UNIVERSITY OF CAPE COAST

A STUDY OF ITERATIVE METHODS IN OPTIMIZATION

HENRIETTA NKANSAH

2009

UNIVERSITY OF CAPE COAST

A STUDY OF ITERATIVE METHODS IN OPTIMIZATION

BY

HENRIETTA NKANSAH

THESIS SUBMITTED TO THE DEPARTMENT OF MATHEMATICS & STATISTICS
OF THE SCHOOL OF PHYSICAL SCIENCES, UNIVERSITY OF CAPE COAST
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR AWARD OF
MASTER OF PHILOSOPHY DEGREE IN MATHEMATICS

DECEMBER 2009

DECLARATION

Candidate's Declaration

I hereby declare that this thesis is the result of my own original work and that no part of it has been presented for another degree in this University or elsewhere.

Candidate's Signature: .... Date: ....

Name: ....

Supervisors' Declaration

We hereby declare that the preparation and presentation of the thesis were supervised in accordance with the guidelines on supervision of thesis laid down by the University of Cape Coast.

Principal Supervisor's Signature: .... Date: ....

Name: PROF. FRANCIS BENYAH

Co-Supervisor's Signature: .... Date: ....

Name:...

ABSTRACT

Optimization techniques are called into play everyday in decision making processes involving resource allocation in almost every discipline. However, the development and implementation of algorithm for solving optimization problems are fraught with many difficulties and frustrations.

The aim of this thesis has been to examine and identify some of the problems associated with the development and implementation of a class of optimization algorithms and to develop a means of improving upon them. The study made use of functions such as the Rosenbrock's function that are known to be a good test of robustness of these techniques.

The study covered a number of algorithms in both unconstrained and constrained optimization. Some of the problems encountered were in the implementation of the Modified Newton's method. It was discovered that if at any iterate $\mathbf{x}_k$, the Hessian matrix $\mathbf{H}(\mathbf{x}_k)$ is not positive definite, then $-\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ is not a descent direction. In this case, we look for a new direction where the Hessian matrix will be positive definite. Some of the suggestions proposed in the literature did not always lead to a descent direction. A new iterate could be found from $\mathbf{x}_{k+1} = \mathbf{x}_k - \lambda_k \mathbf{v}_k$ where $\mathbf{v}_k$ is the eigenvector corresponding to the negative eigenvalue of $\mathbf{H}(\mathbf{x}_k)$. However, if this fails then an alternative is to make use of the Hessian at the previous iterate $\mathbf{H}(\mathbf{x}_{k-1})$ to compute the next iterate $\mathbf{x}_{k+1}$ which will hopefully give a positive definite Hessian. If this also fails, then setting $\mathbf{H}(\mathbf{x}_k)^{-1} = \mathbf{I}_n$ converts the Modified Newton's method into the Steepest Descent method, where $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$.

The study also revealed that to determine the various critical points of a given function, it may require more than one technique. All the computations in this work made use of OCTAVE, a public domain mathematical software.

ACKNOWLEDGEMENTS

I am most grateful to my Principal Supervisor, Professor Francis Benyah of the University of the Western Cape, Cape Town, South Africa for his invaluable suggestions, comments and guidance throughout the stages of this work.

I am also grateful to my Co-Supervisor, Dr.(Mrs) N.G Mensah, Head of Department of Mathematics and Statistics, University of Cape Coast, for coordinating every stage of this work and for her constructive guidance to the success of this work.

I am indebted to Professor B.K Gordor, Vice Dean of School of Physical Science, University of Cape Coast, for his motivation and guidance throughout the programme. My indebtedness is also to Professor F.K.A. Allotey, President of the Institute of Mathematical Science, Accra, for providing me with financial support throughout the period of the programme. The several conferences that he also arranged for me to participate were of immense benefit to me. Thanks are also extended to all the lecturers and staff of the Department of Mathematics and Statistics for their various support and contributions to the success of this work.

Lastly, I am grateful to my family for standing by me to the very end of the programme.

DEDICATION

To my family

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Background of the Study

Optimization is the art and science of allocating scarce resources to the best possible effect. The day-to-day actions chosen by decision-makers are often modeled as solutions of maximization or minimization problems. A variety of practical problems involving decision making can be formulated as mathematical optimization problems. Mathematical optimization has become an indispensable tool in many disciplines. Optimization techniques are called into play everyday in questions of industrial planning, resource allocation, scheduling, system design, analysis and operations.

A firm, may decide to choose policies that maximizes its sales, and profit; it may on the other hand, choose a policy that minimizes its costs. A government may choose policies to maximize its chance of re-election. A petroleum refiner, for example, must decide where to buy crude oil, where to ship it for processing, what products to convert it to, where to sell those products, and at what prices. His overall objective will be to maximize his profits.

An airline must decide on how to route its planes and schedule its crews at minimum cost while meeting constraints on airplane flight hours between maintenance and maximum flight time for crews.

In *portfolio optimization*, we seek the most profitable way of investing some capital in a set of n assets. The i-th component $\mathbf{x}_i$ of the vector $\mathbf{x}$ represents the investment in the i-th asset. The constraints might represent the limitations on the total amount available for investment, or minimum acceptable value of the expected return on the whole portfolio. The objective or cost function might represent the risk or variance of the portfolio return. In such a case, the optimization problem in Equation 1.1 corresponds to choosing a portfolio allocation that minimizes risk among all possible allocations that meet the investor's requirements.

In *data fitting*, the problem is to find a model, from a family of potential models, that best fits some observed data and prior information. Given a set of data points, the *curve of best-fit* is the curve that minimizes the errors between the observed data and the predicting function. Here, the variables are the parameters of the model; the constraints may represent some prior information about required limits on the parameters. The objective function might be a measure of misfit or prediction error between the observed value and the value predicted by the model. The optimization problem in this case, will be to find the model parameters that are consistent with prior information and give the smallest prediction error.

In mathematics, the term **optimization** or **mathematical programming** refers to the study of problems in which one seeks to minimize or maximize a real-valued function, by systematically choosing the values of real or integer variables from within an allowed set.

Statement of the Problem

Given a function

$$f : S \subseteq \mathbb{R}^n \to \mathbb{R} \tag{1.1}$$

we seek an element $\mathbf{x}^* \in S$ such that

(a) $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in S$, (in a minimization problem), or such that

(b) $f(\mathbf{x}^*) \geq f(\mathbf{x})$ for all $\mathbf{x} \in S$, (in a maximization problem).

Such a formulation is called an **optimization problem** or a **mathematical programming problem**. Many real-world and theoretical problems may be modelled in this general framework.

The optimization problem given in Equation 1.1 is an abstraction of making the best possible choice of a vector $\mathbf{x}^*$, in $\mathbb{R}^n$ from a set of possible choices. The function f is called the **objective function**, or **cost function**. The variable $\mathbf{x}^*$ represents the choice made and the value of the objective function $f(\mathbf{x})$ represents the cost of choosing $\mathbf{x}^*$. Since minimizing f is equivalent to maximizing $-f$, we will consider only minimization problems. (We can also think of $-f$ as the value, or utility of choosing $\mathbf{x}^*$).

In Equation 1.1, if the set $S = \mathbb{R}^n$, then we say that the problem is **unconstrained**; in this case Equation 1.1 takes the form

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}}\, f(\mathbf{x}) \tag{1.2}$$

Example 1.1.

$$\min_{\mathbb{R}^2} \; f(\mathbf{x}) = (x_1 + x_2 - 2)^2 + (x_1 - x_2 + 3)^4 \tag{1.3}$$

On the other hand, if $S \subset \mathbb{R}^n$, (S is a proper subset of $\mathbb{R}^n$) then we say that the problem is **constrained**.

Typically, this is often specified by a set of *equalities* and *inequalities* that limit the possible choices of $\mathbf{x}$. The elements of S are called *feasible* solutions. A constrained minimization problem is of the form

$$\text{minimize f}(\mathbf{x}) \tag{1.4a}$$

subject to

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, p \tag{1.4b}$$

$$h_i(\mathbf{x}) = 0, \quad i = 1, \cdots, q \tag{1.4c}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective functional, and the functions $h : \mathbb{R}^n \to \mathbb{R}^p$ and $g : \mathbb{R}^n \to \mathbb{R}^q$ are the equality and inequality constraints, respectively. A feasible solution that minimizes the objective function is called an **optimal solution**. A solution of the optimization problem corresponds to a choice that has minimum cost (or maximum utility), that meets the required constraints and/or specifications.

An important class of optimization is **linear programming**, where the objective function and the constraints are all linear.

$$\text{minimize } \mathbf{c}^{\mathrm{T}}\mathbf{x} \tag{1.5a}$$

subject to

$$\mathbf{a}_i^T\mathbf{x} \leq \mathbf{b}_i, \quad i = 1, \cdots, r \tag{1.5b}$$

$$\mathbf{a}_i^T\mathbf{x} = \mathbf{b}_i, \quad i = r+1, \cdots, m \tag{1.5c}$$

Here, the vectors $\mathbf{c}$, $\mathbf{a}_i \in \mathbb{R}^n$, and the scalars $\mathbf{b}_i$, are the problem parameters that specify the objective function and the constraints.

When the objective function or any of the constraints are nonlinear, we call Equation 1.4 a **nonlinear programming (NLP) problem** or **nonlinear optimization** problem.

Many of the large scale optimization techniques for solving linear programming problems today can trace their origins to methods developed during World War II to deal with the massive logistical issues raised by huge armies having millions of men and machines. Techniques that promised to improve the effectiveness of the war effort were desperately needed. For instance, there was the need for optimum allocation of gasoline supplies among competing campaigns.

The fundamentals of the first practical, large-scale optimization technique in Linear Programming, the **simplex method**, were developed during World War II. The simplex method was perfected shortly after the war when the first electronic computers were becoming available. In fact, the early history of computing is closely intertwined with the history of practical optimization.

In the early years, the vast majority of all calculation on electronic computers was devoted to optimization via the simplex method.

Although allocating resources to activities is the most common type of application, linear programming has numerous other important applications as well. In fact, any problem whose mathematical model fits the general format for the linear programming model is a linear programming problem. Furthermore, a remarkably efficient solution procedure, called the *simplex method* is available for solving linear programming problems of even enormous size. These are some of the reasons why linear programming has had tremendous impact in industry in recent decades.

A sub-field of Linear Programming is **integer programming**, which studies linear programming problems in which some or all variables are constrained to take on integer values.

Purpose of the Study

Studies into Optimization techniques are well documented in the literature. Many of these researchers have proposed various modifications to some of the Optimization techniques to go round problems that have been encountered in the implementation of the procedures and algorithms of these techniques.

This research is also concerned with the study of Optimization techniques. The purpose of the study is as follows:

1. to carry out a general study of the various Optimization techniques;

2. to highlight the strengths and weaknesses of various Optimization techniques by the use of some test functions;

3. to identify possible problems associated with the implementation of some of the procedures and algorithms that are used in Optimization techniques;

4. to be able to make appropriate recommendations for solving possible problems associated with proposed methods of some techniques.

Notations and Mathematical Background

Basic concepts in Optimization rely on techniques in multi-variable calculus and numerical linear algebra, see for example, Broyden (1975).

A vector in $\mathbf{x} \in \mathbb{R}^n$ will be represented by

$$\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$$

where the superscript T denote transposition of $\mathbf{x}$. An $n \times n$ matrix will be represented by a bold upper case letter

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

The **gradient vector**, or vector of first partial derivatives of the function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, is defined as

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}_{\mathbf{x}}$$

The gradient vector, $\nabla f(\mathbf{x})$, points in the direction where f is increasing most rapidly, which is the direction of **steepest ascent**. It follows that $-\nabla f(\mathbf{x})$ points in the direction where the function is decreasing most rapidly, called the direction of **steepest descent**.

The **Hessian matrix**, or matrix of second-order partial derivatives of f, written as $\nabla^2 f(x)$, and denoted by

$$\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

is square and symmetric if f is twice continuously differentiable. The Hessian gives information about the curvature of f at any given point $\mathbf{x}$.

For a function of one variable, $h : \mathbb{R} \rightarrow \mathbb{R}$, the curvature at a point x_k is simply the second derivative $h''(x_k)$ at that point. For a function of several variables, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, the curvature at a point $\mathbf{x}_k$ is given by the Hessian matrix $\mathbf{H}(\mathbf{x}_k)$.

Example 1.2. Let

$$f(\mathbf{x}) = x_1^4 + x_1 x_2 + (1 + x_2)^2. \tag{1.6}$$

The gradient of the function f at $\mathbf{x}$ is given by

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 4x_1^3 + x_2 \\ \\ x_1 + 2 + 2x_2 \end{bmatrix}$$

and the Hessian is given by

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 12x_1^2 & 1 \\ \\ 1 & 2 \end{bmatrix}$$

At the point $\mathbf{x}_0 = [0; 0]$,

$$\nabla f(\mathbf{x}_0) = \begin{bmatrix} 0 \\ \\ 2 \end{bmatrix}$$

and the Hessian is

$$\mathbf{H}(\mathbf{x}_0) = \begin{bmatrix} 0 & 1 \\ \\ 1 & 2 \end{bmatrix}$$

At another point, say $\mathbf{x}_1 = [0.75, -0.25]$,

$$\nabla f(\mathbf{x}_1) = \begin{bmatrix} 1.4375 \\ 2.2500 \end{bmatrix}$$

and the Hessian is

$$\mathbf{H}(\mathbf{x}_1) = \begin{bmatrix} 6.74998 & 1 \\ 1 & 2 \end{bmatrix}$$

Definition 1.1. Quadratic Forms

Let $\mathbf{A}$ be an $n \times n$ symmetric matrix. The **quadratic form** $q(\mathbf{x})$, associated with the matrix $\mathbf{A}$ is defined by

$$q(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}.$$

For $n = 2$, a quadratic form on $\mathbb{R}^2$ takes the form

$$q(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = a_{11}x_1^2 + 2a_{12}x_1x_2 + a_{22}x_2^2$$

Similarly, a quadratic form on $\mathbb{R}^3$ takes the form

$$\begin{aligned} q(\mathbf{x}) &= \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2 + 2a_{12}x_1x_2 + 2a_{13}x_1x_3 + 2a_{23}x_2x_3 \end{aligned}$$

A quadratic form $q(\mathbf{x})$ is called

(a) **positive definite** if $q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all nonzero vectors $\mathbf{x} \in \mathbb{R}^n$;

(b) **positive semidefinite** if $q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for all nonzero vectors $\mathbf{x} \in \mathbb{R}^n$;

(c) **negative semidefinite** if $q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} \leq 0$ for all nonzero vectors $\mathbf{x} \in \mathbb{R}^n$;

(d) **negative definite** if $q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} < 0$ for all nonzero vectors $\mathbf{x} \in \mathbb{R}^n$;

(e) **indefinite** if

(i) $q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for some $\mathbf{x} \in \mathbb{R}^n$ and

(ii) $q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} < 0$ for other $\mathbf{x} \in \mathbb{R}^n$.

Each of the following is a necessary and sufficient condition for a real symmetric matrix $\mathbf{A}$ to be positive definite.

(a) $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all nonzero vectors $\mathbf{x} \in \mathbb{R}^n$;

(b) Each eigenvalue of $\mathbf{A}$ satisfies $\lambda_i > 0$.

A simpler test for positive definiteness, using **leading principal minors** is given below.

Definition 1.2. Leading Principal Minors

Given an $n \times n$ matrix $\mathbf{A}$, the k-th **leading principal minor** M_k is the determinant of the $k \times k$ submatrix obtained by deleting the last $n - k$ rows and columns from $\mathbf{A}$.

Example 1.3. For a 3×3 matrix $\mathbf{A}$, the three leading principal minors are

$$M_1 = |a_{11}|, \quad M_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \quad M_3 = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

Test for Definiteness Using Principal Minors

Let $\mathbf{A}$ be an $n \times n$ symmetric matrix. If

(a) $M_k > 0,\ k = 1, \cdots, n$ then $\mathbf{A}$ is positive definite.

(b) $M_k \geq 0,\ k = 1, \cdots, n$ then $\mathbf{A}$ is positive semidefinite.

(c) $M_k \neq 0$, $k = 1, \cdots, n$, but $M_k > 0$, and $M_k < 0$, for different values of k, then $\mathbf{A}$ is indefinite.

(d) A symmetric $n \times n$ matrix $\mathbf{A}$ is positive definite $\iff$ $-\mathbf{A}$ is negative definite. Since

$$\mathtt{det}(-\mathbf{A}) = (-1)^n \mathtt{det}(\mathbf{A}) \begin{cases} > 0, & \text{if } n \text{ is even} \\ < 0, & \text{if } n \text{ is odd,} \end{cases}$$

if

(i) $M_k < 0$, for odd k, and $M_k > 0$ for even k, then $\mathbf{A}$ is negative definite.

(ii) $M_k \leq 0$, for odd k, and $M_k \geq 0$ for even k, then $\mathbf{A}$ is negative semidefinite.

Example 1.4. The following are examples of quadratic forms on $\mathbb{R}^2$.

(a) $$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2x_1^2 + 3x_2^2, \text{ (positive definite)}$$

(b) $$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} -2 & 0 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = -2x_1^2 - 3x_2^2, \text{ (negative definite)}$$

(c) $$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 4x_1^2 - 5x_2^2, \text{ (indefinite)}$$

(d) $$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1^2 - 2x_1x_2 + x_2^2, \text{ (positive semidefinite).}$$

The graphs of the above quadratic forms are given in Figure 1 below.

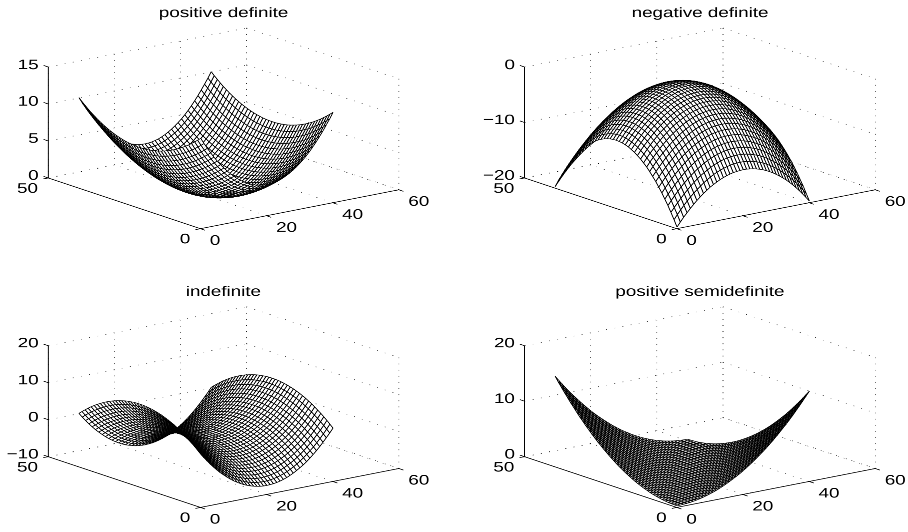

Figure 1: Graphs of various quadratic forms in standard position

The quadratic form of a function given by

$$q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + c$$

has its gradient to be

$$\nabla q(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}.$$

This is because

$$\begin{aligned}\nabla[\frac{1}{2}(\mathbf{x}^T(\mathbf{A}\mathbf{x}))] - \nabla(\mathbf{b}^T\mathbf{x}) + \nabla(c) &= \frac{1}{2}[\mathbf{I}\mathbf{A}\mathbf{x} + \mathbf{A}^T\mathbf{x}] - \nabla(\mathbf{x}^T\mathbf{b}) \\ &= \frac{1}{2}[\mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{x}] - \mathbf{I}\mathbf{b} \\ &= \mathbf{A}\mathbf{x} - \mathbf{b}\end{aligned}$$

since the matrix $\mathbf{A}$ is symmetric. The consequence of this is that $\nabla q(\mathbf{x}) = 0$, and that q has a stationary point at the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$. Therefore, solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, finds the minimum of q.

Convex Sets and Convex Functions

Some functions have a nonnegative second derivative everywhere, and such functions are referred to as convex. A simple example is the function $f(x) = x^2$. For functions of several variables $f : \mathbb{R}^n \to \mathbb{R}$, a nonnegative second derivative means that the Hessian matrix $\mathbf{H}(\mathbf{x})$ is nonnegative definite (that is, all eigenvalues are larger than or equal to zero). The reason we are interested in convex functions is found in the following useful results:

Definition 1.3. Convex sets A set S is *convex* if the line segment joining any two points x, y in S also lies in S. That is, for any $x, y \in S$, and $0 \leq t \leq 1$, we have

$$tx + (1-t)y \in S.$$

Definition 1.4. Convex functions

(a) A real-valued function f defined on an interval (or on any convex subset S of some vector space) is called **convex**, if for any two points x and y in its domain, and any t in [0, 1], we have

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y). \tag{1.7a}$$

(b) A real-valued function f, is said to be **strictly convex** if

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y) \tag{1.7b}$$

for any t in [0, 1] and $x \neq y$.

(c) A real-valued function f, is said to be **concave** if $(-f)$ is convex.

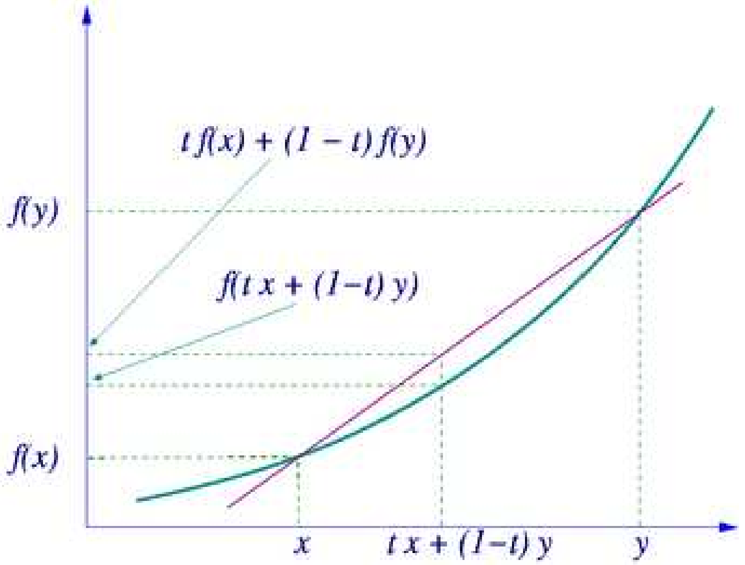

Figure 2: A convex function on an interval

Examples of convex functions $f : \mathbb{R} \to \mathbb{R}$ are $f(x) = x^2$, $f(x) = e^x$, $f(x) = |x|$. Example of concave functions are $f(x) = \ln(x)$, $f(x) = \sqrt{(}x)$.

With $t = 1/2$, Equation 1.7a becomes

$$f\left(\frac{x+y}{2}\right) \le \frac{f(x)+f(y)}{2} \tag{1.8}$$

for all x and y in S.

Theorem 1.1. A continuously differentiable function f is convex on a set C, if and only if $f(y) \ge f(x) + f'(x)(y - x)$, for all $x, y, \in C$.

Proof. ($\Longrightarrow$:) Assume f is convex. Then, for all $x, y \in C$,

$$\begin{aligned} f((1-t)x + ty) &\le (1-t)f(x) + tf(y) \\ f(x + t(y - x)) &\le f(x) + t(f(y) - f(x)) \\ \frac{(x + t(y - x)) - f(x)}{t} &\le f(y) - f(x) \end{aligned}$$

Taking the limit as $t \to 0$ gives

$$\begin{aligned} f'(x)(y-x) &\leq f(y)-f(x) \\ f'(x)(y-x)+f(x) &\leq f(y) \end{aligned}$$

($\Longleftarrow$:) Let $x=(1-t)x_1+tx_2$. Then, by assumption

$$f(x_1) \geq f(x)_(x_1-x)f'(x)$$

$$f(x_2) \geq f(x)_(x_2-x)f'(x)$$

Geometrically, the above theorem says that a convex function lies above all of its tangents.

$$f(y) \geq f(x)+f'(x)(y-x)$$

if at $x=x^*$, $f'(x^*)=0$ then $f(y) \geq f(x^*)$, $\forall y \in \mathbb{R}$

Remark 1.1. Any local minimum of a convex function is also a global minimum. A strictly convex function will have at most one global minimum.

A twice differentiable function of one variable is convex on an interval if and only if its second derivative is non-negative there. This gives a practical test for convexity. If its second derivative is positive then it is strictly convex, but the converse does not hold, as shown by $f(x)=x^4$.

More generally, a continuous, twice differentiable function of several variables is convex on a convex set if and only if its Hessian matrix is positive semidefinite on the interior of the convex set.

(a) Let f be convex. If $\mathbf{x}^*$ satisfies the first order conditions, then $\mathbf{x}^*$ is a global minimum point of f.

(b) Let f be strictly convex. If $\mathbf{x}^*$ satisfies the first order condition, then $\mathbf{x}^*$ is a strict global minimum point of f.

Example 1.5. Suppose that $f(\mathbf{x})$ is a convex function, then we want to show that the set of global minimizers of f is a convex set. To show this we let $\mathbf{S}$ denote the set of the global minimizers. Obviously, we have

$$f(\mathbf{x}) = \min f$$

for any $\mathbf{x} \in \mathbf{S}$. Take any $\mathbf{x}, \mathbf{y} \in \mathbf{S}$, we just need to show that

$$\lambda\mathbf{x} + (1-\lambda)\mathbf{y} \in \mathbf{S}$$

for any $\lambda \in (0,1)$. Now by convexity and for equality to hold,

$$\min f \leq f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}) = \min f$$

Local Convergence

Most Optimization problems are solved numerically by computing approximations $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots$ which hopefully converge to the solution $\mathbf{x}^*$.

Definition 1.5. Convergence

Let $\mathbf{x}_n \subset \mathbb{R}^n$ and $\mathbf{x}^* \in \mathbb{R}^n$. Then

(a) $\mathbf{x}_n \rightarrow \mathbf{x}^*$ q-quadratically if $\mathbf{x}_n \rightarrow \mathbf{x}^*$ and there is $K > 0$ such that

$$\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \leq K\|\mathbf{x}_n - \mathbf{x}^*\|^2$$

(b) $\mathbf{x}_n \rightarrow \mathbf{x}^*$ q-superlinearly with q-order $\alpha > 1$ if $\mathbf{x}_n \rightarrow \mathbf{x}^*$ and there is $K > 0$ such that

$$\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \leq K\|\mathbf{x}_n - \mathbf{x}^*\|^\alpha$$

(c) $\mathbf{x}_n \rightarrow \mathbf{x}^*$ q-superlinearly if

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = 0$$

(d) $\mathbf{x}_n \to \mathbf{x}^*$ q-linearly with q-factor $\sigma \in (0,1)$ if

$$\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \leq \sigma \|\mathbf{x}_n - \mathbf{x}^*\|$$

for n sufficiently large.

Maximum and Minimum Values of a function

Definition 1.6. Let $f(\mathbf{x})$ be a real-valued function defined on some set S . A point $\mathbf{x}^*$ in S is called

(a) a **critical point** of $f(\mathbf{x})$ if $\nabla f(\mathbf{x}^*)$ exists and is equal to zero;

(b) **a global minimizer** for $f(\mathbf{x})$ on S if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in S;

(c) a **strict global minimizer** for $f(\mathbf{x})$ on S if $f(\mathbf{x}^*) < f(\mathbf{x})$ for all $\mathbf{x}$ in S such that $\mathbf{x} \neq \mathbf{x}^*$;

(d) a **local minimizer** for $f(\mathbf{x})$ if there is a vector $\boldsymbol{\delta}$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in S for which $\mathbf{x}^* - \boldsymbol{\delta} < \mathbf{x} < \mathbf{x}^* + \boldsymbol{\delta}$;

(e) a **strict local minimizer** for $f(\mathbf{x})$ if there is a vector $\boldsymbol{\delta}$ such that $f(\mathbf{x}^*) < f(\mathbf{x})$ for all $\mathbf{x}$ in S for which $\mathbf{x}^* - \boldsymbol{\delta} < \mathbf{x} < \mathbf{x}^* + \boldsymbol{\delta}$ and $\mathbf{x} \neq \mathbf{x}^*$.

Theorem 1.2. Suppose that $f(\mathbf{x})$ is a differentiable function on some set S. If $\mathbf{x}^*$ is a local minimizer or maximizer of $f(\mathbf{x})$, then either $\mathbf{x}^*$ is an endpoint of S or $\nabla f(\mathbf{x}^*) = 0$.

We state the following theorem for the case where $\mathbf{x}$ in the above definition is a single variable.

Theorem 1.3. Suppose that $f(x)$, $f'(x)$,$f''(x)$ are all continuous on an interval I and that $x^* \in I$ is a critical point of $f(x)$.

(a) If $f''(x) \geq 0$ for all $x \in I$, then x^* is a global minimizer of $f(x)$ on I .

(b) If $f''(x) > 0$ for all $x \in I$ such that $x \neq x^*$, then x^* is a strict global minimizer of $f(x)$ on I .

(c) If $f''(x^*) > 0$,then x^* is a strict local minimizer of $f(x)$ on I.

We consider the cases in the above theorem in the three figures below:

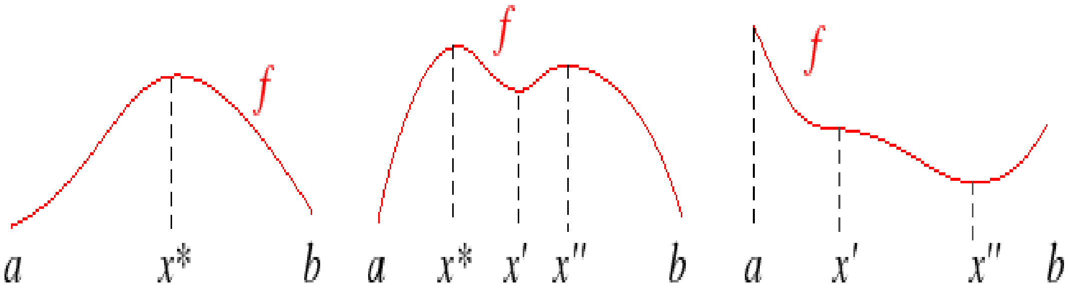

Figure 3: Stationary points of f on the interval $I = [a, b]$

(a) In the left figure, the unique stationary point x^* is the global maximizer.

(b) In the middle figure, there are three stationary points: x^*, x' and x''. The point x^* is the global maximizer, while x' is a local minimizer, and x'' is a local maximizer.

(c) In the right figure, there are two stationary points: x' and x''. The point x' is neither a local maximizer nor a local minimizer; x'' is a global minimizer.

(d) It is therefore, clear that

   (i) a stationary point is not necessarily a maximizer or minimizer; for example, the point x' in the right-hand figure.

   (ii) a maximizer (local or global) is not necessarily a stationary point; for example, the point a in the right-hand figure.

From the discussion above we see that being a stationary point is neither a *necessary* condition nor a *sufficient* condition for being a maximum or minimum point. Although a maximizer (or a minimizer) may not be a stationary point, the only case in which it is not is when it is one of the endpoints of the interval I on which f is defined. That is, any point interior to this interval that is a maximum (or a minimum) must be a stationary point. The relationship between stationary points and maximizers (minimizers) is stated in the following Theorem.

Theorem 1.4. Let f be a differentiable function of a single variable defined on the interval $I = [a, b]$. If a point x in the interior of I is a local or global maximizer or minimizer of f then $f'(x) = 0$.

This result gives a necessary condition for x to be a maximizer (or a minimizer) of f: if it is a maximizer (or a minimizer) and is between a and b then x is a stationary point of f. The condition is obviously not sufficient for a point to be a maximizer; the condition is satisfied also, for example, at points that are minimizers. Since the first-derivative is involved, we refer to the condition as a **first-order** condition.

