Int J Edu Sci, 17(1-3): 116-128 (2017)
DOI: 10.1080/09751122.2017.1323496

A Comparison of Newton's Method and Two of its Modifications for the Solution of Nonlinear Optimization Problems

Taiwo Lukumon Abiodun[1] and Olusesan Adeyemi Adelabu[2*]

[1]Faculty of Physical Sciences, University of Ilorin, P.M.B.1515 Ilorin, Nigeria
[2]Faculty of Science and Agriculture, University of Fort Hare, Alice, P.B. X1314, Eastern Cape, 5700, South Africa
E-mail:[1]<taiwolukman@yahoo.co.uk>, [2]<aolusesan@ufh.ac.za>*

KEYWORDS Newton's Method. Optimization Problems. Quasi-Newton's Method. Test Functions

ABSTRACT This paper presents two iterative modifications of the Newton's method for solving unconstrained optimization problems. Each of the two methods requires an update formula which replaces an inverse matrix and maintains positive definiteness property. The methods are based on the recurrence of matrix factorizations. The paper also show the behaviour and performance of the methods at each iteration. Numerical results are presented to compare the performance of the Newton's methods and the two modifications.

INTRODUCTION

Optimization is essentially the art, science and mathematics of choosing the best among a given set of finite or infinite alternatives. Also, optimization is an interdisciplinary subject cutting through the boundaries of mathematics, economics, engineering, natural science and many other fields of human endeavor in which decisions are made at several stages. The ultimate goal of all such decisions is either to minimize the effort required and inconveniences or to maximize the desired benefit. Since the effort required or the benefit desired in any practical situation can be expressed as a function of certain decision variables, optimization can be defined as the process of finding the conditions that gives the maximum or minimum value of a function. If a point x corresponds to the minimum value of function $f(x)$, the same point also corresponds to the maximum value of the negative of the function, $-f(x)$.

Optimization can be taken to mean minimization since the maximum of a function can be found by seeking the minimum of the negative of the same function. According to Gill and Murray (1974), several methods have been developed for solving different types of optimization problems, but the Newton's and Quasi-Newton's methods will be considered in this work. The Newton's and Quasi-Newton's methods have the following scheme of iteration (Gill and Murray 1974)

$$x_{k+1} = x_k - [H(x_k)]^{-1}\nabla f(x_k)$$

and

$$x_{k+1} = x_k - \alpha_k [B(x_k)]\nabla f(x_k)$$

where $H(x)$ is the Hessian matrix, $B(x)$ is the update matrix and a_k is the optimal step-length. Major contributions in this area have been made by Davidon, Fletcher, Goldfarb, Broyden, Powell and Shanno. The methods are suitable for large scale problems because the amount of storage required by the algorithm can be controlled by the user (Jiang and Yan 2010).

However, this study investigates the behaviour and performance of the methods with numerical results.

Optimization Problems

An optimization problem can be a maximization or a minimization problem stated as (Li and Fukushima 2001):
find $x = (x_1, x_2, \cdots, x_n)$, which minimizes

$$f = f(x)$$

[*]Address for correspondence:
Olusesan Adeyemi Adelabu
Faculty of Science and Agriculture,
University of Fort Hare, Alice,
P.B. X1314, Eastern Cape, 5700,
South Africa
Cell: +27736521664,
*E-mail:*aolusesan@ufh.ac.za

subject to the constraints

where the variable is an n-dimensional vector called the decision vector or variable, $f(x)$ is the objective function, $g_i(x)$ and $h_j(x)$ are respectively refers to as the equality and inequality constraints.

Design Variables

Any system is described by a set of quantities some of which are viewed as variables during the design process and some of which are preassigned parameter. All quantities that can be treated as variables are called design or decision variables x_i, $i=1,2,...,n$. They are collectively represented as a design vector

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

Design Constraint

In practice, the design variable cannot be selected arbitrarily but have to satisfy certain requirements. These restriction that must be satisfied are called design constraints (Yuan and Byrd 1995). Design constraints may represent limitations on the performance or behaviour of the system or physical limitations (Yuan and Byrd 1995). For example, in an optimization problem with only inequality constraint $g_i(x) \leq 0$, the set of values of x that satisfy the equation $g_i(x) = 0$, form a hyper surface in the design space which is called constraint surface.

