QUASI NEWTON METHODS

ATAKLTI ALEMAYU HAREGOT

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE
DEPARTMENT OF MATHEMATICS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT OF THE DEGREE OF MASTER OF SCIENCE IN MATHEMATICS (OPTIMIZATION)

ADVISOR: BERHANU GUTA (PhD.)

Jun, 2018

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY
DEPARTMENT OF MATHEMATICS

The undersigned hereby certify that they have read and recommend to the department of mathematics for acceptance of this thesis entitled **"Quasi- Newton Methods"** by **Ataklti Alemayu** in partial fulfillment of the requirements for the degree of Master of Science in Optimization.

Advisor: Dr. Berhanu Guta

Signature: _____

Date _____

Examiner 1: Dr. Tadesse Bekeshie

Signature: _____

Date June 13/2018

Examiner 2: Dr. Addisalem Abathun

Signature: _____

Date _____

Contents

Acknowledgement

First of all I would like to thank to my God for the successful preparation of this work.

Next, I would like to express my heart-felt gratitude to my respected advisor Dr. Berhanu Guta for his genuine advise and constructive comments, for his supervision and reviewing my work sacrificing his golden time from the very beginning up to what is now, for this continuous encouragements and treatments in preparation of my study.

Also, I would like to thank my graduate classmates for they supported and assistance that I have received while writing this paper as well as in my study.

At last, but not least I would like to thank my respected wife w/ro Smret K. and my son Adonyas A. for all the joy, happiness and others they brought in to my life.

Abstract

In this work, we investigate quasi Newton methods for solving unconstrained optimization problems. We consider two different quasi-Newton update formulas, namely, Broyden-Fletcher-Goldfarb and shanno (BFGS) update and Davidon-Fletcher-powell (DFP) update. Line search method is used to find the step length at each iteration. The methods are tested on seven benchmark probelems and comparisons are made among Newton's method, quasi-Newton methods (BFGS and DFP updates) and steepest descent method. Also comparisons are made between the quasi-Newton methods (BFGS and DFP updates). Finally conclusions are drawn upon the obtained results.

Introduction

Consider the unconstrained minimization problem

$$\min f(x)$$
$$x \in \Re^n$$

where $f \in c^1(\Re^n, \Re)$.
The iterative methods are used to solve unconstrained optimization problems in order to get the minimal value of the function problems where the gradient is zero or the termination criteria satisfied. The iterative methods for non linear programming (NLP) are generally grouped into two classes. These are line search methods and trust-region methods. Trust-region methods are one of the important numerical optimization methods in solving NLP problems. These methods usually determines the step size before the improving direction is gained after the step forward, then the model believed to be a good representation of the original objective function. If the improvement is gained, then the model is not to be believed as a good representation of the original objective function with in that region. There are available methods to find an approximate solution of the given trust-region subproblems. Some of them are dogleg method and subspace minimization method [*see* [2], [3]].

Line search methods are one of the basic methods for solving optimization problems. These methods employ descent directions as search directions with the aim of reducing the objective functions by taking an appropriate step length with in this search direction. Some available line search methods are steepest descent method, Newton method and Quasi-Newton method.

As listed above Newton's method is one of the available line search methods to solve unconstrained optimization problems. To solve unconstrained optimization problems, Newton's method needs the function value, gradient and the true Hessian matrix of the objective function. But Solving such problems by Newton method has the disadvantage of being computationally expensive. The inverse of the Hessian has to be calculated in every iteration and that is rather costly. Moreover in some applications the Hessian matrix may not be invertible and positive definite every where. Such types of drawbacks of Newton's method can be solved by different solution approaches such as steepest descent (gradient) method and Quasi-Newton method [*see* [1],[2],[3],[4]].
The steepest descent method only uses the function value and the gradient of the objective function to solve unconstrained optimization problems. But it has slow convergence rate (i.e, linear rate) due to its zigzagging nature, and this may leads the method inefficient.
Quasi-Newton methods only uses the function value, the gradient of the objective function, and which need not compute the true Hessian matrix but generates a series of Hessian approximations to solve unconstrained optimization problems. And also it maintains positive definitive and fast convergence rate (i.e, super linear rate) [3],[4].
So, the **main objective** of this project is to mitigate the drawbacks of Newton's method by Quasi-Newton methods using line search methods in solving unconstrained optimization problems .

Hence the iterative formula for Quasi-Newton methods will be defined as:

$$x_{k+1} = x_k + t_k d_k$$

where t_k is step length and d_k is descent direction.

Also this study contains three chapters. The first chapter is about the pre-request (preliminaries) of the chapters next to it. In the second chapter we will discuss about quasi-Newton method and we focus on two quasi-Newton updates, namely Broyden-Fletcher-Goldfarb and shanno (BFGS) update and Davidon-Fletcher-powell (DFP) update and also we discuss the numerical interpretations based on steepest descent method, Newton's method and quasi-Newton methods(BFGS and DFP updates) in chapter three. In my numerical implementation, the programs are all written in MATLAB code, and the stopping criteria that I have used in the algorithms (steepest descent method, Newton's method and quasi-Newton methods (BFGS and DFP updates) is

$$\|\nabla f(x_k)\| \leq 10^{-4}$$

and the result of the MATLAB code is approximated into four digits after decimal. At the end of chapter three the overall conclusions of the work are drawn and MATLAB codes based on quasi-Newton methods, Newton's method and steepest descent method are attached.

Chapter 1

Preliminaries

In this chapter, different definitions, theorems, iterative procedures, some descent methods and other fundamental concepts which are helpful for this work are considered in different sections.

1.1 Definition and Theorems

Definition 1.1. *Let A be symmetric matrix of order n.Then A is said to be*

1) Positive definite if $x^TAx > 0$, for all non-zero $x \in \Re^n$.

2) Negative definite if $x^TAx < 0$ for all non-zero $x \in \Re^n$.

3) Indefinite if x^TAx can have both positive and negative values.

The following theorem is used to check whether a symmetric matrix A is positive definite.

Theorem 1.1. *A symmetric matrix A with order n is positive definite if and only if all the eigne values of A are positive.*

In chapter two we will prove the positive definiteness of the quasi-Newton updates (BFGS and DFP). For the result we follow, we need define Cholesky factorization and we state Cauchy-Schwartz inequality.

Definition 1.2. *(Cholesky Factorization): Let A be a symmetric and positive definite. The chelosky factorization of A is $A = LL^T$, where L is a lower triangular matrix.*

Theorem 1.2. *(Cauchy-Schwartz Inequality): For any to vectors x and y in $\Re^n$, the Cauchy-Schwartz inequality*

$$| < x, y > | \leq \|x\|\|y\|$$

holds. Furthermore, equality holds if and only if $x = \alpha y$ for some $\alpha \in \Re$.

Proof. [**see [1]**]. □

Also, there is an important theorem to drive the inverse Hessian approximation of quasi-Newton methods (BFGS and DFP updates) in chapter 2.

Theorem 1.3. *(**Sherman-Morrison-Woodbury Formula**)*
Suppose $A \in \Re^{n\times n}$ is an invertible square matrix and $u, v \in \Re^n$ are column vectors. If $I + v^T Au$ is invertible, then $A + uv^T$ is invertible and it's inverse is given by

$$(A + uv^T)^{-1} = A^{-1} - A^{-1}u(I + v^T A^{-1}u)^{-1}v^T A^{-1}$$

Note 1.1. *The Sherman-Morrison-Woodbury formula can be written as:*

$$\left(A + \sum u_i v_i^T \right)^{-1} = A^{-1} - A^{-1}UC^{-1}V^T A^{-1}$$

where $c_{ij} = \delta_{ij} + v_i^T A^{-1} u_j$, $U = (u_i)$ and $V = (v_j)$ $\qquad i, j = 1, 2, ..., k.$

Optimal Solution and Optimal Value

In this subsection we will define optimal solution, optimal value, global minimizer and local minimizer of a given function f.
Definition 1.3. *Let $f(x)$ be objective function and $S \subseteq \Re^n$ be its feasible set.*

1) x^* is called optimal solution for a minimization problem

$$\begin{array}{c} min\ f(x) \\ x \in S \end{array}$$

if $x^* \in S$ and $f(x^*) \leq f(x)$, for all $x \in S$.Such x^* is also called global minimizer of f on S.

2) x^* is called a local minimizer of f on S if there is $\epsilon > 0$ such that $f(x^*) \leq f(x)$,for all $x \in N_\epsilon(x^*) \cap S$, where $N_\epsilon(x_0) := \{x \in S : \|x - x_0\| < \epsilon\}$, neighborhood of x_0.

3) If $x^* \in S$ is an optimal solution of f, then $f(x^*)$ is called the optimal value.

Note 1.2. *If S is the whole vector space $\Re^n$, then the problem is said to be unconstrained optimization problem.*

Convex Functions

A functional $f : \Re^n \longrightarrow \Re$ is said to be convex if for any $x, y \in \Re^n$ and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

Note 1.3. *a) f is strictly convex if for every distinct $x, y \in \Re^n$, and $\lambda \in (0, 1)$,*

$$f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda)f(y).$$

b) f is concave if and only if $-f$ is convex

Some Properties of Convex Functions

Theorem 1.4. *Let $f : \Re^n \longrightarrow \Re$ be a convex function.If x_0 is local minimizer of f, then it is global minimizer of f.*

Theorem 1.5. *Let $f : \Re^n \longrightarrow \Re$ be differentiable.*

1) x_0 is a minimizer of f if $\nabla f(x_0) = 0$.

2) Let f be twice continuously differentiable. The Hessian matrix of f is positive semi-definite at each $x \in \Re^n$ if and only if f is convex on $\Re^n$.

Next, we will see the overview of iterative solution approaches of optimization problem.

Overview of Iterative Solution Approach

Most numerical methods for optimization are iterative.That is, starting from an initial guess x_0, construct a sequence of points $x_0, x_1, x_2, ..., x_k, ...$ which converges to a desired solution.

Generic iterative procedure to find min $f(x), x \in \Re^n$ is described as follow.

Step 1: Start with an initial guess x_0.
Let $k = 0$.

Step 2: Check whether x_k satisfies a termination criteria.

- ▲ If Yes Stop.That is, x_k is a desired solution.
- ▲ If No,GOTO Step 3.

Step 3: From the current iterate x_k determine the next iterate point x_{k+1} such that the value of f is reduced, that is, $f(x_{k+1}) < f(x_k)$.
Replace k by $k + 1$ and repeat from step 2.

Note 1.4. *i) Such method in which $f(x_{k+1}) < f(x_k)$ for all k is called descent method.*

ii) At Step 2, a termination criteria is mainly an optimality condition: $\|\nabla f(x_k)\| = 0$, or $\|\nabla f(x_k)\| < \epsilon$, for a sufficiently small $\epsilon > 0$ or terminates when there is no progress, that is, $\|x_{k+1} - x_k\| < \epsilon$.

iii) At Step 3 to construct x_{k+1} such that $f(x_{k+1}) < f(x_k)$, most numerical methods need at x_k a direction (non-zero vector) d_k along which f decreases at least near x_k. That is, we need $d_k \in \Re^n$ such that $f(x_k + td_k) < f(x_k)$ for all $t \in (0, \delta)$ for some $\delta > 0$. Such d_k is called a descent direction of f at x_k.

Now, before proceed to the next sections, we need to define descent direction d_k.

Definition 1.4. *Consider a function $f : \Re^n \to \Re$, and a point $x_k \in \Re^n$ be given. A non-zero vector $d_k \in \Re^n$ is called a descent direction of f at x_k if and only if there is $\delta > 0$ such that $f(x_k + td_k) < f(x_k)$ for all $t \in (0, \delta)$.*

Also the following theorem (1.5) helps us to check whether a non-zero vector d_k is a descent direction of f at a given point x_k.

Theorem 1.6. *Let $f : \Re^n \to \Re$ be a differentiable function at $x_k \in \Re^n$ and d_k be a non-zero vector in $\Re^n$. If $\nabla f(x_k)^T d_k < 0$, then d_k is a descent direction of f at x_k.*

Other important issues for iterative methods are convergence and its rate (speed) of convergence.

Convergence and Rate of Convergence

- Convergence: ensures that the sequence of iterates x_k converges to a desired solution x^*, where x^* is optimal solution, at least locally.

The following theorem is used frequently to check convergence of iterative procedures.

Theorem 1.7. *Let $\{x_k\}$ be the sequence of iterates. If there is some $r \in (0,1)$ and some $k_0 \geq 0$ such that $\|x_{k+1} - x^*\| \leq r\|x_k - x^*\|, \quad \forall k \geq k_0$, then $x_k \to x^*$.*

Proof. With out loss of generality, start from x_0. Let $\|x_0 - x^*\| = d_0$
We need to show that $\|x_k - x^*\| \to 0$ as $k \to \infty$. Now

$$\begin{aligned}
\|x_1 - x^*\| &\leq r\|x_0 - x^*\| \leq rd_0 \\
\|x_2 - x^*\| &\leq \|x_1 - x^*\| \leq r^2 d_0 \\
\|x_3 - x^*\| &\leq \|x_2 - x^*\| \leq r^3 d_0 \\
&\vdots \\
\|x_k - x^*\| &\leq \|x_{k-1} - x^*\| \\
\Rightarrow 0 \leq \|x_k - x^*\| &\leq r^k d_0 \qquad r \in (0,1) \\
\Rightarrow \quad 0 \leq \lim_{k\to\infty} \|x_k - x^*\| &\leq \lim_{k\to\infty} r^k d_0 \\
\Rightarrow \quad 0 \leq \lim_{k\to\infty} \|x_k - x^*\| &\leq 0 \qquad (as \;\; 0 < r < 1, \quad \lim_{k\to\infty} r^k d_0 = 0) \\
\Rightarrow \quad \lim_{k\to\infty} \|x_k - x^*\| &= 0 \\
\Rightarrow \quad x_k &\to x^*
\end{aligned}$$

. □

- Rate of Convergence: Measures how fast the sequence does converge. There are three common rate of convergence, These are linear rate of convergence, supper linear rate of convergence and quadratic rate of convergence.

Now, let us define them as follow.

Definition 1.5. *Let x_k be a sequence in $\Re^n$ that converges to x^*. The rate of convergence is said to be*

a) linear if there is a constant $r \in (0,1)$ *and* $k_0 \geq 0$ *such that*

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq r \quad for \quad all \quad k \geq k_0.$$

b) super linear if

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

c) quadratic if there is a constant $M > 0$ *and* $k_0 \geq 0$ *such that*

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} \leq M \quad for \quad all \quad k \geq k_0.$$

Note 1.5. *Quadratic rate of convergence is the fastest, and supper linear rate of convergence is faster than linear rate of convergence.*

Now, before proceed to the next section, we state an important theorem used to study minimizers of smooth functions, so called Taylor's theorem. Because this theorem is central to analysis throughout this project.

Theorem 1.8. *(Taylor's theorem): Suppose that* $f : \Re^n \longrightarrow \Re$ *is continuously differentiable and that* $p \in \Re^n$*. Then we have that*

$$f(x+p) = f(x) + \nabla f(x+tp)^T p$$

for some $t \in (0,1)$*. Moreover, if* f *is twice continuously differentiable, we have that*

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+tp)pdt$$

and that

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+tp)p$$

for some $t \in (0,1)$*.*

1.2 Line Search Method

Consider the unconstrained minimization problem

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

where $f \in c^1(\Re^n, \Re)$.
Line search is a one -dimensional search refers to an optimization problem for one variable functions. Each iteration of a line search method computes a search direction d_k and then decides how far to move along the direction. For given x_k, the iterative point is given by

$$x_{k+1} = x_k + t_k d_k$$

The key is to find the search direction d_k and a suitable step length t_k.

$$\begin{array}{c} \text{Let} \qquad \phi : \Re \to \Re \\ \phi(t) = f(x_k + td_k) \quad t > 0 \end{array}$$

So, the problem that initially departs from x_k and finds a step size t_k in the direction d_k such that $\phi(t_k) < \phi(0)$ is a line search about t.

Definition 1.6. *If an interval $[a, b]$ contains a minimizing point of ϕ, then $[a, b]$ is said to be an interval of uncertainty (Iou) for ϕ.*

The general procedure of line search method to find minimization of $f(x)$ subject to $x \in \Re^n$ is described as follow.

Algorithm 1.1. *(Generic line search method)*

1) *Choose an initial iterate x_0 and sufficiently small tolerance $\epsilon > 0$*
   set $k = 0$

2) *Find a search direction d_k at iterate x_k*

3) *Calculate a suitable step length $t_k > 0$ which minimizes $\phi(t)$ using a one-dimensional search.*

4) *Set $x_{k+1} = x_k + t_k d_k$*

5) *If $\| \nabla f(x_{k+1}) \| < \epsilon$ stop, the current iterate point is the minimizer. Otherwise go to step two and repeat the procedure.*

If we find t_k such that the objective function in the direction d_k is minimized. That is,

$$f(x_k + t_k d_k) = \min f(x_k + td_k) \text{ for } t > 0$$

Such a line search is called exact line search or optimal line search, and t_k is called optimal step length.
If t_k is not optimal solution of the one dimensional problem but satisfies certain conditions, then such a line search is called inexact line search or approximate line search.
The following are desired conditions that the step size t_k required to satisfy:
Wolfe Conditions

i) $f(x_k + t_k d_k) \leq f(x_k) + c_1 t_k \nabla f(x_k)^T d_k$ for some $0 \leq c_1 \leq 1$

ii) $\nabla f(x_k + t_k d_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k$ for some $c_1 < c_2 < 1$

The following is a generic exact line search procedure to find minimization of $f(x)$ subject to $x \in \Re^n$

Algorithm 1.2. *(Generic exact line search method)*

1) *Choose sufficiently small $\epsilon >$ and an initial guess $x_1 \in \Re^n$*

2) *while $\| \nabla f(x_k) \| > \epsilon$ do:*

▲ *Determine a descent direction* d_k *of* f *at* x_k

▲ *Find minimizer of* f *along* d_k*, that is, let* $\phi(t) = f(x_k + td_k), t \geq 0$

► *Determine Iou [a,b] of* $\phi(t)$

► *Solve min* $\phi(t)$*,* $a \leq t \leq b$

3) Set $x_{k+1} = x_k + t_k d_k$*, where* t_k *is a solution of min* $\phi(t)$*,* $t \in [a, b]$

end

we have procedure to determine a desired inexact step length t_k. The following procedure can be used to determine t_k that satisfies the wolfe conditions.

