Republic of Iraq
Ministry of Higher Education
and scientific Research
Babylon University
College of Education for pure sciences
Department of Mathematic

Solving Numerical Optimization Problem By Using Quasi - Newton Methods

A Research

Submitted To the Mathematic Department College of Babylon As partial Fulfillment of the requirement for the Degree of Higher Diploma Education /Mathematic

By
Saad Mohammed Bashan

Supervised By

Dr. Ahmed Sabah Al jilawi

2023 A.D **1444 A.H**

﴿ يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ

وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ وَاللَّهُ

بِمَا تَعْمَلُونَ خَبِيرٌ ﴾

صدق الله العلي العظيم

(المجادلة 1

Supervisor certificate

I certify that the research entitled "**Solving Numerical Optimization Problem By Using Quasi-Newton methods** " for the student **"Saad Mohammed Bashan "** was prepared under my supervision in University of Babylon / college of Education for pure science as a partial requirement for the Degree of Higher Diploma Education /Mathematic.

Signature:

Name : Dr. Ahmed Sabah Al jilawi

Scientific grade :

Date: / / 2023

According to the available recommendation , I forward this project for debate by the examining committee .

Signature:

Name: Dr. Azal Jafar Musa

Head of mathematics Department

Scientific grade: Asst. Prof.

Date: / / 2023

Examining Committee Certification

We certify that we have read this research entitled "**Solving Numerical Optimization Problem By Using Quasi-Newton Methods**" as examining committee examined the student **"Saad Mohammed Bashan"** in its contents and that in our opinion it is adequate for the partial fulfillment of the requirement for the Degree of Higher Diploma Education / Mathematics .

Chairman

Signature:
Name:
Title:
Date: / /2023

Member

Signature:
Name:
Title:
Date: / /2023

Member

Signature
Name:
Title :
Date: / /2023

Approved by the dean of college of education for pure science:

Signature

Name: Dr. Bahaa Hussien Salih Rabee

Title: Professor

Date: / /2023

Scientific super visor's Certification

This is to certify that I have read this thesis, entitled **"Solving Numerical Optimization Problem By Using Quasi-Newton Methods"** and I found that this thesis is qualified for debate.

Signature:

Name:

Scientific grade:

Date: / / 2023

Linguistic super visor's Certification

This is to certify that I have read this thesis, entitled "**Solving Numerical Optimization Problem By Using Quasi-Newton Methods**" and I found that this thesis is qualified for debate.

Signature:

Name: Dr.

Scientific grade:

Date: / /2023

Dedication

Give this humble effort

To my family

To everyone who supported me and

Gave me support

Acknowledgments

I would like to express my thanks to the Merciful God and then thanks " my supervisor" **Dr. Ahmed Sabah Al jilawi** for standing with me and supporting me to bring this work into big success. and thanks to all those who contributed to this work.

Contents

List of Figures

Abstract

Many techniques for solving general nonlinear unconstrained optimization problems involve iteratively minimizing a model function that satisfies certain interpolation conditions. These conditions provide a model that behaves like the objective function in the neighborhood of the current iterate. The model functions often involve second-order derivatives of the objective function, which can be expensive to calculate. The fundamental idea behind quasi-Newton methods is to maintain an approximation to the Hessian matrix. The practical success of quasi-Newton methods has spurred a great deal of interest and research that has resulted in a considerable number of variations of this idea. The analytical difficulties associated with characterizing the performance of these algorithms means there is a real need for practical testing to support theoretical claims. The aim of this work is to investigate the best possible solution, for general nonlinear unconstrained optimization problems and four methods were tested, namely (Newton Methods, Quasi-Newton Methods , The Conjugate Gradient Methods , Steepest Descent Method) also , me compared The results proved that the Quasi-Newton Methods is better in terms of speed and accuracy.

Chapter one

1.1 Applied Mathematics. [1]

Involves the application of mathematics to problems which arise in various areas, e.g., science, engineering or other diverse areas, and/or the development of new or improved methods to meet the challenges of new problems.

The application of mathematics to real-world problems with the dual goal of explaining observed phenomena and predicting new, as yet unobserved, phenomena. Therefore, the emphasis is on both the mathematics, e.g. the development of new methods to meet the challenges of new problems, and the real world.

The problems come from various applications, such as physical and biological sciences, engineering, and social sciences. Their solutions require knowledge of various branches of mathematics, such as analysis, differential equations, and stochastics , utilizing analytical and numerical methods. Very often our faculty members and students interact directly with experimentalists to see their research results come to life.

1.2 Optimization. [2]

The human activities there is a desire to deliver the most with the least. For example in the business point of view maximum profit is desired from least investment; maximum number of crop yield is desired with minimum investment on fertilizers; maximizing the strength, longevity, efficiency, utilization with minimum initial investment and operational cost of various household as well as industrial equipment s

and machineries. To set a record in a race, for example, the aim is to do the fastest (shortest time).

The concept of optimization has great significance in both human affairs and the laws of nature which is the inherent characteristic to achieve the best or most favorable (minimum or maximum) from a given situation. In addition, as the element of design is present in all fields of human activity, all aspects of optimization can be viewed and studied as design optimization without any loss of generality. This makes it clear that the study of design optimization can help not only in the human activity of creating optimum design of products, processes and systems, but also in the understanding and analysis of mathematical/physical phenomenon and in the solution of mathematical problems. The constraints are inherent part if the real world problems and they have to be satisfied to ensure the acceptability of the solution. There are always numerous requirements and constraints imposed on the designs of components, products, processes or systems in real-life engineering practice, just as in all other fields of design activity. Therefore, creating a feasible design under all these diverse requirements/constraints is already a difficult task, and to ensure that the feasible design created is also 'the best' is even more difficult.

1.3 Linear Programming. [3]

Linear programming (LP) is subfield of optimization. Linear programming is a mathematical technique for finding optimal solutions to the problems. Linear Programming deals with the problem of optimizing a linear objective function subject to linear equality and inequality constraints on the decision variables. Linear programming is not a programming language like C++, Java, or Visual Basic, its

mathematical model. A feasible solution is a solution that satisfies all of the constraints. The
feasible set or feasible region is the set of all feasible solutions. Finally, an optimal solution is the feasible solution that produces the best objective function value possible.

1.4 Nonlinear Programming .[4] [5]

The general nonlinear programming (nlp) problem (for continuous variables) is to find $\bar{X}$ so as to optimize $f(\bar{X}), \bar{X} = (x_1, \ldots, x_n) \in R^n$ where $\bar{X} \in \mathcal{F} \subseteq$ S. The set S $\subseteq R^n$ defines the search space and the set $\mathcal{F} \subseteq$ S defines a feasible search space. Usually, the search space S is defined as a n-dimensional rectangle in R^n (domains of variables defined by their lower and upper bounds):

$$l(i) \leq x_i \leq u(i) \quad , 1 \leq i \leq n$$

whereas the feasible set $\mathcal{F} \subseteq$ S is defined by a set of additional m ≥ 0 constraints:

$g_i(\bar{X}) \leq 0$, for j = 1,….. , q, and $h_i(\bar{X}) = 0$, for j = q + 1, ….. ,m.

It is also convenient to divide all constraints into four subsets: linear equations LE, linear inequalities LI, nonlinear equations NE, and nonlinear inequalities NI. Of course, $g_i \in LI \cup NI$ *and* $hj \in LE \cup NE$. In fact, we need not consider linear equations LE, since we can remove them by expressing values of some variables as linear functions of remaining variables and making appropriate substitutions .

Most research on applications of evolutionary computation techniques to nonlinear programming problems has been concerned with

complex objective functions with $\mathcal{F} = \mathrm{S}$. Several test functions used by various researchers during the last 20 years considered only domains of n variables; this was the case with five test functions $F1 - F5$ proposed by De Jong, as well as with many other test cases proposed since then . Only recently several approaches were reported to handle general nonlinear programming problems.

1.5 Quasi-Newton Method. [6]

Quasi-Newton Method is used to solve non-linear optimization Method for multi-variables . This problems as an extension of Newton method is also used to solve second order derivatives of the of the functions in the form of Hessian Matrix. Quasi - Newton objective Method is the most effective method for finding minimizers of a smooth non-linear function when second derivatives are either unavailable or too compute. The algorithm method is used on the functions to difficult to generate solutions and theses solutions

The new point . obtained .be convergent after certain number of iterations by the sum of the previous point and the result is multiplied by the step and the search direction. This process is continued until length

convergent is reached and this method is easy to

1.6 Approximation Methods. [7][8]

Approximation methods are very active area in optimization. Consider the minimizing convex function C : $R^n \rightarrow$ 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 we tried to find:

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

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} .

1.7 Mathematical Algorithm. [9][10]

The word algorithm comes from the name of the ninth-century Muslim Mathematician Abu Abdullah Muhammad ibn Musa al-Khwarizmi. The word algorithm originally referred only to the rules of performing arithmetic using Hindu-Arabic numerals but evolved via European Latin translation of al-Khwarizmi's name into algorithm by the eighteenth century. The use of the word evolved to include all definite procedures for solving problems or performing tasks. In 1939, J. Barkley Rosser (1907-1989) defined the algorithm in an effective mathematical method. It is the sense of a method in which each step is precisely determined and produced, the answer in a finite number of steps. Algorithms find their place in computer programs. The code written in any computer language solves the complex problem in a definite number of steps .

1.8. Classification Of Optimization Problem.

Based on the advanced, we can divide the optimization problems as follows:

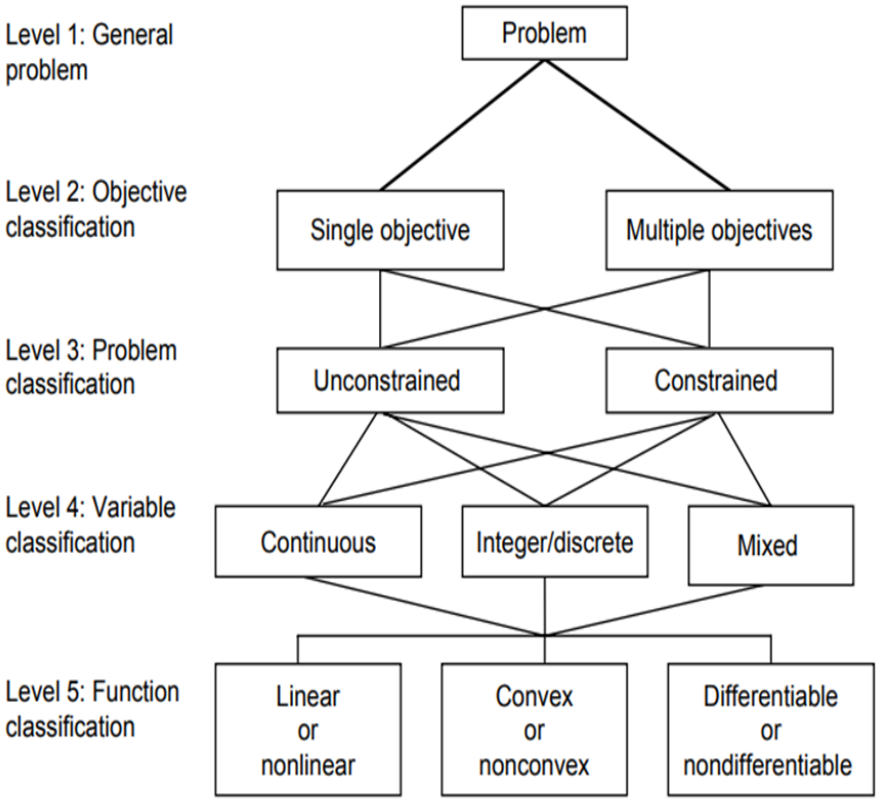


Figure 1.1: Classification of Optimization Problem

Chapter two

Definitions And Properties:

In this section, we offer definitions, notation, and necessary results .