Objective Function

The classical design procedure aims at finding an acceptable design, a design which satisfies the contraints. In general, there are several acceptable designs and the purpose of the optimization is to single out the best possible design. Thus, a criterion has to be selected for comparing different designs. When the criterion is expressed as a function of the design variable it is known as objective function. The objective function is in general specified by physical or economical consideration. However, the selection of an objective function is not trivial because what is the optimal with respect to a certain criterion may be unacceptable with respect to another criterion. Typically, there is a trade off performance cost or performance-reliability; hence the selection of the objective function is one of the most important decisions in the whole design process.

Classification of Optimization Problem

Optimization problem can be classified in several ways.

Classification Based on the Existence of Constraints

Optimization problem can be classified as a constrained or as an unconstrained one depending on whether it involves constraints or not (Yuan and Byrd 1995).

Classification Based on the Nature of Design Variables

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}, \quad g_j(x) \leq 0, \quad j = 1,2,\ldots,m$$

Based on the nature of design variables encountered, optimization problems can be classified into two broad categories. First, the problem is to find values to a set of design parameters that makes some prescribed function of these parameters minimum subject to a certain constraints. For example, the problem of minimum-weight design of a prismatic beam subject to a limitation on the maximum deflection is stated as:

Find

which minimizes

$f(X) = plbd$

subject to the constraints

$\delta_{tip} \leq \delta_{max}$

$b \geq 0$

$d \geq 0$

where p is the density and δ_{tip} is the tip deflection of the beam. This problem is called parameter or static optimization problems. In the second category, the objective is to find a set of design parameters, which are all continuous functions of some other parameter that minimizes an objective function subject to a set of constraints. If the cross sectional dimensions of the rectan-

gular beam are allowed to vary along its length. The optimization problem is stated as:

Find

$$X(t) = \begin{pmatrix} b(t) \\ d(t) \end{pmatrix}$$

which minimize

$$F[X(t)] = \rho \int_0^1 b(t)d(t)dt$$

$$\delta_{tip} \le \delta_{max}, 0 \le t \le l$$

$$b(t) \ge, 0 \le t \le l$$

$$d(t) \ge, 0 \le t \le l$$

Here, the design variables are functions of the length parameter *t.* These types of problem, where each design variable is a function of one or more parameters, is known as a trajectory or dynamics optimization problem.

Classification Based on the Physical Structure of the Problem

This can be classified as optimal control and non-optimal control problems

Optimal Control Problem

An optimal control (OC) problem is a mathematical programming problem involving a number of stages, where each stage evolves from the preceding stage in a prescribed manner. It is usually described by two types of variables: the control (design) and the state variables. The control variables defines the system governing the evolution of the system from one stage to the next and the state variables describe the behavior or status of the system in any stage. The problem is to find a set of control or design variables such that the total objective function (also known as the performance index, PI) that is, overall the stages is minimized subject to a set of constraints on the control and state variables.

The problem can be stated as: find X which minimizes

$$f(x) = \sum_{i=1}^{l} f_i(x_i, y_i)...$$

subject to the constraints

$$h_k(y_k) \le 0, k = 1,2,...,l$$

where x_i is the *i* th control, y_i the *i* th state variable and f_i the contribution of the *i* th stage to the total objective function: g_j, h_k and q_i are function of x_j, y_k and x_i and y_i respectively, and *l* is the total number of stages.

Classification Based on Nature of the Equations

Optimization problem can be classified as linear, quadratic polynomial, non-linear depending upon the nature of the objective function and the constraints. This classification is important because computational methods are usually selected on the basis of such a classification. that is the nature of the function involved indicates the type of solution procedure.

Classification Based on the Permissible Values of the Design Variables

This depends on the values permitted for the design variables. It can be classified as integer or real valued and deterministic or stochastic. that is all the design variable are restricted to take on only integer (discrete) values.