Algorithm 1.3. *(Generic inexact line search method)*

Initial step: *Choose a sufficiently small (not too small)* $\delta > 0$ *such that* $f(x_k + \delta d_k) < f(x_k)$ *and* $0 < c_2 < 1$.
Let $t_k = \delta$

main step: *while* $\nabla f(x_k + t_k d_k)^T d_k \leq c_2 \nabla f(x_k)^T d_k$
$t_k = t_k + \delta$;
end

Remark 1.1. *The last* t_k *at termination of the loop satisfies the wolfe conditions.*

Descent methods with appropriate step length are globally convergent. That is, a descent line search method with step length satisfying Wolfe conditions, the gradient $\nabla f(x_k)$ converges to zero.
Since, in practical computation, theoretically exact optimal step size generally can not be found, and it is expensive to find almost exact step size. Therefore, the inexact line search with less computation load is preferable [*see* [2]].

1.3 The Steepest Descent Method

The steepest descent method is one of the fundamental minimization methods for unconstrained optimization problem. It is a line search method in which the search direction at iterate point x_k is given by

$$d_k = -\nabla f(x_k)$$

where $\nabla f(x_k)$ is the gradient of the objective function.
Since it uses the negative gradient as its descent direction, it is also called the gradient method. The iterative scheme of the steepest descent method is given by

$$x_{k+1} = x_k - t_k \nabla f(x_k)$$

where t_k is step length.
The following is the corresponding algorithm of the steepest descent method to find minimization of a differentiable function $f(x)$ subject to $x \in \Re^n$.

Algorithm 1.4. *(The steepest descent method for min* $f(x)$*,* $x \in \Re^n$*)*

Initial step: s_0: *Choose sufficiently small termination tolerance* $\epsilon > 0$ *and give an initial point* $x_0 \in \Re^n$.*Set* $k = 0$,

Main step: *while* $\|\nabla f(x_k)\| > \epsilon$, *do*

s_1: *Compute the search direction*

$$d_k = -\nabla f(x_k).$$

s_2: *Find the step length* t_k *by line search such that* $f(x_k + t_k d_k) = min\ f(x_k + t d_k)$, $t > 0$. *If* t_k *to be (exact or inexact) solution of min* $f(x_k + t d_k)$, $t > 0$, *then*

s_3: *Compute the iterates by*

$$x_{k+1} = x_k + t_k d_k.$$

s_4: *Set* $k = k + 1$, *and return to step* s_1.

end (% end while)

Output: The last x_k is the approximate (at least local) minimizer of the objective function $f(x)$.
The steepest descent method is globally convergent if the objective function has a minima.Unfortunately, the global convergence does not guarantee that the steepest descent method an effective method. The steepest descent is a local property of the algorithm. For many problems, the steepest descent method is not the actual steepest very slowly with the zigzagging phenomena. The zigzagging of the steepest method can be explained by the following facts. Since from exact line search, we have

$$\nabla f(x_{k+1}) d_k = 0$$

then

$$\nabla f(x_{k+1}) \nabla f(x_k) = d_{k+1} d_k = 0$$

This shows that the two gradients are orthogonal to each other on the successive iterates, and thus two successive directions are also orthogonal, which leads to the zigzagging [**see [2]**]. So, it is rather faster with inexact line search (i.e, t_k satisfying Wolfe condition).

Merits and Demerits of the Steepest Descent Method

The steepest descent method is globally convergent (if the objective function has a minima) starting from any initial point $x_0 \in \Re^n$. Also it does not need to calculate the Hessian matrix in each iteration. It needs only the function values and the gradient of the objective function f to solve unconstrained optimization problems.
One disadvantage of the steepest descent method is its rate of convergence is slow (i.e, linear rate) due to its zigzagging nature.

1.4 Newton's Method

we now consider the Newton's iteration, for which the search direction is given by

$$d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$$

called Newton's direction. The basic idea of Newton's method for unconstrained optimization is to iteratively use the quadratic approximation Q_k to the objective function f at the current iterate x_k and to minimize the approximation Q_k.
Let $f : \Re^n \to \Re$ be twice continuously differentiable, $x_k \in \Re^n$, and the Hessian $\nabla^2 f(x_k)$ positive definite. The approximating function f at the current point x_k by the quadratic approximation Q_k is

$$f(x_k + s) \approx Q(x_k) = f(x_k) + \nabla f(x_k)^T s + \tfrac{1}{2} s^T \nabla^2 f(x_k) s$$
$$\text{where } s = x_{k+1} - x_k$$

Now minimizing $Q_k(s)$
$\nabla Q_k(s) = \nabla f(x_k) + \nabla^2 f(x_k) s$

$$\begin{aligned} \nabla Q_k(s) &= 0 \\ \Rightarrow \nabla f(x_k) + \nabla^2 f(x_k) s &= 0 \\ \Rightarrow \nabla^2 f(x_k) s &= -\nabla f(x_k) \\ \Rightarrow s &= -\nabla^2 f(x_k)^{-1} \nabla f(x_k) \\ \Rightarrow x_{k+1} - x_k &= d_k \end{aligned}$$

$x_{k+1} = x_k + d_k$ which is called Newton's iterative formula. If the Hessian $\nabla^2 f(x_k)$ is positive definite, then the Newton's direction d_k is a descent direction. The prove of this is as follow

$$d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k) \tag{1.1}$$

Then pre-multiplying both sides of equation (1.1) by $\nabla f(x_k)^T$, we get.
$\nabla f(x_k)^T d_k = -\nabla f(x_k)^T \nabla^2 f(x_k)^{-1} \nabla f(x_k)$
Since $\nabla^2 f(x_k)$ is positive definite $\Rightarrow \nabla^2 f(x_k)^{-1}$ is positive definite.

$$\begin{aligned} &\Rightarrow \nabla f(x_k)^T d_k = -\nabla f(x_k)^T \nabla^2 f(x_k)^{-1} \nabla f(x_k) < 0 \\ &\Rightarrow \nabla f(x_k)^T d_k < 0 \end{aligned}$$

$\Rightarrow d_k$ is a descent direction.
The corresponding algorithm to find a minimum point of a twice differentiable function f defined on $\Re^n$ is stated as follow.

Algorithm 1.5. *(Newton's method)*

Input:
- ♦ *The objective function $f(x)$.*
- ♦ *The gradient of the function f(i.e $\nabla f(x)$).*
- ♦ *The Hessian matrix $\nabla^2 f(x)$ and*

♦ *Sufficiently small tolerance* $\epsilon > 0$

Initial step: Choose a suitable initial guess $x_1 \in \Re^n$
$k = 1$;

main step: while $\| \nabla f(x_k) \| > \epsilon$

1) Solve the system of linear equations

$$\nabla^2 f(x_k) d_k = -\nabla f(x_k) \tag{1.2}$$

2) Compute $x_{k+1} = x_k + d_k$, *where* d_k *is the solution of (1.2)*

3) Replace k *by* $k + 1$, *and repeat from step 2*

end

Newton's method has some very nice properties and also has some drawbacks. Now, we discuss merits and demerits of Newton's method as follow.

Merits and Demerits of Newton's Method

Some of the advantages of Newton's method is that its not too complicated in form and it can be used to solve a variety of problems. If the objective function is convex, then Newton's method is very fast (i.e, quadratic rate) and can find an approximate of x^* starting from any initial point $x_0 \in \Re^n$.

The major disadvantages associated with Newton's method, is that the Hessian matrix $\nabla^2 f(x_k)$ as well as its inverse has to be computed for each iteration, which may be difficult to calculate. It needs to solve the equation $\nabla^2 f(x_k) d_k = -\nabla f(x_k)$, which could be time consuming depending on the size of the problem. In this case, if the Hessian matrix calculated at each iteration close to singular matrix, computing the Newton search direction d_k may diverge. When using Newton's method, the Hessian matrix may not always be positive definite. So, the Newton's search direction d_k may not always be a descent direction. If the objective function f is not convex, then the method can be successful only when the initial point is close to x^*. Otherwise the algorithm may not possess the descent property (i.e, $f(x_{k+1}) \not< f(x_k)$ for some k). That is, Newton's method has no global convergence unless the Hessian matrix computed at each iteration is positive definite every where. [**see [1], [2], [3]**].

The expensive computation of the Hessian matrix and its inverse in each iteration of the Newton's algorithm can be avoid when applying steepest descent method, which only uses the function value and the gradient of the objective function. But as we have discussed above about the steepest descent method, it has slow convergence rate (i.e, linear rate) due to its zigzagging nature and this may leads the method inefficient. So, to mitigate the drawbacks of Newton's method we need to use a class of methods that only uses the function value, the gradient of the objective function, and which need not compute the true Hessian but generates a series of Hessian approximations as well as maintains positive definiteness and fast convergence rate (i.e, super linear rate), so called Quasi-Newton methods. We will now look the quasi- Newton methods in the next chapter.

Chapter 2

Quasi-Newton Methods

2.1 Quasi-Newton Method

In chapter one we have seen that Newton's iterative method is $x_{k+1} = x_k - \nabla^2 f^{-1}(x_k)\nabla f(x_k)$, where $\nabla^2 f^{-1}(x_k)$ is the inverse of the true Hessian and $\nabla f(x_k)$ is the gradient of the objective function. However, for various practical problems, the computing efforts of the Hessian (or inverse Hessian) matrices are very expensive or the evaluation of the Hessian is difficult, even the Hessian may not be invertible and positive definite every where. These lead to a class of methods that only uses the function values and the gradient of the objective function and that is closely related to Newton's method.
Quasi-Newton method is such a class of methods which need not compute the Hessian, but generate a series of Hessian approximations. Quasi-Newton method is a line search method, the descent direction at iterate point x_k is given by

$$d_k = -B_k^{-1}\nabla f(x_k)$$

where B_k is a positive definite symmetric matrix, initially identity matrix or a positive definite approximation of the true Hessian matrix $\nabla^2 f(x_k)$ updated at each successive iteration without computing the Hessian matrix.

The iterative scheme of the quasi-Newton method is given by

$$x_{k+1} = x_k + t_k d_k$$

where t_k is step length and d_k is descent direction.

2.1.1 Secant Equation

Instead of computing the Hessian $\nabla^2 f(x_k)$, we like to construct Hessian approximation B_k in the quasi-Newton method. Note that the sequence $\{B_k\}$ possesses positive definitiveness, has a direction $d_k = -B_k^{-1}\nabla f(x_k)$, and behaves like Newton's method. In addition, it is also required that its computation is convenient. In this subsection we reply the question that what conditions does a sequence $\{B_k\}$ satisfy.
Let $f : \Re^n \to \Re$ be twice continuously differentiable on an open set $D \in \Re^n$. Let us consider the model quadratic function at x_{k+1}. That is

$$Q_{k+1}(x) = f(x_{k+1}) + g_{k+1}^T(x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T B_{k+1}(x - x_{k+1}) \qquad (2.1)$$

which satisfies the interpolation condition $Q_{k+1}(x_{k+1}) = f(x_{k+1})$,$\nabla Q_{k+1}(x_{k+1}) = \nabla f(x_{k+1})$. Where $g_{k+1} = \nabla f(x_{k+1})$, and $B_{k+1} = H_{k+1}^{-1}$ is the approximation to the Hessian $\nabla^2(x_{k+1})$. Then finding the derivative of (2.1) yields

$$\nabla Q_{k+1}(x) = \nabla f(x_{k+1}) + B_{k+1}(x - x_{k+1}) \tag{2.2}$$

Setting $x = x_k$, $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
Instead of the interpolation condition $\nabla Q_{k+1}(x_{k+1}) = \nabla f(x_{k+1})$ in Newton's method, the model (2.1) satisfy $\nabla Q_{k+1}(x_k) = g_k$. So, equation (2.2) becomes

$$g_k = g_{k+1} - B_{k+1}s_k$$

$$y_k = B_{k+1}s_k \tag{2.3}$$

Then (2.3) is called secant equation on the Hessian approximation B_{k+1}. Now, we ask to produce inverse Hessian approximation H_{k+1} in the quasi-Newton methods to satisfy $H_{k+1}y_k = s_k$. Since $B_{k+1}^{-1} = H_{k+1}$, from equation (2.3) we have

$$\begin{gathered} y_k = B_{k+1}s_k \\ \Rightarrow B_{k+1}^{-1}y_k = s_k \end{gathered}$$

$$\Rightarrow H_{k+1}y_k = s_k \tag{2.4}$$

which is called the secant condition on the inverse Hessian approximation H_{k+1}.
Pre-multiplying (2.3) by s_k^T gives

$$s_k^T B_{k+1}s_k = s_k^T y_k$$

$\Rightarrow$ if $s_k^T y_k > 0$, the matrix B_{k+1} is positive definite. Usually the inequality $s_k^T y_k > 0$ is called the curvature condition.
In addition to the secant equation, on the updated Hessian approximation B_{k+1} (similar arguments are applied to H_{k+1}), we would like

- B_{k+1} is symmetric matrix.
- B_{k+1} close to B_k.
- If B_k is positive definite, then B_{k+1} is positive definite.

  The condition that B_{k+1} closest to the current matrix B_k is specified by

  $$\|B_{k+1} - B_k\| \leq \|B - B_k\| \quad for \quad all \quad Bs_k = y_k \quad and \quad B = B^T$$

  where s_k and y_k satisfy the curvature condition $s_k^T y_k$ and B_k is symmetric and positive definite. In other words, B_{k+1} should be symmetric and positive definite matrix which differs from B_k with low rank, (that is, $B_{k+1} - B_k$ has low rank).

The following is the general description of the quasi-Newton method to find a minimum of a differentiable function f defined on $\Re^n$.

Algorithm 2.1. *(Generic quasi-Newton algorithm)*

s_0) *choose a starting point* $x_0 \in \Re^n$, *symmetric positive definite matrix* $B_0 \in \Re^{n\times n}$*(say, identity matrix or a positive definite modification of Hessian at* x_0*) and sufficiently small termination tolerance* $\epsilon > 0$.

s_1) *If* $\| \nabla f(x_k) \| \leq \epsilon$, *then stop and output* x_k *as an approximate local minimizer of* f. *Else go to step* s_2

s_2) *compute the quasi- Newton search direction:* $d_k = -B_k^{-1}\nabla f(x_k)$

s_3) *perform a practical line-search for the minimization of the one dimensional problem* $\phi(t) = f(x_k + td_k), t > 0$. *If* t_k *to be (exact or inexact) solution of min* ϕ, *then*

s_4) *compute the new iterate point:* $x_{k+1} = x_k + t_k d_k$.

s_5) *At iterate:* x_{k+1}, *update* B_k *to construct* B_{k+1} *that satisfy (2.3) .*

s_6) *Replace* k *by* $k + 1$ *and go to* s_1.

Since the sequence B_k considered in quasi-Newton algorithm is positive definite, then the search direction d_k computed at step s_2 is descent direction.

2.1.2 Properties of The Hessian Approximation Matrix B_k

p_1 : B_k should be non-singular, so that s_2 is well-defined.

p_2 : B_k should be positive definite such that d_k is a descent direction, so that s_3 is well-defined.

p_3 : B_k should be symmetric, as Hessian are symmetric matrices.

p_4 : B_{k+1} should be computable by "recycling " the quantities $\nabla f(x_{k+1})$, $\nabla f(x_k)$, ... , $\nabla f(x_0)$, d_k, t_k and possibly B_k.

p_5 : B_{k+1} should be close to B_k in a well- defined sense so that B_k converges to $\nabla^2 f(x^*)$ and d_k is allowed to become the Newton step asymptotically.

The major issue in quasi-Newton method is a way of updating the Hessian B_k (or H_k) to construct B_{k+1} (or H_{k+1}) at s_5 in algorithm (2.1) by use of some convenient methods such that the secant conditions (2.3) and(2.4) holds and also the updating matrix B_{k+1} is required to be positive definite symmetric matrix.

2.2 Updating The Hessian Approximation Matrix B_k

Given an $n \times n$ symmetric and positive definite matrix B_k at iterate point x_k so that $d_k = -B_k^{-1}\nabla f(x_k)$, the matrix B_{k+1} at iterate point x_{k+1} is required to have the following properties.

I: B_{k+1} required to be an approximation (or have some feature) of the Hessian at x_{k+1}

II: B_{k+1} should be symmetric and positive definite matrix which differs from B_k with low rank.

Note 2.1. *In quasi-Newton methods, the Hessian approximation matrix B_k updated at each iteration is symmetric and positive definite and hence it is invertible.*

In the following sections we will reply the question, how to form the sequence $\{B_k\}$. Some of the popular updating formulas that satisfies the secant condition (2.3) and/or (2.4)are described below.

