Republic of Iraq
Ministry of Higher Education & Scientific Research
University of Babylon
College of Education for Pure Sciences
Department of Mathematics

On Some Numerical Optimization Methods

A Dissertation
Submitted to the Council of College of Education for Pure Sciences,
/University of Babylon in Partial Fulfillment of the Requirements for
the Degree of Philosophy in Education / Mathematics.

By
Nofl Shaker Kathem Aashor

Supervised by
Assist. Dr. Ahmed Sabah Al-Jilawi

2022 A.D. **1443 A.H.**

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

وَمَن يَتَّقِ اللَّهَ يَجْعَل لَّهُ مَخْرَجًا ﴿٢﴾

وَيَرْزُقْهُ مِنْ حَيْثُ لَا يَحْتَسِبُ وَمَن يَتَوَكَّلْ عَلَى اللَّهِ فَهُوَ حَسْبُهُ إِنَّ اللَّهَ بَالِغُ

أَمْرِهِ قَدْ جَعَلَ اللَّهُ لِكُلِّ شَيْءٍ قَدْرًا ﴿٣﴾

صدق الله العلي العظيم

سورة الطلاق: الآية ٢ و ٣

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table 1: **Notations Used in the dissertation**

$f(x)$	Objective Function
$Minf(x)$	The Minimum of Objective Function
$Maxf(x)$	The Maximum of Objective Function
$g(x)$	The Inequality Constraint
$h(x)$	The Equality Constraint
C	The Convex Set
$\mathbb{R}$	Real Numbers
$\mathbb{R}^n$	Dimensional Space
$\nabla f(x)$	The Gradient of f
$\nabla^2 f(x)$	The Hessian of f
x^*	The Minimizer (Local, Global)
x^{k+1}	The Next Iterate
x^k	The Current Iterate
α^k	The Step Length
$NLPP$	Non-Linear Programming Problems
KKT	Karush-Kuhn-Tucker Conditions
DFP	Method of Davidon, Fletcher, and Powell
$BFGS$	Method of Broyden, Fletcher, Golfarb, and Shanno
$\mathcal{C}$	The Cone
$\mathcal{L}$	The Lagrangian Function
$\mathcal{P}(x)$	The Penalty Function
$B(x)$	The Interior Penalty Function
$E(x)$	The Exterior Penalty Function

$ALAG$	The Augmented Lagrangian Penalty Function
GP	General Problem
NP	Non-deterministic polynomial-time hardness
SDP	General Problem
CG	Conjugate Gradient Method
LP	Linear Programming Problems
$L-BFGS$	Limit memory Method of Broyden, Fletcher, Golfarb, and Shanno
d^k	descent direction

ACKNOWLEDGEMENTS

I might want to express my special thanks to many people who so lavishly take part to this study. My great respect to everyone who helped me scientifically and sentimentally, i every time praise be to Allah who gave me the power to conduct this work.

First of all, my keen supervisor merit a special mention, Professor Dr. Ahmed Sabah. Al-Jilawi. My Ph.D. has been an amazing experience, and i thank Professor Dr. Ahmed Sabah. Al-Jilawi whole- heatedly, not only for his great academic support but also for giving me the chance to learn so many wonderful life skills. He was always there when i needed him. He taught me how to be a good researcher and optimistic, even in the dark. He made every thinks voyage.

I owe special thanks to my family who supported me and helped me throughout my life and during this study. It is very hard to express how much support i have. Finally, things go to all of my friends for almost every support and chance they have provided me to finish my work successfully. I dedicate this work to all of those important people in my world.

Abstract

The goal of this dissertation is to find a better technique that converges faster. The strategy that used the type of approximate methods, we have discussed linear and non-linear optimization problems and how to convert problems from non-linear to linear problems using Lagrange equation. Also, study how to convert constrained optimization problems to unconstrained optimization problems by addition to the object function the penalty parameter, and then touched on other more common methods, which is the method of the augmented Lagrange, and compared the results of the two methods, we got good results to reach the optimal solution by using numerical analysis. However, found a better line search technique and different combinations of optimization and line search algorithm. Each iteration of a line search methods computes a search direction d^k and then decides how far to move along that direction. The iteration is given by

$$x^{k+1} = x^k + \alpha^k d^k$$

where $\alpha^k \geq 0$ and $\alpha^k \in R$ is called the step length. The prosperity of a line search methods depends on efficient choices of both the direction d^k and right the step length α^k. However, NP- problem especially(Portfolio Optimization Problem), using (GUROBI) via the provided Python interface and using regularization we found the solution is same to the original solution and if we using penalty method we will get approximates solutions using closed to optimal solution. However, choose a different problem from the **Big Mac library** and using two kinds of line searches are (HagerZhang Line search and BackTracking), to the same method Augmented Lagrangian a note the convergence to minimize take different behaviour also the same process in Penalty methods. If the change line search for the method will be got different results. After that, changed the line search methods and studied the approach behaviour to the optimal solution. Demonstrated that

for each problem a line search methods must be chosen and that not all line search methods give the same results to solve the constrained problems, so it was clarified that line search methods have a significant impact on obtaining the desired results, in addition to all of the above. Proposed a mathematical formula to reduce the spread of the disease(Coronavirus) by best on a multiple objective functions, and also proposed a mathematical formula for transporting the vaccine from the United States source to the world. All results in this study were carried out using Python language.

Devotion

For you, father, mother, wife, my children and brothers

CHAPTER 1

INTRODUCTION

Each new time Optimization teaching is being approached to hand issues that are a lot bigger and complex than before, as a result of the wide (and developing) optimization of improvement in science [55], improvement is a significant device in choice science and the investigation of actual frameworks. The way toward recognizing target factors and requirements for a given issue is known as modelling. Optimization is an essential numerical strategy pointed toward finding the estimation of factors that offer the base benefit for a numerical capacity [75], advancement calculations are an essential and productive strategy in numerical programming, to show up an answer, for the most part with the assistance of a PC. Optimization process is to finding the best solution from the set of solution to certain problem, liner programming, it is system with liner constraint and liner objective function while if at least one constraint or objective functions are non-liner is it said to be non-liner programming [46], and say convex programming if constraint and objective functions are convex. In mathematics find "min" or "max" value of objective function with i variable where $i = 1, ..., n$. Mathematical optimization is a part of the applied mathematics that is valuable in a wide range of field.

Cryptography: Aggregate improvement is a topic that consists of research A perfect object is a finite set of objects. In many like this Problems, exhaustive search is not

possible. It's working on the area of these optimization problems, where a group feasible solutions are separate or can be reduced to separate, this study is to give a survey of optimal cryptographic keys that can be developed with the help of optimization algorithms, and to address their merits to the real-world scenarios.

1.0.1 Related Work

The earlier works that are itemized in the structure of this dissertation are isolated into five fundamental classifications:

1. **Non-linear Binary Quadratic Problem**
2. **Optimization Algorithm with Line Search**
3. **Mathematical Optimization of Semi-definite programming SDP**
4. **A Line Search Technique for Constrained and Unconstrained Optimization Problem**
5. **General Approximate Augmented Lagrangian and Penalty Methods**

1.0.2 Non-linear Binary Quadratic Problem

Non-linear programming in optimization problems is, system with either object function or constraint non-linear(see equation (2.4)). Binary optimization is a central problem in mathematical optimization and its applications are abundant. The modal for solving the system, if both objective function and constraint are convex then the system convex, several ways to solve, if object is quadratic and the constraint liner quadratic programming techniques are used, several methods are available for solving non-convex problems one of the most famous and most popular of these methods is the Karush–Kuhn–Tucker (KKT) conditions, who first published the conditions in 1951 [44]. Later scholars discovered that the necessary conditions for this problem had been stated by William Karush in his master's dissertation in 1939 [44].

$$\begin{cases} \min_x f(x) \\ \text{subject to} \quad x \in \{-1, 1\}^n \\ \qquad\qquad x \in \omega \subset \mathbb{R}^n \end{cases} \tag{1.1}$$

where the objective function is real value convex function (but not necessarily smooth) on some convex set ω, and the non-convexity of is only caused by the binary constraints.

1.0.3 Optimization Algorithm with Line Search

An algorithm is a line search method if it aims to the minimum of a defined non-linear function by selecting a decent direction vector that is, when it computed iteratively with a right step length. How to find x^{k+1}? such that $f(x^{k+1}) < f(x^k)$, Each iteration of a line search method computes a search direction d^k and then decides how far to move along that direction. the iteration is given by

$$x^{k+1} = x^k + \alpha^k d^k$$

. These conditions, developed in 1969 by Philip Wolfe [29], are an inexact line search stipulation that requires to decreased the objective function by significant amount.

1.0.4 Mathematical Optimization of Semi-definite programming (SDP)

Semi-definite programming is a specific type of optimization where the matrix variables are constrained to be positive semi-definite. This type of programming has been one of the most important developments in mathematical programming [12, 18]. SDP provides convex relaxations for a number of NP hard combinatorial optimization problems such as the quadratic assignment problem and graph partitioning problems. Combinatorial Optimization by Cook, Cunningham, Pulleyblank, and Schrijver [2] gives a nice introduction computational complexity and what it means for a combinatorial

optimization problem to be defined as NPhard. SDP has many applications in many fields such as engineering, combinatorial optimization, statistics, and control theory.

1.0.5 General Approximate Augment Lagrangian and Penalty Methods

Approximation methods are very active area in optimization. Consider the minimizing convex function $C : \mathbb{R}^n \to \mathbb{R}$ on a convex set X. The goal of approximation methods is to replace C and X with approximation C^k and X^k respectively. The approximation method works only if the approximation is easier than the original problem. For each iteration k tried to find:

$$x^{k+1} = arg \min_{x \in X^k} \quad C^k(x), \tag{1.2}$$

then at the next iteration, C^{k+1} and X^{k+1} are generated by the approximation which depends on the new point x^{k+1}.

Many great approximation methods are based on this idea, such as interior point methods, the augmented Lagrangian method, and the penalty method [14]. In this dissertation, focused on the penalty method. Here present a concise history of multiplier strategies [7, 37, 62].

The fundamental papers in the zone of multiplier strategies start with Joh (1943). Kuhn and Tucker (1951) are noticeable researchers who have done broad exploration in the field of multiplier strategies. Their outcomes on essential conditions and adequate conditions are significant in this field. Bolt and Hurwicz (1956) presented the Lagrangian work [30, 48, 61]. Lord (1966) built up the expanded Lagrangian calculations. Hestenes and Power [29, 57] indicated that the calculation is locally merged if the second request adequate conditions are fulfilled. Miele et al. (1971) [54] and Rockafellar (1970) [43]presented an expanded Lagrangian strategy for disparity compelled arched programming. For the simple function optimization with equality and inequality constraints, a common method

is the penalty method. For the optimization problem

$$\begin{cases} \text{minimize} & f(x), \quad x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n \\ \text{subject to} & \phi_i(x) = 0, (i = 1, \ldots, M), \quad \psi_j(x) \leq 0, \quad (j = 1, \ldots, N) \end{cases} \tag{1.3}$$

1.0.6 A Line Search Technique for Constrained and Unconstrained Optimization Problem

There are many approaches which help us to solve unconstrained non-linear optimization problems some of them are: Line search methods, Lagrange method, KKT conditions, Penalty methods etc, will discuss some derivative based line search methods for function $f : \mathbb{R}^n \to \mathbb{R}$ which are introduced below in a general sense.

A general approach to find an optimizer is to apply line search methods. They operate iteratively, which means that they start at an initial guess x_0 for the optimizer and at each iteration compute a step, which should lead to a better solution. The algorithm terminates if an optimizer x^* satisfies certain optimality conditions [26]. Each iteration of a line search method computes a search direction d_k and then decides how far to move along that direction. The success of a line search method depends on effective choices of both the direction d_k and the step length α^k [51].

Line search methods can be applied to unconstrained and constrained optimization problems but there are different strategies that realize the line search approach. If the objective function is convex, an appropriate line search method will find the global solution. If it is not convex, it will probably just find the local minimum next to the initial guess [26].

1.0.7 The Research Objectives

The above aim can be achieved by defining a number of research objectives as follows:

1. To carry out an investigation into the existing mathematical models and to establish the modelling strategy as well as to perform numerical tests

2. To carry out a comprehensive review of the literature concerning the effect of obtaining the best solution of binary quadratic equations quickly and accurately. This will include considering a range of line search algorithms and solution techniques.

3. To identify the range of scenarios and solution techniques that are involved in the mathematical model

4. To develop an initial model based on linear programming and test the validity of equality and inequality of the augmented Lagrangian method. The numerical data will then be obtained and analysed with the aid of a relevant software tool. Initial models were studied and developed into more comprehensive ones. Accordingly, further test was only performed on linear programming but also on semi-definite programming, which was the main goal and results were analysed.

5. The developed mathematical model under a range of different line search algorithms and to identify the most appropriate one.

6. To obtain data from previous benchmarking methods and to validate the theoretical results generated from the models.

CHAPTER 2

BACKGROUND MATHEMATICAL(BASIC CONCEPT)

In this chapter, will discuss the mathematical concepts that be needed later on this dissertation.

2.0.1 Definitions and Properties

In this part, tool up definitions, notation, and necessary results of "Euclidean Space". Provide some of the mathematical background used to solve the new technique of line search. Also, present some basic concepts and facts mathematical analysis, for more on this matter, see, [7, 8, 14, 31, 35, 38, 41, 73]

Definition 2.0.1. [2] Let $f : \mathbb{R}^n \to \mathbb{R}$ and $x \in \mathbb{R}^n$. Then the partial derivative of f at x with respect to x_i is defined as:

$$\frac{\partial f(x)}{\partial x_i} = lim_{t \to 0} \frac{f(x + te_i) - f(x)}{t} \tag{2.1}$$

where e_i is i^{th} unit vector. The gradient of f at x is defined as the column vector

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

The Hessian matrix is defined as the $n \times n$ symmetric matrix:

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

The directional derivative of the function f at x in the direction d is given by:

$$f'(x,d) = lim_{t \to 0^+} \frac{f(x+td) - f(x)}{t} \tag{2.2}$$

say that the function f is differentiable at x if and only if the gradient $\nabla f(x)$ exists and satisfies $\langle \nabla f(x), d \rangle = f'(x,d), \forall d \in \mathbb{R}^n$. Moreover, say that the function f is differentiable over a subset S of $\mathbb{R}^n$ if it is differentiable at every $x \in S$. Otherwise, it's non-differentiable, and f is continuously differentiable over S, if

$$lim_{d \to 0} \frac{f(x+d) - f(x) - \langle \nabla f(x), d \rangle}{\| d \|} = 0, \forall x \in S \tag{2.3}$$

where $\| . \|$ is an arbitrary vector norm.

Example 2.0.1. Solve the following equation:

1. If $f(x,y) = x^2 + xy + y^3 - 1$ then the partial derivative of f with respect to x is $f_x = \partial f/\partial x = 2x + y$ and the partial derivative of f with respect to y is $f_y = \partial f/\partial y = x + 3y^2$
2. Let $f(x,y) = 4 - x^2 - y^2$ then the **gradient** of f is $\nabla f(x,y) = (-2x, -2y)$

Code 1 :**gradient**

```
import␣numdifftools␣as␣nd
def␣rosen1(x):
return␣4␣-␣(x[0])␣**␣2␣-␣(x[1]␣)␣**␣2
grad3␣=␣nd.Gradient(rosen)([3,␣2])
print("Gradient␣of␣f(x,y)=4-x^{2}-y^{2}␣at␣(3,␣2)␣is␣",␣grad3)
output
---Sol---
Gradient␣of␣f(x,y)=4-x^{2}-y^{2}␣at␣(3,␣2)␣is␣␣[-6.␣-4.]
```

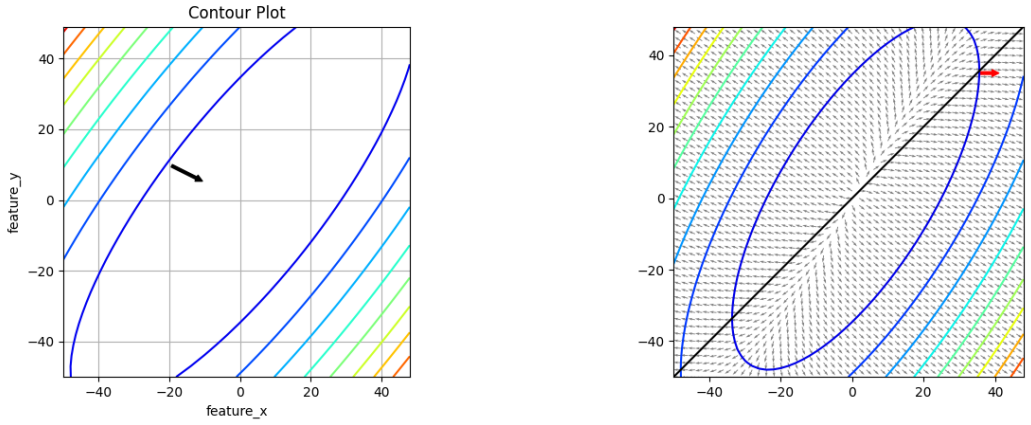


Figure 2.1: The figure shows the contour of the function and its gradient

3. Let $f(x,y) = 5x^2 + xy^3 - y^2$ then the **Hessian** matrix is $f'_x = 10x + y^3$; $f'_y = 3xy^2 - 2y$

$f''_{xx} = 10$, $f''_{xy} = 3y^2 = f''_{yx}$; $f''_{yy} = 6xy - 2$ then $\begin{bmatrix} f''_{xx} & f''_{xy} \\ f''_{yx} & f''_{yy} \end{bmatrix} = \begin{bmatrix} 10 & 3y^2 \\ 3y^2 & 6xy - 2 \end{bmatrix}$

4. If $f(x, y, z) = (xy + 2yz, 2xy^2z)$ then

$$f_1(x, y, z) = xy + 2yz$$
$$f_2(x, y, z) = 2xy^2z$$
$$\nabla f_1(x, y, z) = (y, x + 2z, 2y)$$
$$\nabla f_2(x, y, z) = (2y^2z, 4xyz, 2xy^2)$$

the Jacobian of f at x is defined as

$$\mathcal{J} = \begin{bmatrix} \\ \nabla f_1(x, y, z) \\ \\ \\ \nabla f_2(x, y, z) \\ \\ \end{bmatrix} = \begin{bmatrix} \\ y & x + 2z & 2y \\ \\ \\ 2y^2z & 4xyz & 2xy^2 \\ \\ \end{bmatrix}$$

Code 2 :Jacobian matrix

```
\begin{verbatim*}
import␣symengine
vars␣=␣symengine.symbols('x␣y␣z')␣#␣Define␣x␣and␣y␣variables
f␣=␣symengine.sympify(['x*y+2*y*z'␣,␣'2*x*y**2*z'])␣#␣Define␣function
J␣=␣symengine.zeros(len(f),len(vars))␣#␣Initialise␣Jacobian␣matrix
for␣i,␣fi␣in␣enumerate(f):
for␣j,␣s␣in␣enumerate(vars):
J[i,j]␣=␣symengine.diff(fi,␣s)
print␣(J)
output
[y,␣x␣+␣2*z,␣2*y]
[2*y**2*z,␣4*x*y*z,␣2*x*y**2]
```

5. Find rank of matrix $\mathcal{A} = \begin{bmatrix} 1 & 2 & -1 & 2 & 3 \\ 3 & 7 & 2 & 3 & 7 \\ 5 & 11 & 0 & 7 & 13 \end{bmatrix}$

Multiple the first row by (-3) and adding with second row get the second row; and multiple the first row by (-5) and adding with third row get the third row then after this process convert $\mathcal{A}$ to r the number of row non-zero equal to 2 then the rank of $\mathcal{A} = 2$. Resolve the above example by using the **Python code**

Code 3 :**Rank of matrix**

```
from␣sympy␣import␣*
M␣=␣Matrix([[1,␣2,␣-1,␣2,3],␣[3,␣7,␣2,␣3,␣7],␣[5,␣11,␣0,␣7,␣13]])
print("Matrix␣:␣{}␣".format(M))
M_rref␣=␣M.rref()
print("The␣Row␣echelon␣form␣of␣matrix␣M␣and␣the␣pivot␣columns␣:␣{}".format(M_rref))
output
Matrix([[1,␣2,␣-1,␣2,␣3],␣[3,␣7,␣2,␣3,␣7],␣[5,␣11,␣0,␣7,␣13]])
The␣Row␣echelon␣form␣of␣matrix␣M␣and␣the␣pivot␣columns␣:␣(Matrix([
[1,␣0,␣-11,␣␣8,␣␣7],
[0,␣1,␣␣␣5,␣-3,␣-2],
[0,␣0,␣␣␣0,␣␣0,␣␣0]])
```

Definition 2.0.2. [40] **Vector Norm**:

Introduce the Euclidean space, which means a finite dimensional vector space with an inner product $\langle \cdot, \cdot \rangle$ and the Euclidean norm defined by $\|x\| = \sqrt{\langle x, x \rangle}$ for all vectors x in the vector space.

A function $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is an **inner product** if

1. $\langle x, x \rangle > 0$, $\langle x, x \rangle = 0 \Leftrightarrow x = 0$ (positivity).

2. $\langle x, y\rangle = \langle y, x\rangle$ (symmetric).

3. $\langle x + y, z\rangle = \langle x, z\rangle + \langle y, z\rangle$ (additivity).

4. $\langle \alpha x, y\rangle = \alpha \langle x, y\rangle$ for all $\alpha \in \mathbb{R}$ (homogeneity).

Theorem 2.0.1. ***Cauchy-Schwartz Inequality*** *[40] for any vector* $|u.v| \leq \|u\| \|v\|$

2.1 Optimization Problems

An Optimization or Mathematical Programming problem is defined by four basic components [1]:

1. A set of independent variables that represent the different kinds of decisions to be made. They are usually called decision variables.

2. A set of constant quantities, representing the data of the problem.

3. A set of mathematical expressions of the constraints the decision variables have to satisfy. If there are no constraints the optimization problem is called unconstrained. Otherwise it is called constrained.

4. A mathematical expression of an appropriate measure of performance, termed objective function.

The main aim of an optimization problem is to determine the values of the decision variables that optimize (i.e., minimize or maximize) the objective function and satisfy simultaneously all the existing constraints. An optimization problem may be defined in mathematical terms as:

$$(GP)\begin{cases} \underset{x}{\text{Minimize}} & f(x) \\ \text{subject to} & h_j(x) = 0, \quad j = 1, 2, ..., L \\ & g_i(x) \leq 0, \quad i = 1, 2, ..., M \\ x \in \mathbb{R}^n. \end{cases} \tag{2.4}$$

Where $x = (x_1, x_2, ..., x_n)^T$ is the vector of the decision variables and $\mathbb{R}^n$ is the domain of x. The relations $h_j(x) = 0, j = 1, 2, ..., L$ and $g_i(x) \leq 0, i = 1, 2, ..., M$ are the constraints and $f(x)$ is the objective function of the optimization problem.

Definition 2.1.1. The Objective Function

The objective function represents the model primary goal, which is to be either minimized or maximized.

A point x is feasible if it is in $\mathbb{R}^n$ and satisfies the constraints of problem (see equation (2.4)). The set S of all feasible points defines the feasible region of the optimization problem, i.e.,

$$S := \{x \in \mathbb{R}^n | h_j(x) = 0 \quad for \quad j = 1, 2, ..., L, \quad g_i(x) \leq 0 \quad for \quad i = 1, 2, ..., M\}.$$

.

Definition 2.1.2. Argmin [57]

let $\mathbf{S} \subseteq \mathbf{R^n}$ be non-empty set if $f : S \to R$, the argmin of the minimum is the set of element in S that achieve the global minimum in S.

$$\text{argmin} f(x) = \{x \in S | f(y) \geq f(x), \forall \quad y \in S\}$$

for example if $f : R \to R$ is given by $f(x) = (x + 1)^2 + 3$ then **argmin** $\mathtt{f(x)} = \{-1\}$ where $f(x) = (-1 + 1)^2 + 3 = 3$ is the minimum objective function.

Definition 2.1.3. Convex [11, 14, 24]

A set K is convex (see figure 2.2): If the line segment between any two points in set K, lies in K. This means if for any $x, y \in K$ with $x \neq y$ and any λ with $0 \leq \lambda \leq 1, \lambda x_1 + (1 - \lambda x_2) \in$ K.

- A set is convex: if every point in the set can be seen by other point along an unobstructed straight path between them.

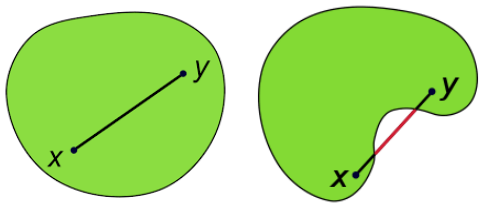


Figure 2.2: Convex set and nonconvex

- Every affine set is also convex.
- Convex combination point of the form $\lambda_1 x_1 + ... + \lambda_k x_k$, where $\lambda_1 + ... + \lambda_k = 1$, $\lambda_i \geq 0 \; i = 1, ..., k$

  Set is convex if and only if it contains every convex combination of it's points .
- Convex hull of the set S denote by

  $$\text{Conv}C = \{\lambda_1 x_1 + ... + \lambda_k x_k, \lambda_1 + ... + \lambda_k, x_i \in C, \lambda_i \geq 0, i = 1, ..., k, \lambda_1 + ... + \lambda_k = 1\}$$

  The **Conv C** is always convex. It is the smallest convex set that contains C, if H any convex set such that $H \subseteq C$ then Canv$C \subseteq H$
- Affine function $f(x) = a^\intercal x + b$ (for any $a \in R^n, b \in R$). They are also concave . $\forall \lambda \in [0, 1]$

  $$\begin{aligned} f(\lambda x + (1-\lambda)y) &= a^\intercal(\lambda x + (1-\lambda)y) + b \\ &= \lambda a^\intercal x + (1-\lambda)a^\intercal y + \lambda b + (1-\lambda)b \\ &= \lambda f(x) + (1-\lambda) f(y). \end{aligned}$$

- The quadratic function is convex if and only if $\mathbb{Q} \geq 0$ (positive semi-definite)

$$f(x) = \min \frac{1}{2} x^\top Q x + c^\top x$$

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &= f(y + \lambda(x-y)) \\ &= \frac{1}{2}(y + \lambda(x-y))^\intercal \mathbb{Q}(y + \lambda(x-y)) + c^\intercal(y + \lambda(x-y)) \\ &= \frac{1}{2} y^\intercal \mathbb{Q} y + \lambda(x-y)^\intercal \mathbb{Q} y + \frac{1}{2}\lambda^2 (x-y)^\intercal + \mathbb{Q}(x-y) + \lambda c^\intercal x + (1-\lambda) c^\intercal y \\ &\leq \frac{1}{2} y^\intercal \mathbb{Q} y + \lambda(x-y)^\intercal \mathbb{Q} y + \frac{1}{2}\lambda (x-y)^\intercal \mathbb{Q}(x-y) + \lambda c^\intercal x + (1-\lambda) c^\intercal y \\ &= \frac{1}{2}\lambda x^\intercal \mathbb{Q} x + \frac{1}{2}(1-\lambda) y^\intercal \mathbb{Q} y + \lambda c^\intercal x + (1-\lambda) c^\intercal y \\ &= \lambda f(x) + (1-\lambda) f(y) \end{aligned}$$

Definition 2.1.4. Line:

Suppose $x_1 \neq x_2$ are two points in $R^n \quad y = \lambda x_1 + (1-\lambda)x_2$ where $\lambda \in R$, form the line segment passing through x_1 and x_2, the parameter $\lambda = 1$ corresponds to $y = x_1$, and the parametric value $\lambda = 0$ corresponds to $y = x_2$ [11].

Definition 2.1.5. Line segment:

Suppose $x_1 \neq x_2$ are two points in $R^n \quad y = \lambda x_1 + (1-\lambda)x_2$ where $\lambda \in R, \quad 0 \leq \lambda \leq 1$, is the line segment between x_1 and x_2 [11].

Definition 2.1.6. [14]

Definition 2.1.7. Affine set:

A set $S \subseteq R^n$ is affine if the line through two distinct point in S is belongs to S, this means $x_1 \neq x_2$ are two points in S [11].

$$y = \lambda x_1 + (1-\lambda)x_2 \quad \text{where} \quad \lambda \in R \quad \text{have} \quad y \in S$$

Can say $x_1, x_2, ..., x_k$ are distinct point where $\lambda_1 + \lambda_2 + ... + \lambda_k = 1$ as an affine combination of the point is $\lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_k x_k$. Indeed every affine set can be expressed as the solution set of system of liner equations, and the converse is also true.

The solution set of system of liner equation S=$\{x : Ax = b\}$ where $A \in R^{m\times n}$ and $b \in R^m$ is an affine such that, $x_1, x_2 \in S$ such that$Ax_1 = b$, $Ax_2 = b$, for$\lambda \in R$;

$$\begin{aligned} A(\lambda x_1 + (1 - \lambda x_2) &= \lambda A x_1 + (1 - \lambda) A x_1 \\ &= \lambda b + (1 - \lambda) b \\ &= b \end{aligned}$$

This shown that the affine combination $\lambda x_1 + (1 - \lambda x_2) \in S$

Definition 2.1.8. [77] **Affine hull**: The set of all affine combination of points in some set $S \subseteq R^n$, and denoted **aff C** $= \{\lambda_1 x_1 + ... + \lambda_k x_k | x_1, ..., x_k \in C, \lambda_1 + ... + \lambda_k = 1\}$, affine set hull is the smallest affine set that contains C, this means if S is any set with $C \subseteq S$, then **aff C** $\subseteq S$. **Affine dimension** of a set S as the dimension of it is **affine hull**

Example 2.1.1. $S = \left\{x \in R^2 | x_1^2 + x_2^2 = 1\right\}$, unit circle in R^2 (of one dimension) but it is **aff S**, of dimension two.

relative interior

define **relint C** $= \{x \in C | B(r, x) \cap$ **aff C** $\subseteq C$for some$(r \geq 0)\}$

Example 2.1.2. Is $f(x) = \ln(x)$ convex or concave ?

Solution:

Code 6 :f(x)=In(x)

```
import numpy as np
import matplotlib.pyplot as plt
x= np.linspace(1,8,50)
# objective function
f=np.log(x)
plt.plot(x, f, color=(1, 0, 1))
```

```
plt.grid()
plt.xlabel('$x␣$␣␣␣␣nofl')
plt.ylabel('$\ln␣x␣$␣␣␣␣nofl')
#␣convexity/␣concavity
a=2
b=7
lamda␣=␣0.4
c=lamda*a␣+(1-lamda)*b
f_a=np.log(a)
f_b=np.log(b)
f_c=np.log(c)
f_c_hat=lamda*f_a␣+␣(1-lamda)*f_b
plt.plot([a,a],[0,f_a],␣color=(1,0,0)␣,␣marker='o'␣,␣label="$f(a)$")
plt.plot([b,b],[0,f_b],color=(0,0,1),␣marker='o',label=
"$f(\lambda␣a␣+␣(1-\lambda)b)$")
plt.plot([c,c],[0,f_c_hat],color=(1/2,2/3,3/4)␣,␣marker='o',label=
"$\lambda␣f(a)␣+␣(1-\lambda)␣f(b)$")
plt.plot([a,b],[f_a,f_b],␣color=(0,1,1))
plt.legend(loc=2)
plt.show()
```

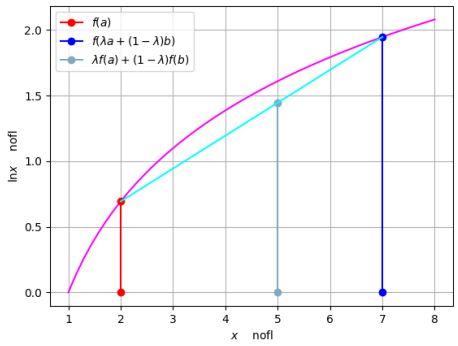


Figure 2.3: The line segment joining any two points on the curve of the function lies below the curve. Then function $f(x) = \ln(x)$ is convex

Example 2.1.3. Is $f(x) = x^2$ convex or concave ?

Solution: by using Python(version 6.1.7601), we are got $\ln(x)$ convex.

Code 7 :$f(x) = x^2$

```
import␣numpy␣as␣np

import␣scipy

import␣matplotlib.pyplot␣as␣plt

x=␣np.linspace(1,8,50)

def␣sq(x):

return␣x**2

x␣=np.linspace(-5,␣5,␣50)

vec_sq=scipy.vectorize(sq)

#objective␣function

f=vec_sq(x)

plt.plot(x,␣f,␣color=(1,␣0,␣1))

plt.grid()

plt.xlabel('$x␣$␣␣␣␣nofl')

plt.ylabel('$x^2␣$␣␣␣␣nofl')

#␣convexity/␣concavity

a=-3
```

```
b=4
lamda␣=␣0.3
c=lamda*a␣+(1-lamda)*b
f_a=sq(a)
f_b=sq(b)
f_c=sq(c)
f_c_hat=lamda*f_a␣+␣(1-lamda)*f_b
plt.plot([a,a],[0,f_a],␣color=(1,0,0)␣,␣marker='o'␣,␣label="$f(a)$")
plt.plot([b,b],[0,f_b],color=(0,1,0),␣marker='o',label="$f(b)$")
plt.plot([c,c],[0,f_c_hat],color=(0,0,1)␣,␣marker='o',label=
"$\lambda␣a␣+␣(1-\lambda)␣b$")
plt.plot([c,c],[0,f_c_hat],␣color=(1/2,2/3,3/4),␣marker='o',label=
"$\lambda␣f(a)\\␣+␣(1-\lambda)f(b)$")
plt.plot([a,b],[f_a,f_b],color=(0,1,1))
plt.legend(loc=2)
plt.show()
```

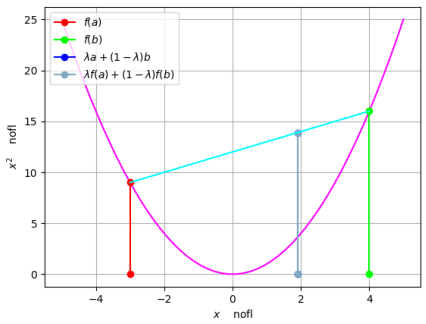


Figure 2.4: The line segment joining any two points on the curve of the function lies upper the curve. Then function is convex

Definition 2.1.9. Epigraph [77]: Let $\mathtt{S} \subseteq \mathtt{R^n}$ be anon-empty convex set the epigraph of a function $f : S \to R$ denotes by $epi(f) \subseteq R^{n+1}$

$$epi(f) = \{(x, \lambda) | f(x) \leq \lambda, x \in S, \lambda \in R\}$$

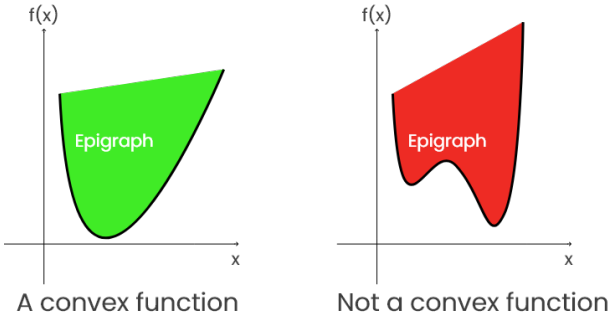


Figure 2.5: The figure illustrates the concept epigraph

Definition 2.1.10. [77] **Closed convex function**:

A convex function f is called closed if it is *epigraph*. So it's closed set. **Note** that the function which is convex and continues on an closed domain is closed function. For instance, the norms are closed convex functions.

Further all differentiable convex function are closed with $Domf = R^n$.

Theorem 2.1.1. *[71]* $\varnothing \neq S \subseteq R^n$, $f(.) : S \to \mathbb{A}$ *Then* f *is convex if and only if* $epif(.)$ *is a convex set.*

Definition 2.1.11. Euclidean balls: Boll in R^n has the form

$$B(x, r) = \{x| \, \|x - x_c\|_2 \leq r\} = x|(x - x_c)^\intercal (x - x_c) \leq r^2$$

since $\|y\|_2 = \sqrt{y^\intercal y}$.

$\mathrm{B}(\mathrm{x}_c, r) = \{x_c + ru| \, \|u\|_2 \leq 1\}$ contains all point within a distance r of the center x_c [40].

The euclidean balls is convex set, let $x_1, x_2 \leq B(X_c, r)$ such that

$$\|x_1 - x_c\|_2 \leq r, \|x_2 - x_c\|_2 \leq r \quad \text{and} \quad 0 \leq \lambda \leq 1$$

$$\begin{aligned} \|\lambda x_1 + (1-\lambda)x_2 - x_c\|_2 &= \|\lambda(x_1 - x_c) + (1-\lambda)(x_2 - x_c)\|_2 \\ &\leq \lambda \|x_1 - x_c\| + (1-\lambda)\|x_2 - x_c\|_2 \\ &\leq r. \end{aligned}$$

Definition 2.1.12. Cone [11] A set $\mathtt{C} \subseteq R^n$ is a cone when with every $x \in C$ the whole ray $\{\lambda x | \lambda \geq 0\}$ also belongs to the set C. This means $\lambda x \in \mathtt{C}, \forall x \in C$ and $\lambda \geq 0$, cone is may or not may be convex also cone may or may not contains the origin. **For example**

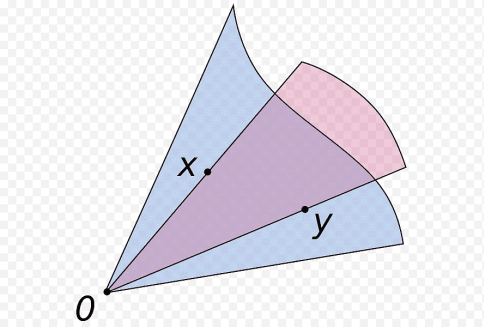


Figure 2.6: The figure shows the convex cone

$\{x \in R^n | x_1 x_2 = 0\}$ cone that contains origin and non convex while $\{x \in R^n | x_1 x_2 \neq 0\}$ cone that not contains origin and non convex, **Proper cones** A cone $K \subseteq R^n$ is called a proper cone if it satisfies

- K is convex (A cone K is convex iff $x + y \in$K) for any $x, y \in$K)
  For example : subspace non-negative or that of R^n.

$$R^n_+ := \{x = (x_1, ..., x_n) : x_1 \geq 0\}$$

for $\{i = 1, ..., n\} = \{x \in R^n :< e_i, x >\geq 0 for i = 1, ..., n\}$

- K is closed (contains it is boundary)

- K is solid, which means it has non-empty interior
- K is pointed, which means it contains no line $\Rightarrow x = 0$

Definition 2.1.13. [4]**Norm balls and norm cones**

Suppose $\|.\|$ is any norm on R^n

1. Norm boll of reduce r and the center x_c, given by $\{x|\,\|x - x_c\| \leq r\}$ **is convex**
2. The norm cone associated with the norm $\|.\|$ is the, $C = \{(x,t) : \|x\| \leq t\} \subseteq R^{n+1}$

Definition 2.1.14. Separating and Supporting hyperplanes [64] [51]

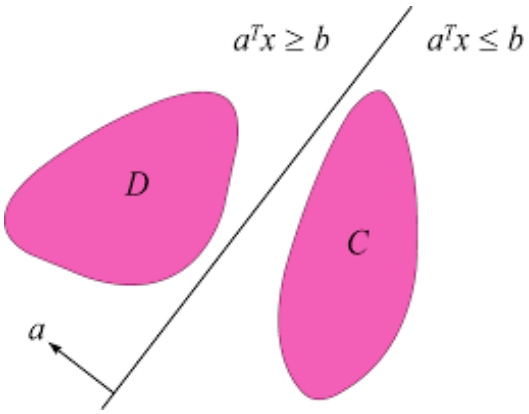


Figure 2.7: Separating and Supporting hyperplanes

Suppose C and D are non-empty disjoint convex sets this means $C \cap D = \phi$ then there exist $a \neq 0$ and b such that $a^\intercal x \leq b$ for all $x \in C$ and $a^\intercal x \geq b \forall \quad x \in D$. In other word the affine function $a^\intercal x - b$ is non-positive on C and non-negative on D the hyperplane $\{x|a^\intercal x = b\}$ is called a separation hyperplane for the set S and D or said to be separation the sets S and D.