Example 1.6. Consider $f(x) = 3x^4 - 4x^3 + 1$.
Since $f'(x) = 12x^3 - 12x^2 = 12x^2(x - 1)$, the only critical points of $f(x)$ are $x^* = 0$ and $x^* = 1$. Also, since $f''(x) = 36x^2 - 24x = 12x(3x - 2)$, we see that $f''(0) = 0$ and $f''(1) = 12$. Therefore, $x^* = 1$ is a strict local minimizer of $f(x)$. To analyze the behaviour of $f(x)$ near $x = 0$, we observe that $x^4 < x^3$ for $0 < x < 1$ so that $f(x) < 1$ just to the right of the origin, while $f(x) > 1$ to the left of the origin. Consequently, the critical point $x^* = 0$ is neither a maximizer nor a minimizer of $f(x)$; rather it is a "horizontal point of inflection" for $f(x)$. So $f(x)$ has no global maximizer on R. The strict local minimizer $x = 1$ is also a strict global minimizer. See Figure 10 in the Appendix for the graph of the function.

We consider another example in which $\mathbf{x}$ is this time in $\mathbb{R}^2$.

Example 1.7. Consider a well-known test function, the Rosenbrock's function, which is also referred to as the Banana function. This is given as

$$f(\mathbf{x}) \quad = \quad 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$\nabla f(\mathbf{x}) \quad = \quad \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}$$

$$\nabla^2 f(\mathbf{x}) \quad = \quad \begin{bmatrix} -400(x_2 - x_1^2) + 800x_1^2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

Solving $\nabla f(\mathbf{x}) = 0$, we get $\mathbf{x} = [1; 1]$ as the only solution. So $f(\mathbf{x})$ has only one critical point. Since $f[1; 1] = 0$ and obviously $f \geq 0$, we know that it is a local minimizer as well as a global minimizer. At $[1; 1]$, we have

$$\nabla^2 f = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$$

This process is illustrated in the following picture.

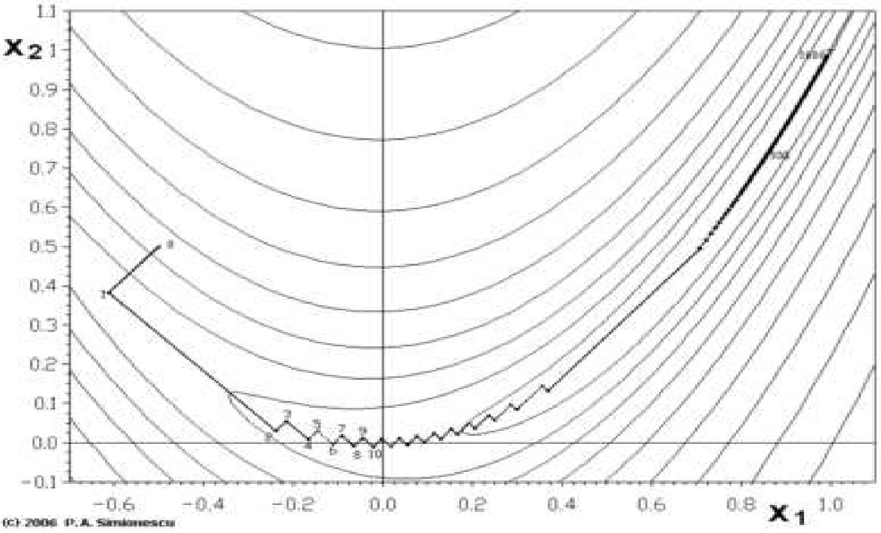

Figure 4: Contours of the Banana function

Theorem 1.5. First Order Conditions

The derivative of a function is always equal to zero at a minimum point.

This fact is called the **first order necessary condition** for a minimum. The reason we include the word ‘necessary’ is that all minimum points satisfy this condition, but not every point satisfying this condition needs to be a minimum point (it could also be for example a maximum). The precise formulation of the first order condition for a function of several variables, $f : \mathbb{R}^n \to \mathbb{R}$, is as follows: Let the derivative of f exists and be continuous. If $\mathbf{x}^*$ is a global minimum point of f, then $\nabla f(\mathbf{x}^*) = 0$. Thus, if we want to know whether a given point $\mathbf{x}$ is a minimum point or not, the first thing we should check is whether the gradient vector is equal to zero at $\mathbf{x}$.

Proof. By Taylor's theorem, for $0 < \theta < 1$,

$$f(\mathbf{x}^* + \mathbf{h}) - f(\mathbf{x}^*) = \nabla f(\mathbf{x}^*)\mathbf{h} + \frac{1}{2}\mathbf{h}^T\mathbf{H}\mathbf{h}|_{\mathbf{x}^*+\theta\mathbf{h}}$$

where $\mathbf{h}$ is a vector of small changes. For sufficiently small $|\mathbf{h}_j|$, the remainder term $\frac{1}{2}(\mathbf{h}^T\mathbf{H}\mathbf{h})$ is of the order $\mathbf{h}_j^2$, and hence the expansion may be approximated as

$$f(\mathbf{x}^* + \mathbf{h}) - f(\mathbf{x}^*) = \nabla f(\mathbf{x}^*)\mathbf{h} + o(\mathbf{h}_j^2) \cong \nabla f(\mathbf{x}^*)\mathbf{h}.$$

Suppose that $\mathbf{x}^*$ is a minimum point; then it is shown by contradiction that $\nabla f(\mathbf{x}^*)$ must vanish. Suppose it does not; then for a specific j, either

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_j} < 0 \quad or \quad \frac{\partial f(\mathbf{x}^*)}{\partial x_j} > 0$$

By selecting $\mathbf{h}_j$ with appropriate sign, it is always possible to have

$$\mathbf{h}_j \frac{\partial f(\mathbf{x}^*)}{\partial x_j} < 0$$

By setting all other $\mathbf{h}_j$ equal to zero, Taylor's expansion yields

$$f(\mathbf{x}^* + \mathbf{h}) < f(\mathbf{x}^*)$$

This result contradicts the assumption that $\mathbf{x}^*$ is a minimum point. Therefore, at any extreme point, the condition

$$\nabla f(\mathbf{x}^*) = 0$$

must be satisfied.

Theorem 1.6. Second Order Conditions

If the first order condition is satisfied and the second derivative is positive, then we have a minimum point. This fact is called the **second order sufficient condition** for a minimum. We include the word 'sufficient' to indicate that all points that satisfy this conditions are minima, but some minima do not satisfy the condition (that is, the condition is not necessary). The precise formulation of the second order condition for a function of several variables, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, is as follows: Let the first and second derivatives both exist and be continuous. If at $\mathbf{x} = \mathbf{x}^*$,

1. $\nabla f(\mathbf{x}^*) = 0$; and
2. $\mathbf{H}(\mathbf{x}^*)$ is positive semidefinite,

then $\mathbf{x}^*$ is a local minimum point of f.

Proof. By Taylor's theorem, for $0 < \theta < 1$,

$$f(\mathbf{x}^* + \mathbf{h}) - f(\mathbf{x}^*) = \nabla f(\mathbf{x}^*)\mathbf{h} + \frac{1}{2}\mathbf{h}^T\mathbf{H}\mathbf{h}|_{\mathbf{x}^*+\theta\mathbf{h}}.$$

Since $\mathbf{x}^*$ is a stationary point, by Theorem 1.2, $\nabla f(\mathbf{x}^*) = 0$. Thus

$$f(\mathbf{x}^* + \mathbf{h}) - f(\mathbf{x}^*) = \frac{1}{2}\mathbf{h}^T\mathbf{H}\mathbf{h}|_{\mathbf{x}^*+\theta\mathbf{h}}.$$

Let $\mathbf{x}^*$ be a minimum point; then, by definition,

$$f(\mathbf{x}^* + \mathbf{h}) > f(\mathbf{x}^*)$$

for all non null $\mathbf{h}$. This means that for $\mathbf{x}^*$ to be a minimum, it must be true that

$$\frac{1}{2}\mathbf{h}^T\mathbf{H}\mathbf{h}|_{\mathbf{x}^*+\theta\mathbf{h}} > 0.$$

However, continuity of the second partial derivative guarantees that the expression $\frac{1}{2}\mathbf{h}^T\mathbf{H}\mathbf{h}$ must yield the same sign when evaluated at both $\mathbf{x}^*$ and $\mathbf{x}^* + \theta\mathbf{h}$. Since $\mathbf{h}^T\mathbf{H}\mathbf{h}|_{\mathbf{x}^*}$ defines a quadratic form, this expression is positive if and only if $\mathbf{H}|_{\mathbf{x}^*}$ is positive-definite. This means that a sufficient condition for the stationary point $\mathbf{x}^*$ to be a minimum is that the Hessian matrix evaluated at the same point is positive-definite.

Example 1.8. Consider the function

$$f(x_1, x_2, x_3) = x_1 + 2x_3 + x_2x_3 - x_1^2 - x_2^2 - x_3^2.$$

The necessary condition

$$\nabla f(\mathbf{x}_0) = 0$$

gives

$$\begin{aligned}
\frac{\partial f}{\partial x_1} &= 1 - 2x_1 = 0 \\
\frac{\partial f}{\partial x_2} &= x_3 - 2x_2 = 0 \\
\frac{\partial f}{\partial x_3} &= 2 + x_2 - 2x_3 = 0
\end{aligned}$$

The solution of these simultaneous equations is given by

$$\mathbf{x}_0 = (\frac{1}{2}, \frac{2}{3}, \frac{4}{3}).$$

To establish sufficiency, we consider

$$\begin{aligned}
\mathbf{H}(\mathbf{x}_0) &= \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix}_{\mathbf{x}=\mathbf{x}_0} \\
&= \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}
\end{aligned}$$

The eigenvalues of $\mathbf{H}(\mathbf{x}_0)$ have the values $-3, -2\ and -1$, respectively. Thus, $\mathbf{H}(\mathbf{x}_0)$ is negative-definite and

$$\mathbf{x}_0 = (\frac{1}{2}, \frac{2}{3}, \frac{4}{3})$$

represents a maximum point.

Theorem 1.7. Suppose that $\mathbf{x}^*$ is a critical point of a function $f(\mathbf{x})$ with continuous first and second partial derivatives on $\mathbb{R}^n$. Then:

(a) $\mathbf{x}^*$ is a global minimizer for $f(\mathbf{x})$ if $(\mathbf{x} - \mathbf{x}^*) \cdot \mathbf{H}(\mathbf{z})(\mathbf{x} - \mathbf{x}^*) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$ and all $\mathbf{z} \in [\mathbf{x}^*, \mathbf{x}]$;

(b) $\mathbf{x}^*$ is a strict global minimizer for $f(\mathbf{x})$ if $(\mathbf{x} - \mathbf{x}^*) \cdot \mathbf{H}(\mathbf{z})(\mathbf{x} - \mathbf{x}^*) > 0$ for all $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} \neq \mathbf{x}^*$ and all $\mathbf{z} \in [\mathbf{x}^*, \mathbf{x}]$;

(c) $\mathbf{x}^*$ is a global maximizer for $f(\mathbf{x})$ if $(\mathbf{x} - \mathbf{x}^*) \cdot \mathbf{H}(\mathbf{z})(\mathbf{x} - \mathbf{x}^*) \leq 0$ for all $\mathbf{x} \in \mathbb{R}^n$ and all $\mathbf{z} \in [\mathbf{x}^*, \mathbf{x}]$;

(d) $\mathbf{x}^*$ is a strict global maximizer for $f(\mathbf{x})$ if $(\mathbf{x} - \mathbf{x}^*) \cdot \mathbf{H}(\mathbf{z})(\mathbf{x} - \mathbf{x}^*) < 0$ for all $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} \neq \mathbf{x}^*$ and all $\mathbf{z} \in [\mathbf{x}^*, \mathbf{x}]$.

Theorem 1.8. Suppose that $\mathbf{x}^*$ is a critical point of a function $f(\mathbf{x})$ with continuous first and second partial derivatives on $\mathbb{R}^n$ and that $\mathbf{H}(\mathbf{x})$ is the Hessian of $f(\mathbf{x})$. Then $\mathbf{x}^*$ is:

(a) a global minimizer for $f(\mathbf{x})$ if $\mathbf{H}(\mathbf{x}^*)$ is positive semidefinite on $\mathbb{R}^n$;

(b) a strict global minimizer for $f(\mathbf{x})$ if $\mathbf{H}(\mathbf{x}^*)$ is positive definite on $\mathbb{R}^n$;

(c) a global maximizer for $f(\mathbf{x})$ if $\mathbf{H}(\mathbf{x}^*)$ is negative semidefinite on $\mathbb{R}^n$;

(d) a strict global maximizer for $f(\mathbf{x})$ if $\mathbf{H}(\mathbf{x}^*)$ is negative definite on $\mathbb{R}^n$.

Example 1.9. (a) A symmetric matrix whose entries are all positive need not be positive definite. For example, the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix}$$

is not positive definite. For if $\mathbf{x} = [1; -1]$, then

$$q(\mathbf{x}) = [1 \ -1] \cdot \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = [1 \ -1] \begin{bmatrix} -3 \\ 3 \end{bmatrix} = -6 < 0.$$

(b) A symmetric matrix with some negative entries may be positive definite. For example, the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix}$$

corresponds to the quadratic form

$$q(\mathbf{x}) = x_1^2 - 2x_1x_2 + 4x_2^2.$$

Since $q(\mathbf{x}) = (x_1 - x_2)^2 + 3x_2^2$, we see that if $\mathbf{x} = (x_1, x_2) \neq (0, 0)$, then $q(\mathbf{x}) > 0$ since $(x_1 - x_2)^2 > 0$ if $x_1 \neq x_2$ and $3x_2^2 > 0$ if $x_1 = x_2$.

(c) The matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

is positive definite because the associated quadratic form $q(\mathbf{x})$ is

$$q(\mathbf{x}) = x_1^2 + 3x_2^2 + 2x_3^2,$$

and so unless $x_1 = x_2 = x_3 = 0$, $q(\mathbf{x}) > 0$. In general, a 3×3 - diagonal matrix

$$\mathbf{A} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}$$

is:

positive definite if $d_i > 0$ for $i = 1, 2, 3$;

((ii) positive semidefinite if $d_i \geq 0$ for $i = 1, 2, 3$;

(iii) negative definite if $d_i < 0$ for $i = 1, 2, 3$;

(iv) negative semidefinite if $d_i \leq 0$ for $i = 1, 2, 3$;

(v) indefinite if at least one d_i is positive and at least one d_i is negative for $i = 1, 2, 3$.

Positive Definite Hessian

The necessary conditions for optimality imply that if a quadratic function f has a local minimum $\mathbf{x}^*$, then $\mathbf{A}$ is positive semidefinite and

$$\mathbf{A}\mathbf{x}^* = \mathbf{b} \tag{1.9}$$

In particular, if $\mathbf{A}$ is semi-positive definite (and hence nonsingular), then the unique global minimizer is the solution of the linear system in Equation 1.9. If $\mathbf{A}$ is a dense matrix and N is not too large, it is reasonable to solve Equation 1.9 by first computing the Cholesky factorization of $\mathbf{A}$.

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T,$$

where $\mathbf{L}$ is a nonsingular lower triangular matrix with positive diagonal. If $\mathbf{A}$ is indefinite, the Cholesky factorization will not exist and the standard implementation will fail because the computation of the diagonal of $\mathbf{L}$ require a real square root of a negative number or a division by zero. If N is very large, $\mathbf{A}$ is sparse, or a matrix representation of $\mathbf{A}$ is not available, and a more efficient approach is the *conjugate gradient iteration.* Our formulation of the algorithm uses $\mathbf{x}$ as both an input and output variable. On input $\mathbf{x}$ contains $\mathbf{x}_0$, the initial iterate, and on output, the approximate solution, $\mathbf{x}^*$. We terminate the iteration if the relative residual is sufficiently small, that is,

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}\| \leq \epsilon \|\mathbf{b}\|,$$

or if too many iterations have been taken.

Indefinite Hessian

If $\mathbf{A}$ is indefinite, the necessary conditions imply that there will be no local minimum. Even so, it will be important to understand some properties of quadratic problems with indefinite Hessians when we design algorithms with initial iterates far from local minimizers. If

$$\mathbf{u}^T \mathbf{A}\mathbf{u} < 0,$$

we say that $\mathbf{u}$ is a direction of negative curvature. In this case, $f(\mathbf{x} + t\mathbf{u})$ will decrease to $-\infty$ as $t \to \infty$.

A General Solution Strategy

The problem of locating a minimum of a function f of several variables is in general difficult, and we should not expect to find some method that is able to take us directly to the desired minimum $\mathbf{x}$, of the function. The usual solution strategy is to divide the minimization problem into many simpler tasks. More specifically, we start from an initial vector $\mathbf{x}$ (provided by the user) and move in small steps towards $\mathbf{x}$ (hopefully) a minimum point. Algorithms of this form are called *iterative.* It is evident that an efficient strategy should be based on the attempt to determine which is the most promising direction or, equivalently, to guess whether the positive curvature information are more significant than the negative curvature information or vice versa. The rule we adopt is based on the rate of decrease of the quadratic model of the objective function. In particular, we compare the decrease of the quadratic model along the negative curvature direction by performing a unit step length along a normalized direction $\mathbf{d}_k$, with the decrease that we would obtain by performing a unit step length along the normalized truncated Newton direction. Since we are interested in solving large scale problems and thus cannot rely on matrix factorizations, we concentrate on iterative methods to compute the search directions.

To plan the next move in such an iteration, we should first get an idea of how the surface looks like in the vicinity of the point we stand in (are we standing in the bottom of a valley, close to a minimum? Or are we up in the hillside, far from the nearest minimum?). A common strategy is to construct a simplified model of the local landscape, and to make further moves based on this model. For example, we may model the local landscape by a (hyper -) plane and then make a move in the direction in which the plane descends the most. The linear approximation can be determined by a Taylor expansion.

Line Search Algorithms

Many well known optimization procedures belong to the class known as *line search algorithms.* Many algorithms have been proposed for solving minimization problems. Such algorithm can be made globally convergent using one of the two basic approaches, the *line search* and the *trust region* approach. We would now concentrate on line search algorithms. Line search is a search method that is used as part of a larger optimization algorithm. The main idea is to determine, at each iteration, a pair of descent directions, $(\mathbf{s}_k, \mathbf{d}_k)$ where, loosely speaking, $\mathbf{s}_k$ represents a direction calculated from positive curvature information given by the Hessian matrix, and $\mathbf{d}_k$ is a negative curvature direction. This pair of directions is then used in a search along the trajectory

At each step of the main algorithm, the line search method searches along the line containing the current point $\mathbf{x}_k$, parallel to the search direction, which is a vector determined by the main algorithm. That is, the method finds the next iterate $\mathbf{x}_{k+1}$ of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha^* \mathbf{d}_k,$$

where $\mathbf{x}_k$ denotes the current iterate, $\mathbf{d}_k$ is the search direction, and α^* is a scalar step length parameter. The line search method attempts to decrease the objective function along the line $\mathbf{x}_k + \alpha \mathbf{d}_k$ by repeatedly minimizing polynomial interpolation models of the objective function. The line search procedure has two main steps:

(a) The *bracketing* phase determines the range of points on the line

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha^* \mathbf{d}_k$$

to be searched. The bracket corresponds to an interval specifying the range of values of α.

(b) The *sectioning* step divides the bracket into subintervals, on which the minimum of the objective function is approximated by polynomial interpolation.

The resulting step length α satisfies the Wolfe conditions:

1. $(\mathbf{x}_k + \alpha \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \alpha \nabla f_k^T \mathbf{d}_k$
2. $\nabla f(\mathbf{x}_k + \alpha \mathbf{d}_k)^T \geq c_2 \alpha \nabla f_k^T \mathbf{d}_k$

where c_1 and c_2 are constants with $0 < c_1 < c_2 < 1$. The first condition requires that α_k sufficiently decreases the objective function. The second condition ensures that the step length is not too small. Points that satisfy both conditions are called acceptable points.

Choice of Search Direction

It seems reasonable to select a search direction such that the function decreases in value as we move in this direction. This is the basic rationale behind all the methods to be described in Chapter Two and Three.

Literature Review

Numerous proposals for minimizing an unconstrained function have been made, particularly since Cauchy's "steepest descent" technique was introduced in 1847. Occasional reviews of the proposals and experience on test problems is published. Unfortunately, these reviews are rarely critical. The result is that variations of the old techniques are reused and the same difficulties are often perpetuated. The published experience tends to give the impression that these methods invariably work. Sometimes a failure is published by Feder (1957), but it is often quickly forgotten. The area of investigation on ways to minimize unconstrained function has only been touched on. Much good work has been done in recent years in this field. Modification of some of the methods to be discussed will probably accelerate the process for problems with special characteristics or structure. Before 1940 relatively little was known about methods for numerical optimization of functions of several variables. There had been some least squares calculations carried out, and steepest descent type methods had been applied in some physics problems. The Newton method in several variables was known, and more sophisticated methods were being attempted such as the self-consistent field method for variational problems in theoretical chemistry. Nonetheless, anything of any complexity demanded armies of assistants operating desk calculating machines. There is no doubt therefore that the advent of the computer was paramount in the development of optimization methods and indeed in the whole of numerical analysis. The 1940s and 1950s saw the introduction and development of the very important branch of the subject known as linear programming. All these methods however had a fairly restricted range of application. The methods were at first very crude and inefficient, but the subject was again revolutionized in 1959 with the publication of a report by Davidon (1973) which led to the introduction of Newton-like methods. Line search descent methods have frequently been used as a means of introducing a degree of reliability into optimization software. In the early days, one common strategy was to choose the step size α_k close to the value given by an exact line search. This is motivated by early theory which shows that the Steepest Descent method with an exact line search is globally convergent to a stationary point by Curry (1944). There is also an unbelievably distant reference to a paper of Cauchy in 1847. However, accurate line searches are expensive to carry out, and there is also the nuisance that the exact minimizer may not exist. Other researchers weakened the line search tolerance considerably and used the descent property merely to force a decrease $f_{k+1} < f_k$ in the objective function on each iteration. This usually turned out to be more efficient. However, merely requiring a decrease in f does not ensure global convergence so there were doubts about the stability of this more efficient approach. There have been many other researches into Conjugate Gradient methods. The relationship of some other methods is discussed by Fletcher (1972a). There are also some other theoretical results: for instance, reset methods are convergent because the Steepest Descent direction is used regularly and exhibit n-step superlinear convergence (McCormick and Pearson, 1969)

$$\|\mathbf{x}_{k+n} - \mathbf{x}^*\| = o(\|\mathbf{x}_k - \mathbf{x}^*\|), \ k = cn + 1, \ c = 0, 1, \cdots$$

However, these results are not really relevant to the practical solution of large problems. A more interesting possibility due to Beale (1972)is that of restarting along directions other than the steepest descent direction. At the expense of increasing the storage requirement, this has been incorporated into an efficient algorithm by Powell (1977b). However, the most interesting of all recent theoretical research results about conjugate gradient methods for minimization is that of Al-Baali(1985) who shows that the non-reset Fletcher-Reeves method with an inexact line search is globally convergent. Zangwill (1965) studied the general penalty function having the form

$$f(\mathbf{x}) + \mathbf{t}G[g(\mathbf{x})]$$

in the space $\mathbb{E}^n$, where $g(\mathbf{x})$ is a q vector whose components, as functions of $\mathbf{x}$, are either zero or non-negative; that is, the problem of concern is

$$\text{minimize} f(\mathbf{x})$$

subject to

$$g_i(\mathbf{x}) \geq 0 \text{ for } i = 1, \cdots, m,$$

$$g_j(\mathbf{x}) = 0 \text{ for } j = m + 1, \cdots, q$$

The loss function is assumed continuous; $G(\mathbf{y}) = 0$ if $\mathbf{y}$ is feasible and is positive otherwise. The other significant assumptions are the continuity of the problem functions, the assumption that the auxiliary function attains a minimum for $\mathbf{t}$ large enough, and the assumption that the feasible region is properly contained in a bounded set. The algorithm proceeds, as usual, minimizing the auxiliary function over $\{\mathbf{t}_k\}$ such that $\mathbf{t}_k \geq 0$ and $\mathbf{t}_k \to \infty$, to generate a corresponding minimizing sequence $\{\mathbf{x}_k\}$. The main result is the existence of a subsequence converging to $\mathbf{x}^*$, a solution of the equation above. Zangwill obtained a dual relationship without convexity assumptions, and other results with convexity and differentiability conditions. Zangwill's work provides a very general treatment of the quadratic penalty function approach to constrained problems in $\mathbb{E}^n$.

Fiacco and McCormick (1967) further developed the quadratic penalty function approach, applying it to the convex programming problem in $\mathbb{E}^n$, where both equality and inequality constraints are permitted. The essential assumption, other than the convexity condition, is the assumption that the set of optimum points of the problem is compact. The procedure was shown to generate a sequence of minimizing points such that every limit point is a solution of the problem. Duality results were obtained. An important fact is that the interior of the feasible region may be empty, implying that the Kuhn-Tucker (1951) constraint qualification- a regularity condition invariably invoked- is not required in order to prove convergence and, hence, optimality. This work was an extension of the results of Pietrzykowski (1962). Significant computational results by this method have only recently been obtained, but it is clear that the technique will prove computationally very effective for nonlinear programming. In a paper, Arrow and Hurwicz (1956) proposed a "differential" method for solving convex programming problems. The method of the problem is essentially as follows: Find a saddle point in $(\mathbf{x}, \mathbf{u})$ of the Lagrangian function $L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) - \sum \mathbf{u}_i g_i(\mathbf{x})$. If there is a point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ such that $L(\bar{\mathbf{x}}, \mathbf{u}) \leq L(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \leq L(\mathbf{x}, \bar{\mathbf{u}})$ for all $\mathbf{u} \geq 0$, then $\bar{\mathbf{x}}$ solves the convex programming problem. Further generalizations and extensions of a number of the above results have recently been obtained by Fiacco (1967). In his work, the theoretical basis for new algorithms is presented, based on modifications of the more general penalty functions that are defined. Strong (1965) proved the validity of the Fiacco and McCormick (1967) procedure based on Carroll's (1961) inverse penalty function for general topological spaces.The central assumptions are continuity of the problem functions and the compactness of the feasible region, from which the convergence of the values of the penalty function, corresponding to any global minimizing feasible interior sequence of points, to the optimal value of a problem is deduced. This constitutes a significant generalization of Carroll procedure and is analogous to the development effected by Butler and Martin (1962) for the quadratic penalty function approach. In 1965 also, we see evidence of the significant application of the quadratic penalty function approach, generalized for multiple variables and multiple inequality constraints. We refer to the work of Beltrami and McGill (1966), who applied the method of constrained variational problems in the theory of search. McGill and Kenneth(1964) also employed an interesting generalization of the Newton-Raphson procedure to operator equations in more general spaces to obtain what they termed an “effective computational method”. Fiacco and McCormick (1967) further developed the quadratic penalty function approach, applying it to the convex programming problem in $\mathbb{E}^n$, where both equality and inequality constraints are permitted. The essential assumption, other than the convexity condition, is the assumption that the set of optimum points of the problem is compact. The procedure was shown to generate a sequence of minimizing points such that every limit point is a solution of the problem. Duality results were obtained. An important fact is that the interior of the feasible region may be empty, implying that the Kuhn-Tucker (1951) constraint qualification-a regularity condition invariably invoked-is not required in order to prove convergence and, hence, optimality. This work was an extension of the results of T.Pietrzykowski (1962). Significant computational results by this method have only recently been obtained, but it is clear that the technique will prove computationally very effective for nonlinear programming. In a paper, Arrow and Hurwicz (1956) proposed a “differential” method for solving convex programming problems. The method of the problem is essentially as follows: Find a saddle point in $(\mathbf{x}, \mathbf{u})$ of the Lagrangian function $L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) - \sum \mathbf{u}_i g_i(\mathbf{x})$. If there is a point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ such that $L(\bar{\mathbf{x}}, \mathbf{u}) \leq L(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \leq L(\mathbf{x}, \bar{\mathbf{u}})$ for all $\mathbf{u} \geq 0$, then $\bar{\mathbf{x}}$ solves the convex programming problem. Further generalizations and extensions of a number of the above results have recently been obtained by Fiacco (1967). In his work, the theoretical basis for new algorithms is presented, based on modifications of the more general penalty functions that are defined.

Outline of the Thesis

This section outlines the contents within each of the five chapters of the thesis, and gives a brief description of these contents.

The Introduction is the first chapter of the study. It looks at the background of the study which brings out the history of stages of applications of Optimization techniques and the need for more innovative efforts in the field. This is followed by the objectives of the study. It gives the Mathematical background and notations that are used in the study of both Unconstrained and Constrained optimization. Next is the review of literature. It discusses research made in the field of Optimization.

Chapter Two is on the study of Descent Methods of Optimization. These methods are the Gradient Descent method and the Conjugate Gradient method.

In Chapter Three, we study the Newton-Like methods for non-linear Optimization. It begins with the Newton's method and then looks at various modifications that have been suggested to this method. Next, the chapter looks at the Quasi-Newton methods and then the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method.

Chapter Four studies the methods for Constrained Optimization. These methods are the Lagrange's method which involve equality constraints and the Kuhn-Tucker method which involve inequality constraints. In this chapter, methods for solving Constrained minimization problems are discussed. These are the Sequential Quadratic Programming methods, the Penalty functions and Barrier functions.

In Chapter Five, we make a summary of all the various observations that have emerged from the study. We then discuss some of these observations. Finally, we draw appropriate conclusions and make few recommendations based on the result of the study.

CHAPTER TWO

UNCONSTRAINED OPTIMIZATION

In this chapter, we consider methods for solving optimization problems of the form

$$\text{minimize } f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^n. \tag{2.1}$$

We assume that f is *smooth*, that is, the first and second partial derivatives of f exists and are continuous. The derivative of a function contains information about the rates of increase or decrease of the function; in the case of a function of several variables, the partial derivatives also give the directions of fastest increase or decrease. If such information about the objective function is known, then it can be used to determine its optimum value more efficiently. We will focus attention on a class of Unconstrained Optimization techniques that use line search algorithms for solving minimization problems.

Gradient Descent Method

The **gradient descent** method is also known as the method of **steepest descent**. Gradient descent is based on the observation that if the real-valued function $f(\mathbf{x})$ is defined and differentiable in a neighborhood of a point $\mathbf{x_k}$, then $f(\mathbf{x})$ decreases fastest in the direction of the negative gradient of f at $\mathbf{x_k}$, that is, in the direction of $-\nabla f(\mathbf{x_k})$. The search starts with an initial guess $\mathbf{x}_0$ and then slide down the gradient, until we are close enough to the solution and that computes the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ such that

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \; k = 0, 1, 2, \cdots .$$

We have

$$f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq f(\mathbf{x}_2) \geq \ldots,$$

so hopefully the sequence $\{\mathbf{x}_k\}$ converges to the desired local minimum. In other words, the iterative procedure is

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha_k \nabla f(\mathbf{x_k}) \tag{2.2}$$

for $\alpha > 0$ which is a small enough number and $\nabla f(\mathbf{x}_k)$ is the gradient at one given point. Now, the question is, how big should the step size α_k be taken in that direction, i.e. what should be the size of α_k? Obviously, we want to move to the point where the function f takes on a minimum value, i.e. where the directional derivative is zero. Using the chain rule, the directional derivative of $f(\mathbf{x}_{k+1})$ is given by

$$\frac{d}{d\alpha_k} f(\mathbf{x}_{k+1}) = \nabla f(\mathbf{x}_{k+1})^T . \frac{d}{d\alpha_k} . \mathbf{x}_{k+1} = -\nabla f(\mathbf{x}_{k+1})^T \nabla f(\mathbf{x}_k). \tag{2.3}$$

Setting Equation 2.3 to zero shows that α_k must be chosen so that $\nabla f(\mathbf{x}_{k+1})$ and $\nabla f(\mathbf{x}_k)$ are orthogonal, i.e.

$$\nabla f(\mathbf{x}_{k+1})^T \cdot \nabla f(\mathbf{x}_k) = 0$$

The next step is then taken in the direction of the negative gradient at this new point and we get a zig-zag pattern as illustrated in Figure 5. This iteration continues until the minimum has been determined within a chosen accuracy ϵ. This is actually a minimization problem along a line and this is known as an exact line search algorithm.