2.3 Davidon-Fletcher-Powell (DFP) Rank-2 Update

In this section we introduce the DFP rank-2 quasi-Newton update that satisfies the secant condition (2.4). This update was first proposed by Davidon and popularized by Fletcher and Powell. Let H_k be the inverse Hessian approximation of the k^{th} iteration and select some vectors $u, v \in \Re^n$. H_{k+1} is formed by adding to H_k two symmetric matrices,each of rank-one. Now, in the case of rank-2 update we Consider an update of the form

$$H_{k+1} = H_k + \alpha uu^T + \beta vv^T \tag{2.5}$$

where α and β are scalars to be determined. Then by the secant equation on the inverse of the Hessian matrix H_{k+1} and by equation (2.5) we get

$$\begin{aligned} (H_k + \alpha uu^T + \beta vv^T)y_k &= s_k \\ H_k y_k + \alpha(u^T y_k)u + \beta(v^T y_k)v &= s_k \end{aligned}$$

$$\alpha(u^T y_k)u + \beta(v^T y_k)v = s_k - H_k y_k \tag{2.6}$$

In this case u and v are not uniquely determined, but one of the possible choices are:

$$u = s_k \text{ and } v = H_k y_k$$

Then substituting the choices of u and v into equation (2.6), we get

$$\begin{aligned} &\alpha(s_k^T y_k)s_k + \beta(y_k^T H_k y_k)H_k s_k = s_k - H_k s_k \\ \Rightarrow\ &\alpha(s_k^T y_k)s_k = s_k \text{ and } \beta(y_k^T H_k y_k)H_k s_k = H_k s_k \\ \Rightarrow\ &\alpha = \tfrac{1}{s_k^T y_k} \text{ and } \beta = \tfrac{-1}{y_k^T H_k y_k} \end{aligned}$$

Also substituting the values of α and β into the rank-2 updating formula (2.5).

$$\begin{aligned} H_{k+1} &= H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{(H_k y_k)(H_k y_k)^T}{y_k^T H_k y_k} \\ &= H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k (y_k^T H_k)}{y_k^T H_k y_k} \end{aligned}$$

$$\Rightarrow H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k} \tag{2.7}$$

The updating formula (2.7) is called the Davidon-Fletcher-powell (DFP) rank two update for the inverse Hessian approximation.

Remark 2.1. *The updating formula (2.7) is only one of the possible choices of the solution of equation (2.6) and with other solutions we obtain different updating formulas.*

Next we must prove that under suitable condition the DFP update maintains positive definitiveness.

2.3.1 Positive Definitiveness of DFP Update

Theorem 2.1. *(positive definitiveness of DFP update)*
Let H_k be symmetric and positive definite, then the DFP update

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k}$$

produces H_{k+1} positive definite if $s_k^T y_k > 0$

Proof. Let $s_k^T y_k > 0$ and consider a $x \neq 0$, $x \in \Re^n$. Then we need to prove that $x^T H_{k+1} x > 0$.

$$\begin{aligned} x^T H_{k+1} x &= x^T (H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k}) x \\ &= x^T H_k x - \frac{x^T H_k y_k y_k^T H_k x}{y_k^T H_k y_k} + \frac{x^T s_k s_k^T x}{s_k^T y_k} \end{aligned}$$

Since H_k is symmetric and positive definite so that there exist a Cholesky factorization $LL^T = H_k$, where L is $n \times n$ lower triangular matrix.

$$\begin{aligned} \Rightarrow x^T H_{k+1} x &= x^T L L^T x - \frac{x^T L L^T y_k y_k^T L L^T x}{y_k^T L L^T y_k} + \frac{x^T s_k s_k^T x}{s_k^T y_k} \\ &= (x^T L)(L^T x) - \frac{(x^T L)(L^T y_k)(y_k^T L)(L^T x)}{(y_k^T L)(L^T y_k)} + \frac{(s_k^T x)^2}{s_k^T y_k} \\ &= (L^T x)^T (L^T x) - \frac{(L^T x)^T (L^T y_k)(L^T y_k)^T (L^T x)}{(L^T y_k)^T (L^T y_k)} + \frac{(s_k^T x)^2}{s_k^T y_k} \end{aligned}$$

Setting $a = L^T x$ and $b = L^T y_k$

$$\begin{aligned} \Rightarrow x^T H_{k+1} x &= a^T a - \frac{a^T b b^T a}{b^T b} + \frac{(s_k^T x)^2}{s_k^T y_k} \\ &= \frac{(a^T a)(b^T b) - (b^T a)^2}{b^T b} + \frac{(s_k^T x)^2}{s_k^T y_k} \end{aligned}$$

From the cauchy-Schwartz inequality we have $(a^T a)(b^T b) \geq (b^T a)^2$

$$\Rightarrow \frac{(a^T a)(b^T b) - (b^T a)^2}{b^T b} \geq 0$$

$$\Rightarrow x^T H_{k+1} x = \frac{(a^T a)(b^T b) - (b^T a)^2}{b^T b} + \frac{(s_k^T x)^2}{s_k^T y_k} > 0$$

$$\Rightarrow x^T H_{k+1} x > 0$$

$\Rightarrow$ the updated inverse Hessian approximation H_{k+1} is positive definite.
Hence the DFP update maintains positive definitiveness if $s_k^T y_k > 0$. □

Theorem 2.2. *If the step length t_k satisfies Wolfe conditions, then the condition $s_k^T y_k$ holds and hence the DFP update produce positive definite matrix.*

Proof. From the minimum search algorithm, we have

$$s_k = t_k d_k \tag{2.8}$$

with step length $t_k > 0$, and d_k is descent direction.
Also by Wolfe condition (ii), we have

$$\nabla f(x_k + t_k d_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k, 0 < c_2 < 1$$

Then subtract $\nabla f(x_k)^T d_k$ from both sides of the Wolfe (ii), we get.

$$\begin{aligned}
&\nabla f(x_k + t_k d_k)^T d_k - \nabla f(x_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k - \nabla f(x_k)^T d_k \\
\Rightarrow &(\nabla f(x_k + t_k d_k)^T - \nabla f(x_k)^T) d_k \geq (c_2 - 1) \nabla f(x_k)^T d_k > 0 \\
\Rightarrow &(\nabla f(x_k + t_k d_k)^T - \nabla f(x_k)^T) d_k > 0 \\
\Rightarrow &(\nabla f(x_k + t_k d_k) - \nabla f(x_k))^T s_k > 0 \qquad \text{by (2.8).} \\
&\Rightarrow s_k^T y_k > 0
\end{aligned}$$

The condition $s_k^T y_k$ holds, and hence the DFP update produce positive definite matrix. □

2.3.2 The DFP Update for The Hessian Approximation

Proposition 2.1. *The DFP update for the Hessian approximation (say B_k) that corresponds to the update of the inverse Hessian approximation (2.7) becomes:*

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T + y_k s_k^T B_k}{y_k^T s_k} + \frac{y_k y_k^T}{y_k^T s_k}(1 + \frac{s_k^T B_k s_k}{y_k^T s_k}) \tag{2.9}$$

Proof. consider the Sherman-Morrison-Woodbury formula with rank $k = 1$, and

$$u_1 = v_1 = \frac{s_k}{(y_k^T s_k)^{\frac{1}{2}}}$$

and

$$u_2 = -v_2 = \frac{H_k y_k}{(y_k^T H_k y_k)^{\frac{1}{2}}}$$

Let us now, consider a 2×2 matrix $I + v^T H_k^{-1} u$ with $c_{ij} = \delta_{ij} + v_i^T H_k u_j$.

$$\begin{aligned}
\Rightarrow c_{11} &= \delta_{11} + v_1^T H_k u_1 \\
&= 1 + \frac{s_k^T H_k^{-1} y_k}{(y_k^T s_k)^{\frac{1}{2}} (y_k^T s_k)^{\frac{1}{2}}}
\end{aligned}$$

Setting $B_k = H_k^{-1}$,

$$c_{11} = 1 + \frac{s_k^T B_k y_k}{y_k^T s_k}$$

.

$$\begin{aligned} c_{22} &= \delta_{22} + v_2^T H_k u_2 \\ &= 1 + v_2^T H_k^{-1} u_2 \\ &= 1 - \frac{y_k^T H_k H_k^{-1} y_k}{(y_k^T H_k y^k)^{\frac{1}{2}} (y_k^T H_k y_k)^{\frac{1}{2}}} \\ &= 1 - \frac{y_k^T H_k y_k}{y_k^T H_k y_k} = 0. \end{aligned}$$

$$\begin{aligned} c_{12} &= \delta_{12} + v_1^T H_k u_2 \\ &= 0 + \frac{s_k^T H_k^{-1} H_k y_k}{(y_k^T s_k)^{\frac{1}{2}} (y_k^T H_k y_k)^{\frac{1}{2}}} \\ &= \frac{s_k^T y_k}{(y_k^T s_k)^{\frac{1}{2}} (y_k^T H_k y_k)^{\frac{1}{2}}} \\ &= \frac{(s_k^T y_k)^{\frac{1}{2}}}{(y_k^T H_k y_k)^{\frac{1}{2}}}. \end{aligned}$$

$$\begin{aligned} c_{21} &= \delta_{12} + v_2^T H_k u_1 \\ &= 0 - \frac{y_k^T H_k H_k^T s_k}{(y_k^T H_k y_k)^{\frac{1}{2}} (y_k^T s_k)^{\frac{1}{2}}} \\ &= \frac{(y_k^T s_k)^{\frac{1}{2}}}{(y_k^T H_k y_k)^{\frac{1}{2}}}. \end{aligned}$$

Then the matrix C has the following form:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} e_1 & e_2 \\ e_3 & 0 \end{pmatrix}$$

Where $e_1 = 1 + \frac{s_k^T B_k s_k}{y_k^T s_k}$, $e_2 = \frac{(s_k^T y_k)^{\frac{1}{2}}}{(y_k^T H_k y_k)^{\frac{1}{2}}}$, $e_3 = -\frac{(y_k^T s_k)^{\frac{1}{2}}}{(y_k^T H_k y_k)^{\frac{1}{2}}}$. And then the inverse of matrix C is given by

$$C^{-1} = \begin{pmatrix} 0 & \frac{1}{e_3} \\ \frac{1}{e_2} & -\frac{e_1}{e_2 e_3} \end{pmatrix}$$

Then applying Sherman-Morrison-Woodbury formula on the inverse Hessian approximation H_{k+1}, we have:

$$\begin{aligned} H_{k+1} &= H_k^{-1} - H_k^{-1} u C^{-1} v^T H_k^{-1} \\ \Rightarrow B_{k+1} &= B_k - B_k u C^{-1} v^T B_k \end{aligned}$$

Setting $P = B_k u$, $L = B_k v$, where $P = (p_i)$, $L = (l_i)$ and $p_i = B_k u_i$, $l_i = B_k v_i$. Now, the matrix product$(PC^{-1}L^T)$ is $\Re^{n\times 2} \times \Re^{2\times 2} \times \Re^{2\times n}$.

$$\begin{aligned}
PC^{-1}L^T &= \begin{pmatrix} p_1 & p_2 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{e_3} \\ \frac{1}{e_2} & -\frac{e_1}{e_2 e_3} \end{pmatrix} \begin{pmatrix} l_1^T \\ l_2^T \end{pmatrix} \\
&= \begin{pmatrix} \frac{p_2}{e_2} & \frac{p_1}{e_2} - \frac{p_2 e_1}{e_2 e_3} \end{pmatrix} \begin{pmatrix} l_1^T \\ l_2^T \end{pmatrix} \\
&= \frac{p_2 l_1^T}{e_2} + \frac{p_1 l_2^T}{e_3} - \frac{e_1}{e_2 e_3} p_2 l_2^T \\
&= \frac{B_k u_2 v_1^T B_k}{e_2} + \frac{B_k u_1 v_2^T B_k}{e_3} - \frac{e_1}{e_2 e_3} B_k u_2 v_2^T B_k
\end{aligned}$$

Substituting the values of $e_{i's}$, $u_{i's}$ and $v_{i's}$, we get:

$$= \frac{(y_k^T H_k y_k)^{\frac{1}{2}} B_k H_k y_k s_k^T B_k}{(s_k^T y_k)^{\frac{1}{2}} (y_k^T H_k y_k)^{\frac{1}{2}} (y_k^T s_k)^{\frac{1}{2}}} + \frac{(y_k^T H_k y_k)^{\frac{1}{2}} B_k s_k y_k^T H_k B_k}{(y_k^T s_k)^{\frac{1}{2}} (y_k^T s_k)^{\frac{1}{2}} (y_k^T H_k y_k)^{\frac{1}{2}}} - \frac{y_k^T H_k y_k}{y_k^T s_k}(1 + \frac{s_k^T B_k s_k}{y_k^T s_k}) \frac{B_k H_k y_k y_k^T H_k B_k}{(y_k^T H_k y_k)^{\frac{1}{2}} (y_k^T H_k y_k)^{\frac{1}{2}}}$$

$$\begin{aligned}
\Rightarrow PC^{-1}L^T &= \frac{y_k s_k^T B_k}{s_k^T y_k} + \frac{B_k s_k y_k^T}{y_k^T s_k} - \frac{y_k y_k^T}{y_k^T s_k}(1 + \frac{s_k^T B_k s_k}{y_k^T s_k}) \\
B_{k+1} &= B_k - PC^{-1}L^T \\
&= B_k - \frac{y_k s_k^T B_k}{s_k^T y_k} - \frac{B_k s_k y_k^T}{y_k^T s_k} + \frac{y_k y_k^T}{y_k^T s_k}(1 + \frac{s_k^T B_k s_k}{y_k^T s_k}) \\
&= B_k - \frac{B_k s_k y_k^T + y_k s_k^T B_k}{y_k^T s_k} + \frac{y_k y_k^T}{y_k^T s_k}(1 + \frac{s_k^T B_k s_k}{y_k^T s_k})
\end{aligned}$$

□

Note 2.2. *As we have proved on theorem (2.1), the inverse Hessian approximation H_{k+1} produced by DFP update is positive definite, which also guarantees that its Hessian approximation B_{k+1} is positive definite.*

The updated matrix B_{k+1} should be close to the current matrix B_k. It meaning that the symmetric and positive definite matrix B_{k+1} differ from B_k with low rank. This is equivalent to solve the problem specified by

$$\|B_{k+1} - B_k\| \leq \|B - B_k\| \quad for \quad all \quad Bs_k = y_k \quad and \quad B = B^T \tag{2.10}$$

where s_k and y_k satisfy the curvature condition $s_k^T y_k$ and B_k is symmetric and positive definite.

Note 2.3. *Similar arguments can be applied to the condition of closeness of H_{k+1} to H_k by replacing B_k, B_{k+1}, B, s_k and y_k by H_k, H_{k+1}, H, y_k and s_k respectively on (2.10)*

Algorithm 2.2. *(Quasi-Newton method with DFP update)*

Initial step: *Choose sufficiently small tolerance $\epsilon > 0$, an initial guess $x_1 \in \Re^n$, and an $n \times n$ symmetric and positive definite matrix H_1 (say an $n \times n$ identity matrix or a positive modification of inverse Hessian at x_1)*
Let $k = 0$;

Main step: while $\| \nabla f(x_k) \| > \epsilon$, *do*

1) Compute search direction: $d_k = -H_k \nabla f(x_k)$;

2) Approximate min $f(x+td_k), t > 0$ *by line search and if* t_k *be (exact or inexact) solution of the one-dimensional problem min* $f(x + td_k), t > 0$ *go to step (3).*

3) $x_{k+1} = x_k + t_k d_k$

4) To update H_k

- *Let* $s_k = x_{k+1} - x_k$ *and* $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
- *Compute* H_{k+1} *by (2.7)*

5) set $k = k + 1$ *and go to step (1)*

end

DFP method has the following important properties:

1) For a quadratic function (under exact line search)

   a) DFP update has quadratic termination. That is, $H_n = A^{-1}$, where A is an $n \times n$ symmetric and positive definite Hessian matrix of the quadratic function and H_n is Hessian approximation updated by DFP method.

   b) DFP update has hereditary property, that is $H_i y_j = s_j$ for $j < i$.

2) For a general function

   a) DFP update maintains positive definiteness.

   b) DFP update is supper linear convergent.

   c) For a strictly convex function, under exact line search, DFP method is globally convergent.

The positive definiteness of DFP update is already proved in theorem (2.1). The convergence properties of DFP update will be established in the subsections (2.5.1) and (2.6.2). In the remainder of this section we shall prove the other two important properties: quadratic termination of the method and the hereditary property of the method in theorem (2.3) below. Before proceed to theorem (2.3), we need to state and prove the lemma (2.1) which is important to prove the quadratic termination property.

Lemma 2.1. *Let* $Q(x) = \frac{1}{2}x^T Ax - b^T x + c$ *be a quadratic function with* $A \in \Re^{n \times n}$ *symmetric and positive definite, and* $y_k = \nabla Q(x_{k+1}) - \nabla Q(x_k)$ *and* $s_k = x_{k+1} - x_k$. *Then* $y_k = As_k$

Proof.

$$\begin{aligned} y_k &= \nabla Q(x_{k+1}) - \nabla Q(x_k) \\ &= Ax_{k+1} - b - (Ax_k - b) \\ &= Ax_{k+1} - b - Ax_k + b \\ &= Ax_{k+1} - Ax_k \\ &= A(x_{k+1} - x_k) \\ &= As_k \end{aligned}$$

□

Theorem 2.3. *(Property of DFP update)*
Let $f(x) = \frac{1}{2}x^T Ax - b^T + c$ be a quadratic function with symmetric and positive definite Hessian $A \in \Re^{n\times n}$. Then for any starting point x_0 any starting symmetric and positive definite inverse Hessian H_0, and let the iterates x_k and H_k produced by the sequence s_k. That is,

1) $x_{k+1} = x_k + s_k$

2) H_{k+1} *is produced by the DFP update (2.7)*

where $s_k = t_k d_k$ with t_k is scalar obtained by exact line search and the directions $d_{i's}$, $i = 0, 1, 2, ..., k$ are A-conjugates. Then for $0 < j < k$ the sequence s_k generated from DFP update satisfies

(a) $g_k^T s_j = 0$ *[orthogonality property]*

(b) $H_k y_j = s_j$ *[Hereditary property]*

(c) If s_0, s_1, s_2,..., s_{n-1} are linearly independent, then $H_n = A^{-1}$.

