{x}^1$ is now selected. This constraint is then linearized around $\boldsymbol{x}^1$ and added to the set of linear constraints used in Step 2. The resulting linear program is solved and the solution is denoted by $\boldsymbol{x}^2$.
Step 4	Suppose that a sequence of solutions $\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^k$ has been found. Then the procedure stops if $g_i(\boldsymbol{x}^k) \leq \varepsilon$, where ε is some small positive number that has been predetermined before the start of the procedure. If however, there is a constraint violated at $\boldsymbol{x}^k$, then the constraint is linearized and added to the linear problem whose solution gave $\boldsymbol{x}^k$. The optimum is then found of this new linear problem and denoted by $\boldsymbol{x}^{k \ 1}$. This process continues until the stopping criteria has been reached.

This procedure will obtain the global minimum for the problem only if the objective function and the constraints are convex functions of $\boldsymbol{x}$.

5.4 Other Methods

Other methods which use approximation are, separable programming and the sequential unconstrained technique. In separable programming the non-linear function is again represented as a sum of several linear functions of a single variable each. The problem is now linear and can be solved using linear techniques, for example the simplex method. The solution is only an approximation to the original problem and its accuracy depends on how well the approximating linear program represents the original non-linear problem. The

sequential unconstrained technique uses a different idea. Instead of approximating the problem with linear problems it uses a series of unconstrained non-linear maximization or minimization problems. This technique is also the approach that is used in many of the path following interior point methods as it also uses a barrier function.

The interior point method has become increasingly popular as it seems to work rather well for most problems. This method was discussed in detail in Chapter 3 and Chapter 4 for solving specifically the quadratic programming problem. Due to its popularity the method has been modified to solve nonlinear programming problems [BSV2002].

Another class of problems that can be used for solving optimization problems are the Genetic algorithms. Genetic algorithms are a class of search techniques that use simplified forms of the biological processes of selection. The basic idea of these algorithms is to begin with a randomly selected initial population and evaluate the fitness of the members and the fittest members are selected. The selected members are then used to produce a new population. The old population is then replaced with the new population and again tested for fitness. This process continues until the population is deemed fit enough. Genetic algorithms itself is a large research area and a detailed explanation thereof does not fit within the scope of this dissertation [Cha2002].

5.5 Summary

This chapter briefly discussed a selection of algorithms for solving non-linear programming problems. Since the quadratic programming problem is a non-linear programming problem the algorithms used for solving non-linear programming can be used for solving quadratic problems. This is also the reason for introducing these algorithms. The approach these methods use to solve unconstrained problems, has been used to attempt to solve or at least to find a local optimum for constrained optimization problems. The global optimum can only be guaranteed if the problem is linear or if the objective function and the constraints are convex functions.

One of the well know techniques for solving unconstrained optimization problems are the gradient based methods. These methods are iterative procedures that attempts to find the

solution by moving towards the optimum from one solution point to another, which hopefully improves the objective function value. This is done by calculating the direction (gradient) and the step length in which to move towards the next feasible point. Many gradient method variations have been developed one of which has been briefly introduced here, in particular Zoutednijk's method of feasible direction. These variations differ in the calculation of the direction and the step length.

Another technique for solving the non-linear programming problem is to approximate the problem either with linear functions or a set of unconstrained problems. Cutting plane methods and separable programming are techniques that use linear approximations and the sequential unconstrained technique replaces the original problem with an unconstrained but still non-linear problem.

Amongst other methods that can be used to solve non-linear programming problems is the interior point method which falls into the class of barrier methods. This method, first introduced in Chapter 3, is one of the most widely used methods for solving optimization problems. Another technique for solving optimization problems are the genetic algorithms, which itself is a wide research area.

Chapter 3 to 5 introduced a selection of algorithms for solving quadratic and general non-linear programming problems. In the next chapter some of these algorithms will be selected in order to compare the efficiency of some of these algorithms by using a selection of test examples.

Chapter 6

Evaluation

6.1 Introduction

In the previous chapters several algorithms for solving the quadratic optimization problem were discussed. In this chapter a selection of these algorithms and their efficiency will be compared.

The efficiency of an algorithm, depends on a number of factors. The first is the complexity of the problem. This depends on the size of the programming problem, measured in terms of the number of unknown variables and the number of constraints [Lue1973]. The size of programming problems can be classified into three different types:

Small scale problems, which have five or less variables and constraints;

Intermediate scale problems, which have from five to a hundred variables and constraints; and

Large scale problems, for which the number of variables and constraints range from a hundred to thousands and more.

Another factor is the level of complexity of the algorithm. This is determined by the difficulty of the mathematical computations and the time required to compute the calculations at each iteration. This factor also effects the amount of resources needed for the calculations.

These factors effect the efficiency of the algorithm in terms of the number of iterations and the time it takes to obtain the optimal solution. In the following section a selection of algorithms will be compared in terms of the complexity and the number of iterations it has taken to obtain the solution.

6.2 Comparison of Algorithms

The efficiency of algorithms can be compared theoretically, by considering all computations necessary for each iteration. The complexity of the computations can provide an indication

of the efficiency of one algorithm compared with another. The theory, may provide an upper bound or a worst case scenario, on the number of iterations an algorithm will need to obtain the optimal solution. However, in many cases the solution is obtained in fewer iterations, therefore the upper bound cannot conclusively be used to determine whether one algorithm is better than another. Another way to compare the algorithms is to compare the number of iterations required to solve the same set of problems.

However, it is possible for an algorithm to obtain a solution in more iterations than another, but in a shorter period of time. Therefore iterations alone are not necessarily an indicator of efficiency and the amount of time taken to solve the given problem is also required.

From experience, it is known that the efficiency of an algorithm may vary from one problem to another even if the size of the problem remains the same. For example, algorithm 1 can solve a problem more efficiently than algorithm 2. It is possible that by changing some of the constraints, without changing the size of the problem that algorithm 2 will be more efficient than algorithm 1.

To further complicate matters, problems are usually solved using computers, therefore the efficiency of an algorithm also depends on how well the algorithm was implemented and other factors, such as the memory management of the operating system. Therefore it cannot be said with certainty, that an algorithm will always be better than another, it is only possible to generalize and state that one algorithm will usually be better than another.

The following sections will contain a theoretical comparison of some of the algorithms discussed in the dissertation and a quantitative evaluation in terms of the number of iterations. Two factors determined the selection of algorithms to be compared. First, is the frequency of use of the algorithms in practice and second, the availability of software for testing. As already mentioned Wolfe's method, the interior point method and the gradient method are the most commonly used and implemented methods for solving quadratic programming problems. Due to this popularity and the availability of software, these algorithms were selected for comparison. There was no software available for any of the remaining algorithms. Beale's algorithm and Theil-Van de Panne method were selected because they are not variations of either the simplex method, the interior point method nor

the gradient method and their comparison may provide information as to why these methods are not widely used in practice. Beale's algorithm was selected for implementation as it can solve a wider range of problems than Theil-Van de Panne and Dantzig's algorithms, which will only be compared theoretically.

6.2.1 Theoretical Comparison of Selected Algorithms

Wolfe's method

One of the most commonly used algorithms for solving quadratic programing problems is Wolfe's algorithm. This algorithm modifies the simplex method for linear programming for solving quadratic programming problems. It has been shown that the simplex method is an exponential time algorithm. An n-dimensional feasible region has 2^n vertices. In the worst case of the simplex method, all the vertices will be visited and thus the optimum will be found in exponential time [Win1995]. Many variations of the simplex method have been implemented. These differ in the selection criteria for variables that will leave and enter the basis or in the manner in which they solve the linear system produced by the Kuhn-Tucker conditions [Boo1964]. It has been shown by M. J. Best [Bes1984] that many variations in the simplex method are equivalent in the sense that they construct identical sequences of iterates when applied to a common problem.

Wolfe's method and Dantzig's Method

One of the variations of the simplex method discussed in the dissertation is Dantzig's algorithm. Both the Wolfe's algorithm and Dantzig's algorithm begin with the Kuhn-Tucker conditions. Computationally, Dantzig's algorithm may have an advantage as it can solve problems for which the matrix $\boldsymbol{Q}$ is positive semi-definite without any further modifications unlike in Wolfe's algorithm, which uses the long form for solving such problems [VW1964]. In terms of iterations however, these algorithms are similar in the fact that, they use the same simplex transformation to go from one tableau to the next. These algorithms differ in the selection criteria for the variables that are to enter the basis and to leave the basis [Boo1964]. Dantzig's algorithm may have an advantage over Wolfe's algorithm provided that an initial basic feasible solution is immediately evident. However, if the initial basic feasible solution is not available, another method, for example phase one of the simplex method can be used to obtain this solution. Wolfe's method however, always uses phase one of the simplex method to obtain an initial basic feasible solution. Both of these

algorithms can be easily implemented as very few modifications of the simplex method for linear programming are needed to turn the code into code for quadratic programming.

Wolfe's method and Beale's method

Beale's algorithm is similar to Wolfe's method, even though Wolfe's method uses Kuhn-Tucker conditions and Beale's method does not. Both of these methods use simplex transformation to compute the next solution. The main advantage of Wolfe's algorithm is the ease of implementation as stated in the above paragraph. This is not the case with Beale's method as it requires the calculation of matrix inverse and partial derivatives at each iteration. Due to these reasons, the computations of Beale's method at each iteration are more complex than Wolfe's method. The advantage of Beale's method is that extensions of the algorithms are more easily developed and Beale's method can solve a wider range of problems, specifically the non-convex case problems [Bea1959]. Wolfe's method may however, be more efficient in terms of the number of iterations. Wolfe's method requires m iterations at the beginning to eliminate the s-variables from the basis, which corresponds in Beales' method of expressing $\boldsymbol{Q}$ as a function of non-basic variables only. Then Wolfe's method requires at least $n \quad m$ further iterations to make all the w_i non-basic. Some of these iterations may not be needed in Beale's method provided that the first trial solution is good. Beale's method however, may introduce free variables and therefore a final solution cannot be obtained until all the free variables have been removed [Bea1959]. Therefore, in terms of efficiency, Beale's algorithm and Wolfe's algorithm will produce the same number of iterations provided that no free variables need to be introduced during the computational procedure of Beale's method [DK1968].

Theil-Van de Panne algorithm

The efficiency of Theil-Van de Panne algorithm depends largely on the size of the problem. This method seems to work well as long as the constrained minimum satisfies few constraints exactly [TV1960]. This is due to the fact that every constraint that is violated is incorporated into the computation at the next iteration. At each iteration it must be verified whether all constraints are satisfied or not. Provided that the number of constraints is small this computation will not take much time, however, for large scale problems this procedure could be inefficient. A large number of constraints causes another problem. The calculations at each of the iterations require computing of the inverse of matrices, therefore

the more constraints that are violated the larger the matrices become and the calculation of the inverse becomes computationally expensive [TV1960]. It is possible that the initial solution can satisfy all the constraints, at which point the optimal solution would be reached in the first iteration, however, if this is not the case the procedure will only work well for small scale problems or problems for which few constraints are violated. As soon as the problems start to increase in size the likelihood of constraints being violated increases and the efficiency of the algorithm drops.

Wolfe's method and the interior point method

The most frequently used algorithms for implementation of commercially and freely available software packages are the simplex method and the interior point method. This is due to the fact that they work well in practice for solving a variety of problems. These methods are also the most commonly compared methods for solving linear and quadratic programming problems. It has been shown that the interior point methods may work better for large scale problems with more than 100 000 constraints and variables, however for smaller problems the simplex method works as well or better [Van2001]. In the worst case the simplex method has exponential complexity, while the interior point method has polynomial complexity. In practice however, the simplex method also has polynomial complexity and therefore the theoretical difference does not have much significance [Mil2000]. As discussed earlier (Section 3.2 and Section 3.4) the simplex method finds the solution on the exterior of the feasible region while the interior point method are in the interior. Provided that there is a unique solution both methods will find the same solution however, for non-convex problems this may not be the case. Each iteration of the interior point method is computationally more expensive then in the simplex method as it requires the computation of an inverse. Much research has been done to find a way of finding the inverse efficiently, so that the matrices are no longer inverted explicitly. The inverse is calculated by solving a series of related triangular linear systems, which involves finding lower triangular matrix. These methods can be implemented very efficiently when dealing with sparse matrices [JB2003].