Example 2.1.4. Separation of an affine and the convex set, suppose C is convex and $D = \{F_U + g|u \in R^m\}$, where $F \in R^{n\times m}$ suppose C and D are disjoint. There are $a \neq 0$ and b such that $a^\intercal x \leq b$ for all $x \in C$ And $a^\intercal x \geq b \forall x \in D$.

Now for $a^\intercal x \geq b \forall x \in D$ means $a^\intercal F u > b - a^\intercal g$ for all $u \in R^m$. But a liner function is bounded below on R^m only one it is zero, so conclude $a^\intercal F = 0$(and hence $b \leq a^\intercal g$),

thus, conclude that there exist $a \neq 0$ such that $F^\intercal a = 0$ and $a^\intercal x \leq a^\intercal g$ for all $x \in C$. A hyperplane divided R^n into two half space, A(closed) half space is a set of the form $\{x|a^\intercal X \leq b\}$, **half space convex, but not affine**.

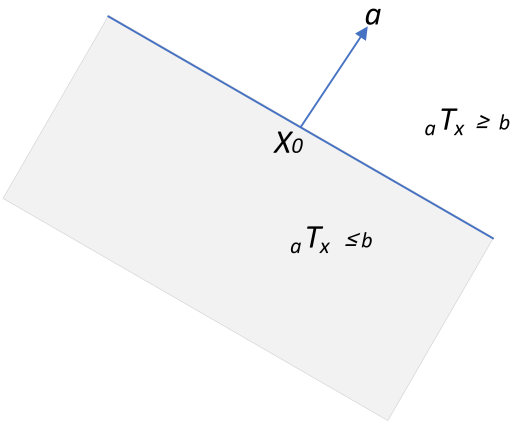


Figure 2.8: Half Space

Definition 2.1.15. global minimum and local minimum [5,14]: A point x^* is called a local minimize if $f(x^*) \leq f(x)$ for all $x \in \mathcal{B}(x^*, \delta)$ where $\delta > 0$, A point x^* is called a strict local minimum if $f(x^*) < f(x)$ for all, $x \in \mathcal{B}(x^*, \delta)$ where $\delta > 0$ with $x \neq x^*$
A point x^* is called a global minimize if $f(x^*) \leq f(x)$ for all $x \in R$, a point x^* is called a strict global minimum if $f(x^*) < f(x)$ for all $x \in R^n$, with $x \neq x^*$.

Example 2.1.5. Find the local minimum and the global minimum to the function $f(x) = x^2 + 9\sin(x)$

solution

The global minimum =[-1.28190918], and Local minimum found 3.733770752876532

Code 4 :Global minima

```
import␣numpy␣as␣np
from␣scipy␣import␣optimize
import␣matplotlib.pyplot␣as␣plt
x␣=␣np.arange(-10,␣10,␣0.1)
def␣f(x):
```

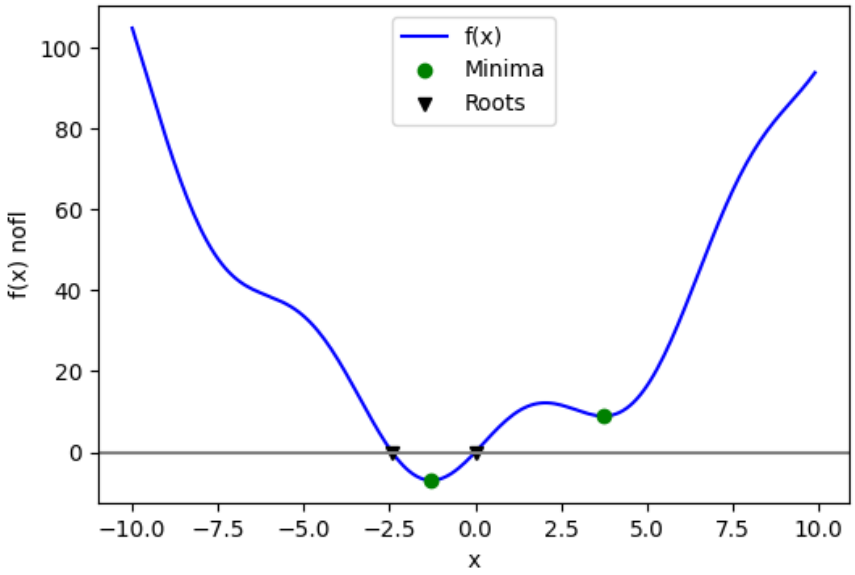


Figure 2.9: The figure shows local and global minimum

```
return␣x**2+␣9*np.sin(x)
#␣Global␣optimization
grid␣=␣(-10,␣10,␣0.1)
xmin_global␣=␣optimize.brute(f,␣(grid,␣))
print("Global␣minima␣found␣%s"␣%␣xmin_global)
#␣Constrain␣optimization
xmin_local␣=␣optimize.fminbound(f,␣0,␣10)
print("Local␣minimum␣found␣%s"␣%␣xmin_local)
root␣=␣optimize.root(f,␣1)␣␣#␣our␣initial␣guess␣is␣1
print("First␣root␣found␣%s"␣%␣root.x)
root2␣=␣optimize.root(f,␣-2.5)
print("Second␣root␣found␣%s"␣%␣root2.x)
fig␣=␣plt.figure(figsize=(6,␣4))
ax␣=␣fig.add_subplot(111)
#␣Plot␣the␣function
ax.plot(x,␣f(x),␣'b-',␣label="f(x)")
```

```
# Plot the minima
xmins = np.array([xmin_global[0], xmin_local])
ax.plot(xmins, f(xmins), 'go', label="Minima")
# Plot the roots
roots = np.array([root.x, root2.x])
ax.plot(roots, f(roots), 'kv', label="Roots")
# Decorate the figure
ax.legend(loc='best')
ax.set_xlabel('x')
ax.set_ylabel('f(x) nofl')
ax.axhline(0, color='gray')
plt.show()
Global minima found [-1.28190918]
Local minimum found 3.733770752876532
First root found [0.]
Second root found [-2.42764]
```

Example 2.1.6. **Eggholder Function** Global optimization aims to find the global minimum of a function within given bounds, in the presence of potentially many local minima. Eggholder function denotes by

$$f(x) = -(x_2 - 47)sin(\sqrt{|x_2 + \frac{x_1}{2} + 47|}) - x_1 sin(\sqrt{|x_1 - (x_2 + 47)|})$$

. The Eggholder function is a difficult function to optimize, because of the large number of local minima. The function is usually evaluated on the square $x_i \in [-512, 512]$, for all i = 1, 2. where the optimal point $x^* = (512, 404.2319)$ and the objective function value is $f(x^*) = -959.6407$ typically, global minimizers efficiently search the parameter space, while using a local minimizer under the hood. SciPy contains a number of good

global optimizers. Here, will use those on the same objective function, namely the (aptly named) eggholder function

Code 5 :Eggholder

```
import␣matplotlib.pyplot␣as␣plt
import␣numpy␣as␣np
from␣mpl_toolkits.mplot3d␣import␣Axes3D
from␣scipy␣import␣optimize
def␣eggholder(x):
return␣(-(x[1]␣+␣47)␣*␣np.sin(np.sqrt(abs(x[0]/2␣+␣(x[1]␣␣+␣47))))
-x[0]␣*␣np.sin(np.sqrt(abs(x[0]␣-␣(x[1]␣␣+␣47)))))
bounds␣=␣[(-512,␣512),␣(-512,␣512)]
x␣=␣np.arange(-512,␣513)
y␣=␣np.arange(-512,␣513)
xgrid,␣ygrid␣=␣np.meshgrid(x,␣y)
xy␣=␣np.stack([xgrid,␣ygrid])
fig␣=␣plt.figure()
ax␣=␣fig.add_subplot(111,␣projection='3d')
ax.view_init(45,␣-45)
def␣eggholder(x):
return␣(-(x[1]␣+␣47)␣*␣np.sin(np.sqrt(abs(x[0]/2␣+␣(x[1]␣␣+␣47))))
-x[0]␣*␣np.sin(np.sqrt(abs(x[0]␣-␣(x[1]␣␣+␣47)))))
bounds␣=␣[(-512,␣512),␣(-512,␣512)]
ax.plot_surface(xgrid,␣ygrid,␣eggholder(xy),␣cmap='terrain')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('eggholder(x,␣y)␣␣')
results␣=␣dict()
results['shgo']␣=␣optimize.shgo(eggholder,␣bounds)
```

```
results['shgo']
print(results)
plt.show()
```

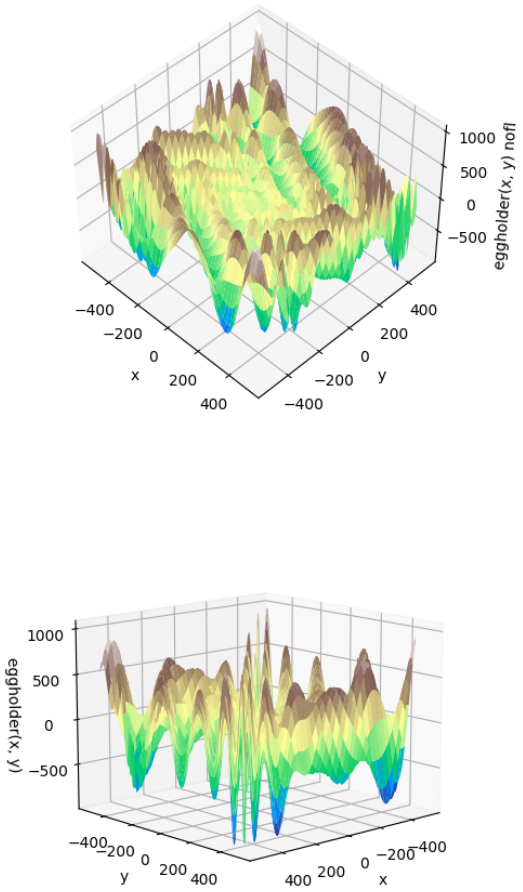


Figure 2.10: The figure shows the optimal point $x^* = (512, 404.2319)$

Definition 2.1.16. Feasible Region [14]

The feasible region or the feasible set is the set of all possible values of the problem that satisfies all the constraints of the problem. Otherwise, the region is said to be infeasible if no solution exists which satisfies all the constraints. The constraints could be equalities and inequalities. The feasible point is a substitution in the objective point.

For any feasible set x, the active set $A(x)$ of the inequality constraints denoted by E:

$$A(x) = E \cup \{j \in L | h_j(x) = 0\}.$$

2.2 Linear Programming

Each new time Optimization teaching is being approached to hand issues that are a lot bigger and complex than before, as a result of the wide (and developing) Optimization of improvement in science. Improvement is a significant device in choice science and the investigation of actual frameworks. The way toward recognizing target factors and requirements for a given issue is known as modelling [46, 55, 75].

Mathematical Formulation

$$\begin{cases} \min(\max) & \mathtt{Q}(\mathtt{x}) \\ \text{subject to} & x \in \mathtt{X} \end{cases} \tag{2.5}$$

where $\mathtt{x} \in \mathtt{R}^{\mathtt{n}}$parameters and $\mathtt{Q}$ is objective function requirement minimize or maximize. Want find x^* such that $Q(x^*) \leq Q(x)$ for all $\mathtt{x} \in \mathtt{X}$, always can say

$$\text{maximize} Q(x) \equiv -minimize Q(x)$$

the feasible set X of linear programming, and the regain boundaries illustrate the constraints.

The general optimization problem in the form

$$\begin{cases} \text{minimize} & f^0(x) \\ \text{subject to} & f^i(x) \leq 0, \quad i = \{1, \ldots, I\}, \\ & g^i(x) = 0, \quad i = \{1, \ldots, J\}, \\ & x \in \Omega. \end{cases} \tag{2.6}$$

where I and J are sets of indicator for equality and inequality constraints. Object function and equality and inequality constraints are linear. Some problems from LP

1. Finding the shortest way between the two cities.
2. Location Problem

   Finding a suitable location between two cities to build an airport, so that the distance between the airport and the two cities is the minimal.
3. Transportation Problem

   Find the best way to satisfy the requirement of demand points using the capacities of supple point.

Example 2.2.1. Graphically obtain a solution to the following

$$\begin{cases} \min_x & \mathtt{f(x)} = 6x_1 + 5x_2 \\ \text{subject to} & x_1 + 5x_2 \leq 5 \\ & 3x_1 + 2x_2 \leq 12 \\ & x_1, x_2 \geq 0 \end{cases}$$

Code 8 :Optimal of LP

```
import␣numpy␣as␣np
import␣mpmath␣as␣mp
import␣matplotlib.pyplot␣␣as␣plt
from␣scipy.spatial␣import␣ConvexHull
points=␣np.array([[0,␣0],␣[0,␣5],␣[2,␣3],␣[4,␣0]])
plt.fill(points[:,␣0],␣points[:,␣1],␣'k',␣alpha=0.3)
p=9*np.arange(1,␣5)
print(p)
x=␣np.linspace(0,␣4,␣100)
na=np.newaxis
```

```
x_line=x[:,␣na]
y_line␣=␣p[na,␣:]/5.0␣-␣6.0/5.0*x[:,␣na]
bx=plt.plot(x_line,␣y_line,␣'-')
x1␣=␣np.linspace(0,␣4,␣100)
x2␣=␣5*np.ones(100)␣-␣x1
ax=plt.plot(x1,␣x2)
x2=␣6*np.ones(100)␣-␣3.0/2.0*x1
cx=␣plt.plot(x1,␣x2)
sol␣=␣np.zeros((2,␣1))
sol[0]=2
sol[1]=3
A=sol[0]
B=sol[1]
plt.plot(A,␣B,␣'o')
for␣xy␣in␣zip(A,␣B):plt.annotate('(%s,␣%s)'␣%␣xy,␣xy=xy,␣textcoords='data')
plt.grid()
plt.xlabel('X_1␣nofl␣')
plt.ylabel('X_2␣nofl')
plt.legend([bx[2],␣ax[0],␣cx[0]],␣['$6x_1+5x_2=27$',
␣'$x_1+x_2=5$',␣'$3x_1+2x_2=12$'],␣loc='best',␣prop={'size':11})
plt.xlim(0,␣4)
plt.ylim(0,␣5)
plt.show()
```

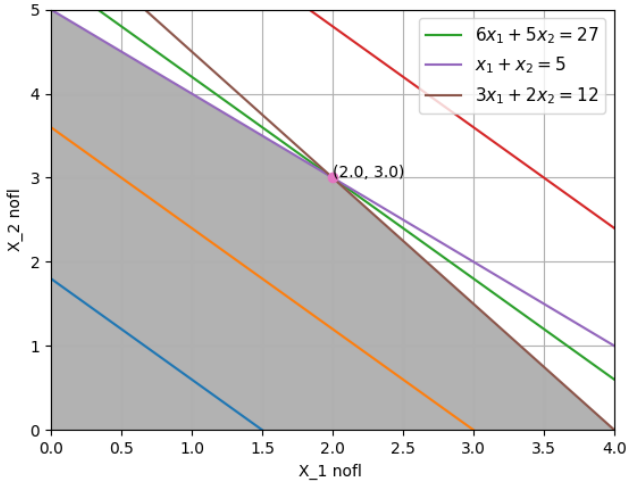


Figure 2.11: The figure shows optimal a solution LP and the feasible region

Example 2.2.2. Find the optimal solution of the linear programming

$$\begin{cases} \max_x & \mathtt{f(x)} = 3x_1 + 2x_2 \\ \text{subject to} & 2x_1 + x_2 \leq 10 \\ & 5x_1 + 6x_2 \geq 4 \\ & -3x_1 + 7x_2 = 8 \end{cases}$$

solution

$$\begin{cases} \min_x & \mathtt{f(x)} = 3x_1 + 2x_2 \\ \text{subject to} & 2x_1 + x_2 \leq 10 \\ & -5x_1 - 6x_2 \leq -4 \\ & -3x_1 + 7x_2 = 8 \end{cases}$$

Code 9 :Optimal solution of LP

```
import␣numpy␣as␣np
from␣scipy.optimize␣import␣linprog
c=np.array([3,2])
A_ub=np.array([[2,␣1],␣[-5,␣-6]])
```

```
b_ub=np.array([10,␣-4])
A_eq=np.array([[-3,␣7]])
b_eq=np.array([8])
res=linprog(-c,A_ub=␣A_ub,b_ub=b_ub,␣A_eq=A_eq,b_eq=b_eq␣)
print(res)
x:␣array([3.64705882,␣2.70588235])
```

Then the optimal solution is $[3.64705882, 2.70588235]$

2.3 Duality

Given a particular linear programming problem, there is associated with it another (unique) linear program called its dual. In general, duality theory addresses itself to the study of the connection between two related linear programming problems, where one of them, the primal, is a maximization (minimization) problem and the other, the dual, is a minimization (maximization) problem [45]. Moreover, this association is such that the existence of an optimal solution to any one of these two problems guarantees an optimal solution to the other, with the result that their extreme values are equal [45, 52]. To gain some additional insight into the role of duality in linear programming, note briefly that the solution to any given linear programming problem may be obtained by applying the simplex method to either its primal or dual formulation, the reason being that the simplex process generates an optimal solution to the primal-dual pair of problems simultaneously. So with the optimal solution to one of these problems obtainable from the optimal solution of its dual, there is no reason to solve both problems separately [52, 53]. Throughout this section consider the primal program in standard form:

$$\begin{aligned} &maximize \quad z = c^T x \\ &\text{subject to} \quad Ax \leq b \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{2.7}$$

and its corresponding dual.

$$\begin{aligned} &minimize \quad z = b^T w \\ &\text{subject to} \quad A^T w \geq c \\ &\qquad\qquad w \geq 0 \end{aligned} \tag{2.8}$$

Now let us derive the dual of LP. Suppose that have a maximization linear program in standard form.

$$\begin{cases} \text{maximize} & c_1 x_1 + \ldots c_n x_n \\ & \\ & a_{1,1} x_1 + \ldots + a_{1,n} x_n \leq b_1 \\ & \vdots \\ \text{subject to} & \\ & a_{m,1} x_1 + \ldots + a_{m,n} x_n \leq b_m \\ & \\ & x_1 \geq 0, \ldots, x_n \geq 0 \end{cases} \tag{2.9}$$

For every choice of anon-negative scaling factors $y_1, \ldots, y_m$, can derive the inequality

$$y_1 \cdot (a_{1,1} x_1 + \ldots + a_{1,n} x_n)$$

$$+ \cdots$$

$$+ y_n \cdot (a_{m,1} x_1 + \ldots + a_{m,n} x_n)$$

$$\leq y_1 b_1 + \cdots y_m b_m$$

which is true for every feasible solution $(x_1, \ldots, x_n)$ to the linear programming. can

rewrite the inequality as

$$(a_{1,1}y_1 + \cdots a_{m,1}y_m) \cdot x_1$$

$$+ \cdots$$

$$+ (a_{1,n}y_1 \cdots a_{m,n}y_m) \cdot x_n$$

$$\leq y_1 b_1 + \cdots y_m b_m$$

So get that a certain linear function of the x_i is always at most a certain value, for every feasible $(x_1, \ldots, x_n)$. The trick is now to choose the y_i so that the linear function of the x_i for which, get an upper bound is, in turn, an upper bound to the cost function of $(x_1, \ldots, x_n)$. Can achieve this if choose the y_i such that

$$c_1 \leq a_{1,1}y_1 + \cdots a_{m,1}y_m$$

$$\vdots$$

$$c_n \leq a_{1,n}y_1 \cdots a_{m,n}y_m$$

Now, see that for every non-negative $(y_1, \ldots, y_m)$ that satisfies (3), and for every $(x_1, \ldots, x_n)$ that is feasible for (2)

$$c_1 x_1 + \ldots c_n x_n$$

$$\leq (a_{1,1}y_1 + \cdots a_{m,1}y_m) \cdot x_1$$

$$+ (a_{1,n}y_1 \cdots a_{m,n}y_m) \cdot x_n$$

$$\leq y_1 b_1 + \cdots y_m b_m$$

Clearly, want to find the non-negative values $y_1, \ldots, y_m$ such that the above upper bound is as strong as possible, that is want to

$$\begin{cases} \text{minimize} & b_1 y_1 + \cdots \cdot b_m y_m \\ & a_{1,1}y_1 + \ldots + a_{m,1}y_m \geq c_1 \\ & \vdots \\ \text{subject to} & \\ & a_{n,1}y_1 + \ldots + a_{m,n}y_m \geq c_n \\ & y_1 \geq 0, \ldots, y_m \geq 0 \end{cases} \tag{2.10}$$

So, find out that if want to find the scaling factors that give us the best possible upper bound to the optimum of a linear program in standard maximization form, end up with a new linear program, in standard minimization form.

2.4 Matrix Calculus

A matrix, for which horizontal and vertical dimensions are the same (that is, an $n \times n$ matrix), is called a square matrix of order n. [78]

A square matrix $A \in \mathbb{R}^{n \times n}$ is called symmetric if $A = A^T$, that is, $(A)_{ij} = (A)_{ji}$ for all $i, j \in \{1, ..., n\}$ and $(A)_{ij}$ is the element of matrix A in row i of column j. The matrix $A^T \in \mathbb{R}^{n \times n}$ is called the transpose of A.

A square matrix $A \in \mathbb{R}^{n \times n}$ is called positive definite if

$$x^T Ax > 0 \quad forall \quad x \in \mathbb{R}^n, \quad x \neq 0$$

and negative definite if

$$x^T Ax < 0 \quad forall \quad x \in \mathbb{R}^n, \quad x \neq 0$$

Correspondingly, a square matrix $A \in \mathbb{R}^{n \times n}$ is called positive semi-definite if

$$x^T Ax \geq 0 \quad forall \quad x \in \mathbb{R}^n$$

and negative semi-definite if

$$x^T Ax \leq 0 \quad forall \quad x \in \mathbb{R}^n$$

A matrix which is neither positive nor negative semi-definite is called indefinite.

An inverse of matrix $A \in \mathbb{R}^{n \times n}$ is a matrix $A^{-1} \in \mathbb{R}^{n \times n}$ such that:

$$AA^{-1} = A^{-1}A = I,$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix.

A scalar λ is called an eigenvalue of the matrix $A \in \mathbb{R}^{n \times n}$ if

$$Ax = \lambda x$$

for some non-zero vector $x \in \mathbb{R}^n$. The vector x is called an eigenvector associated to the eigenvalue λ.

Remark: Symmetric matrix have special eigenvalue properties

1. The eigenvalues of **A** are real .
2. There exist unit-norm and mutually orthogonal eigenvectors $x_1, ..., x_n$ associated with eigenvalues λ_1 ,...., λ_n of **A** ; $\mathbf{A}x_i = \lambda_i x_i$, with $\|x\| = 1$ for all i=1,..,n : $x_i^\top x_j = 0$ for all

$i \neq j$

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i x_i x i^{\top}$$

Definition 2.4.1. Trace of matrix $\mathcal{A} \in R^{n\times n}$ is defined by $tr(X) = \sum_{i=1}^{n} \mathcal{A}_{ii}$, **Indeed** trace of matrix is linear [28], let $\mathcal{A}$,$\mathcal{B} \in \mathcal{R}^{n\times n}$ and let λ , $\alpha \in \mathcal{R}$

$$\begin{aligned} tr(\alpha\mathcal{A} + \lambda\mathcal{B}) &= \sum_{i=1}^{n}(\alpha\mathcal{A}_{ii} + \lambda\mathcal{B}_{ii}) = \sum_{i=1}^{n}\alpha\mathcal{A}_{ii} + \sum_{i=1}^{n}\lambda\mathcal{B}_{ii} \\ &= \alpha\sum_{i=1}^{n}\mathcal{A}_{ii} + \lambda\sum_{i=1}^{n}\mathcal{B}_{ii} \\ &= \alpha\mathbf{tr}(\mathcal{A}) + \lambda\mathbf{tr}(\mathcal{B}) \end{aligned}$$

Definition 2.4.2. Polyhedron and Spetrahedron [56]

A polyhedron ; $\mathcal{P} \in \mathcal{R}^n$ is the set of the point that satisfies affinity number of liner inequality.

$\mathcal{P} = (x \in \mathcal{R}^n : \mathcal{A} \leq b)$ where $(\mathcal{A}, b)$ is an $mx(n+1)$ matrix. Can say which is the set linear qualities and inequalities

$$\{x \in R^n | \mathbf{Ax} = b, \mathbf{cx} \leq d, (\mathtt{A}, \mathtt{B})\text{matrex}, (\mathtt{a}, \mathtt{b})\text{vector}\}$$

A spetrahedron is the intersection of the cone of positive semi-definite matrices with affine liner space. **In fact** every polyhedron is spetrahedron but the spetrahedron is far richer geometric objects than polyhedron.

proposition 1. *[80] The characterizations of positive semi-definite matrices*

$$\begin{aligned}
&\mathbf{X} \succeq 0 \\
&\Leftrightarrow y^T X y \geq 0, \forall y \in \mathbb{R}^n \\
&\Leftrightarrow \textit{All eigenvalues of} \quad X \quad \textit{are} \geq 0 \\
&\Leftrightarrow \textit{Sylvester's critierion holds: all} \quad 2^n - 1 \quad \textit{principal minors of} \quad X \quad \textit{are nonnegative} \\
&\Leftrightarrow X = MM^T, \quad \textit{for some} \quad n \times k \quad \textit{matrix} \quad M.
\end{aligned}$$

This is called a Cholesky factorization.

Now let us derive the duilty of (**SDP**) a semi-definite program is an optimization problem of the form

$$\begin{cases}
\min_{X \in S^{n \times n}} & \mathbf{Tr}(\mathbf{C}X) \\
\text{subject to} & \mathbf{Tr}(A_i X) = b_i, \quad i = 1, ..., m \\
& \mathbf{X} \succeq 0 \\
& \text{where} \quad \mathbf{C}, \mathbf{A_i} \in \mathbf{S^{n \times n}} \quad \text{and} \quad b_i \in \mathbf{R^n}
\end{cases} \tag{2.11}$$

where $\mathbf{S^{n \times n}}$ denoted the set of $n \times n$ real symmetric matrices and **Tr** denote the trace of a matrix. It has two types of constraints

(a) Affine constraints in the entries of the decision matrix $\mathbf{X}$

(b) A constraint forcing some matrix to be positive semi-definite. The trace notation is use as a convenient way of expressing affine constraints in the entries of our unknown matrix. If $\mathbf{A}$ and $\mathbf{X}$ are symmetric we have

$$\mathbf{Tr}(\mathbf{AX}) = \sum_{ij} \mathbf{A_{ij}} \mathbf{X_{ij}}$$

Infact **SDP** is very natural generalization of **LP** but the expressive power of **SDPs** is much richer than **LPs**, can solve **SDPs** efficiently (in polynomial time to arbitrary accuracy).

Now Let us deriving the dual of an **SDP**, consider the primal **SDP**:

$$\begin{cases} \min & \mathbf{Tr}(\mathbf{C}X) \\ \text{subject to} & \mathbf{Tr}(A_iX) = b_i, i = 1, ..., m \quad \mathbf{X} \succeq 0 \end{cases} \tag{2.12}$$

and denoted its optimal value by $\mathbf{p}^* = \inf\{\mathbf{Tr}(\mathbf{C}X), \mathbf{Tr}(A_iX), i = 1, ..., m\}$, to derive the dual, defined the Lagrangian function

$$\mathbf{L}(x, \lambda, \mu) = Tr(CX) + \sum_i \lambda_j(b_i - Tr(A_iX)) - Tr(X\mu)$$

and the dual function

$$\mathbf{D}(\lambda, \mu) = min_X \mathbf{L}(X, \lambda, \mu)$$

the dual problem is then given by

$$\begin{cases} \max_{\lambda,\mu} & \mathbf{D}(\lambda, \mu) \\ \text{subject to} & \mu \succeq 0 \end{cases} \tag{2.13}$$

Definition 2.4.3. [22] **The Proximal Operator**

The proximal operator $prox_f : \mathbb{R}^n \to \mathbb{R}^n$ of the convex function $\mathbb{R}^n \to \mathbb{R}^n$ is defined by $prox_f(y) = argmin_{x\in\mathbb{R}^n}(f(x) + \frac{1}{2}\|x - y\|_2^2)$ the function $x \to f(x) + \frac{1}{2}\|x - y\|_2^2$ is sturgeon convex and has a unique minimizer for every $y \in \mathbb{R}$(f is strongly convex function on set S if there is a constant $b > 0$such that for all $x_1, x_2 \in S$ and for all $\lambda \in [0, 1]$) the **proximal operator** of f with parameter $\alpha > 0$ is given by

$$prox_{\alpha f}(y) = argmin_{x\in\mathbb{R}^n}(f(x) + \frac{1}{2\alpha}\|x - y\|_2^2)$$

. Form the following properties hold for all $\alpha > 0$

- **The Proximal Operator of aconstant** if $f \equiv c$ (c constant) for some $c \in R$

$$prox_{\alpha f}(y) = argmin_{x \in \mathbb{R}^n}(f(x) + \frac{1}{2\alpha}\|x - y\|_2^2)$$
$$= prox_{\alpha f}(y) = argmin_{x \in \mathbb{R}^n}(c + \frac{1}{2\alpha}\|x - y\|_2^2)$$

satisfy the first optimality conditions (the gradient of the objective function=0), therefore $prox_f(x) = x$

- **The Proximal Operator of linear function** let $a \in R^n, b \in R$ and $f(x) = a^\intercal x + b$ then the proximal operator is

$$prox_{\alpha f}(y) = argmin_{x \in \mathbb{R}^n}(f(x) + \frac{1}{2\alpha}\|x - y\|^2)$$
$$= prox_{\alpha f}(y) = argmin_{x \in \mathbb{R}^n}(a^\intercal x + b + \frac{1}{2\alpha}\|x - y\|^2)$$

satisfy the first optimality conditions (the gradient of the function=0), therefore $prox_{\alpha f}(y) = (y - ab)$

- **The Proximal Operator of Convex Quadratic Function**

let $\mathbb{R}^n \to R$ be given by $f(x) = \frac{1}{2}x^\intercal \mathbf{A}x + b^\intercal x + c$, where $\mathbf{A} \in S_+^n$, $b \in R$ and $c \in R$ the vector $prox_f(x)$ is minimizer of the problem.

$$prox_f(y) = argmin_{x \in R^n}\left\{\frac{1}{2}x^\intercal \mathbf{A}x + b^\intercal x + c + \frac{1}{2}\,||\,x - y\,||^2\right\}$$

and satisfy the first optimality conditions (the gradient of the objective function= 0

). therefore .

$$\begin{aligned} \mathbf{A}x + b + y - x &= 0 \\ (\mathbf{A} + \mathbf{I})x + (b - y) &= 0 \\ (\mathbf{A} + \mathbf{I})x &= (y - b) \\ prox_f(x) &= (\mathbf{A} + \mathbf{I})^{-1}(y - b) \end{aligned}$$

Definition 2.4.4. Feasible set of (SDPs) is called a spectrahedron. Every polyhedron is a spectrahedron (this is because every LP can be written as an SDP), but spectrahedra are far richer geometric objects then polyhedra (this is the reason why SDP is in general more powerful than LP [78].

Spectrahedra are always convex sets:

- Positive semi-definite $n \times n$ matrices form a convex set .
- A fine constraints define a convex set.
- Intersection of convex sets is convex.

When say an SDP is a convex optimization problem

- The objective is an affine function of the entries of the matrix.
- The feasible set is a convex set.
- However, the feasible set is not written in the explicit functional form "convex functions ≤ 0, affine function $= 0$

To get a functional form, one can write an SDP as an infinite LP:

- Replace $X \succeq 0$ with linear constraints $y_i^T X y_i \geq 0$, for all $y \in \mathbb{R}^n$
- We can reduce this to be a countable infinity by only taking $y \in \mathbb{Z}^n$

Alternatively, can write an SDP as a non-linear program by replacing $X \succeq 0$ with $2^n - 1$ minor inequalities coming from Sylvester's criterion. However, treating the matrix constraint $X \succeq 0$ directly is often the right thing to do.

Lemma 2.4.1. [38] LP as a special case of SDP.

Consider an LP

$$\begin{cases} \min_{x \in \mathbb{R}^n} & \mathsf{c}^T x \\ \text{subject to} & a_i^T x = b_i, i = 1, \ldots, m \quad x \geq 0 \end{cases} \tag{2.14}$$

for a vector v,, let $\text{diag}(v)$ denoted the diagonal matrix with v on its diagonal. Then, can write our LP as the following SDP :

$$\begin{cases} \min_X & \text{Tr}(\text{diag}(c)X) \\ \text{subject to} & \text{Tr}\left(\text{diag}\left(a_i\right) X\right) = b_{i,} i = 1, \ldots, m_s \quad X \succeq 0 \end{cases} \tag{2.15}$$

so LP is really a special case of SDP where all matrices are diagonal positive semi-definiteness for a diagonal matrix simply means non-negativity of its diagonal elements.

Now; the optimal of duilty problem, if $\mathbf{p}^*$ is the optimal of primal problem then $\mathbf{D}(\lambda, \mu) \leq \mathbf{p}^*$

Consider the primal SDP:

$$\begin{cases} \min_{X \in \mathbb{S}^{\text{nxn}}} & \text{Tr}(CX) \\ \text{subject to} & \text{s.t. } \text{Tr}\left(A_i X\right) = b_i, i = 1, \ldots, m \quad X \succeq 0 \end{cases} \tag{2.16}$$

and denoted its optimal value by p^*, to derive the dual, defined the Lagrangian function

$$L(X, \lambda, \mu) = \text{Tr}(CX) + \sum_i \lambda_i \left(b_i - \text{Tr}\left(A_i X\right)\right) - \text{Tr}(X\mu)$$

want to show for any $\lambda, \mu \succeq 0$, have

$$D(\lambda, \mu) \leq p^*$$

first prove a basic linear algebra fact, namely, if $A \succeq 0$ and $B \succeq 0$ then $\mathrm{Tr}(AB) \geq 0$. Indeed, as $A \succeq 0$ and $B \succeq 0, \exists M, N$ such that $A = MM^T$ and $B = NN^T$ using the Cholesky decomposition, then

$$\mathrm{Tr}(AB) = \mathrm{Tr}\left(MM^T NN^T\right) = \mathrm{Tr}\left(N^T MM^T N\right) = \left\|M^T N\right\|_F^2 \geq 0$$

Now, let X^* be a primal optimal solution.
Then in view of the fact that $b_i - \mathrm{Tr}\left(A_i X^*\right) = 0$, and $X^*, \mu \succeq 0$, have

$$L\left(X^*, \lambda, \mu\right) = \mathrm{Tr}\left(CX^*\right) - \mathrm{Tr}\left(X^* \mu\right) = p^* - \mathrm{Tr}\left(X^* \mu\right) \leq p^*$$

where the last inequality follows from the claim just proved above. Hence, see that

$$D(\lambda, \mu) = \min_X L(X, \lambda, \mu) \leq p^*$$

Example 2.4.1. consider the problem

$$\begin{cases} \min_x \quad x_{11} + x_{12} \\ \text{subject to} \quad x_{11} + x_{22} = 1 \\ \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \succeq 0 \quad (\succeq \text{ means positive definite}) \end{cases}$$

Is denoted by semi-definite program.

$$\begin{cases} \min_x \quad (1 \quad 1 \quad 0) \begin{pmatrix} x_{11} \\ x_{12} \\ x_{22} \end{pmatrix} \\ \text{subject to} \quad (1 \quad 0 \quad 1) \begin{pmatrix} x_{11} \\ x_{12} \\ x_{22} \end{pmatrix} = 1 \\ x_{11} \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix} + x_{12} \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} + x_{22} \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix} \preceq \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{cases}$$

Code 10 :Optimal solution of SDP

```
from␣cvxopt␣import␣matrix
from␣cvxopt␣import␣solvers
c=matrix([1.,␣1.,␣0.])
G␣=␣[matrix([[-1.,␣0.,␣0.,␣0.],␣[0.,␣-1,␣-1,␣0.],␣[0.,␣0.,␣0.,␣-1.]])]
Aval␣=␣matrix([1.,␣0.,␣1.],␣(1,␣3))
bval=␣matrix([1.])
h=␣[matrix([[0.,␣0.],␣[0.,␣0.]])]
sol=solvers.sdp(c,␣Gs=G,␣hs=h,␣A=Aval,␣b=bval)
print(sol['x'])
Optimal␣solution␣found.
[␣1.46e-01]
[-3.54e-01]
[␣8.54e-01]
```

Remark

- Linear Programming (LP)
- (Convex) Quadratic Programming (QP)
- (Convex) Quadratically Constrained Quadratic Programming (QCQP)
- Second Order Cone Programming (SOCP)
- Semi-definite Programming (SDP)
- $LP \subseteq$ (convex)$QP \subseteq$ (convex)$QCQP \subseteq SOCP \subseteq SDP$

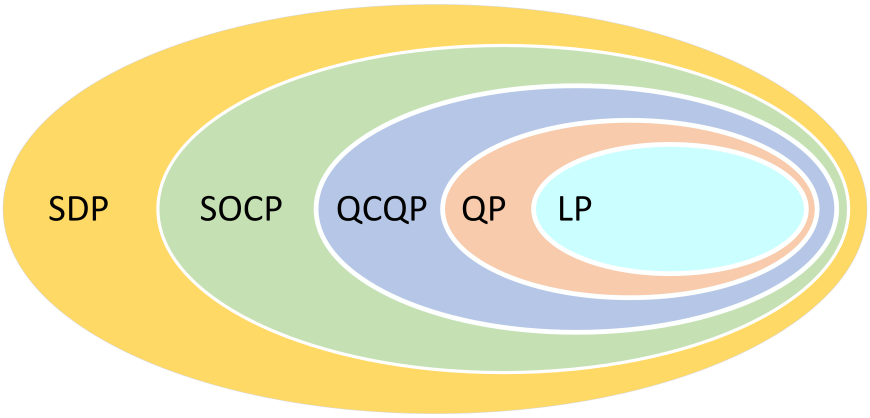


Figure 2.12: Linear Programming subset of Semi-definite Programming

CHAPTER 3

NON-LINEAR PROGRAMMING(APPROXIMATE METHODS)

In this chapter, present general constrained non-linear programming problems. A penalty approach is a method that uses a sequence of unconstrained problems. This approach replaces a constrained problem with unconstrained problems by adding a penalty term to an objective function. It can be said that, goal makes the penalty parameter infinitely large, in order to get an optimum solution. However, a single unconstrained minimization problem with a fairly sized penalty parameter can generate an optimal solution to the original issue. Obtain an optimal solution to these functions, it is not necessary solve an endless sequence of penalty problems. To ease the numerical difficulty involved with the need pick the penalty parameter to infinity to restore the optimum solution to the original issue. In addition, comparing between internal and external penalty function, also using the augmented Lagrange method to solve the constrained non-linear programming problems.