Proof. (a): From lemma(2.1), we have

$$\begin{aligned} y_k = As_k = Ax_{k+1} - b - (Ax_k - b) \\ \Rightarrow g_{k+1} - g_k = Ax_{k+1} - b - (Ax_k - b) \end{aligned}$$

we prove the relation by induction. To show $g_k^T s_j = 0$, it suffices to show $g_k^T d_j = 0$. Now for $k = 1$, the relation becomes

$$\begin{aligned} g_1^T d_0 &= (Ax_1 - b)^T d_0 \\ &= x_1^T Ad_0 - b^T d_0 \\ &= (x_0 + t_0 d_0^T) Ad_0 - b^T d_0 \\ &= x_0^T Ad_0 - (\frac{g_0^T d_0}{d_0^T Ad_0}) d_0^T Ad_0 - b_0^T d_0 \\ &= x_0^T Ad_0 - g_0^T d_0 - b^T d_0 \\ &= (Ax_0 - b)^T d_0 - g_0^T d_0 \\ &= g_0^T d_0 - g_0^T d_0 = 0 \end{aligned}$$

Thus the orthogonality property holds for $k = 1$.
Suppose the relation holds for $0 < j \leq k-1$.That is, $g_k^T d_j = 0$, for $j \leq k-1$. Next we need to show that $g_{k+1}^T d_j = 0$, for $j \leq k$. fix $k > 0$ and $0 \leq j < k$, since $g_{k+1} = g_k + t_k Ad_k$, then we have

$$g_{k+1}^T d_j = g_k^T d_j + t_k d_k^T Ad_j \tag{2.11}$$

but by A-conjugate definition, $d_k^T Ad_j = 0$, for $j \neq k$ and by induction hypothesis $g_k^T d_j = 0$ for $j \leq k-1$, so the right side of equation (2.11) is zero. This implies

$$g_{k+1}^T d_j = 0, \text{ for } 0 \leq j < k$$

It remains to show for $j = k$ the relation holds.

$$\begin{aligned} g_{k+1}^T d_k &= (Ax_{k+1} - b)^T d_k \\ &= (x_k + t_k d_k)^T A d_k - b^T d_k \\ &= (x_k - \frac{g_k^T d_k}{d_k^T A d_k} d_k)^T A d_k - b^T d_k \\ &= x_k^T A d_k - \frac{g_k^T d_k}{d_k^T A d_k} d_k^T A d_k - b^T d_k \\ &= x_k^T A d_k - g_k^T d_k - b_k^T d_k \\ &= (Ax_k - b)^T d_k - g_k^T d_k \\ &= g_k^T d_k - g_k^T d_k = 0 \end{aligned}$$

□

Proof. (b) For $k = 1$, the hereditary property becomes $H_1 y_0 = s_0$ which is the secant condition of the update.
Now suppose for $k > 0$, the relation holds.That is,

$$H_k y_j = s_j \text{ for } 0 < j < k$$

Next we need to prove that the relation holds for $k+1$. From DFP update (2.7), we have

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k} \tag{2.12}$$

Pre-multiplying both sides of (2.12) by y_j we get

$$\begin{aligned} H_{k+1} y_j &= H_k - \frac{H_k y_k y_k^T H_k y_j}{y_k^T H_k y_k} + \frac{s_k s_k^T y_j}{s_k^T y_k} \\ &= s_j - \frac{H_k y_k (g_{k+1}^T - g_k^T) s_j}{y_k^T H_k y_k} + \frac{s_k 0}{s_k^T y_k} \\ &= s_j - \frac{H_k y_k (g_{k+1}^T s_j - g_k^T s_j)}{y_k^T H_k y_k} \\ &= s_j \end{aligned}$$

Next we need to show that for $j = k$ the hereditary property holds true. Pre-multiplying both sides of the DFP update (2.7) by y_k

$$\begin{aligned} H_{k+1} y_k &= H_k y_k - \frac{H_k y_k y_k^T H_k y_k}{y_k^T H_k y_k} + \frac{s_k s_k^T y_k}{s_k^T y_k} \\ &= H_k y_k - H_k y_k + s_k \\ &= s_k \end{aligned}$$

□

Proof. (c) From Hereditary property and Lemma 2.1, we have

$$H_n y_k = s_k \tag{2.13}$$

$$A s_k = y_k \tag{2.14}$$

Then substituting (2.14) into (2.13) we get

$$H_n A s_k = s_k$$

Since $s_{k's}$, k=0, 1, 2, ..., n-1 are linearly independent, it follows that

$$H_n A = I$$
$$\Rightarrow H_n = A^{-1}$$

□

The DFP quasi-Newton updating formula is quite effective, but it turns out that in the case of large non-quadratic problems the algorithm has the tendency of sometimes getting "stuck". This phenomenon is attributed to B_k(or H_k) becoming nearly singular. Also the DFP method is less effective in correcting bad Hessian approximations. This property is believed to be the reason for its poorer practical performance [**see [1], [3]**]. In the next section, we discuss another rank-2 quasi-Newton updating formula that alleviates this problem.

2.4 Broyden-Fletcher-Goldfarb-Shanno(BFGS) Rank-2 Update

In 1970, an alternative rank-2 quasi-Newton update was suggested independently by Broyden, Fletcher, Goldfarb, and Shanno. The method is now called the BFGS rank-2 update. In this section we drive this rank-2 update by the concept of duality and describes its theoretical properties on inverse Hessian update produced by DFP update.
Consider an update for the inverse Hessian, say

$$H_{k+1} = U(H_k, y_k, s_k)$$

which satisfy the scant condition on the inverse Hessian approximation (that is, $H_{k+1} y_k = s_k$). Then by exchanging $H_k \rightleftarrows B_k$ and $y_k \rightleftarrows s_k$, we obtain the update for Hessian approximation, that is, $B_{k+1} = U(B_k, s_k, y_k)$, which satisfy the scant condition.
B_{k+1} is formed by adding to B_k two symmetric matrices, each of rank-one. Now, we consider an update of the form

$$B_{k+1} = B_k + auu^T + bvv^T \tag{2.15}$$

where $u, v, \in \Re^n$. a and b are constants to be determined.
Now, by secant equation on the Hessian B_{k+1} we have:

$$B_{k+1} s_k = y_k$$

$$\Rightarrow (B_k + auu^T + bvv^T)s_k = y_k$$
$$B_k s_k + a(u^T s_k)u + b(v^T s_k)v = y_k$$

$$\Rightarrow a(u^T s_k)u + b(v^T s_k)v = y_k - B_k s_k \tag{2.16}$$

Equation (2.16) has not a unique solution. Let us now choice $u = y_k$ and $v = B_k s_k$.

$$\Rightarrow a(y_k^T s_k)y_k + b(s_k^T B_k s_k)B_k s_k = y_k - B_k s_k$$
$$\Rightarrow a(y_k^T s_k)y_k = y_k$$

and

$$b(s_k^T B_k s_k)B_k s_k = -B_k s_k$$
$$\Rightarrow a = \frac{y_k}{(y_k^T s_k)y_k}$$

and

$$b = \frac{-1}{s_k^T B_k s_k}$$

Substituting the value of a, b, u and v to the updating formula (2.15), we get

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \tag{2.17}$$

The updating formula (2.17) is called the Broyden-Fletcher-Goldfarb-Shanno (BFGS) rank-2 update.

Remark 2.2. *The updating formula (2.17) is only one of the possible choice and with other solutions we obtain different update formulas.*

2.4.1 Positive Definitiveness of The BFGS Update

In this subsection we will prove positive definitiveness of BFGS update.

Theorem 2.4. *(positive definitiveness of BFGS update): Given B_k is symmetric and positive definite, then the BFGS update*

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$$

produce B_{k+1} positive definite if $y_k^T s_k > 0$

Proof. Let $y_k^T s_k > 0$ and consider $x \neq 0$, $x \in \Re^n$. Now, we need to prove that $x^T B_{k+1} x > 0$.

$$x^T B_{k+1} x = x^T (B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k})x$$
$$= x^T B_k x + x^T \frac{y_k y_k^T}{y_k^T s_k} x - x^T \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} x$$

Since B_k is symmetric and positive definite matrix, so that there is a Cholescky factorization $LL^T = B_k$, where L is a lower triangular matrix.

$$\begin{aligned} x^T B_{k+1} x &= x^T (LL^T) x + x^T \frac{y_k y_k^T}{y_k^T s_k} x - x^T \frac{(LL^T) s_k s_k^T (LL^T)}{s_k^T (LL^T) s_k} x \\ &= (x^T L)(L^T x) - \frac{(x^T L)(L^T s_k)(s_k^T L)(L^T x)}{(s_k^T L)(L^T s_k)} + x^T \frac{y_k y_k^T}{y_k^T s_k} x \\ &= (L^T x)^T (L^T x) - \frac{(L^T x)^T (L^T s_k)(L^T s_k)^T (L^T x)}{(L^T s_k^T)(L^T s_k)} + \frac{(y_k^T x)^2}{y_k^T s_k} \end{aligned}$$

Putting $a = L^T x$ and $b = L^T s_k$

$$\begin{aligned} \Rightarrow x^T B_{k+1} x &= a^T a - \frac{a^T b b^T a}{b^T b} + \frac{(y_k)^T x}{y_k^T s_k} \\ &= \frac{(a^T a)(b^T b) - (b^T a)^2}{b^T b} + \frac{(y_k)^T x}{y_k^T s_k} \end{aligned}$$

Then from the Cauchy-Schwartz inequality, we have $(a^T a)(b^T b) \geq (b^T a)^2$
$\Rightarrow \frac{(a^T a)(b^T b) - (b^T a)^2}{b^T b} \geq 0$
$\Rightarrow x^T B_{k+1} x = \frac{(a^T a)(b^T b) - (b^T a)^2}{b^T b} + \frac{(y_k)^T x}{y_k^T s_k} > 0$
$\Rightarrow x^T B_{k+1} x > 0$
$\Rightarrow B_{k+1}$ is positive definite if $y_k^T s_k > 0$. □

Theorem 2.5. *If the step length t_k satisfies Wolfe conditions, then the curvature condition $y_k^T s_k$ holds and hence the BFGS update produce positive definite matrix.*

Proof. From a minimum search algorithm, we have

$$s_k = x_{k+1} - x_k = t_k d_k$$

with $t_k > 0$ is step length and d_k is descent direction. Also from the second Wolfe condition for line search, we have

$$\nabla f(x_k + t_k d_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k$$

for some $0 < c_2 < 1$
Then subtracting $\nabla f(x_k)^T d_k$ from both sides of Wolfe-(ii).

$$\begin{aligned} \nabla f(x_k + t_k d_k)^T d_k - \nabla f(x_k)^T d_k &\geq c_2 \nabla f(x_k)^T d_k - \nabla f(x_k)^T d_k \\ (\nabla f(x_k + t_k d_k)^T - \nabla f(x_k)^T) d_k &\geq c_2 \nabla f(x_k)^T d_k - \nabla f(x_k)^T d_k \\ (\nabla f(x_k + t_k d_k) - \nabla f(x_k))^T d_k &\geq (c_2 - 1) \nabla f(x_k)^T d_k \\ \Rightarrow y_k^T d_k &\geq (c_2 - 1) \nabla f(x_k)^T d_k > 0 \\ y_k^T \frac{s_k}{t_k} &> 0 \\ y_k^T s_k &> 0 \end{aligned}$$

□

Hence the curvature condition holds and BFGS update produce positive definite matrix.
A naive implementation of the BFGS formula (2.17) is not efficient for large unconstrained minimization problems, because it requires the system $B_k d_k = -\nabla f(x_k)$ to be solved for the step direction d_k thereby increase the cost of the step computation. To avoid this we need an equivalent formula for the inverse of the approximate Hessian in the next subsection. To derive this formula, we consider Sherman-Morrison-woodbury formula on theorem (1.3).

2.4.2 The BFGS Update for The Inverse of The Approximation Hessian

We can derive a version of the BFGS algorithm that works with the inverse Hessian approximation H_k rather than B_k. The update formula for H_k is obtained by simply applying the sherman-Morrison-Woodbury formula on theorem (1.3) to (2.17) as follow.

Proposition 2.2. *By using the Sherman-Morrison-Woodbury formula the BFGS update for the inverse Hessian approximation (say H) becomes*

$$H_{k+1} = H_k - \frac{H_k y_k s_k^T + s_k y_k^T H_k}{s_k^T y_k} + \frac{s_k s_k^T}{s_k^T y_k}(1 + \frac{y_k^T H_k y_k}{s_k^T y_k}) \tag{2.18}$$

or equivalently

$$H_{k+1} = (I - \frac{s_k y_k^T}{s_k^T y_k}) H_k (I - \frac{y_k s_k^T}{s_k^T y_k}) + \frac{s_k s_k^T}{s_k^T y_k} \tag{2.19}$$

Proof. Consider the sherman-morrison-woodbury formula with rank $k = 2$ and $u_1 = v_1 = \frac{y_k}{(s_k^T y_k)^{\frac{1}{2}}}$ and $u_2 = -v_2 = \frac{B_k s_k}{(s_k^T B_k s_k)^{\frac{1}{2}}}$. Now, we consider $C = I + v^T B_k^{-1} u$ is a 2×2 matrix with $c_{ij} = \delta_{ij} + v_i^T B_k^{-1} u_j$ where δ_{ij} is the (i, j) entry of a 2×2 identity matrix.

$$\begin{aligned}
c_{11} &= \delta_{11} + v_1^T B_k^{-1} u_1 \\
&= 1 + (\frac{(y_k}{(s_k^T y_k)^{\frac{1}{2}}})^T B_k^{-1} (\frac{y_k}{(s_k^T y_k)^{\frac{1}{2}}}) \\
&= 1 + \frac{y_k^T}{(y_k^T s_k)^{\frac{1}{2}}} B_k^{-1} \frac{y_k}{(s_k^T y_k)^{\frac{1}{2}}} \\
&= 1 + \frac{y_k^T B_k^{-1} y_k}{(y_k^T s_k s_k^T y_k)^{\frac{1}{2}}} \\
&= 1 + \frac{y_k^T B_k^{-1} y_k}{((s_k^T y_k)^{\frac{1}{2}})^2} \\
&= 1 + \frac{y_k^T B_k^{-1} y_k}{s_k y_k}
\end{aligned}$$

Setting $H_k = B_k^{-1}$

$\Rightarrow c_{11} = 1 + \frac{y_k^T H_k y_k}{s_k y_k}$

$$\begin{aligned} c_{22} &= \delta_{22} + v_2^T B_k^{-1} u_2 \\ &= 1 + (\frac{-B_k s_k}{(s_k^T B_k s_k)^{\frac{1}{2}}})^T B_k^{-1} (\frac{B_k s_k}{(s_k^T B_k s_k)^{\frac{1}{2}}}) \\ &= 1 - \frac{s_k^T B_k^T B_k^{-1} B_k s_k}{((s_k^T B_k s_k)^2)^{\frac{1}{2}}} \\ &= 1 - \frac{s_k^T B_k s_k}{s_k^T B_k s_k} = 0 \end{aligned}$$

$$\begin{aligned} c_{12} &= \delta_{12} + v_1^T B_k^{-1} u_2 \\ &= 0 + (\frac{y_k}{(s_k^T y_k)^{\frac{1}{2}}})^T B_k^{-1} (\frac{B_k s_k}{(s_k^T B_k s_k)^{\frac{1}{2}}}) \\ &= \frac{y_k^{-1} B_k^{-1} B_k s_k}{(y_k^T s_k)^{\frac{1}{2}} (s_k^T B_k s_k)^{\frac{1}{2}}} \\ &= \frac{y_k^T s_k}{(y_k^T s_k)^{\frac{1}{2}} (s_k^T B_k s_k)^{\frac{1}{2}}} \\ &= \frac{(y_k^T s_k)^{\frac{1}{2}}}{(s_k^T B_k s_k)^{\frac{1}{2}}} \end{aligned}$$

$$\begin{aligned} c_{21} &= \delta_{21} + v_2^T B_k^{-1} u_1 \\ &= 0 + \frac{(-B_k s_k)^T B_k^{-1} y_k}{(s_k^T B_k s_k)^{\frac{1}{2}} (s_k^T y_k)^{\frac{1}{2}}} \\ &= \frac{-s_k^T B_k B_k^{-1} y_k}{(s_k^T B_k s_k)^{\frac{1}{2}} (s_k^T y_k)^{\frac{1}{2}}} \\ &= -\frac{s_k^T y_k}{(s_k^T B_k s_k)^{\frac{1}{2}} (s_k^T y_k)^{\frac{1}{2}}} \\ &= -\frac{(s_k^T y_k)^{\frac{1}{2}}}{(s_k^T B_k s_k)^{\frac{1}{2}}} \end{aligned}$$

Now, the matrix C has the form
$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a & b \\ d & 0 \end{pmatrix}$.
Where $a = 1 + \frac{y_k^T B_k^{-1} y_k}{s_k y_k}$, $b = \frac{(s_k^T y_k)^{\frac{1}{2}}}{(s_k^T B_k s_k)^{\frac{1}{2}}}$, $d = -\frac{(s_k^T y_k)^{\frac{1}{2}}}{(s_k^T B_k s_k)^{\frac{1}{2}}}$. And then we get $C^{-1} = \begin{pmatrix} 0 & \frac{1}{d} \\ \frac{1}{b} & \frac{-a}{bd} \end{pmatrix}$
Setting $A = H_k u$ and $B = H_k v$, where $A = (a_i)$, $B = (b_i)$ and $a_i = H_k u_i$, $b_i = H_k v_i$.
Then from Sherman-Morrison-Woodbury formula on theorem (1.3), we have

$$\begin{aligned} H_{k+1} &= H_k - H_k u C^{-1} v^T H_k \\ &= H_k - A C^{-1} B^T \end{aligned}$$

Here the matrix product $(AC^{-1}B^T)$ is $\Re^{n\times 2}\times\Re^{2\times 2}\times\Re^{2\times n}$ so,

$$\begin{aligned}
AC^{-1}B^T &= \begin{pmatrix} a_1 & a_2 \end{pmatrix}\begin{pmatrix} 0 & \frac{1}{d} \\ \frac{1}{b} & \frac{-a}{bd}\end{pmatrix}\begin{pmatrix} b_1^T \\ b_2^T\end{pmatrix} \\
&= \begin{pmatrix} \frac{a_2}{b} & \frac{a_1}{d}-\frac{aa_2}{bd}\end{pmatrix}\begin{pmatrix} b_1^T \\ b_2^T\end{pmatrix} \\
&= \frac{a_2b_1^T}{b}+(\frac{a_1}{d}-\frac{aa_2}{bd})b_2^T \\
&= \frac{a_2b_1^T}{b}+\frac{a_1b_2^T}{d}-\frac{aa_2b_2^T}{bd} \\
\Rightarrow AC^{-1}B^T &= \frac{H_ku_2v_1^TH_k}{b}+\frac{H_ku_1v_2^TH_k}{d}-\frac{a}{bd}H_ku_2v_2^TH_k
\end{aligned}$$