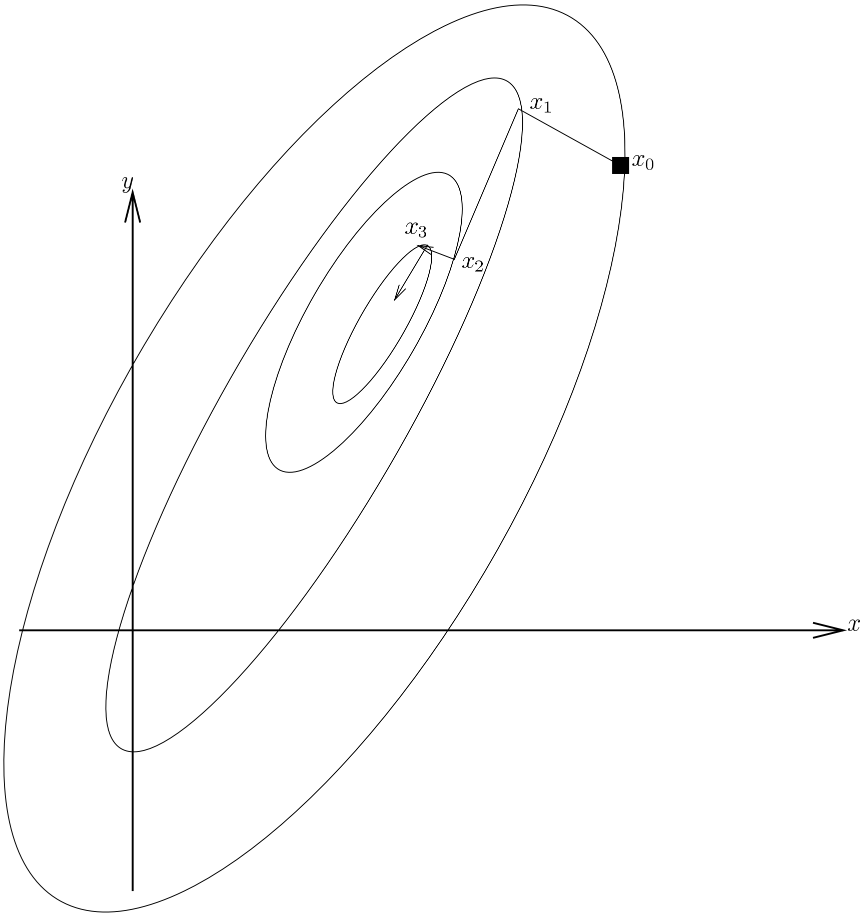

Figure 5: Geometric representation of the Gradient Descent method

Alternatively, one can start with a chosen value for α_k, which, if necessary, will be modified during the iterations, making sure that the function decreases at each iteration. This works better than the line search and is a bit simpler. This will take many more iterations to reach the minimum, but each iteration may take much less time than by using a line search.

The following example illustrates how to determine the value of the step size α.

Example 2.1. To find the value of α which minimizes the function

$$\begin{aligned} f(x,y,z) &= 2x^2-2x+y^2-4y+z^2-6z+13 \\ \alpha_0 &= \min_{\alpha>0}\{f((x_0,y_0,z_0)^T-\alpha\nabla f(x_0,y_0,z_0)^T)\} \end{aligned}$$

$$\nabla f = \begin{bmatrix} 4x-2 \\ 2y-4 \\ 2z-6 \end{bmatrix}$$

Let $[x_0; y_0; z_0] = [1; 1; 1]$

$$\nabla f = \begin{bmatrix} 2 \\ -2 \\ -4 \end{bmatrix}$$

$$\begin{aligned} \alpha_0 &= \min_{\alpha>0}\{f((x_0, y_0, z_0)^T - \alpha\nabla f(x_0, y_0, z_0)^T)\} \\ &= \min_{\alpha>0}\{f((1,1,1)^T - \alpha(2,-2,-4)^T)\} \\ &= \min_{\alpha>0}\{f \begin{bmatrix} 1-2\alpha \\ 1+2\alpha \\ 1+4\alpha \end{bmatrix}\} \\ &= \min_{\alpha>0}\{5 - 24\alpha + 28\alpha^2\} \end{aligned}$$

Therefore the value of α which minimizes f is $\frac{3}{7}$.

However, the method of steepest descent is only linearly convergent, despite its use of derivative information. It is often used as a “starter” method for other algorithms. Because it converges for most initial guesses, several iterations of steepest descent may be used to improve an initial guess that is then handed to another, faster(but less robust) algorithm. This is done not only for minimization algorithms but also for root - finding algorithms such as Newton’s method for systems. Suppose we attempt to solve $f(\mathbf{x}) = 0$ where $f : \mathbb{R}^n \to \mathbb{R}^n$, that is,

$$f(\mathbf{x}) = \begin{bmatrix} f_1(x_1, \cdots, x_n) \\ f_2(x_1, \cdots, x_n) \\ \vdots \\ f_n(x_1, \cdots, x_n) \end{bmatrix}$$

by Newton's method for systems or some other root- finding method with some initial guess $\mathbf{x}_0$. If we find that the method fails to converge, then we can use the method of steepest descent on the function

$$\|f(\mathbf{x})\|^2 = f_1^2(x) + \cdots + f_n^2(x)$$

(which is nonnegative and has minima precisely where f has zeros) starting from this $\mathbf{x}_0$. After several iterations of steepest descent, we return to Newton's method with what we hope is an improved initial guess.

Example 2.2. Consider the problem of minimizing a function

$$(1 - x_1)^2 + x_2^2$$

using the steepest descent method.

Using Equation 2.2, with starting point $\mathbf{x}_0 = [0; 0]$. We get the following iterates

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 - \ \alpha_0 \nabla f(\mathbf{x}_0) \\
\nabla f(\mathbf{x}_0) &= [-2; 0] \\
\alpha_0 &= \min_{\alpha > 0} \{ f((\mathbf{x}_0)^T - \alpha \nabla f(\mathbf{x}_0)^T) \} \\
&= \min_{\alpha > 0} \{ f((0,0)^T - \alpha(-2,0)^T) \} \\
&= \min_{\alpha > 0} \{ f(2\alpha, 0) \} \\
&= \min_{\alpha > 0} \{ (1 - 2\alpha)^2 \}
\end{aligned}$$

$$\Rightarrow (1 - 2\alpha) = 0$$

$$\Rightarrow \alpha_0 = 0.5$$

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0) \\
&= [0;0] - 0.5[-2;0] \\
&= [1;0] \\
\mathbf{x}_2 &= \mathbf{x}_1 - \alpha_1 \nabla f(\mathbf{x}_1) \\
\nabla f(\mathbf{x}_1) &= [0;0] \\
&= \min_{\alpha>0}\{f((1,0)^T - \alpha_1(0,0)^T)\} \\
&= \min_{\alpha>0}\{f(1,0)\}
\end{aligned}$$

$$\Rightarrow \alpha_1 = 0$$

$$\mathbf{x}_2 = [1;0]$$

Thus, the iterations converge to a critical point $[1;0]$ just after one iteration. See Figure 11 in the Appendix for the graph of the given function. Now to confirm whether the point is a minimum or not, we evaluate the Hessian at that point. This gives

$$\nabla^2 f[1;0] = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

which is positive definite and that is the condition for a critical point to be minimum. Another reason for converging very fast is because the condition number of the Hessian matrix is 1. This is not surprising because as the condition number becomes very large, it affect the rate of convergence, thereby taking a lot of iterations for it to converge to a required solution. The effect of the condition number on the rate of convergence to a critical point will be explained later in this chapter.

Example 2.3. Considering the problem of minimizing the function

$$f(x_1, x_2) = (x_2 - x_1^2)^2 + (1 - x_1)^2$$

using the steepest descent algorithm.

The algorithm is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$$

Let $\mathbf{x}_0 = [0; 0]$, then the first iterate is

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0) \\
\nabla f(\mathbf{x}_0) &= [-2; 0] \\
\alpha_0 &= \min_{\alpha>0}\{f(x_0, y_0)^T - \alpha \nabla f(x_0, y_0)^T\} \\
&= \min_{\alpha>0}\{f(0, 0)^T - \alpha(-2, 0)^T\} \\
&= \min_{\alpha>0}\{f(2\alpha, 0)\} \\
&= \min_{\alpha>0}\{16\alpha^4 + (1 - 2\alpha)^2\} \\
&= \min_{\alpha>0}\{16\alpha^4 + 4\alpha^2 - 4\alpha + 1\} \\
&\Rightarrow 64\alpha^3 + 8\alpha - 4 = 0 \\
&\Rightarrow 16\alpha^3 + 2\alpha - 1 = 0
\end{aligned}$$

Solving for α gives 0.29488. Now to get the first iterate $\mathbf{x}_1$, we substitute $\alpha_0 = 0.29488$ into the method's algorithm. That is,

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0) \\
&= [0; 0] - 0.29488[-2; 0] \\
&= [0.58976; 0]
\end{aligned}$$

The vector $[0.58976; 0]$ is the first iterate and it also serves as the new initial guess for the next iteration. At each iteration, the value of α has to be calculated. Now, to get the next iterate $\mathbf{x}_2$, we follow the same algorithm. That is,

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha_1 \nabla f(\mathbf{x}_1)$$

We repeat the procedure until we attain a minimum. After going through three iterations, we realized that the solution converges to the minimum $[1; 1]$. A summary of it is given in the table below.

Table 1: Number of iterations and the step size at each iterate

No. of iteration	α	$\mathbf{x}_k$
0	0.29488	[0; 0]
1	0.49995	[0.58976; 0]
2	0.13169	[0.58974; 0.34778]
3	0.13269	[0.99507; 0.98847]

The iterates converge after three iterations. This is because the Hessian matrix of the function is well-conditioned with a small condition number equal to 1.

Example 2.4. Consider the problem of minimizing the function

$$100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

using the steepest descent method. The function above is known as the Rosenbrock's function sometimes also called the Banana function. The nature of the Rosenbrock's function makes it very difficult to get to the minimum point. The reason is that the minimum point lies in a long valley. It is therefore used to test the robustness of unconstrained minimization algorithms. Using Equation 2.2 with a starting point $\mathbf{x}_0 = [0; 0]$, we get the following iterates.The table below shows some iterates in the application of the steepest descent to the function.

Table 2: Number of iterations and the condition number of the Hessian of the Rosenbrocks function

Iteration	$\mathbf{x}$	Condition No. of Hessian
0	[0; 0]	100
10	[0.310945; 0.085604]	60.578
100	[0.58593; 0.33975]	331.81
1000	[0.92741; 0.85909]	1653.2

The iterates are converging to the desired solution at a very slow rate. The slow rate of convergence can be attributed to the ill-conditioning of the Hessian of the objective function. We note that because the condition number of the Hessian matrix of the function is 100, it makes the matrix ill-conditioned.The condition number is the ratio of the largest eigenvalue to the smallest eigenvalue. If this ratio is large, it makes it difficult for the algorithm to converge to a solution.

Example 2.5. Consider the problem of minimizing the function

$$f(\mathbf{x}) = \lambda_1(x_1 - 3)^2 + \lambda_2(x_2 - 2)^2.$$

Suppose that $\lambda_1 = 1000$ and $\lambda_2 = 1$. It is clear by inspection that the optimal value is $\mathbf{x}^* = [3; 2]$ and that the value of the objective function at this point is $f(\mathbf{x}^*) = 0$. With an initial guess of $\mathbf{x}_0 = [0; 0]$, the steepest descent method converges to

$\mathbf{x} = [3.0000000;\ 1.9999994]$ after 302 iterations and 10,273 scalar function evaluations. Suppose $\lambda_1 = 1000$ is changed to $\lambda_1 = 1$, the steepest descent method converges in just two iterations. To see why this makes such a difference, we consider the condition number of the Hessian matrix which is given as

$$\begin{aligned} \kappa(\mathbf{H}) &= \|\mathbf{H}\| \cdot \|\mathbf{H}^{-1}\| \\ &= \max\{|\lambda_1|,\ |\lambda_2|\} \max\{|\lambda_1^{-1}|,\ |\lambda_2^{-1}|\} \\ &= \left(\frac{|\lambda_1|}{|\lambda_2|}\right) \end{aligned}$$

Thus, when $\lambda_1 = 1000$ and $\lambda_2 = 1$, we have $\kappa(\mathbf{H}) = 1000$, which makes the Hessian matrix ill-conditioned. Conversely, when $\lambda_1 = 1$ and $\lambda_2 = 1$, the Hessian matrix is well conditioned with $\kappa(\mathbf{H}) = 1$.

In general, when the Hessian matrix is ill-conditioned, the steepest descent method converges very slowly.

From Example 2.4, it was clear that the steepest descent algorithm fails to find the minimum in a possible shortest time thereby making the method insufficient. The major drawbacks in this method identified are as follows:

1. The algorithm can take many iterations to converge towards a local minimum.

2. Finding the optimal α per step can be time-consuming. Conversely, using a fixed α can yield poor results.

3. The rate of convergence can be very slow if the Hessian of the objective function is ill-conditioned.

Improvement on the weaknesses of the steepest descent method gives rise to another method known as the Newton's method. The Newton's method is examined in the next chapter.

The Rate of Convergence for the Case of a Quadratic Function

A simple line search descent method is the steepest descent method in which the search direction $\mathbf{s}_k = -\nabla f(\mathbf{x}_k)$ for all k. In practice, the method usually exhibits oscillatory behaviour which usually terminates far from the solution owing to round-off effects. Therefore, making the method inefficient and unreliable. The local convergence result for steepest descent does predict the possibility of an arbitrarily slow rate of linear convergence. This inadequacy of the steepest descent method can be put down to a failure in the model situation, perhaps because the steepest descent property along the line holds only at $\alpha = 0$ and not for all α. A type of model which in practice does usually give rise to methods with a rapid rate of convergence is the *quadratic model*. One possible reason is that this model is to some extent associated with the property of second order convergence. Descent methods based on a quadratic model have proved to be very powerful in practice.

Theorem 2.1. An algorithm exhibits *linear convergence* in the objective function values if there is a constant $\boldsymbol{\delta} < 1$ such that for all k sufficiently large, the iterates $\mathbf{x}_k$ satisfy:

$$\frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*)}{f(\mathbf{x}_k) - f(\mathbf{x}^*)} \leq \boldsymbol{\delta},$$

where $\mathbf{x}^*$ is some optimal value of the problem.

The statement above suggests that the optimality gap shrinks by at least $\boldsymbol{\delta}$ at each iteration, that is, if $\boldsymbol{\delta} = 0.1$, for example, then the iterates gain an extra digit of accuracy in the optimal objective function value at each iteration. This effect then speeds up the rate of convergence. On the other hand, if $\boldsymbol{\delta} = 0.9$, which is very close to 1, it slows down the rate of convergence. The quantity $\boldsymbol{\delta}$ is called the *convergence constant*. Our aim is to make the constant smaller rather than larger. The convergence constant depends very much on the condition number of the Hessian matrix $\mathbf{H}(\mathbf{x})$, that is, the ratio of the largest to the smallest eigenvalue of the Hessian matrix at the optimal solution $\mathbf{x}^*$. Lets consider the case where the objective function $f(\mathbf{x})$ is itself a simple quadratic function of the form:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x}$$

where $\mathbf{A}$ is a positive definite symmetric matrix. We will suppose that the eigenvalues of $\mathbf{A}$ are

$$\lambda_1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = \lambda_n > 0,$$

that is, λ_1 and λ_n are the largest and the smallest eigenvalues of $\mathbf{A}$ respectively. The optimal solution of the problem is computed as:

$$\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{b}$$

and by direct substitution shows that the optimal objective function value is:

$$f(\mathbf{x}^*) = -\frac{1}{2}\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}.$$

Let $\mathbf{x}_k$ denote the current point in the steepest descent algorithm and let $\mathbf{d}_k$ denote the current direction, which is the negative of the gradient, that is,

$$\mathbf{d}_k = -\nabla f(\mathbf{x}_k) = -\mathbf{A}\mathbf{x}_k - \mathbf{b}.$$

To get the next iterate of the steepest descent algorithm, we compute the step size, α, that is,

$$\begin{aligned} f(\mathbf{x}_k + \alpha\mathbf{d}_k) &= \frac{1}{2}(\mathbf{x}_k + \alpha\mathbf{d}_k)^T\mathbf{A}(\mathbf{x}_k + \alpha\mathbf{d}_k) + \mathbf{b}^T(\mathbf{x}_k + \alpha\mathbf{d}_k) \\ &= \frac{1}{2}\mathbf{x}_k^T\mathbf{A}\mathbf{x}_k + \alpha\mathbf{d}_k^T\mathbf{A}\mathbf{x}_k + \frac{1}{2}\alpha^2\mathbf{d}_k^T\mathbf{A}\mathbf{x}_k + \mathbf{b}^T\mathbf{x}_k + \alpha\mathbf{b}^T\mathbf{d}_k \\ &= f(\mathbf{x}_k) - \alpha\mathbf{d}_k^T\mathbf{d}_k + \frac{1}{2}\alpha^2\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k. \end{aligned}$$

Optimizing the value of α in this last expression yields

$$\alpha = \frac{\mathbf{d}_k^T\mathbf{d}_k}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k},$$

and the next iterate of the algorithm is then

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}_k = \mathbf{x}_k + \frac{\mathbf{d}_k^T\mathbf{d}_k}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}\mathbf{d}_k,$$

and

$$\begin{aligned} f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k + \alpha\mathbf{d}_k) &= f(\mathbf{x}_k) - \alpha\mathbf{d}_k^T\mathbf{d}_k + \frac{1}{2}\alpha^2\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k \\ &= f(\mathbf{x}_k) - \frac{1}{2}\frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*)}{f(\mathbf{x}_k) - f(\mathbf{x}^*)} &= \frac{f(\mathbf{x}_k) - \frac{1}{2}\frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k} - f(\mathbf{x}^*)}{f(\mathbf{x}_k) - f(\mathbf{x}^*)} \\ &= 1 - \frac{\frac{1}{2}\frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}}{\frac{1}{2}\mathbf{x}_k^T\mathbf{A}\mathbf{x}_k + \mathbf{b}^T\mathbf{x}_k + \frac{1}{2}\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}} \\ &= 1 - \frac{\frac{1}{2}\frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}}{\frac{1}{2}(\mathbf{A}\mathbf{x}_k + \mathbf{b})^T\mathbf{A}^{-1}(\mathbf{A}\mathbf{x}_k + \mathbf{b})} \\ &= 1 - \frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{(\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k)(\mathbf{d}_k^T\mathbf{A}^{-1}\mathbf{d}_k)} \\ &= 1 - \frac{1}{\beta} \end{aligned}$$

where

$$\beta = \frac{(\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k)(\mathbf{d}_k^T \mathbf{A}^{-1} \mathbf{d}_k)}{(\mathbf{d}_k^T \mathbf{d}_k)^2}$$

The following inequality provides an upper bound on the value of β.

Theorem 2.2. Kantorovich Inequality

Let λ_1 and λ_n be the largest and the smallest eigenvalues of $\mathbf{A}$, respectively. Then

$$\beta \leq \frac{(\lambda_1 + \lambda_n)^2}{4\lambda_1 \lambda_n}.$$

Now, applying this inequality to the above analysis, gives

$$\begin{aligned} \frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*)}{f(\mathbf{x}_k) - f(\mathbf{x}^*)} &= 1 - \frac{1}{\beta} \\ &\leq 1 - \frac{4\lambda_1\lambda_n}{(\lambda_1 + \lambda_n)^2} \\ &= \frac{(\lambda_1 - \lambda_n)^2}{(\lambda_1 + \lambda_n)^2} \\ &= \left(\frac{\frac{\lambda_1}{\lambda_n} - 1}{\frac{\lambda_1}{\lambda_n} + 1} \right)^2 \\ &=: \delta. \end{aligned}$$

By definition, $\frac{\lambda_1}{\lambda_n}$ is always at least 1. If $\frac{\lambda_1}{\lambda_n}$ is small, that is, not much bigger than 1, then the convergence constant δ will be much smaller than 1. However, if $\frac{\lambda_1}{\lambda_n}$ is large, then the convergence constant δ will be only slightly smaller than 1. The reason for ensuring that we obtain a small value of the ratio $\frac{\lambda_1}{\lambda_n}$ is to reduce the condition number of the Hessian matrix, thereby speeding up the rate of convergence. We illustrate this with an example.

Example 2.6. Consider the problem of minimizing the function

$$100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

with a starting point of $\mathbf{x}_0 = [0; 0]$, then $\nabla f(\mathbf{x}_0) = [-2; 0]$. The descent direction $\mathbf{d} = -\nabla f(\mathbf{x}_0) = [2; 0]$. The table below gives the Sensitivity of Steepest Descent Convergence Rate to the Eigenvalue Ratio.

Table 3: Sensitivity of Steepest Descent Convergence Rate to the Eigenvalue Ratio

λ_1	λ_n	Upper Bound on δ	k	Condition No.
237.21622	7.97875	0.87407	2	29.731
279.17358	4.60848	0.93609	10	60.578
476.63669	1.43646	0.98800	100	331.81
744.95842	0.66666	0.99634	500	1117.4
889.92511	0.53830	0.99757	1000	1653.2

From Table 3, the condition number of the Hessian matrix keeps on increasing as we increase the number of iteration. This is because as the condition number gets large, the Hessian matrix becomes ill-conditioned and this is embedded in the problem. The table also shows the relationship between the condition number and the upper bound on δ on the convergence rate. We see that when the condition number is small, the upper bound on δ is also small both of which ensure faster rate of convergence (i.e. small value of k). On the other hand, when the condition number is large the upper bound on δ is correspondingly high, leading to a very slow convergence rate (i.e. large value of k).

We compare the Rosenbrock's function with the modified Rosenbrock's function given by

$$(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

The graphs of the two functions within the interval
$[X, Y] = (-2 : 0.1 : 2, -2 : 0.1 : 2)$ are given in Figures 6 and 7 respectively.

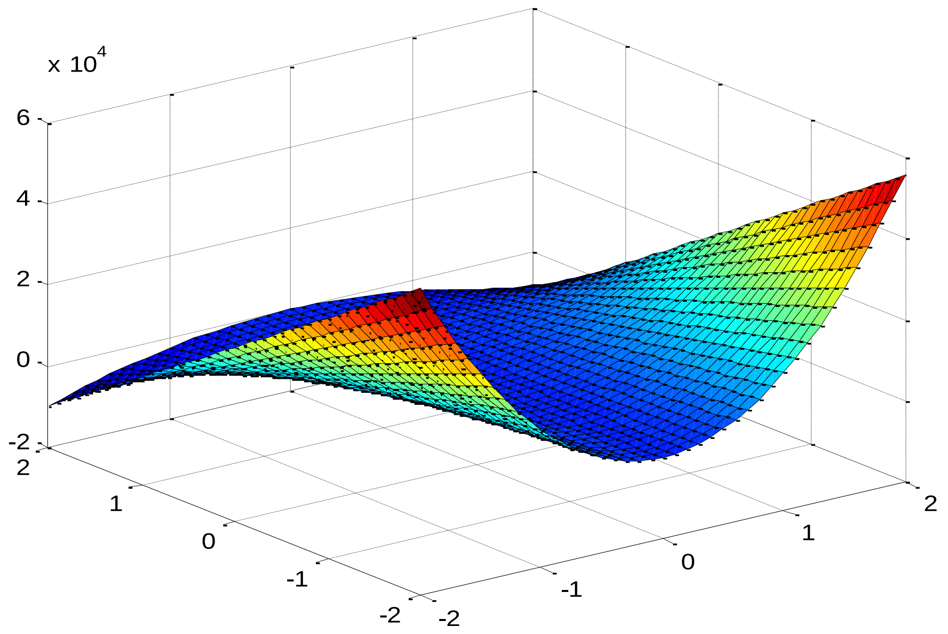

Figure 6: The graph of the Rosenbrock's function in the interval $[X,Y]=(-2:0.1:2,\,-2:0.1:2)$

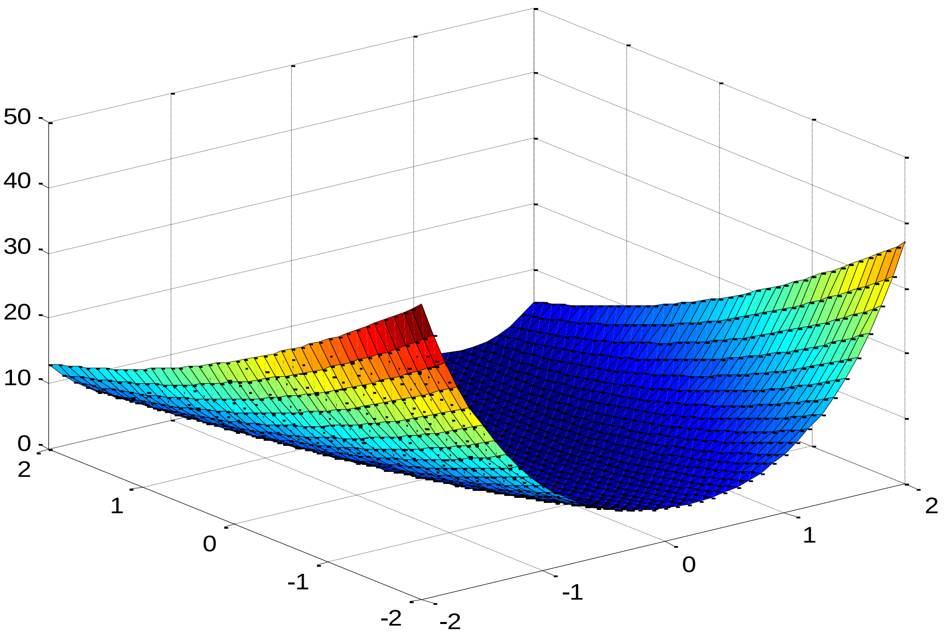

Figure 7: The graph of the Modified Rosenbrock's function in the interval $[X,Y]=(-2:0.1:2,\,-2:0.1:2)$

We observe that within the specified interval, the base of the Modified Rosenbrock's function is more clearly defined than that of the Rosenbrock's function itself. This suggests that we could obtain a faster convergence rate for the Modified Rosenbrock's than the Rosenbrock's function. The graphs of the two functions within the interval $[X, Y] = (-5 : 0.1 : 5, -5 : 0.1 : 5)$ are given in Figures 12 and 13 in the Appendix.

The differences in the bases of the two graphs are as a result of wide differences between the largest and the smallest eigenvalues of the Hessian matrix, $\mathbf{H}(\mathbf{x}_k)$ of the Rosenbrock's function as compared to the Modified Rosenbrock's function. We examine the behaviour of the two functions more analytically. In Table 4, the first column shows the number of iterations, the second column shows the condition number of the Rosenbrock's function and the third column shows the condition number for the Modified Rosenbrock's function.

Table 4: The condition number of the Rosenbrock's and the Modified Rosenbrock's functions at various iterations

k	κ(Rosenbrock)	κ(Modified Rosenbrock)
2	29.731	12.710
10	60.578	25.240
100	331.81	33.962
500	1117.4	33.962
1000	1653.2	33.962

From Table 4, it can be seen that as the condition number of the Rosenbrock's function increases, that of the Modified Rosenbrock's function remains constant for a number of iterations. As a result the Modified Rosenbrock's function is able to converge in less than 100 number of iterations whiles the Rosenbrock's function exhibit a slow rate of convergence.

The results in Table 4 illustrate a major weakness of the Steepest Descent method: it is not an appropriate method when the Hessian matrix is ill-conditioned.

Conjugate Gradient Method

Conjugate gradient methods are widely used for large scale unconstrained optimization problems. Most of conjugate gradient methods do not always generate a descent search direction, so the descent condition is usually assumed in the analysis as implementation.

The conjugate gradient method is an algorithm for the numerical solution of particular systems of linear equations, namely those whose matrix is symmetric and positive definite. The conjugate gradient method is an iterative method which terminates in at most n steps if no rounding-off errors are encountered, therefore it can be applied to sparse systems which are too large to be handled by direct methods such as the Cholesky decomposition. Such systems arise regularly when numerically solving partial differential equations.

The main aim of conjugate gradient method is to associate conjugacy properties with the steepest descent method in an attempt to achieve both efficiency and reliability. The conjugate gradient minimization method of Fletcher and Reeves (1964) was developed directly from the conjugate gradient method of Hestenes and Stiefel (1952) for solving linear systems.

The conjugate gradient method were developed for the purpose of solving

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

when $\mathbf{A}$ is positive definite and they are also used for minimizing

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^n.$$

Definition 2.1. Given a positive definite matrix $\mathbf{A}$, two vectors $\mathbf{u}$, $\mathbf{v}$ are **conjugate with respect to A** (or $\mathbf{A}$-conjugate) if

$$\mathbf{u}^T\mathbf{A}\mathbf{v} = 0$$

which is an inner product $(\mathbf{u}, \mathbf{v})$ defined by the matrix $\mathbf{A}$. This inner product in turn induces a vector norm $\|\mathbf{u}\|_{\mathbf{A}} = \left(\mathbf{u}^T\mathbf{A}\mathbf{u}\right)^{\frac{1}{2}}$, called the $\mathbf{A}$-norm.

If the vectors $\mathbf{d}_1, \cdots, \mathbf{d}_n$ are nonzero and pairwise $\mathbf{A}$-conjugate, then they are linearly independent, and hence form a basis for $\mathbb{R}^n$. Then $\{\mathbf{d}_1, \cdots, \mathbf{d}_n\}$ is said to be the set of **conjugate directions**(with respect to $\mathbf{A}$). Given a general quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^n,$$

$$\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$$

at any point. We choose the starting point and an initial direction to be the direction of steepest descent

$$\begin{aligned} \mathbf{d}_0 &= -\nabla f(\mathbf{x}_0) \\ &= \mathbf{b} - \mathbf{A}\mathbf{x}_0 \end{aligned}$$

Starting with an initial guess $\mathbf{x}_0$, we choose α_0 to minimize $f(\mathbf{x}_0 + \alpha_0\mathbf{d}_0)$ since $\mathbf{d}_0$ is a descent direction.

We then have

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0\mathbf{d}_0.$$

In general, we compute these at each iteration:

$$\begin{aligned} \mathbf{r}_k &= \mathbf{b} - \mathbf{A}\mathbf{x}_k \\ \alpha_k &= \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k} \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k\mathbf{d}_k \\ \beta_k &= \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_{k-1}^T\mathbf{r}_{k-1}} \\ \mathbf{d}_k &= \mathbf{r}_k + \beta_k\mathbf{d}_{k-1} \end{aligned}$$

The algorithm outlined is illustrated in Example 2.7.

Example 2.7. Let

$$f(\mathbf{x}) = -\mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x},$$

$\mathbf{A} = [3\ \ -1; -1\ \ 3]$ and $\mathbf{b} = [4; -2]$.

$$\nabla f = -\mathbf{b} + \mathbf{A}\mathbf{x}$$

$$-\nabla f = \mathbf{b} - \mathbf{A}\mathbf{x}$$

At $\mathbf{x} = [0; 0]$, $-\nabla f = \mathbf{r}_0$ which gives the initial direction.

$$\alpha_k = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}$$

$$\alpha_0 = \frac{\mathbf{r}_0^T\mathbf{r}_0}{\mathbf{d}_0^T\mathbf{A}\mathbf{d}_0}$$

where $\mathbf{r}_0$ is the same as the initial direction $\mathbf{d}_0$.

$$\alpha_0 = \frac{[4\ \ -2]\begin{bmatrix}4\\-2\end{bmatrix}}{[4\ \ -2]\begin{bmatrix}3 & -1\\-1 & 3\end{bmatrix}\begin{bmatrix}4\\-2\end{bmatrix}}$$

$$\alpha_0 = \frac{20}{76} = \frac{5}{19}$$

$$\mathbf{x}_1 = (\mathbf{x}_0 + \alpha_0\mathbf{d}_0) = \frac{5}{19}\begin{bmatrix}4\\-2\end{bmatrix} = \begin{bmatrix}1.0526\\-0.5263\end{bmatrix}$$

$$\mathbf{r}_1 = \mathbf{b} - \mathbf{A}\mathbf{x}_1 = \begin{bmatrix}4\\-2\end{bmatrix} - \begin{bmatrix}3 & -1\\-1 & 3\end{bmatrix}\begin{bmatrix}1.0526\\-0.5263\end{bmatrix} = \begin{bmatrix}0.3159\\0.6315\end{bmatrix}$$

$$\mathbf{d}_1 = \mathbf{r}_1 - \frac{\mathbf{r}_1^T\mathbf{A}\mathbf{r}_1}{\mathbf{r}_0^T\mathbf{A}\mathbf{r}_0}\mathbf{d}_0$$

$$\mathbf{d}_1 = \begin{bmatrix} 0.3159 \\ 0.6315 \end{bmatrix} - \frac{[0.3159 \;\; 0.6315] \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 0.3159 \\ 0.6315 \end{bmatrix}}{[4 \;\, -2] \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ -2 \end{bmatrix}} \begin{bmatrix} 4 \\ -2 \end{bmatrix}$$

$$\mathbf{d}_1 = \begin{bmatrix} 0.3159 \\ 0.6315 \end{bmatrix} - 0.0249 \begin{bmatrix} 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 0.4155 \\ 0.5817 \end{bmatrix}$$

$$\alpha_1 = \frac{\mathbf{r}_1^T \mathbf{r}_1}{\mathbf{d}_1^T \mathbf{A} \mathbf{d}_1}$$

$$\alpha_1 = \frac{[0.3159 \;\; 0.6315] \begin{bmatrix} 0.3159 \\ 0.6315 \end{bmatrix}}{[0.4155 \;\; 0.5817] \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 0.4155 \\ 0.5817 \end{bmatrix}}$$

$$\alpha_1 = \frac{0.4986}{1.0497} = 0.4750$$

$$\mathbf{x}_2 = (\mathbf{x}_1 + \alpha_1 \mathbf{d}_1) = \begin{bmatrix} 1.0526 \\ -0.5263 \end{bmatrix} + 0.475 \begin{bmatrix} 0.4155 \\ 0.5817 \end{bmatrix} = \begin{bmatrix} 1.25 \\ -0.25 \end{bmatrix}$$

We see that the iterates converge after only two iterations. This is so because the function involved in the implementation is quadratic. If the function is not quadratic it may take more number of iterations for the iterate to converge.