INTRODUCTION

Optimization theory and techniques are new topics in applied mathematics, operations research, and computational mathematics with a large range of applications in scientific and engineering, business administration, and space technology [38, 76]. This topic participates in the optimum solution of problems that are determined mathematically [9], that is, in light of a practical issue, through many schemes and using scientific methods and tools, a better solution to the problem can be obtained. In the late 1940's, optimization became a separate topic when George Bernard Dantzig [39] introduced the popular simplex algorithm for linear programming [39]. Since the 1950s, when conjugate gradient techniques and quasi-Newton techniques were introduced, non-linear programming evolved considerably. Many modern optimization approaches can now solve challenging and large-scale optimization problems and become a necessary tool for solving problems in various fields [21]. Optimization may contain constrained or unconstrained problems. A limitation in non-linear programming leads to more difficulties in seeking a minimum as comparison to the unconstrained limitations. Also many situations may be defined based on the influence of the restrictions on the objective function. Penalty methods are popular techniques for working with constrained optimization problems. This approach replaces a constrained problem with unconstrained problems by adding a penalty term to an objective function. They presumed that f(x) and the inequality restrictions are convex and the equality restrictions are affine functions to show that the "Karush–Kuhn–Tucker" point in the original optimization problem is a minimizer of the exact absolute value penalty function in the related penalized optimization problem with a sufficiently great value of a penalty parameter.

3.1 Convert Non-liner Programming to Liner Programming

Theorem 3.1.1. ***First Order Necessary Condition*** *[13] If x^* is a local minimizer of $f(x)$ and $f(x)$ is continuously differentiable in an open neighbourhood of x^* ($f \in \mathcal{C}^1$) then*

$$\nabla f(x^*) = 0$$

That is , $f(x)$ is stationary at x^*

Theorem 3.1.2. ***Second Order Necessary Conditions*** *[13] If x^* is a local minimizer of $f(x)$ and $\nabla^2 f(x)$ is continuously differentiable in an open neighborhood of x^* ($f \in \mathcal{C}^2$) then .*

$$\nabla f(x^*) = 0$$
$$d^\intercal \nabla^2 f(x^*) d \geq 0$$

Theorem 3.1.3. ***Second Order Sufficient Condition*** *[13] Suppose $f(x)$ and $\nabla^2 f(x)$ is continuously differentiable in an open neighborhood of x^* if the following to condition are satisfy then x^* is a local minimum of $f(x)$.*

$$\nabla f(x^*) = 0$$
$$d^\intercal \nabla^2 f(x^*) d > 0$$

this means $\nabla^2 f(x) \succ 0$ where d is belong to feasible direction .

Definition 3.1.1. Karush-Kuhu-Tucker(KKT) condition for a non-linear programming

problem [55].

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & g_j(x) \leq 0 \quad \text{for} \quad j = 1, \ldots, p \\ & h_k(x) = 0 \quad \text{for} \quad k = 1, \ldots, m \\ & x \in \mathtt{R^n} \end{cases} \tag{3.1}$$

the Lagrangian $L(x, u, v) = f(x) + \sum_{i=1}^{m} v_i h_i(x) + \sum_{j=1}^{p} u_j g_j(x)$

$\Rightarrow L(x, u, v) = f(x) + v^\top h(x) + u^\top g(x)$

(a) Gradient Conditions

$$\frac{\partial(x^*, u^*, v^*)}{\partial x_i} = \frac{\partial f(x^*)}{\partial x_i} + \sum_{i=1}^{m} {v^*}_i \frac{\partial h_j(x^*)}{\partial x_i} + \sum_{j=1}^{p} {u^*}_j \frac{\partial g_j(x^*)}{\partial x_i} = 0$$

where $\quad i = 1, \ldots, n$

(b) Feasibility Check

$$g_j(x) \leq 0 \quad \text{for} \quad j = 1, \ldots, p$$

$$h_k(x) = 0 \quad \text{for} \quad k = 1, \ldots, m$$

The inequality and equality constrained have to be satisfied, the candidate optimal solution will satisfy the equality constrained as will as inequality constrained.

(c) Switching Condition

${u^*}_j g_j = 0, j = 1, \ldots, p$ The non-negativity of Lagrange multiplier for inequality constraints ${u_j}^* \geq 0 \quad j = 1, \ldots, p$

(d) Regularity Check

gradient of active constraints must be linearly independent

The points that satisfy all these KKT conditions are known as KKT point, it is likely candidate for optimum and the KKT condition " first order conditions for optimality"

Theorem 3.1.4. *If $f(x)$ is convex and $g_i(x) \quad \forall i \in I$ are convex function then a feasible KKT point is optimal [20].*

Theorem 3.1.5. ***KKT Necessary Optimality Conditions*** *[20] If x^o is a local maximum , there is multipliers $\mu_i \geq 0 \quad \forall i \in I$ and $\lambda_j \geq 0 \quad \forall j \in J$ such that*

$$\nabla f(x^o) - \sum_{i \in I} \mu_i \nabla g_i(x^o) - \sum_{j \in J} \lambda_j \nabla h_j(x^o) = 0$$

Theorem 3.1.6. ***KKT Sufficient Optimality Conditions*** *[20] If $f(x)$ is concave, $g_i(x) \quad \forall i \in I$ are convex functions and $h_j \quad \forall j \in J$ are affine (linear) then the feasible KKT point is optimal .*

Example 3.1.1. Locate all of the point of the KKT point for the following problem. Are these points local solution?Are the global solution ?

$$\begin{aligned} \textit{minimize} \quad & x_1^2 + x_2^2 - 4x_1 - 4x_2 \\ \textit{subject} \quad & x_1^2 \leq x_2 \\ & x_1 + x_2 \leq 2 \end{aligned}$$

solution: First write the problem in the standard from required for the application of the KKT theory :

$$\begin{aligned} f_o(x_1, x_2) &= x_1^2 + x_2^2 - 4x_1 - 4x_2 \\ f_1(x_1, x_2) &= x_1^2 - x_2 \\ f_2(x_1, x_2) &= x_1 + x_2 - 2 \end{aligned}$$

$$L((x_1, x_2), (u_1, u_2)) = x_1^2 + x_2^2 - 4x_1 - 4x_2 + \mu_1(x_1^2 - x_2) + \mu_2(x_1 + x_2 - 2)$$

Let us now write the KKT condition

(a) (Primal Feasibility) $x_1^2 \leq x_2$ and $x_1 + x_2 \leq 2$

(b) (Dual Feasibility) $0 \leq \mu_1$ and $0 \leq \mu_2$

(c) (Complementary) $\mu_1(x_1^2 - x_2) = 0$ and $\mu_2(x_1 + x_2 - 2) = 0$

(d) (Stationary of the Lagrangian)$\nabla_x L((x_1, x_2), (\mu_1, \mu_2)) = 0$

$$4 = 2x_1 + 2\mu_1 x_1 + \mu_2$$
$$4 = 2x_2 - \mu_1 + \mu_2$$

The global minimizer for the object function is $(x_1, x_2) = (2, 2)$ if this point is feasible, it will be the global solution and the multiples would both be zero. But it is not feasible, both constraint are active at the solution. In this case KKT pair$((x_1, x_2), (\mu_1, \mu_2))$ satisfy

$$x_2 = x_1^2$$
$$2 = x_1 + x_2$$
$$4 = 2x_1 + 2\mu_1 x_1 + \mu_2$$
$$4 = 2x_2 - \mu_1 + \mu_2$$

$x_1^2 + x_1 - 2 = (x_1 + 2)(x_1 - 1) = 0$ so $x_1 = -2$ or $x_1 = 1$

Thus, either$(x_1, x_2) = (-2, 4)$ or $(x_1, x_2) = (1, 1)$

Since(x_1, x_2) is closer the global minimizer of the object f_0 to see (x_1, x_2) if it KKT point plug $(1, 1) = (x_1, x_2)$, get

$2 = 2\mu_1 + \mu_2$ and $2 = -\mu_1 + \mu_2$ then get $\mu_1 = 0$ and $\mu_2 = 2$

Since $\mu_1, \mu_2 \geq 0$ then $(1, 1)$ KKT point and by the convexity it is global solution .

Example 3.1.2. Use the Lagrangian method to solve constraint optimization problems

$$\begin{cases} \min_x & \mathtt{f(x)} = 4(x_1)^2 + 2(x_2)^2 \\ \text{subject to} & f_1(x) = 3x_1 + x_2 - 8 = 0 \\ & f_2(x) = 15 - 2x_1 - 4x_2 \geq 0 \\ & x_1, x_2 \geq 0 \end{cases} \tag{3.2}$$

solution: The Lagrangian

$$L(x,\lambda) = f(x) + \lambda_1 f_1(x) - \lambda_2 f_2(x) = 4(x_1)^2 + 2(x_2)^2 + \lambda_1(3x_1 + x_2 - 8) - \lambda_2(15 - 2x_1 - 4x_2)$$

$$\nabla L(x,\lambda) = \begin{pmatrix} 8x_1 + 3\lambda_1 + 2\lambda_2 \\ 4x_2 + \lambda_1 + 4\lambda_2 \\ 3x_1 + x_2 - 8 \\ -2x_1 - 4x_2 + 15 \end{pmatrix} = 0 \Rightarrow \begin{pmatrix} 8 & 0 & 3 & 2 \\ 0 & 4 & 1 & 4 \\ 3 & 1 & 0 & 0 \\ 2 & 4 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 8 \\ 15 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 1.7 \\ 2.9 \\ -3.12 \\ -2.12 \end{pmatrix}$$

Code 11:Use Lagrangian Method to Solve Constraint Optimization Problems

```
import numpy as np
import matplotlib.pyplot as plt
A = np.matrix('8,0,3,2;,0,4,1,4;,3,1,0,0;,2,4,0,0')
b= np.matrix('0,;,0,;,8;,15')
xs= np.array(np.linalg . inv(A)*b)
r=np.sqrt(4*xs[0]**2+2*xs[1]**2)
print(xs)
print(r)
```

The solution is:

$$x_1 = 1.7, x_2 = 2.9, \lambda_1 = -3.12, \lambda_2 = -2.12, f(x_1, x_2, \lambda_1, \lambda_2) = [5.32728824]$$

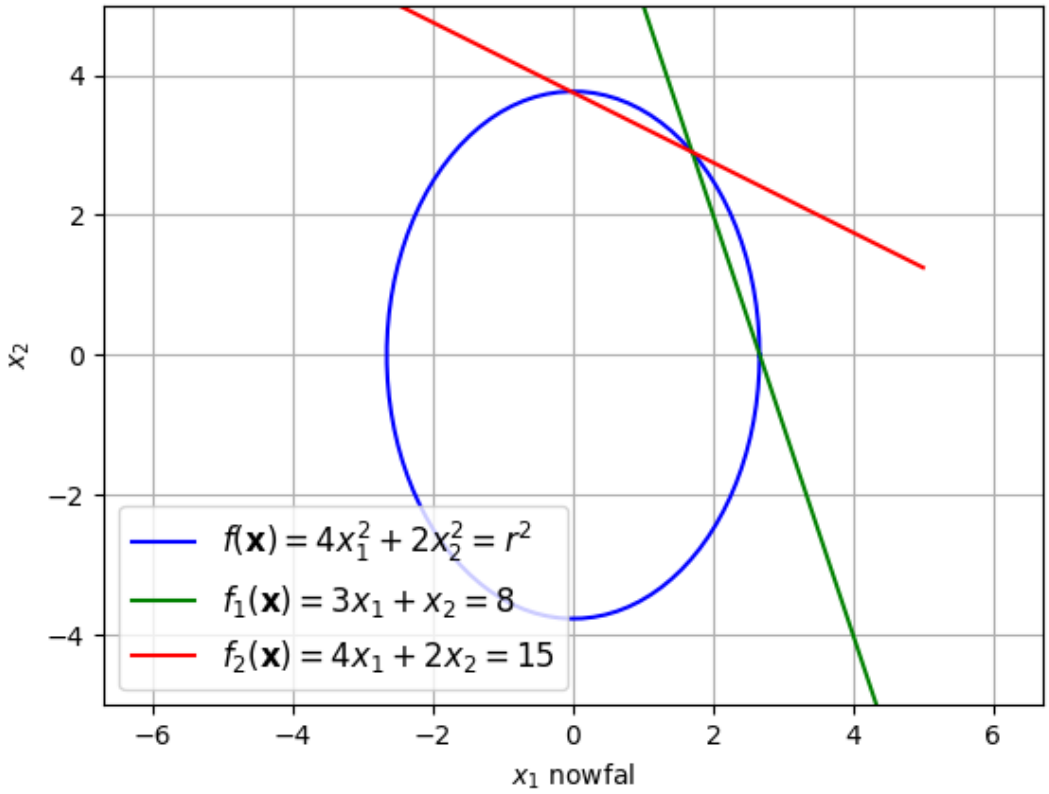


Figure 3.1: Figure shown the optimal solution at point $x^* = (1.7, 2.9)$

3.2 Solve A General Constrained Non-Linear Programming

Non-linear Programming (NLP) problems were the objective or at least one of the constraints are defined by non-linear functions [46].

The general form of a non-linear optimization problem is:

$$\begin{cases} \underset{x}{\text{Minimize}} & f(x) \\ \text{subject to} & h_j(x) = 0, \quad j = 1, 2, ..., L \\ & g_i(x) \leq 0, \quad i = 1, 2, ..., M \\ & x \in \mathbb{R}^n. \end{cases} \tag{3.3}$$

Where, assume that all the functions are smooth. The feasible set of the (NLP) is given by:

$$X = \{x \in \mathbb{R}^n | h_j(x) = 0 \quad for \quad j = 1, ...L; g_i(x) \leq 0 \quad for \quad i = 1, ..., M\}.$$

through out this dissertation our interest is in solving non-linear programming problems by classifying them primarily as unconstrained and constrained optimization problems. Particularly, if the feasible set $X = \mathbb{R}^n$, the optimization problem is called an unconstrained optimization problem where as the problems of type(see equation (3.3))are said to be constrained optimization problems.

Generally, Optimization problems can be classified as unconstrained optimization problem and constrained optimization problems.

There is a group of ways to solve non-linear programming [55, 62, 75].

(a) Sequential linear programming method

(b) Successive quadratic programming method

(c) Penalty function method

(d) Lagrange multiplier method (Augmented Lagrange Multiplier Method)

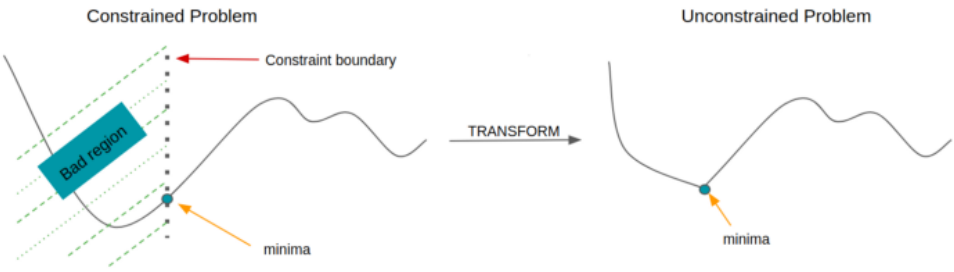


Figure 3.2: shown the effect of the constrained on the problems

3.2.1 Non-linear Programming

The name suggest a non-linear programming problem will be solved by a series of solving linear programming problem [62].

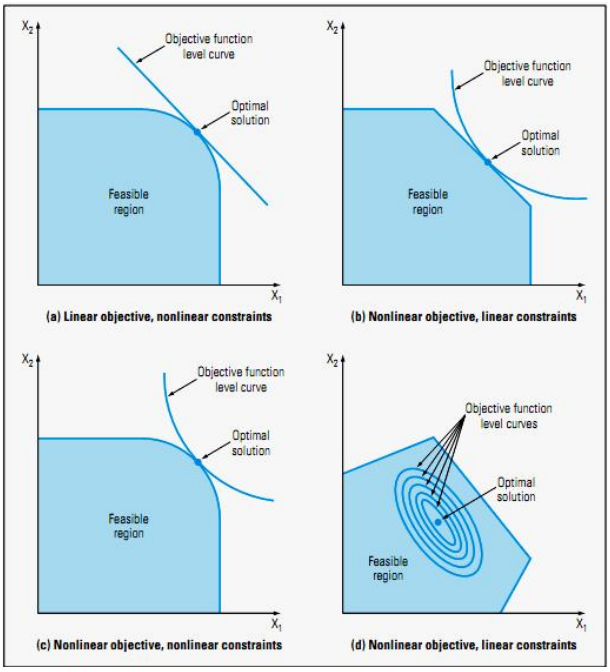


Figure 3.3: Type of NLP

Note that for linear programming problem, the optimal solution will always be a corner point but in case of non-linear programming problem, the optimal solution may lie anywhere within the feasible region.

Let us talk about the methods that are available for solution of constrained non-linear programming problems can be classified into two broad categories: direct methods and

indirect methods

(a) **Direct methods** [62]

Handle the constraints in an explicit manner

i. Random search methods

ii. Heuristic search methods

iii. sequential linear programming method

iv. sequential quadratic programming method

v. Generalized reduced gradient method

(b) **Indirect methods** [37]

Solve the constrained problem as a sequence of unconstrained problem

i. Sequential unconstrained minimization

ii. Penalty function method

iii. Augment Lagrange multiplier method

3.2.2 Characteristics of a constrained optimization

(a) **No Effect of Constraints** :

The constraint may have no effect on the optimum point, in this case the constrained optimum is the same as the unconstrained optimum. In this case the minimum point x^* can be found by using the necessary and sufficient conditions $\nabla f(x^*) = 0$, $\nabla^2 f(x) =$ positive definite

(b) **Constrained Optimum will usually be different**

Define x^* be optimal but the constraints when they are present will usually have a different optimal than the unconstrained case, look to diagram can see from the contours that you see approach this point, the function value increases for$50, 55$to56.

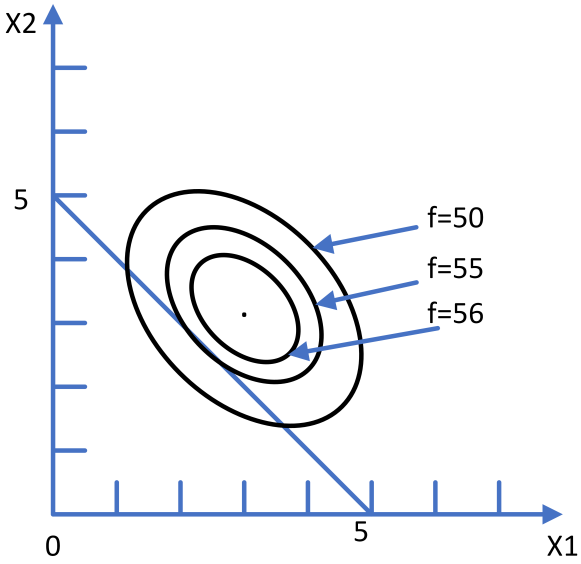


Figure 3.4: optimum $x_1 = 2.5, x_2 = 2.5$ unconstrained optimum $x_1 = 3, x_2 = 3.5$

(c) **Optimum occurs on constraint boundary according to the KKT necessary conditions**

The negative of the gradient can be expressed as a positive linear combination of the gradients of the active constraints.

$$-\nabla f(x^*) = \sum_{i=1}^{m} {v_j}^* \nabla h(x^*) + \sum_{j=1}^{p} {u_j}^* \nabla g(x^*)$$

(d) **Multiple Optimal**

If the objective function has multiple unconstrained local minima, the constrained problem may also have multiple minima. Sometimes, even if the objective function has a single unconstrained minimum the constraints may introduce additional local minima.

Definition 3.2.1. Simplex Method

Standard technique in linear programming for solving an optimization problem, typically one involving a function and several constraints expressed as inequalities. The inequalities

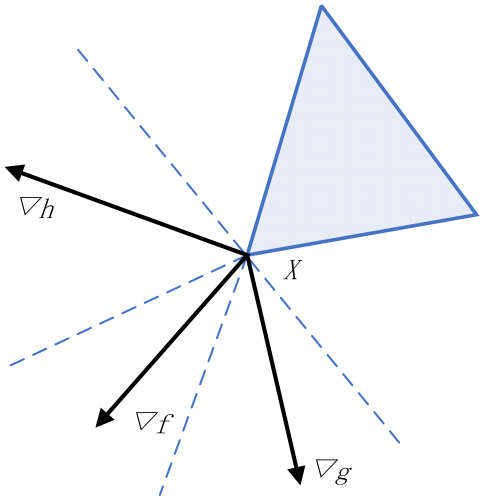


Figure 3.5: positive linear combination of the gradients

define a polygonal region, and the solution is typically at one of the vertices. The simplex method is a systematic procedure for testing the vertices as possible solutions [70].

3.3 Sequential unconstrained minimization

The original non-linear programming problem is solved by solving a series of linear programming problem (LPP), each LPP is obtained by approximating the non-linear and constraint functions using first order Taylor series expansions about the current estimate of the optimum (decision/design variable vector) x_i [37]. The LPP can be solved by simplex method to find the new design vector x_{i+1}, if x_{i+1} is not the desired (converged) solution, the problem is realized about the point x_{i+1} and the procedure is repeated until find the optimum solution x^*.

Consider the NLP

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{cases} \tag{3.4}$$

this differs from standard LPP only in the presence of non-linear objective function $f(x)$. Now the optimal solution can lie any where within the feasible region (not necessarily on

the corner points only)the linearisation of the above problem around some feasible point x^o is clearly an LPP and it will have an optimal solution at a feasible corner point, if the feasible region is bounded

$$\begin{cases} \min & \bar{\mathtt{f}}(\bar{\mathtt{x}}, \mathtt{x}^\circ) \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{cases} \tag{3.5}$$

Infact the solution of the approximate problem is related to the solution of the original problem by using type of line search is required in order to approach the solution of the original problem from the approximate solution.

Algorithm 1 [30]: Frank-wolfe Algorithm.

(a) Define starting point $x^{(0)}$; termination parameters $\epsilon > 0$, $\delta > 0$

compute $\nabla f(x^t)$, if $\left\|\nabla f(x^t)\right\| \leq \epsilon$

stop,else go to step (b)

(b) Solve LP subproblem

$$\begin{cases} \min & \nabla f(x^{(t)})y \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{cases} \tag{3.6}$$

Let y^t be the optimal solution to the LPP

(c) Find α^t which solves:

$$\begin{cases} \min f(x^t + \alpha(y^t - x^t)) \\ 0 \leq \alpha \leq 1 \end{cases} \tag{3.7}$$

(d) Calculate : $x^{(t+1)} = x^{(t)} + \alpha^{(t)}(y^{(t)} - x^{(t)})$

(e) Convergence check:

$$\text{If } \left\|x^{(t+1)} - x^{(t)}\right\| < \delta \left\|x^{(t+1)}\right\|$$

And if

$$\left\|f(x^{(t+1)} - x^{(t)})\right\| < \epsilon \left\|f(x^{(t+1)})\right\|$$

Then stop, else go to step (a)

Now consider the following general NLP .

$$\begin{cases} \min & f(x) \\ \text{subject to} & g_i(x) \geq 0 \quad j = 1, \ldots, j \\ & h_k(x) = 0 \quad k = 1, \ldots, K \\ & {x_i}^{LB} \leq x_i \leq {x_i}^{UB} \end{cases} \tag{3.8}$$

solution to this LPP give us $x^{(t+1)}$, if $x^{(t+1)}$ is infeasible reinitialize at $x^{(t+1)}$ given an estimate x^t, the following linear approximation problem can be constructed at x^t

$$\begin{cases} \min & f(x^t) + \nabla f(x^t)(x - x^t) \\ \text{subject to} & g_i(x^t) + \nabla g_i(x^t)(x - x^t) \geq 0 \quad j = 1, \ldots, j \\ & h_k(x^t) + \nabla h_i(x^t)(x - x^t) = 0 \quad k = 1, \ldots, K \\ & {x_i}^{LB} \leq x_i \leq {x_i}^{UB} \end{cases} \tag{3.9}$$

Example 3.3.1. Consider the following NLP, will convert the NLP problem to linear problem .

$$\begin{cases} \min & f(x) = {x_1}^2 + {x_2}^2 \\ \text{subject to} & g(x) = {x_1}^2 + x_2 \geq 0 \\ & h(x) = 2 - x_1 + {x_2}^2 = 0 \\ & 0.5 \leq x_1 \leq 2.5 \quad 0 \leq x_2 \leq 3 \end{cases} \tag{3.10}$$

want to construct the linear approximation to this problem at $x^o = (2, 1)$ $\nabla f(x^t) = \begin{bmatrix} 2x_1 \\ \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ \\ 2 \end{bmatrix}$ $f(2, 1) = 5, \text{Now} \quad f(x^t) + \nabla f(x^t)(x - x^t)$

$$= 5+[4 \quad 2]\begin{bmatrix} x_1-2 \\ \\ x_2-1 \end{bmatrix} = 5+4(x_1-2)+2(x_2-1)$$

$$\nabla g(x^t) = \begin{bmatrix} 2x_1 \\ \\ -1 \end{bmatrix} = \begin{bmatrix} 4 \\ \\ -1 \end{bmatrix} \qquad g(2,1)=3, \text{Now} \quad g(x^t)+\nabla g(x^t)(x-x^t)$$

$$= 3+[4 \quad -1]\begin{bmatrix} x_1-2 \\ \\ x_2-1 \end{bmatrix} = 3+4(x_1-2)-(x_2-1)$$

$$\nabla h(x^t) = \begin{bmatrix} -1 \\ \\ -2x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ \\ -2 \end{bmatrix}$$

h(2,1)=-1, Now $\quad h(x^t)+\nabla h(x^t)(x-x^t)$

$$= \text{-1}+[-1 \quad -2]\begin{bmatrix} x_1-2 \\ \\ x_2-1 \end{bmatrix} = -1-(x_1-2)-2(x_2-1)$$ If put all these to gather

$$\begin{cases} \min & f(\bar{x},x^o) = 5+4(x_1-2)+2(x_2-1) \\ \text{subject to} & g(\bar{x},x^o) = 3+4(x_1-2)+(x_2-1) \\ & h(\bar{x},x^o) = -1-(x_1-2)-2(x_2-1) \\ & 0.5 \le x_1 \le 2.5 \quad 0 \le x_2 \le 3 \end{cases} \tag{3.11}$$

the solution to this LPP is $x^1 = (\frac{11}{9}, \frac{8}{9})$

$g(x^1) = 0.6049 > 0, h(x^1) = -0.0123 \neq 0$, therefore reinitialize the problem at x^1 continue

until, get $x^* = (1, 1)$ as optimal solution.

3.3.1 Quadratic Programming Problems

consider the following general NLP

$$\begin{cases} \min & f(x) \\ \text{subject to} & g_i(x) \geq 0 \quad j = 1, \ldots, p \\ & h_k(x) = 0 \quad k = 1, \ldots, m \\ & x \in R \end{cases} \tag{3.12}$$

here $f(x), g(x)$ and $h(x)$ all are in general and non-linear functions. The quadratic programming problem here, the objective function $f(x)$ is quadratic function, the constraint $g(x), h(x)$ are liner functions. Quadratic programming problem has a special form where the objective function is non-linear, but to the extent that it is a quadratic function and similar to linear programming problem all the constraints are liner functions. Perhaps, can intuitively thing at this stage that it may be possible to solve a quadratic programming problem by use of a linear programming problem the way have solve linear programming [12, 18].

Problem maybe it is possible by some modifications will also be able to solve the quadratic programming problem following the method of solution of linear programming problem, there some applications of quadratic programming problems

(a) Least square approximations and estimation

(b) Portfolio Optimization

(c) Signal and image processing, computer, vision, etc.

(d) Optimal control, linear model, predictive control,etc

(e) PDF-constrained optimization problems in CFD, shape optimization, etc

(f) Sequential Quadratic Programming (SQP) methods for NLP

Let us define a general (QP) problem

$$\begin{cases} \min_x & q(x) = c^\top x + \dfrac{1}{2} x^\top H x \\ \text{subject to} & A^\top x \leq b \\ & B^\top x = e \\ & x \geq 0 \\ & x = [x_1, x_2, \ldots, x_n]^\top \\ & c = [c_1, \ldots, c_n]^\top \\ & e = [e_1, \ldots, e_p]^\top \\ & d = [b_1, \ldots, b_m] \end{cases} \tag{3.13}$$

$H = n \times n$ Hessian matrix

$A = n \times m$ constant matrix

$B = n \times p$ constant matrix

Example 3.3.2.

$$\begin{cases} \min_x & f(x) = -6x_1 + 2{x_1}^2 - 2x_1x_2 + 2{x_2}^2 \\ \text{subject to} & x_1 + x_2 = 2 \\ & x_1, x_2 \geq 0 \end{cases}$$

Compare with the general form(see equation (3.13)) .

$$\begin{cases} \min_x & f(x) = \begin{bmatrix} -6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \text{subject to} & \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2 \\ & C = \begin{bmatrix} -6 & 0 \end{bmatrix} \\ & B = \begin{bmatrix} 1 & 1 \end{bmatrix} \\ & e = [2] \\ & H = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \end{cases}$$

KKT conditions for Quadratic Programming Problems(see equation (3.13)), introduce slack variables S to convert inequalities qualities constraint $A^\top x + S = b$, $S \geq 0$ and $S \in R^n$. Define Lagrangian :
$L = c^\top x + \frac{1}{2} x^\top H x + u^\top (A^\top x + S - b) + v^\top (B^\top x - e) - w^\top x$ note that the non-negativity constraint $x \geq 0$ has been written as $-x \leq 0$

$$\textbf{Gradient Condition} \begin{cases} \dfrac{\partial L}{\partial x} = c^\top + H x^\top + A u^\top + B v^\top - w^\top = 0 \end{cases} \tag{3.14}$$

$$\textbf{Feasibility Check} \begin{cases} A^\top x + S - b = 0 \\ B^\top x - e = 0 \\ x \geq 0 \end{cases} \tag{3.15}$$

Indeed this linear problem, solve these conditions for x, u, v, S and w, since(see equation (3.15))and(see equation (3.14))linear, so can write them into matrix notions

complimentary Slackness:

$$\begin{cases} u_i s_i = 0, & i = 1, \ldots, m; s_i, u_i \geq, i = 1, \ldots, m \\ w_i x_i = 0, & i = 1, \ldots, n; w_i \geq, i = 1, \ldots, n \end{cases} \tag{3.16}$$

There are n number of inequality constraint, for each of them need slack variables, these leads to $u_i s_i = 0$ and n decision variables needs $w_i x_i = 0, i = 1, \ldots, n$, note that $s_i u_i \geq 0$ and $w_i \geq 0$. Because the slack variables and Lagrangian multipliers for the inequality constraints are non-negative, can solve these conditions for x, u, v, s and w that will give the KKT point and that maybe a candidate for optimal point. Lagrangian multiplier for equality constraint is free in sign. Write it as $v = y - z$ with $y, z \geq 0$, can define the following matrix and vectors:

$$N = \begin{bmatrix} H & A & -I_{(n)} & O_{(n\times m)} & B & -B \\ A^\top & O_{(n\times m)} & O_{(n\times m)} & I_{(n)} & O_{(m\times n)} & O_{(m\times n)} \\ B^\top & O_{(p\times m)} & O_{(p\times n)} & O_{(p\times m)} & O_{(p\times p)} & O_{(p\times p)} \end{bmatrix}_{(n+m+p)\times(2n+2m+2p)}$$

Now the KKT conditions can be written as : $NX = D$ complimentary slackness conditions reduces to

$$\begin{cases} X_i X_{n+m+i} = 0, & i = 1, \ldots, (n+m) \\ X_i \geq 0, & i = 1, \ldots, (2n+2m+2p) \end{cases} \tag{3.17}$$

$$\text{where } X = \begin{bmatrix} x \\ u \\ w \\ S \\ y \\ z \end{bmatrix}_{(2n+2w+2p)} , D = \begin{bmatrix} -c \\ b \\ e \end{bmatrix}_{(n+m+p)}$$

Except the complimentary slackness condition, all the equation in the transformed KKT condition are linear. A solution of the linear equation in $NX = D$ that satisfy the complimentary slackness condition and non-negativity constraint. The complimentary slackness conditions is non-linear in variable $X_i, i = 1, \ldots, (n + m)$. Thus, the simplex procedure need to be changed to accommodate this, A procedures developed by $wolfe(1959)$ and later refined by $Hadley(1964)$ can be used and the simplex method will converge in a finite number of steps [55]. Provided H is a positive definite matrix. This method is based on the phase I of the two-phase simplex method.

3.3.2 Solution of NLP by Quadratic Approximation

Similar to sequential linear programming strategy can solve a general NLP by successively approximating it as a QP problem a round the current estimate of the solution(x^o), note that a function $f(x)$ can be approximated by it is quadratic approximation around x^o as follows :

$$q(x; x^o) = f(x^o) + \nabla f(x^o)^\top (x - x^o) + \frac{1}{2}(x - x^o)^\top \nabla^2 f(x^o)(x - x^o)$$

Algorithm 1 : Solution of NLP by Successive Quadratic Approximation Algorithm

(a) Formulate the QP problem

$$\begin{cases} \min \quad \nabla f(x^{(t)})d + \frac{1}{2}d^{\top}\nabla^2 f(x^{(t)})d \\ \text{subject to} \\ g_j(x^{(t)}) + \nabla g_j(x^{(t)})d^{\top} \geq 0, j = 1, \ldots, J \\ h_k(x^{(t)}) + \nabla h_k(x^{(t)})d^{\top} \geq 0, k = 1, \ldots, K \\ x_i \geq 0, i = 1, \ldots, N \end{cases} \tag{3.18}$$

the objective function is the quadratic function and notes that the constraints have been converted to linear functions by performing. Taylor series expansion this have learned during successive linear programming [25, 58].

(b) Solve the QP problem and set : $x^{t+1} = x^t + d$.

(c) Check for convergence. If converged stop, else go to step a.

Example 3.3.3. Solve by successive QP strategy

$$\begin{cases} \min & f(x) = 6\frac{x_1}{x_2} + \frac{x_2}{{x_1}^2} \\ \text{subject to} & g(x) = x_1 + x_2 - 1 \geq 0 \\ & h(x) = x_1 x_2 - 2 = 0 \end{cases} \tag{3.19}$$

initial feasible estimate : $x^o = (2, 0)$, $f(x^o) = 12.25$,$h(x^o) = 0$, $g(x^o) = 2 > 0$

$$\nabla f(x) = \begin{bmatrix} 6x_2^{-1} - 2x_2{x_1}^{-3} \\ \\ -6x_1{x_2}^{-2} + {x_1}^{-2} \end{bmatrix}$$

$$\nabla^2 f(x) = \begin{bmatrix} 6x_2{x_1}^{-4} & -6{x_2}^{-2} - 2{x_1}^{-3} \\ -6{x_2}^{-2} - 2{x_1}^{-3} & 12x_1{x_2}^{-3} \end{bmatrix}, \nabla h(x) = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$$

the inequality constraint already linear, the first QP sub-problem will be :

$$\begin{cases} \min & f(x)(\frac{23}{4}, -\frac{47}{4})d + \frac{2}{2}d^\top \begin{bmatrix} \frac{3}{8} & -\frac{25}{4} \\ \frac{-25}{4} & 24 \end{bmatrix} \\ \text{subject to} & (1,1)d + 2 \geq 0 \\ & (1,2)d = 0 \end{cases}$$

the second constraints can be use to write $d_1 = -2d_2$ now the solution of single variable problem can be obtained as $d^o = (-0.9207, 0.4604)$, the new point become :

$$\begin{cases} x^1 = x^o + d^o = (1.0793, 1.4604) \\ f(x^1) = 5.687 \\ h(x^1) = -0.424 \\ g(x^1) > 0 \end{cases}$$

The objective function has improved. However equality constraint is violated, can continue the procedure and find the optimum with high accuracy. True optimum is $x^* = (1, 2)$, $f(x^*) = 5$, already not very in just one iteration.

3.3.3 Penalty Function Method

The penalty function method and augmented Lagrangian multiplier method are called "Transformation Methods " as these methods transform the original constrained problem to a sequence of unconstrained problems. The motivation for the transformation method is that can solve the constrained optimization problem using algorithms for unconstrained

problems which can be solved efficiently and reliably. In penalty function method the constrained problem is transformed into a sequence of unconstrained problems by adding penalty terms for each constraint violation, these methods are classified depending on how they handle inequality constraints, all transformation methods treat equality constraint in asimilar way.

Let $x^{(t)}, t = 1, 2, 3, \ldots$ approximate the stationary points of the sequence of associated unconstrained problems(stated that a general non-linear programming problem will be converted to a sequence of unconstrained problem user penalty function approach) [18, 35, 60].

There are two types of penalty function method.

- **Interior point methods** :

  The sequence x^t contains feasible point such method is also called **Barrier**

- **Exterior point methods**:The sequence x^t contains infeasible points.

Consider the constrained optimization problem$(P) : \min_{x \in S} f(x)$ where $f : R^n \to R$ is continuous and S is a constraint set in R^n. Let us define the penalty program q and let (P_{unc}) is the unconstrained problem. $q : \min\limits_{x \in R^n} f(x) + cp(x)$ where $c > 0$, $p : R^n \to R$ is the penalty function where

$$p(x) = \begin{cases} 0 & \text{if} \quad x \in S \\ +\infty & \text{if} \quad x \notin S \end{cases} \tag{3.20}$$

when $c \to \infty$, the solution of the penalty problem q will converge to a solution of the constrained problem P.