Classification Based on the Separability of the Function

$g_{ij}(x_i) \le 0, j = 1,2,...,m, i = 1,2,...,l$

Optimization problems can be classified as separable and non-separation programming problems based on the separability of objective and constraint functions.

Separable Programming Problem

A function *f (x)* is said to be separable if it can be expressed as the sum of *n* single variable functions, $f_1(x_1), f_2(x_2),...., f_n(x_n)$

$$f(x) = \sum_{i=1}^{n} f_i(x_i)$$

the problem can be stated as find *X* which minimize

$$f(x) = \sum_{i=1}^{n} f_i(x_i)$$

subject to

$$g_j(x) = \sum_{i=1}^{n} g_{ij}(x_i) \le b_i, j = 1,2,...., m$$

where b_j is a constant.

Classification Based on the Number of Objective Function

Optimization problem can be classified as single and multi-objective programming problems.

Single Objective Programming Program can be stated as find X which minimize $f(x)$ subject to

$$g_i(x) \le 0, i = 1,2,....,m$$

Multi-objective Programming Program can be stated as find x which minimize $f_i(x)$, $(f_2(x),.....,$ $f_k(x)$ subject to

$$g_i(x) \le 0, i = 1,2,....,m$$

where $f_{i,}$ $f_2,.....,$ f_k denote the objective functions to be minimize simultaneously.

METHODOLOGY

Some existing methods of solving optimization problems include (Jiang and Yan 2010):

1. Analytical Method
2. Penalty Function Method
3. Simplex Method
4. Lagrange Multiplier Method
5. Bracketing Method
6. Fibonacci Method
7. Golden Search Method
8. Gradient Method
9. Newton Method
10. Quasi-Newton method

Aim

The purpose of this project work is to investigate the behavior and performances as well as compare and discuss the numerical results of Newton's method, Davidon Fletcher Powell and Broyden Fletcher Goldfarb Shanno Methods. Also, to compare the numerical computational results of Newton's methods and its modifications (Quasi-Newton's methods).

Newton's Methods for Optimization Problems

Newton's method sometimes called Newton-Raphson method is a root finding method which uses first and second derivatives (Jiang and Yan 2010). Given a starting point, the researchers construct a quadratic approximation to the objective function that matches the first and the second derivative value at that point. The researchers then minimize the approximate function instead of the original objective function. The researchers then use minimizer of the approximate function as the starting point in the next step and repeat the procedure iteratively.

One Dimensional Non-linear Optimization Problems

Suppose, the researchers are confronted with the problem of minimizing a function of a single real variable x. The researchers assume that at each measurement point x_k and can calculate $f(x_k)$, $f'(x_k)$ and $f''(x_k)$. The researchers consider the quadratic approximation of the function $f(x)$ at $x=x_k$ using the Taylor's Series Expansion (Martinez 2000).

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

The researchers set the derivative of equation (2.1) equal to zero for the minimum of $f(x)$, they obtain

$$f'(x) = f'(x_k) + f''(x_k)(x - x_k) = 0$$

If x_k represent an approximation to the minimum of $f(x)$, then (2.2) can be re-arranged to obtain an improved approximation as

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

Equation (2,3) is equivalent to using a quadratic approximation for the function $f(x)$ and applying the necessary conditions. Thus, the iterative process (2.3) can be assumed to have converged when the derivative $f'(x_{k+1})$ is close to zero.

where ε is a small quantity.

Algorithm for Newton Method

The Newton's Method consists of the following steps :

1. Select an initial point x_0 and small value epsilon set $k=0$
2. Compute $f'(x_k)$ and $f''(x_k)$
3. Calculate, $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$ Compute $f'(x_{k+1})$
4. If $|f'(x_{k+1})| \le \varepsilon$, terminate else,
5. Set $k = k+1;$ go to Step 2.

Remarks

1. Newton method requires both first and second order derivatives of $f(x)$.
2. If $f''(x_k) = 0$, the Newton iterative method has a powerful convergence property known as quadratic convergence.