2.1 The Objective Function.[4] [11]

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 R^n and satisfies the constraints of . The set F of all feasible points defines the feasible region of the optimization problem, i.e.,

$S := \{x \in R^n \mid hj(x) = 0 \text{ for } j = 1, 2, \ldots, L, g_i(x) \leq 0 \text{ for } i = 1, 2, \ldots, M\}$.

Optimization problems can be classified according to the type of the objective and constraint functions:

2.1.1 **Linear Programming (LP) problems**, where the objective and constraints are defined by linear functions.

The general form of a linear programming problem:

$$\begin{cases} \text{Minimize } C^T x \\ \text{subject to } Ax = a \\ Bx \leq b \\ x \in R^n \end{cases}$$

Here the objective function $f(x) = C^T x$, where the feasible set

$$X = \{x \in R^n \mid Ax = a, Bx \leq b\}$$

is defined using linear function.

2.1.2 Nonlinear Programming problems

where the objective or at least one of the constraints are defined by nonlinear functions.

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

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

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

$$X = \{ x \in R^n | \, hj(x) = 0 \; for \; j = 1,\dots L; \; gi(x) \le 0 \; for \; i = 1,\dots,M \}.$$

Through out this our interest is in solving Non - linear programming problems by classifying them primarily as unconstrained and constrained optimization problems. Particularly, if the feasible set $X = x \in R^n$ the optimization problem is called an unconstrained optimization problem where as the problems of type are said to be constrained optimization problems. Generally, Optimization problems can be classified as unconstrained optimization problem and constrained optimization problems(see Figure).

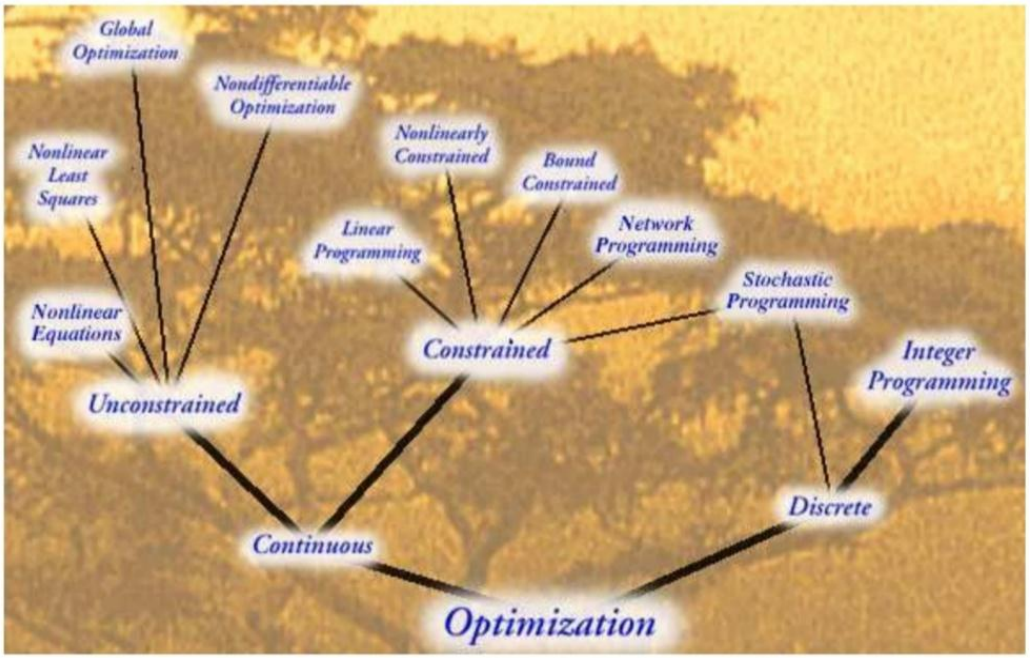


Figure: 2.1 constrained optimization problems

2.2 Constraint Optimization[12]

The general form optimization problems constraint with equality and inequality

$$\begin{cases} \text{minimize } f(X) \\ \text{subject to } g(X) = 0 \\ h(X) \leq 0 \\ X > 0 \end{cases}$$

Solution methods for constrained minimization problem are .

2.3 Feasible Region . [13]

The common region determined by all the constraints including non -negative constraints x, y >0 of a linear programming problem is called the feasible region for the problem. The region other than the feasible region is called an infeasible region. The feasible region is always a convex polygon

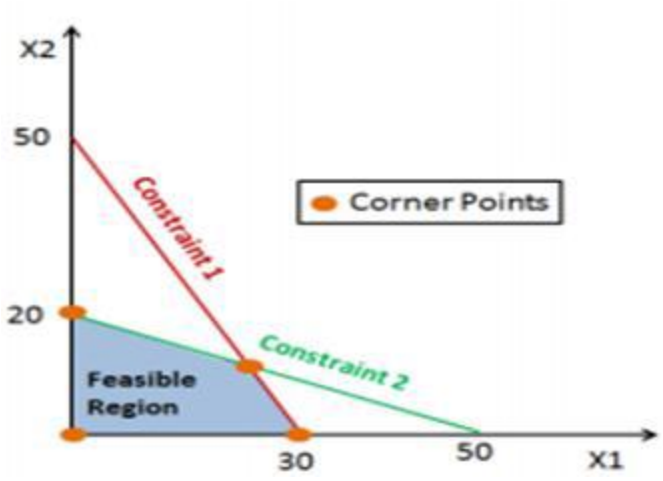


Figure: 2.2 Feasible Region

2.4 Bounded and Unbounded Region .[13]

A feasible region of a system of linear inequalities is said to be bounded, if it can be enclosed within a circle. Otherwise, it is called unbounded.

2.5 Feasible Solutions. [11]

Points within and on the boundary of the feasible region represent feasible solutions of the constraints. Any point outside the feasible region is called an infeasible solution.

2.6 Optimal Feasible Solution . [11]

Any point in the feasible region that gives the optimal value of the objective function is called the optimal feasible solution.

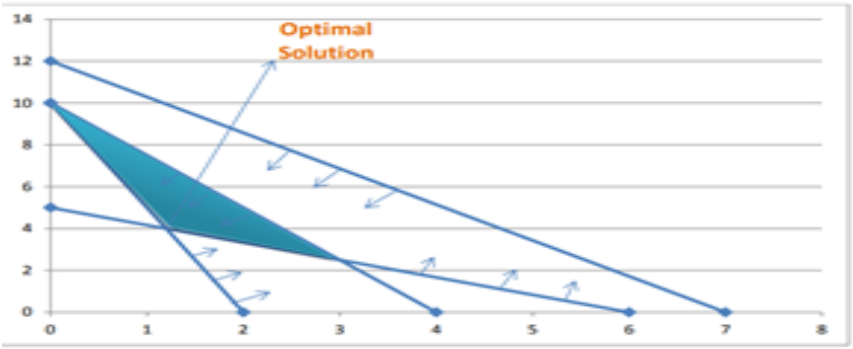


Figure:2.3 Optimal Feasible Solution

2.7 Definition [14]

Let $x^* \in S$ is a local minimizer of f over S if

$\exists \varepsilon > 0$ such that $f(x^*) \leq f(x), for\ all\ x \in S \cap B\varepsilon(x^*)$.

Where $B\varepsilon(x^*) = \{x \in R^n | |x - x^*| \leq \varepsilon\}$, and $|.|$ denotes the usual Euclidean norm. A point x^* is called a strict local minimizer if

$f(x^*) < f(x), for\ all\ x \in S \cap B\varepsilon(x^*) \backslash \{x^*\}$.

2.8 Definition .[14]

Let $x^* \in S$. We say that $x^*$is a global minimizer of f over S if f attains its lowest value over S at x^* (2.4).

$$f(x^*) \leq f(x), for\ all\ x \in S.$$

A point x∗ is called a strict global minimizer if

$$f(x^*) < f(x), forall\ x \in S\backslash\{x^*\}$$

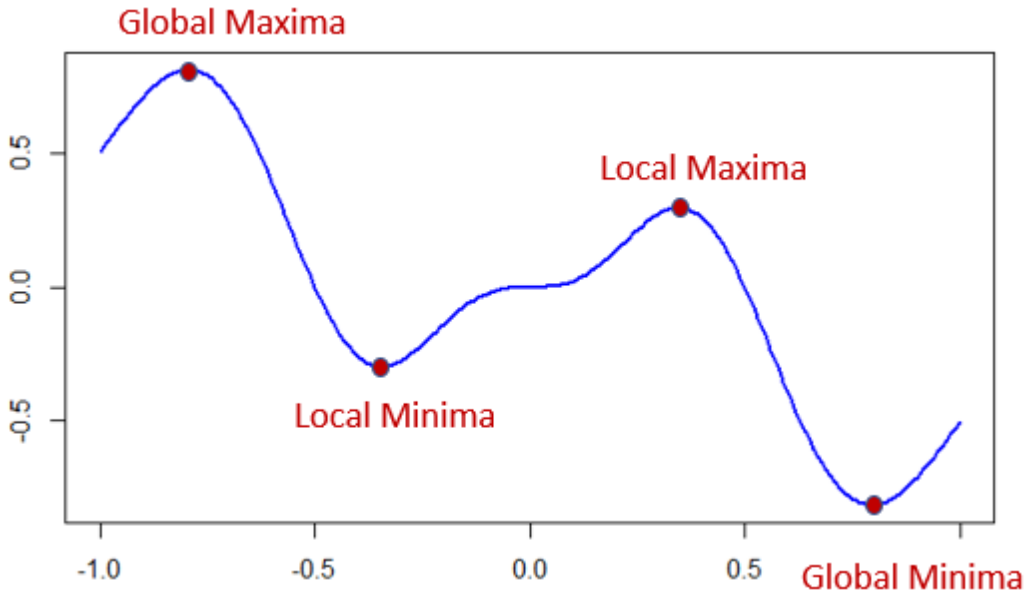


Figure:2.4 Local and Global Minima

2.9 Speed of Convergence .[15]

Assume that an optimization algorithm generates a sequence $\{x^k\}$ which converges to x^*, interesting to study how fast does the sequence convergence to x^*. Definition. The sequence $\{x^k\}$ converges to x^* as $\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^p} = \beta,\ \beta \leq \infty$,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 $\|x^{k+1} - x^*\|$ represent the distance

between x^{k+1} and x^*while $\|x^k - x^*\|^p$represent the distance between x^k and x then seeking to find x^{k+1} closed to x^*, so asymptotically, can see

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

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

2.10 Definition [16]

Let $f : R^n \to R$ and $x \in R^n$. Then the Partial derivative of f at x with respect to x_i is defined as

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

where e_i is ith unit vector. The Gradient of f at x is defined as the column vector

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

The Hessian matrix is defined as the n × n symmetric matrix

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1 \partial x_1} & \frac{\partial^2 f_1}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_1}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f_n}{\partial x_n \partial x_1} & \frac{\partial^2 f_n}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f_n}{\partial x_n \partial x_n} \end{bmatrix}$$

The Directional derivative of the function f at x in the direction d given by

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

We say the function f is Differentiable at x.