Lemma 3.3.1. [36] If $x^k = argmin_x q(x, c^n)$ and $c^{k+1} > c^k$ then

(a) $q(c_k, x_k) \leq q(c_{k+`}, x_{k+1})$

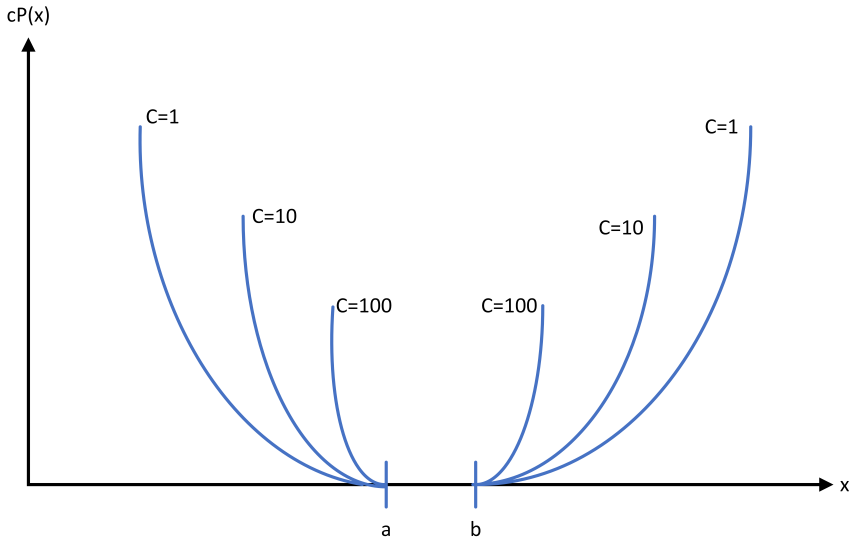


Figure 3.6: Quadratic Penalty Method

proof

$$\begin{aligned} q(c^{k+1}, x^{k+1}) &= f(x^{k+1}) + c^{k+1}p(x^{k+1}) \\ &\geq f(x^{k+1}) + c^{k}p(x^{k+1}) \quad (\because c^{k+1} > c^{k} > 0) \\ &\geq f(x^{k}) + c^{k}p(x^{k+1}) \quad (\because x^{k}\text{is the minimizer of} \quad q(c^{k}, x)) \\ \therefore q(c^{k+1}, x^{k+1}) &\geq q(c^{k}, x^{k}) \quad (\because q(c, x^{k}) = f(x^{k}) + c^{k}p(x^{k+1}) \end{aligned}$$

(b) $p(x^{k}) \geq p(x^{k+1})$

proof:

$$\begin{aligned} f(x^{k}) + c^{k}p(x^{k}) &\leq f(x^{k+1}) + c^{k}p(x^{k+1}) (\because x^{k} \text{ is the minimizer of} \quad q(c^{k}, x)) \\ f(x^{k+1}) + c^{k+1}p(x^{k+1}) &\leq f(x^{k+1}) + c^{k+1}p(x^{k}) (\because x^{k+1} \text{ is the minimizer of} \quad q(c^{k+1}, x)) \\ c^{k}p(x^{k}) + c^{k}p(x^{k+1}) &\leq c^{k}p(x^{k+1}) + c^{k+1}p(x^{k}) (\text{by adding together above equation}) \\ \Rightarrow (c^{k+1} - c^{k})p(x^{k+1}) &\leq (c^{k+1} - c^{k})p(x) \\ \therefore p(x^{k+1}) &\leq p(x^{k}) (\because c^{k+1} > c^{k} \\ \Rightarrow c^{k+1} - c^{k} &> 0) \end{aligned}$$

(c) $f(x^k) \leq f(x^{k+1})$

proof:

$$\begin{aligned} f(x^{k+1}) + c^k p(x^{k+1}) &\geq f(x^k) + c^k p(x^k) \quad (\because x^k \text{is the minimizer of } q(c^k, x)) \\ &\geq f(x^k) + c^k p(x^{k+1}) \quad (\because p(x^k) \geq p(x^{k+1})) \\ \therefore f(x^{k+1}) &\geq f(x^k) \end{aligned}$$

(d) $f(x^*) \geq q(c^K, x^k) \geq f(x^k)$

proof:

Let x^* be the optimal value of our original constrained problem (P) with constraint set S.

$$\begin{aligned} f(x^*) &= f(x^*) + c^k p(x^*) \quad (\because x^* \in S \Rightarrow p(x^*) = 0) \\ &\geq f(x^k) + c^k p(x^k) \geq f(x^k) \quad (\because x^k \text{is the minimizer of } q(c^k, x), \text{and} c^k > 0, p(x^k) \geq 0) \\ f(x^*) &\geq q(c^{k+1}, x^{k+1}) \geq f(x^k) \quad \forall k \end{aligned}$$

Penalty for Problem with Mixed Constraints [69]

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in \mathtt{R^n} \end{cases} \tag{3.21}$$

where penalty function

$$p(x) = \begin{cases} 0 & \text{if} \quad g(x) \leq 0 \quad \text{and} \quad h(x) = 0 (\text{x is interior piont}) \\ > 0 & \text{if} \quad g(x) > 0 \quad \text{or} \quad h(x) \neq 0 (\text{x is exteral piont}) \end{cases} \tag{3.22}$$

Then

$$p(x) = \sum_{i=1}^{m} [\max\{0, g_i(x)\}]^q + \sum_{k}^{i=1} |h_i(x)|^q, q \geq 1$$

Algorithm 1 : Penalty Function Method (to solve NLP)

(a) Input: $\{c^k\}, \epsilon, \quad k = 0 \ldots \infty$

(b) Set $k := 0$, initialize x^k

(c) While $(q(x^k, c^k) - f(x^k)) > \epsilon$

i. $x^{k+1} = \text{argmin}_x q(x, c^k)$

ii. $k := k + 1$

end while Output: $x^* = x^k$

output $x^* = x^k$ a stationary of $f(x)$

Different Penalty Terms

Different penalty terms are used for equality and inequality constraints.

(a) Parabolic penalty for equality constraints $p = c\{h(x)\}^2$, the feasible points always satisfy $h(x) = 0$. Any infeasible point is penalized by an amount proportionate to the square of constraint violation, the extent of penalty is controlled by penalty parameter c. Initially, a small value of c is used and then it is increased gradually, since all infeasible points are penalized, this is an exterior penalty term .

(b) Infinite Barrier Penalty for inequality constraint $R = c \sum_{j \in J} |g_j(x)|$

Here, the parameter c is large number (say, 10^{20}), $\bar{J}$ represents the set of violated constraints at the current point, $g_j(x) > 0$ for all $j \in \bar{J}$

Thus a penalty proportionate to the constraint violation is added to the objective function, since all infeasible points are penalized, this is an exterior penalty term.

(c) *Log* penalty, for inequality constrains; $p = c \ln[-g(x)]$, for infeasible points $g(x) > 0$. Thus, this penalty cannot assign penalty to infeasible points. More penalty is added to the points that are close to the constraint boundary (points with very small $g(x)$). Here we start with large c and then gradually reduce the value of c. Since all feasible points are penalize, this an interior penalty term.

(d) Bracket Operator: Handles both constraints $p = c < g(x) >^2$ where, $< g(x) >= \max(0, g(x))$, this is mostly used to handle inequality constraints. here we start with a small value of c and then gradually increase the value of c. Since all infeasible points are penalized, this is an exterior penalty term.

(e) Inverse Penalty for inequality constraints $p = -c\frac{1}{g(x)}$, here we start with a large value of c and then gradually reduce the value of c, since all feasible points are penalized, this is an interior penalty term.

Advantages and Disadvantage

Advantages:

(a) It can be started from an infeasible point for exterior penalty function method.

(b) Unconstrained optimization method can be directly used (there a host of powerful robust unconstrained optimization methods which can be used directly to solve the non-linear programming problem). Conceptually the penalty function approach is extremely simple in terms of implementation it is also simple and since the unconstrained optimization techniques can be used it has advantages.

Disadvantage:

There are some disadvantage also the function becomes yield condition

(a) The function become ill-conditioned as the value of the penalty terms is increased. The gradient value may become large and algorithm may show divergence.

(b) This method does not satisfy the constraints exactly, it is not suitable for optimization problems where feasibility must be ensured in all iteration.

Example 3.3.4. solve the NLP by using the exterior penalty method

$$\begin{cases} \min_x & \mathtt{f(x)} = 4{x_1}^2 + 2{x_2}^2 \\ \text{subject to} & g_1(x) = 3x_1 + x_2 - 8 = 0 \\ & g_2 = 15 - 2x_1 - 4x_2 \geq 0 \\ & x \in \mathtt{R^n} \end{cases} \tag{3.23}$$

Optimal lies at $x^* = (1.7, 2.9)$(see figure 3.1) and $f(x^*) = 28.38$ where the KKT multiple $v = -3.12; \mu = -2.12$. Let us denote the function

$$\psi(x_1, x_2) = (3x_1 + x_2 - 8)^2 + max\left\{0, -2x_1 - 4x_2 + 15\right\}^2$$

viol1= $|3x_1 + x_2 - 8|$

viol2= $\max\{0, -2x_1 - 4x_2 + 15\}$

violation= $\max\{$viol1,viol2$\}$ our goal find $\min\limits_{x_1,x_2} F(x_1, x_2) = 4{x_1}^2 + 2{x_2}^2 + \mu(3x_1 + x_2 - 8)^2 + \mu \max\{0, 15 - 2x_1 - 4x_2\}^2$

k	μ_k	x_k	$f(x_k)$	$Violation$
1	10	(1.65144317 2.89733285)	27.698133521103465	0.14833762589048582
2	100	(1.6948548 2.89991757)	28.309175040725997	0.015518020201623628
3	1000	(1.69948235 2.89999377)	28.372888715085388	0.001559187734709866
4	10000	(1.6999482 2.89999939)	28.37928849030861	0.00015600515412739213
5	100000	(1.69999482 2.89999993)	28.37992875546405	1.5613819851267863e-05
6	1000000	(1.69999948 2.89999999)	28.37999278373912	1.573410138178133e-06
7	10000000	(1.69999995 2.89999999)	28.3799991866957	1.6941103631040733e-07

Table 3.1: The penalty function method, the finally point is,(1.699999952.89999999) which is closed to the exact optimal solution.

Example 3.3.5. consider the non-linear programming

$$\begin{cases} \min_x & f(x) = (x_1 - 1)^2 + (x_2 - 2)^2 \\ \text{subject to} & 2x_1 - x_2 \leq 0 \\ & x_1 - 5 \\ & x_1, x_2 \geq 0 \end{cases}$$

It will be solved by using the Interior point methods, this means, the sequence x^t contains feasible point such method is also called **Barrier Methods** [10], looking for minimizing the function $q(x, c^k) = (x_1 - 1)^2 + (x_2 - 2)^2 - c^k \ln(2x_1 - x_2)^2 - c^k \ln(x_1 - 5)$. The first derivative of (f) with respect to (x_1) looks as follows: $[\frac{\partial q}{\partial x} = 2(x_1 - 1) - c^k \cdot \frac{2c^k}{2x_1 - x_2} - c^k \cdot \frac{c^k}{x_1 - 5}] = 0$ and the first derivative of (q) with respect to (x_2) looks as follows: $[\frac{\partial q}{\partial x} = 2(x_2 - 2) + c^k \cdot \frac{c^k}{2x_1 - x_2} = 0.$

Start in the feasible region of the problems. Note that as $(c^k \to 0)$ with $(k \to \infty)$, the solution converges to (x^*). That is, the solution to the original optimization problem! This method can readily be converted into a numerical algorithm that uses an unconstrained optimization method. Pick a number(c) such that $(0 < c < 1)$. Starting from $(k = 1)$,

minimize $(q(x, c^k))$ using any unconstrained optimization method. Finally, feed in the results of the minimization as the new starting point to the minimisation method, and increment (k)

Iterations	X coordinate	Y coordinate	Objective value
1	0.5	3	87.9166666666667
2	-1.8539239813419	4.01515489440643	45.251947655381
3	-2.292850433984	3.07854129204232	44.1177121737031
4	-2.24890491417822	3.0509818637007	44.1107544487288
5	-2.2477807046726	3.05270801468533	44.1107497308982
6	-2.24795915595488	3.05291437076109	44.1107495579245

Table 3.2: The interior penalty function method

Initial Objective Function Value: 8.791667e+01

Number of Iterations for Convergence: 6

Point of Minima: [-2.247959e+00,3.052914e+00]

Objective Function Minimum Value: 4.411075e+01

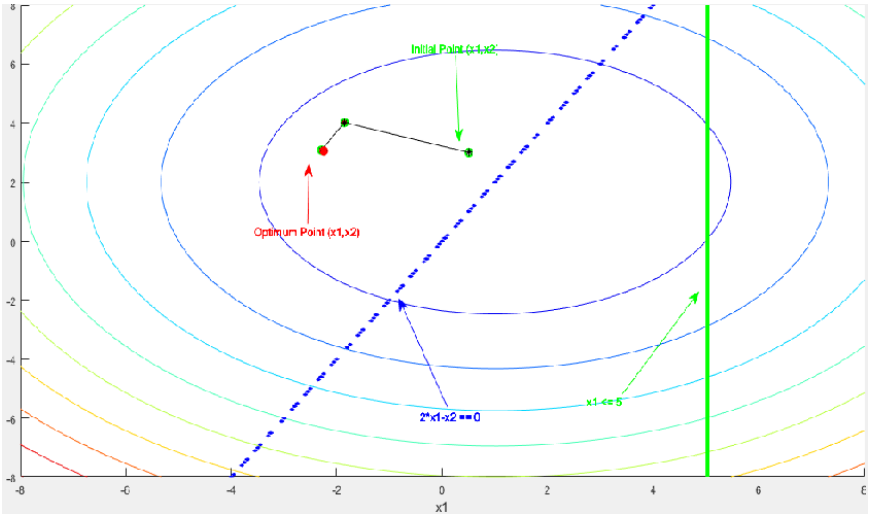


Figure 3.7: The interior penalty function method

Finally

Consider the problem

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & e(x) = 0 \end{cases} \tag{3.24}$$

and the first order necessary condition is that there exists (x^*, μ^*) be a KKT point $(\nabla f(x^*) + \mu^* \nabla e(x^*) = 0)$ and the penalty function $q(x, c) = f(x) + cp(x)$ the optimal solution depend c go to infinity as $c \to \infty, q(x, c) = f(x)$(as c go to infinity there can be some numerical difficulties associated with this optimization problem).

Consider the perturbed problem

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & e(x) = \theta \end{cases} \tag{3.25}$$

Let us consider the Penalty function

$$\begin{aligned} q(\hat{x}, c) &= f(x) + c(e(x) - \theta)^2 \\ &= f(x) - 2c\theta e(x) + ce(x)^2 \quad \text{(ignoring constant term} = \theta^2) \\ &= f(x) + \mu e(x) + ce(x)^2 \quad [\mu = -2c\theta] \\ &= L(x, \mu) + ce(x)^2 \\ &= \hat{L}(x, \mu, c) \end{aligned}$$

$[L(x, \mu) = f(x) + \mu e(x) \quad \text{Lagrange function uasing } x, \mu \quad \text{as the variables}]$

To that Lagrangian function there is a extra quantity which is added, therefore. It is called "Augmented Lagrangian function". The important observation, the $\nabla_x \hat{L}(x^*, \mu^*, c) = 0, \quad \forall c$, this gradient vanishes irrespective of the value of c.

Let x^* be a local minimal then the Lagrange at (x^*, μ^*), denotes

$$\begin{aligned}\nabla_x \hat{q}(x^*, c) &= \nabla_x \hat{L}(x^*, \mu^*, c) \\ &= \nabla_x L(x^*, \mu^*) + 2ce(x^*)\nabla e(x^*)\end{aligned}$$

since x^* local minimal then $\nabla_x L(x^*, \mu^*) = 0$ and $e(x^*) = 0$ because x^* feasible point
How to get an estimate of μ^*?. Let ${x_c}^*$ be a minimizer of $L(x, \mu, c)$, Therefore

$$\begin{aligned}\nabla_x L({x_c}^*, \mu, c) &= \nabla f({x_c}^*) + \mu \nabla e({x_c}^*) + ce({x_c}^*)\nabla e({x_c}^*) = 0 \\ \therefore \nabla f({x_c}^*) &= -\mu \nabla e({x_c}^*) - ce({x_c}^*)\nabla e({x_c}^*) = 0 \\ &= \nabla e({x_c}^*)(-\mu - ce({x_c}^*))\end{aligned}$$

where $-\mu - ce({x_c}^*)$ estimated of μ^* the idea is that every iteration we get an estimate of μ^* and repeat the procedure.

3.3.4 Augmented Lagrange Multiplier Method

Augmented Lagrange method attempts to alleviate some of the problems of the penalty function method. In this method, there is no need for the penalty parameter R to go to infinity. The augmented Lagrange multiplier method combines both Lagrange multipliers and penalty function methods. Let us first consider an optimization problem with only equality. [26]
Constraints the problem

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & h_k(x) = 0 \quad \text{for} \quad k = 1, \ldots, m \\ & x \in \mathtt{R^n} \end{cases} \tag{3.26}$$

The Lagrangian function for the equality constrained problem is.

$$L(x, \lambda) = f(x) + \sum_{k=1}^{m} \lambda_k h_k(x)$$

where λ_k , $k = 1, \ldots, m$ are Lagrange multipliers. The augmented Lagrangian function is defined by using exterior penalty function approach to define the new objective function.

$$AL(x, \lambda, R) = f(x) + \sum_{k=1}^{m} \lambda_k h_k(x) + R \sum_{k=1}^{m} \{h_k(x)\}^2 \tag{3.27}$$

Where R is the penalty parameter. Note that the function $AL(x, \lambda, R)$, reduces to the Lagrangian if $R = 0$ and to the classical penalty function if all $\lambda_j = 0$. The solution to this $AL(x, \lambda, R)$ will lead to the solution of the original constrained non-linear programming problem. If the Lagrange multipliers are fixed at their optimum values ${\lambda^*}_j$, the minimization of $AL(x, \lambda, R)$ gives the solution of the original constrained problem in one step for any value of R, Then there is no need to minimize the function $AL(x, \lambda, R)$ for an increasing sequence of values of R, since the values of ${\lambda^*}_j$ are not known in advance, an iterative scheme is used to find the solution of the problem. In the first iteration ($k = 1$), the values of ${\lambda^*}_j$ are chosen as zero, the value of R is set equal to an arbitrary constant, and the function $AL(x, \lambda, R)$ is minimize with respect to x to find $x^{*(k)}$, the values of ${\lambda_j}^k$ and R are then updated to start next iteration. can update rules for $\lambda_j(\lambda_j$ which is the Lagrange multipliers) .

$${\lambda_j}^{k+1} = {\lambda_j}^k + 2R_k h_j(x^k), \qquad j = 1, \ldots, m$$

where k denotes the steps or the iterations and x^k is the starting vector used in the minimization of $AL(x, \lambda, R)$. The value of R_k is updated as

$$R_{k+1} = CR_k \qquad \text{with} \quad c > 1$$

.

Algorithm 1 : Augmented Lagrangian Method

(a) Input: c, ϵ

(b) Set $k := 0$, initialize x^k, μ^k

(c) While $(\hat{L}(x^k, \mu^k, c) - f(x^k)) > \epsilon$

i. $x^{k+1} = \text{argmin}_x \hat{L}(x^K, \mu^k, c)$

ii. $\mu^{k+1} = \mu^k + ce(x^k)$

iii. $k := k + 1$ end while Output: $x^* = x^k$

output $x^* = x^k$ a stationary of $f(x)$

Augmented Lagrangian Multiplier Method for Inequality constraints

Constraints the problem

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & g_j(x) \leq 0 \quad \text{for} \quad j = 1, \ldots, p \\ & x \in \mathtt{R^n} \end{cases} \tag{3.28}$$

The Lagrangian function

$$L(x, \lambda) = f(x) + \sum_{j=1}^{p} \lambda_j \left[g_j(x) + {\sigma_j}^2 \right]^2$$

where ${\sigma_j}^2$ is the slack variables [74]. The augmented Lagrangian function denotes by

$$L(x, \lambda) = f(x) + \sum_{j=1}^{p} \lambda_j \left[g_j(x) + {\sigma_j}^2 \right]^2 + \sum_{j=1}^{p} R \left[g_i(x) + {\sigma_j}^2 \right]^2$$

The above form is equivalent to :

$$L(x,\lambda) = f(x) + \sum_{j=1}^{p} \lambda_j \alpha_j + R \sum_{j=i}^{p} {\alpha_j}^2$$

where , $\alpha_j = \max\left\{ g_j(x, -\frac{\lambda_j}{2R_k}) \right\}$ update rules for λ_j:

$${\lambda_j}^{k+1} = {\lambda_j}^k + 2R_k h_j(x^k), \ldots, j = 1, \ldots, p$$

Augmented Lagrangian Multiplier Method for Mix Constraints

Consider the problem

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & g_j(x) \le 0 \quad \text{for} \quad j = 1, \ldots, p \\ & h_k(x) = 0 \quad \text{for} \quad k = 1, \ldots, m \\ & x \in \mathtt{R^n} \end{cases} \tag{3.29}$$

Augmented Lagrangian function
$L(x,\lambda) = f(x) + \sum_{j=i}^{p} \lambda_j \alpha_j + \sum_{k=1}^{m} \lambda_{p+k} h_k(x) + R \sum_{j=i}^{p} {\alpha_j}^2 + R \sum_{k=1}^{m} \{h_k(x)\}^2$
where $\alpha_j = \max\left\{ g_j(x), -\frac{\lambda_j}{2R_k} \right\}$
update rules for λ_j:

$${\lambda_j}^{k+1} = {\lambda_j}^k + 2R_k \max\left\{ g_j(x), -\frac{{\lambda_j}^k}{2R_k} \right\}, \qquad j = 1, \ldots, p$$

$${\lambda_j}^{k+1} = {\lambda_j}^k + 2R_k h_j(x^k), \qquad j = 1, \ldots, m$$

There are some advantages if using the augmented Lagrangian multiplier method.

(a) The value of penalty parameter R need be increased to infinity for convergence.

(b) The starting vector x^1 dose not need be feasible.

(c) It is possible to achieve $g(x) = 0$ and $h(x) = 0$ precisely and non-zero values of Lagrange multipliers for the active constraints.

As the penalty method the Augmented Lagrange Method has Barrier method.
Typically applicable to inequality constrained problems

$$\begin{cases} \min_x & \mathtt{f(x)} \\ \text{subject to} & h_j \leq 0 \qquad j = 1, \ldots, l \end{cases} \tag{3.30}$$

Let $X = \{x : h_j \leq 0, j = 1, \ldots, l\}$, there is some Barrier functions defined on the interior of x

$$B(x) = -\sum_{j=1}^{1} \frac{1}{h_j(x)} \quad \text{or} \quad B(x) = -\sum_{j=1}^{1} \log(-h_j(x))$$

Approximate problem using Barrier function (for $c > 0$)

$$\begin{cases} \min_x & \mathtt{f(x)} + \dfrac{1}{c} B(x) \\ \text{subject to} & x \in \text{Interior of } X \end{cases} \tag{3.31}$$

Example 3.3.6. Solve the non-linear programming problem to find the optimal solution by use augmented Lagrangian method.

$$\begin{cases} \min_x & \mathtt{f(x)} = 4{x_1}^2 + 2{x_2}^2 \\ \text{subject to} & g_1(x) = 3x_1 + x_2 - 8 = 0 \\ & g_2 = 15 - 2x_1 - 4x_2 \geq 0 \\ & x \in \mathtt{R^n} \end{cases} \tag{3.32}$$

Optimal lies at $x^* = (1.7, 2.9)$ and $f(x^*) = 28.38$ where the KKT multiple $v = -3.12; \mu = -2.12$(see figure 3.1). Let us denote the function.
$\psi(x_1, x_2) = (3x_1 + x_2 - 8)^2 + max\left\{0, -2x_1 - 4x_2 + 15 + \dfrac{u}{2\mu}\right\}^2$
$- \dfrac{u^2}{4\mu^2} + \dfrac{1}{\mu} v(3x_1 + x_2 - 8) + \mu(3x_1 + x_2 - 8)$ Our goal is to find $\min_{x_1,x_2} \mathtt{F(x)} = 4{x_1}^2 +$

$$2{x_2}^2 + \mu \max\left\{0, 15 - 2x_1 - 4x_2 + \frac{u}{2\mu}\right\}^2 -$$
$$\frac{u^2}{4\mu} + v(3x_1 + x_2 - 8) + \mu^2(3x_1 + x_2 - 8)$$

k	μ_k	x_k	u	v	$f(x_k)$	$Violation$
1	10	1.65144, 2.89733	0	0	27.6981335211	0.148337625890
2	10	1.69700, 2.9018	2.1556450431	-2.96675251780	28.36115250	0.00710729066560
3	10	1.69977,2.90016	2.12466212840	-3.1088983311	28.3788558744	0.00051222019428
4	10	1.69998, 2.90001	2.1203815517	-3.11914273500	28.3799138726	3.953792841571157
5	10	1.69999, 2.90000	2.1200303307	-3.1199334935	28.379993346	3.09008350551920
6	10	1.6999, 2.90000	2.1200021437	-3.1199952952	28.3799994970	2.33029684437724

Table 3.3: The augmented Lagrangian method

The finally point is $(1.69999995, 2.90000)$ corresponding to

$$u = 2.1200021437, v = -3.1199952952$$

and $f(1.69999995, 2.90000) = 28.3799994970$ which is closed to the exact optimal solution $= 28.38$.

CHAPTER 4

NEW APPROACH LINE SEARCH TECHNIQUE

4.1 Introduction

This chapter is to find a better line search technique and different combinations of optimization and line search algorithms [27]. Line search technique is on the off chance that it looks for the base of a descent direction vector, when processed iteratively with a sensible advance size [47]. The solution methods for unconstrained optimization problems can be broadly classified into gradient-based and non-gradient based search methods. As the name suggests, gradient-based methods require gradient information in determining the search direction. The gradient-based methods discussed in this chapter are steepest descent.

$$Davidon–Fletcher–Powell(DFP), Broyden–Fletcher–Goldfarb–Shanno(BFGS)$$

Newton [33]. The search direction computed by these methods uses the gradient information, Hessian information, or a combination of these two. Some methods also make an approximation of the Hessian matrix. Once the search direction is identified, one needs to evaluate how much to move in that direction so as to minimize.

4.2 Unconstrained Problem

Consider the unconstrained minimization problem, let f be real value function, assume f is bounded below.

$$\begin{cases} \min & \mathtt{f(x)} \\ \text{subject to} & x \in \mathtt{R^n} \end{cases} \tag{4.1}$$

Definition 4.2.1. Let $x \in R^n$ if $\exists d \in R^n$ and $\delta > 0$ such that $f(x + \alpha d) < f(x) \forall \alpha \in (0, \delta)$ then the direction d is said to be descent direction of f at x.
the set of descent direction denote by$\left\{d \in R^n : g^k d^\top < 0\right\}$ where $f \in C^1$ and $x \in R^n$,$g^k = g(x) = \nabla f(x)$. In general if the gradient of f at x made obtuse angle with d then, can say d descent direction [80].

An iterative optimization algorithm generates a sequence $\left\{x^k\right\}_k \geq 0$, which converges to local minimum.

Algorithm 1 : Unconstrained minimization algorithm

(a) Initialize x^o set $k := 0$

(b) while stopping condition is not satisfied at x^k

i. Find x^{k+1} such that $f(x^{k+1}) < f(x^k)$

ii. $k := k + 1$ end while

output $x^* = x^k$ a local min of $f(x)$

.

In the above algorithm, have a set of questions that will seek to answer in this chapter.

- How to find x^{k+1} such that $f(x^{k+1}) < f(x^k)$? there are many strategic methods to find x^{k+1} will discuss later.
- Which stopping condition can be used ?

- Does the algorithm convergence, under what conditions it will be converge, how fast does it converge ? speed of convergence means the number of iterations.
- Does the speed and the converge depend on the initial point?

Stopping Condition

The practical stopping conditions to determinant the stopped condition are [3]:

(a) $\left\|g(x^k)\right\| \leq \epsilon$ when ϵ smalls positive number define by user the converge point depend on the what ϵ chosen.

(b) $\left\|g(x^k)\right\| \leq \epsilon(1 + |f(x^k)|)$

(c) $\dfrac{f(x^k) - f(x^{k+1})}{|f(x^k)|}$

Speed of convergence

Assume that an optimization algorithm generates a sequence $\left\{x^k\right\}$ which converges to x^*, interesting to study how fast does the sequence convergence to x^*.

Definition 4.2.2. The sequence $\left\{x^k\right\}$ converges to x^* as $\lim\limits_{k\to\infty} \dfrac{\left\|x^{k+1} - x^*\right\|}{\left\|x^k - x^*\right\|^p} = \beta$, $\beta \leq \infty$, where p is the order of convergence and β is said to be converge rate, since the numerator and denominator of quantity positive then $\beta > 0$, as the $\left\|x^{k+1} - x^*\right\|$ represent the distance between x^{k+1} and x^* while $\left\|x^k - x^*\right\|^p$ represent the distance between x^k and x then seeking to find x^{k+1} closed to x^*, so asymptotically, can see

$$\left\|x^{k+1} - x^*\right\| = \beta \left\|x^k - x^*\right\|^p$$

the important point to note that if the value of p higher then the convergence is fast [33].

Example 4.2.1. Let $p = 1$, $0 < \beta < 1$ it is called linear convergence, for $\beta = 0.1$ it is more closed to zero and $\left\|x^0 - x^*\right\| = 0.1$

$$\left\|x^{k+1} - x^*\right\| = \beta \left\|x^k - x^*\right\|^p$$

can say $\left\|x^k - x^*\right\| : 10^{-1}, 10^{-2}, 10^{-3}, \ldots$

If, choose β more closed to one let us take $\beta = 0.9$

and $\left\|x^0 - x^*\right\| = 0.1$ $\left\|x^0 - x^*\right\| = 0.1, 0.09, 0.081, 0.0729 \ldots$ if are comparing between the result above, note that when β is closed to zero then the rate converges faster linearly convergence, when $p = 2$ and $\beta > 0$ this called the quadratic convergence.

Let us take $\beta = 1$, $\left\|x^0 - x^*\right\| = 0.1$ $\left\|x^k - x^*\right\| = 10^{-1}, 10^{-2}, 10^{-4}, 10^{-8} \ldots$ if, comparing between the result above note that the quadratic convergence is faster than linear convergence.

Definition 4.2.3. Suppose an algorithm generates a convergent sequence $\left\{x^k\right\}$ to x^* such that $\lim_{k\to\infty} \dfrac{\left\|x^{k+1} - x^*\right\|}{\left\|x^k - x^*\right\|} = 0$ and $\lim_{k\to\infty} \dfrac{\left\|x^{k+1} - x^*\right\|}{\left\|x^k - x^*\right\|^2} = \infty$ this convergence is called super linear convergence and it is between liner convergence and quadratic converge [33].

Example 4.2.2.

- The sequence with $x^k = 1 + a^k$ where $0 < a < 1$ converges to 1 linearly, with convergence rate $\beta = a$
- The sequence $x^k = a^{2^k}$ where $0 < a < 1$ converges to zero quadratically, with convergence rate, $\beta = 1$.
- The sequence $1 + k^{-k}$ converges super linearly to 1.

4.3 Unconstrained minimization

Algorithm 2 : Unconstrained Minimization Algorithm

(a) Initialize x^o set ϵ, $k := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. Find x^{k+1} such that $f(x^{k+1}) < f(x^k)$

ii. $k := k + 1$ end while

output $x^* = x^k$ a stationary of $f(x)$

.

The new point x^{k+1} shall satisfy $f(x^{k+1}) < f(x^k)$, there are two steps to find the x^{k+1}

- Find a descent direction d^k for f at x^k
- Take positive step length α^k along d^k such that $f(x^{k+1}) < f(x^k)$ where $x^{k+1} = x^{k+1} + \alpha^{k+1} d^k$

then by this new information, can rewrite the algorithm

Algorithm 3 : Unconstrained minimization algorithm

(a) Initialize x^o set $k := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. Find a descent direction d^k for f at x^k

ii. find $\alpha^k (> 0)$ along d^k such that $f(x^k + \alpha^k d^k) < f(x^k)$

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. $k := k + 1$ end while

output $x^* = x^k$ a stationary of $f(x)$

4.4 Step Length Determination

How to choose $\alpha^k > 0$ a long d^k such that $f(x^{k+1}) < f(x^k)$, there are two ways to choose α^k

(a) **Exact Line search** [80]

For d^k such that $g^k d^k < 0$ determine $\alpha^k > 0$ by solving the optimization problem

$$\alpha^k = \text{argmin}_{\alpha^k > 0} \phi(\alpha) = f(x^k + \alpha^k d^k)$$

at each iteration, need to solve the one dimensional optimization problem to minimize $\phi(\alpha)$ to find α^k, infact this computational expansive and in most of case, no need to minimize $\phi(\alpha)$ our the main problem is minimize $f(x)$, very import notion that the initial point is not crucial [15].

(b) **Inexact Line search** [80]

Formulate a criterion that assures the steps are neither too long nor too short, pick a good initial step size. Construct a sequence of updates that satisfy the above criterion after very few steps.

Example 4.4.1. Consider the problem $\min x^2$ since the function convex then the local and the global minimum is same and equal to zero($x^* = 0$), will use python code to find the minimize of this problem, let $x^k = (-1)^k(2^{-k})$ and $d^k = (-1)^k$, $k = 0, 1, 2 \ldots, \{x\} = \left\{1, -\frac{1}{2}, \frac{1}{4} \ldots\right\}, \{f\} = \left\{1, \frac{1}{4}, \frac{1}{8} \ldots\right\}$, at every new iteration $f(x^{k+1}) \leq f(x^k) \forall k = 0, 1, 2$ the sequence $\left\{x^k\right\}$ doesn't converge.

Code 12 :**The Global Minimum**

```
import math
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from scipy import optimize
def f(x):
return x**2
def grad(x):
return 2*x
def gd(x, grad, alpha, max_iter=10):
```

```
xs = np.zeros(1 + max_iter)
xs[0] = x
for i in range(max_iter):
x = x - alpha * grad(x)
xs[i+1] = x
return xs
def gd_momentum(x, grad, alpha, beta=0.9, max_iter=10):
xs = np.zeros(1 + max_iter)
xs[0] = x
v = 0
for i in range(max_iter):
v = beta*v + (1-beta)*grad(x)
vc = v/(1+beta**(i+1))
x = x - alpha * vc
xs[i+1] = x
return xs
alpha = 0.1
x0 = 1
xs = gd(x0, grad, alpha)
xp = np.linspace(-1.2, 1.2, 100)
plt.plot(xp, f(xp))
plt.plot(xs, f(xs), 'o-', c='red')
for i, (x, y) in enumerate(zip(xs, f(xs)), 1):
plt.text(x, y+0.2, i,
bbox=dict(facecolor='yellow', alpha=0.5), fontsize=14)
pass
plt.show()
alpha = 0.95
```

```
xs = gd(1, grad, alpha)
xp = np.linspace(-1.2, 1.2, 100)
plt.plot(xp, f(xp))
plt.plot(xs, f(xs), 'o-', c='red')
for i, (x, y) in enumerate(zip(xs, f(xs)), 1):
plt.text(x*1.2, y, i,
bbox=dict(facecolor='yellow', alpha=0.5), fontsize=14)
pass
plt.show()
```

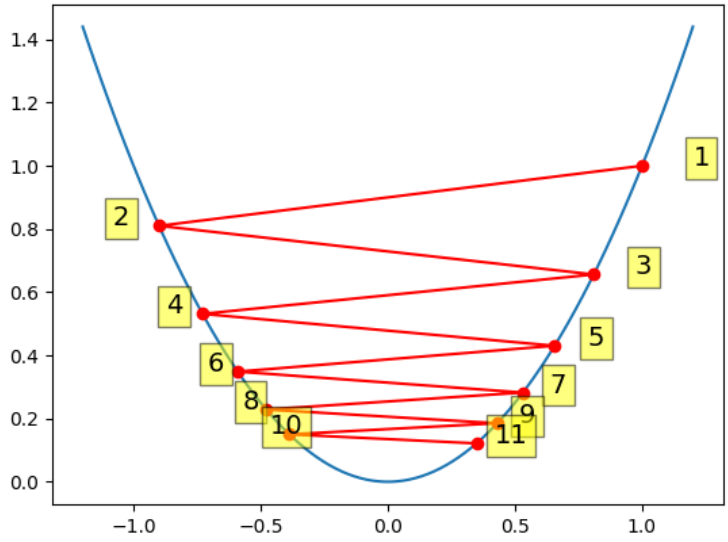


Figure 4.1: Small decrease in function values relative to step length and small step length The sequence x^k does not converge to minimize. Small decrease in function values relative to the step length the decreasing of the value functions very slow. If use same the problem in the above example with the different decent direction $d^k = -1$, at every iteration $f(x^{k+1}) \leq f(x^k) \forall k = 0, 1, 2 \ldots$, $\lim_{k \to \infty} x^k \neq x^*$, the sequence x^k doesn't converge to x^*

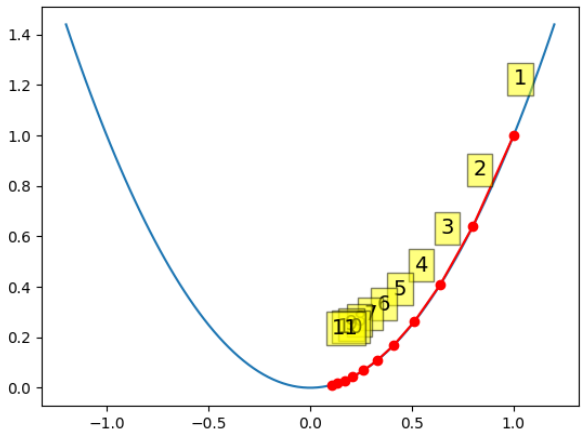


Figure 4.2: Note:Choose the initial point not curtail

Finally:

If doing exact line search, that it will be expansive computation. One thing interesting to find α^k such that the sequences generated by the iteration converge to minimize, so for that need to define some condition's to bypass.

4.5 Sufficient Decrease Condition

The insufficient reduction in f at each step causes, it fails to converge to the minimizer of the problem. To avoid this behaviour need a concept discuss next [37, 42].

(a) **Armijo's condition**

It ensures sufficient decrease in the function value define as $\phi_1(\alpha) = f(x^k)+c_1\alpha g^{k\top}d^k$, $c_1 \in (0, 1)$ choose α^k such that

$$f(x^k + \alpha^k d^k) \leq \phi_1(\alpha^k)$$

if look to right hand side of $\phi_1(\alpha)$, note ${g^k}^\top d^k$ is rate of decrease of f in the direction d^k, unfortunately this condition not make sure that steep length is small.

(b) **Goldenstein's condition**

It ensures that step lengths are not small, will choose α such that

$$\phi_2(\alpha) = f(x^k) + c_2\alpha g^{k\top}d^k, \quad c_2 \in (c_1, 1)$$

(c) **Armijo-Goldstein condition's**

If compare between the Armijo's condition and Goldenstein's condition will getting

$$\phi_2(\alpha^k) \leq f(x^k + \alpha^k d^k) \leq \phi_1(\alpha^k)$$

(d) **Wolfe's condition's**

It ensures sufficient rate of decrease of function value in the given direction choose α^k such that

$$\phi' \geq c_2\phi'(0), \quad c_2 \in (c_1, 1)$$

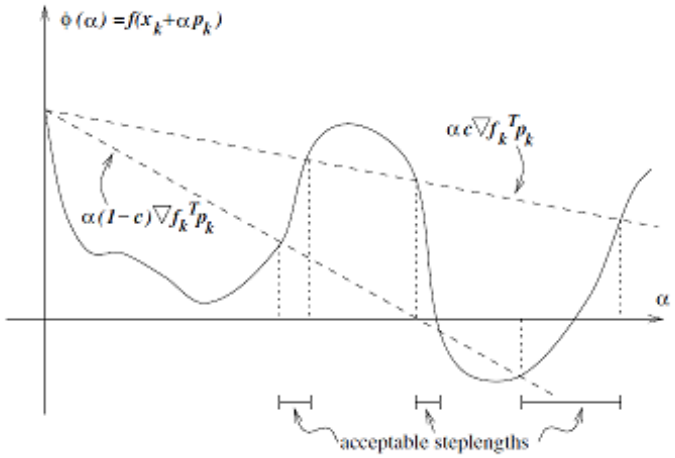


Figure 4.3: This figure shown the acceptable step lengths

Algorithm 4 : Unconstrained Minimization Algorithm

(a) Initialize x^o set $k := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. Find a descent direction d^k for f at x^k

ii. find $\alpha^k(> 0)$ along d^k such that

- $f(x^k + \alpha^k d^k) < f(x^k)$
- α^k satisfies Armijo-wolfe conditions

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. $k := k + 1$ end while

output $x^* = x^k$ a stationary of $f(x)$

.

Since neither chose any second order information so, there is no guaranty that the

stationary point will be the local minimize, the Armijo-wolfe condition or Armijo-golden condition is to take care of sufficient decrease as well as the sufficient step length. Now, if ensure that those conditions are satisfied, then the algorithm will converge to minimization.