The general m-file implementation of the algorithm is given as follows:

function $y = cgm(\mathbf{A}, \mathbf{b}, \mathbf{x}_0)$

$n = size(\mathbf{x}_0)$

$\mathbf{r}_0 = \mathbf{b} - (A * \mathbf{x}_0)$

$\mathbf{p}_0 = \mathbf{r}_0;$

$for\ k = 1 : n$

```
$a_{k-1} = (\mathbf{r}'_{k-1} * \mathbf{r}_{k-1})/(\mathbf{d}'_{k-1} * \mathbf{A} * \mathbf{d}_{k-1})$
$\mathbf{x}_k = \mathbf{x}_{k-1} + a_{k-1} * \mathbf{d}_{k-1}$
$\mathbf{r}_k = \mathbf{r}_{k-1} - a_{k-1} * (\mathbf{A} * \mathbf{d}_{k-1})$
$b_k = (\mathbf{r}'_k * \mathbf{r}_k)/(\mathbf{r}'_{k-1} * \mathbf{r}_{k-1})$
$\mathbf{d}_k = \mathbf{r}_k + b_k * \mathbf{d}_{k-1}$
end
$\mathbf{x}$
end
```

The algorithm outlined is illustrated in Example 2.9.

Example 2.8. Consider the problem of minimizing the function

$$f(x_1, x_2) = (x_2 - x_1^2)^2 + (1 - x_1)^2$$

using the conjugate gradient method with a starting point of $\mathbf{x}_0 = [0; 0]$, we get the following iterates:

Table 5: Values of function and gradient at each iterate

Iteration	$\mathbf{x}_k$	$\nabla f(\mathbf{x}_k)$	$f(\mathbf{x}_k)$
0	[0; 0]	[-2; 0]	1
2	[0.95504; 0.89219]	[-0.013851; -0.039824]	0.0024178
4	[0.99947; 0.99872]	[-1.9024 e-004; -4.3446 e-004]	3.2740 e-007
6	[1; 1]	[-3.9023 e-007; -9.3200 e-007]	1.4880 e-012

From Table 5, we observe that $\nabla f(\mathbf{x}_k)$ steadily approaches zero and the function evaluations at $\mathbf{x}_k$ is decreasing. These two indicate that the point of convergence is truly one of a minimum. The table shows that the iterates converge after six iterations using the MATLAB implementation of Davidon-Fletcher-Powell algorithm with self-scaling for solving an n-dimensional unconstrained minimization problem.

We see from Table 5 that since the function is not quadratic it takes more than two iterations for convergence. This illustrates the general case when the function involved is not quadratic.

Example 2.9. Consider the problem of minimizing the function

$$f(x_1, x_2) = x_1^4 + x_1 x_2 + (1 + x_2)^2$$

using the conjugate gradient method with a starting point of $\mathbf{x}_0 = [0; 0]$, we obtain the first iterate to be $\mathbf{x}_1 = [0.69588;\ -1.34794]$ and that is the minimum of the function. The solution is quick. This suggests that the function exhibits a lot of quadratic properties. We see these properties in the graph of f illustrated in Figures 14a and 14b in the Appendix. In fact, it can be seen that the function is a quadratic in x_2.

CHAPTER THREE

NEWTON-LIKE METHODS OF UNCONSTRAINED OPTIMIZATION

Newton's Method

Many methods for solving minimization problems are variants of Newton method, which requires the specification of the Hessian matrix of second derivatives. A quadratically convergent method can be attained by using Newton's method to perform root-finding on $f'(x)$, that is $f'(x) = 0$. If $f : \mathbb{R} \to \mathbb{R}$, then Newton's method applied to $f'(x)$ can be written as

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{3.1}$$

if f is twice continuously differentiable. Since we are dealing with f', the convergence theory requires that f be four times continuously differentiable such that f' will be thrice continuously differentiable. Let assume that x_0 is sufficiently close to a minimum x^* of f and that $f''(x^*)$ is nonzero, then we expect the **quadratic convergence** of the method to x^*. Note that if x_0 is not sufficiently close to a minimum, then it converges to a maximum of f. If x_0 is an approximation of the location of the minimum, then near x_0 with the idea of Taylor's Theorem, we have

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2 \tag{3.2}$$

that is, near x_0, $f(x)$ is approximated by the quadratic model

$$q(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2.$$

The minimum of this quadratic occurs where $q'(x) = 0$. Differentiating Equation 3.2 with respect to x and setting $f'(x)$ to zero gives

$$0 \approx 0 + f'(x_0) \cdot 1 + f''(x_0)(x - x_0)$$

Noting that $f'(x_0)$ and $f''(x_0)$ are constants, this leads to the Newton's method

$$x_1 = \mathbf{x}_0 - \frac{f'(x_0)}{f''(x_0)}.$$

Hence, if x_0 is an approximation of the location of the minimum x^*, then x_1 should be a better approximation of x^*. Therefore, Newton's method for minimization is given in Equation 3.1 and is truly an optimization method based on Newton's method for root - finding.

Now, to choose a good initial guess x_0 for this method, it depends on whether

1. $f'(x_0) = 0$ and

2. $f''(x_0) > 0$

The choice of an initial point(x_0) is crucial to obtaining the type of critical point that is desired. If x_0 is chosen close to a minimum point, it leads to a minimum point. On the other hand, if x_0 is chosen close to a maximum point, we automatically end up with a maximum point. Thus, if we seek a minimum point, but x_0 is taken close to a maximum point, the resulting iterates become divergent from the desired minimum point. This phenomenon is a particular feature of the Newton's method.

Example 3.1. Consider the problem of minimizing the function

$$f(x) = (x - 5)(x - 12)(x - 30)$$

using the Newton's method.

From the graph in Figure 8, it is clear that the only local minimum of this cubic function lies between $x = 30$ and $x = 40$.

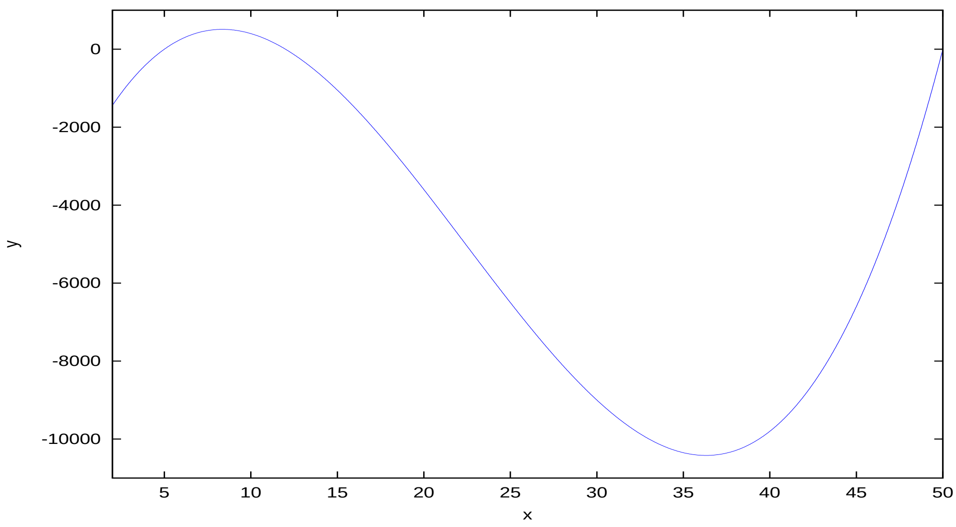

Figure 8: The graph of $f(x) = x^3 - 47x^2 + 570x - 1800$

With initial guess $x_0 = 30$, we have

$$\begin{aligned}
x_1 &= x_0 - \frac{f'(x_0)}{f''(x_0)} \\
&= x_0 - \frac{3x_0^2 - 94x_0 + 570}{6x_0 - 94} = 24.8 \\
x_2 &= x_1 - \frac{f'(x_1)}{f''(x_1)} = 23.3 \\
x_3 &= x_2 - \frac{f'(x_2)}{f''(x_2)} = 23.1 \\
x_4 &= x_3 - \frac{f'(x_3)}{f''(x_3)} = 23.1
\end{aligned}$$

To test whether the function has reached its local minimum point, we can evaluate the second derivative of f at that point ($x_0 = 23.1$). That is,

$$f''(23.1) = 6(23.1) - 94 = 44.6 > 0.$$

Since the sign of the second derivative is positive, it means that a local minimum is attained.

Newton's method is rarely employed for one - dimensional minimization; however, Newton's method and its variants are very commonly used for multi-dimensional minimization.

The multi-dimensional form may be derived by applying Newton's method to find a stationary point of $f : \mathbb{R}^n \to \mathbb{R}$ by solving the nonlinear system of equations $\nabla f(\mathbf{x}) = 0$. Alternatively, we can use Taylor series for a function of several variables to obtain the multi-dimensional analogue of Equation 3.1. Using the Taylor's series, we have

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T\mathbf{H}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \cdots, \quad (3.3)$$

where $\mathbf{H}(\mathbf{x})$ (also denoted by $\nabla^2 f(\mathbf{x})$) is a matrix called the Hessian matrix, and given by

$$\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

The Hessian matrix is symmetric if f is twice continuously differentiable. Consider Equation 3.3 and assume that the higher-order terms are negligible. If we take $\mathbf{x}_0$ to be $\mathbf{x}^*$, and since $\nabla f(\mathbf{x}^*)$ is zero, we have

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T\mathbf{H}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + \cdots \\ &= f(\mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T\mathbf{H}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + \cdots \end{aligned}$$

Since $f(\mathbf{x}^*)$ is the local minimum value of f, it must be that

$$(\mathbf{x} - \mathbf{x}^*)^T\mathbf{H}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \geq 0$$

at least for $\mathbf{x}$ near $\mathbf{x}^*$; if the minimum is a strict local minimum, then we must have

$$(\mathbf{x} - \mathbf{x}^*)^T\mathbf{H}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) > 0$$

$(\mathbf{x} \neq \mathbf{x}^*)$. The quantity $\mathbf{x} - \mathbf{x}^*$ is just a vector. Hence by using vector calculus either on Newton's method for finding the roots applied to $\nabla f(\mathbf{x}) = 0$ to find its minimum (assuming in either case that $\mathbf{H}(\mathbf{x}^*)$ is positive definite), we arrive at newton's method for minimization:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}^{-1}(\mathbf{x}_k)\nabla f(\mathbf{x}_k).$$

Therefore, if $\mathbf{H}(\mathbf{x}^*)$ is positive definite, then it is necessarily nonsingular, and so $H^{-1}(\mathbf{x})$ exists at the minimum $\mathbf{x} = \mathbf{x}^*$. We write Newton's method for minimization as

$$\mathbf{H}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\nabla f(\mathbf{x}_k), \tag{3.4}$$

Then

1. solve for the step $\boldsymbol{\delta}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$,
2. compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k$.

We consider the case of a sufficiently differentiable function of two variables $f : \mathbb{R}^2 \to \mathbb{R}$. The gradient is

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

and the Hessian is

$$\mathbf{H}(x,y) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

Newton's method then takes the form

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

where the Hessian and gradient on the right - hand side are evaluated at $(x, y) = (x_k, y_k)$. Since the inverse of the Hessian is usually expensive to compute we make use of Equation 3.4 to determine $[x_{k+1}; y_{k+1}]$.

Example 3.2. Consider the problem of minimizing the function $f(x, y) = -e^{-x^2-y^2}$ using the Newton's method.

We have

$$\begin{aligned} \frac{\partial f}{\partial x} &= 2xe^{-x^2-y^2} \\ \frac{\partial f}{\partial y} &= 2ye^{-x^2-y^2} \end{aligned}$$

so

$$\nabla f(x, y) = \begin{bmatrix} 2xe^{-x^2-y^2} \\ 2ye^{-x^2-y^2} \end{bmatrix}$$

The Hessian matrix of $f(x, y)$ is given by

$$\mathbf{H}(x, y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

where

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= 2e^{-x^2-y^2}(1 - 2x^2) \\ \frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x} &= -4xye^{-x^2-y^2} \\ \frac{\partial^2 f}{\partial y^2} &= 2e^{-x^2-y^2}(1 - 2y^2) \end{aligned}$$

Hence Newton's method for finding the minimum of $f(x, y)$ is

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} - \begin{bmatrix} 2e^{-x^2-y^2} & -4xye^{-x^2-y^2} \\ -4xye^{-x^2-y^2} & 2e^{-x^2-y^2} \end{bmatrix}^{-1} \times \begin{bmatrix} 2xe^{-x^2-y^2} \\ 2ye^{-x^2-y^2} \end{bmatrix}$$

Taking $[x_0; y_0] = [0.3; 0.3]$ as the starting point

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix} - \begin{bmatrix} -0.82 & -0.18 \\ -0.18 & -0.82 \end{bmatrix} \begin{bmatrix} -0.3 \\ -0.3 \end{bmatrix} = \begin{bmatrix} -0.16875 \\ -0.16875 \end{bmatrix}$$

Similarly, the values of $\mathbf{x}_k$ for four iterates with corresponding $f(\mathbf{x}_k)$, $\nabla f(\mathbf{x}_k)$ and $\|\nabla f(\mathbf{x}_k)\|_\infty$ are given in Table 6.

Table 6: Iterations showing the function and gradient values at each iterate

k	$\mathbf{x}_k$	$f(\mathbf{x}_k)$	$\nabla f(\mathbf{x}_k)$	$\|\nabla f(\mathbf{x}_k)\|_\infty$
0	[0.3;0.3]	-0.83527	[0.50116; 0.50116]	0.50116
1	[-0.16875; -0.16875]	-0.94464	[-0.31882; -0.31882]	0.31882
2	[0.02169; 0.02169]	-0.99906	[0.04335; 0.04335]	0.04335
3	[0; 0]	-1.00000	[0; 0]	0.00000
4	[0; 0]	-1.00000	[0; 0]	0.00000

We see from Table 6 that as the iterations increase, the values of the function at the various iterates decrease. This shows that we are actually moving towards a minimum point. The values of $\nabla f(\mathbf{x}_k)$ are also diminishing. Now, setting a stopping condition of $\nabla f(\mathbf{x}_k) \leq \epsilon = 10^{-3}$, we observe that $\nabla f(\mathbf{x}_k)$ is less than the tolerance value after four iterations. Thus, we take $[-2.6481; \ -1.5933]$ as the critical point. At this point, we note that the corresponding value of the $\|\nabla f(\mathbf{x}_k)\|_\infty = 2.2665 \times 10^{-4}$ is much less than the tolerance. Furthermore, the Hessian

$$\mathbf{H}(\mathbf{x}_4) = \begin{bmatrix} 2 & -2.7756 \times 10^{-7} \\ -2.7756 \times 10^{-7} & 2 \end{bmatrix}$$

is positive definite. Therefore the minimum point of f is $[0; 0]$ and is obtained after four iterations from the initial point $[0.3; 0.3]$.

Example 3.3. Consider the problem of minimizing the function

$$f(x_1, x_2) = (x_2 - x_1^2)^2 + (1 - x_1)^2$$

Now, if we take $\mathbf{x}_0 = [0; 0]$, then $g(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) = [-2; 0]$ and

$$\mathbf{H}(\mathbf{x}_0) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

Clearly, $\mathbf{H}(\mathbf{x}_0)$ is positive definite. From Equation 3.4, solving for $\boldsymbol{\delta}\mathbf{x}_0$, we obtain

$$\boldsymbol{\delta}\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Hence,

$$\mathbf{x}_1 = \boldsymbol{\delta}\mathbf{x}_0 + \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and

$$\nabla f(\mathbf{x}_1) = [4; -2].$$

Now,

$$\mathbf{H}(\mathbf{x}_1) = \begin{bmatrix} 14 & -4 \\ -4 & 2 \end{bmatrix}$$

which is positive definite (i.e. leading principal minors are both positive). The change in $\mathbf{x}_1$, is then given by

$$\boldsymbol{\delta}\mathbf{x}_1 = \begin{bmatrix} 1.752 \times 10^{-7} \\ 1 \end{bmatrix}.$$

Hence,

$$\mathbf{x}_2 = \boldsymbol{\delta}\mathbf{x}_1 + \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and

$$\nabla f[1; 1] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The Hessian

$$\mathbf{H}(\mathbf{x}_2) = \begin{bmatrix} 10 & -4 \\ -4 & 2 \end{bmatrix}$$

is positive definite. Thus, at $\mathbf{x}_2 = [1; 1]$ we have satisfied the condition for a critical point to be minimum. Therefore, the minimum point of f is $[1; 1]$ and is obtained after two iterations from the initial point $\mathbf{x}_0 = [0; 0]$. It would be recalled that under the Steepest Descent method, the minimum point of the Modified Rosenbrock's function using the same initial point $\mathbf{x}_0 = [0; 0]$ was obtained after three iterations. Thus, comparing the two methods, we realize that Newton's method convergences faster than the Steepest Descent.

Example 3.4.

$$\text{minimize} : f(x_1, x_2) = x_1^4 + x_1 x_2 + (1 + x_2)^2$$

Using an initial guess of $\mathbf{x}_0 = [0.75; -1.25]$, we obtain

$$\begin{aligned} \nabla f(\mathbf{x}_0) &= [0.43750; 0.25000], \text{and} \\ \mathbf{H}(\mathbf{x}_0) &= \begin{bmatrix} 6.74999 & 1 \\ 1 & 2 \end{bmatrix} \end{aligned}$$

From Equation 3.4,

$$\mathbf{H}(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\nabla f(\mathbf{x}_k),$$

we obtain

$$\mathbf{x}_1 = \begin{bmatrix} 0.7 \\ -1.35 \end{bmatrix}$$

with

$$\nabla f(\mathbf{x}_1) = \begin{bmatrix} 2.2 \times 10^{-2} \\ -5.5511 \times 10^{-12} \end{bmatrix}$$

Similarly,

$$\mathbf{x}_2 = \begin{bmatrix} 0.69591 \\ -1.34796 \end{bmatrix}$$

with

$$\nabla f(\mathbf{x}_2) = \begin{bmatrix} 1.3104 \times 10^{-4} \\ -1 \times 10^{-5} \end{bmatrix};$$

and

$$\mathbf{x}_3 = \begin{bmatrix} 0.69588 \\ -1.34794 \end{bmatrix}$$

with

$$\nabla f(\mathbf{x}_3) = \begin{bmatrix} -2.3294 \times 10^{-5} \\ 5.5511 \times 10^{-12} \end{bmatrix}.$$

Using a tolerance of $\epsilon = 10^{-5}$ we see that $\nabla f(\mathbf{x}_3) = [0; 0]$ and the Hessian

$$\mathbf{H}(\mathbf{x}_3) = \begin{bmatrix} 5.81097 & 1 \\ 1 & 1.99999 \end{bmatrix}$$

is positive definite. Thus, $\mathbf{x}_3 = [0.69588; -1.34794]$ is a minimum point of $f(\mathbf{x})$. We recall that in Example 2.9, the minimum point of this function under Conjugate Gradient method is the same as $\mathbf{x}_3$ and was attained in just one iteration with the starting point of $\mathbf{x}_0 = [0; 0]$. We note that $\mathbf{x}_0 = [0; 0]$ is farther from $\mathbf{x}_3$ than the starting point $\mathbf{x}_0 = [0.75; -1.25]$ used in this case.

If we use $\mathbf{x}_0 = [0; 0]$ in this example, we obtain

$$\nabla f(\mathbf{x}_0) = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

and the

$$\mathbf{H}(x_0) = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$

This matrix is indefinite and hence we have a saddle point at $\mathbf{x}_0 = [0; 0]$. This means that the expression

$$\mathbf{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k)$$

in the Newton's method fails to be a descent direction. To ensure a descent direction, we make use of an alternative direction $\mathbf{v}_k$ which is the eigenvector corresponding to the negative eigenvalue of the Hessian matrix at $\mathbf{x}_k$ in the steepest descent method. That is,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k.$$

Using this method, we compute the various iterates for given values of the step size α_k and $\mathbf{v}_k$. These iterations are given in the following table.

Table 7: Number of iterations, the step size and the descent direction

k	α_k	$\mathbf{v}_k$	$\mathbf{x}_k$
0	0.21571	[-0.92388; 0.38268]	[0; 0]
1	0.0051142	[-0.89608; 0.44388]	[-0.199292; 0.082549]
2	0.00024236	[-0.89451; 0.44704]	[-0.203875; 0.084819]
3	0.000040420	[-0.89444; 0.44719]	[-0.204092; 0.084927]
4	0.000040420	[-0.89443 ; 0.44722]	[-0.204128; 0.084945]
5	0.000040420	[-0.89443 ; 0.44722]	[-0.204128; 0.084945]

We see from Table 7 that the iterations converge after five iterations to the critical point $[-0.204128; 0.084945]$. It is interesting to note that this minimum point is different from the minimum point $[0.69588; -1.34794]$ obtained under the Conjugate Gradient method after only one iteration.

In the above example, we encountered a situation where the Hessian at a point was indefinite and therefore could not continue with the Newton's method. At this point, the steps taken to obtain a positive definite Hessian basically combines the Newton's method with the Steepest Descent method. This procedure is in effect a modification of the Newton's method to achieve a positive Hessian. The procedure is further illustrated in the next section.

Modified Newtons's Method

In Newton's method,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$$

the expression $\mathbf{s}_k = -\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ is a descent direction if $\mathbf{H}(\mathbf{x}_k)^{-1}$ is positive definite. The algorithm for damped Newton's method is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k) \tag{3.5}$$

for some damping sequence $\{\alpha_k\}_{k=0}^{k=\infty}$, $0 < \alpha_k \leq 1$, and $\alpha_k \to 1$ as $k \to \infty$. When $\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ is not a descent direction, a modified Newton's method substitutes a descent direction $\mathbf{s}_k$ for $\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$. One possible way to enforce descent is to use Equation 3.5. Newton's method and the method of steepest descent are both of the form of Equation 3.5 for some sequence α_k and some matrix $\mathbf{H}(\mathbf{x}_k)$. From Equation 3.5, the modified Newton's method becomes the steepest descent method

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\nabla f(\mathbf{x}_k). \tag{3.6}$$

That is, we obtain the modified Newton's method by simply taking
$\mathbf{H}(\mathbf{x}_k)^{-1} = \mathbf{I}_k$, where $\mathbf{I}_k$ is the identity matrix of dimension k.

In most cases, it is still possible to compute $\mathbf{s}_k$ from

$$\mathbf{s}_k = -\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$$

and to search along $\pm\mathbf{s}_k$, the sign chosen to ensure a descent direction.

The theoretical arguments against the generalized Newton's method are, first, if $\mathbf{H}(\mathbf{x}_k)$ is not a positive definite matrix, a move in the direction given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k),$$

where $\alpha_k > 0$ is chosen to minimize f along $-\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$, starting from $\mathbf{x}_k$, may result in an increase rather than a decrease in $f(\mathbf{x})$, yielding $\alpha_k = 0$, which terminate the process at $\mathbf{x}_k$. The second theoretical objection is that $\mathbf{H}(\mathbf{x}_k)$ may not have an inverse, even if $f(\mathbf{x})$ is convex. The modified second-order method to be discussed takes into account these two relevant objections to the generalized Newton's method. The direction vector $\mathbf{s}_k$ is generated according to two rules. In both cases,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k,$$

where α_k is chosen to be the smallest value of $\alpha \geq 0$ for which $\mathbf{x}_k + \alpha_k \mathbf{s}_k$ is a local minimum of $f(\mathbf{x}_k + \alpha_k \mathbf{s}_k)$. The rules(recommended by Fiacco and McCormick, 1967) are as follows:

1. If $\mathbf{H}(\mathbf{x}_k)$ has a negative eigenvalue, let $\mathbf{s}_k$ be a vector where

$$(\mathbf{s}_k)^T \mathbf{H}(\mathbf{x}_k)\mathbf{s}_k < 0 \ \text{ and } (\mathbf{s}_k)^T \nabla f(\mathbf{x}_k) \leq 0. \tag{3.7}$$

2. If $\mathbf{H}(\mathbf{x}_k)$ has all eigenvalues greater than or equal to zero, choose $\mathbf{s}_k$ such that either

$$\mathbf{H}(\mathbf{x}_k)\mathbf{s}_k = 0, \ \ \mathbf{s}_k^T \nabla f(\mathbf{x}_k) < 0$$

or

$$\mathbf{H}(\mathbf{x}_k)\mathbf{s}_k = -\nabla f(\mathbf{x}_k).$$

The rationale for Rule 1 is that if the second partial derivative matrix has a negative eigenvalue there are certain directions along which the function $f(\mathbf{x})$ decreases and along which the *rate* of decrease also decreases. That is, for a vector $\mathbf{s}$ satisfying Equation 3.5,

$$\frac{df(\mathbf{x}_k + \alpha_k \mathbf{s}_k)}{d\alpha_k} = \mathbf{s}_k^T \nabla f(\mathbf{x}_k) \leq 0, \text{and}$$

$$\frac{d^2 f(\mathbf{x}_k + \alpha_k \mathbf{s}_k)}{d\alpha_k^2} = \mathbf{s}_k^T \mathbf{H}(\mathbf{x}_k)\mathbf{s}_k < 0$$

at $\alpha_k = 0$.

Now, to get the value of $\mathbf{s}_k$, we factorize the matrix $\mathbf{H}(\mathbf{x}_k)$ as

$$\mathbf{H}(\mathbf{x}_k) = \mathbf{LDL}^T,$$

where $\mathbf{L}$ is a nonsingular lower triangular matrix and $\mathbf{D}$ is a diagonal matrix. The conditions for the factorization of $\mathbf{H}(\mathbf{x}_k)$ are as follows:

1. If $\mathbf{D}$ has all positive diagonal elements, solve for $\mathbf{s}_k = -\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$.

2. If $\mathbf{D}$ has all nonnegative diagonal elements, and at least one is zero, the vector $\mathbf{s}_k$ is generated according to the second rule above.

3. If $\mathbf{D}$ has some diagonal elements that are negative, solve $\mathbf{L}^T\mathbf{v} = \mathbf{a}_k$, where $\mathbf{a}_k$ is a column vector with jth component= 0 if the jth diagonal element of $\mathbf{D} > 0$ and with jth component =1 if the jth diagonal element of $\mathbf{D} < 0$. Let $\mathbf{s}_k = \mathbf{v}$ if $\mathbf{v}^T\nabla f(\mathbf{x}_k) \leq 0$, and $\mathbf{s}_k = -\mathbf{v}$ otherwise.

We illustrate the use of the rules outlined above by considering again the minimization problem in Example 3.4.

Example 3.5. Consider the problem of minimizing the function

$$f(x_1, x_2) = x_1^4 + x_1x_2 + (1 + x_2)^2$$

with a starting point of $\mathbf{x}_0 = [0; 0]$, we obtain $\nabla f(\mathbf{x}_0) = [0; 2]$ and

$$\mathbf{H}(\mathbf{x}_0) = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}.$$

Here, the matrix is indefinite. From the Newton's method,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$$

and $\mathbf{s}_k = -\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ is not a descent direction. Therefore, we look for a new direction, $\mathbf{v}$. The Newton's method then becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{v}_k$$

To obtain $\mathbf{v}$, we factorize the Hessian matrix $\mathbf{H}(\mathbf{x}_0)$ as $\mathbf{LDL}^T$. In this factorization,

$$\mathbf{D} = \begin{bmatrix} 2 & 0 \\ 0 & -0.5 \end{bmatrix}$$

Now, because some of the diagonal entries are negative, we use the second condition to get $\mathbf{a}$. We then solve

$$\begin{aligned} \mathbf{L}^T\mathbf{v} &= \mathbf{a} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ \Rightarrow \mathbf{v} &= \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} \end{aligned}$$

where

$$\mathbf{a}_k = \begin{cases} 1, & \text{if } \mathbf{d}_k < 0 \\ 0, & \text{otherwise} \end{cases}$$

which is the required descent direction. Using the steepest descent algorithm to compute the first iterate, we obtain

$$\mathbf{x}_1 = \begin{bmatrix} -0.20363 \\ 0.10181 \end{bmatrix}$$

The Hessian at $\mathbf{x}_1$ is

$$\mathbf{H}(\mathbf{x}_1) = \begin{bmatrix} 0.49758 & 1 \\ 1 & 1.99999 \end{bmatrix}$$

The matrix $\mathbf{H}(\mathbf{x}_1)$ is not positive definite since the leading principal minors are not both positive. The process therefore cannot continue. Thus the recommendation of Fiacco and McCormick (1967) seizes to work. To overcome this problem, we need to search along a different direction since $\mathbf{x}_1$ is in the same direction as $\mathbf{v}$. We observe that if we choose the direction specified by $\nabla f(\mathbf{x}_1)$ we obtain

$$\mathbf{x}_2 = \begin{bmatrix} -0.23656 \\ -0.86608 \end{bmatrix}$$

At $\mathbf{x}_2$, the Hessian is

$$\mathbf{H}(\mathbf{x}_2) = \begin{bmatrix} 0.67150 & 1 \\ 1 & 2 \end{bmatrix}$$

which is positive definite and therefore we switch back to the Newton's method.

The following table shows the iterates, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{12}$.

Table 8: Values of the function and gradient at each iterate for the Descent and Newton's Methods

Method	k	$\mathbf{x}_k$	$\nabla f(\mathbf{x}_k)$	$f(\mathbf{x}_k)$
	0	[0; 0]	[0; 2]	1
Steepest	1	[-0.20363; 0.10181]	[0.068041; 2]	1.1950
Newton	2	[-0.23656; -0.86608]	[-0.919032; 0.031278]	0.22594
Newton	3	[5.2134; -3.6067]	[563.17; -8.3901×10^{-6}]	726.70
Newton	4	[3.4840; -2.7415]	[166.41; 9.8662×10^{-4}]	140.81
Newton	5	[2.3375; -2.1688]	[48.921; 1.9252×10^{-6}]	26.153
Newton	6	[1.5857; -1.7928]	[14.156; 7.9298×10^{-6}]	4.1081
Newton	7	[1.1086; -1.5543]	[3.8962; -2.6482×10^{-7}]	0.094751
Newton	8	[0.83521; -1.4176]	[0.91289; 2.2768×10^{-7}]	-0.52299
Newton	9	[0.71923; -1.35961]	[0.12858; 3.8630×10^{-7}]	-0.58096
Newton	10	[0.69670; -1.34835]	[0.0043342; 6.8773×10^{-8}]	-0.58244
Newton	11	[0.69589; -1.34794]	[5.5268×10^{-6}; 1.8097×10^{-9}]	-0.58245
Newton	12	[0.69588; -1.34794]	[-5.5511×10^{-12}; 5.5511×10^{-12}]	-0.58245

From the table, we observe that $\nabla f(\mathbf{x}_k)$ steadily approaches zero, and the function values are decreasing. After twelve iterations, the iterates converged to the solution $\mathbf{x}_{12} = [0.69588; -1.34794]$

Other Forms of Modification

1. If the Hessian matrix is not positive definite, we look for another search direction which is an eigenvector, $\mathbf{v}_k$, of $\mathbf{H}(\mathbf{x}_k)$ corresponding to the negative eigenvalue and that gives a direction along which f can be reduced. The algorithm then becomes the steepest descent whenever $\mathbf{H}(\mathbf{x}_k)$ is not positive definite, and it is given by the equation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k$$

2. An alternative approach is to modify the Newton's search direction by giving it a bias towards the steepest descent vector $-\nabla f(\mathbf{x}_k)$. This is achieved by adding a scalar multiple of the unit matrix to $\mathbf{H}(\mathbf{x}_k)$ and solving the system

$$(\mathbf{H}(\mathbf{x}_k) + \mathbf{vI})\mathbf{s}_k = -\nabla f(\mathbf{x}).$$

This idea was first developed by (Levenberg, 1944; Marquardt, 1963). The idea is to choose $\mathbf{v}$ so that the modified matrix $\mathbf{H}(\mathbf{x}_k)+\mathbf{vI}$ is positive definite.