Simplex methods tend to hinder the efficiency of the algorithm when solving large, degenerate problems. Such problems typically arise in transportation and scheduling applications. The accompanying formulations are often very sparse thus making them

prime candidates for interior point methods. When the formulations are dense, or in some cases when only a few columns of the A matrix are dense, interior point methods are not likely to perform well. Such problems nullify the efficiencies associated with solving triangular systems. Dense problems, however, are not in general degenerate so simplex-type methods are the better choice [JB2003].

As with simplex methods, there are many variations of the interior point method. Each of these variations is been designed to improve the efficiency and speed of the currently available implementations. Many articles are available comparing the efficiency of different interior point methods, [VS1991], [UU2000], [Gor1999]. For example in an article on interior point algorithms by R. J. Vanderbei and D. Shanno [VS1991], they compare two different implementations of the interior point method with their implementation and show that their methods outperforms the other implementations for both convex and non-convex problems.

The gradient methods

The gradient method is one of the oldest and simplest methods used for solving optimization problems. The rate of convergence of the gradient methods depends on the selection of the starting point and the size of the step length. If the size of the step length is chosen too small the convergence rate will be slow, if the size of the step length is chosen too large the iteration may overstep the optimum point. This could then happen repeatedly and the method may not converge to a solution [Lue1973]. The convergence rate also depends on the ratio of the lengths of the axes of the elliptical contours of f, that is, on the eccentricity of the ellipsoid. This means that as the contours of f become more eccentric, the convergence rate becomes slower [Lue1973]. An iteration of the gradient method is computationally less expensive that an iteration for the other algorithms as it does not require the calculation of the inverse of matrices or any other matrix operations like the other methods.

6.2.2 Quantitative Evaluation

As mentioned previously, it is not the purpose of this dissertation to formulate the quadratic programming problem but rather to solve it, therefore random problems were generated by the author, in order to obtain the quantitative data needed to compare the efficiency of the algorithms. All the problems generated for evaluation are small scale, convex quadratic

programming problems. The problems are small scale due to the fact that the software available for testing did not allow for a uniform input of data, and the problems had to be entered manually which would make entering problems with hundreds of variables an extremely time consuming and error prone process. All the problems generated are given in the form

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad\quad \boldsymbol{x} \ge 0 \end{aligned}$$

and can be seen in Appendix C.1. It is also assumed that $\boldsymbol{b} \ge 0$.

One of the deliverables of this dissertation is to collect software, freely available in the public domain, that can be used for solving quadratic programming problems. A trial version of Xpress Solver developed by Frontline Systems [Fro2002] was used to evaluate Wolfe's method and the gradient method. The gradient method to be compared is the conjugate gradient method. A student version of Xpress IVE developed by Dash Optimization [Das2002] was used to evaluate the interior point method. As mentioned earlier, Beale's algorithm was selected for implementation and was implemented using the Visual Basic programming language.

Since the problems generated for evaluating the algorithms are small scale problems they can be solved in a very short period of time, typically in less than 0.2 seconds. Additionally, it would be unrealistic to compare the time taken to solve the problem by commercial applications with those implemented by the author as the time depends on how well an algorithm was implemented. By comparing the time, the efficiency of the implementation will also be compared rather than only the efficiency of the algorithm, which is not desired result. For these two reasons, the evaluation will focus on the number of iterations and not the time required to obtain the solution. In fact, even by comparing the number of iterations, it cannot be concluded that one algorithm is more efficient than another. For example, an iteration for algorithm 1 is less complex than for algorithm 2, however, algorithm 1 solves a problem in say twice as many iterations as algorithm 2. Therefore, although algorithm 1 computed the solution in more iterations, the overall computations required may be the same for both algorithm. As stated earlier, the gradient methods is a method for which one iteration is less computationally complex than an iteration for the other methods. Table 6.1 shows the number of iterations observed in solving each of the problems.

Table 6.1 The comparison of iterations for different methods.

			Number of Iterations			
	n	**m**	**Wolfe's Method**	**Beale's Algorithm**	**Interior Point Method**	**Conjugate Gradient Method**
Problem 1	3	3	5	3	5	10
Problem 2	4	3	6	4	6	14
Problem 3	4	4	5	5	6	10
Problem 4	5	2	6	5	6	11
Problem 5	5	4	4	6	4	7
Problem 6	6	5	6	10	6	11
Problem 7	7	4	8	8	8	16
Problem 8	8	5	5	5	5	10
Problem 9	9	6	6	6	6	16
Problem 10	10	7	9	12	9	16

Wolfe's algorithm, the interior point method and the gradient method produced the same objective function values for all problems in Table 6.1. In Beale's algorithm some of the solutions differed slightly in one or two decision variables, due to small rounding errors.

As stated before, the approach used by Wolfe's algorithm and Beales' algorithm are similar. Both methods use simplex transformations, however, Beale's algorithm requires the computation of an inverse, several other matrix operations and the calculation of partial derivatives at each iteration and thus making it more complex. The size of the matrices in Beale's method is determined by the size of the problem, therefore for large scale problems it is clear that Beale's method will have a much larger computational overhead compared to Wolfe's method and the gradient method. From Table 6.1 it can be seen that Beale's method and Wolfe's method required roughly the same number of iterations for most of the problems. As stated previously, if Wolfe's method and Beale's method have the same number of iterations it can be said that no free variables entered the basis of Beale's method. However, this is not always the case, as it is possible that some of the variables were equal to zero and therefore did not have to enter the basis as was the case with some of the problems.

The interior point method, similarly to Beale also requires the matrix inverse and other matrix operations in every iteration. There are many articles available comparing Wofle's

algorithm and the interior point method [Mil2000] [Van2001] [JB2003]. These articles show that both of these methods work well in practice for most types of problems. As can be seen in Table 6.1, both methods solve the problems in less than ten iterations, in fact Wolfe's method and the interior point method solved all problems but one, in the same number of iterations. As every iteration in the interior point method requires more computations than Wolfe's method, it could be said that Wolfe's method will be more efficient if the problem is solved with the same or less number of iterations than the interior point method. However, as shown by R. J. Vanderbei [Van2001] the interior point method seems to work better in practice for large scale problems.

From Table 6.1 it can be seen that the gradient method takes almost twice as many iterations to obtain the solution as Wolfe's method and the interior point method. However, one iteration of the gradient method requires fewer computations than an iteration of the other methods, therefore this method is not necessarily less efficient. Also as mentioned earlier, the rate of convergence of the gradient method depends on the initial starting point and the size of the step length. A number of tests were run on the problems, by changing the starting point for the algorithm and the results showed that if the starting point was closer to the actual solution the convergence rate was faster. Similarly the further away the starting point was from the actual solution the slower the convergence rate. Another test that was run, was changing the size of the step length. The number of iterations in Table 6.1 were obtained with a step length of 0.0001. The set of problems was then run by changing the size of the step length to 0.001 and 0.01. It was observed that the convergence rate improved for some of the problems by 1 or 2 iterations. It is possible to significantly improve the convergence rate of the gradient method by changing both the initial starting point along with the step size. This however, could prove to be a time consuming process as these values would have to be estimated individually for each of the problems. Although the gradient method may have a slower convergence rate than Wolfe's and the interior point methods, it remains a popular method due to its ease of implementation and is often used as a reference to other techniques [Lue1973].

In conclusion, it has become clear why Wolfe's method, the interior point method and the gradient method are used frequently in practice. Beales' method however, has not proven to be very popular, due to its computational overhead. From the theoretical discussion and

the results obtained in Table 6.1, it is difficult to conclude that one algorithm is better than another, as the complexity of the iterations plays a mayor role in the efficiency of the algorithm.

6.3 Available Software

There are many commercially available software packages for solving optimization problems. These packages have been designed to handle large scale problems and provide a wide range of features. Most of these packages however, are usually expensive. Free packages are available for solving optimization problems, that can be downloaded from the Internet. However, these packages are usually limited in their functionality and can only solve specific types of problems. Most of the free packages have been designed for teaching purposes and educational use.

Most of the available packages implement variations of the simplex method and the interior point method to solve optimization problems as these methods have proved to work well in practice. Some of the software packages available are given as follows:

The Xpress IVE software developed by Dash Optimization [Das2002] has a selection of algorithm that enables solving large scale linear and non-linear programming problems. The software provides its own programming language with which a problem can be specified. A free student version is available for download at the Dash Optimization web site.

Solver for Microsoft Excel is a package available from Frontline Systems [Fro2002]. The package provides add-ins for MS Excel, which provide a selection of algorithms and other criteria to be selected for solving optimization problems. The advantage of this package is that problems are specified using spreadsheets and wizards, therefore no coding is required. This makes the package considerably easier to use.

LANCELOT [Lan2002] is a free package written in Fortran for solving large scale linear and non-linear programming problems.

LOQO [Van2002] is a package written in C, implementing the interior point method, for solving linear and non-linear programming problems.

GOAL [Mar2003] is a free optimization tool based on genetic algorithms. It allows complex problems to be specified in Visual Basic Script.

There are many other commercially available packages available for solving optimization problems. A listing of many other software packages can be found on the following web site plato.la.asu.edu/guide.html.

6.4 Summary

The aim of this chapter was to provide a better understanding of the performance of algorithms and how they are evaluated in practice. The algorithms were compared theoretically and quantitatively by solving a number of convex quadratic programming problems.

The efficiency of algorithms can be determined by comparing various algorithms with respect to each other in terms of the complexity of iterations, the number of iterations and the time taken to obtain the solution. These characteristics are dependent on the algorithm and the problem and therefore the efficiency of the algorithms will vary for different problems. For this reason it difficult to conclude that one algorithm is better than another. In general, it can only be said that one algorithm is more efficient than another for a specific sample of practical problems.

In the theoretical evaluation the algorithms were compared by discussing the complexity of the computations at each iteration and the convergence rate of the algorithm. From the quantitative evaluation it was shown that Wolfe's method, Beale's method and the interior point method work well for most of the test problems. It was shown, that although the number of iterations for the gradient method was larger than for Wolfe's method and the interior point method, this method may not necessarily be less efficient as an iteration for the gradient method is less complex. It was also shown how the choice of the initial starting point and the size of the step length influences the speed of the convergence rate of the gradient method.
In addition to the evaluation of the algorithms, this chapter provided a brief survey of the software available for solving quadratic programming problems. There are many free and commercial software packages available capable of solving large scale optimization

problems of various types. Most of the packages implement a variation of the simplex and the interior point methods as these have proved to work well in practice.

Chapter 7

Summary and Future Research

7.1 Introduction

The goal of this dissertation was to provide a survey of quadratic programming algorithms. A further goal was to select some of the algorithms and evaluate them for efficiency. The primary motivation behind initiating this project was the importance of non-linear optimization. Although linear programming is powerful for representation of many real world situations, there are still many problems for which linear models are not sufficient, for example, the fixed charge problem. This creates the need for non-linear models and for algorithms capable of solving them. In order to restrict the scope of the dissertation, a special case of non-linear programming, namely quadratic programming was selected as the focus of the dissertation.

This chapter concludes the dissertation by reviewing the research conducted in this dissertation and discussing areas for future research.

7.2 Summary of Research

Optimization is the process whereby we seek to find the best or optimal value of a function, usually subject to certain constraints or restrictions. The function with it's restrictions is a mathematical optimization model that represents certain aspects of the physical environment [Ars2003]. Optimization problems arise in a wide variety of fields that include science, engineering, economy and management. The theory of optimization attempts to find solutions for these problems. This dissertation provides an overview of a selection of algorithms for solving a particular type of optimization problem, the quadratic programming problems.

Chapter 2 provided an overview of the basic mathematical concepts necessary for understanding the algorithms that were discussed in the further chapters. This chapter

discussed the criteria that are used to identify local and global minima and maxima and points of inflection for a function of one and many variables for unconstrained and constrained problems. An important result was established regarding convex and concave functions, which plays a mayor role in obtaining the solution for optimization problems. It has been established that if it is known that a function is convex or concave, then any local minimum or maximum point obtained is the global minimum or maximum point. However, if a function is neither convex nor concave, than the global optimum, nor in fact the local optimum may never be established.