Substituting the values of $a, b, d, u_{i's}$ and $v_{i's}$, we get

$$AC^{-1}B^T = \frac{(s_k^TB_ks_k)^{\frac{1}{2}}H_kB_ks_ky_k^TH_k}{(s_k^TB_ks_k)^{\frac{1}{2}}(y_k^Ts_k)^{\frac{1}{2}}(s_k^Ty_k)^{\frac{1}{2}}}$$

+

$$\frac{(s_k^TB_ks_k)^{\frac{1}{2}}H_ky_ks_k^TB_kH_k}{(s_k^Ty_k)^{\frac{1}{2}}(s_k^Ty_k)^{\frac{1}{2}}(s_k^TB_ks_k)^{\frac{1}{2}}}-(1+\frac{y_k^TH_ky_k}{s_k^Ty_k})\frac{H_kB_ks_ks_k^TB_kH_k}{((s_k^TB_ks_k)^2)^{\frac{1}{2}}}\frac{((s_k^TB_ks_k)^2)^{\frac{1}{2}}}{(y_k^Ts_k)^{\frac{1}{2}}(s_k^Ty_k)^{\frac{1}{2}}}$$

$$\begin{aligned}
&= \frac{s_ky_k^TH_k}{(y_k^Ts_k)^{\frac{1}{2}}(s_k^Ty_k)^{\frac{1}{2}}}+\frac{H_ky_ks_k^T}{((s_k^Ty_k)^2)^{\frac{1}{2}}}-\frac{(1+\frac{y_k^TH_ky_k}{s_k^Ty_k})s_ks_k^T}{(y_k^Ts_k)^{\frac{1}{2}}(s_k^Ty_k)^{\frac{1}{2}}} \\
&= \frac{s_ky_k^TH_k}{s_k^Ty_k}+\frac{H_ky_ks_k^T}{s_k^Ty_k}-(1+\frac{y_k^TH_ky_k}{s_k^Ty_k})\frac{s_ks_k^T}{s_k^Ty_k} \\
&= \frac{H_ky_ks_k^T+s_ky_k^TH_k}{s_k^Ty_k}-\frac{s_ks_k^T}{s_k^Ty_k}(1+\frac{y_k^TH_ky_k}{s_k^Ty_k}) \\
\Rightarrow H_{k+1} &= H_k-AC^{-1}B^T \\
\Rightarrow H_{k+1} &= H_k-\frac{H_ky_ks_k^T+s_ky_k^TH_k}{s_k^Ty_k}+\frac{s_ks_k^T}{s_k^Ty_k}(1+\frac{y_k^TH_ky_k}{s_k^Ty_k})
\end{aligned}$$

□

Note 2.4. *1) The updating formula (2.19) is a straightforward calculation of the update (2.18).*

2) As we have proved on theorem 2.4, the Hessian approximation B_{k+1} produced by BFGS method is positive definite, which also guarantees that its inverse Hessian approximation H_{k+1} is positive definite.

The corresponding algorithm to find a minimum of a differentiable function f defined on $\Re^n$ is defined as follow.

Algorithm 2.3. *(Quasi-Newton method with BFGS update)*

Initial step: *Choose sufficiently small tolerance $\epsilon > 0$, an initial guess $x_1 \in \Re^n$, and an $n \times n$ symmetric and positive definite matrix H_1 (say an $n \times n$ identity matrix or a positive modification of inverse Hessian at x_1)*
Let $k = 0$;

Main step: *while $\| \nabla f(x_k) \| > \epsilon$, do*

1) Compute search direction: $d_k = -H_k \nabla f(x_k)$;

2) Approximate min $f(x + td_k), t > 0$ by line search and if t_k be (exact or inexact) solution of the one-dimensional problem, min $f(x + td_k), t > 0$ go to step (3).

3) $x_{k+1} = x_k + t_k d_k$

4) To update H_k

► *Let $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$*

► *Compute H_{k+1} by (2.19)*

5) set $k = k + 1$ and go to step (1)

end

BFGS method has the following important properties:

1) For a quadratic function (under exact line search)

   a) BFGS update has quadratic termination. That is, $H_n = A^{-1}$, where A is an $n \times n$ symmetric and positive definite Hessian matrix of the quadratic function and H_n is Hessian approximation updated by BFGS method.

   b) BFGS update has hereditary property, that is $H_i y_j = s_j$ for $j < i$.

2) For a general function

   a) BFGS update maintains positive definiteness.

   b) BFGS update is supper linear convergent.

   c) For a strictly convex function, under exact line search, BFGS method is globally convergent.

   d) The BFGS formula has very effective self-correcting properties. If the matrix B_k (or H_k) incorrectly estimates the curvature in the objective function, and if this bad estimate slows down the iteration, then The Hessian approximation will tend to correct itself within a few steps [**see [3]**].

The positive definiteness of BFGS update is already proved on theorem (2.4). The convergence properties of BFGS update will be established in the subsections (2.5.2) and (2.6.1). In the remainder of this section we shall prove the other two important properties: quadratic termination of the method and the hereditary property of the method on theorem (2.6) below.

Theorem 2.6. *(property of BFGS)*
Let $f(x) = \frac{1}{2}x^T Ax - b^T + c$ be a quadratic function with symmetric and positive definite Hessian $A \in \Re^{n\times n}$. Then for any starting point x_0 any starting symmetric and positive definite inverse Hessian H_0, and let the iterates x_k and H_k produced by the sequence s_k. That is,

1) $x_{k+1} = x_k + s_k$

2) H_{k+1} *is produced by the BFGS update (2.19)*

where $s_k = t_k d_k$ with t_k is a scalar obtained by exact line search and $d_{i's}, i = 0, 1, 2, ..., m$ are conjugates. Then for $0 < j < k$ the sequence s_k generated from BFGS update satisfies

(a) $g_k^T s_j = 0$ *[orthogonality property]*

(b) $H_k y_j = s_j$ *[Hereditary property]*

(c) If s_0, s_1, s_2,..., s_{n-1} are linearly independent, the $H_n = A^-$ and the method terminates at x_m with $m < n$.

Proof. (a): The prove is the same as the prove of (a) on theorem 2.3. □

Proof. (b): For $k = 1$, the hereditary property becomes $H_1 y_0 = s_0$ which is the secant condition of the update.
Now suppose for $k > 0$, the relation holds. That is,

$$H_k y_j = s_j \text{ for } 0 < j < k$$

Now, we need to prove that the relation holds for $k + 1$. post-multiplying both sides of the BFGS update (2.19) by y_j, we get

$$\begin{aligned} H_{k+1}y_j &= (I - \frac{s_k y_k^T}{s_k^T y_k})H_k(y_j - \frac{s_k^T y_j}{y_k s_k^T}y_k) + \frac{s_k s_k^T y_j}{s_k^T y_k} \\ &= (I - \frac{s_k y_k^T}{s_k^T y_k})H_k y_j \\ &= s_j - \frac{y_k^T s_j}{s_k s_k^T}y_k \\ &= s_j \end{aligned}$$

It remains to show that for $j = k$ the condition holds. Pre-multiplying both sides of the DFP update (2.19) by y_k, we have

$$\begin{aligned} H_{k+1}y_k &= (I - \frac{s_k y_k^T}{s_k^T y_k})H_k(y_k - \frac{s_k^T y_k}{y_k s_k^T}y_k) + \frac{s_k s_k^T y_k}{s_k^T y_k} \\ &= s_k. \end{aligned}$$

□

Proof. (c): From Hereditary property and Lemma 2.1, we have

$$H_n y_k = s_k \qquad (I)$$

$$A s_k = y_k \qquad (II)$$

Then substituting (II) into (I), we get

$$H_n A s_k = s_k$$

Since $s_{k's}$,k=0, 1, 2, ..., n-1 are linearly independent, it follows that

$$H_n A = I$$
$$H_n = A^{-1}$$

□

Note 2.5. *The DFP and BFGS updating formulas are duals or complementary of each other, in the sense that one can be obtained from the other by the interchanges* $s_k \rightleftharpoons y_k$ *and* $B_k \rightleftharpoons H_k$.

2.5 Global Convergence of Quasi-Newton Methods

In this section we discuss the global convergence for quasi-Newton methods (BFGS and DFP updates) under exact line search and inexact line search respectively in subsection (2.5.1) and (2.5.2). In the discussion of this section, we need the following assumptions.

Assumption 2.1. *a)* $f : \Re^n \longrightarrow \Re$ *is continuously differentiable on convex set* D.

b) $f(x)$ *is strongly convex, i.e., there exist a positive constants* m *and* M *such that for all* $x \in L(x) = \{x | f(x) \leq f(x_0)\}$, *which is convex, we have*

$$m\|u\|^2 \leq u^T \nabla^2 f(x) u \leq M\|u\|^2, \forall u \in \Re^n \qquad (2.20)$$

The assumption (b) *implies that* $\nabla^2 f(x)$ *is positive definite on* $L(x)$, *and that* f *has a unique minimizer* x^* *in* $L(x)$.

2.5.1 Global Convergence under Exact Line Search

We begin the discussion in case of exact line search. Let

$$\bar{G} = \int_0^1 \nabla^2 f(x_k + \tau s_k) d\tau \qquad (2.21)$$

then from Taylor's theorem that

$$y_k = \bar{G} s_k \qquad (2.22)$$

We have

$$m \leq \frac{y_k^T s_k}{\|s_k\|^2} = \frac{s_k \bar{G}_k s_k}{\|s_k\|^2} \leq \qquad (2.23)$$

and

$$\frac{1}{M} \leq \frac{\|s_k\|^2}{y_k^T s_k} \leq \frac{1}{m} \tag{2.24}$$

Since also

$$\frac{\|y_k\|^2}{s_k^T y_k} = \frac{s_k^T \bar{G_k}^2 s_k}{s_k^T \bar{G} s_k}$$

if we let $z_k = \bar{G_k}^{\frac{1}{2}} s_k$, then

$$\frac{\|y_k\|^2}{s_k^T y_k} = \frac{z_k^T \bar{G} z_k}{z_k^T z_k} \leq M \tag{2.25}$$

In addition, we have

$$\|y_k\| \leq \|\bar{G}\| \|s_k\|, \quad \|s_k\| \leq \|\bar{G_k}^{-1}\| \|y_k\|$$

which give

$$\frac{\|y_k\|}{\|s_k\|} \leq M \tag{2.26}$$

and

$$\frac{\|s_k\|}{\|y_k\|} \leq \frac{1}{m} \tag{2.27}$$

Therefore from the above discussion, we have

Lemma 2.2. *Let $f : \Re^n \longrightarrow \Re$ satisfy assumption (2.1). Then*

$$\frac{\|s_k\|}{\|y_k\|}, \quad \frac{\|y_k\|}{\|s_k\|}, \quad \frac{s_k^T y_k}{\|s_k\|^2}, \quad \frac{s_k^T y_k}{\|y_k\|^2}, \quad \frac{\|y_k\|^2}{s_k^T y_k}$$

are bounded.

Lemma 2.3. *under exact line search, $\sum \|s_k\|^2 and \sum \|y_k\|^2$ are convergent.*

Proof. [*see* [2]]. □

Lemma 2.4. *For all vectors x, the inequality*

$$\|g(x)\|^2 \geq m[f(x) - f(x^*)] \tag{2.28}$$

holds, where $f(x^*)$ is the minimizer of $f(x)$.

Proof. [*see* [2]]. □

Theorem 2.7. *Suppose that $f(x)$ satisfies Assumption (2.1). Then, under exact line search , the sequence x_k generated by DFP method converges to the minimizer x^* of f.*

Proof. Consider DFP formula of the inverse Hessian approximation

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k}. \tag{2.29}$$

and DFP method formula of the Hessian approximation

$$B_{k+1} = (I - \frac{y_k s_k^T}{s_k^T y_k}) B_k (I - \frac{s_k y_k^T}{s_k^T y_k}) + \frac{y_k y_k^T}{s_k^T y_k}. \tag{2.30}$$

By computing the trace of (2.30), we have

$$Tr(B_{k+1}) = Tr(B_k) - 2\frac{s_k^T B_k y_k}{s_k^T y_k} + \frac{(s_k^T B_k s_k)(y_k^T y_k)}{(s_k^T y_k)^2} + \frac{y_k^T y_k}{s_k^T y_k} \tag{2.31}$$

The middle two terms can be written as

$$\begin{aligned} -2\frac{s_k^T B_k y_k}{s_k^T y_k} + \frac{(s_k^T B_k s_k)(y_k^T y_k)}{(s_k^T y_k)^2} &= t_k[2\frac{g_k^T y_k}{s_k^T y_k} + \frac{(-g_k^T s_k)(y_k^T y_k)}{(s_k^T y_k)^T}] \\ &= t_k 2\frac{g_k^T y_k + y_k^T y_k}{s_k^T y_k} \\ &= \frac{\|g_{k+1}\|^2 - \|g_k\|^2}{g_k^T H_k g_k} \end{aligned} \tag{2.32}$$

Since $g_{k+1}^T s_k = 0$, then

$$\begin{aligned} g_{k+1} H_{k+1} g_{k+1} &= g_{k+1}^T [H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}] g_{k+1} \\ &= g_k^T [H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}] g_k \\ &= g_k^T [H_k - \frac{H_k g_k g_k^T H_k}{y_k^T H_k y_k}] g_k \\ &= \frac{(g_k^T H_k g_k)(g_{k+1}^T H_k g_{k+1})}{g_{k+1} H_k g_k + g_{k+1}^T H_{k+1} g_{k+1}} \end{aligned}$$

By finding the inverse number of the above, expression, we get

$$\frac{1}{g_{k+1} H_{k+1} g_{k+1}} = \frac{1}{g_{k+1}^T H_k g_{k+1}} + \frac{1}{g_k^T H_k g_k} \tag{2.33}$$

Using (2.32) and (2.33), then (2.31) becomes

$$Tr(B_{k+1}) = Tr(B_k) + \frac{\|g_{k+1}\|^2}{g_{k+1} H_k g_{k+1}} - \frac{\|g_k\|^2}{g_k^T H_k g_k} - \frac{\|g_{k+1}\|^2}{g_{k+1} H_k g_{k+1}} + \frac{\|y_k\|^2}{s_k^T y_k} \tag{2.34}$$

By recurrence, we obtained

$$Tr(B_{k+1}) = Tr(B_0) + \frac{\|g_{k+1}\|^2}{g_{k+1}^T H_{k+1} g_{k+1}} - \frac{\|g_0\|^2}{g_0^T H_0 g_0} - \sum_{j=0}^{k} \frac{\|g_{j+1}\|^2}{g_{j+1}^T H_j g_{j+1}} + \sum_{j=0}^{k} \frac{\|y_j\|^2}{s_j^T y_j} \tag{2.35}$$

Therefore, by Lemma (2.2), there exists a positive number M which is independent of k, such that

$$Tr(B_{k+1} \leq \frac{\|g_{k+1}\|^2}{g_{k+1}^T H_{k+1} g_{k+1}} - \sum_{j=0}^{k} \frac{\|g_{j+1}\|^2}{g_{j+1}^T H_j g_{j+1}} + Mk \tag{2.36}$$

In the left part, we will prove that if the theorems does not hold, then the sum of the last two terms in (2.36) is positive.
Now consider the trace of H_{k+1}. From (2.29),we have

$$Tr(H_{k+1}) = Tr(H_0) - \sum_{j=0}^{k} \frac{\|H_j y_j\|^2}{y_j^T H_j y_j} + \sum_{j=0}^{k} \frac{\|s_j\|^2}{s_j^T y_j} \tag{2.37}$$

Since H_{k+1} is positive definite, the right hand side of (2.37) is positive. By Lemma (2.2) $m > 0$ which is independent of k, such that

$$\sum_{j=0}^{k} \frac{\|H_j y_j\|^2}{y_j^T H_j y_j} < \frac{k}{m} \tag{2.38}$$

Note that

$$(y_j^T H_j y_j)^2 \leq \|H_j y_j\|^2 \|y_j\|^2 \tag{2.39}$$

and

$$\begin{aligned} y_j^T H_j y_j &= g_{j+1}^T H_j g_{k+1} + g_j^T H_j g_j + 2 g_{j+1}^T d_j \\ &> g_{j+1}^T H_j g_{k+1} \end{aligned} \tag{2.40}$$

By the positive definiteness of H_j and exact line search, then by using (2.38), (2.39) and (2.40) in turn, we obtain

$$\sum_{j=0}^{k} \frac{g_{j+1}^T H_j g_{j+1}}{\|y_k\|^2} \leq \sum_{j=0}^{k} \frac{y_j^T H_j y_j}{\|y_j\|^2} \leq \sum_{j=0}^{k} \frac{\|H_j y_j\|^2}{y_j^T H_j y_j} \leq \frac{k}{m} \tag{2.41}$$

By using cauchy-Schwartz inequality and (2.41)

$$\begin{aligned} \sum_{j=0}^{k} \frac{\|g_{j+1}\|^2}{g_{j+1}^T H_j g_{j+1}} &\geq \frac{(\sum_{j=0}^{k} \frac{\|g_{j+1}\|}{\|y_j\|})^2}{\sum_{j=0}^{k} \frac{g_{j+1}^T H_j g_{j+1}}{\|y_j\|^2}} \\ &\geq \frac{m}{k} (\sum_{j=0}^{k} \frac{\|g_{j+1}\|}{\|y_j\|})^2 \end{aligned} \tag{2.42}$$