3. Another alternative is by modifying the Hessian matrix in the form

$$\mathbf{H}(\mathbf{x}_k) + \mathbf{D}$$

where $\mathbf{D}$ is diagonal and it is used to determine the search direction. This modification was developed by (Murray, 1972 ; Hebden, 1973).The modification occurs as the matrix is being factorized.

Quasi - Newton Methods

Quasi-Newton methods are intended for the situation where the Hessian is expensive or difficult to calculate. Quasi-Newton methods use only first derivatives to build an approximate Hessian over a number of iterations. This approximation is updated at each iteration by a matrix of low rank. In unconstrained minimization, the original quasi-Newton equation is

$$\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{y}_k,$$

where $\mathbf{y}_k$ is the difference of the gradients at the last two iterates. A new quasi-Newton equation is given by

$$\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{y}_k^*,$$

where $\mathbf{y}_k^*$ is the sum of $\mathbf{y}_k$ and $\mathbf{A}_k\mathbf{s}_k$ where $\mathbf{A}_k$ is some matrix . Two choices are given for $\mathbf{A}_k$ which carry some second order information of the Hessian of the objective function.

In Quasi-Newton method, we use matrices which approximate the Hessian matrix or its inverse, instead of the Hessian matrix or its inverse in Newton's method. The matrices are normally denoted by

$$\mathbf{B} \simeq \mathbf{H}(\mathbf{x}) \text{ and } \mathbf{D} \simeq \mathbf{H}(\mathbf{x})^{-1}.$$

The matrices can be produced in many different ways ranging from very simple techniques to highly advanced schemes, where the approximation is built up and adjusted dynamically on the basis of information about the first derivatives, obtained during the iteration. The simplest and most straight-forward Quasi-Newton method is obtained if the elements of the Hessian matrix are approximated by finite differences. In each coordinate direction, $\mathbf{e}_i (i = 1, \cdots, n)$, a small increment δ_i is added to the corresponding element of $\mathbf{x}$ and the gradient in this point is calculated. The i^{th} column of a matrix $\mathbf{B}$ is calculated as the difference approximation

$$\frac{(\nabla f(\mathbf{x}+\delta_i\mathbf{e}_i)-\nabla f(\mathbf{x}))}{\delta_i}.$$

After this, the symmetric matrix $\mathbf{B}=\frac{1}{2}(\mathbf{B}+\mathbf{B}^T)$ is formed. Extra gradient evaluations are avoided in Quasi - Newton methods but instead, updating formulas where the $\mathbf{B}$ or $\mathbf{D}$ matrices are determined from information about the iterates, $\mathbf{x}_1, \mathbf{x}_2, \cdots$ and the gradients of the cost function, $\nabla f(\mathbf{x}_1), \nabla f(\mathbf{x}_2), \cdots$ gathered during the iteration steps. Thus, in each iteration step the $\mathbf{B}$ or $\mathbf{D}$ matrix is changed so that it finally converges towards $\mathbf{H}(\mathbf{x}^*)$, $\mathbf{x}^*$ being the minimizer. Approximations to the inverse Hessian are preferred rather than approximations to the Hessian itself. This is because, the computational labour in the updating is the same no matter which of the matrices we update. Also, if we have an approximate inverse, then the search direction is found simply by multiplying the approximation with $-\nabla f$. This is an $O(n^2)$ process whereas the solution of the linear system with $\mathbf{B}$ as coefficient matrix is an $O(n^3)$ process. Another possibility is to use approximations to the Cholesky factor of the Hessian matrix, determined at the start of the iteration and updated in the iteration. Using this, we can find the solution of the system in $O(n^2)$ operations. A classical Quasi - Newton method with updating always includes a line search. Basically, there are two different approaches (line search or trust region) which defines two classes of methods. We shall confine ourselves to the line search approach.

Iteration step in Quasi - Newton with updating and line search

We take $\mathbf{B}$ (or $\mathbf{D}$) as the current approximation to $\mathbf{H}(\mathbf{x})$ or $\mathbf{H}(\mathbf{x})^{-1}$.
Solve $\mathbf{B}\mathbf{h}_{qn} = -\nabla f(\mathbf{x})$ or compute $\mathbf{h}_{qn} := -\mathbf{D}\nabla f(\mathbf{x})$.
Line search along $\mathbf{h}_{qn}$ giving $\mathbf{h}_{qn} := \alpha\mathbf{h}_{qn}$; $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{h}_{qn}$ Update $\mathbf{B}$ to $\mathbf{B}_{new}$ (or $\mathbf{D}$ to $\mathbf{D}_{new}$).

The Quasi - Newton Condition

An updating formula must satisfy the Quasi - Newton condition, which may be derived in many ways. The condition is also referred to as the **secant condition**, because it is closely related to the secant method for non - linear equations with one unknown. Let $\mathbf{x}$ and $\mathbf{B}$ be the current iterate and approximation to $\mathbf{H}(\mathbf{x})$. Given these, the first parts of the iteration step can be performed yielding $\mathbf{h}_{qn}$ and hence $\mathbf{x}_1$. The objective is to calculate $\mathbf{B}_{new}$ by a correction of $\mathbf{B}$. The correction must contain some information about the second derivatives. This information is only approximate. It is based on the gradients of f at the two points. Now, consider the Taylor expansion of ∇f around $\mathbf{x}+\mathbf{h}_{qn}$:

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}+\mathbf{h}_{qn}) - \mathbf{H}(\mathbf{x}+\mathbf{h}_{qn})\mathbf{h}_{qn} + \cdots \tag{3.8}$$

We assume that we can neglect the higher order terms, and with the notation

$$\mathbf{y} = \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_0),$$

Equation 3.8 leads to the relation

$$\mathbf{y} \simeq \mathbf{H}(\mathbf{x}_1)\mathbf{h}_{qn}.$$

Therefore, we require that $\mathbf{B}_{new}$ should satisfy

$$\mathbf{B}_{new}\mathbf{h}_{qn} = \mathbf{y}$$

This is the Quasi - Newton condition. The same arguments leads to the alternative formulation of the Quasi - Newton condition,

$$\mathbf{D}_{new}\mathbf{y} = \mathbf{h}_{qn}.$$

The Quasi - Newton condition only supplies n conditions on the matrix $\mathbf{B}_{new}$ (or $\mathbf{D}_{new}$) but the matrix has n^2 elements. Therefore, additional conditions are needed to get a well defined method.

In the Quasi - Newton methods that we describe, the $\mathbf{B}$ or ($\mathbf{D}$) matrix is updating in each iteration step. We produce $\mathbf{B}_{new}$ (or $\mathbf{D}_{new}$) by adding a correction term to the present $\mathbf{B}$ or ($\mathbf{D}$). Important requirements to the updating are that it must be simple and fast to perform and yet effective. This can be obtained with a recursive relation between successive approximations,

$$\mathbf{B}_{new} = \mathbf{B} + \mathbf{W},$$

where $\mathbf{W}$ is a correction matrix. In most methods, $\mathbf{W}$ is a **rank - one matrix**, that is,

$$\mathbf{B}_{new} = \mathbf{B} + \mathbf{a}\mathbf{b}^T$$

or a rank - two matrix, that is,

$$\mathbf{B}_{new} = \mathbf{B} + \mathbf{a}\mathbf{b}^T + \mathbf{u}\mathbf{v}^T,$$

where $\mathbf{a}, \mathbf{b}, \mathbf{u}, \mathbf{v} \in \mathbb{R}^n$. Hence, $\mathbf{W}$ is an outer product of two vectors or a sum of two such products. Often, $\mathbf{a} = \mathbf{b}$ and $\mathbf{u} = \mathbf{v}$; this is a simple way of ensuring that $\mathbf{W}$ is symmetric.

Comparison with Conjugate Gradient Algorithms

The Quasi - Newton methods are closely related to conjugate directions and conjugate gradient algorithms when applied to quadratic functions. When minimizing a general non - quadratic function, Quasi - Newton methods(BFGS in particular) typically perform better. Partially, this is due to the fact that Quasi -Newton methods, in addition to generating conjugate directions, also tend to approximate the Hessian matrix, and hence, close to the optimum, the directions they generate tend to approximate the Newton's direction. This observation holds true regardless of the starting matrix $\mathbf{D}_1$, and hence it is typically unnecessary to restart Quasi - Newton methods. Also by numerical analysis, Quasi - Newton methods tend to be less sensitive to the accuracy of line searches performed at each iteration. On the other hand, compared to conjugate gradient methods, Quasi - Newton methods require more storage space, and each iteration requires more computation. (In particular, they have to store the matrix $\mathbf{D}_k$ and compute $\mathbf{D}_k \nabla f(\mathbf{x}_k)$ at each iteration, while the number of function and gradient evaluations is the same as in conjugate gradient algorithms.) In general, both conjugate gradient and quasi - Newton methods require significantly less work per iteration than Newton's method (unless special structure of the problem allows for efficient direct computation of the Hessian and its inverse). Quasi - Newton methods (typically with BFGS update of one form of another) are usually the algorithms of choice in unconstrained optimization software. The optimization toolbox in MATLAB implements quite a few gradient descent methods in its function fminunc. The software allows us to change the algorithm used to DFP Quasi - Newton method which approximate the inverse of the Hessian, or to steepest descent.

The Broyden Family

Many Quasi - Newton methods are advantageous due to their fast convergence and absence of second - order derivative computation. What makes a Quasi - Newton method work is the choice of the matrix $\mathbf{B}_k$ at each iteration. The important idea behind the methods is that two successive iterates $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ together with the gradients $\nabla f(\mathbf{x}_k)$ and $\nabla f(\mathbf{x}_{k+1})$ contain curvature(that is, Hessian) information, in particular,

$$(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)) \approx H(\mathbf{x}_{k+1})(\mathbf{x}_{k+1} - \mathbf{x}_k)$$

The above approximation is an equality when the function is quadratic. Therefore, at every iteration we would like to choose $\mathbf{B}_{k+1}$ to satisfy

$$\mathbf{B}_{k+1}\mathbf{q}_k = \mathbf{p}_k \tag{3.9}$$

where $\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{q}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$. Equation 3.9 is the Quasi - Newton condition, or the secant equation. Suppose that at every iteration, we update the matrix $\mathbf{B}_{k+1}$ by taking the matrix $\mathbf{B}_k$ and adding a "correction" term $\mathbf{C}_k$. Then the secant equation becomes

$$(\mathbf{B}_k + \mathbf{C}_k)\mathbf{q}_k = \mathbf{p}_k \Rightarrow \mathbf{C}_k\mathbf{q}_k = \mathbf{p}_k - \mathbf{B}_k\mathbf{q}_k \tag{3.10}$$

Equation 3.10 leaves us a lot of flexibility in selecting the correction matrix $\mathbf{C}_k$. The most popular update methods come from the following (parametric) family of matrices. (we note that the subscript k is omitted in most of the following formulas for simplicity, here $\mathbf{B} = \mathbf{B}_k$):

$$C_B(\xi) = \frac{pp_t}{p_tq} - \frac{\mathbf{B}qq_t\mathbf{B}}{q_t\mathbf{B}q} + \xi\tau vv_t \tag{3.11}$$

where $v = \dfrac{p}{p_tq} - \dfrac{\mathbf{B}q}{\tau}$, $\tau = q_t\mathbf{B}q$

The choice of the scalar $\xi \in [0, 1]$, which parameterizes the family of matrices C, gives rise to several popular choices of updates. In particular:

1. Setting $\xi = 0$ at each iteration, we obtain the DFP (Davidson-Fletcher-Powell) update.

   $$C_{DFP} = C_B(0) = \frac{pp_t}{p_tq} - \frac{\mathbf{D}qq_t\mathbf{D}}{q_t\mathbf{D}q},$$

   where $\mathbf{B} = \mathbf{D}$

   which is historically the first Quasi - Newton method.

2. Setting $\xi = 1$ at each iteration, we obtain the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update:

   $$C_{BFGS} = C_B(1) = \frac{pp_t}{p_tq}[1 + \frac{q_t\mathbf{D}q}{p_tq}] - \frac{\mathbf{D}qp_t + pq_t\mathbf{D}}{p_tq}.$$

   The resulting method has been shown to be superior to other updating schemes in its overall performance.

3. A general member of the Broyden family in Equation 3.11 can therefore be written as a convex combination of the two above updates:

$$C_B(\xi) = (1-\xi)C_{DFP} + \xi C_{BFGS}$$

The following two results demonstrate that Quasi - Newton methods generate descent search directions and the initial approximation $\mathbf{D}_1$ is positive definite,and, when applied to a quadratic function, result in conjugate direction methods.

Broyden-Fletcher-Goldfarb-Shanno Method

The **Broyden-Fletcher-Goldfarb-Shanno** (BFGS) method is a method for solving an unconstrained nonlinear optimization problem.

The BFGS method is derived from the Newton's method treated earlier on which is a class of optimization techniques that seeks the stationary point of a function, where the gradient is 0. Newton's method assumes that the function can be locally approximated as a quadratic in the region around the optimum, and use the first and second derivatives to find the stationary point.

In Quasi-Newton methods, the Hessian matrix of the function to be minimized does not need to be computed at every stage. The Hessian is updated by analyzing successive gradient vectors instead. Quasi-Newton methods are a generalization of the secant method to find the root of the first derivative for multidimensional problems. In multi-dimensions, the secant equation is under-determined, and Quasi-Newton methods differ in how they constrain the solution. The BFGS method is one of the most successful members of this class.

The BFGS method is termed a secant updating method. The general aim of these methods is to preserve symmetry of the approximate Hessian as well as maintain positive definiteness.

The Quasi- Newton method reduces work load per iteration and the BFGS method makes sure that a Quasi - Newton step is always a descent direction.

We start with an initial guess $\mathbf{x}_0$ and a symmetric positive definite approximate Hessian, B_0 (which is usually taken as an identity matrix.)

The search direction $\mathbf{p}_k$ at stage k is given by the solution of the analogue of the Newton equation

$$B_k\mathbf{p}_k = -\nabla f(\mathbf{x}_k).$$

A line search in the direction $\mathbf{p}_k$ is then used to find the next point $\mathbf{x}_{k+1}$.

Instead of requiring the full Hessian matrix at the point $\mathbf{x}_{k+1}$ to be computed as B_{k+1}, the approximate Hessian at stage k is updated by the addition of two matrices.

$$B_{k+1} = B_k + U_k + V_k$$

Both U_k and V_k are rank-one matrices but have different bases. The rank one assumption here means that we may write

$$C = \mathbf{a}\mathbf{b}^T$$

So equivalently, U_k and V_k construct a rank-two update matrix which is robust against the scale problem often suffered in the gradient descent searching.

CHAPTER FOUR

CONSTRAINED OPTIMIZATION

In the previous chapter, we considered the minimization of a function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$,without any restrictions on $\mathbf{x}$. In this chapter, we now consider the minimization of a function $f(\mathbf{x})$ where $\mathbf{x}$ is constrained to lie in some subspace of $\mathbb{R}^n$. Constraints can either be **equality constraints** or **inequality constraints**. Since the case of a scalar-variable follows easily from the vector one, only the latter is discussed in detail in this chapter.

Basic Notations and Examples

We consider nonlinear programming problems (NLP) of the form

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to } h(\mathbf{x}) = 0 \\ &g(\mathbf{x}) \leq 0 \end{aligned} \tag{4.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective functional and the functions $h : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^p$ are the equality and inequality constraints respectively. The functions f, g and h are assumed to be at least twice continuously differentiable.

Definition 4.1. Feasible set

The set of points that satisfy the equality and inequality constraints, that is,

$$\mathbb{F} := \{\mathbf{x} \in \mathbb{R}^n \| \ h(\mathbf{x}) = 0, \ g(\mathbf{x}) \leq 0\}$$

is called the feasible set of the NLP in Equation 4.1. Its elements are referred to as feasible points.

In terms of the feasible set, the NLP can be written in a more compact form as

$$\min_{\mathbf{x} \in \mathbb{F}} f(\mathbf{x})$$

To illustrate the impact of the constraints on the solution of an NLP, we first explain the concept of the Lagrange's method.

Equality Constraints and the Lagrange's Method

A typical equality constrained minimization problem has the form

$$\min_{\mathbf{x}} \; f(\mathbf{x})$$

$$\text{subject to } g_i(\mathbf{x}) = 0 \qquad i = 1, 2, \cdots, p \tag{4.2}$$

where $f(\mathbf{x})$ is the scalar-valued objective function and $g_i(\mathbf{x})$ is the vector-valued constraint function.

The classical approach to solving Equation 4.2 is the method of Lagrange multipliers. This approach converts the constrained optimization problem into an unconstrained one, thereby allowing the use of the techniques described in the previous section. The Lagrangian of the constrained minimization problem is defined to be the scalar-valued function

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T g(\mathbf{x}) \tag{4.3}$$

The following theorem gives the necessary condition for solving Equation 4.3.

Theorem 4.1. Let $\mathbf{x}_0$ denote a local solution to the constrained minimization problem given above where the gradients $\nabla_x(g_1(\mathbf{x}_0)), \cdots, \nabla_{\ x}(g_M(\mathbf{x}_0))$ of the constraint function's components are linearly independent. Then there exists a unique vector $\boldsymbol{\lambda}_0$ such that

$$\nabla_x(L(\mathbf{x}_0, \boldsymbol{\lambda}_0)) = \nabla f(\mathbf{x}) + \boldsymbol{\lambda}_i \Sigma \nabla g(\mathbf{x}) = 0.$$

Furthermore, the quadratic form

$$\mathbf{y}^T\nabla_x^2(L(\mathbf{x}_0, \boldsymbol{\lambda}_0))\mathbf{y}$$

is non-negative for all $\mathbf{y}$ satisfying $(\nabla_x(g(\mathbf{x})))^T\mathbf{y} = 0$.

The latter result in the theorem says that the Hessian of the Lagrangian evaluated at its stationary points is non-negative definite with respect to all vectors orthogonal to the gradient of the constraint. This result generalizes the notion of a positive definite Hessian in unconstrained minimization problems.

The following is a simple example that illustrates the use of the Lagrange's method.

Example 4.1. Consider the NLP

$$\begin{array}{ll} \text{minimize} & x_1 + x_2 \\ \text{subject to} & h(\mathbf{x}) = x_1^2 + x_2^2 - 2 = 0. \end{array}$$

The Lagrangian is given by

$$L(\mathbf{x}, \lambda) = x_1 + x_2 + \lambda(x_1^2 + x_2^2 - 2)$$

$$\frac{\partial L}{\partial x_1} = 1 + 2\lambda x_1, \qquad \frac{\partial L}{\partial x_2} = 1 + 2\lambda x_2, \qquad \frac{\partial L}{\partial \lambda} = x_1^2 + x_2^2 - 2$$

Equating the first two derivatives to zero and solving for $\mathbf{x}$ gives

$$\mathbf{x} = (-\frac{1}{2\lambda}, -\frac{1}{2\lambda})$$

subject to the condition $x_1^2 + x_2^2 - 2$, we obtain

$$\Rightarrow \frac{1}{4\lambda^2} + \frac{1}{4\lambda^2} = 2 \Rightarrow \lambda = \pm\frac{1}{2}.$$

When $\lambda = \frac{1}{2}$, $\mathbf{x} = [-1; -1]$ and when $\lambda = -\frac{1}{2}$, $\mathbf{x} = [1; 1]$

Now, when $\lambda = \frac{1}{2}$, $\frac{\partial^2 L}{\partial x_1^2} = 2\lambda > 0$ and the determinant of the Hessian of L

$$\frac{\partial^2 L}{\partial x_1^2} . \frac{\partial^2 L}{\partial x_2^2} - (\frac{\partial^2 L}{\partial x_1 \partial_2})^2 = 4\lambda^2 > 0$$

The results show that the Hessian of L is positive definite for the value of $\lambda = \frac{1}{2}$. Therefore $\mathbf{x} = [-1; -1]$ is the minimum point of the given objective function.

It is possible to obtain problems with two or more constraints and hence, require a vector of Lagrange multipliers to determine a minimum point. In particular, there are problems that involve quadratic objective functions or functions that could be approximated by a quadratic function. Quadratic approximation is necessary in minimization problems for the following reasons:

1. one can determine a unique minimum (or maximum);

2. it is easy to handle;

3. the Hessian is a constant.

Before illustrating how to obtain a solution to a minimization problem that involve a vector of Lagrange multipliers, we briefly review the theory of solution to quadratic objective function subject to linear constraints.

Quadratic Programming

Now, given the minimization problem

$$\begin{aligned} \min_{\mathbf{x}} \; & f(\mathbf{x}) \\ \text{subject to } & h(\mathbf{x}) = 0 \end{aligned}$$

The function $f(\mathbf{x})$ may be approximated by a quadratic function of the form

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

where $b = \nabla f$ and c is a constant. The quadratic approximation is possible if the function is at least twice continuously differentiable. The constraint $h(\mathbf{x})$ is also approximated by a linear function

$$h(\mathbf{x}) = \mathbf{Cx} - \mathbf{d}$$

The Lagrangian of a constrained minimization problem is then defined to be the scalar-valued function

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}(\mathbf{Cx} - \mathbf{d})$$

From Theorem 4.1,

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}} &= \mathbf{Ax} + \mathbf{b} + \mathbf{C}^T\boldsymbol{\lambda} = 0; \text{ and} \\ \frac{\partial L}{\partial \boldsymbol{\lambda}} &= \mathbf{Cx} - \mathbf{d} \end{aligned}$$

Thus, the solution $\mathbf{x}^*$ satisfies the matrix equation

$$\begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -\mathbf{b} \\ \mathbf{d} \end{bmatrix} \tag{4.4}$$

Example 4.2. Consider the following Quadratic Programming Problem:

$$\min\ z = 0.20x_1^2 + 0.08x_2^2 + 0.18x_3^2 + 0.10x_1x_2 + 0.04x_1x_3 + 0.06x_2x_3$$

subject to

$$\begin{aligned} 0.14x_1 + 0.11x_2 + 0.10x_3 &= 120 \\ x_1 + x_2 + x_3 &= 1000 \end{aligned}$$

This can be put into the form,

$$\min\ f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Ax},$$

$$\text{subject to } \mathbf{Cx} - \mathbf{d} = 0$$

where

$$\mathbf{A} = \begin{bmatrix} 0.40 & 0.10 & 0.04 \\ 0.10 & 0.16 & 0.06 \\ 0.04 & 0.06 & 0.36 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 0.14 & 0.11 & 0.10 \\ 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{d} = \begin{bmatrix} 120 \\ 1000 \end{bmatrix}$$

The optimality condition is given by

$$\begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{d} \end{bmatrix}$$

Denote

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix}$$

and

$$\mathbf{N} = \begin{bmatrix} \mathbf{0} \\ \mathbf{d} \end{bmatrix}$$

Here, $\mathbf{M}$ is a 5×5 matrix and $\mathbf{N}$ is a 5×1 matrix and are given by

$$\mathbf{M} = \begin{bmatrix} 0.4 & 0.1 & 0.04 & 0.14 & 1 \\ 0.1 & 0.16 & 0.06 & 0.11 & 1 \\ 0.04 & 0.06 & 0.36 & 0.1 & 1 \\ 0.14 & 0.11 & 0.10 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \qquad \mathbf{N} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 120 \\ 1000 \end{bmatrix}$$

The optimality condition can now be written as

$$\mathbf{MY} = \mathbf{N},$$

where $\mathbf{Y}$ represent

$$\begin{bmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix}.$$

Solving, we obtain

$$\mathbf{Y} = \begin{bmatrix} 380.95 \\ 476.19 \\ 142.86 \\ -2761.90 \\ 180.95 \end{bmatrix}$$

Therefore, the solution of the minimization problem is

$$\mathbf{x}^* = \begin{bmatrix} 380.95 \\ 476.19 \\ 142.86 \end{bmatrix}$$

and is specified by the vector of the Lagrange multiplier

$$\boldsymbol{\lambda}^* = \begin{bmatrix} -2761.90 \\ 180.95 \end{bmatrix}$$

It remains to show that $\mathbf{x}^*$ is the solution of the minimization problem. The condition for $\mathbf{x}^*$ to be the solution of the minimization problem is that

$$f_{11} > 0, \qquad \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{vmatrix} > 0, \qquad |\mathbf{M}| > 0$$

In this example,

$$f_{11} = 0.40, \qquad \begin{vmatrix} 0.40 & 0.10 \\ 0.10 & 0.16 \end{vmatrix} = 0.054, \text{ and} \qquad |\mathbf{M}| = 0.00042$$

Since all the three values are positive, it means that $\mathbf{M}$ is positive definite. As a result, the value of $\mathbf{x}^*$ is the solution of the minimization problem.

Example 4.3. Consider the following Quadratic Programming Problem:

$$\begin{aligned} \min z &= 3x_1^2 + 4x_2^2 + x_3^2 + 2x_4^2 - 2x_1x_2 + 2x_1x_3 - 2x_1x_4 + 2x_2x_3 \\ &- 2x_2x_4 - 2x_3x_4 + x_1 + x_2 + x_3 + x_4 \end{aligned}$$

subject to

$$\begin{aligned} 2x_1 + x_2 + 3x_3 + x_4 &= 5 \\ 4x_1 + x_2 + x_3 + 4x_4 &= 4 \end{aligned}$$

This can be put into the form,

$$\min\ f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x},$$

$$\text{subject to } \mathbf{C}\mathbf{x} - \mathbf{d} = 0$$

where

$$\mathbf{A} = \begin{bmatrix} 6 & -2 & 2 & -2 \\ -2 & 8 & 2 & -2 \\ 2 & 2 & 2 & -2 \\ -2 & -2 & -2 & 4 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} 2 & 1 & 3 & 1 \\ 4 & 1 & 1 & 4 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{d} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

The optimality condition is given by

$$\begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -\mathbf{b} \\ \mathbf{d} \end{bmatrix}$$

Denote

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix}$$

and

$$\mathbf{N} = \begin{bmatrix} -\mathbf{b} \\ \mathbf{d} \end{bmatrix}$$

Here, $\mathbf{M}$ is a 6×6 matrix and $\mathbf{N}$ is a 6×1 matrix and are given by

$$\mathbf{M} = \begin{bmatrix} 6 & -2 & 2 & -2 & -2 & -4 \\ -2 & 8 & 2 & -2 & -1 & -1 \\ 2 & 2 & 2 & -2 & -3 & -1 \\ -2 & -2 & -2 & 4 & -1 & -4 \\ -2 & -1 & -3 & -1 & 0 & 0 \\ -4 & -1 & -1 & -4 & 0 & 0 \end{bmatrix}, \ \mathbf{N} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 5 \\ 4 \end{bmatrix}$$

The optimality condition can now be written as

$$\mathbf{MY} = \mathbf{N},$$

where $\mathbf{Y}$ represent

$$\begin{bmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix}.$$

Solving, we obtain

$$\mathbf{Y} = \begin{bmatrix} -0.12578 \\ -0.24741 \\ 1.56776 \\ 0.79569 \\ 0.49068 \\ 0.32574 \end{bmatrix}$$

Therefore, the solution of the minimization problem is

$$\mathbf{x}^* = \begin{bmatrix} -0.12578 \\ -0.24741 \\ 1.56776 \\ 0.79569 \end{bmatrix}$$

and is specified by the vector of the Lagrange multiplier

$$\boldsymbol{\lambda}^* = \begin{bmatrix} 0.49068 \\ 0.32574 \end{bmatrix}$$

Now, we use $\mathbb{R}^4$ null space to determine whether or not $\mathbf{x}^*$ is the required solution. We observe that in the $\mathbb{R}^4$ space, the projected gradient of the Lagrangian matrix given by $[0; 0; 0; 0]$

is a zero vector. In addition, the projected Hessian matrix given by

$$\begin{bmatrix} 5.5130 & -1.6858 \\ -1.6858 & 7.4082 \end{bmatrix}$$

is positive definite, i.e. both its eigenvalues 4.5267 and 8.3945 are positive. These results show that $\mathbf{x}^*$ is the solution of the minimization problem specified by $\boldsymbol{\lambda}^*$.

Sequential Quadratic Programming Methods

The sequential quadratic programming (SQP) algorithm is one of the most successful general methods for solving nonlinear constrained optimization problems. In this method, we find a step away from the current point by minimizing a quadratic model of the problem. It also attempts to solve a nonlinear programming directly rather than converting it to a sequence of unconstrained minimization problems. At each step, a local model of the optimization problem is constructed and solved, yielding a step (hopefully) toward the solution of the original problem. In unconstrained minimization problems, only the objective function must be approximated, and the local model is quadratic but in the nonlinear programming problem, both the objective function and the constraint must be modelled. An SQP method uses a quadratic model for the objective function and a linear model for the constraint. A nonlinear programming in which the objective function is quadratic and the constraints linear is called a quadratic programming(QP). We consider the general method of SQP for solving constrained nonlinear programming problem.

Given a current iterate $\mathbf{x}_k$ of a solution $\mathbf{x}^*$, h can be approximated by

$$h(\mathbf{x}_k + \delta\mathbf{x}) = \nabla h(\mathbf{x}_k)^T \delta\mathbf{x} + h(\mathbf{x}_k),$$

and so the constraint

$$h(\mathbf{x}) = 0$$

is replaced by

$$\nabla h(\mathbf{x}_k)^T \delta\mathbf{x} + h(\mathbf{x}_k) = 0.$$

Rather than finding the Taylor's approximation to f, we obtain the approximation of the Lagrangian

$$L(\mathbf{x}_k+\delta\mathbf{x};\ \boldsymbol{\lambda}_k) = \frac{1}{2}\delta\mathbf{x}.\nabla^2 L(\mathbf{x}_k;\ \boldsymbol{\lambda}_k)\delta\mathbf{x}+\nabla L(\mathbf{x}_k;\ \boldsymbol{\lambda}_k).\delta\mathbf{x}+L(\mathbf{x}_k;\boldsymbol{\lambda}_k) \qquad \text{(for } \delta\mathbf{x} \text{ near } 0).$$

The solution to the problem

$$\min \qquad \frac{1}{2}\delta\mathbf{x}.\nabla^2 L(\mathbf{x}_k;\ \boldsymbol{\lambda}_k)\delta\mathbf{x} + \nabla L(\mathbf{x}_k;\ \boldsymbol{\lambda}_k).\delta\mathbf{x} + L(\mathbf{x}_k;\boldsymbol{\lambda}_k)$$

subject to

$$\nabla h(\mathbf{x}_k)^T \delta\mathbf{x} + h(\mathbf{x}_k) = 0$$

yields improved values of $\mathbf{x}^*$ and $\boldsymbol{\lambda}_k$.