Another important concept discussed in this chapter, was the Kuhn-Tucker conditions. These conditions provide a set of necessary and sufficient conditions for optimality for constrained problems and also provide the basis for many of the algorithms discussed in the dissertation. The Kuhn-Tucker conditions reformulate a non-linear programming problem to a set of linear equations with two non-linear equations. Provided that a solution is found for the linear system of equations, that satisfies the two non-linear equations, the solution for the original problem is obtained. The result suggests the possibility of solving quadratic programming problems by making use of modified linear programming algorithms.

In this chapter the general optimization problem was defined and specific cases of optimization problems have been identified according to their type of the objective function and constraints. These were: linear programming, quadratic programming and non-linear programming. The quadratic programming problem has a quadratic objective function and linear constraints and forms the focus of the dissertation.

Having established an understanding of the basic mathematical concepts, the next step was to discuss the algorithms for solving quadratic programming problems. Chapter 3 gave an overview of a selection of algorithms available for solving convex quadratic programming problems. The selection of the algorithms was based on the popularity of an algorithm and the availability of material. The algorithms were classified into three categories, namely the active set methods, the ellipsoid methods and the interior point methods.

The active set methods search for the optimum on the exterior of the feasible region. The idea of these methods is to add constraints for which a given criteria holds to an active set. The constraints are added to and removed from the active set depending on whether they satisfy the given criteria or not. One of the most widely used active set methods is Wolfe's algorithm which was described in detail in Chapter 3. This method is based on Phase One of the Two Phase simplex method for solving linear programming problems. It has been developed in two forms: a short form and a long form.

There are many variations of this method available and most software packages that solve quadratic programming problems implement a variation of Wolfe's algorithm. One of the variations of Wolfe's algorithm, namely Dantzig's algorithm was discussed in the dissertation. It was concluded that the primary distinction between the variations of Wolfe's algorithm is in the selection criteria for choosing the variables that enter and leave the basis and in the way the linear system obtained by the Kuhn-Tucker conditions is solved.

Other active set methods discussed in the dissertation were Beale's method and the Theil-Van de Panne procedure. Both Beale's method and Theil-Van de Panne algorithm do not begin with the Kuhn-Tucker conditions. Beale's method uses similar criteria for selecting the variables that enter and leave the basis as the simplex method and the same simplex transformation rules for computing the new tableau. The procedure of Theil-Van de Panne begins by obtaining a solution to the objective function without regard to any of the constraints. The solution obtained is then used to check if any of the constraints have been violated. If there are any violated a new solution point is obtained by incorporating each of the constraints in the next iteration.

Another type of method are the ellipsoid methods. These methods are not widely used and it has been shown that they do not work well in practice. For this reason these methods were not discussed in detail in this dissertation and are left for future research.

The third type of method that has received much attention is the interior point method, first proposed by N. K. Karmarkar [Mil2000] for solving linear programming problems. These methods differ greatly from the active set methods as they search through the interior of the feasible region rather than the exterior. These methods can be classified into two types:

affine scaling methods and path following methods. The affine scaling methods are no longer the method of choice as they are believed not to be polynomial time algorithms [Van2001] therefore the path following method was discussed in the dissertation. The interior point methods have become very popular and like the simplex method are implemented in most software packages.

Chapter 4 discussed a selection of algorithms for solving quadratic programming problems for which the objective function is not necessarily convex nor concave. For this type of problem the algorithms may not find the global optimum point, nor in fact, the local optimum point, however, often any improvement in the objective function value is preferred above no improvement at all.

Beale's algorithm and the interior point method have been discussed in detail in Chapter 3 for solving convex quadratic programming problems. Both of these methods can be used to obtain local minimum points for non-convex quadratic programming problems. Some modifications to these algorithms was discussed in Chapter 4 that improves the chance of obtaining the solution when solving non-convex quadratic programming problems.

Another algorithm that can be used to obtain local optimum points for non-convex quadratic programming problems discussed in Chapter 4, is Keller's algorithm. This algorithm is a generalization of the algorithm introduced by Dantzig and later by Van de Panne and Winston. It begins with the Kuhn-Tucker conditions like many of the convex case algorithms and then constructs a schema, similar to the simplex tableau. Keller's algorithm then selects from three possible types of transformations to construct a new schema. The choice of transformation depends on the characteristics of the current schema. A new schema is constructed until a stopping criteria is reached.

Chapter 4 also briefly discussed a problem that frequently arises in optimization problems, called the fixed charge problem. It was shown that the fixed charge problem (an integer programming problem) can be converted to a non-convex quadratic programming problem, which can be solved by the algorithms discussed in the dissertation.

Chapter 5 briefly discussed a selection of algorithms for solving general non-linear programming problems. These algorithms can be used to solve the quadratic programming problem as it is a restricted case of non-linear programming problems. The approach used by some algorithms for solving unconstrained problems has been used for algorithms for solving non-linear programing problems. One such method is the gradient method, widely used for solving unconstrained optimization problems.

The gradient method is an iterative procedure and searches for the minimum point by moving in the direction of the largest negative gradient and calculates the length of the step by which to move towards the next solution point. Many methods that use the gradient approach have been developed. One of these, Zoutednijk's method of feasible direction was briefly introduced in Chapter 5. The variations of the gradient methods differ in the calculation of the direction and the step length in which to move to the next feasible point.

Another technique for solving the non-linear programming problem is to approximate the problem with linear functions or a set of unconstrained problems. Cutting plane methods and separable programming are methods that use such techniques. The cutting plane method, discussed in Chapter 5, solves the non-linear programming problem by approximating the non-linear function and the constraints with straight line segments, called cutting planes. The linear problem that is constructed is then solved by using linear programming algorithms, for example, the simplex method. The idea therefore, is to create a sequence of cutting planes, one at a time, until the optimal solution is reached.

Other methods that can be used to solve non-linear programming problems are the interior point method discussed in Chapter 3 and 4 and genetic algorithms. The genetic algorithms were briefly mentioned in Chapter 5 as a solution method for solving non-linear programming problems. Genetic algorithms themselves are a wide research area and therefore do not fall within the scope of this dissertation.

Chapter 6 provided a theoretic and quantitative comparison of the efficiency of the quadratic programming algorithms. The theoretical evaluation compared the algorithms by discussing the complexity of the computations at each iteration and the maximum number of iterations needed to obtain the solution. In the quantitative evaluation a number of

problems were compared in terms of the number of iterations needed to obtain the solution. It was shown that Wolfe's method, Beale's method and the interior point method work well for most of the test problems, however Beale's algorithm may not be a popular choice for implementation due to its computational overhead. Although the number of iterations for the gradient method was larger than for Wolfe's method and the interior point method, the computations required at each iteration are considerably less than for the other methods, therefore the gradient method is not necessarily less efficient. It was also shown how the choice of the initial starting point and the size of the step length can improve or worsen the convergence rate of the gradient method. From the theoretical discussion and the quantitative results it can be seen that the efficiency of algorithms depends on the characteristics of an algorithm and the problem and therefore the efficiency of the algorithms will vary from one problem to another. For this reason it difficult to conclude that one algorithm is better than another. In general, it can only be said that one algorithm is more efficient than another for a specific sample of practical problems.

In addition to the evaluation of the algorithms, this chapter provided a brief survey of the software available for solving quadratic programming problems. Many free and commercial software packages are available capable of solving large scale optimization problems of various types. Most of the packages implement a variation of the simplex and the interior point methods as these methods have proved to work well in practice.

7.3 Future Research

Some of the possible projects for future research include the following:

The ellipsoid methods have only been mentioned in this dissertation. The dissertation could be extended with a detailed discussion on ellipsoid methods, providing algorithms and discussions on why these methods do not work as well as the other methods.

Conduct a study on the genetic algorithms for solving quadratic programming problems. Compare the efficiency of these algorithms with respect to the algorithms mentioned in this dissertation, for solving quadratic programming problems and other algorithms for solving general non-linear programming problems. Comparing the efficiency of these algorithms to other algorithms that may or may not have been

mentioned in the dissertation, for solving non-convex quadratic programming problems.

There are many more methods that have not been discussed in this dissertation and therefore it could be extended to incorporate other algorithms available for quadratic programming.

The execution of an algorithms could be represented graphically by using a diagram similar to a flow chart diagram. Each computational step and the sequence of execution could be represented in the diagram. This diagram can aid in the understanding of how an algorithm works. By assigning some cost value to each execution step in the diagram and recording the number of times each step was executed in solving the problem, it would be possible to visualize the "performance" of the algorithm. Further, the diagrams could be used to compare various algorithms and the performance of a single algorithm across several problems.

Bibliography

[And1998] E. D. Andersen, *Linear Optimization: Theory, Methods and Extentions*, Department of Management, Odense University, 1998.

[Ars2003] H. Arsham, *Deterministic Modeling: Linear Optimization with Applications*, ubmail.ubalt.edu/~harsham/opre640a/partVIII.htm, Retrieved June 2003.

[Bea1959] E. M. Beale, *On Quadratic Programming*, Naval Research Logistics Quarterly, Vol.6, 1959, pp.227-243.

[Bes1984] M. J. Best, *Equivalence of Some Quadratic Programming Algorithms*, Mathematical Programming, Vol.30, 1984.

[BF1993] R. L. Burden and J. D. Faires, *Numerical Analysis*, PWS Publishing Company, 1993.

[BJR+1997] A. B. Berkelaar, B. Jansen, K. Roos, and T. Terlaky, *An Interioir-Point Approach to Parametric Convex Quadratic Prgramming*, 1997.

[Boo1964] C. G. Boot, *Quadratic Programming: Algorithms, Anomalies, Applications*, North-Holland Publishing Company, 1964.

[BR2001] M. J. Best and K. Ritter, *Quadratic Programming Active Set Analysis and Computer Programs*, 2001, Unpublished.

[Bri1989] L. Brickman, *Mathematical Introduction to Linear Programming and Game Theory*, Springer-Verlag New York Inc., 1989.

[BSS1993] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley and Sons Inc., 1993.

[BSV2002] H.Y. Benson, D.F. Shanno, and R.J. Vanderbei, *Interior Point Methods for Nonconvex Nonlinear Programming: Complementarity Constraints,* Department of Combinatorics and Optimization, University of Waterloo, 2002.

[CC1978] Y. Y. Chang and R. W. Cottle, *Least-Index Resolution of Degeneracy in Quadratic Programming*, Mathematical Programming, Department of Operations Research, Stanford University, Vol.18, 1978, pp.127-137.

[Cha2002] P. Charbonneau, *An Introduction to Genetic Algorithms for Numerical Optimization*, National Center for Atmospheric Research, 2002.

[CR1999] V. Chandru and M. R. Rao, *175 Years of Linear Programming*, Resonance: The Journal of Science Education, 1999.

[CS1970] L. Cooper and D. Steinberg, *Introduction to Methods of Optimization on Acoustics, Speech, and Signal Processing*, W. B. Saunders Company, 1970.

[Das2002] Dash Optimization, *XPRESS IVE*, www.dashoptimization.com, Retrieved February 2002.

[DK1968] H. C. De Kock, *Sommige Metodes en Toepassings van Kwadratiese Programmering (Some Methods and Applications of Quadratic Programming)*, 1968.

[DK1975] G. de V. De Kock, *The Fixed Charge Problem*, 1975, Unpublished.

[Fou2001] R. Fourer, *Nonlinear Programming Frequently Asked Question*, www-unix.mcs.anl.gov/oto/Guide/faq/nonlinear-programming-faq.html, Optimization Technology Center Northwestern University, Retrieved November 2001.

[Fro2002] Frontline Systems Inc., *XPRESS Solver*, www.solver.com, Retrieved February 2002.

[FW1956] M. Frank and P. Wolfe, *An Algorithm for Quadratic Programming*, Naval Research Logistics Quarterly, Princeton University, Vol.3, 1956, pp.95-110.

[GGM+1984] P. E. Gill, N. I. M. Gould, W. Murray, M. A. Saunders, and M. H. Wright, *A Weighted Gram-Schmidt Method for Convex Quadratic Programming*, Mathematical Programming, Vol.30, 1984, pp.176-196.

[Gor1999] I. F. Gorodnitsky, *An Extension of an Interior Point Mathod for Entropy Minimization*, Cognitive Science Department, UC of San Diego, Vol.3, 1999.

[Gou2000] N. Gould, *Quadratic Programming Theory and Methods*, Rutherford Appleton Laboratory , Retrieved November 2000.

[Gou2002] N. Gould, *Personal Communication*, Rutherford Appleton Laboratory, 2002.