3. If the starting point for the iterative process is not close to the true solution *x*, then the Newton iterative process might diverge.

Unconstrained Non-linear Multi-dimensional Optimization Problems

The researchers consider again the quadratic approximation of the function $f(x)$ at $x=x_k$. Using the tailor series expansion (Yuan and Byrd 1995).

where $H(x_k)$ is the matrix of second partial derivatives known as Hessian Matrix of f evaluated at the point x_k

$$H=\begin{bmatrix} \frac{\partial^2 f}{\partial x^2_1} & \frac{\partial^2 f}{\partial x_1 \delta x_2} & \frac{\partial^2 f}{\partial x_1 \delta x_n} \\ \frac{\partial^2 f}{\partial x_2 \delta x_1} & \frac{\partial^2 f}{\partial x_n \delta x_2} & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \frac{\partial^2 f}{\partial x^2_n} \end{bmatrix}$$

By setting the partial derivative of (2.5) equal to zero for the maximum of $f(x)$,

$$\frac{\partial f(x)}{\partial x_j} = 0$$

For all $j = 1,2,\ldots, n$ The researchers obtain

$$\nabla f = \nabla f(x_k) + H(x_k)(x^* - x_k) = 0$$

If $H(x_k)$ is non-singular, the researchers obtain an improve approximation $\chi = x_{k+1}$ from (2.7) as

$$\nabla f(x_k) = -H(x_k)(x^* - x_k)$$

$$x^* - x_k = -\frac{\nabla f(x_k)}{H(x_k}$$

$$x_{k+1} = x_k - H(x_k)^{-1}\nabla f(x_k),$$

since higher-order terms have been neglected from (2.5), the recursive formula (2.8) represent Newton's method.

Algorithm for Newton's Method

1. Select an initial point x_0 and $\varepsilon \leq 10^{-6}$.
2. Set $k=0$.
3. Compute $\nabla f(x_k)$ and $H(x_k)$ and $H(x_k)$, If $\|\nabla f(x_k)\| < \varepsilon$, terminate, else, go to step 4.
4. Update:
$$x_{k+1} = x_k - H(x_k)^{-1}\nabla f(x_k)$$
5. Set $K=K+1$, go to step 3.

Remarks

1. The method requires both first and second order derivative of $f(x)$.
2. It is sensitive at initial point.
3. Converges quadratically near the optimum.
4. If $f(x)$ is non-quadratic function, Newton's method may sometimes diverge and may converge to saddle point.
5. It requires storing of $\nabla f(x_k)$ and $H(x_k)$.
6. It requires the inversion of the matrix $H(x_k)$ at each step.
7. It also requires the evaluation of the quantity $H(x_k)^{-1} \nabla f(x_k)$ at each step.

In the next section, two of the modifications of the Newton's method for optimization are considered.

Modifications of Newton's Method

Modification of Newton's method considered in this work is the Quasi-Newton method (Jiang and Yan 2010).

$$f(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$$

Quasi-Newton Methods

This is a method of Modification of Newton's method (Jiang and Yan 2010). The basic idea behind this method is approximating the Hessian matrix $H(x_k)$ by another matrix $[A(x_k)]$ or $[H(x_k)]^{-1}$ by another matrix $[B(x_k)]$ of the iterative process of Newtons method

$$x_{k+1} = x_k - \alpha^*_k [H(x_k)]^{-1}\nabla f(x_k)$$

Using only the first partial derivative of f. If $[H(x_k)]^{-1}$ is approximated by $[B(x_k)]$, then equation (3.1) can be expressed as

$$x_{k+1} = x_k - \alpha^*_k [B(x_k)]\nabla f(x_k)$$

where α_k is the optimal step length along the direction

and ensure that

$$f(x_{k+1}) < f(x_k)$$

$$\alpha_k = \underset{\alpha \geq 0}{argmin}\ f(x_k - \alpha S_k)$$

Approximation of the Inverse Hessian

Let $B_0, B_1, B_2, \ldots$ be a successive approximations of the inverse $H(x_k)^{-1}$ of the Hessian

and $\nabla f(x_k)$ be . The researchers now derive a condition that the approximation should satisfy.