Theorem 4.5.1. *Consider the problem* $\min_x f(x), \quad x \in R$ *let* $f \in C^1$ *and* f *be bounded below, assume any iteration* $x^{k+1} = x^k + \alpha^k d^k$, *where* d^k *is a decent direction chosen such that* $\cos^2\theta_k > \delta(> 0)$ *where* θ *is the angle between* d^k *and* g^k *and* α^k *satisfies the wolfe's conditions and Armijo's condition's [67], also assume* ∇f *is Lipschitz continuous then the optimization algorithm either terminates in a finite number of iterations or* $\left\|\nabla f(x^k)\right\| \cos^k\theta_k \to 0$

***poof**:*

The function volumes be $\left\{f^k\right\}$, $k \geq 0$, *in every iteration that the function decreasing* $f^{k+1} < f^k$, $k \geq 0$*(monotonically decreasing), let us consider stopping condition* $\left\|g^k\right\| < \epsilon$ *where* $\nabla f^k = g^k$ *since at every iteration* k *of the optimization algorithm, the direction* d^k *is chosen such that* ${g^k}^\top d^k < 0$, *also since define* $\phi(\alpha) = f(x^k + \alpha d^k)$, $\alpha d^k(> 0)$ *is chosen such that satisfies the wolfe's conditions and Armijo's condition's*

$$f^{k+1} \leq f^k + c_1 \alpha {g^k}^\top \qquad c_1 \in (0, 1) \tag{4.2}$$

$$\phi'(\alpha^k) \geq c_2 \phi'(0) \qquad c_2 \in (c_1, 1) \tag{4.3}$$

these condition are satisfied at every iteration of the algorithm. So, this will automatically guarantee that the value of the function decreases in every iteration. So, after finding α^k, *do* $f^{k+1} < f^k$ *for all* $k \geq 0$ *where* $x^{k+1} = x^k + \alpha^k d^k$, *since* f *bounded Below then* f^k *is monotonically decreasing sequence which is also bounded below.* $\left\{f^k\right\} \to f^*$ *where* $f^* < \infty$, *then* $f^0 - f^k < \infty \quad \forall k \geq 0$, *therefor* $\lim_{k\to\infty} f^0 - f^k < \infty$ *Using Armijo's condition's,*

$\alpha^{j's}$ *are chosen such that*

$$f^{k+1} \leq f^k + c_1 \alpha^k g^{k\top} d^k \tag{4.4}$$

$$\leq f^0 + c_1 \sum_{j=0}^{k} \alpha^j g^{j\top} d^j \tag{4.5}$$

therefor $\infty > f^0 - f^{k+1} \geq -c_1 \sum_{j=0}^{k} \alpha^j g^{j\top} d^j$ *this implies* $-c_1 \sum_{j=0}^{k} \alpha^j g^{j\top} d^j < \infty$, *since* $-c_1 < 0$, $\alpha^j > 0$ *and* $g^{j^\top} d^j < 0$.

therefore sum of infinitely many positive terms is finite, this implies, beyond certain iteration k, $\alpha^k g^{j^\top} d^j = 0$, *using wolfe's conditions,* α *chosen such that*

$$\phi^{'}(\alpha^k) \geq c_2 \phi^{'}(0) \qquad c_2 \in (c_1, 1) \tag{4.6}$$

$$\textit{sine} \quad \phi(\alpha^k) = f(x^k + \alpha^k d^k) \tag{4.7}$$

$$\textit{then} \quad \phi^{'}(\alpha^k) = \nabla f(x^k + \alpha^k d^k) d^k \tag{4.8}$$

$$= g^{j^\top} d^j \tag{4.9}$$

$$\phi^{'}(\alpha^k) \geq c_2 \phi^{'}(0) \qquad c_2 \in (c_1, 1) \tag{4.10}$$

$$g^{k+1^\top} d^k \geq c_2 g^{k^\top} d^k \tag{4.11}$$

$$(g^{k+1} - g^k)^k \geq (c_2 - 1) g^{k^\top} d^k \tag{4.12}$$

since g *be Lipschitz continuous that is* $\exists L, 0 < L < \infty$, *such that* $\left\| g^{k+1} - g^k \right\| \leq L \left\| x^{k+1} - x^k \right\|$, *but have* $x^{k+1} = x^k + \alpha^k d^k$ *then.*

$$\left\| g^{k+1} - g^k \right\| \leq L \alpha^k \left\| d^k \right\| \tag{4.13}$$

$$(g^{k+1} - g^k)^\top d^k \leq L \alpha^k d^{k^\top} d^k \tag{4.14}$$

However, using Wolfe's conditions

$$(g^{k+1} - g^k)^\top d^k \geq (c_2 - 1) g^{k\top} d^k \tag{4.15}$$

$$L\alpha^k {d^k}^\top d^k \geq (c_2 - 1) g^{k\top} d^k \tag{4.16}$$

$$\alpha^k \geq \frac{(c_2 - 1) g^{k\top} d^k}{L \left\| d^k \right\|^2} \qquad \textit{multiply} \quad \textit{by} \quad g^{k\top} d^k \tag{4.17}$$

$$\alpha^k {d^k}^\top d^k \leq \frac{(c_2 - 1)(g^{k\top} d^k)^2}{L \left\| d^k \right\|^2} \tag{4.18}$$

$$-c_1 \alpha^k g^{k\top} d^k \geq c_1 \frac{(c_2 - 1)(g^{k\top} d^k)^2}{L \left\| d^k \right\|^2} \tag{4.19}$$

since d^k change at each iteration so need to throughout d^k form last inequality let θ_k be the angle between g^k and d^k

$$-c_1 \alpha^k g^{k\top} d^k \geq c_1 \frac{(c_2 - 1)}{L} \frac{\left\| g^k \right\|^2 \left\| d^k \right\|^2 \cos^2 \theta_k}{\left\| d^k \right\|^2} \tag{4.20}$$

$$-c_1 \alpha^k g^{k\top} d^k \geq c_1 \frac{(c_2 - 1)}{L} \left\| g^k \right\|^2 \cos^2 \theta_k \tag{4.21}$$

therefore get bound of $\alpha^k g^{k\top} d^k$ but, using Armijo's conditions $-c_1 \sum_{k=0}^{\infty} \alpha^k g^{k\top} d^k \leq \infty$, therefore $c_1 \frac{(c_2 - 1)}{L} \sum_{k=0}^{\infty} \left\| g^k \right\|^2 \cos^2 \theta_k < \infty$ this implies $\left\| g^k \right\|^2 \cos^2 \theta_k \to 0$, since at every iteration, d^k is chosen such that $g^{k\top} d^k < 0$ and $\cos^2 \theta_k \geq \delta > 0$ then, ensure that $\cos^2 \theta_k$ does not go to zero at any point of time so there is only way to $\left\| g^k \right\|^2 \cos^2 \theta_k \to 0$ is $\left\| g^k \right\|^2 \to 0$

Infact, in this theorem, didn't use x^0 the initial point at any time so the result was derived irrespective of the initial point and this powerful result is called the globe convergence theorem.

Sufficient Decrease and Backtracking

Many times while dealing with practical problems it might be difficult to ensure that Armijo's-Wolfe conditions or Armijo-Goldstein conditions are satisfied [13]. So, in such

cases, it is proposed to use backtracking line search in combination with Armijo's, note that Armijo-Goldstein conditions choose α^k such that

$$\phi_2(\alpha^k) \leq f(x^k + \alpha^k d^k) \leq \phi_1(\alpha^k) \tag{4.22}$$

$$\text{where} \quad \phi_1(\alpha^k) = f(x^k) + c_1 \alpha {g^k}^\top, c_1 \in (0,1) \tag{4.23}$$

$$\text{and} \quad \phi_2(\alpha^k) = f(x^k) + c_2 \alpha {g^k}^\top, c_2 \in (c_1,1) \tag{4.24}$$

Algorithm 2 : Backtracking Line Search algorithm

(a) chose $\hat{\alpha}, (> 0) \quad p \in (0,1), c_1 \in (0,1), \text{set} \alpha = \hat{\alpha}$

(b) while$\phi_1(\alpha^k) \geq f(x^k) + c_1 \alpha {g^k}^\top, c_1 \in (0,1)$

i. $\alpha := \rho\alpha$

ii. end while

output $\alpha^k = \alpha$

.

The backtracking line search algorithm initially choose some value of $\hat{\alpha} > 0$, set $\alpha = \hat{\alpha}$ while the Armijo conditions are not satisfies at the given α this means $\phi_1(\alpha^k) \geq f(x^k) + c_1 \alpha {g^k}^\top, c_1 \in (0,1)$, then reduce the α by multiplying by $\rho(\alpha := \rho\alpha)$where ($\rho$ is positive fraction)and the process is repeated till Armijo conditions are satisfied, this will automatically ensure that, certain from large steps length and coming back to the smaller step length.

4.6 Descent Direction

Consider the unconstrained minimization problem $\min_{x \in R^n} f$ where $f \in C^1(R^n, R)$ with Lipschitz continuous gradient$g(x)$, let $g^k \neq 0$ and $d^k = -A^k g^k$ where A^k is a symmetric matrix.

If $A^k \succ 0$(positive definite) ${g^k}^\top d^k = -{g^k}^\top A g^k < 0 \Rightarrow d^k$ is descent direction, infact A^k is matrix which will rotate the direction $-g^k$ suitably [68].

If $A^k = I$ is(identity matrix) in this case $d^k = -g^k$ such direction is called steepest descent direction (is direction when the matrix A^k is identity matrix).

The different optimization algorithms use different case of A^k therefor there is different decent direction.

How to find d^k? consider the first order approximation to $f(x)$ about x^k

$$f(x) \approx \hat{f}(x) \approx f(x^k) + {g^k}^\top (x - x^k) = f(x^k) + {g^k}^\top d$$

Since $f(x^k)$ fixed quantity and g^k fixed quantity so only unknown quantity d, since trying to minimize the function $f(x)$ the best direction with respect to this first order approximation, will be the direction which will minimize $g^k d^k$ so, the maximum decreases in $\hat{f}(x)$ which is possible by solving

$$\begin{cases} \min_d & {\mathtt{g}^\mathtt{k}}^\top \mathtt{d} \\ \text{subject to} & \mathtt{d}^\top \mathtt{d} = \mathtt{1} \end{cases} \tag{4.25}$$

Let θ_k be the angle between g^k and d

$$\begin{aligned} {g^k}^\top d &= \left\|g^k\right\| \|d\| \cos\theta_k \\ &= \left\|g^k\right\| \cos\theta_k \qquad (\|d\| = 1) \end{aligned}$$

Since g^k fixed quantity then the only way to minimize ${g^k}^\top d$ is by minimizing $\cos\theta_k$ or chose d such that ${g^k}^\top d$ is minimized, when $\cos\theta_k$ is minimum value of $\cos\theta_k$ is -1 so there fore the solution is $d^k = \dfrac{-g^k}{\|g^k\|}$.

4.6.1 The Steepest-Descent Method

The steepest descent method is one of the oldest and well-known search techniques for minimizing multi-variable unconstrained optimization problems. This method has played an important role in the development of advanced optimization algorithms. It is a first-order derivative iterative optimization algorithm whose convergence is linear for the case of quadratic functions [49, 79]. If take steps in the direction of a negative gradient of the function at the given current point to find a local minimum point, then this procedure is called Gradient Descent [49]. On the other hand, if take steps in the direction of the positive gradient at the current point to find a local maximum point, then such procedure is known as Gradient Ascent. The performance of the steepest descent method depends on the initial choice of a point x_0. If start far away from the optimum, the method converges rapidly but as get closer to the optimum, it becomes very sluggish [6]. A new choice always alters the convergence characteristics. The steepest descent method is also known as Cauchy's method [49]. Steepest descent (SD) method is a simple gradient method for the unconstrained optimization:

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{4.26}$$

where $f(x)$ is a continuous differential function in $\mathbb{R}^n$, now present the following steepest descent algorithm.

Algorithm : Steepest Descent Method Algorithm

Use the steepest descent direction $d_{SD}^k = -g^k$

(a) Initialize x^o and ϵ,set $k := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. $d^k = -g^k$

ii. Find $\alpha^k (> 0)$ along d^k such that

- $f(x^k + \alpha^k d^k) \leq f(x^k)$
- α^k satisfies Armigo-Wolfe condition

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. k:=k+1 end while

output $x^* = x^k$ a stationary of $f(x)$

Example 4.6.1. Apply steepest descent method to minimize the function

$$f(x) = 2x_1^2 + 2x_2^2 + 6x_1 + 13 \tag{4.27}$$

with a starting point $x_{(0)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

The gradient and Hessian of function f are:

$$\nabla f(x) = \begin{bmatrix} 4x_1 + 6 \\ 4x_2 \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

The gradient of f at $x_{(0)}$ is:

$$\nabla f(x_{(0)}) = \begin{bmatrix} 10 \\ 8 \end{bmatrix}$$

The step length α_0 for quadratic function is computed as:

$$\alpha_0 = \frac{\nabla^T f(x_{(0)}) \nabla f(x_{(0)})}{\nabla^T f(x_{(0)}) A \nabla f(x_{(0)})} = \frac{\begin{bmatrix} & \\ 10 & 8 \end{bmatrix} \begin{bmatrix} 10 \\ \\ 8 \end{bmatrix}}{\begin{bmatrix} & \\ 10 & 8 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 10 \\ \\ 8 \end{bmatrix}} = \frac{100 + 64}{\begin{bmatrix} & \\ 40 & 32 \end{bmatrix} \begin{bmatrix} 10 \\ \\ 8 \end{bmatrix}} = \frac{164}{656} = 0.25$$

The next point is generated as:

$$x_{(1)} = x_{(0)} - \alpha_0 \nabla f(x_{(0)}) = \begin{bmatrix} 1 \\ \\ 2 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 10 \\ \\ 8 \end{bmatrix} = \begin{bmatrix} 1 \\ \\ 2 \end{bmatrix} - \begin{bmatrix} 2.5 \\ \\ 2 \end{bmatrix} = \begin{bmatrix} -1.5 \\ \\ 0 \end{bmatrix}$$

The gradient of f at $x_{(1)}$ is:

$$g_{(1)} = \nabla f(x_{(1)}) = \begin{bmatrix} 4(-1.5) + 6 \\ \\ 4 \times 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix}$$

Since the gradient of f at $x_{(1)}$ is zero, thus the steepest descent algorithm stops. The minimum point is $x^* = x_{(1)} = \begin{bmatrix} -1.5 \\ \\ 0 \end{bmatrix}$, and the minimum value of the objective function is $f(x^*) = 8.5$.

Example 4.6.2. Find optimal solution of unconstraint min $f(x) = (x_1 - 7)^2 + (x_2 - 2)^2$ by using steepest descent method.

solution then the gradient $\nabla f(x) = g(x) = \begin{pmatrix} 2(x_1 - 7) \\ \\ 2(x_2 - 2) \end{pmatrix}$ since $g(x^*) = \begin{pmatrix} 0 \\ \\ 0 \end{pmatrix}$ then $x^* = \begin{pmatrix} 7 \\ \\ 2 \end{pmatrix}$ and $H(x) = \begin{pmatrix} 2 & 0 \\ \\ 0 & 2 \end{pmatrix}$ is minimal point behaviour of the steepest descent algorithm (exact line search) applied to $f(x)$, would take us to the solution in exactly one step.

Code 13 :Behaviour of The Steepest Descent Method at Circular Contour

```
import numpy as np
def p1(X, a=1, b=100):
x, y = X
return (x-7)**2 +(y-2)**2
def p1_grad(X, a=1, b=100):
x, y = X
return np.array([
2 *(x-7),
2 * (y-2)
])
def p1_hess(X, a=1, b=100):
x, y = X
return np.matrix([
[2, 0],
[0, 2 ]
])
def CG(J_grad, J_hess, x_init, epsilon=1e-10, max_iterations=1000):
x = x_init
```

```
for i in range(max_iterations):
x = x - np.linalg.solve(J_hess(x), J_grad(x))
if np.linalg.norm(J_grad(x)) < epsilon:
return x, i +1
return x, max_iterations
x_init = [-1,2]
x_min ,it = CG(p1_grad, p1_hess, x_init,)
print('x* =', x_min)
print('p1(x*) =', p1(x_min))
print('Grad p1(x*) =', p1_grad(x_min))
print('Iterations =',it)
x* = [7. 2.]
p1(x*) = 0.0
Grad p1(x*) = [0. 0.]
Iterations = 1
```

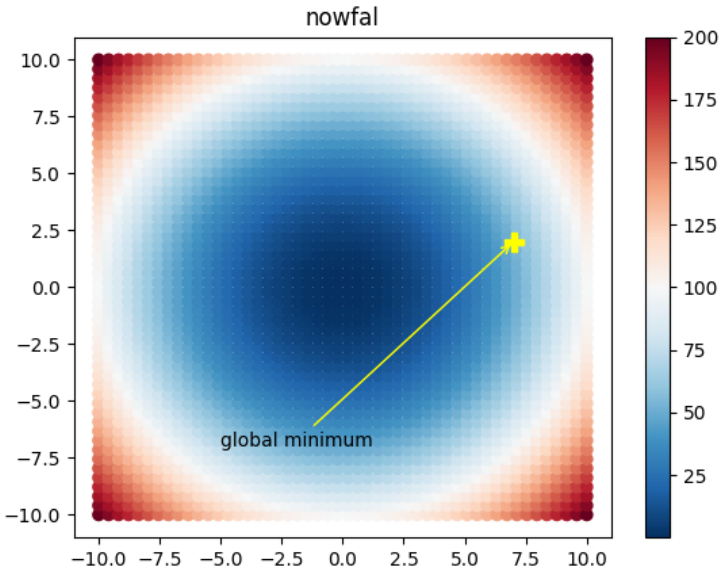


Figure 4.4: the steepest descent and function have circular contour, the solution in exactly one step

Example 4.6.3. Find optimal solution of unconstraint min $f(x) = 4x^2{}_1 + x^2{}_2 - 2x_1x_2$ by using steepest descent method.

Solution $\nabla f(x) = g(x) = \begin{pmatrix} 8x_1 - 2x_2 \\ 2x_2 - 2x_1 \end{pmatrix}$ and $H(x) = \begin{pmatrix} 8 & -2 \\ -2 & 2 \end{pmatrix}$ which is elliptical quadratic contours and clearly $x^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ is the solution of the problem(elliptical contours) on the organ, have the minimum function value let us applied steepest descent method with exact line search let the initial point $x^0 = (-1, -2)$.

Code 14 : Behavior of The Steepest Descent Method at Elliptical Contours

```
import numpy as np

def p2(X, a=1, b=100):

x, y = X
```

```
return 4*x**2 +y**2-2*x*y
def p2_grad(X, a=1, b=100):
x, y = X
return np.array([
8*x-2*y,
2*y-2*x
])
def p2_hess(X, a=1, b=100):
x, y = X
return np.matrix([
[8, -2],
[-2, 2 ]
])
def CG(J_grad, J_hess, x_init, epsilon=1e-10, max_iterations=1000):
x = x_init
for i in range(max_iterations):
x = x - np.linalg.solve(J_hess(x), J_grad(x))
if np.linalg.norm(J_grad(x)) > epsilon:
return x, i +1
return x, max_iterations
x_init = [-1,-2]
x_min ,it = CG(p2_grad, p2_hess, x_init,)
print('x* =', x_min)
print('p2(x*) =', p2(x_min))
print('Grad p2(x*) =', p2_grad(x_min))
print('Iterations =',it)
```

will reach the minimal after 1000 iteration

```
x* = [0. 0.]
```

```
p2(x*) = 0.0
Grad p2(x*) = [0. 0.]
Iterations = 1000
```

When function have elliptical contour, use the steep descent method will reach the optimal solution after 1000 iteration.

Example 4.6.4. Apply the steepest descent method (Rosenbrock function)

$$\min f(x) = 100(y - x^2)^2 + (1 - x)^2$$

the minimum objective occurs at $x^* = \begin{pmatrix} 0 \\ \\ 0 \end{pmatrix}$ and the minimum objective function value is$f(x^*) = 0$.

Code 15 :Rosenbrock function

```
import numpy as np


def rosenbrock(X, a=1, b=100):
    x, y = X
    return (a - x)**2 + b * (y - x**2)**2


def rosenbrock_grad(X, a=1, b=100):
    x, y = X
    return np.array([
        2 * (x - a) - 4 * b * x * (y - x**2),
        2 * b * (y - x**2)
    ])


def rosenbrock_hess(X, a=1, b=100):
```

```
    x, y = X
    return np.matrix([
        [2 - 4 * b * (y - 3 * x**2), -4 * b * x],
        [-4 * b * x, 2 * b]
    ])
def gradient_descent(J_grad, x_init, alpha=0.01, epsilon=1e-10,
 max_iterations=1000):
    x = x_init
    for i in range(max_iterations):
        x = x - alpha * J_grad(x)
        if np.linalg.norm(J_grad(x)) < epsilon:
            return x, i + 1
    return x, max_iterations
x_init = (.6,.6)
x_min, it = gradient_descent(rosenbrock_grad, x_init, alpha=0.002
, max_iterations=5000)
print('x* =', x_min)
print('Rosenbrock(x*) =', rosenbrock(x_min))
print('Grad Rosenbrock(x*) =', rosenbrock_grad(x_min))
print('Iterations =', it)
Output:x* = [0.99652641 0.99305096]
Rosenbrock(x*) = 1.2085206548502745e-05
Grad Rosenbrock(x*) = [-0.00139494 -0.00278579]
Iterations = 5000
```

k	x_1	x_2	$f(x^k)$	$\| x^k - x^* \|$	$\| g^k \|$
1	0.6	0.6	5.92	0.5657	75.50
2	0.72	0.52	0.0792	0.5601	0.3938
3	0.78	0.61	0.046	0.4414	0.245
4	0.9914	0.9828	$7.45 \times 10^{-}5$	0.0192	0.0069
5	0.99652641	0.99305096	$1.81 \times 10^{-}5$	0.0020	$9.97 \times 10^{-}5$

Table 4.1: The Newton method

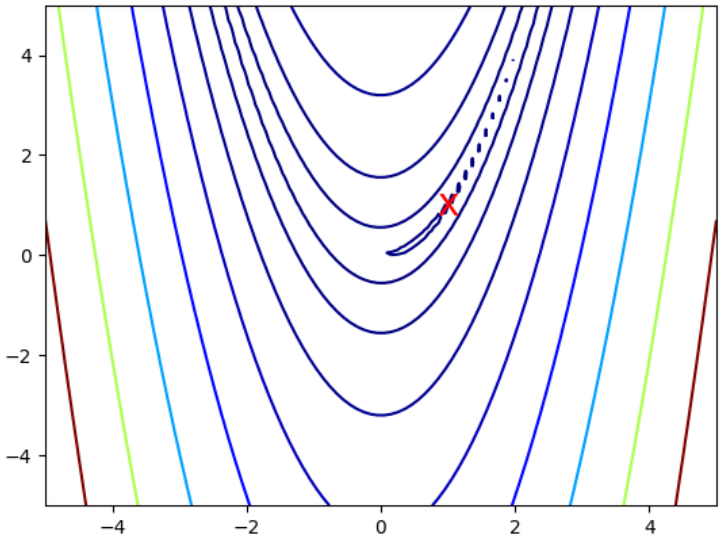


Figure 4.5: Generalized n-dimensional version of the Rosenbrock function

Lemma

Let $H \in R^{n \times n}$ be a symmetric positive definite matrix [23], let λ_1 and λ_n be the smallest and largest eigenvalues of H, for any $x \neq 0$

$$\frac{(x^\top x)^2}{(x^\top Hx)(x^\top H^{-1}x)} \geq \frac{4\lambda_n \lambda_n}{(\lambda_1 + \lambda_n)^2}$$

4.6.2 Convergence of Steepest Descent Method Quadratic Case

Consider the problem $\min f(x)_{x\in R^n} = \frac{1}{2}x^\top Hx - c^\top x$, where H is asymmetric positive-define matrix so the problem is the quadratic the reason for studying these quadratic function is that for many function even if they are non-quadratic near x^* or near their minimum, the behaviour of the function like quadratic function, need to study the steepest descent method for quadratic case first, before move on to non-quadratic case [34, 67]. In the former examples of our mention, if do the steepest descent method with exact line search be have in this particular way zigzagging direction to the quadratic function which have elliptical contours, the gradient of the above equation is $g(x) = Hx - c$ the optimal point $g(x^*) = Hx^* - c = 0 \Rightarrow x^* = H^{-1}c$, since H is positive define matrix. So, the H^{-1}since in the steeped descent method at each iteration the direction choose as $d^k = -g^k$ then $x^{k+1} = x^k - \alpha^k g^k$ and also assume that exact line search is use in each iteration then the step length α^k at each iteration it easy to computing, have $f(x) = \frac{1}{2}x^\top Hx - c^\top x, g^k = Hx^k - c$ [23].

Want to find out what is the value of $\alpha > 0$ such that $\phi(\alpha)$ minimized $\phi(\alpha) = f(x^k - \alpha d^k) = f(x^k - \alpha g^k)$, where α chose by exact line search $\alpha^k = arg\min_{\alpha>0}\phi(\alpha) \Rightarrow \phi'(\alpha) = 0 \Rightarrow$

$\nabla f(x^k - \alpha g^k)^\top (g^k) = 0$

$\Rightarrow (Hx^k - \alpha Hg^k - c)^\top g^k = 0$

but $\quad Hx^k - c = g^k$

$\Rightarrow (g^k - \alpha Hg^k)^\top g^k = 0$

$\alpha^k = \dfrac{{g^k}^\top g^k}{{g^k}^\top Hg^k} \Rightarrow x^{k+1} = x^k + \alpha^k d^k$

$x^{k+1} = x^k + \dfrac{{g^k}^\top g^k}{{g^k}^\top Hg^k} d^k \Rightarrow x^{k+1} = x^k - \dfrac{{g^k}^\top g^k}{{g^k}^\top Hg^k} g^k$

At what rate does $\{x^k\}$ converge to minimize.

Let us define $E(x^k) = \frac{1}{2}(x^k - x^*)H(x^k - x^*)$ where $(E(x^k) > 0 \quad \text{if} \quad x^k \neq x^*\ )$ note that $E(x^*) = f(x^k) + \frac{1}{2}{x^*}^\top Hx^* \quad (\frac{1}{2}{x^*}^\top Hx^*)$ is constant, some time $E(x^k)$ is said to be error function. Note that the minimize of $E(x^k)$ is the same as minimize of $f(x^k)$, will study

the behaviour of steepest descent with respect $E(x)$. Define

$$y^k = x^k - x^* \therefore \quad Hy^k = g^k \quad \text{using}$$

$$x^{k+1} = x^k - \frac{{g^k}^\top g^k}{{g^k}^\top H g^k} g^k$$

$$\frac{E(x^k) - E(x^k + 1)}{E(x^k)} = \frac{(x^k - x^*)H(x^K - x^*) - (x^{k+1} - x^*)^\top H(x^{k+1} - x^*)}{{y^k}^\top H y^k}$$

$$= \frac{2\alpha^k {g^k}^\top g^k - {\alpha^k}^2 {g^k}^\top H g^k}{{y^k}^\top H y^k}$$

$$\text{substituting} \quad \alpha^k = \frac{{g^k}^\top g^k}{{g^k}^\top H g^k}, \quad \text{get}$$

$$\frac{E(x^k) - E(x^{k+1})}{E(x^k)} = \frac{({g^k}^\top g^k)^2}{({g^k}^\top H g^k)({g^k}^\top H^{-1} g^k)}$$

This quantity is still depended on the matrix H^{-1} and H when $g^k \neq 0$, will use this inequality Kantorovich Inequality

$$\frac{E(x^k) - E(x^{k+1})}{E(x^k)} = \frac{({g^k}^\top g^k)^2}{({g^k}^\top H g^k)({g^k}^\top H^{-1} g^k)}$$

$$\geq \frac{4\lambda_1 \lambda_n}{(\lambda_1 + \lambda_n)^2}$$

therefore

$$E(x^{k+1}) \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_1 + \lambda_n}\right)^2 E(x^k)$$

decreasing sequence of $E(x^k)$ therefore $E(x^k) \Rightarrow 0$ and $x^k \Rightarrow x^*$(H is positive definite) with respect to E, the steepest descent method converges linearly with convergence rate no greater than $\left(\frac{\lambda_n - \lambda_1}{\lambda_1 + \lambda_n}\right)^2$ actual convergence rate depends upon x^0, define the condition number of H, $r = \frac{\lambda_n}{\lambda_1}$ convergence rate of the steepest descent method depends on the condition number of H or $r = \frac{\lambda_n}{\lambda_1}$, $r = 1$(circular contours)$\Rightarrow$ convergence is one iteration, $r >> 1$ (elliptical contours)$\Rightarrow$ convergence required many more iterations before the algorithm reach the point x^* so the convergence is very slow, for non-quadratic function rate of convergence to x^* depends on the condition number of $H(x^*)$.

4.6.3 Convert the Quadratic Function of Elliptical Contour to One with Circular Contour

Consider the problem $\min f(x) = \frac{1}{2}x^\top Hx - c^\top x$, where H is symmetric positive definite matrix, let $H = LL^\top$ be the Cholesky decomposition of H (If H is positive definite, there exists an upper triangular matrix L such that $H = LL^\top$
This decomposition is unique, and it is called the Cholesky Decomposition), define $y = L^\top x$ therefore the function $f(x)$ is transformed to the function $h(y)$ and $h(y) = f(L^{-\top}y)$

$$\begin{aligned}\frac{1}{2}x^\top Hx - c^\top x &= \frac{1}{2}(yL^{-\top})H(yl^{-1}) - c^\top L^{-1}y \\ &= \frac{1}{2}(yL^{-\top})LL^{-\top}(yL^{-1}) - c^\top L^{-1}y \\ &= \frac{1}{2}yy^\top - c^\top yL^{-\top}\end{aligned}$$

the Hessian matrix of $h(y)$ is I let us apply the steepest decent method in $y-$space

$$y^{k+1} = y^k - \nabla h(y^k)$$
$$y^{k+1} = y^k - (\nabla f(L^{-1}y)L^{-1}) \qquad (\times L^{-1})$$

will getting

$$L^{-1}y^{k+1} = L^{-1}y^k - \nabla f(L^{-1}y)L^{-1}L^{-1}$$

since $H = LL^\top, H^{-1} = L^{-\top}L^{-1}$

$$y = L^\top x \Rightarrow x = yL^{-\top}$$
$$x^{k+1} = y^{k+1}L^{-\top} \Rightarrow x^{k+1} = x^k - \nabla f(x^k)H^{-1}$$

4.7 Newton Method

Consider the problem $\min\limits_x f(x)$ assume that f is twice continuously differentiable and also assume that f be bounded below, the search direction based on the first and second derivative information denote by $d^k = -H^{-1}g^k$, where g^k is the gradient at x^k.

Every iteration by using Taylor series to approximate f at x^k by a quadratic function and then finding the minimum of the quadratic function to get x^{k+1} and this process is repeated until the criterion is satisfied [34, 42].

$$f(x) \approx f_q(x) = f(x^k) + {g^k}^\top (x - x^k) + \frac{1}{2}(x - x^k)^\top H^\top (x - x^k)$$

where H^k is the Hessian at the current point x^k, the new iteration is obtain by minimizing the quadratic function $f_q(x)$ this means that $x^{k+1} = \text{argmin}_x f_q(x)$ the gradient of f_q denote by $\nabla f_q(x) = 0 \Rightarrow x^{k+1} = x^k - (H^k)^{-1} g^k$(assuming H^k is in revertible), if write it in our visual form $x^k + \alpha^k d^k$, chosen $\alpha^k = 1$ and ${d_N}^- (H^k)^{-1} g^k$ is called the Newton direction. The Newton direction is descent direction if H is positive definite ${g^k}^\top {d_N}^k = -{g^k}^\top (H^k)^{-1} g^k < 0$ (it satisfy when $H \succ 0$)

Consider the problem to minimize $f(x) = \frac{1}{2} x^\top H x - c^\top x$ where H is a symmetric positive matrix $g(x) = 0 \Rightarrow x^* = H^{-1} c$ is a strict local minimum let $x^o \in R^o$ be any point $g(x^o) = Hx^o - c$, $H(x^o) = H$ using classical Newton method $x^1 = x^o - H^{-1}(Hx^o - c) = H^{-1} c = x^*$ for a quadratic function, if apply classical Newton method starting from any point can reach the solution in exactly one step, assuming that the initial point is not the optimal point compare this with the behaviour of the steepest descent method depends alto on the condition number of the Hessian matrix $H = 1$ converges in one step starting from any point and if the condition number of the a Hessian matrix $H > 1$ then that the zigzagging takes place for a typical starting point.

In general using classical Newton method, the minimum of a strictly convex quadratic function (with in revertible Hessian matrix) is attained in one iteration form any starting point.

Algorithm : Newton Method Algorithm

(a) Initialize x^o and ϵ,set $k := 0$

(b) while $\left\| g(x^k) \right\| \geq \epsilon$

i. $d^k = -(H^{k^{-1}}) g^k$

ii. find $\alpha^k = 1$

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. K:=K+1 end while

output $x^* = x^k$ a stationary of $f(x)$

.

Requires $O(n^3)$ computational effort for every iteration(at every iteration there is an order n^3) computational(step2(a)) effort needed and that will be expensive if the number of iterations go up, not only that the order in to computational effort is needed but the Newton method also requires order n^2computational storage because, need to store the Hessian matrix at every iteration and that requires order n^2 storage so both storage wise as will as the computational effort wise Newton method is expensive compared to stick descent method steep because does not use any second order information, so there is no need to store any matrix and there is no need to invert any matrix while in Newton method need to store H^k as will as invert it in every iteration and that is going to be very expensive computationally for large dimensional problems.

Also in this algorithm there is no guarantee that d^k descent direction (choose in step 2), notation algorithm does not cheek the positive definiteness of the Hessian matrix at any iteration k, one more notation same time the matrix H^k also could be close to a singular matrix, therefore inverting a matrix which is closed to a singular matrix will give rise to some numerical difficulties those are not handled in the classical Newton algorithm.

Another draw back of algorithm is that there is no guarantee that objective function value decreases at every iteration and this is because there is no line search which done here($\alpha^k = 1$) (no guarantee that $f(x^{k+1}) < f(x^k)$), also algorithm sensitive to initial point (for non-quadratic problem)

Definition 4.7.1. An iteration optimization algorithm is said to be locally convergent if for each solution x^*, there exists $\delta > 0$ such that for any initial point $x^o \in B(x^*, \delta)$, the algorithm produces a sequence $\{x^k\}$ which converges to x^*,(δ is function of x^*) [50].

Theorem 4.7.1. *Let $f : R \to R$, $f \in C^2$, let $x^* \in R$ be such that $g(x^*) = 0$ and $g'(x^*) > 0$ then, provided x^o is sufficiently closed to x^*, the sequence $\{x^k\}$ generated by classical Newton algorithm converges to x^* with an order of convergence two(locally convergent) [50].*

Proof. Since $f : R \to R$, $f \in C^2$ consider the problem min $f(x)$, and since $x^* \in R$ be such that $g(x^*) = 0$ and $g'(x^*) > 0$ assume that x^o is sufficiently closed to x^*, suppose apply classical Newton algorithm to minimize $f(x)$.
At k^{th} iteration

$$\begin{aligned} x^{k+1} &= x^k - \frac{g(x^k)}{g'(x^k)} \\ x^{k+1} - x^* &= x^k - x^* - \frac{g(x^k) - g(x^*)}{g'(x^k)} \\ &= -\frac{(g(x^k) - g(x^*) + g'(x^k)(x^* - x^k)}{g'(x^k)} \end{aligned}$$

If assume that $f \in C^3$(or $g \in C^2$), then using truncated Taylor series

$$g(x^*) = g(x^k) + g'(x^k)(x^* - x^k) + \frac{1}{2}g''(\bar{x^k})(x^* - x^k)^2$$

, where $\bar{x^k} \in LS(x^*, x^k)$
Therefore

$$x^{k+1} - x^* = \frac{1}{2}\frac{g''(\bar{x^k})}{g'(x^k)}(x^k - x^*)^2$$

got relation ship between (x^{k+1} and x^*),($x^k$and x^*)^2, if recall the definition of convergence of algorithm, this would turn out to be order 2convergence,

$$\left|x^{k+1} - x^*\right| = \frac{1}{2}\frac{\left|g''(\bar{x^k})\right|}{|g'(x^k)|}\left|(x^k - x^*)^2\right|$$

suppose there exist α_1 and α_2 such that $\left|g''(\bar{x^k})\right| < \alpha_1$ for all $\bar{x^k} \in Ls(x^*, x^k)$

$\left|g'(x^k)\right| > \alpha_2$ for all x^k sufficiently close to x^* then

$$\left|x^{k+1} - x^*\right| \leq \frac{1}{2}\frac{\alpha_1}{\alpha_2}\left|(x^k - x^*)^2\right|$$

where $\frac{1}{2}\frac{\alpha_1}{\alpha_2} > 0$ and constant (order two convergence if $x^k \to x^*$)

$$\left|x^{k+1} - x^*\right| \leq \frac{\alpha_1}{2\alpha_2}(\left|(x^k - x^*)\right|)\left|(x^k - x^*)\right|$$

where $\frac{\alpha_1}{2\alpha_2}(\left|(x^k - x^*)\right|)$ required to be < 1, then got the distance between x^{k+1} and x^* less than distance between x^k and x^*

$$\left|x^{k+1} - x^*\right| < \left|x^k - x^*\right|, \forall k$$

.

Choose α_1 and α_2 in some way such that the inequality

$$\left|x^{k+1} - x^*\right| < \left|x^k - x^*\right|, \forall k$$

holds. Now, $g(x^*) = 0$ and $g'(x^*) > 0$ since $g' \in C^o(g')$ continues $\exists \eta > 0 \ni g'(x) > 0$, $\forall x \in (x^* - \eta, x^* + \eta)$

$$\alpha_1 = \max_{x \in (x^* - \eta, x^* + \eta)} |g''(x)|$$

$$\alpha_2 = \min_{x \in (x^* - \eta, x^* + \eta)} |g'(x)|$$

$$\text{Therefore} \left|\frac{1g''\bar{x^k}}{2g'\bar{x^k}}\right| \leq \frac{\alpha_1}{2\alpha_2}$$

preferable to choose $x^o \in (x^* - \eta, x^* + \eta)$ also, want $\beta\left|x^k - x^*\right| < 1, \forall k$ that is $\left|x^k - x^*\right| < \frac{1}{\beta}, \forall k \Rightarrow x^k \in (x^* - \frac{1}{\beta}, x^* + \frac{1}{\beta})$, therefore choose $x^o \in (x^* - \eta, x^* + \eta) \bigcap (x^* - \frac{1}{\beta}, x^* + \frac{1}{\beta})$.

Now, will show if x^o choose in this regain then x^k converge to x^*

$$\begin{aligned}
\left|x^k - x^*\right| &\leq \left|x^{k-1} - x^*\right|^2 \\
\beta\left|x^k - x^*\right| &\leq \left(\beta\left|x^{k-1} - x^*\right|\right)^{2^k} \\
\left|x^k - x^*\right| &\leq \frac{1}{\beta}\underbrace{\left(\left|x^{k-1} - x^*\right|\right)^{2^k}}_{<1} \Rightarrow \lim_{\beta\to\infty}\left|x^k - x^*\right| = 0
\end{aligned}$$

□

the problem is that the initialization of x^o require knowledge of x^* and our aim is to minimize $f(x)$ to get x^*. So, this knowledge of x^* is not there cannot initialize x^o properly, so that, can get global convergence of Newton method. In other words, the Newton method does depend a lot on x^o.

4.7.1 Modification of Newton Method

Given x^k and ${d_N}^k = -(H^k)^{-1}g^k$, there is two possibilities one is that the Hessian matrix is not invertible or is close to singular. So in that case this direction really does not make much sense.

Secondly, the Hessian matrix is suppose negative definite so in that case this direction is not descent direction, will modify Newton method so that, this expression is positive definite matrix. Add $\zeta_k \geq 0$, suppose if you fix some constant $\delta > 0$ find the smallest $\zeta_k \geq 0$ such that the smallest eigenvalue of the matrix $(H^k + \zeta_k I)$ is greater than δ. Therefore, $d^k = -(H^k + \zeta_k I)^{-1}g^k$ is a descent direction [becouse$(H^k + \zeta_k I)$] positive definite. If recall that the classical Newton method does not use line search($\alpha^k = 1$) instead of. That can, if are given x^k and $d^k = -(H^k + \zeta_k I)^{-1}g^k$ will use line search techniques to determine α^k and $x^{k+1} = x^k + \alpha^k d^k$.