[GW2001] M. Gertz and S. Wright, *Object-Orientated Software for Quadratic Programming*, Argonne National Laboratory, 2001.

[Had1962] G. Hadley, *Linear Programming*, Addison-Wesley Publishing Company Inc., 1962.

[Had1964] G. Hadley, *Nonlinear and Dynamic Programming*, Addison-Wesley Publishing Company Inc., 1964.

[HD1968] W. M. Hirsch and G. B. Dantzig, *The Fixed Charge Problem*, Naval Research Logistics Quarterly, Vol.15, 1968, pp.413-424.

[Her1994] D. den Hertog, *Interior Point Approach to Linear, Quadratic and Convex Programming. Algorithms and Complexity*, Kluwer Academic Publishers, 1994.

[Hou1960] H. S. Houthakker, *The Capacity Method of Quadratic Programming*, Econometrica, Vol.28, 1960, pp.62-87.

[JB2003] P. A. Jensen and J. F. Bard, *Interior Point Methods*, www.me.utexas.edu/~jensen/ORMM/supplements/methods/lpmethod/S4_interior.pdf, Retriever May 2003.

[Kar1984] N. K. Karmarkar, *A New Polynomial Time Algorithm for Linear Programming*, Combinatorica, Vol. 4, 1984, pp.373-395.

[Kel1973] E. L. Keller, *The General Quadratic Optimization Problem*, Mathematical Programming, California State University, 1973.

[KKO1966] H. P. Kunzi, W. Krelle, and W. Oettli, *Nonlinear Programming*, Blaisdell Publishing Company, 1966.

[KTZ1971] H. P. Kunzi, H. G. Tzschach, and C. A. Zehnder, *Numerical Methods of Mathematical Optimization*, Academic Press, 1971.

[Lan2002] LANCELOT, *CCLRC Daresbury Laboratory,* www.dl.ac.uk/home.html, Retrieved February 2002.

[Lue1973] D. G. Luenberg, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley Publishing Company Inc., 1973.

[Mar2003] A. Martin, *GOAL*, www.geocities.com/geneticoptimization/, Retrieved November 2003.

[Mat2004] The MathWorks, Inc., *LDL Factorization*, www.mathworks.com/access/helpdesk/help/toolbox/dspblks/ldlfactorization.shtml, Retrieved January 2004.

[Mil2000] R. E. Miller, *Optimization: Foundations and Applications*, John Wiley and Sons Inc., 2000.

[Sas1999] S. Sastry, *Nonlinear systems: Analysis, Stability and Control*, Springer, 1999.

[Sim1975] D. M. Simmons, *Nonlinear Programming for Operations Research*, Prentice Hall, Inc., 1975.

[Tah1997] H. A. Taha, *Operations Research An Introduction*, Prentice-Hall, Inc., 1997.

[TV1960] H. Theil and C. Van de Panne, *Quadratic Programming as an Extention of Conventional Quadratic Maximization*, Management Science, Vol.7, 1960, pp.1-20.

[UU2000] M. Ulbrich and S. Ulbrich, *Superlinear Convergence of Affine-Scaling Interior-Point Newton Methods for Infinite-Dimensional Nonlinear Problems with Pointwise Bounds*, SIAM Journal Control Optimization, Vol.38, 2000, pp.1938-1984.

[Van1994] R. J. Vanderbei, *LOQO: An Interior Point Code for Quadratic Programming*, Department of Operations Research, Princeton University, 1994.

[Van2001] R. J. Vanderbei, *Linear Programming: Foundations and Extentions*, Department of Operations Research, Princeton University, 2001.

[Van2002] R. J. Vanderbei, *LOQO*, www.orfe.princeton.edu/~loqo/, Retrieved February 2002.

[VB1994] L. Vandenberghe and S. Boyd, *Semidefinite Programming*, Information Systems Laboratory, Stanford University, 1994.

[VBW2002] L. Vandenberghe, S. Boyd, and S. Wu, *Determinant Maximization with Linear Matrix Inequality Constraints*, Information Systems Laboratory Stanford University, Retrieved November 2002.

[VS1991] R. J. Vanderbei and D. F. Shanno, *An Interior Point Algorithm for Nonconvex Nonlinear Programming*, Computational Optimization and Applications, Statistics and Operations Research, Princeton University, 1991.

[VW1964] C. Van De Panne and A. Whinston, *The Simplex and the Dual Method for Quadratic Programming*, Operations Research Quarterly, Vol.15, 1964, pp.355-388.

[Win1995] W. L. Winston, *Introduction to mathematical Programming*, Duxbury Press, 1995.

[Wol1959] P. Wolfe, *The Simplex Method for Quadratic Programming*, Econometrica, Vol.27, 1959, pp.382-398.

[Zan1969] W. I. Zangwill, *Nonlinear Programming: A Unified Approach*, Prentice-Hall Inc., 1969.

Appendix A

A.1 Taylor's Theorem

Taylor's Theorem for *f(x)*

Taylor's series for $f(x)$ says that an approximation to the value of the function in a small neighborhood of a particular point, x_0, can be found to any degree of accuracy, provided the value of $f(x)$ and its derivatives are known at x_0 [Mil2000]. Denote by x_0 a point near x then Taylor's Theorem then can be used to express $f(x)$ as

(A.1.1) $$f(x) = f(x_0) \quad f'(x_0)(x \quad x_0) \quad \frac{f''(x_0)}{2!}(x \quad x_0)^2 \qquad \frac{f^{(n)}(x_0)}{n!}(x \quad x_0)^n \quad R_n$$

where R_n is the remainder. As $x \to x_0$, R_n will become negligible as n becomes higher. Subtracting $f(x_0)$ from both sides, the second order approximation is generated by eliminating derivatives higher than two, giving

(A.1.2) $$f(x) \quad f(x_0) \quad f'(x_0)(x \quad x_0) \quad \frac{f''(x_0)}{2}(x \quad x_0)^2.$$

A local minimum point has been identified at p when $f(x) \quad f(p) \geq 0$. Let $p = x_0$. Using (A.1.2) get the following

(A.1.3) $$f'(p)(x \quad p) \quad \frac{f''(p)}{2}(x \quad p)^2 \geq 0.$$

Since there are x's on either side of p, the sign of $(x \quad p)$ is positive when $x > p$ and negative when $x < p$. Therefore the term $f'(p)(x \quad p)$ must drop out, and this is only possible if $f'(p) = 0$. The following term then remains

(A.1.4) $$f(x) \quad f(p) \quad \left(\frac{1}{2}\right) f''(p)(x \quad p)^2 \geq 0.$$

$\frac{1}{2}(x \quad p)^2$ is positive, so that $f(x) \quad f(p) \geq 0$ if and only if $f''(p) \geq 0$, which is the second order necessary condition. The same kind of argument can be used for maximization.

Taylor's Theorem for *f(x)*

Taylor's series for a function $f(\boldsymbol{x})$ of n variables is given as follows. The first order approximation is given as

(A.1.5) $$f(\boldsymbol{x}) \quad f(\boldsymbol{x}_0) = [\quad f(\boldsymbol{x}_0)]'(\boldsymbol{x} \quad \boldsymbol{x}_0).$$

Assume that the second partial derivatives of $f(\boldsymbol{x})$ exist and are continuous. Then the second order approximation to $f(\boldsymbol{x}) \quad f(\boldsymbol{x}_0)$ results from eliminating all differentials above the second

(A.1.6) $$f(\boldsymbol{x}) \quad f(\boldsymbol{x}_0) \quad [\quad f(\boldsymbol{x}_0)]'(\boldsymbol{x} \quad \boldsymbol{x}_0) \quad \frac{1}{2}(\boldsymbol{x} \quad \boldsymbol{x}_0)'\boldsymbol{H}(\boldsymbol{x}_0)(\boldsymbol{x} \quad \boldsymbol{x}_0).$$

A local minimum of $f(\boldsymbol{x})$ at $\boldsymbol{p}$ is defined by $f(\boldsymbol{x}) \geq f(\boldsymbol{p})$ for all $\boldsymbol{x}$ in some neighborhood of $\boldsymbol{p}$. In terms of the second order approximation in (A.1.6) the minimum condition can be expressed as

(A.1.7) $$[\quad f(\boldsymbol{p})]'(\boldsymbol{x} \quad \boldsymbol{p}) \quad \frac{1}{2}(\boldsymbol{x} \quad \boldsymbol{p})'\boldsymbol{H}(\boldsymbol{p})(\boldsymbol{x} \quad \boldsymbol{p}) \geq 0.$$

with $\boldsymbol{p} = \boldsymbol{x}_0$. Now $\boldsymbol{x} \quad \boldsymbol{p}$ can be either positive or negative, therefore to achieve non-negativity $f(\boldsymbol{p})$ must be zero. This means then in (A.1.7)

(A.1.8) $$\frac{1}{2}(\boldsymbol{x} \quad \boldsymbol{p})'\boldsymbol{H}(\boldsymbol{p})(\boldsymbol{x} \quad \boldsymbol{p}) \geq 0.$$

The left hand side is the quadratic form in $(\boldsymbol{x} \quad \boldsymbol{p})$ and so to satisfy the inequality $\boldsymbol{H}(\boldsymbol{p})$ must be positive semi-definite.

A.2 Newton's Method

This is an iterative approach for finding a solution for a non-linear system of equations where each iteration consists of a solution of a linear system. at each iteration a new estimate of the solution is obtained by improving the previous estimate. Consider first the one dimensional case, that is, a function of one variable.

For a given function $f(x)$ we want to find x such that $f(x) = 0$. It is not possible to find a step direction of the optimal solution for a non-linear system, therefore it is approximated by the first two terms of the Taylor's series expansion (see Appendix A.1). The Taylor expansion of $f(x)$ around some point x_0 is given by

$$f(x) \quad f(x_0) \quad f'(x_0)(x \quad x_0) \quad \frac{f''(x_0)}{2}(x \quad x_0)^2.$$

To get the minimum of the function we take the derivative and set it to zero, which gives a linear system of equations which can be solved to give the step direction

$$0 = f'(x_0) \quad f''(x_0)(x \quad x_0)$$
$$x = x_0 \quad \frac{f'(x_0)}{f''(x_0)}.$$

Suppose that a sequence x_k of estimates can be generated with the initial approximation x_0, that is

(A.2.1) $$x_k = x_{k-1} - \frac{f'(x_{k-1})}{f''(x_{k-1})}.$$

A new point x_k is determined from x_{k-1} until $f(x_k) \approx 0$.

Extending this idea in multiple dimensions. Consider the function $f(\boldsymbol{x})$, then by differentiating (A.1.6) we get

$$\boldsymbol{x} = \boldsymbol{x}_0 - (\boldsymbol{H}(\boldsymbol{x}_0))^{-1} f(\boldsymbol{x}_0)'$$

Generating the sequence in multiple dimensions

(A.2.2) $$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - (\boldsymbol{H}(\boldsymbol{x}_{k-1}))^{-1} f(\boldsymbol{x}_{k-1})'.$$

The current solution $\boldsymbol{x}$ is updated until the current solution $f(\boldsymbol{x}) \approx 0$.

Appendix B

B.1 Simplex Algorithm

The simplex algorithm is a pivoting algorithm for solving linear programming problems of certain type. This procedure was invented by George Dantzig in 1947 and has since become extremely useful [Bri1989] and has been adapted to solve certain class of quadratic programming problems. It is therefore introduced here for reference purposes [Had1964].

Before discussing the simplex method some concepts are first introduced that will serve as a basis for understanding of the simplex method. Consider the linear programming problem.

$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad\quad \boldsymbol{x} \ge 0 \end{aligned} \tag{B.1.1}$$

In order to obtain the solution, the problem has to be written in *standard form*. A problem is in standard form when all the constraints are equations except for the non-negativity constraints, the right hand side element of each constraint equation is non-negative, all the variables are non-negative and the objective function is either minimization or maximization.