2.11 Symmetric Matrices .[17]

The matrices in numerical optimization that are frequently encountered are symmetric.

Let $X = [xij]$ which is square matrix.So the matrix X is said Symmetric if $X = X^T$ $(i.e., xij = xji)$. Let S^n be of all the symmetric $n \times n$ matrices:

$$S^n = \{X \in R^{n\times n} : X = X^T\}$$

The notation $X > 0$ $(X \geq 0)$ which is X said to be symmetric and Positive definite (semide finite); for all $u \in R^n$($u^T X u = \sum x_{ij} u_i u_j \geq 0$ for all $u \in R^n$ Furthermore, we let S_+^n denote the set of symmetric positive semi definite matrices:

. $S_+^n = \{ X \in R^n : X \geq 0 \}$

2.12 Convex Sets and Convex Functions

Definition 2.12.1 [18] Convex Set :If the line segment between any two points in C ,lies in C. Then The set C is Convex Set

$$(\theta X_1 + (1 - \theta)X_2 \in C) ,\forall (X_1, X_2) \in C, \forall \theta \in [0,1]$$

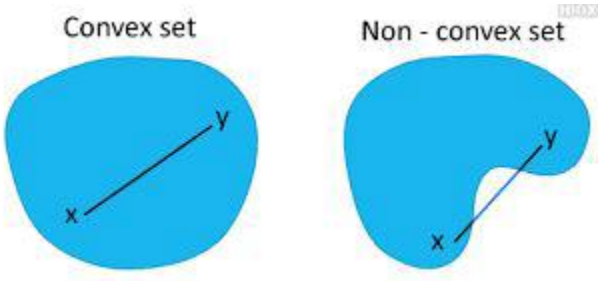


Figure:2.5 Convex Set

Example 2.1

$\Omega 1 = \{z \in R^n : \| z \| \leq 1\}$

For any $z, u \in \Omega 1, \lambda \in (0,1)$ then $\| z \| \leq 1$, $\| u \| \leq 1$

we have

$$\| \lambda z + (1 - \lambda)u \| \leq \| \lambda z \| + \| (1 - \lambda)u \| = \lambda \| z \| + (1 - \lambda)$$

$$\| u \| \leq \lambda(1) + (1 - \lambda)1 = 1$$

2.12.2 Definition. [18] Convex Function: A function f : $R^n \longrightarrow$ R is Convex, if for

every $x_1, x_2 \in R^n, 0 \leq t \leq 1$ the inequality

$$f(t\, x_1 + (1 - t)\, x_2) \leq tf\,(x_1) + (1 - t)\, f(x_2)$$

If the above inequality is true as a strict inequality for all $x_1 \neq x_2$ and for all

$$t \in (0, 1),$$

then f is called a strictly convex function

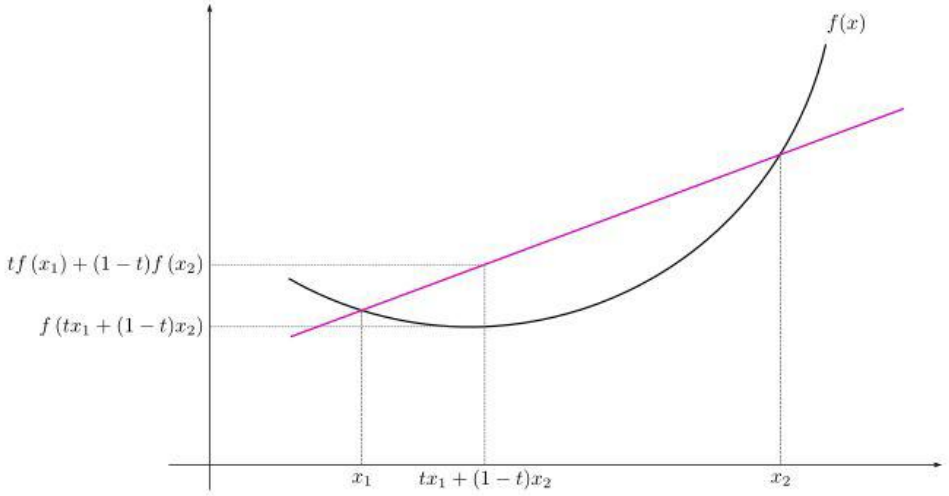


Figure: 2.6 The General Convex Function

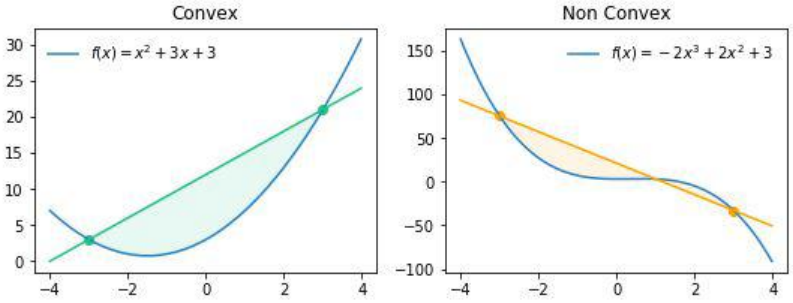


Figure:2.7 Convex Function

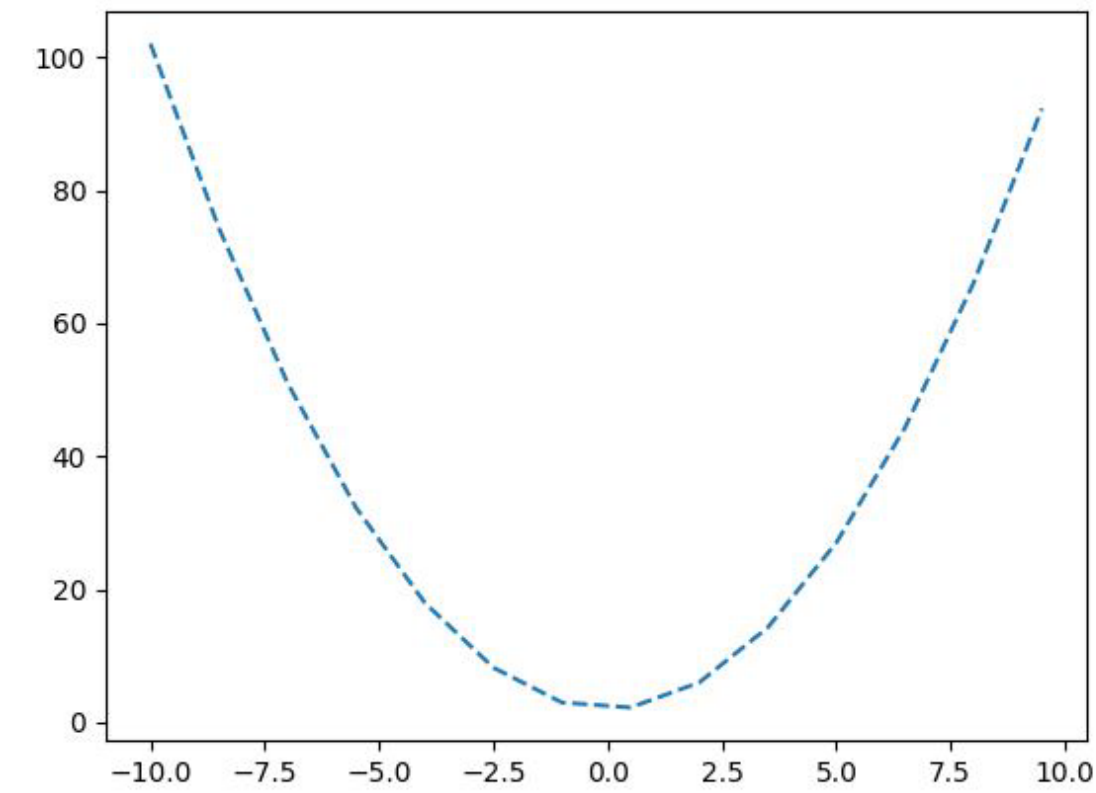
Example2.2. Find the optimal solution to unconstrained optimization problem by using python

$$f(x) = x^2 + 2$$

```
# plot a convex target function
from numpy import arange
from matplotlib import pyplot
# objective function
def objective(x):
 return ( x)** 2 + 2
# define range
r_min, r_max = -10.0, 10.0
# prepare inputs
inputs = arange(r_min, r_max, 1.5)
# compute targets
targets = [objective(x) for x in inputs]
# plot inputs vs target
pyplot.plot(inputs, targets, '--')

pyplot.show()
```

Code 1:Use Python to prove the function is convex function



Convex Function 3

2.13 Optimization Algorithm . [2]

Optimization algorithms execute as equence of iterations to solve an optimization problem. The algorithm starts with an initial guess point and creates a sequence of points (k) that converges to an optimal point to satisfy certain optimality conditions .The rate at which the iterations approach the optimal point is called the convergence rate of the algorithm. The algorithm converges faster if it takes a short time to obtain the optimal solution

2.14 The general Algorithm formula .[19][20]

An algorithm is a collection of sequential instructions that is used to solve com- putational problems. The method involved in the algorithm is either sequential or iterative. The instructions used in the algorithm should be well-defined so that these can be implemented as a computer program. A computer program is a collection of sequential instructions written by a computer programmer in any programming language that performs a specific task when executed by a computer. Optimization algorithms execute a sequence of iterations to solve an Optimization problem. The algorithm starts with an initial guess point and creates a sequence of points $x^{(k)}$ that converges to an optimal point to satisfy certain optimality conditions. The rate at which the iterations approach the optimal point is called the convergence rate of the algorithm. The algorithm converges faster if it takes a short time to obtain the optimal solution