Example 4.4. Define $f : \mathbb{R}^2 \to \mathbb{R}$ and $h : \mathbb{R}^2 \to \mathbb{R}$ by

$$\begin{aligned} f(\mathbf{x}) &= (x_2-2)^2 - x_1^2, \\ h(\mathbf{x}) &= 4x_1^2 + x_2^2 - 1. \end{aligned}$$

Then taking $\mathbf{x}_0 = (\frac{1}{20}, \frac{5}{4})$ and $\lambda_0 = -\frac{1}{2}$, the solution of the nonlinear programming

$$\min f(\mathbf{x})$$

$$\text{subject to } h(\mathbf{x}) = 0$$

reduces to that of

$$\min \qquad \frac{1}{2}\delta\mathbf{x}_0.\nabla^2 L(\mathbf{x}_0;\ \lambda_0)\delta\mathbf{x}_0 + \nabla L(\mathbf{x}_0;\ \lambda_0).\delta\mathbf{x}_0 + L(\mathbf{x}_0;\lambda_0)$$

subject to

$$\nabla h(\mathbf{x}_0)^T \delta\mathbf{x}_0 + h(\mathbf{x}_0) = 0$$

This simplifies to

$$\min \; -3\delta x_1^2 + \frac{1}{2}\delta x_2^2 - \frac{3}{10}\delta x_1 - \frac{11}{4}\delta x_2 + \frac{219}{800}$$

subject to

$$\frac{2}{5}\delta x_1 + \frac{5}{2}\delta x_2 + \frac{229}{400} = 0$$

Using Equation 4.4, we obtain the value of $\delta \mathbf{x}_0 = [\delta x_1; \delta x_2]$ as the solution to the equation $\mathbf{MY} = \mathbf{N}$, where

$$\mathbf{M} = \begin{bmatrix} -6 & 0 & 0.4 \\ 0 & 1 & 2.5 \\ 0.4 & 2.5 & 0 \end{bmatrix}, \qquad \mathbf{N} = \begin{bmatrix} -0.3 \\ -2.75 \\ -0.5725 \end{bmatrix}$$

Solving the equation, we obtain

$$\mathbf{Y} = \begin{bmatrix} -0.0173 \\ -0.226232 \\ -1.009507 \end{bmatrix}$$

where

$$\delta \mathbf{x}_0 = \begin{bmatrix} -0.0173 \\ -0.226232 \end{bmatrix}, \qquad \text{and } \lambda_1 = -1.009507$$

The new iterate $\mathbf{x}_1$ is then given by

$$\mathbf{x}_1 = \mathbf{x}_0 + \delta \mathbf{x}_0 = \begin{bmatrix} 0.032700 \\ 1.023768 \end{bmatrix}.$$

Next, we obtain the minimization of the Lagrangian

$$\min \qquad \frac{1}{2}\delta \mathbf{x}_1 . \nabla^2 L(\mathbf{x}_1;\; \lambda_1)\delta \mathbf{x}_1 + \nabla L(\mathbf{x}_1;\; \lambda_1).\delta \mathbf{x}_1 + L(\mathbf{x}_1; \lambda_1).$$

subject to

$$\nabla h(\mathbf{x}_1)^T \delta \mathbf{x}_1 + h(\mathbf{x}_1) = 0$$

This will yield values for $\mathbf{x}_2$ and λ_2. The procedure then continues until we obtain the solution of the problem to be $\mathbf{x}^* = [0.00997; 1.00000]$.

Inequality Constraints and the Kuhn-Tucker Conditions

When some of the constraints are inequalities, the Lagrange multiplier technique can be used, but the solution must be checked carefully in its details. But first, the optimization problem with equality and inequality constraints is formulated as

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) \\ \text{subject to } g(\mathbf{x}) = 0 \\ \text{and } h(\mathbf{x}) \leq 0 \end{aligned}$$

As before, $f(\mathbf{x})$ is the scalar-valued objective function and $g(\mathbf{x})$ is the equality constraint function; $h(\mathbf{x})$ is the inequality constraint function. The key result which can be used to find the analytic solution to this problem is to first form the Lagrangian function.

Example 4.5. Consider the constrained nonlinear minimization problem

$$\text{minimize } (x_2 + 100)^2 + 0.01x_1^2$$

$$\text{subject to g}(\mathbf{x}) := \cos x_1 - x_2 \leq 0.$$

Without constraint, the NLP has the unique solution $\mathbf{x} = (0, -100)^T$. With the constraint, there are infinitely many solutions near to the points

$$\mathbf{x} = (k\pi, -1)^T, \; k = \pm 1, \pm 3, \pm 5, \cdots .$$

Example 4.6. A single inequality constraint

Consider the inequality constrained NLP

$$\begin{aligned} \text{minimize} \quad & x_1 + x_2 \\ \text{subject to} \quad & g(\mathbf{x}) = x_1^2 + x_2^2 - 2 \leq 0. \end{aligned}$$

The feasible region is the closed ball $\bar{B}_{\sqrt{2}}(0)$ with radius $\sqrt{2}$ around the origin. As in the previous example, the solution is $\mathbf{x}^* = (-1,-1)^T$.

The following conditions is the general statement of the Lagrange multiplier technique for constrained optimization problems.

The Kuhn - Tucker Conditions

The necessary conditions for constrained problems have been developed by Kuhn - Tucker for a general nonlinear optimization problem with equality and inequality constraints.

This optimization problem written in terms of minimizing $f(\mathbf{x})$ is given by

$$\text{minimize} \;\; f(\mathbf{x})$$

$$\text{subject to } g_j(\mathbf{x}) \leq 0; \;\; j = 1, \cdots, p$$

$$h_j(\mathbf{x}) = 0; \;\; j = 1, \cdots, q$$

where $f(\mathbf{x}), g_j(\mathbf{x})$ and $h_j(\mathbf{x})$ are twice continuously differentiable real valued functions. Any value of $\mathbf{x}$ that satisfies the constraint equations is called a feasible solution to the problem in the Kuhn - Tucker theory. Then to locate points that can potentially be local minima of the equation and satisfy the constraints equations, the Kuhn - Tucker necessary conditions are used. Only under special circumstances are sure of the existence of single global minimum because the problem defined above may have several local minima. These conditions are written in terms of the Lagrangian Function for the problem. To start with, we consider the special case of equality constraints only. Using the Lagrange multiplier technique, we define the Lagrangian function as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{j=1}^{ne} \boldsymbol{\lambda}_j h_j(\mathbf{x})$$

where $\boldsymbol{\lambda}_j$ are unknown Lagrangian multipliers. The necessary conditions for a stationary point are

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{x}_i} &= \frac{\partial f}{\partial \mathbf{x}_i} - \sum_{j=1}^{ne} \boldsymbol{\lambda}_j \frac{\partial h_j}{\partial \mathbf{x}_i} = 0; \quad i = 1, \cdots, n \\
\frac{\partial L}{\partial \boldsymbol{\lambda}_j} &= h_j(\mathbf{x}) = 0; \quad j = 1, \cdots, ne
\end{aligned}$$

These conditions, however, apply only at a **regular point**, that is, at a point where the gradients of the constraints are linearly independent. If we have constraints gradients that are linearly dependent, it means that we can remove some constraints without affecting the solution. At a regular point, these two equations represent $n + ne$ equations for the ne Lagrange multipliers and the n coordinates of the stationary point. The situation is somewhat more complicated when inequality constraints are present. To be able to apply the Lagrange multiplier method, we first transform the inequality constraints to equality constraints by adding the appropriate nonnegative **slack variables**. That is, the inequality constraints are written as

$$g_j(\mathbf{x}) - \mathbf{t}_j^2 = 0, \quad j = 1, \cdots, n_g$$

where $\mathbf{t}_j$ is a slack variable which measures how far the jth constraint is from being critical. We can now form a Lagrangian function as

$$L(\mathbf{x}, \mathbf{t}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{j=1}^{n_g} \boldsymbol{\lambda}_j (g_j - \mathbf{t}_j^2).$$

Differentiating the Lagrangian function with respect to $\mathbf{x}, \boldsymbol{\lambda}, \mathbf{t}$ we obtain

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{x}_i} &= \frac{\partial f}{\partial \mathbf{x}_i} - \sum_{j=1}^{n_g} \boldsymbol{\lambda}_g \frac{\partial g_j}{\partial \mathbf{x}_i} = 0, \quad i = 1, \cdots, n \\
\frac{\partial L}{\partial \boldsymbol{\lambda}_j} &= -g_j + \mathbf{t}_j^2 = 0, \quad j = 1, \cdots, n_g
\end{aligned}$$

$$\frac{\partial L}{\partial \mathbf{t}_j} = 2\boldsymbol{\lambda}_j \mathbf{t}_j = 0, \quad j = 1, \cdots, n_g$$

The last two equations imply that when an inequality constraint is not critical (so that the corresponding slack variable is non - zero) then the Lagrange multiplier associated with the constraint is zero. These three equations are the necessary conditions for a stationary regular point. We should note that for inequality constraints, a regular point is one where the gradients of the active constraints are linearly independent. These conditions are modified slightly to yield the necessary conditions for a minimum which is known as the Kuhn - Tucker conditions. The Kuhn - Tucker conditions may be summarized as follows:

A point $\mathbf{x}$ is a local minimum of an inequality constrained problem only if a set of nonnegative $\boldsymbol{\lambda}_j$'s may be found such that:

1. $$\frac{\partial L}{\partial \mathbf{x}_i} = \frac{\partial f}{\partial \mathbf{x}_i} - \sum_{j=1}^{n_g} \boldsymbol{\lambda}_g \frac{\partial g_j}{\partial \mathbf{x}_i} = 0, \quad i = 1, \cdots, n$$
   is satisfied.

2. The corresponding $\boldsymbol{\lambda}_j$ is zero if a constraint is not active.

Example 4.7. Locate the five Kuhn - Tucker points of the following problem, and determine their character, that is, maximum, minimum or saddle point.

$$\min\ f(\mathbf{x}) = x_1 x_2$$

$$\begin{aligned} \text{subject to :} x_1 + x_2 &\leq 1 \\ -x_1 + x_2 &\leq 1 \\ -x_1 - x_2 &\leq 1 \\ x_1 - x_2 &\leq 1 \end{aligned}$$

The function being optimized is the classic saddle point function which is constrained by planes. The first step in the procedure is to locate the stationary points by ignoring the inequality constraints, that is, $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$. If this point satisfies the constraints as inequalities, an optimum may have been found. For this problem:

$$\frac{\partial y}{\partial x_1} = x_2 = 0$$
$$\frac{\partial y}{\partial x_2} = x_1 = 0$$

The Kuhn-Tucker point is $x_0 = [0; 0]$, and evaluating its character by the unconstrained sufficiency conditions gives the following result.

$$\frac{\partial^2 y}{\partial x_1^2} = 0, \qquad \frac{\partial^2 y}{\partial x_2^2} = 0$$
$$\frac{\partial^2 y}{\partial x_1 x_2} = 1, \qquad \frac{\partial^2 y}{\partial x_2 x_1} = 1$$

and

$$\mid \mathbf{H}_1 \mid = 0, \qquad \mid \mathbf{H}_2 \mid = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} = -1$$

The point $x_0 = [0; 0]$ is a saddle point, and the constraints are satisfied. One constraint equation at a time is selected, and the character of the Kuhn-Tucker point is determined. Beginning with the first constraint equation as an inequality, that is, $\lambda_1 \neq 0$ and considering the other three as equalities, that is, $\lambda_2 = \lambda_3 = \lambda_4 = 0$, gives the following equation for the Lagrangian function

$$L(x_1, x_2, \lambda_1) = x_1 x_2 + \lambda_1(x_1 + x_2 - 1)$$

and

$$\begin{aligned} \frac{\partial L}{\partial x_1} &= x_2 + \lambda_1 = 0, \\ \frac{\partial L}{\partial x_2} &= x_1 + \lambda_1 = 0, \\ \frac{\partial L}{\partial \lambda} &= x_1 + x_2 - 1 = 0 \end{aligned}$$

solving gives:

$$x_1 = \frac{1}{2}, \quad x_2 = -\frac{1}{2}, \quad \lambda = -\frac{1}{2} \qquad y(\frac{1}{2}, \frac{1}{2}) = \frac{1}{4}$$

The sign of the Lagrange Multiplier is negative, and by the Kuhn-Tucker necessary conditions, the point can be a maximum, since the other constraint equations are satisfied as inequalities. The procedure is repeated for the other three constraint equations, each considered individually as an equality. The results for the Kuhn-Tucker points are summarized in the following table:

Table 9: Kuhn-Tucker results

Character:	max	min	max	min	saddle
y:	$\frac{1}{4}$	-$\frac{1}{4}$	$\frac{1}{4}$	-$\frac{1}{4}$	0
x_1:	$\frac{1}{2}$	-$\frac{1}{2}$	-$\frac{1}{2}$	$\frac{1}{2}$	0
x_2:	$\frac{1}{2}$	$\frac{1}{2}$	-$\frac{1}{2}$	-$\frac{1}{2}$	0
λ :	-$\frac{1}{2}$	$\frac{1}{2}$	-$\frac{1}{2}$	$\frac{1}{2}$	0

The character of each of the stationary points is based on the Kuhn-Tucker necessary conditions. However, constraint qualifications and sufficient conditions are needed to give a general method.

In the theorems developed by Kuhn and Tucker (1951), the constraint equations must satisfy certain conditions at the Kuhn-Tucker points, and these conditions are called **constraint qualifications**. As given in Bazaraa and Shetty (1993), there are several forms of constraint qualifications; and one, according to Gill (1974b), is important for nonlinear constraints. This is the condition that the gradients of the constraint equations at the Kuhn-Tucker point are linearly independent. This constraint qualification is required for the necessary conditions given by Kuhn - Tucker. The same concepts used for unconstrained problems are followed to develop the sufficient conditions for constrained problems. This involves expanding the Lagrangian function in a Taylor series about the Kuhn-Tucker point located using the necessary conditions. The Taylor series is simplified by neglecting third and higher order terms to give a function that contains only terms involving second partial derivatives evaluated at the Kuhn-Tucker point. This gives a differential quadratic form, and a test similar to the one for the unconstrained problem is obtained to determine if the Kuhn-Tucker point is a maximum, minimum, or saddle. The sufficient conditions for the case of both inequality and equality constraints are more elaborate than if only equality constraints are involved.

A geometrical interpretation of the Kuhn - Tucker conditions is illustrated in Figure 9 for the case of two constraints. ∇g_1 and ∇g_2 denote the gradients of the two constraints which are orthogonal to the respective constraint surfaces. The vector **S** shows a typical feasible direction which does not lead immediately to any constraint violation. For the two constraints case we get

$$-\nabla f = -(\lambda_1 \nabla g_1 + \lambda_2 \nabla g_2).$$

To improve upon the design we proceed from A in a direction **S** that is usable and feasible.

To be usable, a small move along the direction should decrease the objective function, so it must form an acute angle with $-\nabla f$. To be feasible, **S** should form an obtuse angle with $-\nabla g_1$ and $-\nabla g_2$.

From the figure, any vector which forms an acute angle with $-\nabla f$ will also form an acute angle with either $-\nabla g_1$ and $-\nabla g_2$. Thus, Kuhn - Tucker conditions mean that no feasible design with reduced objective function is to be found in the neighbourhood of A.

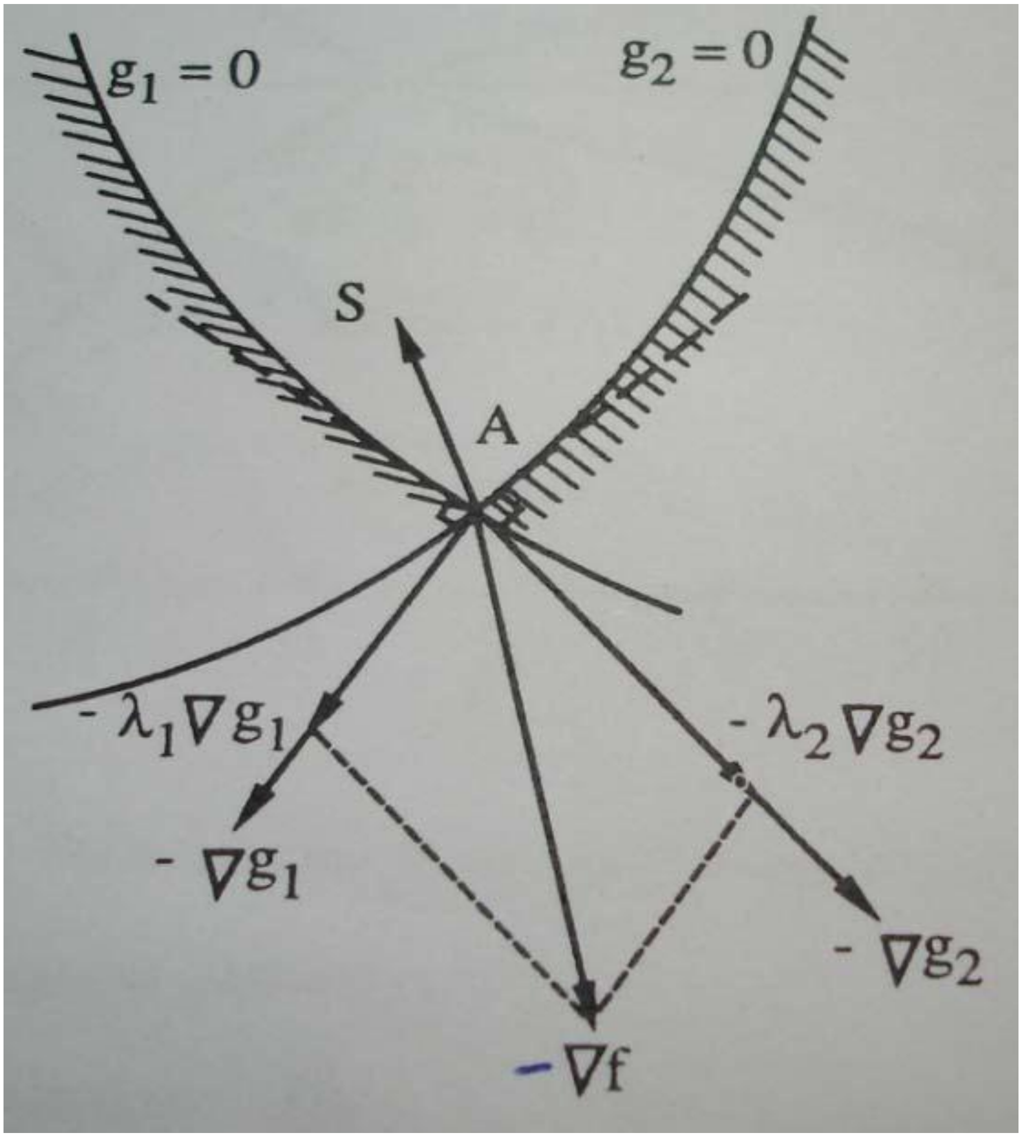

Figure 9: Geometric representation of the Kuhn-Tucker conditions for two constraints

Mathematically, the condition that a direction $\mathbf{S}$ be feasible is written as

$$\mathbf{S}^T\nabla g_j \geq 0, \ j \in \mathbf{I}_A$$

where $\mathbf{I}_A$ is the set of active constraints. Equality is permitted only for linear or concave constraints. The condition for a usable direction (one that decreases the objective function) is

$$\mathbf{S}^T\nabla f < 0.$$

Multiplying by $\mathbf{S}_i$ and summing over i we obtain,

$$\mathbf{S}^T\nabla f = \sum_{j=1}^{n_g} \lambda_j \mathbf{S}^T\nabla g_j.$$

So, it is impossible if the λ_j's are positive. If the Kuhn-Tucker conditions are satisfied at a point it is impossible to find a direction with a negative slope for objective function that does not violate the constraints. In some cases, although it is possible to move in a direction which is tangent to the active constraints and perpendicular to the gradient, that is,

$$\mathbf{S}^T\nabla f = \mathbf{S}^T\nabla g_j = 0, \ j \in \mathbf{I}_A,$$

the effect of such a move on the objective function and constraints can be determined only from higher derivatives. In some cases, a move in this direction could reduce the objective function without violating the constraints even though the Kuhn-Tucker conditions are met. Therefore, the Kuhn-Tucker conditions are necessary but not sufficient for optimality.

The Kuhn - Tucker conditions are sufficient when the number of active constraints is equal to the number of design variables. When the number of active constraints is not equal to the number of design variables, sufficient conditions for optimality require the second derivatives of the objective function and constraints. A sufficient condition for optimality is that the Hessian matrix of the Lagrangian function is positive definite in the subspace tangent to the active constraints. For example, if we take the case of equality constraints, the Hessian matrix of the Lagrangian is

$$\nabla^2 L \equiv \nabla^2 f - \sum_{j=1}^{ne} \lambda_j \nabla^2 h_j$$

The sufficient condition for optimality is that

$$\mathbf{S}^T(\nabla^2 L)\mathbf{S} > 0$$

for all $\mathbf{S}$ for which $\mathbf{S}^T \nabla h_j = 0, \quad j = 1, \cdots, ne$. When inequality constraints are present, the vector S also needs to be orthogonal to the gradients of the active constraints with positive Lagrange multipliers. For active constraints with zero Lagrange multipliers,$\mathbf{S}$ must satisfy

$$\mathbf{S}^T \nabla g_j \geq 0,$$

when $g_j = 0$ and $\lambda_j = 0$

Example 4.8. Consider the problem of minimizing the function

$$f(\mathbf{x}) = -x_1^3 - 2x_2^2 + 10x_1 - 6 - 2x_2^3$$

subject to

$$\begin{aligned} g_1 = 10 - x_1 x_2 &\geq 0, \\ g_2 = x_1 &\geq 0, \\ g_3 = 10 - x_2 &\geq 0 \end{aligned}$$

The Kuhn - Tucker conditions are

$-3x_1^2 + 10 + \lambda_1 x_2 - \lambda_2 = 0$

$-4x_2 - 6x_2^2 + \lambda_1 x_1 + \lambda_3 = 0$

We have to check for all possibilities of active constraints. The simplest case is when no constraints are active, $\lambda_1 = \lambda_2 = \lambda_3 = 0$. We get $x_1 = 1.826,\ x_2 = 0,\ f = 6.17$. The Hessian matrix of the Lagrangian

$$\nabla^2 L = \begin{bmatrix} -6x_1 & \lambda_1 \\ \lambda_1 & -4 - 12x_2 \end{bmatrix}$$

is clearly negative definite, so that this point is a maximum. We assume that the first constraints is active, $x_1 x_2 = 10$, so that $x_1 \neq 0$ and g_2 is inactive and therefore $\lambda_2 = 0$. We have two possibilities for the third constraint. If it is active, we get $x_1 = 1,\ x_2 = 10,\ \lambda_1 = -0.7,\ \lambda_3 = 639.3$ so this point is neither a minimum nor a maximum. If the third constraint is not active, $\lambda_3 = 0$, we obtain the following equations

$$\begin{aligned} -3x_1^2 + 10 + \lambda_1 x_2 &= 0 \\ -4x_2 - 6x_2^2 + \lambda_1 x_1 &= 0 \\ x_1 x_2 &= 0 \end{aligned}$$

The only solution for these equations that satisfies the constraints on x_1 and x_2 is $x_1 = 3.847,\ x_2 = 2.599,\ \lambda_1 = 13.24,\ f = -73.08$. This point satisfies the Kuhn - Tucker condition for a minimum. However, the Hessian of the Lagrangian at that point

$$\nabla^2 L = \begin{bmatrix} -23.08 & 13.24 \\ 13.24 & -35.19 \end{bmatrix}$$

is negative definite, so that it cannot satisfy the sufficiency condition. In fact, an examination of the function f at neighbouring points along $x_1 x_2 = 10$ reveals that the point is not a minimum.

Next we consider the possibility that g_1 is not active, so that $\lambda_1 = 0$, and

$$\begin{aligned} -3x_1^2 + 10 - \lambda_2 &= 0 \\ -4x_2 - 6x_2^2 + \lambda_3 &= 0 \end{aligned}$$

We have already considered the possibility of both λ's being zero, so we need to consider only three possibilities of one of these Lagrange multipliers being nonzero, or both being nonzero. The first case is $\lambda_2 \neq 0,\ \lambda_3 = 0$, so that $g_2 = 0$, and we get

$x_1 = 0, x_2 = 0, \lambda_2 = 10, f = -6$ or $x_1 = 0, x_2 = -\frac{2}{3}, \lambda_2 = 10, f = -6.99$.

Both points satisfy the Kuhn- Tucker conditions for a minimum, but not the sufficiency conditions. In fact, the vectors tangent to the active constraints ($x_1 = 0$ is the only one) have the form $s^T = (0, a)$, and it is easy to check that $s^T \nabla^2 L s < 0$. It is also easy to check that these points are indeed no minima by reducing x_2 slightly.

The next case is $\lambda_2 = 0, \lambda_3 \neq 0$, so that $g_3 = 0$. We get

$x_1 = 1.826, x_2 = 10, \lambda_3 = 640, f = -2194$

This point satisfies the Kuhn -Tucker conditions, but it is not a minimum either. It is easy to check that $\nabla^2 L$ is negative definite in this case, so that the sufficiency condition could not be satisfied.

Finally, we consider the case $x_1 = 0, x_2 = 10, \lambda_2 = 10, \lambda_3 = 640$, $f = -2206$. Now,the Kuhn -Tucker conditions are satisfied, and the number of active constraints is equal to the number of design variables, so that it is the minimum.

Convex Problems

There is a class of problems namely convex problems, for which the Kuhn-Tucker conditions are not only necessary but also sufficient for a global minimum. A set of points $\mathbf{S}$ is convex whenever the entire line segment connecting two points that are in $\mathbf{S}$ is also in $\mathbf{S}$. That is, If $x_1 x_2 \in \mathbf{S}$, then $\alpha x_1 + (1-\alpha)x_2 \in \mathbf{S}, \quad 0 < \alpha < 1$

A function is convex if

$$f[\alpha x_2 + (1-\alpha)x_1] \leq \alpha f(x_2) + (1-\alpha) f(x_1), \quad 0 < \alpha < 1$$

It can be shown that a function of n variables is convex if its matrix of second derivatives is positive semi-definite. A convex optimization problem has a convex objective function and a convex feasible domain. It can be shown that the feasible domain is convex if all the inequality constraints $\mathbf{g}_j$ are concave and the equality constraints are linear. A convex optimization problem has only one minimum, and the Kuhn-Tucker conditions are sufficient to establish it. Most optimization problems encountered in practice cannot be shown to be convex. However, the theory of convex programming is still very important in structural optimization, as we often approximate optimization problems by a series of convex approximations.

Constrained Nonlinear Algorithms

Consider the constrained nonlinear programming problem P as

$$\begin{gathered} P : \min_{\mathbf{x}} \; f(\mathbf{x}) \\ \text{subject to } h_i(\mathbf{x}) = 0 \;\; i = 1, \cdots, k \\ g_i(\mathbf{x}) \leq 0 \;\; i = 1, \cdots, m \end{gathered} \tag{4.5}$$

whose feasible region we will denote as

$$\mathbb{F} := \{\mathbf{x} \in \mathbb{R}^n | g_i(\mathbf{x}) \leq 0, \; i = 1, \cdots, m, \; h_i(\mathbf{x}) = 0, \; i = 1, \cdots, k\}.$$

Barrier and Penalty methods are designed to solve P by instead solving a sequence of specially constructed unconstrained optimization problems.

In a penalty method, the feasible region of P is expanded from $\mathbb{F}$ to all of $\mathbb{R}^n$, but a large cost or "penalty" is added to the objective function for points that lie outside of the original feasible region $\mathbb{F}$.

In a barrier method, we presume that we are given a point $\mathbf{x}_0$ that lies in the interior of the feasible region $\mathbb{F}$, and we impose a very large cost on feasible points that lie ever closer to the boundary of $\mathbb{F}$, thereby creating a "barrier" to exiting feasible region.

Penalty Methods

Consider the constrained optimization problem P:

$$P : \min_{\mathbf{x}} \ f(\mathbf{x})$$

$$\text{subject to } h_i(\mathbf{x}) = 0 \ \ i = 1, \cdots, k$$

$$g_i(\mathbf{x}) \leq 0 \ \ i = 1, \cdots, m$$

By converting the constraints "$h_i(\mathbf{x}) = 0$" to "$h_i(\mathbf{x}) \leq 0, \ -h_i(\mathbf{x}) \leq 0$," we can assume that P is of the form

$$P : \min_{\mathbf{x}} \ f(\mathbf{x})$$

$$\text{subject to } g(\mathbf{x}) \leq 0,$$

where we write $g(\mathbf{x}) := (g_1(\mathbf{x}), \cdots, g_m(\mathbf{x}))^T$ for convenience.

Definition 4.2. A function $p(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is called a **penalty function** for P if $p(\mathbf{x})$ satisfies:

1. $p(\mathbf{x}) = 0$ if $g(\mathbf{x}) \leq 0$ and

2. $p(\mathbf{x}) > 0$ if $g(\mathbf{x}) \nleq 0$.

Example 4.9.

$$p(\mathbf{x}) = \sum_{i=1}^{m} (\max\{0, g_i(\mathbf{x})\})^2.$$

We then consider solving the following **penalty program**:

$$P(c) : \min_{x} \ f(\mathbf{x}) + cp(\mathbf{x})$$

$$\text{subject to } \mathbf{x} \in \mathbb{R}^n$$

for an increasing sequence of constants c as $c \to +\infty$. We should note that in the program $P(c)$ we are assigning a penalty to the violated constraints. The scalar quantity c is called the **penalty parameter**. Let $c_k \geq 0, \ k = 1, \cdots, \infty$, be a sequence of penalty parameters that satisfies $c_{k+1} > c_k$ for all k and

$$\lim_{k \to \infty} c_k = +\infty.$$

Let $q(c, \mathbf{x}) = f(\mathbf{x}) + cp(\mathbf{x})$, and let $\mathbf{x}_k$ be the exact solution to the program $P(c_k)$, and let $\mathbf{x}^*$ denote any optimal solution of P.

The following Lemma presents some basic properties of penalty methods:

Lemma 4.1. 1. $q(c_k, \mathbf{x}_k) \leq q(c_{k+1}, \mathbf{x}_{k+1})$

2. $p(\mathbf{x}_k) \geq p(\mathbf{x}_{k+1})$

3. $f(\mathbf{x}_k) \leq f(\mathbf{x}_{k+1})$

4. $f(\mathbf{x}^*) \geq q(c_k, \mathbf{x}_k) \geq f(\mathbf{x}_k)$

Proof. 1. We have

$$\begin{aligned} q(c_{k+1}, \mathbf{x}_{k+1}) &= f(\mathbf{x}_{k+1}) + c_{k+1}p(\mathbf{x}_{k+1}) \\ &\geq f(\mathbf{x}_{k+1}) + c_k p(\mathbf{x}_{k+1}) \\ &\geq f(\mathbf{x}_k) + c_k p(\mathbf{x}_k) \\ &= q(c_k, \mathbf{x}_k) \end{aligned}$$

2.
$$f(\mathbf{x}_k) + c_k p(\mathbf{x}_k) \leq f(\mathbf{x}_{k+1}) + c_k p(\mathbf{x}_{k+1})$$

and

$$f(\mathbf{x}_{k+1}) + c_{k+1} p(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + c_{k+1} p(\mathbf{x}_k)$$

Thus,

$$(c_{k+1} - c_k)p(\mathbf{x}_k) \geq (c_{k+1} - c_k)p(\mathbf{x}_{k+1}),$$

whereby $p(\mathbf{x}_k) \geq p(\mathbf{x}_{k+1})$.

3. From the proof of (1),

$$f(\mathbf{x}_{k+1}) + c_k p(\mathbf{x}_{k+1}) \geq f(\mathbf{x}_k) + c_k p(\mathbf{x}_k).$$

But $p(\mathbf{x}_k) \geq p(\mathbf{x}_{k+1})$, which implies that$f(\mathbf{x}_{k+1}) \geq f(\mathbf{x}_k)$.

4. $f(\mathbf{x}_k) \leq f(\mathbf{x}_k) + c_k p(\mathbf{x}_k) \leq f(\mathbf{x}^*) + c_k p(\mathbf{x}^*) = f(\mathbf{x}^*)$.

Theorem 4.2. Penalty Convergence Theorem

Suppose that $f(\mathbf{x}), g(\mathbf{x})$ and $p(\mathbf{x})$ are continuous functions. Let $\{\mathbf{x}_k\}$, $k = 1, \cdots, \infty$, be a sequence of solutions to $P(c_k)$. Then any limit point $\bar{\mathbf{x}}$ of $\{\mathbf{x}_k\}$ solves P.