At this stage the system (B.1.1) is not in standard form. It is possible that $b_i \le 0$ for some i, say $i = q \quad 1, \ldots, m$. The right hand side of these inequalities can then be made positive by multiplying those inequalities by -1. The obtain the system in standard form, the inequalities are then changed to equations by adding the appropriate slack or surplus variables as follows: the inequality

$$a_{ij} x_j \le b_i \quad i = 1, \ldots, q, \quad j = 1, \ldots, n \tag{B.1.2}$$

is changed to standard form by adding slack variables $y_i, i = 1, \ldots, q$ such that

$$a_{ij} x_j \quad y_i = b_i \quad i = 1, \ldots, q, \quad j = 1, \ldots, n, \tag{B.1.3}$$

and the inequality

$$a_{ij} x_j \ge b_i \quad i = q \quad 1, \ldots, m, \quad j = 1, \ldots, n \tag{B.1.4}$$

is changed by adding surplus variables $y_i, i = q \quad 1, \ldots, m$ to obtain

$$a_{ij} x_j \quad y_i = b_i \quad i = q \quad 1, \ldots, m, \quad j = 1, \ldots, n. \tag{B.1.5}$$

The standard form of the problem (B.1.1) is

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \\ &\textit{Subject to} && a_{ij}x_j \quad y_i = b_i \quad i = 1,\ldots,q, \quad j = 1,\ldots,n, \\ & && a_{ij}x_j \quad y_i = b_i \quad i = q \quad 1,\ldots,m, \quad j = 1,\ldots,n. \\ & && x_j \geq 0 \quad j = 1,\ldots,n. \end{aligned}$$

Rewriting the standard form in matrix notation

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \\ &\textit{Subject to} && \hat{\boldsymbol{A}}\boldsymbol{x} \quad \boldsymbol{D}\boldsymbol{y} = \hat{\boldsymbol{b}} \\ & && \boldsymbol{x} \geq 0 \end{aligned} \tag{B.1.6}$$

In problem (B.1.6) the matrix $\hat{\boldsymbol{A}}$ contains the same values as $\boldsymbol{A}$ for the first q rows and negative values of $\boldsymbol{A}$ for the rows q 1 to m. The matrix $\boldsymbol{D}$ is a diagonal matrix with q entries +1 and q 1 to m entries -1. The column vector $\hat{\boldsymbol{b}}$ has all entries positive. It will therefore be assumed from this point onward that $\hat{b}_i \geq 0$ for all i, for if $\hat{b}_i \leq 0$ for some i then the expression can always be written such that the right hand side is positive.

Once the problem has been written in standard form the simplex procedure now consists of two parts. In the first part an initial extreme point is found. In the second part, a finite sequence of extreme points is generated such that the objective function decreases or at least remains constant with each new point generated.

The initial feasible solution can be obtained in a number of ways. If the matrix $\boldsymbol{D}$ is the identity matrix then the initial basic feasible solution can be obtained by setting $\boldsymbol{D}\boldsymbol{y} = \boldsymbol{x}_B = \hat{\boldsymbol{b}}$. Alternatively if m linearly independent column vectors of $\hat{\boldsymbol{A}}$ can be identified, then these can be used as the initial basic feasible solution. Let for example, $\boldsymbol{B}$ be a matrix whose columns are any m linearly independent columns from $\hat{\boldsymbol{A}}$. The initial basic solution is then given by $\boldsymbol{x}_B = \boldsymbol{B}^{1}\hat{\boldsymbol{b}}$. However, searching the matrix $\hat{\boldsymbol{A}}$ for linearly independent vectors is a time consuming process. One of the simplest ways of obtaining an initial feasible solution that is commonly used in practice, is to introduce m variables $\boldsymbol{s} = (s_1,\ldots,s_m)$ to all the constraints with the coefficient of 1. At this point the initial feasible solution is such that $\boldsymbol{s} = \hat{\boldsymbol{b}} \geq 0$. These variables have no relevance or meaning to the original problem and are merely added so that we can have a feasible solution to start the simplex method. Therefore these variables are called *artificial variables*. The artificial variables are then added to the constraints as follows:

$$\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x}$$

$$\text{Subject to} \quad \hat{\boldsymbol{A}}\boldsymbol{x} \quad \boldsymbol{D}\boldsymbol{y} \quad \boldsymbol{s} = \hat{\boldsymbol{b}}$$
$$\boldsymbol{x}, \boldsymbol{s} \geq 0$$

Although the artificial variables do not have to be added to the constraints for which the $\hat{b}_i$ s in (B.1.1) are positive, these problems are usually solved using computers, therefore the artificial variables are added to all the constraints for simplicity. However, if the artificial variables have been introduced, the problem is no longer the same as the original problem and therefore the artificial variables have to be removed. This can be done by the Two-Phase simplex method which will be introduced in Appendix B.2.

Consider now the problem for which there is no need to introduce artificial variables, that is $\hat{\boldsymbol{b}} \geq 0$. The initial basic solution is then given by $\boldsymbol{x}_B = \boldsymbol{B}^{-1}\hat{\boldsymbol{b}}$. The second part of the simplex method can now start. In order to begin the problem is arranged in a so called simplex tableau, which is constructed as follows.

			$\hat{c}_1$	$\hat{c}_2$		$\hat{c}_j$		$\hat{c}_n$
$\boldsymbol{c}_B$	Basic Variable	$\boldsymbol{x}_B$	$\boldsymbol{a}_1$	$\boldsymbol{a}_2$	…	$\boldsymbol{a}_j$	…	$\boldsymbol{a}_n$
c_{B1}	$\hat{b}_1$	$x_{B1} = y_{10}$	y_{11}	y_{12}		y_{1j}		y_{1n}
c_{Bm}	$\hat{b}_m$	$x_{Bm} = y_{m0}$	y_{m1}	y_{m2}		y_{mj}		y_{mn}
		z	$z_1 \quad \hat{c}_1$	$z_2 \quad \hat{c}_2$		$z_j \quad \hat{c}_j$		$z_n \quad \hat{c}_n$

The first column $\boldsymbol{c}_B$ gives the coefficients of the variables in the basis. The second column gives the variables that are in the basis. The third column $\boldsymbol{x}_B$ gives the current value of the variables in the basis. The rest of the columns $\boldsymbol{a}_j$ list the values for all variables $\hat{\boldsymbol{A}}$ as well as any artificial variables that have been added. The last row of column $\boldsymbol{x}_B$ is $z = \sum_{i=1}^{m} (c_{Bi} \quad x_{Bi})$ and the entries $z_j \quad \hat{c}_j = \sum_{i=1}^{m} (c_{Bi} \quad y_{ij}) \quad \hat{c}_j, j = 1, \ldots, n$.

Once the initial tableau has been constructed the procedure continues as follows.

Step 1	If all $z_j \quad \hat{c}_j \leq 0$ then the current basic feasible solution is optimal since any increase of an independent variable could not decrease the value of the objective function because of the non-negative $z_j \quad \hat{c}_j$ [Mil2000] [KKO1966]. If there are any $z_j \quad \hat{c}_j > 0$ in the simplex tableau then proceed to Step 2.

Step 2	Consider now that there is at least one $z_j \quad \hat{c}_j > 0$, then the solution is not minimal. Determine if every $z_j \quad \hat{c}_j$ has a positive number in its column. If $y_{ij} \leq 0$ for all i and j, then the solution is unbounded. This means that the problem has no finite minimum value of $f(\boldsymbol{x})$, that is, $f(\boldsymbol{x})$ can be arbitrarily decreased without violating the constraint $\boldsymbol{x} \geq 0$ and the computational procedure is complete.
Step 3	Consider that there is at least one $y_{ij} > 0$. Select the column with the largest positive $z_j \quad \hat{c}_j$, that is, the column with the largest positive value as this will produce the largest decrease. Call this column p. That is, select $z_p \quad \hat{c}_p = max(z_j \quad \hat{c}_j), \quad i \quad y_{ip} > 0$. If there should be a tie, any one of the columns can be selected, though it is a common practice to select the one with the lower subscript [Had1962]. The variable in column $\boldsymbol{a}_p$ is then selected to enter the basis.
Step 4	The next step is to determine which variable is to be removed from the basis. In the column $\boldsymbol{a}_p$ determine r, such that (B.1.7) $\frac{x_{Br}}{y_{rp}} = \min_i \left\{ \frac{x_{Bi}}{y_{ip}}, y_{ip} > 0 \right\}$, that is, r denotes the row of the smallest ratio. This ration is called θ-ratio [Bri1989]. If there is more than one θ-ratio the same then any one can be selected (but as mentioned in Step 3, the one with the smallest subscript is usually selected). Column r of the basis is then removed. This means that the variable in the rth row of the column "basic variable" is removed and replaced by $\boldsymbol{a}_p$. The entry y_{rp} is referred to as a *pivot*.

Step 5	The next step is to compute the new solution and to construct a new tableau. The transformation is done as follows. Pivot the tableau at the entry y_{rp} selected in the previous step. Thus column p replaces column r in the basis and the following transformation is performed: (B.1.8) $\hat{y}_{ij} = y_{ij} \quad y_{ip}\frac{y_{rj}}{y_{rp}} \quad j=0,\ldots,n, \quad i=1,\ldots,m \quad 1, \quad i \neq r$ (B.1.9) $\hat{y}_{rj} = \frac{y_{rj}}{y_{rp}}$ Every row in the tableau can be transformed by using (B.1.8) except for row $r,$ where the subscript i refers to a row in the tableau. Row r is transformed by using equation (B.1.9).
Step 6	The final step in the computation of the new tableau consists of changing the $\boldsymbol{c}_B$ and "basic variable" column. Only the r th entry of each of these columns is changed. The number c_{Br} is replaced by c_p and the variable in the r th position under "basic variable" is replaced by $\boldsymbol{a}_p$.

Once the new tableau is obtained, a check is done to see if the solution is optimal. If it is not, the procedure in the above table is repeated again until the optimal solution is found or the solution is found to be unbounded. Once the final tableau is obtained it contains all the necessary information about the optimal solution. Under the column "basic variable" are the variables in the optimal feasible solution. Under the column with the heading $\boldsymbol{x}_B$ are the values of the variables in the optimal feasible solution. The last entry of column $\boldsymbol{x}_B$ labeled z, is the value of the objective function of the problem (B.1.1).

B.2 Two Phase Simplex Method

As has been shown in Appendix B.1 it is frequently necessary to add artificial variables $\boldsymbol{s} = (s_1,\ldots,s_m)$ to easily find the initial basic feasible solution. In order to solve the original problem all the artificial variables must be removed from the basis. This can be achieved by assigning a large positive value (*M*) to each artificial variable, thus making it unprofitable to have such variables in the basis. This procedure is called the big M method however will not be discussed in any further detail.

Another method of treating artificial variables has been developed by Dantzig, Orden and others at the RAND Corporation. They developed a method called the Two-Phase method [Had1962].

The general idea of the two phase method is to in Phase One use the simplex method to remove all the artificial variables from the basis. Phase Two of the method is to minimize the objective function of the original problem starting from a basic feasible solution which either contains no artificial variables or some artificial variables at a zero level.

In Phase One the coefficient value c_B assigned to the artificial variables is 1 and for all the remaining variables 0. In Phase One all the artificial variables $\boldsymbol{s}$ must be eliminated from the basis, that is, $\sum s = 0$. This is done with the usual simplex method described in Appendix B.1 above. At the end of Phase One three cases are possible.

Case 1	If $\sum s > 0$ then the original linear programming problem has no feasible solution as one of the artificial variables appears in the basis at a positive level.
Case 2	If $\sum s = 0$ and there are no artificial variables in the optimal Phase One basis then all the columns corresponding to the artificial variables can be removed from the tableau. The original objective function is now combined with the constraints in the Phase One tableau which yields Phase Two.
Case 3	If $\sum s = 0$ and there is at least one artificial variable in the Phase One basis the optimal solution to the original problem can be found if at the end of Phase One all non-basic artificial variables are removed from the tableau and any variable from the original problem that has a negative coefficient in Row 0 is also removed.

When Phase One results in Case 2 or 3 we go to Phase Two. In Phase Two the actual values $\hat{c}_j$ are assigned to each legitimate variable and 0 to each artificial variable that may appear in the basis at zero level. The function to be optimized in Phase Two is the actual objective function $f(\boldsymbol{x})$. The first tableau of Phase Two is the last tableau of Phase One with the only difference that the last row $z_j \quad \hat{c}_j$ must be altered to take account of the change of the values $\hat{c}_j$.

If Phase One ends in Case 2 then the usual procedure follows to find the variable to enter and to leave the basis. If Phase One ends in Case 3 then if $y_{ij} = 0$ for all j and for

i corresponding to the column containing the artificial variable, the row can be deleted from the tableau. The procedure from Appendix B.1 is then followed to determine which variables leave the basis.