Now suppose that the theorem is not true, that is, there exists $\delta > 0$ such that for all sufficiently large k,

$$\|g_k\| \geq \delta \tag{2.43}$$

Also, by Lemma (2.4), there exists a constant $\eta > 0$ such that

$$f(x_k) - f(x_{k+1}) \geq \frac{1}{2} \eta \|s_k\|^2$$

which gives $\|s_k\| \longrightarrow 0$ and farther $\|y_k\| \longrightarrow 0$. Then by (2.42) and (2.43), we deduce, for k sufficiently large, that

$$\sum_{j=0}^{k} \frac{\|g_{j+1}\|^2}{g_{j+1}^T H_j g_{j+1}} > Mk \tag{2.44}$$

the above inequality implies that the sum of the last two terms in (2.36) is negative. by (2.44) and (2.36), we obtained

$$Tr(B_{k+1}) < \frac{\|g_{k+1}\|^2}{g_{k+1} H_{k+1} g_{k+1}} \tag{2.45}$$

Not that, for a symmetric and positive definite matrix, the inverse of trace is the lower bound of the least eigne values of inverse of the matrix. Then, it follows from (2.45) that

$$\frac{g_{k+1}^T H_{k+1} g_{k+1}}{\|g_{k+1}\|^2} < \mu \tag{2.46}$$

where μ is the lower bound of the least eigenvalue of H_{k+1}. From the property of Rayleigh quotient, we have

$$\frac{g_{j+1}^T H_{k+1} g_{k+1}}{\|g_{k+1}\|} > \mu \tag{2.47}$$

which contradicts (2.46). This contradiction proves that $\{x_k\}$ converges to x^* and that our theorem holds. □

2.5.2 Global Convergence under Inexact Line Search

Now, we study the global convergence of the BFGS method, with a practical line search. Let us rewrite BFGS method as follows

$$x_{k+1} = x_k + s_k = x_k + t_k d_k = x_k - t_k B_k^{-1} g_k \tag{2.48}$$

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k} \tag{2.49}$$

Theorem 2.8. *Let x_0 and B_0 be a starting point and a symmetric positive definite initial matrix, respectively. Suppose that $f(x)$ satisfies Assumption (2.1). Then under inexact line search, the sequence $\{x_k\}$ generated by BFGS method converges to the minimizer x^* of f.*

Proof. By computing the trace and determinant of BFGS formula (2.49), we obtain that

$$Tr(B_{k+1}) = Tr(B_k) - \frac{\|B_k s_k\|}{s_k^T B_k s_k} + \frac{\|y_k\|^2}{y_k^T s_k} \tag{2.50}$$

and

$$det(B_{k+1}) = det(B_k) \frac{y_k^T s_k}{s_k^T B_k s_k} \tag{2.51}$$

Let us define

$$m_k = \frac{y_k^T s_k}{s_k^T s_k}, \qquad M_k = \frac{y_k^T y_k}{y_k^T s_k} \tag{2.52}$$

It follows from (2.50) and (2.52) that

$$m \le m_k \le M, \qquad m \le M_k \le M \tag{2.53}$$

let us also define

$$\cos\theta_k = \frac{s_k^T B_k s_k}{\|s_k\| \|B_k s_k\|}, \qquad q_k = \frac{s_k^T B_k s_k}{s_k^T s_k} \tag{2.54}$$

we the obtain that

$$\frac{\|B_k s_k\|^2}{s_k^T B_k s_k} = \frac{\|B_k s_k\|^2 \|s_k\|^2}{(s_k^T B_k s_k)^2} \frac{s_k^T B_k s_k}{\|s_k\|^2} = \frac{q_k}{\cos^2 q_k} \tag{2.55}$$

In addition, we have from (2.50) that

$$det(B_{k+1}) = det(B_k)\frac{y_k^T s_k}{s_k^T s_k}\frac{s_k^T s_k}{s_k^T B_k s_k} = det(B_k)\frac{m_k}{q_k} \tag{2.56}$$

Now we introduce the following function of a positive definite matrix B_k:

$$\psi(B_k) = Tr(B_k) - \ln(det(B_k)) \tag{2.57}$$

where ln(.) denotes the natural logarithm. By using (2.50)-(2.55), we have

$$\begin{aligned}\psi(B_{k+1}) &= Tr(B_k) + M_k - \frac{q_k}{\cos^2 q_k} - \ln(det(B_k)) - \ln m_k + \ln q_k \\ &= \psi(B_k) + [M_k - \ln m_k - 1] + [1 - \frac{q_k}{\cos^2 q_k} + \ln\frac{q_k}{\cos^2 q_k}] + \ln\cos^2\theta_k \end{aligned} \tag{2.58}$$

Note that the function $h(t) = 1 - t + \ln t \leq 0$ for all $t > 0$. Hence the term inside the square bracket is non positive, and thus by summing both sides of (2.58), we have

$$0 < \psi(B_{k+1}) \leq \psi(B_1) + ck + \sum_{j=0}^{k} \ln\cos^2\theta_j \tag{2.59}$$

where the constant $c = M - \ln m - 1$ is assumed to be positive without loss of generality. It follows that

$$\lim_{k\to\infty} \|g_k\| \cos\theta_k = 0 \tag{2.60}$$

If θ_k is bounded away from 90^0, there is a positive constant δ such that $\cos\theta_k > \delta$, for k sufficiently large, and thus we have our result. Now assume by contradiction, that $\cos\theta_k \to 0$. Then there $k_1 > 0$ such that for all $j > k_1$, we have

$$\ln\cos^2\theta_k < -2c$$

where c is the constant defined above. By using (2.59), we deduce, for all $k > k_1$, that

$$\begin{aligned} 0 < \psi(B_1) + ck + \sum_{j=1}^{k} \ln\cos^2\theta_j + \sum_{j=k_1+1}^{k} (-2c) & \\ = \psi(B_1) + \sum_{j=1}^{k} \in \cos^2\theta_k + 2ck_1 - ck & \\ < 0 & \end{aligned}$$

which gives contradiction. Therefore the assumption $\cos\theta_j \to 0$ is not true, and there exist a sequence $\{x_k\}$ such that

$$\{\cos\theta_k\} \geq \delta > 0.$$

which means

$$\liminf \|\nabla f(x_k)\| = 0 \tag{2.61}$$

Since the problem is strong convex, then (2.61) implies $x_k \longrightarrow x^*$. □

2.6 Convergence Rate of Quasi-Newton Methods

2.6.1 Super Linear Convergence of BFGS Update

In this subsection,we will discuss the super linear convergence of BFGS method. For the result that follow we need to make a precise assumptions about the objective function formally as follow.

Assumption 2.2.

i) The objective function f is twice continuously differentiable.

ii) The level set $L = \{x \in \Re^n | f(x) \leq f(x_0)\}$ is convex, and there exist positive constants m and M such that

$$m\|\|^2 \leq z^T G(x) z \leq M\|z\|^2 \tag{2.62}$$

for all $z \in \Re^n$ and $x \in L$

iii) The Hessian matrix G is Lipschitz continuous at x^*, that is

$$\|G(x) - G(x^*)\| \leq L\|x - x^*\|$$

for all x near x^*, where L is Lipschitz positive constant.

Let $s_k^* = G_*^{-\frac{1}{2}} s_k$, $y_k^* = G_k^{-\frac{1}{2}} y_k$, $B_k^* = G_*^{-\frac{1}{2}} B_k G_*^{-\frac{1}{2}}$, where $G_* = G(x^*)$ and x^* is a minimizer of f. Define

$$\cos\theta_k^* = \frac{s_k^{*T} B_k^* s_k^*}{\|s_k^*\|\|B_k^* s_k^*\|}$$

so that θ_k^* is the angle between s_k^* and $B_k^* s_k^*$

$$q_k^* = \frac{s_k^{*T} B_k^* s_k^*}{\| s_k^*|}$$

and

$$M_k^* = \frac{\|y_k^*\|^2}{y_k^{*T} s_k^*}$$

$$m_k^* = \frac{y_k^{*T} s_k^*}{s_k^{*T} s_k^*}$$

Let $\widetilde{G_k}$ be the average Hessian defined by

$$\widetilde{G_k} = \int_0^1 \nabla^2 f(x_k + \tau t_k d_k) d\tau$$

The property

$$y_k = \widetilde{G_k} t_k d_k = \widetilde{G_k} s_k \tag{2.63}$$

By pre- and post-multiplying the BFGS update formula (2.49) by $G_*^{-\frac{1}{2}}$ and grouping terms appropriately,we obtain:

$$G_*^{-\frac{1}{2}} B_{k+1} G_*^{-\frac{1}{2}} = G_*^{-\frac{1}{2}} B_k G_*^{-\frac{1}{2}} - \frac{G_*^{-\frac{1}{2}} B_k s_k s_k^T B_k G_*^{-\frac{1}{2}}}{s_k^T B_k s_k} + \frac{G_*^{-\frac{1}{2}} y_k y_k^T G_*^{-\frac{1}{2}}}{y_k^T s_k} \tag{2.64}$$

$$B_{k+1}^* = B_k^* - \frac{B_k^* s_k^* s_k^{*T}}{s_k^{*T} B_k^* s_k^*} + \frac{y_k^* y_k^{*T}}{y_k^{*T} s_k^*} \tag{2.65}$$

By computing the trace and determinant of (2.65), we obtain that

$$trace(B^*_{k+1}) = trace(B^*_k) - \frac{\|B^*_k s^*_k\|^2}{s^{*T}_k B^*_k s^*_k} + \frac{\|y^*_k\|^2}{y^{*T}_k s^*_k} \tag{2.66}$$

$$det(B^*_{k+1}) = det(B^*_k)\frac{y^{*T}_k s^*_k}{s^{*T}_k B^*_k s^*_k} = det(B^*_{k+1})\frac{m^*_k}{q^*_k} \tag{2.67}$$

Also we obtain that

$$\frac{\|B^*_k s^*_k\|^2}{s^{*T}_k B^*_k s^*_k} = \frac{\|B^*_k s^*_k\|^2}{\|s^*_k\|^2 (s^{*T}_k B^*_k s^*_k)^2}\frac{s^{*T}_k B^*_k s^*_k}{\|s^*_k\|^2} = \frac{q^*_k}{\cos\theta^*_k} \tag{2.68}$$

We can combine the trace and determinant by introducing the following function of a positive definite matrix B.

$$\psi(B) = trace(B) - det(B) \tag{2.69}$$

where ln(.) denotes the natural logarithm. We then obtain that

$$\psi(B^*_{k+1}) = trace(B^*_k) + M^*_k - \frac{q_k}{\cos\theta^*_k} - \ln(det(B^*_k)) - \ln m^*_k + \ln q^*_k$$

$$\Rightarrow \psi(B^*_{k+1}) = \psi(B_k*) + (M^*_k - \ln m^*_k - 1) + [1 - \frac{q^*_k}{\cos^2\theta^*_k} + \ln\frac{q^*_k}{\cos^2\theta^*_k}$$

$$] + \ln\cos^2\theta^*_k$$

By using (2.63), we have that

$$y_k - G_* s_k = (\widetilde{G_k} - G_*)s_k$$

and thus

$$y^*_k - s^*_k = G_*^{-\frac{1}{2}}(\widetilde{G_k} - G_*)G_*^{-\frac{1}{2}} s^*_k$$

By assumption (2.2) and by Cauchy Schwartz inequality we have

$$\|y^*_k - s^*_k\| \leq \|G_*^{-\frac{1}{2}}\|^2 \|s^*_k\| \|\widetilde{G_k} - G_*\| \leq \|G_*^{-\frac{1}{2}}\|^2 \|s^*_k\| L\epsilon_k$$

where ϵ_k is defined by

$$max\|x_{k+1} - x^*\|, \|x_{k+1} - x^*\|$$

$$\Rightarrow \frac{\|y^*_k - s^*_k\|}{\|s^*_k\|} \leq \|G_*^{-\frac{1}{2}}\|^2 L\epsilon_k$$

$$\Rightarrow \frac{\|y^*_k - s^*_k\|}{\|s^*_k\|} \leq c^*\epsilon_k \tag{2.70}$$

for some positive constant c^*. Now we are in a position to prove a super linear theorem.

Theorem 2.9. *Let f be twice continuously differentiable and the Hessian matrix G be Lipschitz continuous at x^*. Suppose that the sequence generated by the BFGS algorithm converges to a minimizer x^* and that the condition*

$$\sum_{1}^{\infty} \|x_k - x^*\| < \infty \tag{2.71}$$

holds. Then x_k converges to x^* at a super linear rate.

Proof. From (2.70), we have

$$\|y_k^* - s_k^*\| \leq c^* \epsilon_k \|s_k^*\|.$$

It follows that

$$\|y_k^*\| - \|s_k^*\| \leq c^* \epsilon_k \|s_k^*\| \tag{2.72}$$
$$\|s_k^*\| - \|y_k^*\| \leq c^* \epsilon_k \|s_k^*\| \tag{2.73}$$

and from (2.72), we get

$$\|y_k^*\| \leq (1 + c^* \epsilon_k)\|s_k^*\| \tag{2.74}$$

and from (2.73), we get

$$(1 - c^* \epsilon_k)\|s_k^*\| \leq \|y_k^*\| \tag{2.75}$$

also by combining (2.72) and (2.73), we get

$$(1 - c^* \epsilon_k)\|s_k^*\| \leq \|y_k^*\| \leq (1 + c^* \epsilon_k)\|s_k^*\| \tag{2.76}$$

Then squaring (2.70)

$$\begin{aligned}
\|y_k^* - s_k^*\|^2 &\leq c^{*2} \epsilon_k^2 \|s_k^*\|^2 \\
\Rightarrow (y_k^* - s_k^*)^T (y_k^* - s_k^*) &\leq c^{*2} \epsilon_k^2 \|s_k^*\|^2 \\
y_k^{*T} y_k^* - 2 y_k^{*T} s_k^* + s_k^{*T} s_k^* &\leq c^{*2} \epsilon_k^2 \|s_k^*\|^2 \\
\|y_k^*\|^2 - 2 y_k^{*T} s_k^* + \|s_k^*\|^2 &\leq c^{*2} \epsilon_k^2 \|s_k^*\|^2
\end{aligned}$$

by using (2.76), we get

$$\begin{aligned}
(1 - c^* \epsilon_k)^2 \|s_k^*\|^2 - 2 y_k^{*T} s_k^* + \|s_k^*\|^2 \leq \|y_k^*\|^2 - 2 y_k^{*T} s_k^* + \|s_k^*\|^2 &\leq c^{*2} \epsilon_k^2 \|s_k^*\|^2 \\
(1 - c^* \epsilon_k)^2 \|s_k^*\|^2 + \|s_k^*\|^2 - c^{*2} \epsilon_k^2 \|s_k^*\|^2 &\leq 2 y_k^{*T} s_k^* \\
((1 - c^* \epsilon_k)^2 + 1 - c^{*2} \epsilon_k^2) \|s_k^*\|^2 &\leq 2 y_k^{*T} s_k^* \\
((1 - c^* \epsilon_k)(1 - c^* \epsilon_k) + 1 - c^{*2} \epsilon_k^2) \|s_k^*\|^2 &\leq 2 y_k^{*T} s_k^* \\
(2 - 2 c^* \epsilon_k) \|s_k^{*2}\| &\leq 2 y_k^{*T} s_k^* \\
\Rightarrow \quad y_k^{*T} s_k^* &\geq (1 - c^* \epsilon_k) \|s_k^{*2}\| \qquad (d)
\end{aligned}$$

It follow from the definition of m_k^* that

$$m_k^* = \frac{y_k^{*T} s_k^*}{s_k^{*T} s_k^*} \geq \frac{(1 - c^* \epsilon_k)\|s_k^*\|^2}{\|s_k^*\|^2}$$

$$m_k^* = \frac{y_k^{*T} s_k^*}{\|s_k^*\|^2} \geq 1 - c^* \epsilon_k \tag{2.77}$$

and from the definition of M_k^*

$$M_k^* = \frac{\|y_k^{*2}\|}{y_k^{*T}} \leq \frac{(1 + c^*\epsilon_k)\|s_k^*\|^2}{(1 - c^*\epsilon_k)\|s_k^*\|^2}$$

$$M_k^* = \frac{\|y_k^{*2}\|}{y_k^{*T}} \leq \frac{1 + c^*\epsilon_k}{1 - c^*\epsilon_k} \tag{f}$$

Since $x_k \to x^*$, we have that $\epsilon_k \to 0$, and by (f) there exist $c \geq c^*$ such that the following inequality hold for all sufficiently large k.

$$M_k^* \leq 1 + \frac{2c^*}{1 - c^*\epsilon_k}\epsilon_k \leq 1 + c\epsilon_k \tag{2.78}$$

Now,since the function $h(t) = 1 - t + \ln t$ is non-positive for all $t > 0$ therefore,we have

$$-\frac{x}{1-x} - \ln(1-x) = h(-\frac{1}{1-x}) \leq 0$$

Now,for k large enough we can assume that $c^*\epsilon_k < \frac{1}{2}$ and therefore

$$\ln(1 - c^*\epsilon_k) \geq -\frac{c^*\epsilon_k}{1 - c^*\epsilon_k} \geq 2c^*\epsilon_k$$

This relation and (2.78) imply that for sufficiently large k we have

$$\ln m_k^* \geq \ln(1 - c^*\epsilon_k) \geq -2c^*\epsilon_k \geq -2c\epsilon_k$$

We can now deduce that

$$0 < \psi(B_{k+1}^*) \leq \psi(B_k^*) + 3c\epsilon_k + \ln\cos^2\theta_k^* + [1 - \frac{q_k^*}{\cos^2\theta_k^*} + \ln\frac{q_k^*}{\cos^2\theta_k^*}] \tag{2.79}$$

By summing (2.79) and making use of (2.68) we have that

$$\sum_{j=0}^{\infty}(\ln\frac{1}{\cos^2\theta_j^{*2}} - [1 - \frac{q_j^*}{\cos^2\theta_j^*} + \ln\frac{q_j^*}{\cos^2\theta_j^*}]) \leq \psi(B_0^*) + 3c\sum_{j}^{\infty}\epsilon_j < +\infty$$

Since the term in the square bracket is non-positive ,and since $\ln(\frac{1}{\cos^2\theta_j}) \geq 0$ for all j, we obtain the limits $\lim_{j\to\infty}\ln(\frac{1}{\cos^2\theta_j^*}) = 0$ and $\lim(1 - \frac{q_j^*}{\cos^2\theta_j^*} + \ln\frac{q_j^*}{\cos^2\theta_j^*}) = 0$ which imply that

$$\lim_{j\to\infty}\cos\theta_j^* = 1 \quad \textit{and} \quad \lim_{j\to\infty} q_j^* = 1 \tag{2.80}$$

make use of (2.68)

$$\begin{aligned}\frac{\|G_*^{-\frac{1}{2}}(B_k - G_*)s_k\|^2}{\|G_*^{\frac{1}{2}}s_k\|^2} &= \frac{\|(B_k^* - I)s_k^*\|^2}{\|s_k\|^2} \\ &= \frac{\|B_k^*s_k^*\|^2 - 2s_k^{*T}B_k^*s_k^* + s_k^{*T}s_k^*}{s_k^{*T}s_k^*} \\ &= \frac{q_k^{*2}}{\cos^2\theta_k^*} - 2q_k^* + 1\end{aligned}$$

Since by (2.80), the right hand side converges to zero, we conclude that

$$\lim_{k\to\infty} \frac{\|(B_k - G_*)s_k\|}{\|s_k\|} = 0$$

Hence the rate of convergence of BFGS method is super linear. □

2.6.2 Super Linear Convergence of DFP Method

In this subsection we will discuss the super linear convergence of DFP method. For the result we follow we need to make a precise assumptions and we first state the following three lemmas.