Suppose first that the Hessian matrix $H(x_k)$ of the objective function f is constant and independent of x. In other words, the objective function is quadratic, with

and for all x, where

Then

$$g_{k+1} - g_k = A_k(x_{k+1} - x_k)$$

Let

$$\Delta g_k = g_{k+1} - g_k$$

$$\Delta x_k = x_{k+1} - x_k$$

then (3.5) becomes

$$\Delta g_k = A_k \Delta x_k$$

The researchers start with a real symmetric positive definite matrix B_0. The researchers note given k, the matrix $[A^k]^{-1}$ satisfies.

$$(A_k)^{-1} \Delta g_{(i)} = \Delta x_{(i)}, 0 \le i \le k$$

The researchers also impose the requirement that the approximation B_{k+1} of the Hessian satisfy

$$B_{k+1} \Delta g_{(i)} = \Delta x_{(i)}, 0 \le i \le k$$

Moving in n dimensions, the researchers have $\Delta x_{(0)}, \Delta x_{(1)}, ..., \Delta x_{(n-1)}$ yield

$$B_n \Delta g_{(0)} = \Delta x_{(0)}$$

$$B_n \Delta g_{(1)} = \Delta x_{(1)}$$

$$\vdots$$

$$B_n \Delta g_{(n-1)} = \Delta x_{(n-1)}$$

Equation (3.11) can be represented as

$$B_n[\Delta g_{(0)}, \Delta g_{(1)}, ..., \Delta g_{(n-1)}] = [\Delta x_{(0)}, \Delta x_{(1)}, ..., \Delta x_{(n-1)}]$$

and A_k satisfies

$$A_k[\Delta x_{(0)}, \Delta x_{(1)}, ..., \Delta x_{(n-1)}] = [\Delta g_{(0)}, \Delta g_{(1)}, ..., \Delta g_{(n-1)}]$$

and

$$(A_k)^{-1}[\Delta g_{(0)}, \Delta g_{(1)}, ..., \Delta g_{(n-1)}] = [\Delta x_{(0)}, \Delta x_{(1)}, ..., \Delta x_{(n-1)}]$$

Therefore, $[\Delta g_{(0)}, \Delta g_{(1)}, ..., \Delta g_{(n-1)}]$ if is nonsingular, then (A_k) is uniquely determine after n steps, via

$$(A_k)^{-1} = B_n = [\Delta x_{(0)}, \Delta x_{(1)}, ..., \Delta x_{(n-1)}][\Delta g_{(0)}, \Delta g_{(1)}, ..., \Delta g_{(n-1)}]^{-1}$$

The Rank One Correction Formula

In the rank one correction formula, the correction term is symmetric and has the form $a_k Z^{(k)} Z^{(k)^T}$ where $a_k \in \Re$ and $Z^{(k)} \in \Re$.

The general formula for updating the matrix $[B_k]$ can be written as

$$B_{k+1} = B_k + a_k Z^{(k)} Z^{(k)^T}$$

$$rank Z^{(k)} Z^{(k)^T} = \left(rank \begin{bmatrix} Z_1^{(k)} \\ \vdots \\ Z_n^{(k)} \end{bmatrix} [Z_1^{(k)} \cdots Z_n^{(k)}] \right) = 1$$

and hence, the name "rank one" correction which is also called the single-rank symmetric (SRS algorithm).