If $y_{ip} > 0$ for at least one i then instead of removing some legitimate variable, one of the artificial variables is removed. This will ensure that the artificial variables will never become negative. If Phase Two requires no pivots then the optimal solution has been found.

B.3 Perturbation Technique

Consider the maximization problem

(B.3.1)
$$\begin{aligned} &\textit{Minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \tfrac{1}{2}\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad \boldsymbol{x} \ge 0. \end{aligned}$$

The matrix $\boldsymbol{Q}$ may be semi-definite. Assume that this problem has a solution. If the solution is not unique, consider the solution with the smallest norm (length). This solution will be unique because the solution space is convex. Let $\hat{\boldsymbol{x}}$ be a vector of minimal length solving (B.3.1). Thus

$$f(\hat{\boldsymbol{x}}) \ge f(\boldsymbol{x}) \text{ for all } \{\, \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \,\}$$

and if $\boldsymbol{y}$ is a vector such that $f(\boldsymbol{y}) = f(\hat{\boldsymbol{x}})$ and $\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$,

$$\hat{\boldsymbol{x}}'\hat{\boldsymbol{x}} < \boldsymbol{y}'\boldsymbol{y}.$$

Consider the perturbation problem of maximizing

(B.3.2)
$$\begin{aligned} &f_\varepsilon(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \tfrac{1}{2}\boldsymbol{x}'(\boldsymbol{Q} \quad \varepsilon \boldsymbol{I})\boldsymbol{x} \\ &\textit{Subject to} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ &\qquad\qquad \boldsymbol{x} \ge 0 \end{aligned}$$

where $\varepsilon > 0, \varepsilon \to 0$. Whenever the solution to (B.3.2) exists, it will be unique. Denote $\hat{\boldsymbol{x}}_\varepsilon$ for this solution. The aim is to prove that

(B.3.3)
$$\lim_{\varepsilon \to 0} \hat{\boldsymbol{x}}_\varepsilon = \hat{\boldsymbol{x}}.$$

That is, the solution of the slightly perturbed definite problem is arbitrarily close to the solution with the smallest norm of the semi-definite problem. First it must be shown that

(B.3.4)
$$\lim_{\varepsilon \to 0} f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) = f(\hat{\boldsymbol{x}})$$

that is, the maximum value of the objective function of the two problems is, in the limit, the same. First observe that for all vectors $\boldsymbol{x}$

(B.3.5) $$f_\varepsilon(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \tfrac{1}{2}\varepsilon\,\boldsymbol{x}'\boldsymbol{x},$$

as follows from (B.3.2). Hence

$$f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) = f(\hat{\boldsymbol{x}}_\varepsilon) \quad \tfrac{1}{2}\varepsilon\,\hat{\boldsymbol{x}}_\varepsilon'\boldsymbol{x}_\varepsilon \le f(\hat{\boldsymbol{x}}) \quad \tfrac{1}{2}\varepsilon\,\hat{\boldsymbol{x}}_\varepsilon'\boldsymbol{x}_\varepsilon \le f(\hat{\boldsymbol{x}})$$

and

$$f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) \ge f_\varepsilon(\hat{\boldsymbol{x}}) = f(\hat{\boldsymbol{x}}) \quad \tfrac{1}{2}\varepsilon\,\hat{\boldsymbol{x}}'\boldsymbol{x}.$$

The desired conclusion (B.3.4) follows by letting ε approach zero. Next it will be shown that the points $\hat{\boldsymbol{x}}_\varepsilon$ form a bounded set for $\varepsilon > 0$, by proving that $(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}})'(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}}) \le 2\,\hat{\boldsymbol{x}}'\hat{\boldsymbol{x}}$. Let $\hat{\boldsymbol{x}} \quad F$ and $\bar{\boldsymbol{x}} \quad F$, where $F = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}\}$. Then for any $0 \le \lambda \le 1$

$$\boldsymbol{x}_\lambda = (1 \quad \lambda)\,\hat{\boldsymbol{x}} \quad \lambda\,\bar{\boldsymbol{x}} \quad F$$

for $\boldsymbol{A}\boldsymbol{x}_\lambda = (1 \quad \lambda)\,\boldsymbol{A}\hat{\boldsymbol{x}} \quad \lambda\,\boldsymbol{A}\bar{\boldsymbol{x}} \le (1 \quad \lambda)\,\boldsymbol{b} \quad \lambda\,\boldsymbol{b} = \boldsymbol{b}$. Furthermore, for all $\boldsymbol{x}_\lambda$

$$f(\boldsymbol{x}_\lambda) = (1 \quad \lambda)\,f(\hat{\boldsymbol{x}}) \quad \lambda\,f(\bar{\boldsymbol{x}}) \quad \tfrac{1}{2}(\lambda \quad \lambda^2)(\hat{\boldsymbol{x}} \quad \bar{\boldsymbol{x}})'\boldsymbol{Q}(\hat{\boldsymbol{x}} \quad \bar{\boldsymbol{x}}).$$

Using this result with $\lambda = \frac{1}{2}$ gives

$$f_\varepsilon(\tfrac{1}{2}\hat{\boldsymbol{x}}_\varepsilon \quad \tfrac{1}{2}\hat{\boldsymbol{x}}) = \tfrac{1}{2}f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) \quad \tfrac{1}{2}f_\varepsilon(\hat{\boldsymbol{x}}) \quad \tfrac{1}{8}(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}})'(\boldsymbol{Q} \quad \varepsilon\boldsymbol{I})(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}}) \le f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon).$$

Using the result and (B.3.5) yields

$$\begin{aligned}\tfrac{1}{8}(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}})'(\boldsymbol{Q} \quad \varepsilon\boldsymbol{I})(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}}) &\le f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) \quad \tfrac{1}{2}f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) \quad \tfrac{1}{2}f(\hat{\boldsymbol{x}}) \quad \tfrac{1}{4}\varepsilon\,\hat{\boldsymbol{x}}'\hat{\boldsymbol{x}} \\ &= \tfrac{1}{2}[f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) \quad f(\hat{\boldsymbol{x}})] \quad \tfrac{1}{4}\varepsilon\,\hat{\boldsymbol{x}}'\hat{\boldsymbol{x}} \le \tfrac{1}{4}\varepsilon\,\hat{\boldsymbol{x}}'\hat{\boldsymbol{x}}.\end{aligned}$$

Because $(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}})'\boldsymbol{Q}(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}}) \ge 0$, we have

$$\tfrac{1}{8}\varepsilon(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}})'(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}}) \le \tfrac{1}{4}\varepsilon\,\hat{\boldsymbol{x}}'\hat{\boldsymbol{x}}$$

or

$$(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}})'(\hat{\boldsymbol{x}}_\varepsilon \quad \hat{\boldsymbol{x}}) \le 2\,\hat{\boldsymbol{x}}'\hat{\boldsymbol{x}}.$$

To conclude the argument (B.3.3) will be proved. Suppose that (B.3.3) does not hold. Then there is a convergent sequence $\hat{\boldsymbol{x}}_{\varepsilon_1}, \hat{\boldsymbol{x}}_{\varepsilon_2}, \ldots$, with $\varepsilon_1 > \varepsilon_2 \ldots \to 0$ and such that

$$\lim_{n \to \infty} \hat{\boldsymbol{x}}_{\varepsilon_n} = \boldsymbol{y} \ne \hat{\boldsymbol{x}}.$$

Since $f_\varepsilon(\boldsymbol{x})$ is a continuous function in $\boldsymbol{x}$ and ε it follows

$$\lim_{n \to \infty} f_{\varepsilon_n}(\hat{\boldsymbol{x}}_{\varepsilon_n}) = \lim_{\varepsilon \to 0} f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) = f(\boldsymbol{y}).$$

From (B.3.4) it is known that

$$\lim_{\varepsilon \to 0} f_\varepsilon(\hat{\boldsymbol{x}}_\varepsilon) = f(\hat{\boldsymbol{x}})$$

hence $f(\hat{\boldsymbol{x}}) = f(\boldsymbol{y})$. Because $\boldsymbol{y}$ satisfies $\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$ it can be concluded that $\boldsymbol{y} \ne \hat{\boldsymbol{x}}$ is a solution to (B.3.1). By our choice of $\hat{\boldsymbol{x}}$ we have $\boldsymbol{y}'\boldsymbol{y} > \hat{\boldsymbol{x}}'\hat{\boldsymbol{x}}$. A number N can then be

picked so large that for all $n > N$ we have $\hat{\boldsymbol{x}}_{\varepsilon_n}' \hat{\boldsymbol{x}}_{\varepsilon_n} > \hat{\boldsymbol{x}}' \hat{\boldsymbol{x}}$. This leads to the following contradiction

$$f_{\varepsilon_n}(\hat{\boldsymbol{x}}_{\varepsilon_n}) = f(\hat{\boldsymbol{x}}_{\varepsilon_n}) \quad \frac{1}{2}\varepsilon_n \hat{\boldsymbol{x}}_{\varepsilon_n}' \hat{\boldsymbol{x}}_{\varepsilon_n} < f(\hat{\boldsymbol{x}}) \quad \frac{1}{2}\varepsilon_n \hat{\boldsymbol{x}}' \hat{\boldsymbol{x}} = f_{\varepsilon_n}(\hat{\boldsymbol{x}}) \le f_{\varepsilon_n}(\hat{\boldsymbol{x}}_{\varepsilon_n}).$$

Thus the result (B.3.3) has been established.

B.4 Dantzig's Algorithm – Example 2

In the following example, the matrix $\boldsymbol{Q}$ is positive semi-definite.

Minimize $f(\boldsymbol{x}) = 3x_1 \quad 4x_2 \quad x_3 \quad \frac{1}{2}(x_1 \quad 2x_2 \quad x_3)^2$

Subject to $x_1 \quad 2x_2 \quad x_3 \le 4$

$x_1, x_2, x_3 \ge 0$

The initial tableau is given as:

Basic Variable	$\boldsymbol{x_B}$	ξ_1	ξ_2	ξ_3	ξ_4	ω_1	ω_2	ω_3	ω_4
ω_1	-3	-1	-2	1	0	1	0	0	-1
ω_2	-4	-2	-4	2	0	0	1	0	-2
ω_3	-1	1	2	-1	0	0	0	1	-1
ξ_4	4	1	2	1	1	0	0	0	0

This tableau is a standard tableau. Since ω_2 is the most negative, ξ_2 enters the basis and ω_2 leaves the basis since $\frac{4}{4} < \frac{3}{2}$ and $\frac{4}{4} < \frac{4}{2}$.

Basic Variable	$\boldsymbol{x_B}$	ξ_1	ξ_2	ξ_3	ξ_4	ω_1	ω_2	ω_3	ω_4
ω_1	-1	0	0	0	0	1	-1/2	0	0
ξ_2	1	1/2	1	-1/2	0	0	-1/4	0	1/2
ω_3	-3	0	0	0	0	0	1/2	1	-2
ξ_4	2	0	0	2	1	0	1/2	0	-1

This tableau is again in standard form. Here ξ_3 enters the basis and ξ_4 leaves the basis.

Basic Variable	$\boldsymbol{x_B}$	ξ_1	ξ_2	ξ_3	ξ_4	ω_1	ω_2	ω_3	ω_4
ω_1	-1	0	0	0	0	1	-1/2	0	1
ξ_2	1 1/2	1/2	1	0	1/4	0	-1/8	0	1/4
ω_3	-3	0	0	0	0	0	1/2	1	-2
ξ_3	1	0	0	1	1/2	0	1/4	0	-1/2

At this point the tableau is non-standard since ξ_4 and ω_4 are both non-basic. In this case ω_4 must enter the basis. The ratio is computed and clearly ω_3 leaves the basis to give the following tableau.

Basic Variable	x_B	ξ_1	ξ_2	ξ_3	ξ_4	ω_1	ω_2	ω_3	ω_4
ω_1	-1	0	0	0	0	1	-1/2	0	0
ξ_2	1 1/8	1/2	1	0	1/4	0	-1/16	1/8	0
ω_4	1 1/2	0	0	0	0	0	-1/4	-1/2	1
ξ_3	1 3/4	0	0	1	1/2	0	1/8	-1/4	0

The tableau is standard again. ξ_1 enters the basis and ξ_2 leaves the basis.