Assumption 2.3. *a) $f : \Re^n \to \Re$ is twice continuously differentiable on an open convex set $D \subset \Re^n$.*

b) There is a strong local minimizer $x^* \in D$ with $\nabla^2 f(x^*)$ symmetric and positive definite.

c) There is a neighborhood $N(x^*, \epsilon)$ of x^* such that

$$\|\nabla^2 f(\overline{x}) - \nabla^2 f)(x)\| \leq \gamma\|\overline{x} - x\|, \quad \forall x,\ \overline{x} \in N(x^*, \epsilon) \ \text{ and } \ \gamma \ \text{ is constant}$$

Lemma 2.5. *Let $M \in \Re^{n\times n}$ be a nonsingular symmetric matrix. If for $\beta \in [0, \frac{1}{3}]$, the inequality*

$$\|My_k - M^{-1}s_k\| \leq \beta\|M^{-1}s_k\| \tag{2.81}$$

holds, then for any non-zero matrix $E \in \Re^{n\times n}$, we have

$$a) \qquad (1-\beta)\|M^{-1}s_k\|^2 \leq y_k^T s_k \leq (1+\beta)\|M^{-1}s_k\|^2 \tag{2.82}$$

$$b) \qquad \|E[I - \frac{(M^{-1}s_k)(M^{-1}s_k)^T}{y_k^T s_k}]\|_F \leq \sqrt{1-\alpha\theta^2}\|E\|_F \tag{2.83}$$

$$c) \quad \|E[I - \frac{(M^{-1}s_k)(M^{-1}s_k)^T}{y_k^T s_k}]\|_F \leq [\sqrt{1-\alpha\theta^2} + (1-\beta)^{-1}\frac{\|My_k - M^{-1}y_k\|}{\|M^{-1}s_k\|}]\|E\|_F \tag{2.84}$$

Proof. [*see* [2]] □

Lemma 2.6. *Let $\{\phi_{k=1}\}$ and δ_k be sequence of nonnegative numbers satisfying*

$$\phi_{k+1} \leq (1+\delta_k)\phi_k + \delta_k \quad \text{ and } \tag{2.85}$$

$$\sum_{k=1}^{\infty} \delta_k < +\infty \ \text{ then } \ \{\phi_k\} \text{converges.}$$

Proof. [*see* [2]] □

Lemma 2.7. *Under assumptions (2.3),there exist positive constants β_1, β_2, and β_3 such that $\forall x_k, x_{k+1} \in N(x^{x*,\epsilon})$, we have*

$$\|B_{k+1} - \nabla^2 f(x^*)\|_{DFP} \leq [\sqrt{1 - \beta_1\theta_k^2} + \beta_2\delta(x_k, x_{k+1})]\|B_k - \nabla^2 f(x^*)\|_{DFP} + \beta_3\delta(x_k, x_{k+1}) \tag{2.86}$$

$$\textit{where} \qquad \delta(x_k, x_{k+1}) = max\|x_k - x^*\|, \|x_{k+1} - x^*\|$$

$$\theta_k = \frac{\|(\nabla^2 f(x^*))^{-\frac{1}{2}}[B_k - (\nabla^2 f(x^*))^{\frac{1}{2}}]s_k\|}{\|B_k - \nabla^2 f(x^*\|_{DFP}\|(\nabla^2 f(x^*)^{\frac{1}{2}}\|}$$

Proof. [*see* [2]] □

Using the above three Lemmas, we can establish the following super linear convergence theorem of DFP method.

Theorem 2.10. *Suppose the sequence $\{x_k\}$ converges to x^*. Under the assumptions (2.3), DFP method is convergent super-linearly.*

Proof. Write $A = \nabla^2 f(x^*)$. Since $(1 - \beta_1\theta_k^2)^{\frac{1}{2}} \leq 1 - \frac{\beta_1}{2}\theta_k^2$, then (2.85) can be written as

$$(\frac{\beta_1\theta_k^2)}{2})\|B_k - A\|_{DFP} \leq \|B_k - A\|_{DFP} - \|B_{k+1} - A\|_{DFP} + [\beta_2\|B_k - A\|_{DFP} + \beta_3]\delta(x_k, x_{k+1}).$$

Summing both sides yields

$$\frac{1}{2}\beta_1\sum_{k=1}^{\infty}\theta_k^2\|B_k - A\|_{DFP} \leq \|B_1 - A\|_{DFP} + \beta_2\sum_{k=1}^{\infty}\delta(x_k, x_{k+1})\|B_k - A\|_{DFP} + \beta_3\sum_{k=1}^{\infty}\delta(x_k, x_{k+1})$$

Since the sequence $\{x_k\}$ is convergent, then

$$\sum_{k=1}^{\infty}\delta(x_k, x_{k+1}) < \infty$$

.Also,since $\{\|B_k - A\|_{DFP}\}$ is bounded, then

$$\frac{1}{2}\beta_1\sum_{k=1}^{\infty}\theta_k^2\|B_k - A\|_{DFP} < \infty$$

. The limit, $\lim_{k\to\infty}\|B_k - A\|_{DFP}$ exists. Hence, if some subsequence of $\{\|B_k - A\|_{DFP}\}$ converges to zero,the whole sequence converges to zero. Therefore

$$\lim_{k\to\infty}\frac{\|(B_k - A)s_k\|}{\|s_k\|} = 0,$$

and the conclusion holds. Otherwise,if $\|B_k - A\|_{DFP} \geq \omega > 0$, $\forall k \geq k_0$,then $\theta_k \to 0$. Note that

$$\begin{aligned}\frac{\|(B_k - A)s_k\|}{\|s_k\|} &\leq \frac{\|A^{\frac{1}{2}}\|\|A^{-\frac{1}{2}}(B_k - A)s_k)}{\|A^{\frac{1}{2}}\|^{-1}\|A^{\frac{1}{2}}s_k\|} \\ &= \|A\|\|B_k - A\|_{DFP}\frac{\|A^{-\frac{1}{2}}(B_k - A)s_k)\|}{\|B_k - A\|_{DFP}\|A^{\frac{1}{2}s_k}\|} \\ &= \|A\|\|B_k - A\|_{DFP}\theta_k\end{aligned}$$

Then by using $\theta_k \to 0$, we immediately obtain

$$\lim_{k\to\infty} \frac{\|(B_k - A)s_k\|}{\|s_k\|} = 0$$

□

Chapter 3

Numerical Implementation

In this chapter, there are seven test problems considered in table (3.1) below to analyse the performance of the quasi-Newton methods (BFGS and DFP updates) and to compare them with Newton's method and the steepest descent method. In this work, I have considered both convex and non-convex functions to compare the methods and the dimension of the test problems I have used in this study ranges between 2 and 4. And for most of the test problems, two initial points are used, starting from a point that is closer to the solution point and moving to the one that is furthest from it, and with initial inverse Hessian approximation $H_0 = eye(n)$, where $eye(n)$ is an identity matrix with order n and n is the dimension of the test problems. In doing so, it leads us to test the convergence properties and robustness of the methods.

To make comparisons among the algorithm 1.3 (gradient method), algorithm 1.4 (Newton's method), algorithm 2.3 (BFGS) and algorithm 2.4 (DFP) based on the number of iterations, and cosurety of their output to the exact minimizer of the objective function and to make conclusions when the algorithm performs well and when it may not work, for the Wolfe condition, I have used $c_2 = 0.2$ for all test problems, and the initial step length δ may vary from problem to problem but the same for the algorithms corresponding to the same problem (if the method needs to satisfy wolfe conditions) and also I have used the same initial point to compute the same problem by them. In my implementation, the programs are all written in MATLAB code, the stopping criteria that I have used in the four algorithms listed above is $\|\nabla f(x_k)\| \leq 10^{-4}$ and the result of the MATLAB code is approximated into four digits after decimal. The Euclidean norm is used in the convergence test to make these results comparable.

Table 3.1**:** The list of test problem functions with two initial point (initial guesses) and there suitable choice of corresponding initial step length ($t_0 = \delta$) used in this study.

Test problem	n	initial point x_0^T and initial step length δ
$f(x,y) = (x-y)^4 + (x-2)^2$	2	$(0,0), \delta = 0.2$ & $(100,100), \delta = 0.01$
$f(x,y) = x^4 + 2x^2y^2 + y^4$	2	$(-2,2), \delta = 0.01$ & $(1,1), \delta = 0.1$
$f(x,y) = x^2 + y^2 + 10(x+y-4)^2$	2	$(1,1), \delta = 0.01$ & $(1000,1000), \delta = 0.01$
$f(x,y,z) = (x-y)^4 + (x-12)^2 + z^2$	3	$(6,8,2), \delta = 0.1$ & $(100,100,10), \delta = 0.01$
$f(x,y,z) = (x-2)^4 + (x-2y)^2 + \cos(\frac{z}{2})$	3	$(0,3,\pi), \delta = 0.2$ & $(10,0,\frac{\pi}{2}), \delta = 0.1$
$f(x,y,z,u) = (x-y)^4 + (x+z)^2 + (2z+u-8)^2 + 4$	4	$(-1,-1,2,2), \delta = 0.02$
$f(x,y,z) = \exp^{-(x+y)} + x^4 + y^2 + (y+z-6)^2$	3	(0,0,0), $\delta = 0.3$ &$(0,3,10)$, $\delta = 0.2$

Now, first we will see the developed MATLAB codes corresponding to the four algorithms (steepest descent method, Newton method, quasi-Newton method with BFGS and DFP updates) performs well.
Consider the function

$$f(x, y, z) = \exp^{-(x+y)} + x^4 + y^2 + (y + z - 6)^2.$$

To compute the minimizer point and minimum point of f using the algorithms listed above, without loss of generality use initial point $x_0 = [0, 0, 0]^T$ with initial step length $\delta = 0.3$. Then we get the table 3.2,table 3.3, table 3.4 and table 3.5 corresponding to Newton, steepest descent, BFGS and DFP methods respectively.
Table 3.2**:** Output of MATLAB code for Newton's method.

iteration	x	y	z	$f(x, y, z)$	$\|\nabla f(x, y, z)\|$
1	0.0000	0.0000	0.0000	37.0000	17.7200
2	1.0000	0.0000	6.0000	37.0000	17.7200
3	0.7003	0.2019	5.7981	1.3679	3.6507
4	0.5447	0.2289	5.7711	0.6870	0.9682
5	0.4975	0.2392	5.7608	0.6018	0.1850
6	0.4934	0.2401	5.7599	0.5972	0.0138
7	0.4933	0.2401	5.7599	0.5971	0.0001

As we observe from table 3.2 Newton's method terminates at 7^{th} iteration, and gives an output $x^* = (0.4933, 0.2401, 5.7599)^T$ and $f(x^*) = 0.5971$. So, to compare with the other three methods (steepest descent, BFGS,and DFP), let us look the following three tabular values.

Table 3.3**:** Output of MATLAB code for Steepest descent method.

iteration	x	y	z	$f(x,y,z)$	$\|\nabla f(x,y,z)\|$
1	0.3000	3.9000	3.6000	17.4831	11.1949
2	0.2721	0.6645	2.7000	7.7849	6.8311
3	0.3655	1.9647	4.2813	4.0356	4.3531
4	0.3361	0.6675	4.1337	2.2620	2.7996
5	0.4005	1.0963	4.8530	1.4539	1.8702
6	0.3906	0.5361	4.8834	1.0435	1.2678
7	0.4378	0.6815	5.2317	0.8352	0.8802
8	0.4351	0.4226	5.2838	0.7248	0.6175
9	0.4635	0.4724	5.4599	0.6661	0.4388
10	0.4617	0.3472	5.5005	0.6345	0.3138
11	0.4772	0.3639	5.5919	0.6175	0.2260
12	0.4752	0.2391	5.6450	0.6111	0.3413
13	0.4933	0.3120	5.7145	0.6042	0.2387
14	0.4833	0.2430	5.6986	0.6007	0.1669
15	0.4929	0.2774	5.7337	0.5990	0.1172
16	0.4881	0.2432	5.7270	0.5981	0.0823
17	0.4929	0.2595	5.7449	0.5976	0.0580
18	0.4906	0.2425	5.7422	0.5974	0.0408
19	0.4930	0.2503	5.7514	0.5973	0.0288
20	0.4919	0.2418	5.7504	0.5972	0.0203
21	0.4931	0.2455	5.7551	0.5972	0.0144
22	0.4926	0.2412	5.7548	0.5972	0.0102
23	0.4932	0.2429	5.7572	0.5971	0.0072
24	0.4929	0.2408	5.7571	0.5971	0.0051
25	0.4932	0.2416	5.7584	0.5971	0.0036
26	0.4931	0.2405	5.7584	0.5971	0.0026
27	0.4933	0.2409	5.7590	0.5971	0.0018
28	0.4932	0.2404	5.7591	0.5971	0.0013
29	0.4933	0.2405	5.7594	0.5971	0.0009
30	0.4933	0.2403	5.7594	0.5971	0.0007
31	0.4933	0.2403	5.7596	0.5971	0.0005
32	0.4933	0.2402	5.7596	0.5971	0.0003
33	0.4933	0.2402	5.7597	0.5971	0.0002
34	0.4933	0.2402	5.7597	0.5971	0.0002
35	0.4933	0.2402	5.7598	0.5971	0.0001
36	0.4933	0.2402	5.7598	0.5971	0.0001

From table 3.3 we observe that the steepest descent method terminates at 36 iterations and gives an output of $x^* = (0.4933, 0.2402, 5.7598)^T$ and $f(x^*) = 0.5971$. The outputs that we obtained in both table 3.2 and table 3.3 are almost similar, but the steepest descent method required more iterations. This indicates it has slow rate compare to Newton's method. Next we will see the Outputs of the MATLAB codes for Quasi-Newton method with BFGS and DFP updates in table 3.4 and table 3.5 respectively.

Table 3.4**:** Output of MATLAB code for Quasi-Newton method with BFGS update.

iteration	x	y	z	$f(x,y,z)$	$\|\nabla f(x,y,z)\|$
1	0.3000	3.9000	3.6000	17.4831	11.1949
2.	0.4359	1.9874	4.5072	4.3193	4.9808
3	0.2749	0.8543	5.3205	1.0894	1.7862
4	0.5137	0.4446	5.6107	0.6539	0.6460
5	0.4218	0.2883	5.7310	0.6067	0.2310
6	0.4852	0.2425	5.7602	0.5973	0.0277
7	0.4908	0.2410	5.7602	0.5972	0.0091
8	0.4923	0.2405	5.7600	0.5971	0.0036
9	0.4929	0.2403	5.7599	0.5971	0.0014
10	0.4932	0.2402	5.7599	0.5971	0.0006
11	0.4933	0.2401	5.7599	0.5971	0.0002
12	0.4933	0.2401	5.7599	0.5971	0.0001

Table 3.5**:** Output of MATLAB code for Quasi-Newton method with DFP update.

iteration	x	y	z	$f(x,y,z)$	$\|\nabla f(x,y,z)\|$
1	0.3000	3.9000	3.6000	17.4831	11.1949
2	0.4293	2.0231	4.4512	4.4378	5.0047
3	0.2811	0.8765	5.2758	1.1120	1.7843
4	0.5012	0.4558	5.5906	0.6571	0.6386
5	0.4365	0.3028	5.7145	0.6057	0.2206
6	0.4887	0.2532	5.7509	0.5974	0.0406
7	0.4917	0.2452	5.7566	0.5972	0.0159
8	0.4927	0.2421	5.7586	0.5971	0.0064
9	0.4931	0.2409	5.7594	0.5971	0.0026
10	0.4932	0.2404	5.7597	0.5971	0.0010
11	0.4933	0.2403	5.7598	0.5971	0.0004
12	0.4933	0.2402	5.7598	0.5971	0.0002
13	0.4933	0.2401	5.7599	0.5971	0.0001

In both table 3.4 and table 3.5, we get the outputs $x^* = (0.4933, 0.2401, 5.7599)^T$ and $f(x*) = 0.5971$. But DFP method took one more iteration than BFGS method. As we observe from the table 3.2 upto table 3.5, Newton's method computes the minimizer point of the given test problem with less iterations compared to steepest descent method and quasi-Newton methods (BFGS and DFP updates), and the steepest descent method took more iterations compared to Newton's method and quasi-Newton methods.
next, we will see a counter example that Newton's method fails, but the other three methods (BFGS, DFP and the steepest descent methods) performs well.
Now consider the function

$$f(x,y,z) = (x-2)^4 + (x-2y)^2 + \cos(\frac{z}{2}).$$

The minimum point of this function is obtained when $(x-2)^4 = 0$, $(x-2y)^2 = 0$ and $\cos(\frac{z}{2}) = -1$, (i.e, $x^* = (2, 1, 2\pi)^T$), and its minimum value is $f(x^*) = -1$. When we use

the initial point $x_0 = (10, 0, \frac{\pi}{2})^T$, the Hessian matrix $\nabla^2 f(x_0)$ of f becomes

$$\nabla^2 f(x_0) = \begin{pmatrix} 770.0000 & -4.0000 & 0 \\ -4.0000 & 8.0000 & 0 \\ 0 & 0 & -0.1768 \end{pmatrix}$$

with eigne values 770.0210, 7.9790 and -0.1768. This indicates that the Hessian matrix of the objective function f given above is not always positive definite (it is indefinite), and hence f is not convex function.
Now, let us look the outputs for the non convex function f with initial point $x_0 = (10, 0, \frac{\pi}{2})^T$ displayed by the four methods listed above.
Table 3.6: Output of MATLAB code for Newton's method.

iteration	x	y	z	$f(x, y, z)$	$\|\nabla f(x, y, z)\|$
1	10.0000	0.0000	1.5708	4196.7071	2068.3868
2	7.3333	3.6667	-0.4292	4196.7071	2068.3868
3	5.5556	2.7778	0.0067	810.0635	606.8148
4	4.3704	2.1852	-0.0000	160.8195	179.7970
5	3.5802	1.7901	0.0000	32.5693	53.2732
6	3.0535	1.5267	0.0000	7.2359	15.7846
7	2.7023	1.3512	0.0000	2.2318	4.6769
8	2.4682	1.2341	0.0000	1.2433	1.3858
9	2.3121	1.1561	0.0000	1.0481	0.4106
10	2.2081	1.1040	0.0000	1.0095	0.1217
11	2.1387	1.0694	0.0000	1.0019	0.0360
12	2.0925	1.0462	0.0000	1.0004	0.0107
13	2.0617	1.0308	0.0000	1.0001	0.0032
14	2.0411	1.0206	0.0000	1.0000	0.0009
15	2.0274	1.0137	0.0000	1.0000	0.0003
16	2.0183	1.0091	0.0000	1.0000	0.0001

From table 3.6 above, the Newton's method gives us an output, minimizer point $x^* = (2.0183, 1.0091, 0.0000)^T$ and the minimum point $f(x^*) = 1$. But this contradicts to the minimizer point $x^* = (2, 1, 2\pi)^T$) and minimum value $f(x^*) = -1$. Based on this test problem we can conclude that Newton's method fails if the Hessian matrix of the objective function does not preserve positive definiteness every where.
For this test problem, from the starting point $x_0 = (10, 0, \frac{\pi}{2})^T$ and initial step length $\delta = 0.1$, the steepest descent method required 208 iterations, whereas quasi-Newton methods with BFGS and DFP updates took only 53 and 65 iterations, respectively to reduce the gradient norm to the termination tolerance 10^{-4}. Below, I have report the first five and the last two iterations of the steepest descent , BFGS and DFP methods.