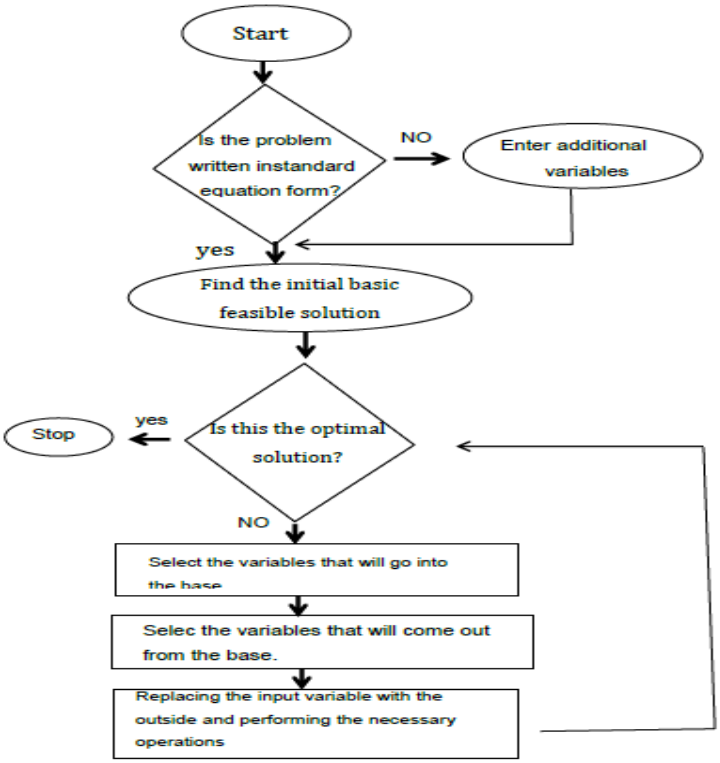


Figure:2.8 The general Algorithm formula

2.15 Optimality Conditions [21]

Optimality conditions characterize analytical properties of local optima and are fundamental for developing numerical methods. Necessary optimality conditions aim to characterize a point x^*, assuming that $x*$ is a local minimizer of the considered problem. Such conditions allow one to determine a set of candidate points that may need to be further analyzed in order to find out if they are local minimizers. In some cases, such analysis can be done using sufficient optimality conditions, providing properties that, if satisfied guarantee that a candidate point $x^*$is alocal minimizer.

2.15 .1 First-order necessary condition . [21]

The first-order necessary conditions (FONC) for unconstrained optimization can be viewed as a special case of the following optimality conditions for a more general set-constrained optimization problem.

Theorem (FONC for a Set-Constrained Problem)

If $f : X \rightarrow R$, where $X \subseteq R^n$, is a continuously differentiable function on an open set containing X, and x^* is a point of local minimum for the set-constrained problem

$$\text{minimize } f(x) \qquad (2.1)$$

subject to $x \in X$

Then for any feasible direction d at x^*, the rate of increase of f at x^*in the direction d is nonnegative: $\nabla f(x^*)^T d \geq 0$.

Proof. Let $x^*, \in X$ be a local minimizer of (2.1). Consider a feasible direction d at x^*, such that $\|d\|= 1$. Using Taylor's theorem for *f*, x^* and $x = x^* + \alpha d$, where $\alpha > 0$, we have :

$$f(x) = f(x^*) + \nabla f(x^*)^T \alpha d + o(\|\alpha d\|)$$
$$= f(x^*) + \alpha \nabla f(x^*)^T d + o(\alpha)$$

Since x^* is a local minimizer of (2.1), from the above equation we have:

$$\frac{f(x) - f(x^*)}{\alpha} = \nabla f(x^*)^T d + \frac{o(\alpha)}{\alpha}$$

Since $f(x) - f(x^*) \geq 0$, $\alpha > 0$, $\frac{o(\alpha)}{\alpha} \to 0$, $\alpha \to 0$ we obtain $\nabla f(x^*)^T d \geq 0$

Indeed, if we assume $f(x^*)^T d < 0$ then selecting $\in$ such that $f(x^*)) \leq f(x)$ for any $x \in B(x^*, \in)$ and the error term $\frac{o(\alpha)}{\alpha}$ in (13.3) is less than $|f(x^*)^T d|$ (such $\in$ exists since $x^*$is a local minimizer and by definition of $o(\alpha)$) would result in $f(x) - f(x^*) < 0$. We obtain a contradiction with the assumption that $f(x^*)) \leq f(x)$ for any $x \in B(x^*, \in)$ As a corollary.

Corollary If x^* is an interior point of X and a local minimizer, then $\nabla f(x^*) = 0$.

Proof. If $x^*$is an interior point of *X*, then any $d \in R^n$ is a feasible direction at x^*. Thus, using Theorem 13.1 for an arbitrary direction $d \neq 0$ and its opposite direction $-d$ we have: $\nabla f(x^*)^T d \geq 0$, $\nabla f(x^*)^T (-d) \geq 0$, therefore, $\nabla f(x^*)^T d = 0$. In particular, if we use $d = [d_1, \ldots, d_n]^T$ with $d_j = 1$, for an arbitrary $j \in \{1, \ldots, n\}$ and $d_i = 0$, for all $i \neq j$, this implies that thei^{th} component of $\nabla f(x^*)$ equals 0. Since this is the case for each *j*, we have $\nabla f(x^*) = 0$.

In the case of an unconstrained problem, $X = R^n$ and any point $x \in R^n$ is an interior point of X. Thus, we obtain the following FONC for unconstrained optimization.

Theorem (FONC for an Unconstrained Problem)

If x^* is a point of local minimum of the unconstrained problem

minimize f(x),

where $f : R^n \rightarrow R$ is continuously differentiable, then $\nabla f(x^*) = 0$.

Definition 2.15.1.1 (Stationary Point) [21](A point x^* satisfying the FONC for a given problem is called a stationary point for this problem.

The following example shows that the FONC is not sufficient for a local minimizer.

Example 2.3 Applying the FONC to $f(x) = x^3$, we have

$$f'(x) = 3x^2 = 0 \quad \leftrightarrow \quad x = 0$$

But, obviously, $x = 0$ is not a local minimizer of $f(x)$. In fact, for any given point x and any small $\epsilon > 0$, there always exist x_*, $x^* \in B(x^*, \in)$ such that

$f(x_*) < f(x) < f(x^*)$, so f(x) does not have any local or global minimum or

maximum.

Next, we prove that a point satisfying the FONC is, in fact, a global minimizer if a problem is convex. The proof is based on the first-order characterization of a convex function. Consider a convex problem $\min_{x \in X}$ f(x). For any $x, y \in X$, a differentiable convex function satisfies

$$f(y) \geq f(x) + \nabla f(x)^T (y - x),$$

hence, if for $x = x^*$: $\nabla f(x^*) = 0$, we obtain

$$f(y) \geq f(x^*)$$

For any $y \in X$. Thus, x^* is a point of global minimum for this problem.

This implies that the FONC in the case of an unconstrained convex problem becomes a sufficient condition for a global minimizer.

Theorem (Optimality Condition for a Convex Problem)

For a convex unconstrained problem minimize f(x), where f : $R^n \rightarrow$R is differentiable and convex, x^* is a global minimizer if and only if

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

2.15 . 2 Second-order optimality conditions . [21]

Next, we derive the second-order necessary conditions (SONC) and second order sufficient conditions (SOSC) for a local minimizer of an unconstrained problem. We assume that the objective function

$f \in C^2(R^n)$, i.e., f is twice continuously differentiable.

Theorem (SONC for an Unconstrained Problem)

If $x^* \in R^n$ *is a local minimizer for the problem* $\min_{x \in R^n} f(x)$, where $f(x) \in C^2(R^n)$, then $\nabla^2 f(x^*)$ is positive semi definite.

Theorem (SOSC for an Unconstrained Problem)

If x^* satisfies the FONC and SONC for an unconstrained problem $\min_{x \in R^n} f(x)$ and $\nabla^2 f(x^*)$ is positive definite, then x^* is a point of strict local minimum for this problem.

Proof. We assume that x^* satisfies the FONC and SONC. Then, for any $d \in R^n$ with $\|d\|= 1$ and any $\alpha > 0$ we have

$$\frac{f(x^* + \alpha d) - f(x^*)}{\alpha^2} = \frac{1}{2} d^T \nabla^2 f(x^*) d + \frac{o(\alpha^2)}{\alpha^2}.$$

If we additionally assume that $\nabla^2 f(x^*)$ is positive definite, then by Rayleigh's inequality,

$$\frac{1}{2} d^T \nabla^2 f(x^*) d \geq \lambda_{min} \|d\|^2 = \lambda_{min} > 0$$

Here λ_{min} denotes the smallest eigenvalue of $\nabla^2 f(x^*)$. Thus, there exists $\in > 0$ such that for any $\alpha \in (0, \in)$ we have $f(x^* + \alpha d) - f(x^*) > 0$. Since d is an arbitrary direction in R^n, $x^*$is a point of strict local minimum by definition. Thus, the FONC, SONC, together with the positive definiteness of $\nabla^2 f(x^*)$, constitute the sufficient conditions for a strict local minimizer.

Example2.4 Consider the function f(x) = $x_1^3 - x_2^3 + 3x_1x_2$. We apply the optimality conditions above to find its local optima. FONC system for this problem is given by

$$\nabla f(x) = \begin{bmatrix} 3x_1^2 + 3x_2 \\ -x_2^2 - 3x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

.

From the second equation of this system we have $x_1 = x_2^2$. Substituting for x_1 in the first equation gives $x_2^4 + x_2 = 0$, which yields $x_2 = 0$ or

$x_2 = -1$. The corresponding stationary points are $\hat{x} = [0, 0]^T$ and $\check{x} = [-1, -1]^T$, respectively. The Hessian of $f(x)$ is

$\nabla^2 f(x) = \begin{bmatrix} 6x_1 & 3 \\ 3 & -6x_2 \end{bmatrix}$ $\nabla^2 f(\hat{x}) = \begin{bmatrix} 0 & 3 \\ 3 & 0 \end{bmatrix}$ $\nabla^2 f(\check{x}) = \begin{bmatrix} 6 & 3 \\ 3 & 6 \end{bmatrix}$

Since the determinant of $\nabla^2 f(\hat{x})$ equals −9, the Hessian is indefinite at $\hat{x}$, and from the SONC, $\hat{x}$cannot be a local optimum. On the other hand, $\nabla^2 f(\check{x})$ is positive definite, hence $\check{x}$is a strict local minimum by the SOSC. Note that

If we fix $x_2 = 0$, the function f is then given by x_1^3 , which is unbounded from below or above. Thus, f has no global minima and no local or global maxima. Its only local minimum is $\check{x} = [-1, -1]^T$

2.16 Continuity and differentiability .[21]

Let $X \subseteq R$. Given a real function $f : X \to R$ and a point $x_0 \in X$, f is called continuous at x_0 if $\lim_{x \to x_0} f(x) = f(x_0)$, and f is called differentiable at x_0 if there exists the limit $\acute{f}(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$, in which case $\acute{f}(x_0)$ is called the first-order derivative, or simply the derivative of f at x_0. If f is differentiable at x_0, then it is continuous at x_0. The opposite is not true in general; in fact, there exist continuous functions that are not differentiable at any point of their domain. If f is continuous (differentiable) at every point of X, then f is called continuous (differentiable) on X. The set of all continuous functions on X is denoted by C(X). The k^{th} order derivative$f^{(k)}(x_0)$ of f at $x_0$can be defined recursively as the first-order derivative of $f^{(k-1)}$at $x_0$for k ≥ 1, where $f(0) \equiv f$. If $\acute{f}$ is a continuous function at x_0, then we call f continuously differentiable or smooth at x_0. Similarly, we call f k-times continuously differentiable at x_0 if $f^{(k)}$ exists and is continuous at x_0. The set of all k times continuously differentiable functions on X is denoted by

C^K(C). Then C^K (X) $\subset C^{(K-1)}(X), k \geq 1$ (where $C^{(0)}(X) \equiv C(X)$).