Basic Variable	x_B	ξ_1	ξ_2	ξ_3	ξ_4	ω_1	ω_2	ω_3	ω_4
ω_1	-1	0	0	0	0	1	-1/2	0	0
ξ_1	2 1/4	1	2	0	1/2	0	-1/8	1/4	0
ω_4	1 1/2	0	0	0	0	0	-1/4	-1/2	1
ξ_3	1 3/4	0	0	1	1/2	0	1/8	-1/4	0

Here again the tableau is non-standard since both ξ_2 and ω_2 are non-basic. ω_2 must enter the basis and ω_1 leaves the basis.

Basic Variable	x_B	ξ_1	ξ_2	ξ_3	ξ_4	ω_1	ω_2	ω_3	ω_4
ω_2	2	0	0	0	0	-2	1	0	0
ξ_1	2 1/2	1	2	0	1/2	-1/4	0	1/4	0
ω_4	2	0	0	0	0	-1/2	0	-1/2	1
ξ_3	1 1/2	0	0	1	1/2	1/4	0	-1/4	0

Since all the x_B are positive the final solution has been reached at $\xi_1 = x_1 = 2\frac{1}{2}$, $\xi_2 = x_2 = 0$ and $\xi_3 = x_3 = 1\frac{1}{2}$. The final value of $f(x)$ decreases from each tableau as follows: 0; -2; -5; $6\frac{1}{2}$; -8; $8\frac{1}{2}$.

B.5 Basic Theorem

Suppose that r is added to the set of binding constraints (S becomes $S \quad r$) then λ^S as given in (3.2.4) becomes

(B.5.1) $$\lambda^{S \quad r} = P_{S}^{\ 1}{}_{r}(\hat{A}_{S \quad r}x \quad \hat{b}_{S \quad r})$$

where

(B.5.2) $$\boldsymbol{P}_{S\ r} = \hat{\boldsymbol{A}}_{S\ r}\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_{S\ r} = \begin{bmatrix}\hat{\boldsymbol{A}}_S \\ \hat{\boldsymbol{A}}_r\end{bmatrix}\boldsymbol{Q}^{-1}\begin{bmatrix}\hat{\boldsymbol{A}}'_S & \hat{\boldsymbol{A}}'_r\end{bmatrix} = \begin{bmatrix}\boldsymbol{P}_S & \boldsymbol{p} \\ \boldsymbol{p}' & \gamma\end{bmatrix}$$

so that $\boldsymbol{P}_S = \hat{\boldsymbol{A}}_S\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_S$, $\boldsymbol{p}' = \hat{\boldsymbol{A}}_r\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_S$ and $\gamma = \hat{\boldsymbol{A}}_r\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_r$. To compute $\boldsymbol{\lambda}^{S\ r}$ we must first compute $\boldsymbol{P}_S^{-1}{}_r$, hence

(B.5.3) $$\boldsymbol{P}_S^{-1}{}_r = \begin{bmatrix}\boldsymbol{P}_S^{-1} + \dfrac{\boldsymbol{P}_S^{-1}\boldsymbol{p}\boldsymbol{p}'\boldsymbol{P}_S^{-1}}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}} & \dfrac{\boldsymbol{P}_S^{-1}\boldsymbol{p}}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}} \\ \dfrac{\boldsymbol{p}'\boldsymbol{P}_S^{-1}}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}} & \dfrac{1}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}}\end{bmatrix}.$$

As we are interested in the signs of $\boldsymbol{\lambda}^{S\ r}$, the Lagrangians associated with the newly created constraint which is the last element of (B.5.1). Therefore using (B.5.3)

(B.5.4) $$\boldsymbol{\lambda}_r^{S\ r} = \begin{bmatrix}\dfrac{\boldsymbol{p}'\boldsymbol{P}_S^{-1}}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}} & \dfrac{1}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{A}}_S\boldsymbol{x} & \hat{\boldsymbol{b}}_S \\ \hat{\boldsymbol{A}}_r\boldsymbol{x} & \hat{\boldsymbol{b}}_r\end{bmatrix}.$$

Now using (3.2.4) the expression can be worked out

(B.5.5) $$\begin{aligned}\boldsymbol{\lambda}_r^{S\ r} &= \frac{1}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}}[\boldsymbol{p}'\boldsymbol{\lambda}^S - (\hat{\boldsymbol{A}}_r\boldsymbol{x} - \hat{\boldsymbol{b}}_r)] \\ &= \frac{1}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}}(\hat{\boldsymbol{A}}_r\boldsymbol{x}^S - \hat{\boldsymbol{b}}_r)\end{aligned}$$

Clearly the scalar $\frac{1}{\gamma - \boldsymbol{p}'\boldsymbol{P}_S^{-1}\boldsymbol{p}}$ in (B.5.5)is positive as it is a diagonal element of the inverse of a positive definite matrix. From (B.5.5) the sign of $\boldsymbol{\lambda}_r^{S\ r}$ depends on the sign of $\hat{\boldsymbol{A}}_r\boldsymbol{x}^S - \hat{\boldsymbol{b}}_r$, such that whenever $\hat{\boldsymbol{A}}_r\boldsymbol{x}^S > \hat{\boldsymbol{b}}_r$ then $\boldsymbol{\lambda}_r^{S\ r} > 0$; whenever $\hat{\boldsymbol{A}}_r\boldsymbol{x}^S = \hat{\boldsymbol{b}}_r$ then $\boldsymbol{\lambda}_r^{S\ r} = 0$ and whenever $\hat{\boldsymbol{A}}_r\boldsymbol{x}^S < \hat{\boldsymbol{b}}_r$ then $\boldsymbol{\lambda}_r^{S\ r} < 0$ and vice versa. This proves statements (1), (2) and (3).

To prove (1'), (2') and (3') the argument can be reversed. Start with a set $S, t \quad S$ where t is the last element in S. Let the vector $\boldsymbol{\lambda}^S$ and $\boldsymbol{\lambda}_t^S$ be known, which means that $\boldsymbol{P}_S^{-1}$ is known as well. Deleting t from S we can compute $\boldsymbol{\lambda}^{S\ t}$ and $\boldsymbol{x}^{S\ t}$. This requires the knowledge of $\boldsymbol{P}_S^{-1}{}_t$ which can be worked out similarly as (B.5.3). Carrying out the computations the following result is obtained

(B.5.6) $$\boldsymbol{\lambda}_t^S = \delta(\hat{\boldsymbol{A}}_t\boldsymbol{x}^{S\ t} - \hat{\boldsymbol{b}}_t)$$

where δ is the last diagonal element of $\boldsymbol{P}_S^{-1}$. Because $\boldsymbol{P}_S^{-1}$ is positive definite δ is positive. We can therefore conclude that $\boldsymbol{\lambda}_t^S \{<,=,>\} 0$, which implies $\hat{\boldsymbol{A}}_t\boldsymbol{x}^{S\ t} \{<,=,>\} \hat{\boldsymbol{b}}_t$. This completes the proof.

B.6 Theil-Van de Panne - Example 2

This example is given here to show how the Theil-Van de Panne procedure works in more detail.

(B.6.1) *Minimize* $f(\boldsymbol{x}) = x_1 \;\; 2x_2 \;\; x_3 \;\; \frac{1}{2}(x_1^2 \;\; 2x_2^2 \;\; 4x_3^2 \;\; 4x_2x_3)$

Subject to

$$x_1 + x_2 + x_3 \leq 10 \quad (1)$$
$$2x_1 \;\; + 3x_3 \leq 11 \quad (2)$$
$$2x_1 + 2x_2 \;\; 5x_3 \leq \;\; 13 \quad (3)$$
$$x_1 \leq 0 \quad (4)$$
$$x_2 \leq 0 \quad (5)$$
$$x_3 \leq 0 \quad (6)$$

For the problem (B.6.1)

$$\boldsymbol{c} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 4 \end{bmatrix} \text{ and } \boldsymbol{Q}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

The first step is to calculate $\boldsymbol{x}$.

(B.6.2)
$$\boldsymbol{x} = \boldsymbol{Q}^{-1}\boldsymbol{c} = \begin{bmatrix} 1 \\ \frac{5}{2} \\ \frac{3}{2} \end{bmatrix}$$

Using the result of (B.6.2) the vector $\boldsymbol{x}$ violates constraints (3) and (4). Next calculate all one element sets $S = \{3\} \text{ and } \{4\}$, that is $\boldsymbol{x}^3$ and $\boldsymbol{x}^4$.

$$\boldsymbol{x}^3 = \boldsymbol{x} \quad \boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_3(\boldsymbol{A}_3\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_3)^{-1}(\hat{\boldsymbol{A}}_3\boldsymbol{x} \quad \hat{\boldsymbol{b}}_3)$$

$$= \begin{bmatrix} \frac{29}{21} \\ \frac{65}{21} \\ \frac{69}{21} \end{bmatrix}$$

with the associated Lagrangian

$$\boldsymbol{\lambda}_3^3 = \frac{25}{21} .$$

The vector $\boldsymbol{x}^3$ violates constraint (2).

$$\boldsymbol{x}^4 = \boldsymbol{x} \quad \boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_4(\hat{\boldsymbol{A}}_4\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_4)^{-1}(\hat{\boldsymbol{A}}_4\boldsymbol{x} \quad \hat{\boldsymbol{b}}_4)$$

$$= \begin{bmatrix} 0 \\ \frac{5}{2} \\ \frac{3}{2} \end{bmatrix}$$

with the associated Lagrangian

$\boldsymbol{\lambda}_4^4 = 1$.

The vector $\boldsymbol{x}^4$ violates constraint (3). Next evaluate all the two element sets $\boldsymbol{x}^{3,2}$ and $\boldsymbol{x}^{4,3}$.

$$\boldsymbol{x}^{3,2} = \boldsymbol{x} \quad \boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_{3,2}(\hat{\boldsymbol{A}}_{3,2}\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_{3,2})^{-1}(\hat{\boldsymbol{A}}_{3,2}\boldsymbol{x} \quad \hat{\boldsymbol{b}}_{3,2})$$

$$= \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

with the associated Lagrangians

$\boldsymbol{\lambda}_2^{3,2} = \frac{527}{525}$ and $\boldsymbol{\lambda}_3^{3,2} = \frac{62}{25}$.

The vector $\boldsymbol{x}^{3,2}$ is feasible since none of the constraints are violated. At this point the vector $\boldsymbol{x}^{3,2}$ is also the optimal solution since both the Lagrangians are positive. Also as stated previously the feasible vector $\boldsymbol{x}^S$ is optimal in and only if for all $h \quad S$ the vector $\boldsymbol{x}^{S \quad h}$ violates constraint h . Therefore in the above example $\boldsymbol{x}^{3,2}$ is the solution, because $\boldsymbol{x}^{3,2 \quad 3} = \boldsymbol{x}^2$ violates constraint (3) as well as $\boldsymbol{x}^{3,2 \quad 2} = \boldsymbol{x}^3$ violates constraint (2). There is now no need to go any further as the optimal solution has been found.

It is possible that a solution vector is found that is not optimal before the optimal vector is found. This is not the case here since the optimal solution has already been found, but in order to show this we must continue the procedure. Thus the next vector is calculated.

$$\boldsymbol{x}^{4,3} = \boldsymbol{x} \quad \boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_{4,3}(\hat{\boldsymbol{A}}_{4,3}\boldsymbol{Q}^{-1}\hat{\boldsymbol{A}}'_{4,3})^{-1}(\hat{\boldsymbol{A}}_{4,3}\boldsymbol{x} \quad \hat{\boldsymbol{b}}_{4,3})$$

$$= \begin{bmatrix} 0 \\ \frac{331}{100} \\ \frac{392}{100} \end{bmatrix}$$

with associated Lagrangians

$\boldsymbol{\lambda}_3^{4,3} = \frac{21}{20}$ and $\boldsymbol{\lambda}_4^{4,3} = \frac{87}{420}$.

The vector $\boldsymbol{x}^{4,3}$ violates constraint (2) therefore the vector $\boldsymbol{x}^{4,3,2}$ - a three element set – has to be evaluated.

$$\boldsymbol{x}^{4,3,2} = \boldsymbol{x} \quad \boldsymbol{Q}^{-1}\bar{\boldsymbol{A}}'_{4,3,2}(\bar{\boldsymbol{A}}_{4,3,2}\boldsymbol{Q}^{-1}\bar{\boldsymbol{A}}'_{4,3,2})^{-1}(\bar{\boldsymbol{A}}_{4,3,2}\boldsymbol{x} \quad \bar{\boldsymbol{b}}_{4,3,2})$$

$$= \begin{bmatrix} 0 \\ \frac{267}{100} \\ \frac{367}{100} \end{bmatrix}$$

with the associated Lagrangians

$\lambda_2^{4,3,2} = \frac{342}{25}$, $\lambda_3^{4,3,2} = \frac{3591}{1250}$ and $\lambda_4^{4,3,2} = \frac{18}{25}$.