Algorithm 5 : Modified Newton Algorithm

(a) Initialize x^o and ϵ and δ,set $k := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

- Find the smallest $\zeta_k \geq 0$ such that the smallest eigenvalue of $H^k + \zeta_k I$ is greater than δ
- set $d^k = -(H^k + \zeta_k I)^{-1} g^k$
- find $\alpha^k (> 0)$ along d^k such that
  - i. $f(x^k + \alpha^k d^k) < f(x^k)$
  - ii. α^k satisfies (Armijo-Wolfe)
- $x^{k+1} = x^k + \alpha^k d^k$
- k:=k+1 end while

output $x^* = x^k$ a stationary of $f(x)$

.

Modified Newton algorithm has global convergence properties and has order of convergence equal to two, also modified is not sensitive to x^o

4.8 Quasi-Newton Method

The class of methods helps to determine the optimum point faster and more proficiently than Newton's method. More precisely, these methods help to determine the global minimum point of a function $f(x)$ that is twice differentiable, the quasi-Newton is same Newton's method with an alteration to the Hessian matrix determination step, but this alteration depend upon the particular class of quasi-Newton method being used. In Newton's obtaining the second order derivatives and thus an estimation of the Hessian matrix at a particular point, while quasi-Newton gradually generate an approximate Hessian matrix by utilizing gradient information from specific previous iterative point

x^k [25, 44]. Suppose now that the search direction has the form $d^k = -B^{k^{-1}}\nabla^k f$, where B^k is positive definite matrix(B^k is either H^k or approximation).

Infact B^k is updated at every iteration by updating formula of quasi-Newton, suppose steps length α^k choose such that satisfy the Arimijo-wolf condition, line search algorithm will always try the step length $\alpha = 1$ first and will accept this value if it satisfies the Arimijo-wolf condition.

Consider the problem $\min_x f(x)$ where f real value function define $f : R^n \to R$ and $f \in C^1$ continuously differentiable.

Let as use Newton method

$$f(x) \approx {f_q}^k(x) = f(x^k) + {g^k}^\top(x - x^k) + \frac{1}{2}(x - x^k)H^k(x - x^k)$$

with Newton direction ${d_N}^k = -(H^k)^{-1}g^k$ where${f_q}^k(x)$ quadratic function at the given point and required approximated with the first and second derivative information.

In quasi-Newton do not use the second order derivative information, then can say since $f \in C^1$, form quadratic model of f at x^k(second derivative not a viable)

$$y_k(x) = f(x^k) + {g^k}^\top(x - x^k) + \frac{1}{2}(x - x^k)^\top B^{k^{-1}}(x - x^k)$$

where B^k is asymmetric positive define matrix and the quasi-Newton direction $d^k = -B^k d^k$.

4.9 Update B^k to get B^{k+1}

Know that the new iteration given by $x^{k+1} = x^k + \alpha^k d^k = x^k - \alpha^k B^k g^k$, construct a quadratic approximation of f at x^{k+1}

$$y_{k+1}(x) = f(x^{k+1}) + {g^{k+1}}^\top(x - x^{k+1}) + \frac{1}{2}(x - x^{k+1})^\top(B^{k+1})^{-1}(x - x^{k+1})$$

.

Require always satisfy $\nabla y_{k+1}(x^k) = \nabla f(x^k)$

$\nabla y_{k+1}(x^{k+1}) = \nabla f(x^{k+1}) = g^{k+1}$

$\nabla y_{k+1}(x^k) = \nabla f(x^k) = g^k = g^{k+1} + (B^{k+1})^{-1}(x^k - x^{k+1})$

Letting $g^{k+1} - g^k = \gamma^k$ and $x^{k+1} - x^k = \delta^k \rightarrow B^{k+1}\gamma^k = \delta$

B^{k+1} (is invertible matrix), this condition denote by quasi-Newton condition, B^{k+1} should be positive definite, to prove that

Claim $B^{k+1} \succ 0$

Since B^{k+1} satisfy quasi-Newton condition, then $B^{k+1}\gamma^k = \delta$

${\gamma^k}^\top B^{k+1}\gamma^k = {\gamma^k}^\top \delta^k > 0 \forall \gamma^k \neq 0.$

If use Wolfe conditions for line search

$$
\begin{aligned}
{g^{k+1}}^\top d^k \geq & c_2 {g^k}^\top d^k, c_2 \in (1,0) \\
{g^{k+1}}^\top d^k - & c_2 {g^k}^\top d^k \geq 0 \\
\Rightarrow ({g^{k+1}}^\top - & c_2 {g^k}^\top) d^k \geq 0 \\
\Rightarrow ({g^{k+1}}^\top - & c_2 {g^k}^\top)(x^{k+1} - x^k) \geq 0
\end{aligned}
$$

since $c_2 > 0$ then can say ${\gamma^k}^\top \delta^k > 0$.

Finally:

If α^k choose satisfy the Wolfe's condition then $\exists B^{k+1}$ such that $B^{k+1}\gamma^k = \delta^k$ where B^{k+1} is symmetric matrix of size n has $\frac{n(n+1)}{2}$ independent variables to be found using n equations and n inequalities.

A new Hessian approximation B^{k+1} is computed as an update of B^k:

$$B^{k+1} = B^k + V^k$$

such that B^{k+1} satisfies the Quasi-Newton condition and V^k the updating matrix at end of iteration k. Next describe three symmetric updates that satisfy the property of positive definiteness.

There are a number of questions trying to explain it next.

(a) Is B^{k+1} symmetric positive definite matrix?.

(b) Is $B^k \approx (H^k)^{-1}$?

(c) Are there any conditions that B^{k+1} should satisfy

4.9.1 Symmetric Rank-One Update

The Symmetric Rank one ($SR1$) method is a quasi-Newton method to update the second derivative (Hessian) based on the derivatives (gradients) calculated at two points. It is a generalization to the secant method for a multidimensional problem. This update maintains the symmetry of the matrix but does not guarantee that the update be positive definite [66]. The sequence of Hessian approximations generated by the $SR1$ method converges to the true Hessian under mild conditions, in theory; in practice, the approximate Hessians generated by the $SR1$ method show faster progress towards the true Hessian then do popular alternatives ($BFGSorDFP$), in preliminary numerical experiments. The $SR1$ method has computational advantages for sparse or partially separable problems.

Let $\alpha \neq 0$, $u \in R^n$, $u \neq 0$, $B^{k+1} = B^k + \alpha\mu\mu^\top$ since α scalar and $\mu\mu^\top$ is symmetric matrix of rank one, called to this method (Rank-one correction) [64], sinceB^k symmetric matrix also $\alpha\mu\mu^\top$ symmetric matrix then B^{k+1} symmetric matrix, want show if B^k positive definite then B^{k+1} positive define and what value of μ andα shall choose to get B^{k+1} positive definite.

Choose α and μ such that B^{k+1} satisfies Quasi-Newton condition $(B^k + \alpha\mu\mu^\top)\gamma^k = \delta^k$,

$\therefore \alpha\mu^\top\gamma^k\mu = \delta^k - B^k\gamma^k$ let $\mu = \delta^k - B^k\gamma^k$

$\therefore \alpha\mu^\top\gamma^k = 1 \Rightarrow \alpha = \dfrac{1}{\mu^\top\gamma^k} \therefore \alpha^{-1} = \mu^\top\gamma^k \Rightarrow \alpha^{-1} = (\delta^k - B^k\gamma^k)^\top\gamma^k.$

Substitute its value of μ and α in getting

$$B_{SR1}^{k+1} = B^k + \frac{(\delta^k - B^k\gamma^k)^\top(\delta^k - B^k\gamma^k)}{(\delta^k - B^k\gamma^k)^\top\gamma^k}$$

infact B^{k+1} obtained using $x^k, x^{k+1}, g^k$and, g^{k+1}

Algorithm 6 : Quasi-Newton Algorithm(Rank-One Correction)

(a) Initialize x^o , ϵ and symmetric positive definite $B^o = I$,set $k := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. Find a descent direction $d^k = -B^k g^k$

ii. find $\alpha^k (> 0)$ along d^k such that

- $f(x^k + \alpha^k d^k) < f(x^k)$
- α^k satisfies Armijo-wolfe conditions

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. Find B^{k+1} using $SR1$

v. $k := k + 1$ end while

output $x^* = x^k$ a stationary of $f(x)$

---

proposition 2. *Hereditary Property [71] For the symmetric rank-one correction applied to quadratic function with positive definite Hessian H.then*

$$B^k\gamma^j = \delta^j \qquad j = 0, \ldots, k-1$$

***proof**:*

Note that $H\delta^k = \gamma^k \forall k$, for $k = 1$, $B^1\gamma^o = \delta^o$(quasi-Newton condition).

Suppose $B^k\gamma^j = \delta^j$, $j = 0 \ldots, k-1$ using rank-one correction and using $j = 0, \ldots, k-1$

$$B^{k+1} = B^k + \frac{(\delta^k - B^k\gamma^k)(\delta^k - B^k\gamma^k)^\top}{(\delta^k - B^k\gamma^k)^\top\gamma^k}$$
$$B^{k+1}\gamma^j = (B^k + \frac{(\delta^k - B^k\gamma^k)(\delta^k - B^k\gamma^k)^\top}{(\delta^k - B^k\gamma^k)^\top\gamma^k})\gamma^j$$
$$= B^k\gamma^j + \frac{(\delta^k - B^k\gamma^k)(\delta^k\gamma^j - \gamma^k B^k\gamma^j)^\top}{(\delta^k - B^k\gamma^k)^\top\gamma^k}$$
$$= B^k\gamma^j + \frac{(\delta^k - B^k\gamma^k)(\delta^k B^k\delta^j - \delta^k B^k\delta^j)^\top}{(\delta^k - B^k\gamma^k)^\top\gamma^k}$$
$$= B^k\gamma^j + \frac{(\delta^k - B^k\gamma^k)(\delta^k H^k\delta^j - \delta^k H^k\delta^j)^\top}{(\delta^k - B^k\gamma^k)^\top\gamma^k}$$
$$= B^k\gamma^j = \delta^j \quad \forall j = 0, \ldots, k-1$$
$$\textit{Also } B^{k+1}\gamma^k = \delta^k \quad \textit{therefore} \quad B^{k+1}\gamma^j = \delta^j \quad \forall j = 0, \ldots, k$$

Finally

Consider the quadratic problem $\min_x \frac{1}{2}x^\top Hx + c^\top x$ where H is a symmetric positive definite matrix. In Newton method choose any initial point x^o and use the Newton direction ${d_N}^o = -H^{-1}g^o$ therefore $x_1 = x^*$ this means, reach the optimal solution at one iteration if start from any initial point while in Quasi-Newton method suppose apply in same problem quasi-Newton method (rank-one correction) to solve this problem at every iteration k

- B^{k+1} is symmetric positive definite
- B^{k+1} is obtained from B^k,x^k,x^{k+1},g^k and g^{k+1}
- B^{k+1} satisfies Quasi-Newton condition $B^{k+1}\gamma^k = \delta^k$

  Note that:

- $$\begin{cases} g^k = Hx^k + c \\ g^{k+1} = Hx^{k+1} + c \end{cases} \Rightarrow g^{k+1} - g^k = (x^{k+1} - x^k)H \Rightarrow \gamma^k = H\delta^k$$

Suppose in addition to Quasi-Newton condition at every iteration

- $k = 0$, satisfy the Quasi-Newton condition$B^1\gamma^o = \delta^o$

- $k = 1$, satisfy the Quasi-Newton condition $B^2\gamma^1 = \delta^1$ and satisfy the extra condition $B^2\gamma^o = \delta^o$
- $k = 2$, satisfy the Quasi-Newton condition $B^3\gamma^2 = \delta^2$ and satisfy the extra condition $B^3\gamma^1 = \delta^1$and $B^3\gamma^o = \delta^o$
- $k = n - 1$, $B^n\gamma^{n-1} = \delta^n$ and other condition satisfy $B^n\gamma^{n-1} = \delta^n$, ... , $B^n\gamma^o = \delta^o$ if this happens then, say that B satisfy "Hereditary Property"

  if Hereditary Property holds then $k = n - 1$, $B^n\gamma^{n-1} = \delta^n$ and other condition satisfy $B^n\gamma^{n-1} = \delta^n$, ... , $B^n\gamma^o = \delta^o$ suppose $(\delta^k - B^k\gamma^k)^\top\gamma^k \neq 0$ in the rank-one correction $B^n(\gamma^{n-1}|\dots|\gamma^1|\gamma^o) = (\delta^{n-1}|\dots|\delta^1|\delta^o)$ using fact $\gamma^k = H\gamma^k$ for every k, have $B^nH(\delta^{n-1}|\dots|\delta^1|\delta^o) = (\delta^{n-1}|\dots|\delta^1|\delta^o)$(replace each γ by $H\delta^k$ and take H commend from each term) if $\delta^o, \dots, \delta^{n-1}$ are linearly independent then $B^nH = I \Rightarrow B^n = H^{-1}$ the end of n iteration, have the matrix B^n to be H^{-1} which is inverse of Hessian

No guaranty in symmetric rank one update, that new matrix B^{k+1} is positive definite even though $B^k$is positive definite. So, decided to use the symmetric two correction update

Remark. • A simple and elegant way to use the information gathered during two consecutive iteration to update B^k

- $B^{k+1} = \delta^k - B^k\delta^k$, where B^k is positive definite , if $B^{k+1} > 0$ then B^{k+1} is positive definite which cannot be guaranteed at every k
- if $(\delta - B^k\gamma)^\top\gamma^k \approx 0$ then numerical difficulties.

following update method have received wide acceptance :

(a) Davidon-Fletcher-Powell(DFP method)

(b) Broyden-Fletcher-Goldfarb-shanno (BFGS method)

Theorem 4.9.1. *Consider the quadratic problem* $\min_x \frac{1}{2}x^\top Hx + c^\top x$ *where* H *is a symmetric positive definite matrix, if the rank-one correction is will defined and*

$\delta^0, \ldots, \delta^{n-1}$ *are linearly independent, if the rank one correction method applied to* $\min f(x)$ *terminates in at most* $n+1$ *iteration with* $B^n = H^{-1}$ *[55].*

Example 4.9.1. By use the quasi-Newton method (rank one correction) solve the problem $\min f(x) = 4x^2 + y^2 - 2xy$, where $x^* = (0,0)^\top$, $H = \begin{pmatrix} 8 & -2 \\ -2 & 2 \end{pmatrix}$ and $H^{-1} = \begin{pmatrix} 0.1667 & 0.1667 \\ 0.1667 & 0.6667 \end{pmatrix}$

solution:

start with $B^0 = I$ and at every iteration will update the B , terminates in at most $n+1$ iteration with $B^n = H^{-1}$, this show up clearly in iteration 2 and the order of problem is two but will reach the optimal solution in 3 iteration.

Table 4.2: Rank one update

k	x	y	B^k	$\|g^k\|$.
0	-2	-2	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	12
1	0	-2	$\begin{pmatrix} 0.1833 & 0.2333 \\ 0.2333 & 0.9333 \end{pmatrix}$	5.65
2	0.1538	0.1536	$\begin{pmatrix} 0.1667 & 0.1667 \\ 0.1667 & 0.6667 \end{pmatrix}$	0.92
3	0	0		0

4.9.2 Rank Two Correction

Given that B^k is symmetric and positive definite matrix let $B^{k+1} = B^k + \alpha\mu\mu^k + \beta vv^\top$

$$
\begin{aligned}
(B^k + \alpha\mu\mu^k + \beta vv^\top)\gamma^k &= \delta^k \\
B^k\gamma^k + \alpha\mu\mu^k\gamma^k + \beta vv^\top\gamma^k &= \delta^k \\
\alpha\mu\mu^k\gamma^k + \beta vv^\top\gamma^k &= \delta^k - B^k\gamma^k \\
\text{let} \quad \alpha\mu^k\gamma^k = 1 \quad &\text{and} \quad \beta v^\top\gamma^k = 1 \\
\alpha^{-1} = {\delta^k}^\top\gamma^k, \beta^{-1} &= -{\gamma^k}^\top B^k\gamma^k
\end{aligned}
$$

Therefore

$$B^{k+1} = B^k + \frac{\delta^k{\delta^k}^\top}{{\delta^k}^\top\gamma^k} - \frac{B^k\gamma^k{\gamma^k}^\top B^K}{{\gamma^k}^\top B^k\gamma^k}$$

This method of rank-two correction is calling DFP method.

DFP update

The Davidon Fletcher and Powell (DFP) method (Fletcher 1970) is one of the most powerful iterative method which is known for minimizing the unconstrained optimization problem of n variables [3]. This iterative method was derived by W. C. Davidon in 1950 who referred this method as the variable metric method and later modified and clarified by R. Fletcher and M. J. D. Powell in 1963 [54]. It was the first Quasi-Newton method to generalize the Secant method for a multidimensional problem. The DFP algorithm was the dominating algorithm for more then a decade and it was found to work well in practice [2]. The DFP algorithm is superior than the rank one correction algorithm because it preserves the positive definiteness of B^k , which is given as

$$B^{k+1}_{DFP} = B^k + \frac{\delta^k{\delta^k}^\top}{{\delta^k}^\top\gamma^k} - \frac{B^k\gamma^k{\gamma^k}^\top B^k}{{\gamma^k}^\top B^k\gamma^k} \qquad \text{DFP method}$$

Shall prove B_{DFP}^{k+1} positive definite, since B_{DFP}^{k+1} is a symmetric matrix let $x \neq 0$, $\gamma^k \neq 0$, $\delta \neq 0$

$$x^\top B_{DFP}^{k+1} x = x^\top B^k x + \frac{(x\delta^{k\top})^2}{\delta^{k\top}\gamma^k} - \frac{(B^k\gamma^k x^\top)^2}{\gamma^{k\top} B^k \gamma^k}$$

since B^k is symmetric then can written $B^k = B^{k\frac{1}{2}} B^{k\frac{1}{2}}$ where $B^{k\frac{1}{2}}$ is symmetric and positive definite,
let $a = B^{k\frac{1}{2}} x, \quad b = B^{k\frac{1}{2}}\gamma^k$

$$\Rightarrow x^\top B_{DFP}^{k+1} x = \frac{(a^\top a)(b^\top b) - (a^\top b)^2}{b^\top b} + \frac{(\delta^{k^k} x)^2}{\delta^{k\top}\gamma^k}$$

- $(a^\top a)(b^\top b) \geq (a^\top b)^2$ (Cauch-schwartz inequality)
- $b^\top b = \gamma^{k\top} B^k \gamma^k > 0 (B^k$is positive definite matrix)

  Note that $x^{k+1} = x^k - \alpha^k B^k g^k \Rightarrow \delta^k = -\alpha^k B^k g^k$ suppose that x^{k+1} is obtained using exact line search $\therefore g^{k+1\top}\delta^k = 0$
- $\delta^{k\top}\gamma^k = \delta^{k\top}(g^{k+1} - g^k) = -g^k\delta^k = \alpha^k g^{k\top} B^k g^k > 0$ therefore $B_{DFP}^{k+1} \succeq 0$ or B_{DFP}^{k+1} is positive semi-definite.

Now show that B_{DFP}^{k+1} is positive definite that is $x^\top B_{DFP}^{k+1} x > 0$, $x \neq 0$, have already shown that $\delta^{k\top}\gamma^k > 0$ suppose $x^\top B_{DFP}^{k+1} x = 0$, $x \neq 0$ therefore either $(a^\top a)(b^\top b) = (a^\top b)^2$ or $(\delta^{k\top} x)^2 = 0$
If $(a^\top a)(b^\top b) = (a^\top b)^2 \Rightarrow a = \mu b \therefore x = \mu\gamma^k$
If $(\delta^{k\top} x)^2 = 0 \Rightarrow \mu\delta^{k\top}\gamma^k = 0$ this true only when $\delta^{k\top}\gamma^k = 0$ but $(\delta^{k\top}\gamma^k > 0)$ then get contradiction

$$\therefore x^\top B_{DFP}^{k+1} x > 0, x \neq 0$$

Algorithm 7 : Quasi-Newton Algorithm(DFP)

(a) Initialize x^o , ϵ and symmetric positive definite $B^o = I$,set $k := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. Find a descent direction $d^k = -B^k g^k$

ii. find $\alpha^k (> 0)$ along d^k such that

- $f(x^k + \alpha^k d^k) < f(x^k)$
- α^k satisfies Armijo-wolfe conditions

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. Find B^{k+1} using DFP method

v. k:=k+1 end while

output $x^* = x^k$ a stationary of $f(x)$

BFGS method

It is possible to update the Hessian rather then its inverse at every iteration. Such popular update formula was suggested independently in 1970 by Broyden, Fletcher, Goldfarb, and Shanno that had proven to be most effective in many applications [2]. Will see its one application in the form of case study to solve the problem of heat conduction. This method is known as the BFGS algorithm. The BFGS method replaces the conjugate gradient technique for solving the non-linear functions. In order to derive the BFGS update, use the concept of duality, or complementarity, [40] . Recall the updating formulas for the approximation of the inverse of the Hessian matrix which were based on the following equations:

Recall Newton direction, ${d_N}^k = -(H^{k^-}1)g^k$, and the Quasi-Newton direction ${d_Q}N^k = -B^k g^k$ and the quasi-Newton condition

$$B^{k+1}\gamma^k = \delta^k \Rightarrow \gamma^k = (B^{k+1})^{-1}\delta^k$$

Let $G^{k+1} = (B^{k+1})^{-1}$ approximate H^{k+1} therefore, get dual formulae

$$B^{k+1}\gamma^k = \delta^k \quad \text{if } B^{k+1}\text{postive definite then it is invertable matrix}$$

$$G^{k+1}\delta^k = \gamma^k$$

$$B^{k+1}_{DFP} = B^k + \frac{\delta^k \delta^{k\top}}{\delta^{k\top}\gamma^k} - \frac{B^k\gamma^k\gamma^{k\top}B^K}{\gamma^{k\top}B^k\gamma^k} \qquad \text{DFP method}$$

$$G^{k+1}_{BFGS} = G^k + \frac{\gamma^k\gamma^{k\top}}{\delta^{k\top}\gamma^k} - \frac{G^k\delta^k\delta^{k\top}G^K}{\delta^{k\top}G^k\delta^k} \qquad \text{BFGS method}$$

Update formula for matrix B using δ^k, γ^k, also can get update for matrix G, can get B^{k+1}_{BFGS} from G^{k+1}_{BFGS} by using the condition

$$B^{k+1}_{BFGS}G^{k+1}_{BFGS} = I$$

Therefore:

$$B^{k+1}_{BFGS} = B + (1 + \frac{\gamma^\top B\gamma}{\delta^\top\gamma^\top})\frac{\delta^\top\delta}{\gamma^\top\delta} - (\frac{\delta\gamma^\top B + B\gamma\delta^\top}{\delta^\top\gamma})$$

Algorithm 8 : Quasi-Newton Algorithm(BFGS)

(a) Initialize x^o , ϵ and symmetric positive definite $B^o = I$,set $k := 0$

(b) While $\left\|g(x^k)\right\| \geq \epsilon$

i. Find a descent direction $d^k = -B^k g^k$

ii. Find $\alpha^k (> 0)$ along d^k such that

- $f(x^k + \alpha^k d^k) < f(x^k)$
- α^k satisfies Armijo-wolfe conditions

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. Find B^{k+1} using BFGS method

v. k:=k+1 end while

output $x^* = x^k$ a stationary of $f(x)$

Broyden Family

Have studied symmetric two rank correction updates, one is the DFB formula and another one is the BFGS formula. these two formulae can be combined to get an update formula which belongs to what is called Broyden family

$$B^{k+1}(\phi) = \phi B^{k+1}_{BFGS} + (1 - \phi) B^{k+1}_{DFP}$$

where $\phi \in [0, 1]$ when $\phi = 0$, have is the DFB update formula and when $\phi = 1$ have the BFGS update formula. Broyden family consists of DFP formula as well as BFGS formula, if choose $\phi \in [0, 1]$ then B^{k+1} will be symmetric and positive definite. Because know that both BFGS and and DFB update formulas for B are symmetric and positive definite, this gives a family of update rules for the matrix B.

Algorithm 8 : Quasi-Newton Algorithm(Broyden Family)

(a) Initialize x^o, ϵ and symmetric positive definite $B^o = I$, set $k := 0$, $\phi \in [0, 1]$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. Find a descent direction $d^k = -B^k(\phi)g^k$

ii. find $\alpha^k (> 0)$ along d^k such that

- $f(x^k + \alpha^k d^k) < f(x^k)$
- α^k satisfies Armijo-wolfe conditions

iii. $x^{k+1} = x^k + \alpha^k d^k$

iv. $B^{k+1}(\phi) = \phi B^{k+1}_{BFGS} + (1 - \phi)B^{k+1}_{DFP}$

v. k:=k+1 end while

output $x^* = x^k$ a stationary of $f(x)$

The $\mathtt{L-BFGS}$ Method Is Specially

Well suited for optimization problems with many variables. Rather than the H_k^{-1} $\mathtt{L-BFGS}$ partake a background marked by the past m updates of the position x and slope $\nabla f(\mathrm{x})$ Limited-memory BFGS (L-BFGS or LM-BFGS) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) using a limited amount of computer memory [7]. It is a popular algorithm for parameter estimation in machine learning. The algorithm's target problem is to minimize $(\mathbf{f}(x))$ over unconstrained values of the real-vector (x) where f is a differentiable scalar function.

Some advantages of the BFGS method

Like the original BFGS, L-BFGS uses an estimate of the inverse Hessian matrix to steer its search through variable space, but where BFGS stores a dense $n \times n$ approximation to the inverse Hessian (n being the number of variables in the problem), L-BFGS stores only

a few vectors that represent the approximation implicitly [80]. Due to its resulting linear memory requirement, the L-BFGS method is particularly well suited for optimization problems with many variables. Instead of the inverse Hessian H^k, L-BFGS maintains a history of the past m updates of the position x and gradient $f(x)$, where generally the history size m can be small (often $m < 10$, $m < 10$). These updates are used to implicitly do operations requiring the H^k-vector product.

4.10 Coordinate Descent Method

Consider the problem $\min_x f(x)$ where $f : R^n \to R$, where $f \in C^1$ is the first order continuously differential for every coordinate variable x_i, $i = 1, \ldots, n$ minimize $f(x)$ with respect to x_i, keeping the other coordinate variable x_j, $j \neq i$ constant [32].

Algorithm 2 : Coordinate Descent Method

(a) Initialize x^o,ϵ; set k:=0

(b) while $\left\|g^k\right\| > \epsilon$

i. for $i = 1, \ldots, n$

ii. ${x_i}^{new} = arg\min_{x_i} f(x)$

iii. $x_i = {x_i}^{new}$ end while

output: $x^* = x^k$ a stationary point of $f(x)$

.

globally convergent method if a search along any coordinate direction yields a unique minimum point.

Example 4.10.1. Consider the problem the $\min_x f(x) = 4{x_1}^2 + x_2^2 - 2x_1x_2$, use coordinate descent method with exact line search to solve this problem $x^o = (-1, -1)^\top$, let $d^o = (1, 0)^\top$ and $x^1 = x^o + \alpha^o d^o$ where $\alpha^o = arg\min_\alpha \phi(\alpha)$

$$\phi_o(\alpha) = f(x^0 + \alpha^0 d^o) : \alpha^0 (> 0)$$

$$\phi_o(\alpha) = f\begin{pmatrix} x^o + \alpha^0 {d_1}^o \\ \\ x^o + \alpha^0 {d_2}^o \end{pmatrix} = f\begin{pmatrix} -1 + \alpha^0(1) \\ \\ -1 + \alpha^0(0) \end{pmatrix} = f\begin{pmatrix} -1 + \alpha^0 \\ \\ -1 \end{pmatrix}$$

$= 4(\alpha - 1)^2 + 1 - 2(-1)(-1 + \alpha)$

$=4(\alpha - 1)^2 + 1 + 2(\alpha - 1)$

$\phi_o{}'(\alpha) = 0 \Rightarrow 8(\alpha - 1) + 2 = 0 \Rightarrow 4(\alpha - 1) + 1 = 0 \Rightarrow \alpha - 1 = \frac{-1}{4} \Rightarrow = \frac{3}{4}$

$\therefore x^1 = (-1, -1)^\top + \frac{3}{4}(1, 0)^\top = (-\frac{1}{4}, -1)^\top \Rightarrow d^1 = (0, 1)^\top; x^2 = x^1 + \alpha^1 d^1$

$x^2 = x^1 + \alpha^1 d^1, \alpha^1 = arg \min_\alpha \phi_1(\alpha)$

$$\phi_o(x^2) = f\begin{pmatrix} \frac{-1}{4} \\ \\ -1 + \alpha \end{pmatrix} = (\alpha - 1)^2 + \frac{\alpha - 1}{2} + \frac{1}{4}$$

$\phi_1{}' = 0 \Rightarrow \alpha^1 = \frac{3}{4}$

$x^2 = x^1 + \alpha^1 d^1 = (-\frac{1}{4}, -\frac{1}{4}) \neq x^*$ So need some more iteration to reach x^*, the Hessian matrix denote by

$$H = \begin{pmatrix} 8 & -2 \\ \\ -2 & 2 \end{pmatrix}$$

since matrix not diagonal and x_1 , x_2 are not separable, then could not be attained in two steps using coordinate descent method (if x^o is not on one of the principal axes of the elliptical contours)

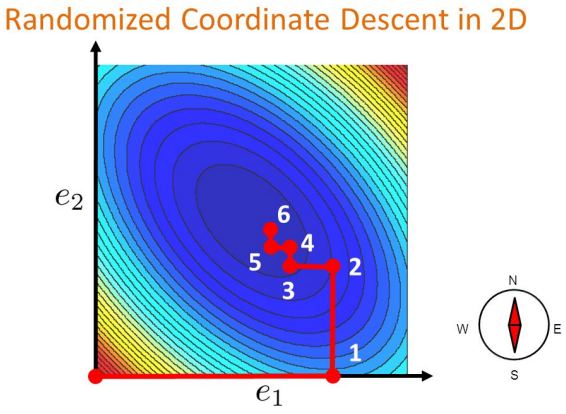


Figure 4.6: coordinate descent method

Example 4.10.2. Consider the problem the $\min_x f(x) = 4{x_1}^2 + {x_2}^2$, use coordinate descent method with exact line search to solve this problem $x^o = (-1,-1)^\top$, let $d^o = (1,0)^\top$ and $x^1 = x^o + \alpha^o d^o$ where $\alpha^o = arg\min_\alpha \phi_o(\alpha) = f(x^O + \alpha d^o) : \alpha(>0)$

$$\phi_o(\alpha) = f\begin{pmatrix} x^o + \alpha {d_1}^o \\ \\ x^o + \alpha {d_2}^o \end{pmatrix} = f\begin{pmatrix} -1+\alpha(1) \\ \\ -1+\alpha(0) \end{pmatrix} = f\begin{pmatrix} -1+\alpha \\ \\ -1 \end{pmatrix} = f\begin{pmatrix} -1+\alpha \\ \\ -1 \end{pmatrix} =$$

$4(\alpha-1)^2 + 1 = 4(\alpha-)^2 + 1 \Rightarrow \phi_o{}'(\alpha) = 0 \Rightarrow 8(\alpha-1) = 0 \Rightarrow 8(\alpha-1) = 0$

$\alpha - 1 = 0$

$\therefore x^1 = (1,-1) + (-1,0)^\top = (0,1)$

$\Rightarrow d^1 = (0,1)^\top; x^2 = x^1 + \alpha^1 d^1$

$x^2 = x^1 + \alpha^1 d^1, \alpha^1 = arg\min_\alpha \phi_1(\alpha)$

$x^2 = x^1 + \alpha^1 d^1 = (0,0) = x^*$ since matrix diagonal and x_1 , x_2 are separable, then could be attained in two steps using coordinate descent method$H = \begin{pmatrix} 8 & 0 \\ \\ 0 & 2 \end{pmatrix}$

Definition 4.10.1. Let $H \in R^{n\times n}$ be a symmetric matrix, the vectors $\{d^o, \ldots, d^{n-1}\}$ are said to be $H-$ conjugate if the are linearly independent and ${d^i}^\top H d^j = 0, \forall i \neq j$ [32]

Theorem 4.10.1. *A convex quadratic function can be minimized in at most n-steps, provided search along conjugate direction of the Hessian matrix [6].*

Proof. consider the problem $\min_x f(x) = \frac{1}{2}x^\top Hx + c^\top x$ where H is a symmetric positive definite matrix (H is not diagonal matrix), let$\{d^o, \ldots, d^{n-1}\}$ be a set of linearly independent direction and $x^o \in R^n$ any $x \in R^n$ can be represented as $x = x^o + \sum_{i=0}^{n-1} \alpha^i d^i$ then rewrite problem as given $\{d^o, \ldots, d^{n-1}\}$ and $x^o \in R^n$, the given problem is to minimize $\psi(\alpha)$ defined as

$$\frac{1}{2}(x^o + \sum_{i=0}^{n-1} \alpha^i d^i)^\top H(x^o + \sum_{i=0}^{n-1} \alpha^i d^i) + c^\top (x^o + \sum_{i=0}^{n-1} \alpha^i d^i)^\top$$

Let us define $D = (d^o|d^1|\ldots|d^{n-1})$ be a matrix denote by $d^o|d^1|\ldots|d^{n-1}$ and $\alpha = (\alpha^o, \ldots, \alpha_{n-1})$ $\psi(\alpha)$ in this compact notation

$$\frac{1}{2}\alpha^\top D^\top HD\alpha + (Hx^o + c)^\top D_\alpha + \frac{1}{2}x^{o\top} Hx^o + c^\top x^o$$

Let $K = \frac{1}{2}x^{o\top} Hx^o + c^\top x^o$ where K is constant H^o, x^o and $c^\top$ are constant, then when $\min \psi(\alpha)$ can ignore K then

$$\min \psi(\alpha) = \frac{1}{2}\alpha^\top DHD\alpha + (Hx^o + c)^\top D\alpha$$

the Hessian matrix in this quadratic function.

$$Q = D^\top HD = \begin{pmatrix} d^{o\top} Hd^o & d^{o\top} Hd^1 & \cdots & d^{o\top} Hd^{n-1} \\ d^{1\top} Hd^o & d^{1\top} Hd^1 & \cdots & d^{o\top} Hd^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ d^{o\top} Hd^o & d^{n-1\top} Hd^1 & \cdots & d^{n-1\top} Hd^{n-1} \end{pmatrix}$$

And the diagonal of $Q = \left({d^o}^\top H d^o, \ldots, {d^{n-1}}^\top H d^{n-1}\right)$. Decides to make the Hessian matrix Q is diagonal matrix, the way to do that is that to make all this half diagonal entries in this matrix equal to zero, in other words when ever $i \neq j$ then ${d^i}^\top H d = 0, \forall i \neq j$

$$Q = D^\top H D = \begin{pmatrix} {d^o}^\top H d^o & 0 & \cdots & 0 \\ 0 & {d^1}^\top H d^1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & {d^{n-1}}^\top H d^{n-1} \end{pmatrix}$$

this matrix it is easy to inveritable

Therefore

$${Q^{-1}}_{ij} = \begin{cases} \dfrac{1}{d^\top H d} & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases}$$

$$\min \psi(\alpha) = \frac{1}{2}\alpha^\top D H D \alpha + (H x^o + c)^\top D \alpha + c^\top (x^o + \sum_i \alpha^i d^i)$$

since $c^\top (x^o + \sum_i \alpha^i d^i$ constant and $\psi(\alpha)$ is separable in terms of $\alpha^o, \ldots, \alpha^{n-1}$ because this separable in terms it is easy to optimize this objective function individually in terms of α, will optimize the problem with respect to α and in the α space you can think of it is a coordinate descent method, take α^1 at a time and optimize with respect α

$$\frac{\partial \psi}{\partial \alpha^i} = 0 \Rightarrow \alpha^{i*} = -\frac{d^i (H x^o + c)}{{d^i}^\top H d^i}$$

Therefore

$$x^* = x^o + \sum_{i=0}^{n-1} \alpha^{i*} d^i$$

$$x^* = x^o + (\sum_{i=0}^{n-1} - \frac{d^i(Hx^o + c)}{{d^i}^\top H d^i})d^i$$

□

Example 4.10.3. Consider the problem the $\min_x f(x) = 4{x_1}^2 + x_2^2 - 2x_1x_2$, use coordinate descent method with exact line search to solve this problem $x^o = (-1,-1)^\top$, let $d^o = (1,0)^\top$ and $x^1 = x^o + \alpha^o d^o$ where $\alpha^o = arg\min_\alpha \phi(\alpha)$

$$\phi_o(\alpha) = f(x^0 + \alpha^0 d^o) : \alpha^0 (> 0)$$

To solve this example using a different technique is to choose the new direction so that it is $H-$conjugate to the previous one, let us choose a non-zero direction d^1 such that ${d^1}^\top H d^o = 0$. Let $d^1 = (a,b)^\top \therefore \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 8 & -2 \\ -2 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0, \Rightarrow 8a - 2b = 0 \Rightarrow a = \frac{b}{4}$.
In this equation, have two variables and one equation for that values a, b a multiple, for example if choose $a = 1 \Rightarrow b = 4$ $\therefore \alpha^1 = \text{argmin}_\alpha \phi_1(\alpha) = f \begin{pmatrix} \alpha - \frac{1}{4} \\ 4\alpha - 1 \end{pmatrix} frac34(4\alpha - 1)^2 \Rightarrow \alpha^1 = \frac{1}{4} x^2 = x^1 + \alpha^1 d^1 = (0,0)^\top = x^*$

Given H, does a set of H-conjugate vectors exist? If yes, how to get a set of such vectors?

(a) Let H$\in R^{n\times n}$ be a symmetric matrix then H has n mutually orthogonal eigenvectors, let v_1 and v_1 be two orthogonal eigenvectors of H; ${v_1}^\top v_2 = 0$; $Hv_1 = \lambda_1 v_1 \Rightarrow {v_2}^\top H v_1 = \lambda_1 v_1 {v_2}^\top$ then ${v_2}^\top H v_1 = 0 \Rightarrow v_1$ and v_2 are $H-$conjugate, can say n orthogonal eigenvectors of H are $H-$conjugate. In general any symmetric matrix always have $H-$conjugate vectors.

(b) Let $d^o, \ldots, d^{n-1}$ be non-zero directions such that are $H-$conjugate

$\left({d^i}^\top H d^j = 0, i \neq j\right)$, if these direction are linear independent then are spanned space which is subspace of R^n, let take $\lambda^i \in R$ such that

$$\sum_{i=0}^{n-1} \lambda^i d^i = 0 \Rightarrow \sum_{i=0}^{n-1} \lambda^i d^i H d^j = 0$$

For every $j = 0, \ldots, n-1$ then $\lambda^i d^i H d^j = 0$, since $d^o, \ldots, d^{n-1}$ are non-zero direction then $d^i H d^j > 0 \Rightarrow \lambda^j = 0 \Rightarrow d^o, \ldots, d^{n-1}$ are liner independent.

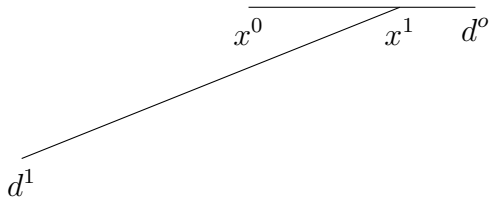
(c) Consider the problem, $\min_{x \in R^2} \frac{1}{2} x^\top H x + c^\top x$, H symmetric and positive definite matrix, let x^* be the solution $\because Hx^* = -c\,[\nabla f(x) = 0]$.