Chapter Three

3.1 Newton's Method .[22][23][24]

Newton's method was invented by Newton to solve the nonlinear one-dimensional problem. Later it was extended to solve multivariable nonlinear Optimization problems. It is well known that the method of steepest descent uses only the first-order derivative (gradient). But, the use of higher derivatives may result in better performing method than the steepest descent method. Newton's method uses both first and second derivatives and gives better performance than the steepest descent method if the initial point is closer to the minimizer. Newton's method has a powerful convergence property called quadratic convergence. Prerequisite knowledge of Taylor's series, partial differentiation, and the concept of Hessian is necessary to understand this technique. The idea is that for a given starting point, we construct a quadratic approximation to the objective function that matches the first and second derivative values at the given initial point. Thereafter, we minimize the approximate (quadratic) function instead of the original objective function. Newton's method is generally used as a hybrid method in conjunction with other methods .

Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be twice continuously differentiable function. We obtain a quadratic approximation of f using the Taylor series expansion of f about the initial point $x^{(0)}$, neglecting the terms of order 3 and higher.

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^T \nabla f(x^{(0)}) + \frac{1}{2}(x - x^{(0)})^T F(x^{(0)})(x - x^{(0)}) := q(x), \quad (3.1)$$

where $F(x^{(0)})$ is Hessian at $x^{(0)}$. The quadratic equation is presented as follows:

$$q(x) = f(x^{(0)}) + (x - x^{(0)})\nabla f(x^{(0)}) + \frac{1}{2}(x - x^{(0)})^T F(x^{(0)})(x - x^{(0)}). \quad (3.2)$$

Our intention is to find minimizer of $f(x)$. Here, we minimize $q(x)$ rather than $f(x)$. Using first-order necessary condition to $q(x)$ to get

$$\nabla q(x) = 0,$$

That is,

$$\nabla f(x^{(0)}) + F(x^{(0)})(x - x^{(0)}) = 0$$

Therefore,

$$F(x^{(0)})(x - x^{(0)}) = -\nabla f(x^{(0)}),$$

That is,

$$(x - x^{(0)}) = -\left(F(x^{(0)})\right)^{-1} \nabla f(x^{(0)}).$$

If $F(x^{(0)}) > 0$, then $q(x)$ achieves a minimum at $x^{(0)}$ and x can be obtained as

$$x = x^{(0)} - (F(x^{(0)})^{-1} \nabla f(x^{(0)}). \quad (3.3)$$

For general current point, the Newton's method is given as follows:

$$x^{(k+1)} = x^{(k)} - \left(F(x^{(k)})\right)^{-1} \nabla f(x^{(k)}). \quad (3.4)$$

For simplicity, if we use the notation $g^{(k)} = \nabla f(x^{(k)})$ then we get the following recursive formula for Newton's method:

$$\boldsymbol{x^{(k+1)} = x^{(k)} - \left(F(x^{(k)})\right)^{-1} g^{(k)}} \quad \textbf{(3.5)}$$

Example3.1 . Solving the problem by use Newton's method to minimize

$$f(x) = \frac{1}{4}x^3 + \frac{3}{4}x^2 - \frac{3}{2}x - 2$$

```
 from scipy import misc
def NewtonsMethod(f, x, tolerance=0.000001):
   while True:
      x1 = x - f(x) / misc.derivative(f, x)
      t = abs(x1 - x)
      if t < tolerance:
         break
      x = x1
   return x
def f(x):
   return (1.0/4.0)*x**3+(3.0/4.0)*x**2-(3.0/2.0)*x-2
x = 4
x0 = NewtonsMethod(f, x)
x1=NewtonsMethod(f, x)
x2= NewtonsMethod(f, x)
x3=  NewtonsMethod(f, x)
x4=  NewtonsMethod(f, x)
print('x: ', x)
print('x0: ', x0)
print("f(x0) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x1: ', x1)
print("f(x1) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x2: ', x2)
print("f(x2) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x3: ', x3)
print("f(x3) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x4: ', x4)
print("f(x4) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
#!/usr/bin/env python
from pylab import *
t = arange(-6.0, 4.0, 0.01)
s= t*t*t/4.0+3.0*t*t/4.0-3*t/2.0-2.0
ax = subplot(111)
ax.plot(t, s)
ax.scatter([-4,-1,2],[0,0,0])
ax.grid(True)
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.set_xlim(-6,6)
ax.set_ylim(-20,20)
text(-3.0, 12,
    r"$f(x)=(1/4)*X^3+(3/4)*X^2-(3/2)*X-2$", horizontalalignment='center',
    fontsize=8)
plt.title("How to implement the Newton's method using python \n for finding the
zeroes of a real-valued function",fontsize=10)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.savefig('NewtonMethodExemple.png')
show()
```

Code 1:Solving newton method by python

(The Results of Solving the Above Example are the Application of Newton's Method)

Iteration	x_n	$f(x_n)$
0	x_0 **=2.0000002745869883**	$f(x_0) = 1.2356416165815176e-06$
1	x_1**=2.0000002745869883**	$f(x_1) = 1.2356416165815176e-06$
2	$x_{2=}$**2.0000002745869883**	$f(x_2) = 1.2356416165815176e-06$
3	x_3**=2.0000002745869883**	$f(x_3) = 1.2356416165815176e-06$
4	$x_{4=}$**2.0000002745869883**	$f(x_4) = 1.2356416165815176e-06$
The required root is 2.0000002745869883		

We determined the maximum and minimum values of the functions and their roots using Newton's method. When the original function has roots of the value of the function equal to zero, it can use Newton's method up to the derivative function to find its origins, starting with the first approximation. Then, by combining the first and second derivatives, we can create a series of approximations x_1, x_2, x_3, x_4 using the tangent lines of the f graph; we might get four iterations. The value becomes an as the number of iterations increases that remains constant (2.0000002745869883).

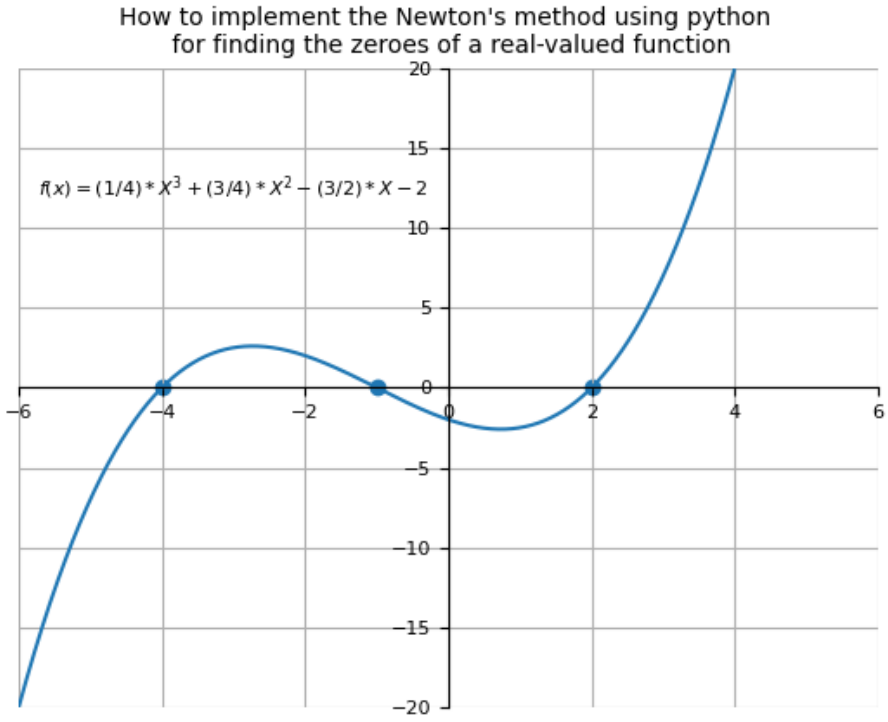


Figure:3.1 Solution result base on newton method In the above diagram ,of the x-axis is computed the root value as (4,−1, 2), and we choose the root is (2)

3.2 Quasi-Newton Methods [23][24].

The Quasi-Newton methods do not compute the Hessian of nonlinear functions. The Hessian is updated by analyzing successive gradient vectors instead. The Quasi−Newton algorithm was first proposed by William C. Davidon, a physicist while working at Argonne National Laboratory, United States in 1959. In Newton's method, we require to compute the inverse of the Hessian at every iterations which is a very expensive computation. It requires an order n cube effort if n is the size of the Hessian. These drawbacks of Newton's method gave motivation to develop the Quasi-Newton methods. The idea of Quasi-Newton method

is that to approximate the inverse of Hessian by some other matrix which should be positive definite so that we can get a good approximation of the Hessian inverse at a given iteration. This saves the work of computation of second derivatives and also avoids the difficulties associated with the loss of positive definiteness. This approximate matrix is updated on every iteration so that as the search direction proceeds the second-order derivative information improves.