Table 3.7**:** Output of MATLAB code for steepest descent method.

iteration	x	y	z	$f(x,y,z)$	$\|\nabla f(x,y,z)\|$
1	6.4000	4.0000	1.6062	378.0641	8.5117
2	5.2800	2.7200	1.6781	116.4369	6.2838
3	4.0320	2.5920	1.7525	19.0160	4.9476
4	3.6800	1.6704	1.8293	8.6911	4.2786
5	2.8723	1.9418	1.9086	2.1797	4.0748
207	2.0000	1.0000	6.2829	-1.0000	0.0002
208	2.0000	1.0000	6.2829	-1.0000	0.0001

From table 3.7, we get a minimizer solution $x^* = (2.0000, 1.0000, 6.2829)^T$ and $f(x^*) = -1.0000$.

Table 3.8**:** Output of MATLAB code for Quasi-Newton method with BFGS update.

iteration	x	y	z	$f(x,y,z)$	$\|\nabla f(x,y,z)\|$
1	6.4000	4.0000	1.6062	378.0641	8.5117
2	5.0103	3.0661	1.6796	84.0433	5.8773
3	4.4060	2.6514	1.7919	34.9391	4.7045
4	3.9159	2.3152	1.9765	14.5349	3.7570
5	3.5125	2.0384	2.2362	5.9896	2.9821
52	1.9999	1.0000	6.2831	-1.0000	0.0001
53	2.0000	1.0000	6.2831	-1.0000	0.0001

As we observed from Table 3.8, the quasi-Newton method with BFGS update gives minimizer solution $x^* = (2.0000, 1.0000, 6.2831)^T$ and $f(x^*) = -1.0000$.

Table 3.9**:** Output of MATLAB code for quasi-Newton method with DFP update.

iteration	x	y	z	$f(x,y,z)$	$\|\nabla f(x,y,z)\|$
1	6.4000	4.0000	1.6062	378.0641	8.5117
2	5.0429	3.0875	1.6779	87.6785	5.9370
3	4.4320	2.6685	1.7903	36.4290	4.7523
4	3.9368	2.3289	1.9738	15.1436	3.7953
5	3.5297	2.0497	2.2288	6.2406	3.0134
64	2.0001	1.0000	6.2833	-1.0000	0.0001
65	2.0000	1.0000	6.2832	-1.0000	0.0001

From table 3.9 above, we get $x^* = (2.0000, 1.0000, 6.2832)^T$ and $f(x^*) = -1$

Recall that the exact minimizer of the give test problem is $x^* = (2, 1, 2\pi)^T$. Now, let us apply the Euclidean norm to make the results we obtained in table 3.7, table 3.8 and table 3.9 comparable.

From table 3.7, we get, $x^* = (2.0000, 1.0000, 6.2829)^T$, then $\|(2, 1, 2\pi)^T - (2.0000, 1.0000, 6.2829)^T\| = 0.0003$.

From table 3.8, we get, $x^* = (2.0000, 1.0000, 6.2831)^T$, then $\|(2, 1, 2\pi)^T - (2.0000, 1.0000, 6.2831)^T\| = 0.0001$.

Also, from table 3.9, we get, $x^* = (2.0000, 1.0000, 6.2832)^T$, then $\|(2, 1, 2\pi)^T - (2.0000, 1.0000, 6.2832)^T\|$ 0.0000. This indicates that the solution we obtained by DFP method is close to the exact minimizer, and BFGS method performs well next to DFP, but better than the steepest

descent method according to this test problem. Although, when we compare them based on the number of iterations they took to achieve the termination criteria, the steepest descent method required more iterations than the BFGS and DFP methods, and DFP method took more iteration than BFGS method. In addition to this, these methods need further discussion to arrive at better conclusion.
Consider the function

$$f(x, y) = (x - y)^4 + (x - 2)^2.$$

It has a unique minimizer point $x^* = (2, 2)^T$ and minimum point $f(2, 2) = 0$. From the starting point $x_0 = (0, 0)^T$, we get, the Hessian matrix $\nabla^2 f(x_0)$ of f

$$\nabla^2 f(x_0) = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$$

with eigne values 2 and 0. It is not positive definite. When we apply Newton's method to minimize this function starting from initial guess $x_0 = (0, 0)^T$, the MATLAB code for Newton's method corresponding to this function displays **NaN** under its columns. (i.e, NaN is an arithmetic representation for Not-a-Number and it is obtained as a result of mathematically undefined operations). Also it displays warning that is, "matrix is singular to working precision". From this, we conclude that Newton's method may fail when the Hessian matrix of the objective function is not positive definite and it close to singular matrix. But the steepest descent method, BFGS method and DFP method computes the minimizer point and minimum point of this function. From the starting point $x_0 = (0, 0)^T$ and $\delta = 0.2$, the steepest descent method required 377 iterations, whereas BFGS and DFP methods took only 28 and 30 iterations, respectively to reduce the gradient norm to the termination tolerance 10^{-4}. The minimizer solution obtained by the steepest descent, BFGS and DFP methods are $x^* = (1.9999, 1.9707)^T$, $x^* = (2.0000, 1.9755)^T$ and $x^* = (2.0000, 1.9774)^T$ respectively. Here also the solution we obtained by DFP method is close to the exact minimizer of f than the solutions we obtained by BFGS and steepest descent methods. And the minimizer solution we obtained by Steepest descent method is far from the exact minimizer compare to the solution obtained by BFGS method. Based on the examples considered above quasi-Newton methods (BFGS and DFP update) is superior than the steepest descent method.

Now, an additional comparison is needed to the quasi-Newton methods (BFGS and DFP updates) to make better conclusion based on the test problems considered on table 3.1. Let us denote the functions (test problems) in row two, row three, row four, row five, row six, row seven and row eight on table 3.1 by f_2, f_3, f_4, f_5, f_6, f_7 and f_8 respectively. In table 3.10 below the values under the columns 4 and 6 are the number of iterations for BFGS and DFP methods, respectively required to reduce the gradient norm to the termination tolerance 10^{-4} and the values under columns 5 and 7 are the minimizer solutions we obtained by BFGS and DFP methods, respectively corresponding to the test problem with the given initial guess x_0 and initial step length δ. Also, on table 3.10 TP stands for test problem.

Table 3.10**:** Numerical results for BFGS and DFP methods.

TP	x_0^T	δ	I_{BFGS}	x_{BFGS}^{*T}	I_{DFP}	x_{DFP}^{*T}
f_2	$(100, 100)$	0.01	771	$(2.0000, 2.0260)$	775	$(2.0000, 2.0261)$
f_3	$(1, 1)$	0.1	38	$(0.0194, 0.0194)$	38	$(0.0194, 0.0194)$
f_4	$(1, 1)$	0.01	564	$(1.9048, 1.9048)$	564	$(1.9048, 1.9048)$
f_5	$(6, 8, 2)$	0.01	588	$(12.0001, 12.0269, 0.0000)$	808	$(12.0000, 11.9883, 0.0000)$
f_5	$(100, 100, 10)$	0.01	758	$(12.0000, 12.0260, 0.0000)$	761	$(12.0000, 12.0263, 0.0000)$
f_6	$(10, 0, \frac{3}{2}\pi)$	0.4	13	$(2.0000, 1.0000, 6.2832)$	16	$(2.0000, 1.0000, 6.2832)$
f_6	$(0, 3, \pi)$	0.2	29	$(2.0000, 1.0000, 6.2832)$	32	$(2.0000, 1.0000, 6.2832)$
f_7	$(-1, -1, 2, 2)$	0.02	297	α	352	β
f_8	$(0, 3, 10)$	0.2	19	$(0.4934, 0.2401, 5.7599)$	22	$((0.4933, 0.2401, 5.7599))$

where $\alpha = (-2.2873, -2.2765, 2.2873, 3.4255)$ and $\beta = (-2.2860, -2.2842, 2.2860, 3.4281)$. As we observe the outputs on table 3.10, the minimizer solutions obtained by both BFGS and DFP methods are almost similar for all the test problems with their corresponding initial guess and initial step length considered on table 3.1. But, when we compare them based on the number of iterations they took to reduce the gradient norm to 10^{-4}, the DFP method required more iterations than the BFGS method. So, based on the test problems considered on this work, we can conclude that the quasi-Newton method with BFGS upadte has better performance than the quasi-Newton method with DFP update.

Conclusion

From this work, it is safe to say that the quasi-Newton methods (BFGS and DFP updates) and the steepest descent method avoid the drawbacks of Newton's method in solving unconstrained optimization problems. The numerical results for the non convex test problems we discussed in chapter three show that quasi-Newton methods has better performance than the Newton's method. From those results, the steepest descent method also mitigate the drawbacks of the Newton's method, but it has slow rate compare to the BFGS and DFP methods. Also, the numerical results show that the BFGS method is faster than the DFP method.

In general, to mitigate the drawbacks of Newton's method in solving unconstrained optimization problems, it is better to use quasi-Newton methods with BFGS update rather than the steepest descent method and quasi-Newton methods with DFP update.

Annex A

```
function BFGS        % used to read functions from the bottom.
     % Quasi-Newton method with BFGS updating formula MATLAB code
     % based on line search method
     % step length identification by inexact line search
k=0;
x_0=input['enter the initial column vector, x_0=']';
n=input['enter dimension of the objective function, n='];
tol=input['give termination tolerance, tol='];
pk=grad(x_0(1), x_0(2),x_0(3), ...,x_0(n));  % gradient at x_k
Hk=eye(n);
while norm(pk)>tol
% compute the search direction
          d=-Hk*pk;
% Evaluating the step length t_k by inexact line search.
 fun=@(t)f(x_0(1) + t*d(1), x_0(2) + t*d(2),x_0(3) + t*d(3),...,x_0(n) + t*d(n));
kk=input['Choose a suitable initial step length such that f(x_k+kk*d<f(x_k)), kk='];
    rr=fun(kk);
      if rr<f(x_0(1),x_0(2),x_0(3),...,x_0(n))
            delta=kk;
      else
            delta=[];       % helps us to retry the initial step length.
      end
           t=delta;
c_2=input['enter Wolfe constant c_2 in between 0.1 and less than 1, c_2='];
        x_new=x_0 + t*d;
ee=grad(x_new(1), x_new(2),x_new(3),...,x_new(n));     % gradient of f(x_(k+1)).
bb=grad(x_0(1), x_0(2),x_0(3),...,x_0(n));             % gradient of f(x_k).
    if ee'*d<c*bb'*d
             t=t+delta;    % increment by delta
    else
            t=t;
end (%end if)
% end of the line search
         x_new=x_0+t*d;         % the new iterate point
         sk=x_new-x_0;
         yk=ee-bb;              % gradient at x(k+1)-gradient at x(k)
% beginning  of BFGS update
              T=yk'*sk;
if T>0
            R=1/T;
            A=eye(n)-R*sk*yk';
            P=R*sk*sk';
            Hk=A*Hk*A'+P;            %the new updated matrix
     else
           Hk=Hk;
```

```
     end (% end if)
% end of BFGS update
        x_0=x_0+t*d;
        pk=grad(x_new(1), x_new(2),x_new(3),...,x_new(n);
         k=k+1;
  end  (% end while)
\%\%\%\%\%\%\%\%\%\%\%\%\\%\%\%\%\%\%\%\\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\
function w=f(x_1,x_2,x_3,...,x_n)      % w is the objective function to be minimize
         w=input['write the objective function here, w='];
function g=grad(x_1,x_2,x_3,...,x_n)   % gradient of f
        g=input['write the gradient of the objective function here, g='];
```

Annex B

```
function SDM        % used to read functions from the bottom.
      % MATLAB code for Steepest descent method
      % based on line search method
      % step length identification by inexact line search
   k=0;
   x_0=input['enter the initial column vector, x_0=']';
   n=input['enter dimension of the objective function, n='];
   tol=input['give termination tolerance, tol='];
   pk=grad(x_0(1), x_0(2),x_0(3), ...,x_0(n));    % gradient at x_k
while norm(pk)>tol
   % compute the search direction
          d=-pk;
  % Evaluating the step length t_k by inexact line search.
 fun=@(t)f(x_0(1) + t*d(1), x_0(2) + t*d(2),x_0(3) + t*d(3),...,x_0(n) + t*d(n));
kk=input['Choose a suitable initial step length such that f(x_k+kk*d<f(x_k)), kk='];
        rr=fun(kk);
   if rr<f(x_0(1),x_0(2),x_0(3),...,x_0(n))
            delta=kk;
   else
            delta=[];       % helps us to retry the initial step length.
      end
           t=delta;
c_2=input['enter Wolfe constant c_2 in between 0.1 and less than 1, c_2='];
        x_new=x_0 + t*d;
ee=grad(x_new(1), x_new(2),x_new(3),...,x_new(n));     % gradient of f(x_(k+1)).
bb=grad(x_0(1), x_0(2),x_0(3),...,x_0(n));             % gradient of f(x_k).
    if ee'*d<c*bb'*d
             t=t+delta;    % increment by delta
    else
            t=t;
end (%end if)
% end of the line search
         x_new=x_0+t*d;         % the new iterate point
        pk=grad(x_new(1), x_new(2),x_new(3),...,x_new(n);
        x_0=x_new;
         k=k+1;
  end  (% end while)
\%\%\%\%\%\%\%\%\%\%\%\%\\%\%\%\%\%\%\%\%\\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\
function w=f(x_1,x_2,x_3,...,x_n)      % the objective function to be minimize
         w=input['write the objective function here, w='];
function g=grad(x_1,x_2,x_3,...,x_n)   % gradient of f
        g=input['write the gradient of the objective function here, g='];
```

Annex C

```
 function NEWTONMETHOD    % used to read functions from the botom
                          % MATLAB code for Newton's Method
     k=0;
     x=input('Enter the column vector, x=');
     n=input['dimension of the objective function, n='];
    tol=input['give the termination tolerance, tol='];
    obj=func(x);        % obj is the objective function to be minimize.
     g=grad(x);         % g is gradient of the objective function
     H=hessian(x)       % H is the hessian matrix of the objective function

while norm(g)>tol
      obj=func(x);
      g=grad(x);
      H=hessian(x);
% Compute the Newton direction
      d=-inv(H)*g;
      x=x+d;            % the new iterate point.
       k=k+1;
end
\%\%\%\%\%\%\%\%\\%\%\%\%\%\%\%\%\\%\%\%\%\%\%\%\%\\%\%\%\%\%\%\%\%
 function w=func(x)           % The objective function to be minimize.
          w=input['write the objective function here,w='];
 function H=hessian(x)        % Hessian matrix of the objective function.
          H=input['write the Hessian matrix here, H='];
```

Reference

[1]: Edwin K.P.Chang and Stanislaw H.$\dot{Z}$ak "An Introduction to Optimization", Copyright ©2001 by John Wiley and Sons, Inc. Canada, Second Edition.

[2]: Wenyu Sun and Ya-Xiang Yuan "Optimization Theory and Methods" Non-Linear Programming, Nanjing Normal University, Nanjing, China and Chince Academy of Science, Beijing, China.

[3]: Jorge Nocedal, Stephen J.Wright"Numerical Optimization" EECS Department, Northwestern University, Evalnston, IL 60208-3118 USA nocedal@eecs. Madison, WI 53706-1613 USA, swright@cs.wise.edu, Second Edition.

[4]: Yang Ding Enkeleida Lushi Qingguo Li "Investigation of Quasi-Newton Methods for Unconstrained Optimization" Simon fraser university, 8888 university drive, Burnaby, B.C., V5A 1S6, Canada.