Let x^o be any initial point $g^o = Hx^o + c$, let d^o be some direction $(d^o \neq 0)$, x^1 is found by doing exact line search along d^o,${g^1}^\top d = 0$ since $g^1 = Hx^1 + c$ then $\left(Hx^1 + c\right) d^o = 0$ and $c = Hx^*$

$$(x^* - x^1)^\top H d^o = (Hx^* - Hx^*)\, d^o = (c - Hx^1)^\top d^o = {g^1}^\top = 0$$

therefore the direction $(x^* - x^1)$ is $H - conjugate$ to d^o

4.11 Spanned property



Suppose start with x^o and along direction d^o applied the exact line search then get x^1 and min function at x^1, from x^1 along vector d^1(direction) applied the exact line search getting x^2 and min function at x^2 (which in two dimension space) and d^o,d^1 are $H - conjocate$ therefore are linear independent. So are span R^2 similarly in the same way, at direction d^3 applied exact line search and min function at x^2 when reach x^n, then find the solution.

Theorem 4.11.1. *Expanding Subspace Theorem [62]*
Consider the problem $\min_x f(x) = \frac{1}{2}x^\top Hx + c^\top x$, *H symmetric positive definite matrix. Let* $d^o, d^1, ..., d^{n-1}$ *be* $H-$*conjugate therefore are linearly independent, let* B^k *denote the subspace of* R^n *spanned by* $d^o, \ldots, d^{k-1}$ *clearly* $B^k \subset B^{k+1}$ *(because* B^{k+1} *will have a extra vector* d^k*) which independent of* $d^o, \ldots, d^{k-1}$.
Let $x^o \in R^n$ *be any arbitrary point and let* $x^{k+1} = x^k + \alpha^k d^k$ *where* α^k *is obtained by doing exact line search* $\alpha^k = arg\min_\alpha f(x^k + \alpha d^k)$. *Then, for all* $k = 0, \ldots, n-1$

- ${g^k}^\top d^j = 0, j = 0, \ldots, k$
- ${g^k}^\top d^k = {g^o}^\top d^k$
- $$\begin{cases} x^k = \text{argmin}_x f(x) \\ x \in x^o + B^k \end{cases}$$

  Given a set of n-directions $d^o, \ldots, d^{n-1}$ *which are* $H-$ *conjugate at* $x^o \in R^n$ *it is easy to determine* α^{j*} *for all* $i = 0, \ldots, n-1$

$$\alpha^{j*} = -\frac{d^{i*}(Hx^o + c)}{{d^i}^\top H d^i}$$

and get $x^* = x^o + \sum_{i=0}^{n-1} \alpha^{i*} d^i$

Proof. Since at k^{th}, $\alpha^k = \text{argmin}_{x \in R} f(x^k + \alpha d^k)$

$$\nabla f(x^k + \alpha^k d^k)^\top d^k = 0$$

$$\nabla f(x^{k+1})^\top d^k = 0 \Rightarrow {g^{k+1}}^\top d^k = 0, \forall k = 0, \ldots, n-1$$

$$\therefore x^k = x^j + \sum_{i=j}^{k-1} \alpha^i d^i \text{(multipled the qountity by H and adding constant c)}$$

$$g^k d^{j-1} = g^i d^{j-1} + \sum_{i=j}^{k-1} \alpha^i d^i H d^{j-1}$$

since $g^{k+1}d^k = 0$ and $(d^0, \ldots, d^{k-1})$ are $H-$conject then $d^i H d^{j-1} = 0$ so, getting $g^k d^{j-1} = 0$, $\forall j = 0, \ldots, k-1$

Can say in general ${g^k}^\top$ is perpendicular on the space that span by all d^i,$\forall j = 0, \ldots, k-1$ this means $g^k \perp B^k$ and $x^2 \in d^1$, $f(x^2) < f(x^1)$, $\forall x^1 \in d^1$. But there is no guarantee that x^2 is minimize in $x^o + B^2$ where B^2 is span by perpendicular d^o, d^1 will explain that x^2 will be minimize of $x^o + B^2$ note that for every $j = 0, \ldots, n-1$, $\alpha^j = \text{argmin}_\alpha f(x^j + \alpha d^j)$ $\therefore f(x^j + \alpha^j d^j) \leq f(x^j + \mu^j d^j)$, $\left[\mu^j \in R\right]$ in this step comparing x^1 with all point on d^o, and find x^1 is the minimal point. Since f quadratic function then can use the Taylor series.

$$f(x^j) + \alpha^j {g^j}^\top d^j + \frac{1}{2}{\alpha^j}^2 \left({d^i}^\top H d^j\right) \leq f(x^j) + \mu^j {g^j}^\top d^j + \frac{1}{2}{\mu^j}^2 \left({d^i}^\top H d^j\right)$$

need to show that

$f(x^k) \leq f(x)$, $\forall x \in x^o + B^k$, rewrite the equation (see equation (4.11.1)) in the form

$$f(x^o + \sum_{j=o}^{k-1} \alpha^j d^j) \leq f(x^o + \sum_{j=0}^{k-1} \mu^j d^j)$$

,$\mu^j \in R$, $\forall j$ for any $x \in x^o + B^k$ use $\mu^i \in R$ but x^k find by using α^j, then

$$f(x^o) + \sum_{j=0}^{k-1} \left(\alpha^j {g^o}^\top d^j + \frac{1}{2}{\alpha^j}^2 {d^j}^\top H d^j\right) \leq f(x^o) + \sum_{j=0}^{k-1} \left(\mu^j {g^o}^\top d^j + \frac{1}{2}{\mu^j}^2 {d^j}^\top H d^j\right)$$

Clam: ${g^j}^\top d^j = {g^o}^\top d^j$, $\forall j$

Consider $x^j = x^o + \sum_{i=0}^{j-1} \alpha^i d^i$ [multiplied by H both side and adding constant c] $\therefore Hx^i + c = Hx^o + c + \sum_{i=0}^{j-1} \alpha^i H d^i \Rightarrow g^j = g^o + \sum_{i=0}^{j-1} \alpha^i H d^i$ multiple both side by $d^j \Rightarrow$.

$$g^j d^j = g^o d^j + \sum_{i=0}^{j-1} \alpha^i {d^j}^\top H d^i \Rightarrow g^j d^j = g^o d^j, \forall j$$

Then get $\left(\alpha^j {g^o}^\top d^j + \frac{1}{2}{\alpha^j}^2 {d^j}^\top H d^j\right) \leq \left(\mu^j {g^o}^\top d^j + \frac{1}{2}{\mu^j}^2 {d^j}^\top H d^j\right)$ will add $f(x^o)$ to both side

$$f(x^o) + \left(\alpha^j {g^o}^\top d^j + \frac{1}{2}{\alpha^j}^2 {d^j}^\top H d^j\right) \leq f(x^o) + \left(\mu^j {g^o}^\top d^j + \frac{1}{2}{\mu^j}^2 {d^j}^\top H d^j\right)$$

$\therefore f\left(x^o + \sum_{j=0}^{k-1} \alpha^j d^j\right) \leq f\left(x^o + \sum_{j=0}^{k-1} \mu^j d^j\right)$,$\mu^j \in R$, $\forall j$
then get $f(x^k) \leq f(x)$, $\forall x \in (x^o + B^k)$.

If have H − conjugate direction $d^o, \ldots, d^{k-1}$, have to determine α^k,where
$\alpha^k = \text{argmin}_\alpha f(x^k + \alpha d^k)$
$x^* - x^o = \sum_{i=0}^{n-1} \alpha^i d^i \Rightarrow {d^k}^\top H(x^* - x^o) = {d^k}^\top H(\sum_{i=0}^{n-1} \alpha^i d^i)$
${d^k}^\top H(x^* - x^o) = (\sum_{i=0}^{n-1} \alpha^i {d^k}^\top H d^i)$
${d^k}^\top H(x^* - x^o) = (\alpha^k {d^k}^\top H d^k)$
$\alpha^k = \frac{{d^k}^\top H(x^* - x^o)}{{d^k}^\top H d^k}$
$\alpha^k = \frac{{d^k}^\top H(x^* - x^k + x^k - x^o)}{{d^k}^\top H d^k}$ α^k still dependent on x^* and x^o want to get to dependence α^k only on x^k.
$\frac{{d^k}^\top H(x^* - x^k) + (x^k - x^o){d^k}^\top H}{{d^k}^\top H d^k}$
$\frac{{d^k}^\top H(x^* - x^k)}{{d^k}^\top H d^k} + \frac{(x^k - x^o){d^k}^\top H}{{d^k}^\top H d^k}$
suppose that after k iterative step and obtaining k, $H-$conjugate directions,

$$x^k - x^o = \sum_{i=0}^{k-1} \alpha^i d^i$$

$$d^k{}^\top H(x^* - x^o) = (\sum_{i=0}^{n-1} \alpha^i d^k{}^\top H d^i)$$

then get

$$\alpha^k = \frac{d^k{}^\top H(x^* - x^k)}{d^k{}^\top H d^i} = \frac{d^k{}^\top (Hx^* - Hx^k)}{d^k{}^\top H d^i}$$
$$= \frac{d^k{}^\top (-c - Hx^k)}{d^\top{}^k H d^k} = \frac{-d^k{}^\top g^k}{d^\top{}^k H d^k}$$

Now shall prove $-g^o, -g^1, \ldots, -g^{n-1}$ are linearly independent set of vectors, use Gram-Schmidt procedure to determine the $H-$conjugate vectors $d^o, \ldots, d^{n-1}$, let $d^o = -g^o$, and $d^k = -g^k + \sum_{j=0}^{k-1} B^j d^j$, $k = 1, \ldots, n-1$,but know $d^o, \ldots, d^{n-1}$ to be $H-$conjugate vectors

$$d^i{}^\top H d^k = -d^i{}^\top H g^k + \sum_{j=0}^{k-1} -d^j{}^\top \beta^j d^i H \qquad (* d^i{}^\top H)$$

$$0 = -d^i{}^\top H g^k + d^j{}^\top \beta^j d^i H, i = 0, \ldots, k-1$$

$$\beta^i = \frac{g^k{}^\top H d^i}{d^j{}^\top H d^i}$$

$$\text{since } d^k = -g^k + \sum_{j=0}^{k-1} \beta^j d^j$$

$$d^k = -g^k + \sum_{j=0}^{k-1} \left(\frac{g^k{}^\top H d^i}{d^k{}^\top H d^i} \right) d^j, k = 1, \ldots, n-1$$

know that span $\{d^o, \ldots, d^{k-1}\}$ = span $\{-g^o, \ldots, g^{k-1}\}$ have already shown that $\{d^o, \ldots, d^{k-1}\}$ are $H-$ conjugate $\Rightarrow g^k \perp B^k$

$\therefore -g^k \perp$ span $\{d^o, \ldots, d^{k-1}\} \Rightarrow -g^k \perp$ span $\{-g^o, \ldots, -g^{k-1}\} \Rightarrow \{-g^o, \ldots, -g^{k-1}\}$ is linearly independ set of vectors

Consider $d^o = -g^o \Rightarrow d^k = -g^k + \sum_{j=0}^{k-1} \left(\frac{g^k{}^\top H d^j}{d^k{}^\top H d^j} \right) d^j$, $\forall k = 1, \ldots, n-1$ since $x^{j+1} = x^j + \alpha^j d^j$ and $g^{j+1} = g^j + \alpha^j H d^j \Rightarrow H d^j = \frac{(g^{j+1} - g^j)}{\alpha^j}$ then get after submitted value of

Hd^j(see equation (4.11.1))

$$d^k = -g^k + \sum_{j=0}^{k-1} \left(\frac{{g^k}^\top (g^{j+1} - g^j)}{{d^k}^\top (g^{j+1} - g^j)} \right) d^j$$
$$= -g^k + \left(\frac{{g^k}^\top g^k}{{d^{k-1}}^\top (g^k - g^{k-1})} \right) d^{k-1}$$

Due to exact line search ${g^k}^\top d^{k-1} = 0$

$d^{k-1} = -g^{k-1} + \beta^{k-2} d^{k-2}$ multiply $-g^{k-1}$

$d^{k-1}(-g^{k-1}) = -g^{k-1}(-g^{k-1}) + \beta^{k-2} d^{k-2}(-g^{k-1})$

$\therefore d^k = -g^k + (\frac{{g^k}^\top g^k}{{g^{k-1}}^\top g^{k-1}}) d^{k-1}$ □

Algorithm 8 :Conjugate Gradient Algorithm of Fletcher-Reeves Method

---

For quadratic function $\frac{1}{2} x^\top H x + c^\top x$,$H$ symmetric positive definite

(a) Initialize x^o , ϵ and $d^o = -g^o$,set $k := 0$

(b) while $\left\| g(x^k) \right\| \geq \epsilon$

i. $\alpha^k = -\frac{{g^k}^\top d^k}{{d^k}^\top H d^k}$

ii. $x^{k+1} = x^k + \alpha^k d^k$

iii. $g^{k+1} = H x^{k+1} + c$

iv. $\beta^k = \frac{{g^{k+1}}^\top g^{k+1}}{{g^k}^\top g^k}$

v. $d^{k+1} = -g^{k+1} + \beta^k d^k$

vi. $k := k + 1$ end while

output $x^* = x^k$ global minim of $f(x)$

Algorithm 8 :Conjugate Gradient Algorithm of Fletcher-Reeves Method

---

Extension to Non-quadratic function $f(x)$

(a) Initialize x^o , ϵ and $d^o = -g^o$,set $K := 0$

(b) while $\left\|g(x^k)\right\| \geq \epsilon$

i. $\alpha^k = \text{argmin}_{\alpha>0} f(x^k + \alpha d^k)$

ii. $x^{k+1} = x^k + \alpha^k d^k$

iii. compute g^{k+1}

iv. if $k < n - 1$

- $\beta^k = \dfrac{{g^{k+1}}^\top g^{k+1}}{{g^k}^\top g^k}$
- $d^{k+1} = -g^{k+1} + \beta^k d^k$
- $k := k + 1$

else

- $x^o = x^{k+1}$
- $d^o = -g^{k+1}$
- $k := 0$

end if

end while

output $x^* = x^k$ a stationary point of $f(x)$

There are different ways to determination β^k

(a) Fletcher-Reeves method ${\beta^k}_{FR} = \dfrac{{g^k}^\top g^k}{{g^{k-1}}^\top g^{k-1}}$

(b) Polak-Ribiere method ${\beta^k}_{PR} = \dfrac{{g^k}^\top (g^k - g^{k-1})}{{g^{k-1}}^\top g^{k-1}}$ If use the quadratic function with exact line search then ${g^k}^\top \perp g^{k-1}$ ${\beta^k}_{FR} = {\beta^k}_{PR}$ are same in this situation

(c) Hestenes-steifel method ${\beta^k}_{Hs} = \dfrac{{g^k}^\top (g^k - g^{k-1})}{({g^{k-1}}^\top g^{k-1}) d^{k-1}}$

4.12 Hager Zhang

The strategy produces a sequence of iterates $\{\mathtt{y}^{\mathtt{k}}\}$ such that $\mathtt{y}^{k} = x^{k} + \eth^{k} d^{k}$, where $\eth^{k} > 0$ is the step length computed how far to move along d^{k} such that

$$\mathtt{G}\left(\mathtt{y}_{\mathtt{k}}\right)^{\mathtt{T}}\left(\mathtt{x}^{\mathtt{k}} - \mathtt{y}^{\mathtt{k}}\right) > \mathtt{0}$$

Also, by monotonically of G, have

$$\mathtt{G}\left(\mathtt{y}^{\mathtt{k}}\right)^{\mathtt{T}}\left(\mathtt{x}^{*} - \mathtt{y}^{\mathtt{k}}\right) = \left(\mathtt{G}\left(\mathtt{z}^{\mathtt{k}}\right) - \mathtt{G}\left(\mathtt{x}^{*}\right)\right)^{\mathtt{T}}\left(\mathtt{x}^{*} - \mathtt{y}^{\mathtt{k}}\right) \leq \mathtt{0}$$

and

$$\mathtt{A}^{\mathtt{k}} = \{\mathtt{x} \in \mathbf{R}^{\mathtt{n}} \mid \mathtt{G}\left(\mathtt{z}_{\mathtt{k}}\right)^{\mathtt{r}}\left(\mathtt{x} - \mathtt{z}_{\mathtt{k}}\right) = \mathtt{0}\}$$

cautiously disengages the current emphasize $\mathtt{x}^{\mathtt{k}}$ from the arrangement x^{*} of the framework in (1). Solodov and Svaiter [77] utilized this reality to present the following emphasize as the projection of x^{k} onto the hyperplane $\mathtt{H}^{\mathtt{k}}$ [57, 59, 60]. Also, the following emphasize is figured by

$$\mathtt{x}^{\mathtt{k+1}} = \mathtt{x}^{\mathtt{k}} - \frac{\mathtt{G}\left(\mathtt{y}^{\mathtt{k}}\right)^{\mathtt{T}}\left(\mathtt{x}^{\mathtt{k}} - \mathtt{z}_{\mathtt{k}}\right)}{\left\|\mathtt{G}\left(\mathtt{z}_{\mathtt{k}}\right)\right\|^{2}}\mathtt{G}\left(\mathtt{y}^{\mathtt{k}}\right)$$

Employ the above technique to develop our method for solving the non-linear monotone system in . First, review he conjugate gradient method (CG) for unconstrained optimization problem

$$\min_{\mathtt{x} \in \mathtt{R}^{2}} \mathtt{f}(\mathtt{x})$$

where $\mathtt{f}$ is a smooth non-linear function whose gradient is available. The conjugate gradient (CG) method is ideal for comprehending due to its low memory requirement and strong global convergence properties [19]. The method generates a sequence of iterates recurrently by

$$\mathtt{x}^{\mathtt{0}} \in \mathbf{R}^{\mathtt{n}}, \quad \mathtt{x}^{\mathtt{k+1}} = \mathtt{x}_{\mathtt{k}} + \alpha^{\mathtt{k}}\mathtt{d}^{\mathtt{k}}, \quad \mathtt{k} \geq \mathtt{0}$$

where α^k is the step-length and d^k is the search direction determined by

$$\mathtt{d^k} = \begin{cases} -G\left(x^k\right), & \text{if } k = 0 \\ -G\left(x^k\right) + \beta^k d^{k-1}, & \text{if } k \geq 1 \end{cases}$$

where $G\left(x^k\right) = \nabla f\left(x^k\right)$ and β^k is a parameter, which characterizes the CG method.

Example 4.12.1. Consider the unconstrained optimization problem.

$$\min_{x \in R^2} f(x) = [1.5 - x_1(1 - x_2)]^2 + \left[2.625 - x_1(1 - {x_2}^3)\right]^2 + \left[2.25 - x_1(1 - {x_2}^2)\right]^2$$

will applying the different technique of line search by using the python programming to the same problem and will comparing the beaver of converge to the minimal point and explain by study result who is better technique. the date using to solve example are. the current point is $x = [1, -3]$and with the stop conditione$\epsilon = 1e - 6$ the fowling output on the python console

(a) Steepest Descent Method

Initial function value= 760.3906, approach to minimize from initial point $x = (1, -3)$ by using steepest descent method as shown by the table1 need to 57 iteration to reach the minimize $x^* = (3.025, 0.474)^\top$. the object value function in iteration one decreasing $f(x_1, x_2) = 11.263$ to $f(x_1, x_2) = 0.640$ as will as getting the small steps with using direction search $d^k = -g^k$. When go to the next iteration which $(0.157, -2.213)$ to $(2.254, 0.040)$ would see that there is a significant decreasing in the objective function value from $f(0.157, -2.213)$ to $f(2.254, 0.040)$ and again there is a significant decrease in the norm of g^k in the iteration progress you will see that in the end of 10^th iteration the function value is0.063 in the end of 20^{th} iteration the function value is very small, if use the spotted criteria of $\left\|g^k\right\| \leq \epsilon = 1e - 6$(see table 4.6).

(b) Newton Method

In newton method the direction search ${d^k}_N = -H^{-1}g^{<}0$ the direction depends on the the inverse of Hessian matrix at every iteration, the number of iteration until reach the minimize is 16 only less than the numbers of iteration in steepest descent method but not all the Hessian matrix positive definite as well as invertible so we need update the method .Note that in some iteration, the search direction is not a descent as the function value increases instead of monotonically decreasing. The method, however, converges to the minimum point(see table 4.3).

(c) The Quasi-Newton method(DFP update) and the quasi-Newton method(BFGS update), in this method will replace Hessian matrix by approximate matrix B^k and at every iteration will use rank two update and if comparison this method with Method Newton method, will reach the optimal in 7 iteration only (see table 4.4, 4.5)

(d) The Conjugate Gradient In this method depend on orthogonal eigenvectors of Hessian matrix, the table(see table 4.7) the behaver of converge to minimizer.

k	x_1	x_2	$f(x_1, x_2)$	$\parallel gradient \parallel$
1	0.875	-2.553	220.450	2200.599
2	0.772	-2.173	66.508	743.760
3	0.683	-1.859	23.624	249.387
4	0.586	-1.634	12.364	81.919
5	0.426	-1.578	10.264	25.159
6	0.169	-1.885	11.373	5.192
7	0.396	-0.942	10.163	11.355
8	0.961	-1.719	30.758	10.397
9	0.834	-1.482	13.413	106.087
10	0.616	-1.395	9.621	34.203
11	0.208	-1.686	11.211	8.789
12	0.911	1.206	22.326	10.929
13	0.383	1.070	16.207	43.067
14	0.052	1.007	15.400	12.654
15	0.001	1.000	15.391	1.523
16	0.000	1.000	15.391	0.015

Table 4.3: The Newton method

Through iteration 8,9 it appears to as clearly. No, guaranty that d^k descent direction.

k	x_1	x_2	$f(x_1, x_2)$	$\parallel gradient \parallel$
1	0.157	-2.213	11.263	2200.599
2	2.256	0.038	0.645	0.616
3	2.551	0.355	0.130	2.928
4	2.555	0.356	0.128	0.880
5	2.927	0.431	0.045	0.865
6	2.975	0.462	0.039	0.602
7	3.026	0.475	0.038	0.076
8	3.025	0.474	0.038	0.029

Table 4.4: The Quasi-Newton method(DFP update)

k	x_1	x_2	$f(x_1, x_2)$	$\parallel gradient \parallel$
1	0.157	-2.213	11.263	2200.599
2	2.256	0.038	0.645	0.616
3	2.551	0.355	0.130	2.928
4	2.555	0.356	0.128	0.880
5	3.014	0.466	0.039	0.865
6	3.022	0.473	0.038	0.210
7	3.025	0.474	0.038	0.003

Table 4.5: The Quasi-Newton method(BFGS update)

In comparison with the previous method(Newton Method), note that the approach to the optimal solution is an orderly approach, this means all direction are descent direction.

k	x_1	x_2	$f(x_k)$	$\parallel gradient \parallel$
1	0.157	-2.213	11.263	2200.599
2	2.254	0.040	0.640	0.616
3	2.252	0.230	0.350	2.889
4	2.487	0.232	0.218	1.125
5	2.486	0.309	0.156	1.561
6	2.618	0.311	0.118	0.566
7	2.618	0.353	0.096	1.028
8	2.705	0.354	0.081	0.349
9	2.704	0.381	0.071	0.743
10	2.767	0.382	0.063	0.238
11	2.767	0.400	0.058	0.565
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
53	3.019	0.472	0.038	0.014
54	3.020	0.472	0.038	0.003
55	3.020	0.472	0.038	0.012
56	3.021	0.472	0.038	0.002
57	3.021	0.473	7 0.038	0.010

Table 4.6: The iterations of Steepest descent method

In Steepest descent method fast convergence with slow steps.

k	x_1	x_2	$f(x_1, x_2)$	$\parallel gradient \parallel$
1	0.157	-2.213	11.263	2200.599
2	2.253	0.041	0.638	0.616
3	2.457	0.325	0.180	2.878
4	2.632	0.412	0.134	1.003
5	3.084	0.508	0.048	1.791
6	3.120	0.499	0.040	0.962
7	3.120	0.497	0.040	0.077
8	3.101	0.489	0.040	0.036
9	3.021	0.471	0.038	0.180
10	3.018	0.472	0.038	0.059
11	3.018	0.472	0.038	0.006

Table 4.7: Conjugate Gradient Method

In conjugate gradient method. Regular convergence without using the second derivative information.

CHAPTER 5

APPLICATION OF OPTIMIZATION PROBLEMS

Chapter numerical results. It is the last part of the study, and the chapter is summarized in three main parts:

Part One Constrained problem Portfolio will choose and it will solve by the Penalty method, Gurobi and regularization, the results were compared between them.

Part Two Implementation $NP - hard$ problem of order n, and do different line search it by L-BFGS)(BackTracking and (L-BFGS)(Hager Zhang) methods, compare the numerical outcomes by using python Language (version-3.7.8rc1) to check how closely these techniques approximate the exact solution. Also improved the theoretical convergence properties.

Part Three The modal of a multiple objective function used. A mathematical formula has been proposed to reduce the number of people infected with the Corona virus, as the objective function represents the rate of spread of the disease and the constraint are represented by three functions

i. Social distancing function.

ii. Adherence to health guidelines.

A mathematical formula will propose for a problem to transfer the Pfizer vaccine manufactured in the United States to the world

All results in the dissertation were calculated by language Python is a high-level programming language that is simple to write and read, easy to learn, uses an object-oriented programming style, is open source, and is saleable. In general, Python can be used to program simple programs for beginners and to accomplish large projects like any other programming language in the same [72]. As this dissertation is about solving semi-definite programming relaxations in the **Big Mac library** [17]. Choose different technique line search for the penalty method, also with augmented Lagrangian methods, also doing review theoretical convergence properties of these methods.

5.1 Software for Solving Portfolio Optimization Problem Using Penalty Function

After learned about the penalty function and how used the penalty function in the previous chapter to solve constrained problems by transforming them into unconstrained problems, also touched on the types of penalty function, internal penalty function, and external penalty, and explained the behaviour of parameter penalty in each of these types, will expand further and try to use the penalty function to find the optimal solution of NP-hard with mix constraint.

In particular, solving portfolio optimization problem using penalty function, compare the results with the results of solving the same problem using Gurobi Optimization Solver(Gurobi Optimizer is generate optimal solutions and enable optimal decision making with the world's fastest and most powerful mathematical optimization solver) [63].

Let us consider the following portfolio problem

$$\begin{cases} \min_x & \mathrm{a}^\top P\mathrm{a} \\ \text{subject to} & n^\top a \geq \lambda \\ & \mathrm{c}^\top a \leq 1 \\ & 0 \leq a_i \leq r_i \quad \forall i = 1, \ldots, m \\ & \|a\|_{\,0} \leq \mathrm{X} \end{cases} \tag{5.1}$$

where $P$and$n$ is covariance matrix mean of m possible assets and $c^\top a \leq 1$ resource constraint. $\|a\|_{\,0}$ denotes the number of non-zero elements in X. It is a classical mean-variance portfolio, what makes it a mixed-integer problem. These requirements come from real-world practice. In general, a portfolio made up of a large number of assets with very small holdings is not desirable mainly because of transactions costs. Because of its wide usage, cardinality constrained portfolio optimization problem has lately been intensively studied, especially from the computational viewpoint. In general, mixed-integer optimization problems are very difficult to solve. While the classical Markowitz model(the Markowitz model put forward by Harry Markowitz in 1952 is a portfolio optimization model; it assists in the selection of the most efficient portfolio by analyzing various possible portfolios of the given securities. Here, by choosing securities that do not 'move' exactly together [65]) is a convex quadratic programming problem with a polynomial complexity, the cardinality constrained problem is a NP-hard problem. Therefore, real-world cardinality constrained problems involving markets with less than a hundred assets have not yet been solved to optimality. This all being said motivates our attempt to solve the problem(see equation(5.1)) using the penalty

method. Firstly, from follows that (see equation(5.1)) can be reformulated as

$$\begin{cases} \min_x & \mathrm{a}^\top P \mathrm{a} \\ \text{subject to} & n^\top a \geq \lambda \\ & \mathrm{c}^\top a \leq 1 \\ & 0 \leq a_i \leq r_i y_i \quad \forall i \in [1, m] \\ & y_i \in \{0, 1\} \quad \forall i \in [1, m] \\ & c^\top y \leq \mathrm{X} \end{cases} \tag{5.2}$$

Also, consider the standard relaxation of the mixed-integer problem:

$$\begin{cases} \min_x & \mathrm{a}^\top P \mathrm{a} \\ \text{subject to} & n^\top a \geq \lambda \\ & \mathrm{c}^\top y \geq n - \mathrm{X} \\ & c^\top a \leq 1 \\ & y_i a_i = 0 \quad \forall i \in [1, m] \\ & 0 \leq a_i \leq r_i \quad \forall i \in [1, m] \\ & 0 \leq y_i \leq 1 \end{cases} \tag{5.3}$$

To test our approach, use the same data as in but reduce the dimension to avoid memory issues. take the first 15 rows from and set the cardinality constraint to be 5. In order to compare the efficiency of the penalty method, solved(see equation (5.2)) directly using GUROBI via the provided Python interface. For more details about GUROBI . Because of dimension reduction, GUROBI has found the optimal point in every example. Our

approach is based on the regularization problem below:

$$\begin{cases} \min_x & \mathrm{a}^\top P \mathrm{a} \\ \text{subject to} & n^\top a \geq \lambda \\ & \mathrm{c}^\top a \geq n - \mathrm{X} \\ & c^\top a \leq 1 \\ & y_i a_i = 0 \quad \forall i \in [1, m] \\ & 0 \leq a_i \leq r_i \quad \forall i \in [1, m] \\ & \pi(a_i, y_i, t) \leq 0 \quad \forall i \in [1, m] \\ & \bar{\pi}(a_i, y_i, t) \leq 0 \quad \forall i \in [1, m] \\ & 0 \leq y_i \leq 1 \quad \forall i \in [1, m] \end{cases} \tag{5.4}$$

where function π and $\bar{\pi}$are of the form:

$$\pi(h, g : t) \begin{cases} (h - t)(g - t) & \text{if} \quad h + g \geq 2t \\ -\frac{1}{2}[(h - t)^2 + (g - t)^2] & \text{if} \quad h + g \leq 2t \end{cases} \tag{5.5}$$

$$\bar{\pi}(h, g : t) \begin{cases} (-h - t)(g - t) & \text{if} \quad h + g \geq 2t \\ -\frac{1}{2}[(h - t)^2 + (g - t)^2] & \text{if} \quad -h + g \leq 2t \end{cases} \tag{5.6}$$

t is called the regularization parameter, it can be easily seen that $\forall t \geq 0$

$$\pi(h, g, t) \leq 0 \Leftrightarrow h \leq t \quad or \quad g \leq t \Leftrightarrow \min\{h, g\} \leq t$$

as well as

$$\bar{\pi}(h, g, t) \leq 0 \Leftrightarrow -h \quad or \quad g \leq t \Leftrightarrow \min\{-h, g\} \leq t$$

(see equation (5.4)) is solved iteratively, also using GUROBI, beginning with the regularization parameter t = 1 and decreasing it by 0.1 in every iteration. The algorithm would stop if the regularization parameter became too small, i.e., $t \leq 10^{-10}$ or

if the violation of the orthogonality conditions was sufficiently small, this means, $\max_{i=1,\ldots,n} |a_i y_i| \leq 10^{-6}$. Convergence result of the regularization method can be found in. Finally, rewrite(see equation (5.2)) using penalty functions as

$$
\begin{cases}
\min_{x,y} & \\
& \mathrm{a}^\top P \mathrm{a} \\
& + \mu \max\left\{0, c^\top a - 1\right\}^2 \\
& + \mu \max\left\{0, \lambda - n^\top a\right\}^2 \\
& + \mu \max\ \left\{0, n - \mathrm{X} - c^\top y\right\}^2 \\
& + \mu \sum_{i=1}^{n} \max\left\{0, \pi(a_i, y_i, t)\right\}^2 \\
& + \mu \sum_{i=1}^{n} \max\left\{0, \bar{\pi}(a_i, y_i, t)\right\}^2 \\
& + \mu \sum_{i=1}^{n} \max\left\{0, a_i - r_i\right\}^2 \\
& + \mu \sum_{i=1}^{n} \max\left\{0, -a_i\right\}^2 \\
& + \mu \sum_{i=1}^{n} \max\left\{0, -y_i\right\}^2 \\
& + \mu \sum_{i=1}^{n} \max\left\{0, y_i - 1\right\}^2
\end{cases}
\tag{5.7}
$$

Where the penalty parameter μ is set to be $\mu = \dfrac{1}{t}$

problem	GUROBI	Regularization	PENALLTY
Orl200-005a	44.75297	44.71297	44.29227
Orl200-005b	66.34850	66.34840	66.03850
Orl200-05a	31.51770	31.41770	31.40874
Orl200-05b	16.06268	16.06268	15.52719

Table 5.1: Solutions of the cardinality constrained portfolio problem

In example Orl200-0050 solve by using regularized is same the original solution, if use the approach approximates solutions very closed to optimal solution.

From tabular can be seen that in every example solution of regularized portfolio problem is the same as the solution of original problem. Our approach approximates solutions very close to the optimal ones.

Code 15 :Solve the Portfolio optimization problem directly use GUROBI

```
from␣gurobipy␣import␣*
import␣pandas␣as␣pd
import␣numpy␣as␣np
size=15
cardinality=5
sigma=np.genfromtxt("or1200-005-a.mat",␣skip_header=1,dtype='int')
sigma=sigma[0:size,0:size]
m=pd.read_csv("or1200-005-a.txt",sep="",index_col=False
,␣skiprows=1,names=['mu_i','ignore'])
rho=np.genfromtxt("or1200-005-a.rho")
rho=rho.item()
m=m.iloc[0:size,0]
bounds=pd.read_csv("or1200-005-a.bds",sep="",
index_col=False,names=['1-i','u-i'])
bounds=bounds.transpose()
u_i=bounds.iloc[1,0:size]
model=Model('portfolio')
vars_=pd.Series(model.addVars(size))
y=pd.Series(model.addVars(size,vtype=GRB.BINARY))
vars_.start=np.zeros(size)
y.start=np.ones(size)
portfolio_risk=vars_.dot(sigma).dot(vars_)
```

```
model.setobjective(portfolio_risk,GRB.MINIMIZE)
model.addConstr(vars_.sum()<=1,'budget')
m=np.asarray(m)
model.addConstr((0<=vars_[i]␣for␣i␣in␣range(size)),'nonnegative')
model.addConstr((vars_[i]<=u_i[i]*y[i]␣for␣i␣in␣rang␣(size)),'u_i')
model.addConstr(y.sum()<=cardinality,'cardinality')
portfolio_retun=m.dot(vars_)
model.addConstr(rho<=m.dot(vars_),'portf_return')
model.addConstr((y[i]<=1␣for␣i␣in␣range(size)),'interva1')
model.addConstr((y[i]<=1␣for␣i␣in␣range(size)),'interva2')
model.update()
model.optimize()
```

Code 16 :Solve the problem of optimizing the regularized portfolio using GUROBI

```
t=1
x_guss=np.zeros(size)
y_guss=np.noes(size)
cond=1
while((t>=10**-10)&(cond>10**-6)):
␣␣␣␣model=Model('portfolio')
␣␣␣␣vars_=pd.Series(model.addVars(size))
␣␣␣␣y=pd.Series(model.addVars(size))
␣␣␣␣min1=pd.Series(model.addVars(size))
␣␣␣␣min2␣=␣pd.Series(model.addVars(size))
␣␣␣␣portfolio_risk=vars_.dot(sigma).dot(vars_)
␣␣␣␣model.setobjective(portfolio_risk,GRB.MINIMIZE)
␣␣␣␣model.addconstr(vars_.sum()<=1,'budget')
␣␣␣␣model.addconstrs(((y[i]<=1)␣for␣i␣in␣range(size)),␣'yL1')
```

```
␣␣␣␣model.addconstrs(((0<=vars_[i])␣for␣i␣in␣range(size)),'u_i')
␣␣␣␣model.addconstrs(((vars_[i]<=u_i[i])␣for␣i␣in␣range(size)),'u_i')
␣␣␣␣model.addconstr(size-cardinality<=y.sum(),'cardinality')
␣␣␣␣model.addconstr(rho<=m.dot(vars_),␣'portf_return')
␣␣␣␣vars_.start=x_guss
␣␣␣␣y.start=y_guss
␣␣␣␣model.addconstrs((min1[i]==min_(vars_[i],y[i])
␣␣␣␣␣for␣i␣in␣range(size)),␣'min1')
␣␣␣␣model.update()
␣␣␣␣model.optimize
␣␣␣␣x_guss=np.array([v.x␣for␣v␣in␣vars_])
␣␣␣␣y_guss=np.array([v.x␣for␣v␣in␣y])
␣␣␣␣t=t*0.01
␣␣␣␣cond=max(abs(np.multiply(x_guss,y_guss)))
```

Code 17 :**Using Penalty Function to Solve Portfolio Optimization Problem**

```
import␣␣math
from␣scipy.optimize␣import␣minimize
def␣phi(a,b,t):
␣␣␣␣if(a+b>=2*t):
␣␣␣␣␣␣␣␣result=(a-t)*(b-t)
␣␣␣␣else:
␣␣␣␣␣␣␣␣result=-1/2*((a-t)**2+(b-t)**2)
␣␣␣␣␣␣␣␣return␣result
def␣phi_(a,b,t):
␣␣␣␣if(-a+b>=2*t):
␣␣␣␣␣␣␣␣result=(-a-t)*(b-t)
␣␣␣␣else:
```

```
␣␣␣␣␣␣␣␣result=-1/2((-a-t)**2+(b-t)**2)
␣␣␣␣␣␣␣␣return␣result
def␣objective(x):
␣␣␣␣obj=x[0:size].dot(sigma).dot(x[0:size])
␣␣␣␣obj+=mu*max(0,x[0:size].sum()-1)**2
␣␣␣␣obj+=mu*max(0,rho-m.dot(x[0:size]))**2
␣␣␣␣obj+=mu*max(0,size-cardinality-x[size:size*2].sum())**2
␣␣␣␣for␣i␣in␣range(size):
␣␣␣␣␣␣␣␣obj+=mu*max(0,x[i]-u_i[i])**2
␣␣␣␣␣␣␣␣obj+=mu*max(0,-x[i])**2
␣␣␣␣␣␣␣␣obj+=mu*max(0,-x[i+size])**2
␣␣␣␣␣␣␣␣obj␣+=␣mu␣*␣max(0,␣x[i␣+␣size]-1)␣**␣2
␣␣␣␣obj␣+=␣mu␣*␣max(0,␣phi(x[i],␣x[i+␣size],t)␣)**␣2
␣␣␣␣obj␣+=␣mu␣*␣max(0,␣phi_(x[i],␣x[i␣+␣size],␣t))␣**␣2
␣␣␣␣obj␣+=␣mu␣*␣max(0,-1/2*((-x[i]-t)**2+(x[i+size]-t)**2))**2
␣␣␣␣return␣obj
viol=math.inf
x0=np.append(np.zeros(size),np.ones(size))
old_viol=viol
eps=10**-10
cond=1
it=0
t=1
while((eps<=viol)&(t>=10**-10)):
␣␣␣␣mu=1\t
␣␣␣␣it+=1
␣␣␣␣x=minimize(objective,x0).x
␣␣␣␣violence=np.zeros(3+5*size)
```

```
    violence[0]=max(0,x[0:size].sum()-1)
    violence[1] = max(0,size-cardinality-x[size:size*2].sum())
    violence[2] = max(0,rho-m.dot(x[0:size]))
    for i in range(size):
        violence[i+3]=max(0,phi(x[i],x[i+size],t))
        violence[i + 3] = max(0, phi_(x[i], x[i + size], t))
        violence[(i + 3)+size] = max(0,-x[i+size])
        violence[(i+3)+2*size]=max(0,x[1+size]-1)
        violence[(i+3)+3*size]=max(0,-x[0])
        violence[(i + 3) + 4 * size] = max(0, x[i]-u_i[i])
        viol=max(violence)
        print('{}->{}'.format(it,x[size].dot(sigma).dot(x[0:size])))
        x0=x
        t=t*0.1
```

5.2 Numerical Resulted

Applied different techniques are BackTracking and Hager Zhang line research on two types of methods are augmented Lagrangian and Penalty Method. The figure(see equation 4.6) shows that the behaviour of the method differs at the convergence to minimizer, if change in the search line. When use limited-memory BFGS (L-BFGS)(BackTracking) note that the beaver of convergence it go faster than limited-memory BFGS (L-BFGS)(Hager Zhang) while Hager Zhang need more time than BackTracking. In table at ($gO5_100.5$), assessed the exhibition of the conjugate gradient and the LBFGS techniques with the Hager-Zhang line search calculation. For is addition, the form inclination strategy required less capacity calls to come to the semi-definite bound. At the point when similar optimization algorithms are run with the BackTracking line search calculation, LBFGS requires altogether less function calls to come to the semi-definite bound. We evaluated the performance of the conjugate

gradient and the LBFGS methods with the Hager-Zhang line search algorithm. In this case, the conjugate gradient method required fewer function calls to reach the semide nite bound. When the same optimization algorithms are run with the BackTracking line search algorithm, LBFGS requires signi

cantly fewer function calls to reach the semide

nite bound. For this reason, we continued using LBFGS with BackTracking for this problem.