3.3 Basics of Quasi-Newton Methods [23][24]

Newton's method is best for quadratic function and it is very good for nonquadratic provided its convergence is of quadratic order. But it fails if the starting point is not sufficiently close to the minimizer x^*, in such cases, it may not converge. Recall the Newton's method

$$x^{(k+1)} = x^{(k)} - \left(F\left(x^{(k)}\right)\right)^{-1} g^{(k)} \qquad (3.6)$$

Note that $\left(F\left(x^{(k)}\right)\right)^{-1}$ is evaluated at every step and then search direction

$$d^{(k)} = -\left(F\left(x^{(k)}\right)\right)^{-1} g^{(k)}$$

is computed. In that case, it may not be a descent method, that is, $f(x^{(k+1)}) \nless f(x^{(k)})$. It is a computational drawback of Newton's method. Therefore, we may try to guarantee that the algorithm has the descent property by doing the following modification:

$$x^{(k+1)} = x^{(k)} - \alpha_k \left(F\left(x^{(k)}\right)\right)^{-1} g^{(k)}, \qquad (3.7)$$

Where α_k , is chosen so that we should have $f(x^{(k+1)}) < f(x^{(k)})$. We apply any line search methods in the direction dit to find the value of a. To avoid the computation of $\left(F\left(x^{(k)}\right)\right)^{-1}$, the Quasi−Newton methods use an approximation in place of $\left(F\left(x^{(k)}\right)\right)^{-1}$.This approximation is updated at every step so that it will have at least some properties of $\left(F\left(x^{(k)}\right)\right)^{-1}$. To get some idea about the properties that an approximation to $\left(F\left(x^{(k)}\right)\right)^{-1}$should satisfy, consider the following formula:

$$x^{(k+1)} = x^{(k)} - \alpha H_k g^{(k)} \qquad (3.8)$$

where H_k, is $n \times n$ real matrix and α is a positive search parameter. Expand f about $x^{(k)}$, we get

$$f\left(x^{(k+1)}\right) = f\left(x^{(k)}\right) + \left(g^{(k)}\right)^T \left(x^{(k+1)} - x^{(k)}\right) + o\left(\left\|x^{(k+1)} - x^{(k)}\right\|\right) \qquad (3.9)$$

Using (3.8) in (3.9), we get

$$f\left(x^{(k+1)}\right) = f\left(x^{(k)}\right) - \alpha\left(g^{(k)}\right)^T H_k g^{(k)} + o\left(\left\|H_k g^{(k)}\right\|\alpha\right). \qquad (3.10)$$

As $\alpha \to 0$, the second term on the right-hand side of the above equation dominates the third term. Thus, to guarantee a decrease in f for small α , we have to obtain

$$\left(g^{(k)}\right)^T H_k g^{(k)} > 0.$$

It is true only when H_k , is positive definite. For this, we have already presented the proof of the following proposition.

Proposition $f \in C^1$, $x^{(k)} \in \mathbb{R}^n$, $g^{(k)} = \nabla f\left(x^{(k)}\right) \neq 0$, and H_k, is an $n \times n$ real symmetric positive definite. If

$$x^{(k+1)} = x^{(k)} - \alpha_k H_k g^{(k)},$$

where $\alpha = \arg\min \alpha \geq 0 f(x^{(k)} - \alpha H_k g^{(k)})$ then

$$\alpha_k > 0, and\ f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right).$$

We use objective function and its gradient to construct an approximation for the inverse of the Hessian matrix. In the next section, we present the method to choose H_k such that iteration may be carried out without any evaluation of the Hessian and solution of set of linear equations.

3.4 Approximating the Inverse Hessian[23][24]

Let $H_0. H_1, H_2, \ldots$ be successive approximation of the inverse $F\left(x^{(k)}\right)^{-1}$. The Hessian $F(x)$ of objective function f is constant and independent of x. Consider a quadratic objective function with Hessian $F(x) = Q$ for all x, where $Q = Q^T$. Then,

$$g^{(k+1)} - g^{(k)} = Q\left(x^{(k+1)} - x^{(k)}\right).$$

Set $\Delta g^{(k)} = g^{(k+1)} - g^{(k)}$ and $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$. Therefore,

$$\Delta g^{(k)} = Q\Delta x^{(k)}$$

That is,

$$\Delta x^{(k)} = Q^{-1}\Delta g^{(k)}.$$

We can also write the above equality in the following form:

$$Q^{-1}\Delta g^{(i)} = \Delta x^{(i)}\ ,\ 0 \le i \le k\ . \qquad (3.11)$$

We also impose the requirement that the approximation H_{k+1} satisfy

$$H_{k+1}\Delta g^{(i)} = \Delta x^{(i)}\ ,0 \le i \le k\ .$$

If n steps are involved, then moving in n directions $\Delta x^{(0)}, \Delta x^{(1)}, \ldots, \Delta x^{(n-1)}$ gives

$$\begin{aligned} H_n\Delta g^{(0)} &= \Delta x^{(0)}, \\ H_n\Delta g^{(1)} &= \Delta x^{(1)}. \\ &\vdots \\ H_n\Delta g^{(n-1)} &= \Delta x^{(n-1)}. \end{aligned}$$

We can represent the set of above equations as

$$H\left[\Delta g^{(0)}, \Delta g^{(1)}, \cdots, \Delta g^{(n-1)}\right] = \left[\Delta x^{(0)}, \Delta x^{(1)}, \cdots, \Delta x^{(n-1)}\right] \quad (3.12)$$

Since $\Delta g^{(i)} = Q\Delta x^{(i)}$, then

$$Q\left[\Delta x^{(0)}, \Delta x^{(1)}, \cdots, \Delta x^{(n-1)}\right] = \left[\Delta g^{(0)}, \Delta g^{(1)}, \cdots, \Delta g^{(n-1)}\right],$$

That is,

$$Q^{-1}\left[\Delta g^{(0)}, \Delta g^{(1)}, \cdots, \Delta g^{(n-1)}\right] = \left[\Delta x^{(0)}, \Delta x^{(1)}, \cdots, \Delta x^{(n-1)}\right].$$

If $\left[\Delta g^{(0)}, \Delta g^{(1)}, \cdots, \Delta g^{(n-1)}\right]$ is nonsingular, then Q^{-1} is determined uniquely after n steps through

$$Q^{-1} = H_n = \left[\Delta x^{(0)}, \Delta x^{(1)}, \cdots, \Delta x^{(n-1)}\right]\left[\Delta g^{(0)}, \Delta g^{(1)}, \cdots, \Delta g^{(n-1)}\right]^{-1}$$

Under these circumstances, we conclude that if H_n satisfies

$$H_n \Delta g^{(i)} = \Delta x^{(i)},$$

where $0 \leq i \leq n-1$ then the algorithm

$$x^{(k+1)} = x^{(k)} - \alpha_k H_k g^{(k)}, \qquad (3.13)$$

where $\alpha_k = \arg min\ \alpha \geq 0$, $f\left(x^{(k)} - \alpha H_k g^{(k)}\right)$, guarantees to solve the quadratic functions of n variables in $n+1$ steps because of the updating next point as

$$x^{(n+1)} = x^{(n)} - \alpha_n H_n g^{(n)}$$

which is equivalent to Newton's algorithm.

The above discussion reflects the basic idea behind the Quasi-Newton algorithms. In fact, we present a result which shows that such algorithms find the minimizer of quadratic functions of n variables in at most n iterations. Although, the Quasi-Newton algorithms have the following form:

$$d^{(k)} = -H_k g^{(k)},$$

$$\alpha_k = \arg\min_{\alpha \geq 0} f\left(x^{(k)} + \alpha d^{(k)}\right) ,$$

and

$$x^{(k+1)} = x^k + \alpha_k d^{(k)} \quad (3.14)$$

The matrices $H_0., H_1$, …. are symmetric. For the quadratic functions, these matrices should satisfy

$$H_{k+1}\Delta g^{(i)} = \Delta x^{(i)}, \quad 0 \leq i \leq k ,$$

where

$$\Delta x^{(i)} = x^{(i+1)} - x^{(i)} = \alpha_i d$$

and

$$\Delta g^{(i)} = g^{(i+1)} = Q\Delta x^{(i)}$$

Note that the Quasi−Newton methods are also conjugate direction methods which can be seen .

Theorem[21] Consider a Quasi−Newton algorithm applied to a quadratic function with Hessian $Q = Q^T$ such that

$$H_k + \Delta g^{(i)} = \Delta x^{(i)} , \quad 0 \leq i \leq k , 0 \leq k \leq n-1 ,$$

where $H_{k+1} = (H_{k+1})^T, if\ \alpha_i \neq 0$ where $0 \leq i \leq k$, then directions $d^{(0)}, d^{(1)}, \ldots . , d^{(k+1)})$ are $Q - conjugate.$

Proof We proceed by mathematical induction, we begin with $k = 0$: Since

$d^{(1)} = -\left(F\left(x^{(1)}\right)\right)^{-1} g^{(1)}$ and $\left(d^{(1)}\right)^T = -\left(g^{(1)}\right)^T H_1$), then

$$\left(d^{(1)}\right)^T Q d^{(0)} = -\left(g^{(1)}\right)^T H_1 Q d^{(0)}$$

Since $d^{(0)} = \frac{x^{(1)} - x^{(0)}}{\alpha_0} = \frac{\Delta x^{(0)}}{\alpha_0}$ then

$$\left(d^{(1)}\right)^T Q d^0 = -\left(g^{(1)}\right)^T H_1 Q \frac{\Delta x^{(0)}}{\alpha_0}$$

Since $Q\Delta x^{(0)} = \Delta g^{(0)}$ then

$$\left(d^{(1)}\right)^T Q d^{(0)} = -\left(g^{(1)}\right)^T H_1 \frac{\Delta g^{(0)}}{\alpha_0} \ . \qquad (3.15)$$

Since $H_1 \Delta g^{(0)} = \Delta x^{(0)}$ then

$$\left(d^{(1)}\right)^T Q d^{(0)} = -\left(g^{(1)}\right)^T \frac{\Delta x^{(0)}}{\alpha_0}. \qquad (3.16)$$

$$\left(d^{(1)}\right)^T Q d^{(0)} = -\left(g^{(1)}\right)^T d^{(0)} \ . \qquad (3.17)$$

Since $\alpha_0 > 0$ is minimizer of $\phi(\alpha) = f\left(x^{(0)} + \alpha d^{(0)}\right)$, therefore

$$\left(d^{(1)}\right)^T Q d^{(0)} = 0,$$

Assume that the result is true for $k-1$, where $k < n-1$, that is, $d^{(0)}, d^{(1)}, \ldots, d^{(k-1)}$ are Q-conjugate. We now prove the result is true for k, that is,

$$\left(d^{(k+1)}\right)^T Q d^{(i)} = 0 \ , \quad 0 \le i \le k \quad .$$

Note that

$$\begin{aligned}\left(d^{(k+1)}\right)^T Q d^{(i)} &= -\left(H_{k+1} g^{(k+1)}\right)^T Q d^{(i)} \\ &= -\left(g^{(k+1)}\right)^T H_{k+1} Q d^{(i)} \\ &= -\left(g^{(k+1)}\right)^T H_{k+1} Q \frac{\Delta x^{(i)}}{\alpha_i}\end{aligned}$$

That is,

$$\begin{aligned}\left(d^{(k+1)}\right)^T Q d^{(i)} &= -\left(g^{(k+1)}\right)^T H_{k+1} \frac{\Delta g^{(i)}}{\alpha_i} \\ &= -\left(g^{(k+1)}\right)^T \frac{\Delta x^{(i)}}{\alpha_i} \\ &= -\left(g^{(k+1)}\right)^T d^{(i)}\end{aligned}$$

Since $d^{(0)}, \cdots, d^{(k)}$ are Q-conjugate by assumption then, we have $\left(g^{(k+1)}\right)^T d^{(i)}$, thus

$$\left(d^{(k+1)}\right)^T Q d^{(i)} = 0$$

From this theorem, we conclude that a Quasi-Newton methods solve a quadratic of n variables in at most n steps.