To derive the rank one correction formula, the researchers aim to find a_k and Z^k. the researchers first consider the condition $B_{k+1} \Delta g^{(k)} = \Delta x^{(k)}$, they have

$$B_{k+1} \Delta g_{(k)} = (B_k + a_k Z^{(k)} Z^{(k)^T}) \Delta g_{(k)} = \Delta x_{(k)}$$

$$\Delta x_{(k)} - B_k \Delta g_{(k)} = (a_k Z^{(k)^T} \Delta g_{(k)}) Z^{(k)}$$

where $Z^{(k)^T} \Delta g_{(k)}$ is a scalar. Hence,

$$Z^{(k)} = \frac{\Delta_{(k)} - \beta^{(k)}}{(\alpha_k Z^{(k)T} \Delta_{g(k)}}$$

$$a_k Z^{(k)} Z^{(k)^T} = \frac{(\Delta x_{(k)} - B_k \Delta g_{(k)})(\Delta x_{(k)} - B_k \Delta g_{(k)})^T}{a_k (Z^{(k)^T} \Delta g_{(k)})^2}$$

$$B_{k+1} = B_k = \frac{(\Delta x_{(k)} - B_k \Delta g_{(k)})(\Delta x_{(k)} - B_k \Delta g_{(k)})^T}{a_k (Z^{(k)^T} \Delta g_{(k)})^2}$$

$$B_{k+1} = B_k + \frac{(\Delta x_{(k)} - B_k \Delta g_{(k)})(\Delta x_{(k)} - B_k \Delta g_{(k)})^T}{\Delta g_{(k)}^T (\Delta x_{(k)} - B_k \Delta g_{(k)})}$$

$$A(x_k)) = A(x_k)^T$$

The Rank Two Update

In rank two updates, the researchers take the sum of two rank one update as

$$B_{k+1} = B_k + a_1 Z_1^{(k)} Z_1^{(k)^T} + a_2 Z_2^{(k)} Z_2^{(k)^T}$$

The researchers now show that (3.23) satisfy the Quasi-Newton condition of (3.10)

$$\Delta x_{(k)} = B_k \Delta g_{(k)} + a_1 Z_1^{(k)} (Z_1^{(k)^T} \Delta g_{(k)}) + a_2 Z_2^{(k)} (Z_2^{(k)^T} \Delta g_{(k)})$$

where $Z_1^{(k)^T} \Delta g_{(k)}$ and $Z_2^{(k)^T} \Delta g_{(k)}$ are scalar. Thus, since $Z_1^{(k)}$ and $Z_2^{(k)}$ are not unique in (3.24), the researchers make a choice to satisfy it.

$$\Delta x_{(k)} = B_k \Delta g_{(k)} + a_1 Z_1^{(k)} (Z_1^{(k)^T} \Delta g_{(k)}) + a_2 Z_2^{(k)} (Z_2^{(k)^T} \Delta g_{(k)})$$

$$Z_1^{(k)} = \Delta x_{(k)}$$

$$Z_2^{(k)} = B_k \Delta g_{(k)}$$

$$a_1 = \frac{1}{Z_1^T} \Delta g_{(k)}$$

$$a_2 = \frac{1}{Z_2^{(k)^T} \Delta g_{(k)}}$$

substituting (3.25) into (3.24), the researchers obtain the rank two update called Davidon Flectcher Powell.

$$B_{k+1} = B_k + \frac{\Delta x_{(k)} \Delta x_{(k)}^T}{\Delta x_{(k)}^T \Delta g_{(k)}} - \frac{[B_k \Delta g_{(k)}][B_k \Delta g_{(k)}]^T}{[B_k \Delta g_{(k)}]^T \Delta g_{(k)}}$$

Equation (2.) can also be expressed as

$$B_{k+1} = B_k + \frac{\alpha_k S_k S_k^T}{S_k^T \Delta g_{(k)}} - \frac{[B_k \Delta g_{(k)}][B_k \Delta g_{(k)}]^T}{[B_k \Delta g_{(k)}]^T \Delta g_{(k)}}$$

where $\Delta x_{(k)} = \alpha_k S_k$.