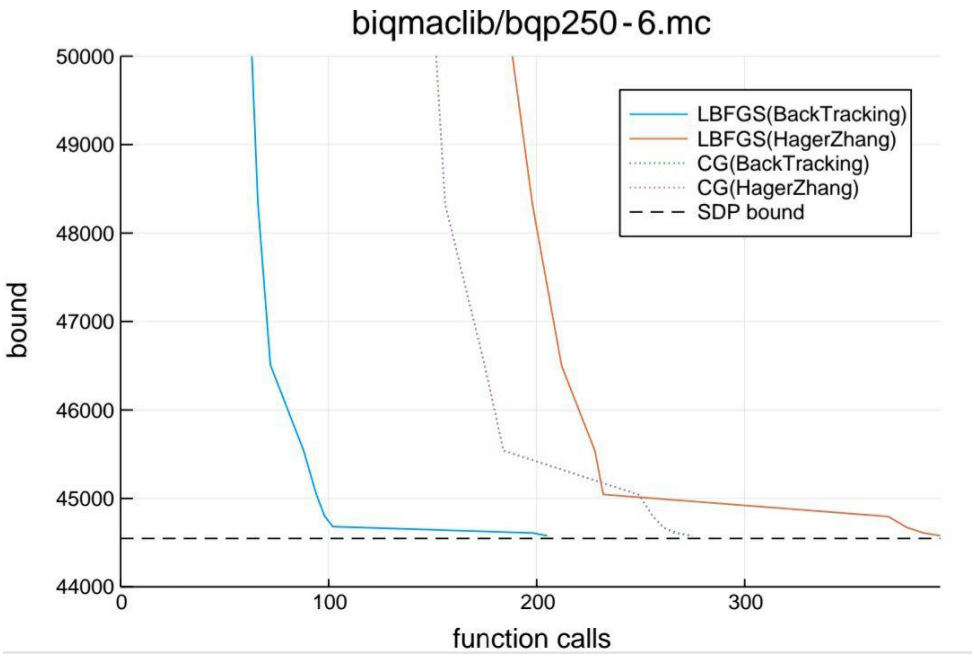


Figure 5.1: $L-BFGC$ with $(BackTracking)$ reach the bound faster than $L-BFGS$ with $Hager-Zhang$

Augmented Lagrangian Method(HagerZhang Line search:)										
	Aug. (LBFGS) m=1		Aug. (LBFGS) m=2		Aug. (LBFGS) m=10		Aug. (BFGS)		Aug. (CQ)	
Problem	time	fcalls	time	fcalls	time	fcalls	time	fcalls	time	fcalls
g05_100.0	1.76	116	1.96	123	1.58	122	1.92	163	0.88	83
g05_100.1	1.24	112	1.28	122	1.31	109	1.40	123	0.76	87
g05_100.2	1.63	132	1.63	126	1.68	144	2.12	191	0.92	105
g05_100.3	1.79	169	1.62	155	1.78	159	1.87	162	1.28	107
g05_100.4	1.41	123	1.52	135	1.48	126	2.01	158	0.82	93
g05_100.5	1.45	114	1.76	113	1.49	120	1.72	119	0.83	89
g05_100.6	1.21	110	1.23	111	1.23	111	1.25	106	0.78	83
g05_100.7	1.25	106	1.20	108	1.26	108	1.20	100	0.77	79
g05_100.8	1.29	123	1.30	123	1.37	126	1.38	130	0.80	94
g05_100.9	1.19	106	1.13	100	1.18	103	1.13	95	0.67	75

Table 5.2: Augmented Lagrangian Method (HagerZhang Line search:)

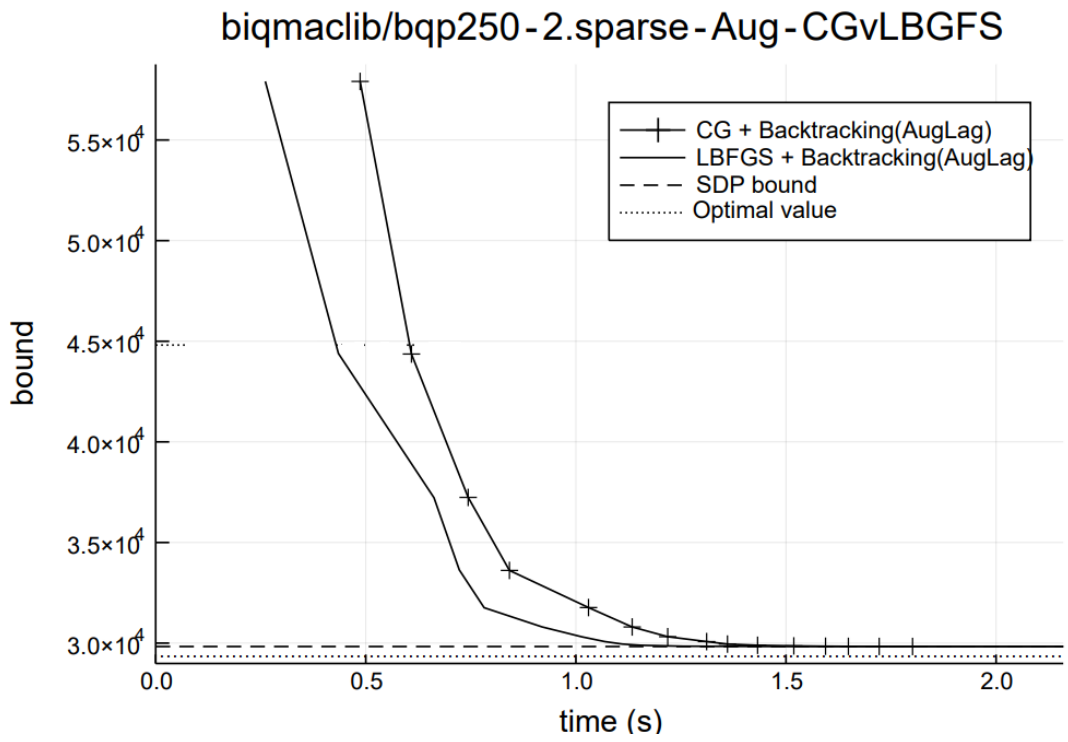


Figure 5.2: Applied different techniques are BackTracking and Hager Zhang line research on two types of methods are augmented Lagrangian and Penalty Method.

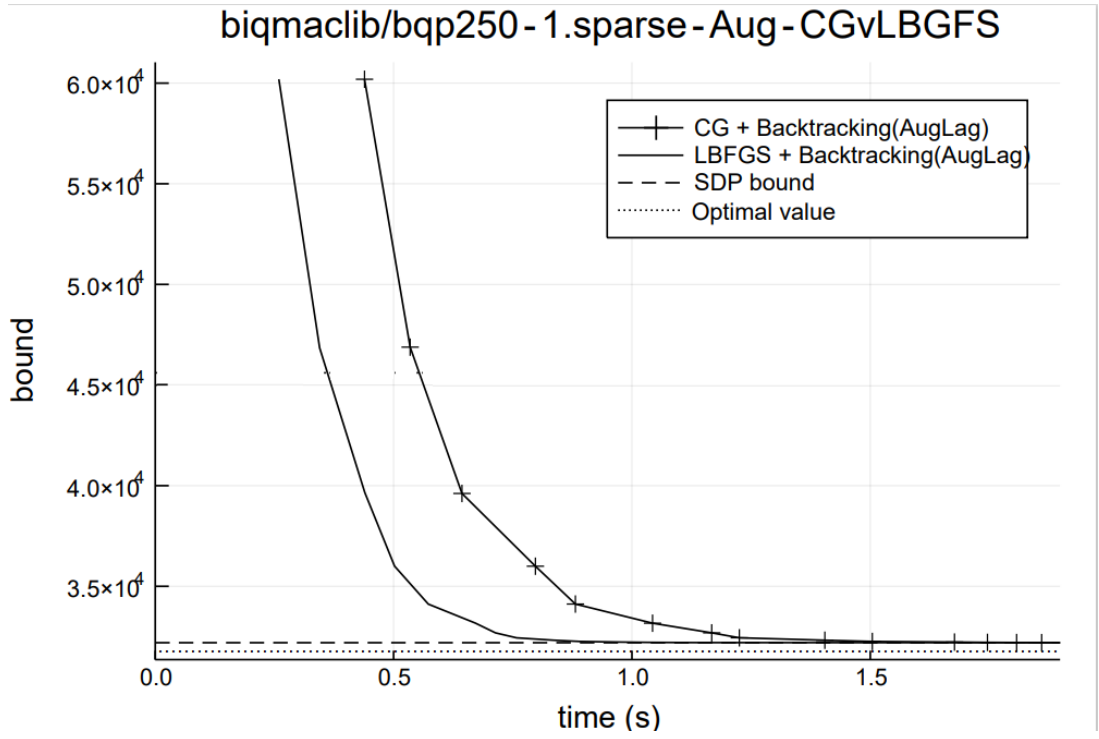


Figure 5.3: Applied different techniques are BackTracking and Hager Zhang line research on two types of methods are augmented Lagrangian and Penalty Method.

Augmented Lagrangian Method(BackTracking Line search)										
	Aug. (LBFGS) m=1		Aug. (LBFGS) m=2		Aug. (LBFGS) m=10		Aug. (BFGS)		Aug. (CQ)	
Problem	time	fcalls	time	fcalls	time	fcalls	time	fcalls	time	fcalls
g05_100.0	0.90	85	1.11	92	0.78	82	1.13	145	0.77	83
g05_100.1	0.91	80	0.91	82	0.34	57	1.01	128	0.83	87
g05_100.2	1.25	96	1.35	106	0.93	87	.97	103	0.84	84
g05_100.3	0.88	93	0.88	94	0.92	94	1.22	165	0.94	105
g05_100.4	1.12	105	1.96	99	1.27	102	1.53	175	1.35	107
g05_100.5	0.90	91	0.89	89	1.04	99	1.23	164	0.79	93
g05_100.6	0.82	84	0.82	82	1.57	113	0.89	104	0.77	89
g05_100.7	0.83	82	0.86	89	0.82	80	0.94	113	0.72	83
g05_100.8	0.90	95	1.03	103	1.08	93	1.44	165	0.91	94
g05_100.9	0.88	84	0.84	83	0.96	71	0.90	107	0.67	75

Table 5.3: CPU time and function calls number of iterations for augmented Lagrangian method(BackTracking Line search)

SDP have solution while NP hard problems have no solution, NP subset of SDP. So, take bounded to NP hard problems, when touch this bounded will be solution. at problem g05-100.0 use Augmented Lagrangian Method(BackTracking Line search) with (LBFGS) touch pounded at 82 ,with BFGS touch pounded at 145 while CG touch pounded with 83 function call.

Penalty Method (BackTracking Line search)										
	Aug. (LBFGS) m=1		Aug. (LBFGS) m=2		Aug. (LBFGS) m=10		Aug. (BFGS)		Aug. (CQ)	
Problem	time	fcalls	time	fcalls	time	fcalls	time	fcalls	time	fcalls
g05_100.0	1.40	136	1.22	137	2.90	151	12.08	2481	1.61	165
g05_100.1	1.39	136	1.29	123	1.30	123	1.87	196	1.69	135
g05_100.2	1.38	147	1.27	141	1.37	139	2.04	296	1.46	191
g05_100.3	1.38	158	1.39	159	1.24	145	2.42	368	1.64	220
g05_100.4	2.38	180	1.51	171	2.14	179	3.20	436	2.22	238
g05_100.5	1.37	157	1.31	157	2.82	173	2.48	389	1.45	196
g05_100.6	1.10	128	1.61	165	1.12	126	2.07	286	1.26	163
g05_100.7	1.18	137	1.22	132	1.02	120	1.96	238	1.16	153
g05_100.8	1.20	139	1.24	142	2.50	154	1.81	284	1.52	182
g05_100.9	1.10	121	1.09	120	2.76	147	1.75	240	1.15	148

Table 5.4: CPU time and function calls number of iterations for Penalty Method (BackTracking Line search)

Penalty Method (HagerZhang Line search)										
	Aug. (LBFGS) m=1		Aug. (LBFGS) m=2		Aug. (LBFGS) m=10		Aug. (BFGS)		Aug. (CQ)	
Problem	time	fcalls	time	fcalls	time	fcalls	time	fcalls	time	fcalls
g05_100.0	2.43	242	2.28	237	2.34	238	3.06	308	1.33	83
g05_100.1	2.06	208	2.15	206	2.75	203	3.27	257	1.29	87
g05_100.2	2.55	254	2.69	266	2.98	264	3.51	357	1.92	82
g05_100.3	3.05	299	2.91	274	2.78	273	4.09	448	1.67	105
g05_100.4	3.16	339	3.06	333	3.06	328	4.81	518	01.89	107
g05_100.5	2.84	268	3.31	286	2.80	271	4.08	420	1.53	93
g05_100.6	2.30	238	2.55	242	2.81	242	4.21	344	1.27	89
g05_100.7	2.39	242	2.10	216	2.16	227	2.74	285	1.17	83
g05_100.8	2.75	265	2.33	231	2.19	233	3.15	355	1.35	94
g05_100.9	2.16	223	2.02	208	2.11	214	2.86	312	1.15	75

Table 5.5: CPU time and function calls number of iterations for Penalty Method (HagerZhang Line search)

Choose a different problem from the **Big Mac library** and using two kinds of line searches are (HagerZhang Line search and BackTracking), to the same method Augmented Lagrangian a note the convergence to minimize take different behaviour also the same process in Penalty methods. If the change line search for the method will be got different results.

5.3 Multiple-Objective Function

After a disease was discovered with new signs, exactly in the Chinese city of Ohan and it quickly spread in that city, and then the world, by the passage of days it became a pandemic that took many lives. Many researchers took care to study the behaviour of

disease spread, and researchers in the field of mathematics tried to find a mathematical formula for the spread process, and in most of those studies the society was classified into three types:

(a) Not infected

(b) An injured under treatment, and this type is divided into two types

- Survivor of disease
- Not a survivor (death)

with adherence to the fashionable guidelines and normalization of city closures and social distancing, the number of injuries has been reduced. A vaccine for this disease was recently discovered in the United States of America. The role of optimization has emerged in solving the problem of transferring this vaccine from the producer to the consumer, and the goal is to reduce the price of the dose in addition to finding appropriate means of transfer.

Infected population initially grows at an exponential rate but after some iterations, do the lock down to the cities and follow health guidelines, the number of infected people starts decreasing in subsequent iterations. "COVID-19" may people infecting and these people can either die, infect other people, or simply get recovered after the disease. This action is typically modelled by an "S I R", model consisting of three kinds of individuals: S = number of susceptible, I = number of infections, and R = number of recovered.

5.3.1 Mathematical Concept

Let $S \subset R^p$ denote a hyper parallelepiped of $x = (x_1 \dots x_p)^\top \in R^p$, the vector $x \in S \subset R^p$ is called deduction vector, it's entries are the deduction variables, and it is bounded by

$$x_i^{inf} \leq x_i \leq x_i^{sup}$$

for every $i = 1 \dots n$ the domain S is known us the deduction variables [76]. suppose that $f : S \subset R^p \to R^q$ with $q \geq 2$ is called the multiple-objective function, where

$$\lambda(x) = (\lambda^1(x) \dots \lambda^q(x))^\top$$

and $\lambda^k : S \to R$ for $k = 1, \dots, q$ are the objectives. The vector x is shown to as a point or a solution criss-cross, since it is inwards the set of elements in the deduction variables space that finally appear a solution for the optimization problem. Let $g : S \to R^p$ and $h : S \to R^q$ the set of equality and inequality constraints that border the subspace to be seek for the optimal solution are obvious us :
$g(x) \leq 0$ and $h(x) \leq 0$, where $g^i(x) \leq 0$, for $i = 1, \dots, p$ and $h^j(x) \leq 0$, for $j = 1, \dots, q$. The set of solutions that satisfy both equality and inequality constraints and let feasible set $\omega = \left\{x \in R^n | x^{inf} \leq x \leq x^{sup}, g(x) = 0, h(x) \leq 0\right\}$.
Let $V = Im(f|_\omega) \{y = f(x) \subset R^m, \forall x \in \omega\}$

$$\begin{cases} \text{minimize} & \lambda(x) = (\lambda^1(x) \dots \lambda^m(x))^\top \\ \text{subject to} & h^j(x) \leq 0, \quad j = \{1, \dots, p\}, \\ & g^i(x) = 0, \quad i = \{1, \dots, q\}, \\ & x^{inf} \leq x \leq x^{sup}. \end{cases} \tag{5.8}$$

there is no single point eligible for minimizing all functions together in a multiple-objective problem, this means that none of the objectives can be improved without delay to at least one of the other objective functions.
Can solve this drag by the nation of the way of solutions and Pare to optimal.
The nation of domination defines a way to compare scope solutions in optimization problems with multiple-objective. A solution x^1 is said to dominate another solution x^2, denote by $x^1 \prec x^2$, if both of the conditions below are satisfied:

(a) Solution x^1 is worst than x^2 such that $\lambda^k(x^1) \leq \lambda^k(x^2) \forall k = 1, \dots, q$

(b) at least one object satisfy $k(x^1) < \lambda^k(x^2) \forall k = 1, \ldots, q$.

The Pare to-optimal which in turn are denoted by $x^8 \in \omega$, that represents the best poise related the minimization of all objective together. An objective vector $z^* \in V$ is called to be a Pare to-optimal if there is no other objective-vector $z \in V$ such that $z_i \leq z_i^* \quad \forall k = 1, \ldots, m$ and $z_i < z_i^*$.

5.3.2 The mathematics formula S-I-R model

The S-I-R model is adopted, to describe the dynamic behaviour of the Coronavirus epidemic in china, the population size fixed not adding to the S, there is also no emigration do not have potion growth by any means recovery gives total immunity so once you recover from whatever this illness is you no longer are susceptible to this depends on how much of virus mutates some other factors but going to assumed for simplicity that once you recover you can not get it again there is a fixed infection rate per day so how many people does an infected person come into and there is fixed recovery time maybe it takes two weeks to, it takes two weeks to recover from this illness that modelling and people are mixed so can characterize the number of contacts per day and amount of time to recover these things are uniform throughout the region and that is affected, r = is the number of contact per day and a= is going to be over the amount of time it takes [73].

- S = susceptibles
- I = infectives
- R = removied

Since the number of non-infected people is constantly changing then

$$\frac{dS}{dt} = -rSI$$

where r rate of contact. The negative sign that appears in the equation because it is decreasing

$$\frac{dI}{dt} = -rSI - aI$$

because susceptibles are go to infectives and

$$\frac{dR}{dt} = aI$$

The initial point it is $S = S_0, I = I_0, R = 0$. Now have three differential equations for are three categories of people within the population so S number of susceptibles is going to decrease, according to the number of contacts between infective x and susceptible i is the number of effective and this will increase due to contact between people and decrease from people either recovering or dying as a result of the disease and finally the removed category people that no longer

$$S + I + R = 0$$

because the total population is given by $S + I + R$.

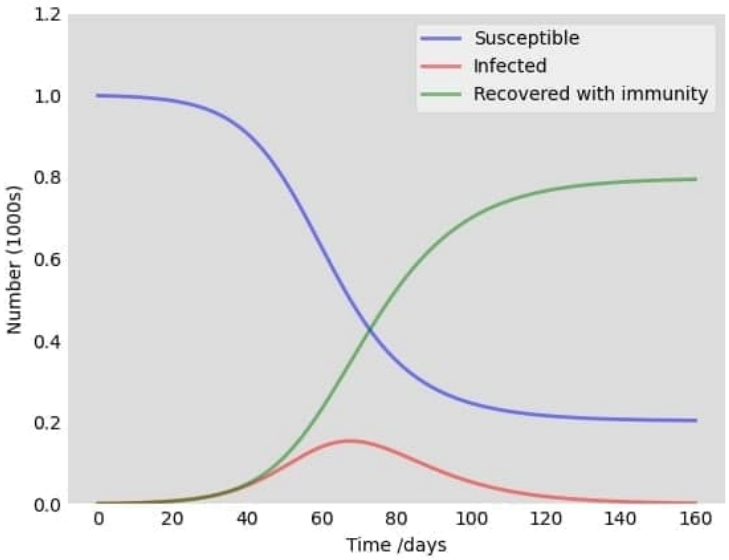


Figure 5.4: Describes an increase and decrease in the numbers of those susceptible, infected and recovered within 365 days

Coronavirus Optimization Algorithm

- define sufferer people. New sufferer people as personage
- removed , survivor as personage
- patient-zero as personage
- iteration as integer
- iteration ← 0
- while iteration <period and sufferer people> 0
- removed ← die
- each i ∈ sufferer people do
- patient-zero ← sufferer people(sufferer,removed,survivor)
- if not-null(patient-zero):
- New sufferer people ← patient-zero

  end if

  end for
- current patter personage ← choose patter New sufferer people
- if fitness of "current patter personage "> "choose patter New sufferer: people "
- patter ← current patter personage

  end if
- survivor ← sufferer(ill) people
- clear (sufferer(ill) people)
- (sufferer(ill) people) ← (new sufferer(ill) people)

- iteration ← iteration + 1

  end while

  return patter-personage

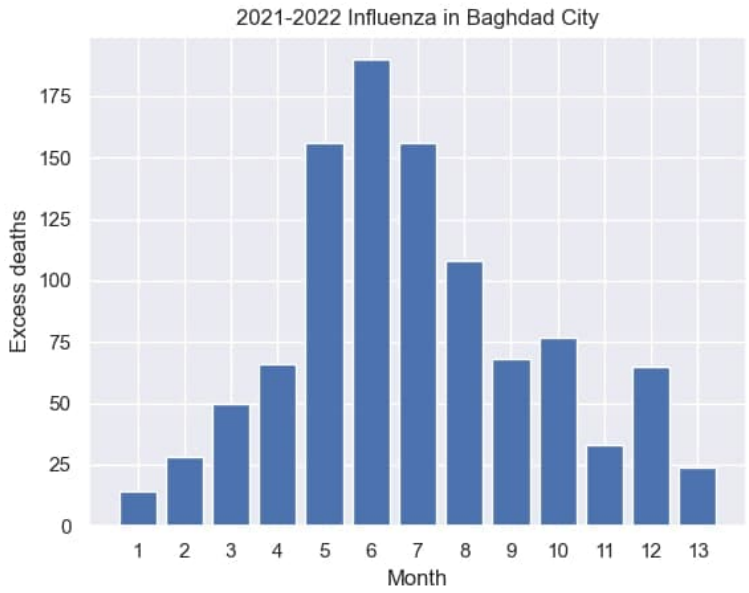


Figure 5.5: Monthly information on the number of persons Excess deaths by the coronavirus in the Baghdad city governorate for 2021.

5.3.3 The suggestion optimization formula for Coronavirus

After identifying the illness and how it is the spreading pattern between humans, so will suggest its sample from society. $N \subset R^n$

- Let f_1 denote by individual which is not yet infected by the illness pathogen(susceptible)
- Let f_2 denote by individual in the incubation period after being infected by the illness pathogen, and with invisible clinical signs(not exposed)
- Let f_3 denote by personage that can infect others(infected active)

- Let f_4 denote by personage who survived after being infected but is no longer infectious and has developed a natural immunity to the illness pathogen

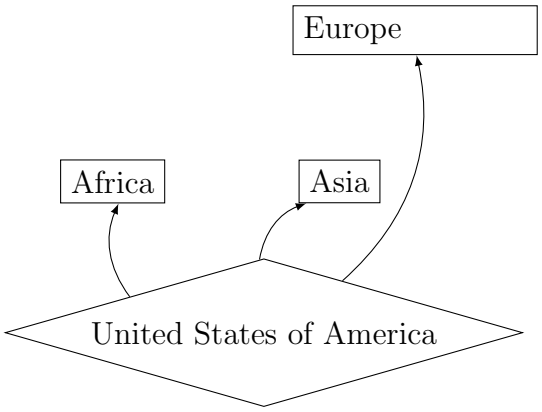
Suppose that the general problem

$$\begin{cases} \text{minimize} & f^i(x) \\ \text{subject to} & h^i(x) \leq 0, \quad i = \{1, \ldots, N\}, \\ & g^i(x) = 0, \quad i = \{1, \ldots, N\}, \\ & k^i(x) = 0, \quad i = \{1, \ldots, N\}, \\ & l^i(x) = 0, \quad i = \{1, \ldots, N\}, \\ & x^{inf} \leq x \leq x^{sup}. \end{cases} \tag{5.9}$$

- where$g^i(x) = 0$ the individual gets the vaccine
- And $h^i(x) \leq 0$ the individual gets the antibiotic
- $k^i(x) = 0$ the society doing the blockdown
- $l^i(x) = 0$ the society follow the instructions healthy

Consider multiple-objective function and looking for finding the optimal solution on the feasible set ω, which is bounded by our constraint is appeared above the aim to minimize the number of individuals are will the infection be an illness so contain the illness.

5.4 The Problem of Transferring Vaccine from Producer to Beneficiary



If assume that the Pfizer vaccine manufactured by American medical companies has been exported from America to countries of the world, including Iraq, and as is known about this vaccine, there are obstacles to its transfer. Because it requires a refrigerated container to below $-80°$. In addition to that, even transportation, the world's aviation must be harnessed by 90%. Because of the high cost of transportation, the price will be added to the price of a single dose of this vaccine. Hence, in this effort, strive to find the lowest transportation cost by use CVOA (Coronavirus Optimization Algorithm)

Finding it is minimizing expensive than this vaccine, in addition to maximizing the rate of export from America to the world [55].

begin with a much-simplified example of a problem that might arise in manufacturing and transportation. A chemical company has 1 factory F suppose three retail outlets $Europe$, $Asia$,$Africa$, the F can produce a^i dose of a certain chemical product every week; a_i is called the capacity of the plant, retail outlet weekly demand of b_j dose of the product, the cost of shipping 100$ and need to exporting million of the vaccine from American to

Europe , *Asia* and $Africa$ are c_{ij} every week

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$\text{subject to } \sum_{j=1}^{3} xij \leq a^i$$

$$\sum_{i=1} x_i j \geq b_j$$

$$xij \geq 0$$

5.5 CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

The goal of the study focused on the following points:

(a) The L-BFGS algorithm works well for large data sets because it needs less memory than standard BFGS. Both algorithms use inverse Hessian matrix estimation to control the search for a variable space.

(b) Goal is to determine the best direction, and the correct step along that direction. In addition, it is possible to update the Hessian instead of inverting it on each iteration.

(c) In order to solve optimization problems, the researchers need best line research. In this dissertation, we studied a complete and detailed analysis of the methods of line research, and also made a comparison between them, the best choice.

(d) Convert constrained optimization problem into unconstrained optimization problem . The technique is used to replace the constrained optimization problems with the unconstrained optimization problems. and then apply the standard techniques such as the exterior penalty function method and the interior penalty method to get good approximate solutions, A detailed study was also made of Argument and compared to the penalty method.

(e) We designed new approach code for fundamental of optimization.

(f) For the numerical analysis, all results were implemented using python language.

FUTURE WORK

(a) The study is built around a continuous function in its domain, which is called a smooth function. In the future, researchers can make a study of the non-smooth function.

(b) The study is built around a monotonous function($f(x^k) < f(x^{k+1})$. In the future, researchers can do the same study on a non-monotonous function.

(c) The study is built around to solve penalty and augmented Lagrange method. In the future, researchers can do the same study on different methods are solve the unconstraint optimization problems .

(d) All the code mentioned in the study is made by python language. Therefore, it is scalable to solve other, more complex problems.

REFERENCES

[1] Ioannis Akrotirianakis. Interior point methods for nonlinear programming and application to mixed integer nonlinear programming. 2000.

[2] Ahmed Al-Jilawi. *Solving the Semidefinite Programming Relaxation of Max-cut Using an Augmented Lagrangian Method*. PhD thesis, Northern Illinois University, 2019.

[3] Ahmad Alhawarat, Zabidin Salleh, and Ibtisam A Masmali. A convex combination between two different search directions of conjugate gradient method and application in image restoration. *Mathematical Problems in Engineering*, 2021, 2021.

[4] Niclas Andréasson, Anton Evgrafov, and Michael Patriksson. An introduction to optimization: Foundations and fundamental algorithms. *Chalmers University of Technology Press: Gothenburg, Sweden*, 1:1–205, 2005.

[5] Tadeusz Antczak. The exact absolute value penalty function method for identifying strict global minima of order m in nonconvex nonsmooth programming. *Optimization Letters*, 10(7):1561–1576, 2016.

[6] Rajesh Kumar Arora. *Optimization: algorithms and applications*. CRC Press, 2015.

[7] Kenneth J Arrow and Leonid Hurwicz. Reduction of constrained maxima to saddle-point problems. In *Traces and Emergence of Nonlinear Programming*, pages 61–80. Springer, 2014.

[8] Adil Bagirov, Napsu Karmitsa, and Marko M Mäkelä. *Introduction to Nonsmooth Optimization: theory, practice and software*. Springer, 2014.

[9] Alexandre Belloni. Lecture notes for iap 2005 course introduction to bundle methods. *Operation Research Center, MIT, Version of February*, 11, 2005.

[10] Hailay Weldegiorgis Berhe. Penalty function methods using matrix laboratory (matlab). *African Journal of Mathematics and Computer Science Research*, 5(13):209–246, 2012.

[11] Dimitri Bertsekas. *Convex optimization theory*. Athena Scientific Belmont, 2009.

[12] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[13] Dimitri P Bertsekas, W Hager, and O Mangasarian. Nonlinear programming. athena scientific belmont. *Massachusets, USA*, 1999.

[14] Dimitri P Bertsekas and Athena Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[15] Nicolas Boumal. An introduction to optimization on smooth manifolds. *Available online, Aug*, 2020.

[16] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[17] Mangeet Chahal, Peter Temesy-Armos, and William J Stewart. Big mac: caseous calcification of the mitral annulus referred for possible cardiac tumor. *Echocardiography*, 28(4):E76–E78, 2011.

[18] Alice Chiche and Jean Charles Gilbert. How the augmented lagrangian algorithm can deal with an infeasible convex quadratic optimization problem. *Journal of Convex Analysis (to appear).[pdf]*, 3(4):5, 2014.

[19] Andrew R Conn, Nick Gould, Annick Sartenaer, and Ph L Toint. Convergence properties of an augmented lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM Journal on Optimization*, 6(3):674–703, 1996.

[20] William Cook, WH Cunningham, WR Pulleyblank, and A Schrijver. Combinatorial optimization. *Oberwolfach Reports*, 5(4):2875–2942, 2009.

[21] P Dimitri Bertsekas. Nonlinear programming. 1999. *Athena, Scientific, Belmont CA*.

[22] Yohanna Ejieji. Nigerian journal of mathematics and applications v olume 24,(2015), 216- 227 c nig. j. math. appl. http://www. njmaman. com.

[23] Mahmoud Mahmoud El-Alem. *A global convergence theory for a class of trust region algorithms for constrained optimization*. PhD thesis, 1988.

[24] Carlos Fernández González. *Métodos matemáticos en problemas de entrelazamiento: convertibilidad de estados, medidas conjuntas y halmiltonianos en PEPS*. PhD thesis, Universidad Complutense de Madrid, 2014.

[25] Michael R Garey and David S Johnson. Computers and intractability, vol. 29, 2002.

[26] Shemels Geletaw. Penalty and augmented lagrangian methods to solve nonlinear optimization problems. 2016.

[27] Philip E Gill, Walter Murray, and Margaret H Wright. *Numerical linear algebra and optimization*. SIAM, 2021.

[28] Giorgio Giorgi and Tinne Hoff Kjeldsen. *Traces and emergence of nonlinear programming*. Springer Science & Business Media, 2013.

[29] N. Gould. A course on continuous optimization. `http://www.numerical.rl.ac.uk/people/nimg/course/index.shtml`, 2006. [Accessed 3-12-2019].

[30] Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[31] JB Hiriart-Urruty and C Lemaréchal. Convex analysis and minimization algorithms. no. 305-306 in grund. der math. wiss, 1993.

[32] Minhui Huang, Shiqian Ma, and Lifeng Lai. A riemannian block coordinate descent method for computing the projection robust wasserstein distance. In *International Conference on Machine Learning*, pages 4446–4455. PMLR, 2021.

[33] Yaochu Jin, Handing Wang, and Chaoli Sun. Classical optimization algorithms. In *Data-Driven Evolutionary Optimization*, pages 41–51. Springer, 2021.

[34] Josef Kallrath. Mathematical solution techniques—the nonlinear world. In *Business Optimization Using Mathematical Programming*, pages 423–446. Springer, 2021.

[35] Christian Kanzow, Daniel Steck, and Daniel Wachsmuth. An augmented lagrangian method for optimization problems in banach spaces. *SIAM Journal on Control and Optimization*, 56(1):272–291, 2018.

[36] Kiflu Kemal. *Interior and Exterior Penalty Methods to Solve Nonlinear Optimization Problems*. PhD thesis, Addis Ababa University. College of Natural Sciences. Department of Mathematics., 2017.

[37] RP King. Necessary and sufficient conditions for inequality constrained extreme values. *Industrial & Engineering Chemistry Fundamentals*, 5(4):484–489, 1966.

[38] Krzysztof C Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical programming*, 46(1-3):105–122, 1990.

[39] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):1–23, 2017.

[40] Anand Kulkarni, Ganesh Krishnasamy, and Ajith Abraham. *Introduction to Optimization*, volume 114, pages 1–7. 09 2017.

[41] Claude Lemarechal. Methods of descent for nondifferentiable optimization (krzysztof c. kiwiel). *SIAM Review*, 30(1):146, 1988.

[42] Xiangli Li, Wenjuan Zhao, and Xiaoliang Dong. A new cg algorithm based on a scaled memoryless bfgs update with adaptive search strategy, and its application to large-scale unconstrained optimization problems. *Journal of Computational and Applied Mathematics*, page 113670, 2021.

[43] Shuai Liu. A simple version of bundle method with linear programming. *Computational Optimization and Applications*, 72(2):391–412, 2019.

[44] Bruno F Lourenço, Ellen H Fukuda, and Masao Fukushima. Optimality conditions for nonlinear semidefinite programming via squared slack variables. *Mathematical Programming*, 168(1-2):177–200, 2018.

[45] David Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[46] Jérôme Malick. The spherical constraint in boolean quadratic programs. *Journal of Global Optimization*, 39:609–622, 2007.

[47] J McKeown, D Meegan, and D Sprevak. *An introduction to unconstrained optimisation*. CRC Press, 1990.

[48] Angelo Miele, EE Cragg, RR Iyer, and AV Levy. Use of the augmented penalty function in mathematical programming problems, part 1. *Journal of Optimization Theory and Applications*, 8(2):115–130, 1971.

[49] Shashi Kant Mishra and Bhagwat Ram. *Introduction to Unconstrained Optimization with R*. Springer Nature, 2019.

[50] Shashi Kant Mishra, Mohammad Esmael Samei, Suvra Kanti Chakraborty, and Bhagwat Ram. On q-variant of dai–yuan conjugate gradient algorithm for unconstrained optimization problems. *Nonlinear Dynamics*, pages 1–26, 2021.

[51] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[52] Michael Panik. *Linear programming and resource allocation modeling*. John Wiley & Sons, 2018.

[53] Nikolaos Ploskas, Nikolaos Samaras, et al. *Linear Programming Using MATLAB®*, volume 127. Springer, 2017.

[54] Prasad Raghavendra and David Steurer. November 18, 2008. 2008.

[55] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121:307–335, 2010.

[56] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010.

[57] R Tyrell Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and applications*, 12(6):555–562, 1973.

[58] R Tyrrell Rockafellar. *Convex analysis*. Number 28. Princeton university press, 1970.

[59] R Tyrrell Rockafellar. Penalty methods and augmented lagrangians in nonlinear programming. In *IFIP Technical Conference on Optimization Techniques*, pages 418–425. Springer, 1973.

[60] R Tyrrell Rockafellar. Augmented lagrange multiplier functions and duality in nonconvex programming. *SIAM Journal on Control*, 12(2):268–285, 1974.

[61] RT Rockafellar. New applications of duality in convex programming. In *Proceedings of the 4th Conference of Probability, Brasov, Romania*, pages 73–81, 1971.

[62] Judah Ben Rosen, Olvi L Mangasarian, and Klaus Ritter. *Nonlinear Programming: Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin, Madison, May 4-6, 1970*. Number 25. Elsevier, 2014.

[63] Edward Rothberg. Gurobi optimization: three founders strive to build a different kind of optimization software company. *OR/MS Today*, 41(2):22–25, 2014.

[64] Alan Rothwell. *Optimization methods in structural design*, volume 242. Springer, 2017.

[65] Mark Rubinstein. Markowitz's" portfolio selection": A fifty-year retrospective. *The Journal of finance*, 57(3):1041–1045, 2002.

[66] Stephan Russenschuck. Mathematical optimization techniques. Technical report, CERN, 1999.

[67] Shivani Sanan and Amanpreet Singh. Quasi-newton methods for multivariate unconstrained non-linear optimization problems.

[68] Bib Paruhum Silalahi, Djihad Wungguli, and Sugi Guritman. Steepest descent method with new step sizes. *International Journal of Mathematical and Computational Sciences*, 9(7):378–384, 2015.

[69] Alice Smith and David Coit. Penalty functions. *Handbook of evolutionary computation*, 97(1):C5, 1997.

[70] Jan Snyman. *Practical mathematical optimization*. Springer, 2005.

[71] Porfirio Suñagua and Aurelio Ribeiro Leite Oliveira. A constructive global convergence of the mixed barrier-penalty method for mathematical optimization problems. *Pesquisa Operacional*, 40, 2020.

[72] Joakim Sundnes. *Introduction to Scientific Programming with Python*. Springer Nature, 2020.

[73] Fellar Tyrrell. Convex analysis, 1970.

[74] Hailay Weldegiorgis. *Penalty and Barrier Function Methods for Constrained Optimization Problems*. PhD thesis, Addis Ababa University. College of Natural Sciences. Department of Mathematics., 2010.

[75] Angelika Wiegele. Biq Mac Library—A collection of Max-Cut and quadratic 0-1 programming instances of medium size. Technical report, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria, 2007.

[76] Angelika Wiegele. Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. *Preprint*, 2007.

[77] Xin-She Yang. *Optimization Algorithms*, volume 356, pages 13–31. 01 1970.

[78] Yinyu Ye. Semidefinite programming. 56(2):370–391, 2003.

[79] Changjun Yu. *A study of optimization and optimal control computation: exact penalty function approach*. PhD thesis, Curtin University, 2012.

[80] Xiaopeng Zhao and Jen-Chih Yao. Linear convergence of a nonmonotone projected gradient method for multiobjective optimization. *Journal of Global Optimization*, pages 1–18, 2021.