Remark [21] Note that the equations that the matrices H_k are required to satisfy do not determine those matrices uniquely. So, we have some freedom.

In the following section, we compute H_{k+1} by adding a correction quantity to H_k.

3.5The quasi-Newton conditions[21] [23][24]

Using the $(n\ +1)-$point secant method as a starting point, let D_k. be the matrix whose columns are composed of the previous n values of $d_k = x_{k+1} - x_k$, and assume for simplicity that the columns of the matrix D_k occur in the order $d_{k-1}, d_{k-2}, \cdots, d_{k-n}$ The $(n+1)-$point secant method gives an approximate Hessian H_{k+1} that may be written in form

$$H_{k+1} = H_k + U_k = H_k + \frac{1}{U_k^T d_k}(u_k - H_k d_k)v_k^T \, , v_k^T d_k \neq 0 \, ,$$

with H_{k+1} satisfying the secant conditions

$$H_{k+1}d_j = H_k d_j, \quad j = k-1 \, , \cdots \, , k-n+1 \, ,$$
$$H_{k+1}d_k = y_k \qquad (3.18)$$

This states that U_k leaves H_k "untouched" along the $n-1$ previous directions $d_{k-1}, d_{k-2}, \cdots, d_{k-n+1}$, and makes H_{k+1} behave like the exact

Hessian along the "new" direction d_k. The last condition is justified by the expansion

$$H_{k+1}d_k = y_k = \nabla f(x_k + d_k) - \nabla f(x_k)$$
$$= \nabla^2 f(x_k)d_k + \int_0^1 \left(\nabla^2 f(x_k + td_k) - \nabla^2 f(x_k)\right)d_k \, dt$$
$$= \nabla^2 f(x_k)d_k + O(L\|d_k\|^2),$$

Where L is the Lipschitz constant. If ∇f is an affine function—i.e., $\nabla f(x) = c + H_x$ for some constant H and c , then the last relation(3.18) is exact, with

$$H_{k+1}d_k = y_k = \nabla f(x_{k+1}) - \nabla f(x_k) = c + Hx_{k+1} - (c + Hx_k) = Hd_k$$

$$= \nabla^2 f(x_k)d_k .$$

If the n directions $d_k , d_{k-1} , \ldots, d_{k-n+1}$ are linearly independent, then the conditions (3.18) are sufficient to define U_k uniquely. This means that some of the conditions must be relaxed in order to define methods based on alternative choices for U_k. The standard strategy is to relax the conditions $H_{k+1}d_j = H_k d_j , j = k-1 , k-2 , \cdots , k-n ,$ but keep the last condition

$$H_{k+1}\, d_k = y_k \ , \qquad (3.19)$$

Which is known as the quasi − Newton condition. The simplest possible form for U_k is the rank − one matrir

$$U_k = u_k v_k^T \qquad (3.20)$$

For some vectors u_k and v_k to be determined. Substituting (3.20) in (3.19)we have

$$H_{k+1}d_k = (H_k + u_k v_k^T)d_k = y_k\ ,\ or\ \ u_k(v_k^T d_k) = y_k - H_k d_k\ , \qquad (3.21)$$

Which has several interesting implications. First, H_k itself satisfies the quasi-Newton condition if $H_k d_k = y_k$ Second, the rows of U_k are orthogonal to d_k (i.e., $U_k d_k = 0$) if $v_k^T d_k = 0$; if $H_k d_k \neq y_k$; the updated matrix H_{k+1} can satisfy the quasi-Newton condition only if $v_k^T d_k \neq 0$. Finally, assuming that $H_k d_k \neq y_k$ and $v_k^T d_k \neq 0$, the quasi-Newton condition will be satisfied by the rank-one matrix $U_k = u_k v_k^T$ only if $(H_k + U_k)d_k = H_k d_k + u_k(v_k^T d_k) = y_k$, so that u_k must be a multiple of $y_k - H_k d_k$, with

$$H_{k+1} = H_k + \frac{1}{v_k^T d_k}(y_k - H_k d_k)v_k^T. \qquad (3.22)$$

One obvious choice of t is $v_k = d_k$, which makes U_k well-defined whenever d_k is not zero. In this case,

$$H_{k+1} = H_k + \frac{1}{d_k^T d_k}(y_k - H_k d_k)d_k^T\ , \qquad (3.23)$$

which is known by several names, including the Broyden update, the good Broyden update, and Broyden′s second update.

Each iteration of a quasi-Newton method requires solution of the $n \times n$ linear system $H_k p_k = -\nabla f(x_k)$, and a "naive" implementation would require $O(n^3)\ flops^1$ to perform this caleulation.

Methods for modifying matrix factorization were not available when quasi-Newton methods were being developed, and so the earliest quasi-Newton updates were de- rived in terms of the $inverse$ Hessian. With exact arithmetic, every update to H_k is associated with an equivalent update to H_k^{-1}, using the Sherman-Morrison- Woodbury formula. For example, if M_k. denotes H_k^{-1}, the inverse form of the general update (**3.23**) is

$$M_{k+1} = M_k + \frac{1}{v_k^T M_k y_k}(d_k - M_k y_k) v_k^T M_k ,$$

from which the inverse form of the Broyden update

$$M_{k+1} = M_k + \frac{1}{d_k^T M_k y_k}\left(d_k - M_k y_k\right) d_k^T M_k , \qquad (3.24)$$

is obtained by setting $v_k = d_k$ as before.

However, it is possible to derive quasi-Newton updates for the inverse Hessian $directly$ by considering update matrices U_k that satisfy the quasi-Newton condition

$$(M_k + U_k) y_k = M_{k+1} y_k = d_k,$$

Using a formula analogous to (3.20) we can write all admissible rank-one updates in the form

$$M_{k+1} = M_k + \frac{1}{w_k^T y_k}(d_k - M_k y_k) w_k^T \ ,$$

Where w_k is arbitrary, except for the fact that it must satisfy $w_k^T y_k \neq 0$. In this case, the "obvious" formula is defined by setting $w_k = y_k$. giving

$$M_{k+1} = M_k + \frac{1}{y_k^T y_k}(d_k - M_k y_k) y_k^T \ ,$$

The derivation of this update predates the Broyden update (**3.24**) because of the perceived additional expense of solving the equations $H_k p_k = -\nabla f(x_k)$ rather than forming the product $p_k = -M_k \nabla f(x_k)$ at each step. This formula is not the same as the $Broyden\ update$ (**3.24**), and is known as $Broyden's\ first\ update$ or the bad B$royden\ update$. It is easy to see that we can write down this formula as an update to H_k , by simply interchanging "M" for "B" and "d" for "y" in the inverse formula (**3.24**) ,giving

$$H_{k+1} = H_k + \frac{1}{d_k^T H_k y_k}(y_k - H_k d_k) y_k^T H_k .$$

3.6 The quasi-Newton Hessian

A quasi-Newton line-search method (or, when the meaning is clear, simply a $quasi - Newton$ method) is typically defined by the sequence of iterates $x_{k+1} = x_k + \alpha_k p_k$ where p_k is the solution of

$$H_k p_k = -\nabla f(x_k) \ , \quad \textbf{(3.25)}$$

α_k satisfies appropriate sufficient decrease conditions, and H_k is updated at every iteration using a quasi-Newton update. When H_k is symmetric and positive definite, p_k of (**3.24**) is the unique minimizer of the

quadratic model $q_k(x)$ (**3.25**), whose Hessian changes at every iteration hence the name variable metric methods originally suggested by Davidon and still used by some authors.

Up to this point, we have not made use of the fact that the Hessian H is actually the second derivative of a scalar-valued function f. The two fundamental properties of H that we plan to exploit are symmetry and definiteness.

An update formula $H_{k+1} = H_k + U_k$ is said to have the property of hereditary symmetry if symmetry of H_k implies symmetry of H_{k+1}, and hereditary positive- definiteness if the positive-definiteness of H_{k+1} follows from that of H_k. A crucial property of symmetric positive-definite approximate Hessians is they may be used directly to define a $quadratic\ model$ whose minimizer can be used to define the next iterate. By analogy with a Newton-based line-search method, a quasi-Newton approximation H_k may be used to define a local quadratic model of f :

$$q_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k). \quad (\mathbf{3.26})$$

which has a unique minimizer only if H_k is symmetric positive definite.

The $curvature$ of f plays a particularly important role in the development and understanding of quasi-Newton methods for minimization. The curvature of f along d_k at an iterate x_k is given by $d_k^T \nabla^2 f(x_k) d_k$, which cannot be computed exactly because H is (by assumption) unknown. However, a Taylor-series expansion of the gradient $\nabla f(x_k + d_k)$ gives

$$\big(\nabla f(x_k + d_k) - \nabla f(x_k)\big)^T d_k \quad = d_k^T \nabla^2 f(x_k) d_k + \int_0^1 d_k^T \big(\nabla^2 f(x_k + t d_k) - \nabla^2 f(x_k)\big) d_k \, dt \ ,$$

and hence, if $\|d_k\|$ is "small enough", the curvature at x_k should be "close" to the curvature at the intermediate point $x_k + td_k$, and

$$y_k^T d_k \approx d_k^T \nabla^2 f(x_k) d_k \ .$$

The quantity $y_k^T d_k$, defined from only first-order information, is known as the $approx - imate\ curvature$ of f at x_k. An important property of the quasi-Newton condition is that it forces the new quadratic model q_{k+1} to have curvature $y_k^T d_k$, along d_k; i.e.,

$$d_k^T H_{k+1} d_k = y_k^T d_k \quad \textbf{(3.27)}$$

We say that the quasi-Newton condition installs the approximate curvature $y_k^T d_k$ as the $exact$ curvature of the new quadratic model. With this interpretation, the approximate Hessian H_k **,** represents (in some sense) curvature information that has been accumulated at iterates preceding x_k1. The move from x_k to x_{k+1} provides further information about curvature that may be incorporated in a new Hessian approximation H_{k+1} .

Procedure [19][20]

The procedure is much the same as regular Newton's Method with a modification to the Hessian Matrix step. Again, the modification depends on the specific type of Quasi-Newton Method being used .

For a given $f(x)$ that is twice-differentiable:

1. Choose a starting point x_o.

2 Calculate search direction by estimating H^{-1} (method varies)

3. Calculate change in x using the following equation:

$$x^{k+1} = x^k - [H^{-1}]_k * grad(x^k)$$

4. Determine new x value, x^1

5. Determine if method has converged using convergence cniteria (gradient)

6. Repeat from step 2 if not optimum

Example 3.2

The following is a brief numerical example of one type of Quasi-Newton Method. This method uses the original inverse Hessian for each iteration.

Objective function:

$$\min f(x) = 2x_1^2 + 3x_2^2$$

Step 1: Choose starting point x_o, $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Step 2: Calculate inverse Hessian (approximate)

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

$\nabla^2 f(x) = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$

$H^{-1} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.3 \end{bmatrix}$

Step 3: Find new x

$x^{k+1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.5 & 0 \\ 0 & 0.3 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$

Step 4: Determine new x value

$x^{k+1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Step 5. Determine if converged

$\nabla f(x) = 0$

Converged!