Proof. Let $\bar{\mathbf{x}}$ be a limit point of $\{\mathbf{x}_k\}$. From the continuity of the functions involved,

$$\lim_{k\to\infty} f(\mathbf{x}_k) = f(\bar{\mathbf{x}}).$$

Also, from the Penalty Lemma,

$$q^* := \lim_{k\to\infty} q(c_k, \mathbf{x}_k) \leq f(\mathbf{x}^*).$$

Thus,

$$\lim_{k\to\infty} c_k p(\mathbf{x}_k) = \lim_{k\to\infty} [q(c_k, \mathbf{x}_k) - f(\mathbf{x}_k)] = q^* - f(\bar{\mathbf{x}}).$$

But $c_k \to \infty$, which implies from the above that

$$\lim_{k\to\infty} p(\mathbf{x}_k) = 0.$$

Therefore, from the continuity of $p(\mathbf{x})$ and $g(\mathbf{x})$, $p(\bar{\mathbf{x}}) = 0$, and so $g(\bar{\mathbf{x}}) \leq 0$, that is, $\bar{\mathbf{x}}$ is a feasible solution of P. Finally, from the Penalty Lemma, $f(\mathbf{x}_k) \leq f(\mathbf{x}^*)$ for all k, and so $f(\bar{\mathbf{x}}) \leq f(\mathbf{x}^*)$, which implies that $\bar{\mathbf{x}}$ is an optimal solution of P.

An often -used class of penalty function is:

$$\mathrm{p}(\mathbf{x}) = \sum_{\mathrm{i}=1}^{\mathrm{m}} [\max\{0, \mathrm{g_i}(\mathbf{x})\}]^{\mathrm{q}}, \text{ where } q \geq 1. \tag{4.6}$$

We note the following:

1. If $q = 1$, $p(\mathbf{x})$ in (1) is called the "linear penalty function". This function may not be differentiable at points where $g_i(\mathbf{x}) = 0$ for some i.
2. Setting $q = 2$ is the most common form of (1) that is used in practice, and is called the "quadratic penalty function".
3. If we denote

$$g^+(\mathbf{x}) = (\max\{0, g_i(\mathbf{x})\}, \cdots, \max\{0, g_m(\mathbf{x})\})^T,$$

then the quadratic penalty function can be written as

$$p(\mathbf{x}) = (g^+(\mathbf{x}))^T (g^+(\mathbf{x})).$$

Kuhn - Tucker Multipliers in penalty Methods

The penalty function $p(\mathbf{x})$ is in actuality a function only of $g^+(\mathbf{x})$, where $g^+(\mathbf{x}) = max\{0, g_i(\mathbf{x})\}$, (the nonnegative part of $g_i(\mathbf{x})$), $i = 1, \cdots, m$. Then we can write $p(\mathbf{x}) = \gamma(g^+(\mathbf{x}))$, where $\gamma(\mathbf{y})$ is a function of $\mathbf{y} \in (\mathbb{R}^m)^+$.

Two examples of this type of penalty function are

$$\gamma(\mathbf{y}) = \sum_{i=1}^{m} \mathbf{y}_i,$$

which corresponds to the linear penalty function, and

$$\gamma(\mathbf{y}) = \sum_{i=1}^{m} \mathbf{y}_i^2,$$

which corresponds to the quadratic penalty function.

We should note that even if $\gamma(\mathbf{y})$ is continuously differentiable, $p(\mathbf{x})$ might not be continuously differentiable, since $g^+(\mathbf{x})$ is not differentiable at points x where $g_i^+(\mathbf{x}) = 0$ for some i. However, if we assume the following:

$$\frac{\partial\gamma(\mathbf{y})}{\partial\mathbf{y}_i} = 0 \quad at\ i = 1, \cdots, m \tag{4.7}$$

then $p(\mathbf{x})$ is differentiable whenever the functions $g_i(\mathbf{x})$ are differentiable, $i = 1, \cdots, m$, and we can write

$$\nabla p(\mathbf{x}) = \sum_{i=1}^{m} \frac{\partial_\gamma(g^+(\mathbf{x}))}{\partial\mathbf{y}_i} \nabla g_i(\mathbf{x}). \tag{4.8}$$

Now, let $\mathbf{x}_k$ solve $P(c_k)$. Then, $\mathbf{x}_k$ will satisfy

$$\nabla f(\mathbf{x}_k) + c_k \nabla p(\mathbf{x}_k) = 0,$$

that is,

$$\nabla f(\mathbf{x}_k) + c_k \sum_{i=1}^{m} \frac{\partial_\gamma(g^+(\mathbf{x}_k))}{\partial\mathbf{y}_i} \nabla g_i(\mathbf{x}_k).$$

Let us define

$$\mathbf{u}_i^k = c_k \frac{\partial_\gamma(g^+(\mathbf{x}_k))}{\partial\mathbf{y}_i}. \tag{4.9}$$

Then

$$\nabla f(\mathbf{x}_k) + \sum_{i=1}^{m} \mathbf{u}_i^k \nabla g_i(\mathbf{x}_k) = 0,$$

and so we can interpret the $\mathbf{u}_k$ as a sort of vector of Kuhn - Tucker multipliers.

Lemma 4.2. Suppose $\gamma(\mathbf{y})$ is continuously differentiable and satisfies Equation 4.7, and that $f(\mathbf{x})$, and $g(\mathbf{x})$ are differentiable. Let $\mathbf{u}_k$ be defined by Equation 4.9. Then, if $\mathbf{x}_k \to \bar{\mathbf{x}}$, and $\bar{\mathbf{x}}$ satisfies the linear independence condition for gradient vectors of active constraints, then $\mathbf{u}_k \to \bar{\mathbf{u}}$, where $\bar{\mathbf{u}}$ is a vector of Kuhn - Tucker multipliers for the optimal solution $\bar{\mathbf{x}}$ of P.

Proof. From the penalty Convergence Theorem, $\bar{\mathbf{x}}$ is an optimal solution of P. Let $\mathbf{I} = \{i \mid g_i(\bar{\mathbf{x}}) = 0\}$ and $\mathbf{N} = \{i \mid g_i(\bar{\mathbf{x}}) < 0\}$. For $i \in \mathbf{N},\ g_i(\mathbf{x}_k) < 0$ for all k sufficiently large, so $\mathbf{u}_i^k = 0$ for all k sufficiently large, whereby $\bar{\mathbf{u}}_i = 0$ for $i \in \mathbf{N}$. From Equation 4.9 and the definition of a penalty function, it follows that $\mathbf{u}_i^k \geq 0$ for $i \in \mathbf{I}$, for all k sufficiently large. Suppose $\mathbf{u}_k \to \bar{\mathbf{u}}$ as $k \to \infty$. Then, $\bar{\mathbf{u}}_i = 0$ for $i \in \mathbf{N}$. From the continuity of all functions involved,

$$\nabla f(\mathbf{x}_k) + \sum_{i=1}^{m} \mathbf{u}_i^k \nabla g(\mathbf{x}_k) = 0$$

implies

$$\nabla f(\bar{\mathbf{x}}) + \sum_{i=1}^{m} \bar{\mathbf{u}}_i \nabla g(\bar{\mathbf{x}}) = 0.$$

From the above remarks, we also have $\bar{\mathbf{u}} \geq 0$ and $\bar{\mathbf{u}}_i = 0$ for all $i \in \mathbf{N}$. Thus, $\bar{\mathbf{u}}$ is a vector of Kuhn - Tucker multipliers. It therefore remains to show that $\mathbf{u}_k \to \bar{\mathbf{u}}$ for some unique $\bar{\mathbf{u}}$.

Suppose $\{\mathbf{u}_k\}_{k=1}^{\infty}$ has no accumulation point. Then $\|\mathbf{u}_k\| \to \infty$. But then define

$$\mathbf{v}_k = \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|},$$

and then $\|\mathbf{v}_k\| = 1$ for all k, and so the sequence $\{\mathbf{v}_k\}_{k=1}^{\infty}$ has some accumulation point $\bar{\mathbf{v}}$. For all $i \in \mathbf{N}$, $\mathbf{v}_i^k = 0$ for k large, whereby $\bar{\mathbf{v}}_i = 0$ for $i \in \mathbf{N}$, and

$$\sum_{i \in I} \mathbf{v}_i^k \nabla g_i(\mathbf{x}_k) = \sum_{i=1}^{m} \mathbf{v}_i^k \nabla g_i(\mathbf{x}_k) = \sum_{i=1}^{m} (\frac{\mathbf{u}_i^k}{\|\mathbf{u}_k\|}) \nabla g_i(\mathbf{x}_k) = \frac{-\nabla f(\mathbf{x}_k)}{\|\mathbf{u}_k\|}$$

for k large. As $k \to \infty$, we have $\mathbf{x}_k \to \bar{\mathbf{x}}$, $\mathbf{v}_k \to \bar{\mathbf{v}}$ and $\|\mathbf{u}_k\| \to \infty$, and so the above becomes

$$\sum_{i \in I} \bar{\mathbf{v}}_i \nabla g_i(\bar{\mathbf{x}}) = 0,$$

and $\|\bar{\mathbf{v}}\| = 1$, which violates the linear independence condition. Therefore, $\{\mathbf{u}_k\}$ is a bounded sequence, and so has at least one accumulation point. Now suppose that $\{\mathbf{u}_k\}$ has two accumulation points, $\tilde{\mathbf{u}}$ and $\bar{\mathbf{u}}$. Note that $\bar{\mathbf{u}}_i = 0$ and $\tilde{\mathbf{u}}_i = 0$ for $i \in \mathbf{N}$, and so

$$\sum_{i \in I} \bar{\mathbf{u}}_i \nabla g_i(\bar{\mathbf{x}}) = -\nabla f(\bar{\mathbf{x}}) = \sum_{i \in I} \tilde{\mathbf{u}}_i \nabla g_i(\bar{\mathbf{x}}),$$

so that

$$\sum_{i \in I} (\bar{\mathbf{u}}_i - \tilde{\mathbf{u}}_i) \nabla g_i(\bar{\mathbf{x}}) = 0.$$

But by the linear independence condition, $\bar{\mathbf{u}}_i - \tilde{\mathbf{u}}_i = 0$ for all $i \in I$, and so $\bar{\mathbf{u}}_i = \tilde{\mathbf{u}}_i$. This then implies that $\bar{\mathbf{u}} = \tilde{\mathbf{u}}$.

Exact Penalty Methods

The idea of an **exact penalty method** is to choose a penalty function $p(\mathbf{x})$ and a constant c so that the optimal solution $\tilde{\mathbf{x}}$ of $P(c)$ is also an optimal solution of the original problem P.

Theorem 4.3. Suppose P is a convex program for which the Kuhn - Tucker conditions are necessary. Suppose that

$$p(\mathbf{x}) := \sum_{i=1}^{m} (g_i(\mathbf{x}))^+.$$

Then as long as c is chosen sufficiently large, the sets of optimal solutions of $P(c)$ and P coincide. In fact, it suffices to choose $c > max_i\{\mathbf{u}_i^*\}$, where $\mathbf{u}^*$ is a vector of Kuhn - Tucker multipliers.

Proof. Suppose $\hat{\mathbf{x}}$ solves P. For any $\mathbf{x} \in \mathbb{R}^n$ we have:

$$\begin{aligned} q(c,\mathbf{x}) = f(\mathbf{x}) + c\sum_{i=1}^{m}(g_i(\mathbf{x}))^+ &\geq f(\mathbf{x}) + \sum_{i=1}^{m}(\mathbf{u}_i^* g_i(\mathbf{x}))^+ \\ &\geq f(\mathbf{x}) + \sum_{i=1}^{m}\mathbf{u}_i^* g_i(\mathbf{x}) \\ &\geq f(\mathbf{x}) + \sum_{i=1}^{m}\mathbf{u}_i^*(g_i(\hat{\mathbf{x}}) + \nabla g_i(\hat{\mathbf{x}})^T(\mathbf{x}-\hat{\mathbf{x}})) \\ &= f(\mathbf{x}) + \sum_{i=1}^{m}\mathbf{u}_i^*\nabla g_i(\hat{\mathbf{x}})^T(\mathbf{x}-\hat{\mathbf{x}}) \\ &= f(\mathbf{x}) - \nabla f(\hat{\mathbf{x}})^T(\mathbf{x}-\hat{\mathbf{x}}) \\ &\geq f(\hat{\mathbf{x}}) = f(\hat{\mathbf{x}}) + c\sum_{i=1}^{m}(g_i(\hat{\mathbf{x}}))^+ = q(c,\hat{\mathbf{x}}). \end{aligned}$$

Thus, $q(c,\hat{\mathbf{x}}) \leq q(c,\mathbf{x})$ for all $\mathbf{x}$, and therefore $\hat{\mathbf{x}}$ solves $P(c)$. Next suppose that $\bar{\mathbf{x}}$ solves $P(c)$. Then if $\hat{\mathbf{x}}$ solves P, we have:

$$f(\bar{\mathbf{x}}) + c\sum_{i=1}^{m}(g_i(\bar{\mathbf{x}}))^+ \leq f(\hat{\mathbf{x}}) + c\sum_{i=1}^{m}(g_i(\hat{\mathbf{x}}))^+ = f(\hat{\mathbf{x}}),$$

and so

$$f(\bar{\mathbf{x}}) \leq f(\hat{\mathbf{x}}) - c\sum_{i=1}^{m}(g_i(\bar{\mathbf{x}}))^+. \tag{4.10}$$

However, if $\bar{\mathbf{x}}$ is not feasible for P, then

$$\begin{aligned} f(\bar{\mathbf{x}}) \geq f(\hat{\mathbf{x}}) + \nabla f(\hat{\mathbf{x}})^T(\bar{\mathbf{x}}-\hat{\mathbf{x}})) &= f(\hat{\mathbf{x}}) - \sum_{i=1}^{m}\mathbf{u}_i^*\nabla g_i(\hat{\mathbf{x}})^T(\bar{\mathbf{x}}-\hat{\mathbf{x}}) \\ &\geq f(\hat{\mathbf{x}}) + \sum_{i=1}^{m}\mathbf{u}_i^*(g_i(\hat{\mathbf{x}}) - g_i(\bar{\mathbf{x}})) \\ &= f(\hat{\mathbf{x}}) - \sum_{i=1}^{m}\mathbf{u}_i^* g_i(\bar{\mathbf{x}}) > f(\hat{\mathbf{x}}) - c\sum_{i=1}^{m}(g_i(\bar{\mathbf{x}}))^+, \end{aligned}$$

which contradicts Equation 4.10. Thus, $\bar{\mathbf{x}}$ is feasible for P. So it implies that

$$f(\bar{\mathbf{x}}) \leq f(\hat{\mathbf{x}}) - c\sum_{i=1}^{m}(g_i(\bar{\mathbf{x}}))^+ = f(\hat{\mathbf{x}})$$

from Equation 4.10, and so $\bar{\mathbf{x}}$ solves P.

Penalty Methods for Inequality and Equality Constraints

The presentation of penalty methods has assumed either that the problem P has no equality constraints, or that the equality constraints have been converted to inequality constraints. For the latter, the conversion is easy to do, but the conversion usually violates good judgement in that it unnecessarily complicates the problem. Furthermore, it can cause the linear independence condition to be automatically violated for every feasible solution. Therefore, instead we consider the constrained optimization problem P with both inequality and equality constraints:

$$\begin{aligned} P : \min_{\mathbf{x}}\ f(\mathbf{x}) \\ \text{subject to } g(\mathbf{x}) \leq 0 \\ h(\mathbf{x}) = 0 \end{aligned}$$

where $g(\mathbf{x})$ and $h(\mathbf{x})$ are vector - valued functions, that is,

$$g(\mathbf{x}) := (g_1(\mathbf{x}), \cdots, g_m(\mathbf{x}))^T$$

and $h(\mathbf{x}) := (h_1(\mathbf{x}), \cdots, h_k(\mathbf{x}))^T$ for notational convenience.

Definition 4.3. A function $p(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is called a **penalty function** for P if $p(\mathbf{x})$ satisfies:

1. $p(\mathbf{x}) = 0$ if $g(\mathbf{x}) \leq 0$ and $h(\mathbf{x}) = 0$

2. $p(\mathbf{x}) > 0$ if $g(\mathbf{x}) \nleq 0$ and $h(\mathbf{x}) \neq 0$.

The main class of penalty functions for this general problem are of the form:

$$p(\mathbf{x}) = \sum_{i=1}^{m}[\max\{0, g_i(\mathbf{x})\}]^q + \sum_{i=1}^{k}|h_i(\mathbf{x})|^q, \text{ where } q \geq 1.$$

All of the results extend naturally to problems with equality constraints and for penalty functions of the above form.

Sequential Penalty Methods

The constrained optimization problem is considerably more difficult to solve than the unconstrained problem. There are several general approaches for attacking the problem (for example, see Luenberger 1965 and Zangwill 1969), but we will restrict our investigation to a single technique, the method of *penalty functions.* The virtue of penalty functions is that they allow us to build directly on our previous results by converting constrained optimization problem into equivalent unconstrained problems. The basic idea is to make the constraints implicit by adding terms to the objective function. These terms are zero when the constraints are satisfied but become large when the constraints are violated. Consequently, the minimization process itself tends to seek out feasible solutions because solutions that violate the constraints are penalized in the sense that they incur a large cost in the generalized objective function. To construct suitable penalty terms, we begin with the equality constraints. Suppose $P(\mathbf{x})$ denotes a penalty function associated with the constraint $p(\mathbf{x}) = 0$. The basic requirement on $P(\mathbf{x})$ is that $P(\mathbf{x}) = 0$ when $p(\mathbf{x}) = 0$ and $P(\mathbf{x}) > 0$ when $p(\mathbf{x}) \neq 0$. By this way, feasible solutions are not penalized, whereas unfeasible solutions that violate $p(\mathbf{x}) = 0$ are penalized. A simple $P(\mathbf{x})$ that satisfies these properties is $P(\mathbf{x}) = p^T(\mathbf{x})p(\mathbf{x})$, which can be written explicitly as

$$P(\mathbf{x}) = \sum_{k=1}^{r} p_k^2(\mathbf{x}). \tag{4.11}$$

$P(\mathbf{x})$ is not only positive when $p(\mathbf{x}) \neq 0$, but the value of $P(\mathbf{x})$ grows as the violation of the constraint becomes more severe. There are many other functions that also qualify as penalty functions. For example, the terms in Equation 4.11 can be replaced by $|p_k(\mathbf{x})|$. This tends to penalize minor violations of the constraint more severely, but the gradient of $P(\mathbf{x})$ is not continuous. Penalty functions for the inequality constraints can be constructed in a similar manner. If $Q(\mathbf{x})$ is a penalty function associated with the constraint, $q(\mathbf{x}) \geq 0$, then it is required that $Q(\mathbf{x}) = 0$ when $q(\mathbf{x}) \geq 0$ and $Q(\mathbf{x}) > 0$ otherwise. One such penalty function which satisfies these properties is

$$Q(\mathbf{x}) = \sum_{k=1}^{s} min^2\{0, q_k(\mathbf{x})\}. \tag{4.12}$$

$Q(\mathbf{x})$ is not only positive when the constraint is violated, but the value of $Q(\mathbf{x})$ increases as the violation becomes more severe. Once penalty functions are constructed for the equality and inequality constraints, they can then be incorporated into the generalized objective function, $F(\mathbf{x})$, as

$$F(\mathbf{x}) = f(\mathbf{x}) + \mu[P(\mathbf{x}) + Q(\mathbf{x})]. \tag{4.13}$$

The penalty parameter $\mu > 0$ controls the relative cost of violating the constraints. The original constrained optimization problem is then replaced by the following parameterized unconstrained optimization problem;

$$\begin{aligned} \text{minimize}: &\quad f(\mathbf{x}) + \mu[P(\mathbf{x}) + Q(\mathbf{x})] \\ \text{subject to}: &\quad \mathbf{x} \in \mathbb{R}^n \end{aligned}$$

As the penalty parameter $\mu > 0$ increases, solutions to the unconstrained minimization problem become increasingly good approximations to the solution of the original constrained optimization problem. However, large values of μ can cause the generalized objective function and its gradient to become very large, particularly if the initial guess $\mathbf{x}_0$, violates the constraints.

Penalty functions can be implemented using the following algorithm:

1. Choose a sequence $\{\mu_k\} \to \infty$, typically $\{1, 10, 10^2, 10^3, \cdots\}$.
2. For each μ_k find a local minimizer, $\mathbf{x}(\mu_k)$ say to $\min_{\mathbf{x}} f(\mathbf{x}, \mu_k)$.
3. Terminate when $\mathbf{c}(\mathbf{x}(\mu_k))$ is sufficiently small.

We illustrate the implementation of this algorithm using the following problem.

Example 4.10.

$$\min f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2$$

subject to

$$p(\mathbf{x}) = x_1 + x_2 - 2 = 0$$

$$q(\mathbf{x}) = x_2 - x_3 - 3 \geq 0$$

We solve this problem using the penalty functions in Equation 4.13. The summary of the solution using the algorithm described above with a starting point of $\mathbf{x}_0 = [0; 0; 0]$ is given in Table 4.2. In the table, the solution of the problem is given for each selected penalty parameter μ_k under various levels of constraints. It also provides the value of the function f and the two constraints $p(\mathbf{x})$ and $q(\mathbf{x})$ at the solution $\mathbf{x}_k = [x_1; x_2; x_3]$.

Table 10: Illustration of Constrained Optimization using Penalty Functions

μ_k	Constraints	x_1	x_2	x_3	$f(\mathbf{x}_k)$	$p(\mathbf{x}_k)$	$q(\mathbf{x}_k)$
1	None	1.00000	2.00000	3.00000	0.00000	1.00000	-4.00000
	Eq.	0.66667	1.66667	3.00000	0.22222	0.33330	-4.33300
	Inq.	1.00000	3.3333	1.66667	3.55560	2.33330	-1.33330
	Both	0.12500	2.75000	1.37500	3.96880	0.87500	-1.62500
10	None	1.00000	2.00000	3.00000	0.00000	1.00000	-4.00000
	Eq.	0.52381	1.52381	3.00000	0.45351	0.04762	-4.47620
	Inq.	1.00000	3.90480	1.09520	7.25620	2.90480	-0.19048
	Both	-0.78886	2.96774	0.24340	11.73500	0.17889	-0.27566
10^2	None	1.00000	2.00000	3.00000	0.00000	1.00000	-4.00000
	Eq.	0.50249	1.50249	3.00000	0.49504	0.00498	-4.49750
	Inq.	1.00000	3.99005	1.00995	7.92060	2.99000	-0.01990
	Both	-0.97691	2.99668	0.02641	13.744000	0.01977	-0.02974
$\vdots$				$\vdots$			
10^5	None	1.00000	2.00000	3.00000	0.00000	1.00000	-4.00000
	Eq.	0.50001	1.50000	2.99997	0.50000	0.00000	-4.50000
	Inq.	0.00000	1.50000	-1.50000	21.50000	-0.49990	0.00000
	Both	0.33334	1.66668	-1.33329	19.33300	0.00000	0.00000
10^6	None	1.00000	2.00000	3.00000	0.00000	1.00000	-4.00000
	Eq.	0.50000	1.50000	3.00000	0.50000	0.00000	-4.50000
	Inq.	0.00000	1.50000	-1.50000	21.50000	-0.50000	0.00000
	Both	0.33333	1.66667	-1.33333	19.33300	0.00000	0.00000
10^7	None	1.00000	2.00000	3.00000	0.00000	1.00000	-4.00000
	Eq.	0.52880	1.47120	2.82718	0.53153	0.00000	-4.35600
	Inq.	0.00000	1.50000	-1.50000	21.50000	-0.50000	0.00000
	Both	0.33333	1.66667	-1.33333	19.33300	0.00000	0.00000

The results in Table 10 for each μ_k were obtained after 10 iterations. Computations were done with a tolerance of 10^{-4}. Thus, we consider all values less than this tolerance to be zero. We see from the table that the solution of the problem does not change for μ_6 and μ_7. Thus, the solution of the problem has converged to the point

$$\mathbf{x}^* = [0.33333; 1.66667; -1.33333]$$

after a large penalty parameter $\mu_6 = 10^{-6}$. At $\mathbf{x}^*$ we realize that the values of both the equality and the inequality constraints are zero. This means that $\mathbf{x}^*$ is the solution of the minimization problem and also satisfies the constraints.

A number of other observations can be made from Table 10. We see that the results for all values of the penalty parameters are the same when there are no constraints. Particularly, the solution of the problem in this case is $[1; 2; 3]$. Thus it is of no relevance to choose a penalty parameter when there are no constraints. The problem then reduces to one of unconstrained optimization which could be solved using any of the unconstrained minimization techniques already discussed in the previous chapters.

Barrier Methods

Definition 4.4. A **barrier function** for P is any function $b(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ that satisfies

1. $b(\mathbf{x}) \geq 0$ for all $\mathbf{x}$ that satisfy $g(\mathbf{x}) < 0$, and

2. $b(\mathbf{x}) \to \infty$ as

$$\lim_{\mathbf{x}} (\max_i\{g_i(\mathbf{x})\}) \to 0.$$

The idea in barrier method is to dissuade points $\mathbf{x}$ from ever approaching the boundary of the feasible region. We consider solving:

$$B(c) : \text{minimize} f(\mathbf{x}) + \frac{1}{c} b(\mathbf{x})$$

$$\text{subject to } g(\mathbf{x}) < 0, \ \mathbf{x} \in \mathbb{R}^n$$

for a sequence of $c_k \to +\infty$. We should note that the constraints "$g(\mathbf{x}) < 0$" are effectively unimportant in $B(c)$, as they are never binding in $B(c)$.

Example 4.11.

$$b(\mathbf{x}) = \sum_{i=1}^{m} \frac{1}{-g_i(\mathbf{x})}$$

Suppose $g(\mathbf{x}) = [x_1 - 4; 1 - x_2], \ \mathbf{x} \in \mathbb{R}^2$. Then

$$b(\mathbf{x}) = \frac{1}{4 - x_1} + \frac{1}{x_2 - 1}.$$

Let $r(c, \mathbf{x}) = f(\mathbf{x}) + \frac{1}{c} b(\mathbf{x})$. Let the sequence $\{c_k\}$ satisfy $c_{k+1} > c_k$ and $c_k \to \infty$ as $k \to \infty$ and where $\mathbf{x}^k$ denote the exact solution to $B(c_k)$.

The following Lemma presents some basic properties of barrier methods.

Lemma 4.3. Barrier Lemma

1. $r(c_k, \mathbf{x}_k) \geq r(c_{k+1}, \mathbf{x}_{k+1})$
2. $b(\mathbf{x}_k) \leq b(\mathbf{x}_{k+1})$
3. $f(\mathbf{x}_k) \geq f(\mathbf{x}_{k+1})$
4. $f(\mathbf{x}^*) \leq f(\mathbf{x}_k) \leq r(c_k, \mathbf{x}_k)$

Proof. 1. $r(c_k, \mathbf{x}_k) = f(\mathbf{x}_k) + \frac{1}{c_k} b(\mathbf{x}_k) \geq f(\mathbf{x}_k) + \frac{1}{c_{k+1}} b(\mathbf{x}_k)$

$$r(c_k, \mathbf{x}_k) \geq f(\mathbf{x}_{k+1}) + \frac{1}{c_{k+1}} b(\mathbf{x}_{k+1}) = r(c_{k+1}, \mathbf{x}_{k+1})$$

2. $f(\mathbf{x}_k) + \frac{1}{c_k}b(\mathbf{x}_k) \leq f(\mathbf{x}_{k+1}) + \frac{1}{c_k}b(\mathbf{x}_{k+1})$ and

   $$f(\mathbf{x}_{k+1}) + \frac{1}{c_{k+1}}b(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \frac{1}{c_{k+1}}b(\mathbf{x}_k).$$

   Summing and rearranging, we have

   $$(\frac{1}{c_k} - \frac{1}{c_{k+1}})b(\mathbf{x}_k) \leq (\frac{1}{c_k} - \frac{1}{c_{k+1}})b(\mathbf{x}_{k+1})$$

   Since $c_k < c_{k+1}$, it follows that $b(\mathbf{x}_{k+1}) \geq b(\mathbf{x}_k)$.

3. From the proof of (1),

   $$f(\mathbf{x}_k) + \frac{1}{c_{k+1}}b(\mathbf{x}_k) \geq f(\mathbf{x}_{k+1}) + \frac{1}{c_{k+1}}b(\mathbf{x}_{k+1}).$$

   But from (2), $b(\mathbf{x}_k) \leq b(\mathbf{x}_{k+1})$. Thus, $f(\mathbf{x}_k) \geq f(\mathbf{x}_{k+1})$.

4. $f(\mathbf{x}^*) \leq f(\mathbf{x}_k) \leq f(\mathbf{x}_k) + \frac{1}{c_k}b(\mathbf{x}_k) = r(c_k, \mathbf{x}_k)$.

Theorem 4.4. Barrier Convergence Theorem

Suppose $f(\mathbf{x})$, $g(\mathbf{x})$ and $b(\mathbf{x})$ are continuous functions. Let $\{\mathbf{x}_k\},\ k = 1, \cdots, \infty$, be a sequence of solutions of $B(c_k)$. Suppose there exists an optimal solution $\mathbf{x}^*$ of P for which $N(\epsilon, \mathbf{x}^*) \cap \{\mathbf{x} |\ g(\mathbf{x}) < 0\} \neq 0$ for every $\epsilon > 0$ where $N(\epsilon, \mathbf{x})$ denote the ball of radius ϵ centered at the point $\mathbf{x}$. Then any limit point $\bar{\mathbf{x}}$ of $\{\mathbf{x}_k\}$ solves P.

Proof. Let $\bar{\mathbf{x}}$ be any limit point of the sequence $\{\mathbf{x}_k\}$. From the continuity of $f(\mathbf{x})$ and $g(\mathbf{x})$,

$$\lim_{k\to\infty} f(\mathbf{x}_k) = f(\bar{\mathbf{x}})$$

and

$$\lim_{k\to\infty} g(\mathbf{x}_k) = g(\bar{\mathbf{x}}) \leq 0.$$

Thus $\bar{\mathbf{x}}$ is feasible for P. For any $\epsilon > 0$, there exist $\tilde{\mathbf{x}}$ such that $g(\tilde{\mathbf{x}}) < 0$ and $f(\tilde{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$. For each k,

$$f(\mathbf{x}^*) + \epsilon + \frac{1}{c_k}b(\tilde{\mathbf{x}}) \geq f(\tilde{\mathbf{x}}) + \frac{1}{c_k}b(\tilde{\mathbf{x}}) \geq r(c_k, \mathbf{x}_k).$$

Therefore for k sufficiently large, $f(\mathbf{x}^*) + 2\epsilon \geq r(c_k, \mathbf{x}_k)$, and since $r(c_k, \mathbf{x}_k) \geq f(\mathbf{x}^*)$ from the 4th point of the Barrier Lemma, then

$$f(\mathbf{x}^*) + 2\epsilon \geq \lim_{k\to\infty} r(c_k, \mathbf{x}_k) \geq f(\mathbf{x}^*).$$

This implies that

$$\lim_{k\to\infty} r(c_k, \mathbf{x}_k) = f(\mathbf{x}^*).$$

We also have

$$f(\mathbf{x}^*) \leq f(\mathbf{x}_k) \leq f(\mathbf{x}_k) + \frac{1}{c_k} b(\mathbf{x}_k) = r(c_k, \mathbf{x}_k).$$

Taking limits we obtain

$$f(\mathbf{x}^*) \leq f(\bar{\mathbf{x}}) \leq f(\mathbf{x}^*),$$

whereby $\bar{\mathbf{x}}$ is an optimal solution of P.