As can be seen the vector $\boldsymbol{x}^{4,3,2}$ is also feasible but since has one of the associated Lagrangians negative it is not an optimal solution. It can also be shown that $\boldsymbol{x}^{4,3,2}$ is not a solution, because $\boldsymbol{x}^{4,3,2\ 4} = \boldsymbol{x}^{3,2}$ does not violate constraint (4).

Appendix C

C.1 Test Examples

All the problems below are convex quadratic programming problems and are written in the following form:

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{x} \quad \boldsymbol{x}'\boldsymbol{Q}\boldsymbol{x} \\ &\textit{Subject to} && \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & && \boldsymbol{x} \ge 0. \end{aligned}$$

Problem 1

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = 3x_1 \quad 2x_2 \quad 5x_3 \quad (6x_1^2 \quad 3x_2^2 \quad 3x_3^2 \quad 4x_1x_2 \quad x_2x_3) \\ &\textit{Subject to} && 5x_1 \quad 8x_2 \quad 4x_3 \le 26 \\ & && 2x_1 \quad 2x_2 \quad 3x_3 \le 14 \\ & && x_2 \quad 4x_3 \le 12 \\ & && x_1, x_2, x_3 \ge 0 \end{aligned}$$

Problem 2

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = 0.2x_1 \quad 1.4x_2 \quad 0.5x_3 \quad 0.8x_4 \quad (2x_1^2 \quad 0.5x_2^2 \quad x_3^2 \quad 1.2x_4^2 \\ & && \qquad x_1x_2 \quad 0.9x_1x_4 \quad 0.4x_2x_3 \quad 1.2x_3x_4) \\ &\textit{Subject to} && x_1 \quad 0.5x_2 \quad 0.2x_3 \quad 0.6x_4 \le 6 \\ & && 0.1x_1 \quad 0.25x_2 \quad 0.4x_3 \le 1.5 \\ & && 0.7x_1 \quad 0.55x_2 \quad 0.4x_4 \le 3 \\ & && x_1, x_2, x_3, x_4 \ge 0 \end{aligned}$$

Problem 3

$$\begin{aligned} &\textit{Minimize} && f(\boldsymbol{x}) = 20x_1 \quad 14x_2 \quad 0.5x_3 \quad 8x_4 \quad (8x_1^2 \quad 12x_2^2 \quad 6x_3^2 \quad 11x_4^2 \\ & && \qquad 3x_1x_2 \quad 6x_1x_3 \quad 2x_1x_4 \quad 4x_2x_3 \quad 4x_2x_4 \quad 1x_3x_4) \\ &\textit{Subject to} && x_1 \quad 2x_2 \quad x_3 \quad 2x_4 \le 12 \\ & && 5x_1 \quad 5x_2 \quad 2x_3 \quad 7x_4 \le 30 \\ & && x_1 \quad 6x_2 \quad 4x_4 \le 7 \\ & && 3x_1 \quad 7x_4 \le 13 \\ & && x_1, x_2, x_3, x_4 \ge 0 \end{aligned}$$

Problem 4

Minimize $f(\boldsymbol{x}) = 10x_1 + 7.5x_2 + 8.2x_3 + 4x_4 + 5x_5 + (4.4x_1^2 + 6x_2^2 + 6x_3^2 + 3x_4^2 + 5.5x_5^2 + 4.8x_1x_2 + 6x_1x_3 + 3.8x_1x_4 + 2x_1x_5 + 2.4x_2x_3 + 6.7x_2x_4 + 4x_2x_5 + x_3x_4 + 2.9x_4x_5)$

Subject to $x_1 + 0.2x_2 + 1.7x_3 + 0.4x_4 + 0.1x_5 \leq 8.8$

$1.6x_1 + 2x_4 + 1.4x_5 \leq 15$

$x_1, x_2, x_3, x_4, x_5 \geq 0$

Problem 5

Minimize $f(\boldsymbol{x}) = 2.25x_1 + 1.75x_2 + 1.25x_3 + 0.85x_4 + 2.45x_5 + (2.45x_1^2 + 1.25x_2^2 + 0.95x_3^2 + 1.15x_4^2 + 1.15x_5^2 + 2.75x_1x_3 + 3.05x_1x_5 + 0.05x_2x_4 + 2.25x_2x_5 + 1.65x_3x_4 + 0.15x_3x_5 + 1.95x_4x_5)$

Subject to $11x_1 + 12x_2 + x_3 + 8x_5 \leq 57$

$5x_2 + 7x_3 + 9x_4 + 5x_4 \leq 39$

$17x_1 + 21x_3 + 15x_5 \leq 184$

$13x_1 + 7x_4 + 20x_5 \leq 92$

$x_1, x_2, x_3, x_4, x_5 \geq 0$

Problem 6

Minimize $f(\boldsymbol{x}) = 0.125x_1 + 0.55x_2 + 2.224x_3 + 1.372x_4 + 0.933x_5 + 1.151x_6 + (1.1x_1^2 + 1.12x_2^2 + 0.5x_3^2 + 1.21x_4^2 + 1.62x_5^2 + 0.4x_6^2 + x_1x_2 + 8x_1x_3 + 2x_1x_5 + 0.23x_2x_3 + 3.47x_2x_5 + 0.7x_2x_6 + 4x_3x_4 + 9x_4x_5 + 6.4x_4x_6 + x_5x_6)$

Subject to $x_1 + 2x_2 + 2x_3 + 4x_4 \leq 31$

$2x_2 + 4x_3 + 6.5x_4 + 2x_5 + 3x_6 \leq 24$

$3x_1 + 0.5x_2 + 2x_3 + 2x_4 + 2.5x_5 + 4x_6 \leq 12$

$0.5x_1 + x_2 + 2x_3 + 4x_4 + 2x_5 + 2x_6 \leq 21$

$5x_1 + 2x_2 + x_5 + 6x_6 \leq 9$

$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$

Problem 7

Minimize $f(\boldsymbol{x}) = 6x_1 \; x_2 \; 2x_3 \; x_4 \; 4x_5 \; x_6 \; 3x_7 \; (2x_1^2 \; x_2^2 \; 2x_3^2 \; x_4^2 \; 4x_5^2 \; 3x_6^2 \; x_7^2 \; 2x_1x_3 \; x_2x_7 \; 3x_3x_5 \; 2x_5x_7 \; 2x_6x_7)$

Subject to
$$12x_1 \; 5x_2 \; 2x_3 \; 13x_4 \; 2x_6 \; 2x_7 \le 350$$
$$14x_2 \; x_3 \; 4x_4 \; 5x_5 \; 23x_7 \le 280$$
$$4x_1 \; 2x_3 \; x_4 \; 5x_5 \; 5x_6 \le 400$$
$$2x_2 \; 3x_4 \; 9x_5 \; 11x_6 \; 3x_7 \le 235$$
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \ge 0$$

Problem 8

Minimize $f(\boldsymbol{x}) = 8x_1 \; 22x_2 \; 13x_3 \; 18x_4 \; 28x_5 \; 14x_6 \; 7x_7 \; x_8 \; (7x_1^2 \; 6x_2^2 \; 3x_3^2 \; 5x_4^2 \; 8x_5^2 \; 4x_6^2 \; 4x_7^2 \; 7x_8^2 \; 12x_1x_2 \; 20x_1x_4 \; 16x_1x_5 \; 12x_1x_6 \; 14x_2x_3 \; 18x_2x_5 \; 25x_2x_6 \; 35x_2x_7 \; 32x_3x_4 \; 8x_3x_6 \; 16x_3x_8 \; 12x_4x_5 \; 12x_4x_6 \; 3x_5x_8 \; 8x_7x_8)$

Subject to
$$0.5x_1 \; 0.8x_2 \; 0.4x_4 \; 0.8x_5 \le 20$$
$$0.8x_1 \; 0.2x_2 \; 1.2x_3 \; 0.8x_4 \; 1.8x_5 \; 1.3x_7 \; 0.8x_8 \le 40$$
$$0.2x_1 \; 1.2x_2 \; 0.6x_3 \; 0.2x_4 \; 0.7x_5 \; 0.4x_6 \; 1.2x_7 \le 25$$
$$0.2x_3 \; 0.2x_4 \; 0.8x_7 \; 1.2x_8 \le 30$$
$$0.6x_1 \; 2.1x_2 \; 0.2x_3 \; 1.2x_6 \le 15$$
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \ge 0$$

Problem 9

Minimize $f(\boldsymbol{x}) = x_1 \; 2x_2 \; 1.2x_3 \; 0.5x_4 \; 1.5x_5 \; 2.5x_6 \; 3x_7 \; x_8 \; 2x_9 \; (2x_1^2 \; x_2^2 \; 4x_3^2 \; 3x_4^2 \; x_5^2 \; 2x_6^2 \; x_7^2 \; x_8^2 \; x_9^2 \; 4x_1x_3 \; 5x_1x_4 \; 2x_1x_7 \; 4x_1x_9 \; 8x_2x_4 \; x_2x_6 \; 3x_2x_8 \; x_3x_6 \; 0.5x_3x_8 \; 2.5x_4x_5 \; 4x_4x_7 \; x_4x_9 \; 0.5x_5x_6 \; 2x_5x_7 \; 4x_6x_7 \; x_6x_8 \; 6x_8x_9)$

Subject to
$$5x_1 \; 6x_3 \; x_4 \; 14x_5 \; 2x_6 \; x_7 \; 3x_8 \; x_9 \le 85$$
$$4x_1 \; x_2 \; 2x_4 \; 2x_6 \; 4x_8 \le 87$$
$$8x_2 \; x_4 \; 4x_5 \; 2x_7 \; 2x_8 \le 149$$
$$x_1 \; 3x_2 \; 2x_5 \; 3x_7 \; x_9 \le 53$$
$$2x_1 \; 5x_2 \; 4x_3 \; 6x_5 \; x_6 \; 5x_7 \; 3x_8 \; 2x_9 \le 68$$
$$5x_3 \; 3x_4 \; 2x_6 \; 3x_7 \; x_9 \le 72$$
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9 \ge 0$$

Problem 10

Minimize $f(\boldsymbol{x}) = 0.25x_1 \; 2x_2 \; 0.5x_3 \; 0.2x_4 \; 7x_5 \; x_6 \; x_7 \; x_8 \; 2x_9 \; 0.05x_{10}$
$(x_1^2 \; x_2^2 \; x_3^2 \; x_4^2 \; x_5^2 \; x_6^2 \; x_7^2 \; x_8^2 \; x_9^2 \; x_{10}^2 \; 2x_1x_2 \; 6x_1x_3$
$x_1x_4 \; 1.99x_1x_6 \; 3x_1x_{10} \; x_2x_5 \; 6x_2x_7 \; 5.1x_2x_8 \; x_2x_9$
$1.11x_2x_{10} \; 0.045x_3x_4 \; 3.5x_3x_7 \; 1.5x_3x_9 \; 2.5x_4x_9$
$1.5x_4x_{10} \; x_5x_6 \; x_5x_7 \; 0.98x_6x_8 \; 1.12x_7x_9 \; 0.9x_7x_{10}$
$x_9x_{10})$

Subject to $1.5x_1 \; 4.5x_3 \; 3x_5 \; 9x_7 \; 2x_9 \le 23$
$2x_2 \; 1.5x_4 \; 0.2x_6 \; 0.4x_8 \; x_{10} \le 9$
$3x_2 \; 13x_5 \; 7.7x_6 \; 1.4x_9 \le 35$
$6x_1 \; 3.8x_5 \; 4.9x_{10} \le 48$
$x_2 \; 0.5x_3 \; 3x_4 \; 2x_7 \le 19$
$x_3 \; 16x_4 \; 4.1x_6 \; 2.2x_8 \; 4.4x_9 \; 5x_{10} \le 77$
$27x_1 \; 25x_4 \; 33x_5 \; 16x_7 \; 16x_9 \le 225$
$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \ge 0$