Step 6: Repeat (if necessary)

Since this example converged, this step is not necessary. If the problem had not converged, solution would have resumed from step 2. Unlike in Newton's Method, the inverse Hessian would not be recalculated exactly. In this particular example, the inverse Hessian would be reused at each step instead of being recalculated. In most cases with Quasi-Newton Methods, there would be some kind of "update" to the original value. Again, these can vary from case to case.

3.7 Comparison between Newton's Method and Quasi-Newton Method

Newton's Method	Quasi-Newton Method
Computationally expensive	Computationally cheap
Slow computation	Fast(er) computation
Need to iteratively calculate second derivative	No need for second derivative
Need to iteratively solve linear system of equations	No need to solve linear system of equations
Less convergence steps	More convergence steps
More precise convergence path	Less precise convergence path

3.8 The Steepest-Descent Method [25][26]

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 [50, 81]. 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 [50]. 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 x0. 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 [50]. Steepest descent (SD) method is a simple gradient method for the unconstrained optimization:

$$\min_{x \in R^n} f(x)$$

where f(x) is a continuous differential function in R^n, now present the following

steepest descent algorithm.

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

(a) Initialize x^0 and ϵ,set k := 0

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

i. = $d^k = -\boldsymbol{g^k}$

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

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

iii. $x^{k+1} = x^k + \alpha^k d^k$
iv. k:=k+1 end while

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

3.9 The Conjugate Gradient Methods [27]

The conjugate gradient (CG) method is ideal for comprehending due to its low memory requirement and strong global convergence properties. The method generates a sequence of iterates recurrently by

$x^0 \in R^n$, $x^{k+1} = x^k + \alpha^k d^k$, $k \geq 0$
where α^k is the step-length and α^k is the search direction determined by

$$\alpha^k = \begin{cases} -G(x^k) & \text{if k} = 0 \\ -G(x^k) + \beta^k d^{k-1} & \text{if k} \geq 1 \end{cases}$$

where G (x^k) and β^k is a parameter, which characterizes the CG method.

.**Example3.3** Consider the unconstrained optimization problem

$$\min_{x \in R^2} f(x) = [\,1.5 - x_1(1 - x_2)]^2 + [\,2.625 - x_1(1 - x_2^3)]^2 + [\,2.25 - x_1(1 - x_2^2)]^2$$

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 conditionϵ = 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^*= $(3.025, 0.474)^T$. 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 is 0.063 in the end of 20^{th} iteration the function value is very small, if use the spotted criteria of $\|g^K\| \leq \epsilon = 1e - 6$(see table 3.4).

(b) Newton Method

In newton method the direction search d^KN = $-H^{-1}$g<0 the direction depends on 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 3.1).

(c) The Quasi-Newton method(DFP update) and the quasi-Newton

Method(BFGS update), in this method will replace Hessian matrix by approximate matrix Bk 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 3.2, 3.3)

(d) The Conjugate Gradient In this method depend on orthogonal eigenvectors of Hessian matrix, the table(see table 3.5) the behaver of converge to minimizer.

Table 3.1: The Newton method

K	x_1	x_2	$f\ (x_1, x_2)$	$\|\|gradient\|\|$
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

Through iteration 8,9 it appears to as clearly. No, guaranty that d^k descent direction.

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. In Steepest descent method fast convergence with slow steps. In conjugate gradient method. Regular convergence without using the second derivative information.

Table 3.2: The Quasi-Newton method(DFP update)

K	x_1	x_2	$f\ (x_1, x_2)$	$\|gradient\|$
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

Table3.3 : The Quasi-Newton method(BFGS update)

K	x_1	x_2	$f\ (x_1, x_2)$	$\|gradient\|$
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	2.551	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

Table3.4: The iterations of Steepest descent method

K	x_1	x_2	$f\ (x_1, x_2)$	$\|\|gradient\|\|$
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	0.063	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	0.014
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 3.5: Conjugate Gradient Method

K	x_1	x_2	$f(x_1, x_2)$	$\|gradient\|$
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

Chapter four

4.1 Advantages:

While the last two rows in the table appear to be disadvantages of the Quasi-Newton Method, the faster computation time can end up balancing them out. For large and complicated problems, this balance is the benefit of the Quasi-Newton Methods over the full Newton's Method owing to the overall faster solution time .

4.2 Disadvantages:

The lack of precision in the Hessian calculation leads to slower convergence in terms of steps. Because of this, Quasi-Newton Methods can be slower (and thus worse) than using the full Newton's Method. This occurs for simple problems where the extra computation time to actually compute the Hessian inverse is low. In this case, the full Newton's Method is better anyway. An additional potential disadvantage to the Quasi-Newton Methods is the need to store the inverse Hessian approximation. This could require a large amount of memory and could thus be detrimental in the cases of large complicated systems.

4.3 Conclusion

Quasi-Newton Methods are an efficient way to optimize functions when either computation or iteration is costly. While their exact methods vary, they all can determine the optimum faster and more efficiently than Newton’s Method when the problems are complex

REFERENCES

[1] Logan, J. David. Applied mathematics. John Wiley & Sons, 2013.

[2] Lee, John A., and Michel Verleysen. Nonlinear dimensionality reduction. Vol. 1. New York: Springer, 2007.

[3] Ford, Matthew. "Matthew Ford's diet: An Application of the Diet Problem." (2006).

[4] De Jong, Kenneth Alan. An analysis of the behavior of a class of genetic adaptive systems. University of Michigan, 1975.

[5] Michalewicz, Zbigniew, and Cezary Z. Janikow. "Handling constraints in genetic algorithms." Icga. 1991.

[6] Griewank, Andreas, and Andrea Walther. "On constrained optimization by adjoint based quasi-Newton methods." Optimization Methods and Software 17.5 (2002): 869-889.

[7]Bertsekas, Dimitri P., W. Hager, and O. Mangasarian. "Nonlinear programming. athena scientific belmont." Massachusets, USA (1999).

[8] Bertsekas, Dimitri, and John Tsitsiklis. Parallel and distributed computation: numerical methods. Athena Scientific, 2015..

[9] Wright, S., and J. Nocedal. "Numerical optimization: Springer Science, 35." (1999). [10] Kirsch, Uri. Structural optimization: fundamentals and applications. Springer Science & Business Media, 2012.

[11] Kwon, Roy H. Introduction to linear optimization and extensions with MATLAB. CRC Press, 2013.

[12] Levitin, Evgeny S., and Boris T. Polyak. "Constrained minimization methods." USSR Computational mathematics and mathematical physics 6.5 (1966): 1-50..

[13] Mishra, Shashi Kant, and Bhagwat Ram. Introduction to unconstrained optimization with R. Springer Nature, 2019.

[14] Sun, Jie, Li-Wei Zhang, and Yue Wu. "Properties of the augmented Lagrangian in nonlinear semidefinite optimization." Journal of optimization theory and applications 129.3 (2006): 437-456.

[15] Alhawarat, Ahmad, 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): 1-15..

[16] Sun, Wenyu, and Ya-Xiang Yuan. Optimization theory and methods: nonlinear programming. Vol. 1. Springer Science & Business Media, 2006.

[17] Füredi, Zoltán, and János Komlós. "The eigenvalues of random symmetric matrices." Combinatorica 1 (1981): 233-241..

[18] Boyd, Stephen P., and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[19] Hennig, Philipp, and Martin Kiefel. "Quasi-Newton methods: A new direction." The Journal of Machine Learning Research 14.1 (2013): 843-865.

[20] Loke, Meng Heng, and Torleif Dahlin. "A comparison of the Gauss–Newton and quasi-Newton methods in resistivity imaging inversion." Journal of applied geophysics 49.3 (2002): 149-162.

[21] Butenko, Sergiy, and Panos M. Pardalos. Numerical methods and optimization: An introduction. CRC Press, 2014.

[22] Geradin, Michel, Sergio Idelsohn, and Michel Hogge. "Computational strategies for the solution of large nonlinear problems via quasi-Newton methods." Computers & Structures 13.1-3 (1981): 73-81.

[23] Bogaers, Alfred EJ, et al. "Quasi-Newton methods for implicit black-box FSI coupling." Computer Methods in Applied Mechanics and Engineering 279 (2014): 113-132. **[24]** Kelley, C. T., and E. W. Sachs. "Quasi-Newton methods and unconstrained optimal control problems." SIAM Journal on Control and Optimization 25.6 (1987): 1503-1516. [25] Mishra, Shashi Kant, and Bhagwat Ram. Introduction to unconstrained optimization with R. Springer Nature, 2019.

[26] Yu, Changjun. A study of optimization and optimal control computation: exact penalty function approach. Diss. Curtin University, 2012

[27] Conn, Andrew R., et al. "Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints." SIAM Journal on Optimization 6.3 (1996): 674-703..

الخلاصة

تتضمن العديد من التقنيات لحل مشكلات التحسين غير المقيدة العامة غير الخطية التقليل المتكرر لوظيفة النموذج التي تفي بشروط استيفاء معينة. توفر هذه الشروط نموذجًا يتصرف مثل الوظيفة الموضوعية في جوار التكرار الحالي. غالبًا ما تتضمن وظائف النموذج مشتقات من الدرجة الثانية لوظيفة الهدف، والتي قد يكون حسابها مكلفًا. الفكرة الأساسية وراء طرق شبه نيوتن هي الحفاظ على تقريب لمصفوفة هيسان. أثار النجاح العملي لأساليب شبه نيوتن قدرا كبيرا من الاهتمام والبحث الذي أدى إلى عدد كبير من الاختلافات في هذه الفكرة. تعني الصعوبات التحليلية المرتبطة بتوصيف أداء هذه الخوارزميات أن هناك حاجة حقيقية للاختبار العملي لدعم الادعاءات النظرية. الهدف من هذا العمل هو البحث عن أفضل الحلول الممكنة لمشاكل التحسين غير المقيدة العامة غير الخطية وتم اختبار طريقتين في ذلك وهما (طرق نيوتن، طرق شبه نيوتن، طريقة التدرج المتقارن ، طريقة النزول الحاد) وتم مقارنتها. أثبتت النتائج أن طرق شبه نيوتن أفضل من حيث السرعة والدقة .

جمهورية العراق

وزارة التعليم العالي

والبحث العلمي

جامعة بابل

كلية التربية للعلوم الصرفة

قسم الرياضيات

حل مشكلة التحسين العددي باستخدام طرقية نيوتن المحدثة

بحث مقدم

الى مجلس كلية التربية للعلوم الصرفة جامعة بابل كجزء

من متطلبات نيل درجة الدبلوم العالي تربية / رياضيات

مقدم من قبل الطالب

سعد محمد بشان

بأشراف الدكتور

أ. م. د احمد صباح الجيلاوي

2023 م 1444هـ