Davidon Fletcher Powell Algorithms

Steps are:

1. Select an arbitrary initial point $x_{(0)}$, a real symmetric matrix B_0 and $\varepsilon=10^{-6}$.
2. Set $k=0$.
3. Compute $g_{(k)}$, If $\| g_{(k)} \| \leq \varepsilon$, terminate, else, set $S_k = -B_k g_{(k)}$.
4. Compute

update

$$x_{(k+1)} = x_{(k)} + \alpha_k S_k$$

5. Compute

$$\Delta x_{(k)} = \alpha_k S_k$$

$$\Delta g_{(k)} = g_{(k+1)} - g_{(k)}$$

$$B_{k+1} = B_k + \frac{\Delta x_{(k)} \Delta x_{(k)}^T}{\Delta x_{(k)}^T \Delta g_{(k)}} - \frac{[B_k \Delta g_{(k)}][B_k \Delta g_{(k)}]^T}{\Delta g_{(k)}^T B_k \Delta g_{(k)}}$$

6. Set $k=k+1$, go to steps 3.

Broyden Fletcher Goldfarb and Shanno (BFGS) Update

The researchers use the concept of duality to obtain the BFGs updates as they recall from 3.8 - 3.14 that the equation $B_{k+1} \Delta g_{(i)} = \Delta x_{(i)}$, $0 \leq i \leq k$. derived from $\Delta g_{(i)} = A_k \Delta x_{(i)}$, $0 \leq i \leq k$. The researchers then formulated update formula for the approximation to the inverse of the Hessian matrix $(A_k)^{-1}$. The researchers require H_{k+1} to satisfy

$$\Delta g_{(i)} = H_{k+1} \Delta x_{(i)}, 0 \leq i \leq k$$

Thus,the researchers interchange $\Delta x_{(i)}$ and $\Delta g_{(i)}$ and and B_k. In particular, the BFGS update for H_k corresponds to the DFP update for B_k.

Hence, the researchers start from DFP update

$$B_{k+1} = B_k + \frac{\Delta x_{(k)} \Delta x_{(k)}^T}{\Delta x_{(k)}^T \Delta g_{(k)}} - \frac{B_k \Delta g_{(k)} \Delta g_{(k)}^T B_k}{\Delta g_{(k)}^T B_k \Delta g_{(k)}}$$

Using the duality concept, the researchers obtain an update equation for approximation H_k of the Hessian

$$H_{k+1} = H_k + \frac{\Delta g_{(k)} \Delta g_{(k)}^T}{\Delta g_{(k)}^T \Delta x_{(k)}} - \frac{H_k \Delta x_{(k)} \Delta x_{(k)}^T H_k}{\Delta x_{(k)}^T H_k \Delta x_{(k)}}$$

To obtain the BFGS update for the approximation of the inverse hessian. The researchers take the inverse H_{k+1} to obtain

$$B_{k+1} = (H_{k+1})^{-1} = (H_k + \frac{\Delta g_{(k)} \Delta g_{(k)}^T}{\Delta g_{(k)}^T \Delta x_{(k)}} - \frac{H_k \Delta x_{(k)} \Delta x_{(k)}^T H_k}{\Delta x_{(k)}^T H_k \Delta x_{(k)}})^{-1}$$

To compute B_{k+1} by inverting the right-hand side of (3.26), the researchers apply the Sherman-Morrison formula stated as

$$(A + UV^T)^{-1} = A^{-1} - \frac{(A^{-1}U)(V^T A^{-1})}{1 + V^T A^{-1} U}$$

where A is a non-singular matrix and U and V are column vector. The researchers obtain

$$B_{k+1} = B_k + \left(1 + \frac{\Delta g_{(k)} B_k \Delta g_{(k)}^T}{\Delta g_{(k)}^T \Delta x_{(k)}}\right) \frac{\Delta x_{(k)} \Delta x_{(k)}^T}{\Delta x_{(k)}^T \Delta g_{(k)}} - \frac{B_k \Delta g_{(k)} \Delta x_{(k)}^T}{\Delta g_{(k)}^T \Delta x_{(k)}} - \frac{B_k \Delta g_{(k)}^T \Delta x_{(k)}}{\Delta g_{(k)}^T \Delta x_{(k)}}$$

$$\alpha_k = \arg\min f(x_k + \alpha S_k), \quad \alpha \geq 0$$

BFGS Update Algorithm

Steps are (Li and Fukushima 2001):