A typical class of barrier functions are:

$$b(\mathbf{x}) = \sum_{i=1}^{m} (-g_i(\mathbf{x}))^{-q}, \text{ where } q > 0.$$

Kuhn - Tucker Multipliers in Barrier Methods

Let $b(\mathbf{x}) = \gamma(g(\mathbf{x}))$, where $\gamma(\mathbf{y}) : \mathbb{R}^m \to \mathbb{R}$, and assume that $\gamma(\mathbf{y})$ is continuously differentiable for all $\mathbf{y} < 0$. Then

$$\nabla b(\mathbf{x}) = \sum_{i=1}^{m} \frac{\partial_\gamma(g(\mathbf{x}))}{\partial \mathbf{y}_i} \nabla g_i(\mathbf{x}),$$

and if $\mathbf{x}_k$ solves $B(c_k)$, then $\nabla f(\mathbf{x}_k) + \frac{1}{c_k}\nabla b(\mathbf{x}_k) = 0$, that is,

$$\nabla f(\mathbf{x}_k) + \frac{1}{c_k} \sum_{i=1}^{m} \frac{\partial_\gamma(g(\mathbf{x}_k))}{\partial \mathbf{y}_i} \nabla g_i(\mathbf{x}_k) = 0. \tag{4.14}$$

Let us define

$$\mathbf{u}_i^k = \frac{1}{c_k}\frac{\partial_\gamma(g(\mathbf{x}_k))}{\partial \mathbf{y}_i}. \tag{4.15}$$

Then Equation 4.15 becomes:

$$\nabla f(\mathbf{x}_k) + \sum_{i=1}^{m} \mathbf{u}_i^k \nabla g_i(\mathbf{x}_k) = 0. \tag{4.16}$$

Therefore we can interpret the $\mathbf{u}_k$ as a sort of vector of Kuhn - Tucker multipliers.

Lemma 4.4. Let P satisfy the conditions of the Barrier Convergence Theorem. Suppose $\gamma(\mathbf{y})$ is continuously differentiable and let $\mathbf{u}_k$ be defined by Equation 4.15. Then if $\mathbf{x}_k \to \bar{\mathbf{x}}$, and $\bar{\mathbf{x}}$ satisfies the linear independence condition for gradient vectors of active constraints, then $\mathbf{u}_k \to \bar{\mathbf{u}}$, where $\bar{\mathbf{u}}$ is a vector of Kuhn - Tucker multipliers for the optimal solution $\bar{\mathbf{x}}$ of P.

Proof. Let $\mathbf{x}_k \to \bar{\mathbf{x}}$, and let $I = \{i|\ g_i(\bar{\mathbf{x}}) = 0\}$ and $N = \{i|\ g_i(\bar{\mathbf{x}}) < 0\}$. For all $i \in N$,

$$\mathbf{u}_i^k = \frac{1}{c_k}\frac{\partial_\gamma(g(\mathbf{x}_k))}{\partial \mathbf{y}_i} \to 0 \ as\ k \to \infty,$$

since $c_k \to \infty$ and $g_i(\mathbf{x}_k) \to g_i(\bar{\mathbf{x}}) < 0$, and $\dfrac{\partial_\gamma(g(\bar{\mathbf{x}}))}{\partial \mathbf{y}_i}$ is finite. Also $\mathbf{u}_i^k \geq 0$ for all i, and k sufficiently large.

Suppose $\mathbf{u}_k \to \bar{\mathbf{u}}$ as $k \to \infty$. Then $\bar{\mathbf{u}} \geq 0$, and $\bar{\mathbf{u}}_i = 0$ for all $i \in N$. From the continuity of all functions involved in Equation 4.16 implies that

$$\nabla f(\bar{\mathbf{x}}) + \sum_{i=1}^{m} \bar{\mathbf{u}}_i \nabla g_i(\bar{\mathbf{x}}) = 0,\ \bar{\mathbf{u}} \geq 0,\ \bar{\mathbf{u}}^T g(\bar{\mathbf{x}}) = 0.$$

Thus $\bar{\mathbf{u}}$ is a vector of Kuhn - Tucker multipliers. It remains to show that $\mathbf{u}^k \to \bar{\mathbf{u}}$ for some unique $\bar{\mathbf{u}}$. The proof that $\mathbf{u}^k \to \bar{\mathbf{u}}$ for some unique $\bar{\mathbf{u}}$ is exactly as in Lemma 4.4.

CHAPTER FIVE

SUMMARY, DISCUSSION, CONCLUSION AND RECOMMENDATION

Summary

This thesis is concerned with a study of methods in solving some standard optimization problems which we encounter in everyday life. The thesis is divided into two main parts: the first part is devoted to the study of unconstrained optimization and the second part is on constrained optimization techniques. Problems used to examine these techniques were basically of two types: There were analytic problems, which were solved by hand; the other problems were computational in nature. These required the use of mathematical softwares such as MATLAB and OCTAVE. The methods of nonlinear programming can generally be classified as either direct or indirect procedures. Examples of direct methods are the gradient algorithms, in which the minimum of a problem is sought by following the fastest rate of decrease of the objective function at a point. In indirect methods, the original problem is first transformed into an auxiliary one from which the optimum is determined. We note that the auxiliary problems in these cases may yield an exact or an approximate solution of the original problem.

The study examined unconstrained optimization techniques such as Steepest Descent method, Conjugate Gradient method, and Newton-like methods. Newton-like methods include the Newton's method, Modified Newton's method, Quasi-Newton method and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. A number of observations were made about these methods of unconstrained optimization techniques using the popular Rosenbrock's function and other functions (eg. the modified Rosenbrock's function) that have characteristics similar to those of the Rosenbrock's function. The use of the Rosenbrock's function in particular was essential in this study because its minimum point lies in a long valley. As a result it is used to test the robustness of unconstrained minimization algorithms.

Three main weaknesses were observed in the performance of the Steepest Descent method. These weaknesses were that: the algorithm can take many iterations to converge towards a local minimum; finding the optimal α per step can be time-consuming; and the rate of convergence can be very slow if the Hessian of the objective function is ill-conditioned. Ill-conditioning arises from wide differences between the largest and the smallest eigenvalues of the Hessian matrix leading to a large condition number. Thus, the method could be useful if one could reduce the condition number or use functions with small condition numbers.

The Conjugate Gradient method is considered the best among the other methods provided the function in question is quadratic in nature. With this, the iterates converge in at most two iterations.

The study has demonstrated how crucial the choice of an initial point $\mathbf{x}_0$ is to obtaining the type of critical point desired. Thus, if $\mathbf{x}_0$ is chosen close to a minimum point, the iterates lead to a minimum point. On the other hand, if $\mathbf{x}_0$ is chosen close to a maximum point we automatically obtain a maximum point.

It also came to light that using the same initial point, $\mathbf{x}_0 = [0; 0]$ say, the Newton's method may give a different result from that obtained from another method (eg. Steepest Descent method).

The implementation of the Newton's method encountered a situation where the Hessian at a point $\mathbf{x}_k$ was indefinite and therefore the vector $-\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ did not provide a descent direction. As a result the Newton's method could not continue. In order to obtain a new direction where the Hessian matrix will be positive a number of suggestions have been made that give rise to the Modified Newton's method. Two of these suggestions are that: a new iterate could be found from $\mathbf{x}_{k+1} = \mathbf{x}_k - \lambda_k \mathbf{v}_k$, where $\mathbf{v}_k$ is the eigenvector corresponding to the negative eigenvalue of $\mathbf{H}(\mathbf{x}_k)$; alternatively, a new iterate is given by $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$, where $\mathbf{s}_k$ is obtain from the equation $\mathbf{L}^T \mathbf{s}_k = \mathbf{a}$ for some vector $\mathbf{a}$ and $\mathbf{L}^T$ is obtained from a factorization of $\mathbf{H}(\mathbf{x}_k)$ as $\mathbf{LDL}^T$. Hopefully, $\mathbf{H}(\mathbf{x}_{k+1})$ should be a positive definite matrix. It has been illustrated that results using these two methods could differ.

Referring to the use of the direction $\mathbf{s}_k$ from the $\mathbf{LDL}^T$ factorization of $\mathbf{H}(\mathbf{x}_k)$ a problem was encountered. It was discovered that the use of $\mathbf{s}_k$ did not always lead to a positive definite Hessian $\mathbf{H}(\mathbf{x}_{k+1})$. It was discovered in this research that the descent direction specified by $\nabla f(\mathbf{x}_k)$ lead to an iterate $\mathbf{x}_{k+1}$ at which the Hessian $\mathbf{H}(\mathbf{x}_{k+1})$ was positive definite. The Newton's method could then continue from this point.

Other modifications of the Newton's method makes it possible to overcome some of its disadvantages. The idea of the Quasi-Newton methods is to use matrices which approximate the Hessian matrix or its inverse, instead of the Hessian matrix or its inverse in the Newton's equation.

Comparing Quasi-Newton methods with Conjugate Gradient methods, it was clear that Conjugate Gradient methods are both less efficient and less robust, and therefore would not be preferred in normal circumstances which involve few variables. However, there is one desirable quality of Conjugate Gradient methods, namely the particularly simple form which requires no matrix operations to form the search direction $\mathbf{s}_k$. Conjugate Gradient methods may be the only methods which are applicable to large problems, that is, problems with hundreds or thousands of variables.

The research also discussed constrained optimization methods. The constrained methods are the Lagrange's methods which works with equality constraints; the Kuhn-Tucker method which is used for problems with inequality constraints; the penalty methods which involve algorithms that uses penalty functions and a penalty parameter that controls relate cost of violating the constraints. These sets of constraints may be only equality constraints, inequality constraints or a combination of both equality and inequality constraints. The constrained optimization problem can be replaced by a parameterized unconstrained optimization problem whose solutions become increasingly good approximations to the solution of the original constrained problem. The implementation of the penalty method can be used to solve constrained optimization problem under varying types of constraints.

Finally, the study examined Barrier function method which determines points such that they do not approach the boundary of the feasible region. Throughout the study, examples have been used to illustrate the implementations of algorithms of optimization methods.

Discussion

Various attempts to accelerate convergence in some unconstrained optimization techniques have been made by various researchers (Fiacco and McCormick) in the past but often without rigorous basis. Various modifications have also been made to techniques that are already in operation. A study of the modifications in this research have revealed that these modifications could not lead to the optimal solution. One result of this research which is illustrated in Example 3.5, has revealed other dimensions to the convergence of the optimization techniques. According to Fiacco and McCormick (1967), if the Hessian matrix of the function is not positive definite, then a new direction must be sort which probably will help to get to the minimum.

From Example 3.5, the assumptions made by the researchers seem not to hold in the sense that choosing a new direction $\mathbf{v}$ such that

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \lambda_k \mathbf{v}_k$$

did not get us anywhere. According to the researchers, we have to express the Hessian matrix as $\mathbf{LDL}^T$ factorization. It came out that the new vector is in the same direction as the iterate $\mathbf{x}_1$. This is because the direction vectors are conjugate in nature. From the findings, the Hessian matrix was indefinite which implies that the Newton method cannot be used. This implies that, we needed to look for a direction that will make the Hessian matrix positive definite and which would lead us to the minimum point. Unfortunately, these modifications recommended by the researchers did not work.

In this research, we were able to go round this problem of indefinite Hessian matrix $\mathbf{H}(\mathbf{x}_1)$ evaluated at the point $\mathbf{x}_1$. To do this, we determined a new direction $\nabla f(\mathbf{x}_1)$ at $\mathbf{x}_1$. Using this new direction, we obtained the next iterate, $\mathbf{x}_2$ at which the Hessian matrix was positive definite. We then switched back to the Newton's method and the iterates were able to converge after twelve iterations.

A number of observations have been made in this research using a number of known functions. Notable among these functions is the function

$$f(x_1, x_2) = x_1^4 + x_1 x_2 + (1 + x_2)^2.$$

We realized that there are clear differences in the critical points of $f(\mathbf{x})$ when we used the Conjugate Gradient method and the Newton's method. The differences might be due the remedy given when the Newton's method seized to work. We noted that the Newton's method failed when $\mathbf{H}(\mathbf{x}_1)$ was not positive definite, and hence did not provide a descent direction. As a result, an alternative descent direction

$$\mathbf{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k)$$

was used via the Steepest descent method. It is noteworthy here that even though the same starting point $\mathbf{x}_0 = [0; 0]$ was used, they led to different critical points

$$[-0.204128; 0.084945] \qquad \text{and} \qquad [0.69588; -1.34794]$$

for Newton's method and Conjugate Gradient method, respectively. These results from the two methods show that the function $f(\mathbf{x})$ has multiple critical points which could not be detected using a single method. It is therefore proper to suggest that multiple Optimization methods be used to determine all critical points of a given function. This will help eliminate the influence of the starting point on the result obtained by any particular method used.

Throughout this work, we have clearly demonstrated the importance of the choice of a good initial guess in the case of the Newton's method. Obtaining the type of critical point depends so much on where you start from. This is in line with the observation of many researchers in this field.

Conclusion and Recommendation

This research studied various methods in Optimization with special interest in Unconstrained Optimization techniques. These methods were examined in the light of known functions: notably, the Rosenbrock's function and its modification and few other functions. The objective was to identify possible problems associated with the implementation of some of the procedures and algorithms that are used in Optimization techniques.

The study has discovered that a recommended modification to the Newton's method, if the Hessian $\mathbf{H}(\mathbf{x}_{k-1})$ is not positive definite, does not lead to the optimal solution. The proposed new direction $\mathbf{x}_{k+1} = \mathbf{x}_k - \lambda_k \mathbf{v}_k$ such that $\mathbf{L}^T\mathbf{v} = \mathbf{a}$, where $\mathbf{L}$ is obtained from the $\mathbf{LDL}^T$ factorization of $\mathbf{H}(\mathbf{x}_{k-1})$, does not always lead to a positive definite Hessian $\mathbf{H}(\mathbf{x}_k)$. Under this circumstance, it is suggested that a descent direction specified by $\nabla f(\mathbf{x}_k)$ be determined. By this direction we obtain $\mathbf{x}_{k+1}$ at which the Hessian $\mathbf{H}(\mathbf{x}_{k+1})$ is positive definite. The Newton's method can then continue from this point.

Different critical points of a function, $f(\mathbf{x})$ were obtained under different methods (namely, the Newton's method and the Conjugate Gradient method) using the same starting point. This indicated that $f(\mathbf{x})$ has multiple critical points which could not be detected by a single method. It is therefore recommended that multiple Optimization techniques be used to determine the critical points of a given function.

In support of the observations of many researchers, this study has illustrated clearly the importance of the choice of a good initial guess in the search for a minimum point of a function, particularly, in the case of the Newton's method. If the initial guess is not good enough, iterations of the procedure do not converge.

REFERENCES

REFERENCES

Al-Baali, M. (1985). *Descent Property and Global Convergence of the Fletcher-Reeves Method with Inexact Line Search*. IMA J. Num. Anal. **5,** pp. 121-124.

Arrow, K.J., Hurwicz, L. (1956). *Reduction of constrained maxima to saddle point problems*. Third Berkeley symposium on Mathematical Statistics and Probability. (J. Neyman, ed.), University of California Press, Berkeley, pp. 1-20.

Bazaraa, M., Sherali, H.D. and Shetty, C.M. (1993). *Nonlinear programming: Theory and Applications*. Wiley, New York.

Beale, E.M.L. (1972). *A derivation of conjugate gradients, in numerical methods for nonlinear optimization*. Academic Press, London.

Beltrami, E.J. and McGill, R. (1966). *A Class of Variational Problems in Search Theory and the Maximum Principle*. Operations Res., **14(2)**, pp. 267-278.

Broyden, C.G. (1975). *Basic Matrices*; Macmillan, London.

Butler, T. and Martin, A. V. (1962). On a method of courant for minimizing Functionals; J. Math. Phys., **41**, pp. 291-299.

Carroll, C.W. (1961). *The Created Response Surface Technique for Optimizing Nonlinear Restrained System*. Operations Res., **9**, pp. 169-184.

Curry, H. (1944). *The Method of Steepest Descent for Nonlinear Minimization Problems*. Quart. Appl. Math., **2**, pp. 258-261.

Davidon, W.C. (1973). *Variable Metric Method for Minimization*. Tech. Report ANL-5990 (revised), AEC Res. and Dev. Report.

Feder, D.P. (1957). *Automatic Lens Design Methods*. J. Opt. Soc. AM., **47**, pp. 902-912.

Fiacco, A.V. (1967). *Sequential unconstrained minimization methods for nonlinear programming*. Northwestern University, Evanston, Ill.

Fiacco, A.V. and McCormick, G.P. (1967). *The Slacked Unconstrained Minimization Technique for Convex Programming*. SIAM J. Appl. Math., **15(3)**, pp. 505-515.

Fletcher, R. (1972a). *Conjugate direction methods in numerical methods for unconstrained optimization*. Academic Press, London.

Fletcher, R. and Reeves, C.M. (1964). *Function Minimization by Conjugate Gradients*. Computer J., **7**, pp. 149-154.

Gill, P.E. and Murray, W. (1974). *Methods for large-scale linearly constrained problems, in numerical methods for constrained optimization*. Academic Press, London.

Goldfarb, D. (1980). *Curvilinear Path Steplength Algorithms for Minimization which use Directions of Negative Curvature*. Math. Prog., **18**, pp. 31 - 40.

Goldstein, A.A. and Price, J.F. (1967). *An Effective Algorithm for Minimization*. Numer. Math., **10**, pp. 184-189.

Hebden, M.D. (1973). *An Algorithm for Minimization Using Exact Second Derivatives*. AERE Harwell Report, TP515.

Hestenes, M.R. and Stiefel, E., (1952) *Methods of Conjugate Gradients for Solving Linear Systems*. J. Res. N.B.S., **49**, pp. 409-436.

Kuhn, H.W. and Tucker, A.W. (1951). *Nonlinear Programming*. J. Neyman, Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley pp. 481-493.

Levenberg, K. (1944). *A Method for the Solution of Certain Nonlinear Problems in Least Squares*. Quart. Appl. Math., **2**, pp. 164-168.

Luenberger, D.G. (1965). *Introduction to linear and nonlinear programming*. MA Addison-Wesley, Reading.

Marquardt, D.W. (1963). *An Algorithm for Least Squares Estimation of Nonlinear Parameters*. SIAM J., **11**, pp. 431-441.

McCormick, G.P. and Fiacco, A.V. (1968). *Nonlinear programming*. John Wiley, New York.

McCormick, G.P. and Pearson, J.D. (1969). *Variable metric methods and unconstrained optimization*. Academic Press, London.

McCormick, G.P. (1977). *A Modification of Armijo's Step Size Rule for Negative Curvature*. Math. Prog., **13**, pp. 111-115.

McGill, R. and Kenneth, P. (1964). *Solution of Variational Problems by Means of a Generalized Newton-Raphson Operator*. AIAA J., **2(10)**, pp. 1761-1766.

Murray, W. (1972). *Second derivative methods in numerical methods for unconstrained optimization*. Academic Press, London.

Pietrzykowski, T., (1962). *Application of the Steepest Descent method to Concave Programming*, R.A. Willoughby, Stiff Differential Systems, pp. 185-189, IFIPS Congress, Munich , North-Holland Publishing Company, Amsterdam.

Powell, M.J.D., (1977b). *Restart Procedures for the Conjugate Gradient Method*. Math. Prog., **12**, pp. 241-254.

Strong, R.E., (1965): *A Note on the Sequential Unconstrained Minimization Technique for Nonlinear Programming*. Management Sci., **12**, pp. 142-144.

Zangwill W. I. (1969). *Nonlinear programming.* A unified approach, Englewood Cliffs, NJ: Prentice Hall.

Zangwill, W.I., (1965) *Nonlinear programming by sequential unconstrained maximization*. University of California, Berkeley.

APPENDIX

GRAPHS OF FUNCTIONS USED FOR THE IMPLEMENTATION OF OPTIMIZATION TECHNIQUES

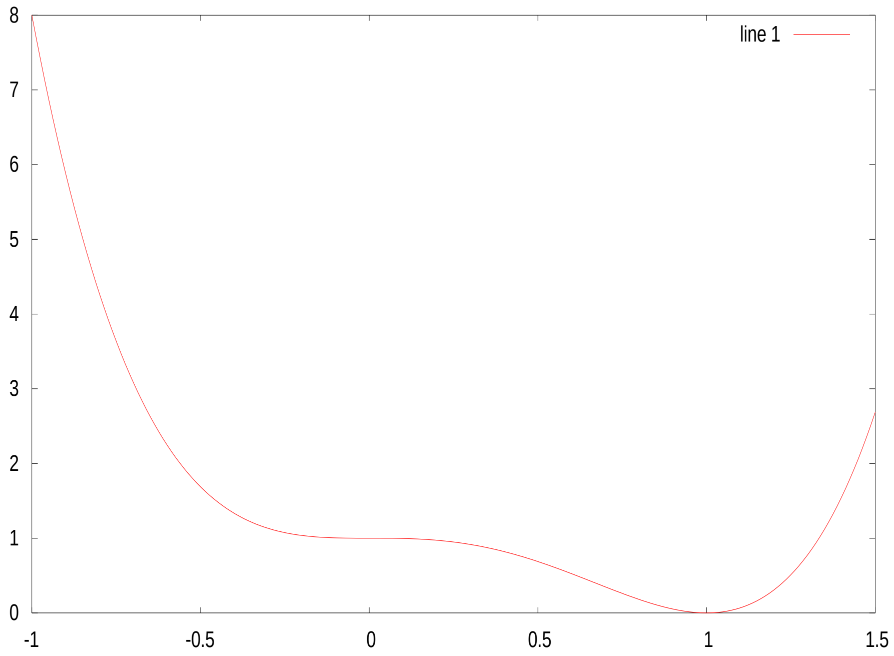

Figure 10: The graph of $y = 3x^4 - 4x^3 + 1$

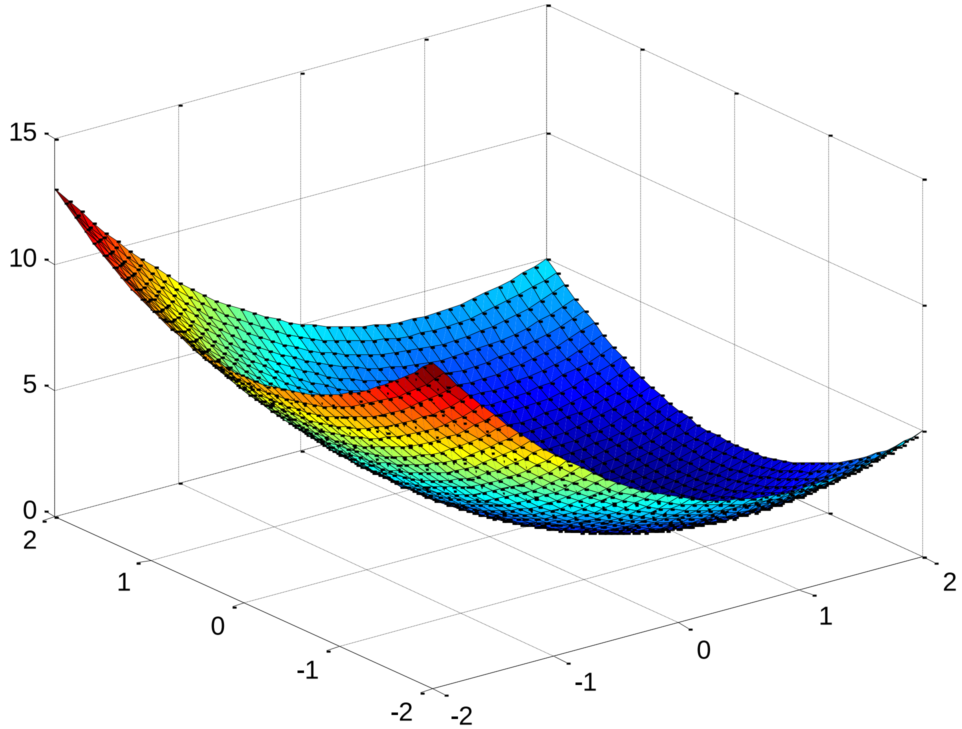

Figure 11: The graph of $z = (1 - x)^2 + y^2$ in the interval $[X, Y] = (-2 : 0.1 : 2, -2 : 0.1 : 2)$

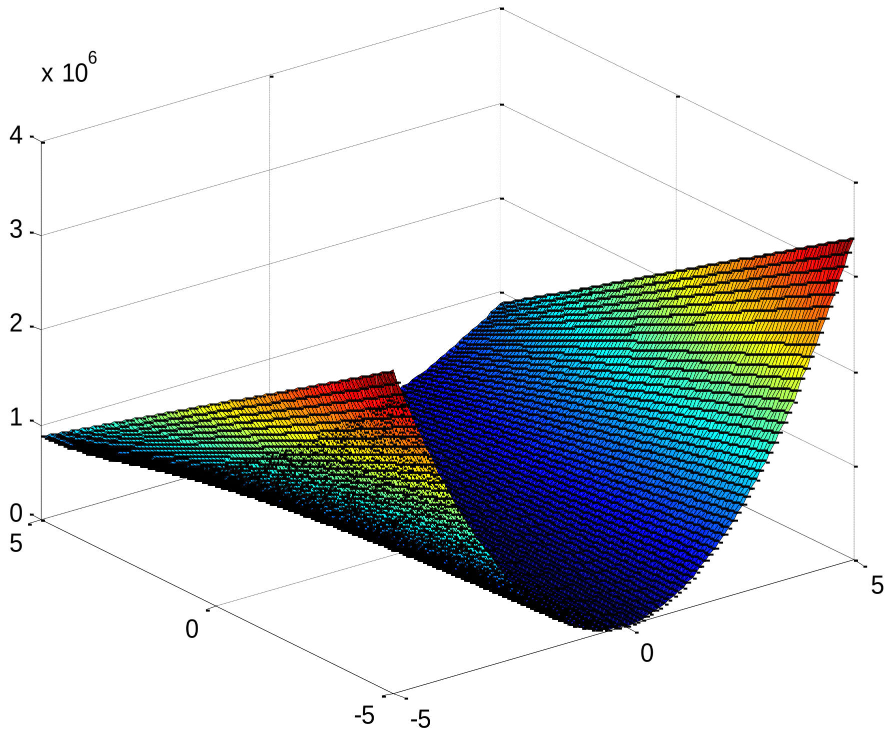

Figure 12: The graph of $z = 100\left(y - x^2\right)^2 + (1 - x)^2$ in the interval $[X, Y] = (-5:0.1:5,\ -5:0.1:5)$

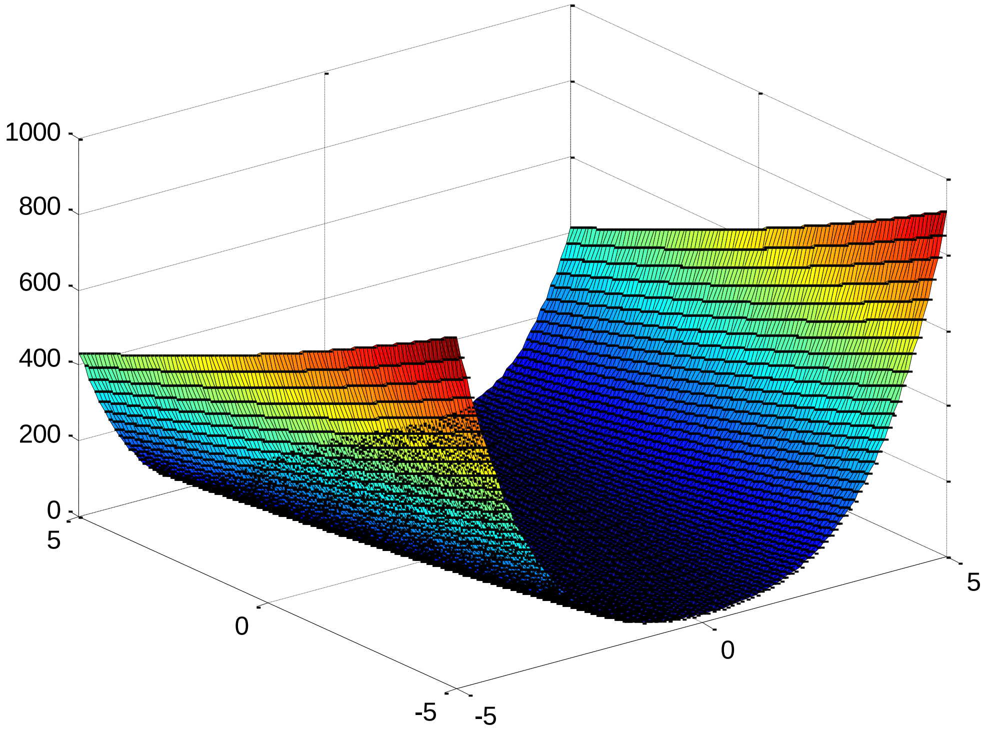

Figure 13: The graph of $z = (y - x^2)^2 + (1 - x)^2$ in the interval $[X, Y] = (-5 : 0.1 : 5, \ -5 : 0.1 : 5)$

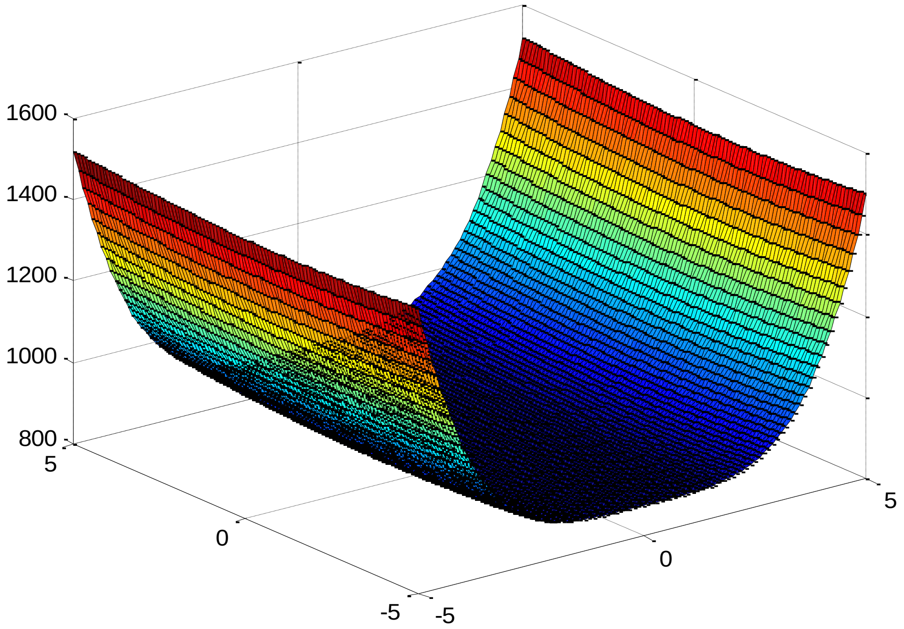

Figure 14(a): The graph of $z = x^4 + xy + (1 + y)^2$ in the interval $[X, Y] = (-5 : 0.1 : 5, \; -5 : 0.1 : 5)$

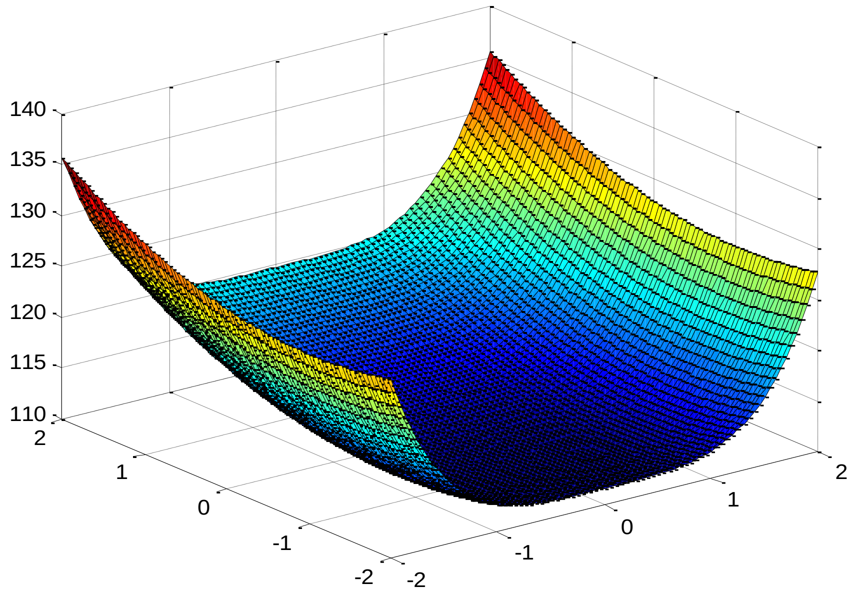

Figure 14(b): The graph of $z = x^4 + xy + (1 + y)^2$ in the interval $[X, Y] = (-2 : 0.05 : 2, \; -2 : 0.05 : 2)$