1. Select an arbitrary initial point $x^{(0)}$ and a nxn real symmetric matrix B_0 and $\varepsilon = 10^{-6}$.
2. Set $k=0$.
3. Compute $g_{(k)}$, If $\| g_{(k)} \| \leq \varepsilon$, Terminate; else set $S_k = -B_k g_{(k)}$.
4. Compute

update $x_{(k+1)} = x_{(k)} + \alpha_k S_k$

5. Compute $g_{(k+1)}$, If $\| g_{(k+1)} \| \leq \varepsilon$, terminate, else, go to 6
6. Compute

$$\Delta x_{(k)} = \alpha_k S_k$$

7. Update the Hessian matrix as

$$B_{k+1} = B_k + \left(1 + \frac{\Delta g_{(k)} B_k \Delta g_{(k)}^T}{\Delta g_{(k)}^T \Delta x_{(k)}}\right) \frac{\Delta x_{(k)} \Delta x_{(k)}^T}{\Delta x_{(k)}^T \Delta g_{(k)}} - \frac{B_k \Delta g_{(k)} \Delta x_{(k)}^T}{\Delta g_{(k)}^T \Delta x_{(k)}} - \frac{B_k \Delta g_{(k)}^T \Delta x_{(k)}}{\Delta g_{(k)}^T \Delta x_{(k)}}$$

8. Set $k = k+1$, go to step 3.

RESULTS

Numerical Results

The researchers report the Numerical results of the Newton's Method (NM), Davidon Fletcher Powell (DFP) and Broyden Fletcher Goldfarb Shanno (BFGS) in this section. The problems solved are standard test functions based on unconstrained optimization problems. The results of the experiments are summarized in the tables where the algorithms of NM, DFP and BFGS are implemented respectively.

The comparison of the performance of the methods are also presented.

Example 1: Extended Trigonometric Function

$$Minimize\ f(x) = \sum_{i=1}^{4}\left(\left(4 - \sum_{j=1}^{4} cosx_j\right) + i(1 - cosx_i) - sinx_i\right)^2, x_0 = [0.2,0.2,0.2,0.2]$$

Table 1 shows that Newton's method converges at third iterations considering four variables and the objective function is also reducing.

Table 2 shows that DFP method converges at eight iterations considering four variables and the objective function is also reducing.

Table 3 shows that BFGS method converges at sixth iterations considering four variables and the objective function is also reducing.

Example 2 : Hager Function

$$Minimize\ f(x) = \sum_{i=1}^{10}(exp(x_i) - \sqrt{ix_i}), x_0 = [1,1,1,...,1]$$

Table 4 shows that Newton's method converges at fifth iterations considering ten variables and the objective function is also reducing.

Table 5 shows that DFP method converges at fourteen iterations considering ten variables and the objective function is also reducing.

Table 6 shows that BFGS method converges at eight iterations considering ten variables and the objective function is also reducing.

Example 3 : Quadratic QF2 Function

$$Minimize\ f(x) = \sum_{i=1}^{10} i(x_i^2 - 1)^2 - x_n, x_0 = [0.5,0.5,0.5,...,0.5]$$

The numerical results of the Newton's method for Example 3 in Table 7 shows that Newton's method converges at sixth iterations considering ten variables and the objective function is also reducing.

Table 8 depicts that DFP method converges at twenty-one iterations considering ten variables and the objective function is also reducing.

Table 9 shows that BFGS method does not converges and terminate at seventh iterations considering ten variables and the objective function values are irregular..

Example 4 : Quartc Function(CUTE)

$$Minimize\ f(x) = \sum_{i=1}^{10}(x_i - 1)^4, x_0 = [2.0,2.0,2.0,...,2.0]$$

Table 10 shows that Newton's method converges at twelveth iterations considering ten variables and the objective function is also reducing.

Table 11 indicates that DFP method converges at second iterations considering ten variables and the objective function is also reducing.

BFGS Method for Example 4

Table 12 shows that BFGS method converges at second iterations considering ten variables and the objective function is also reducing.

Table 13.1 shows that the value of objective function for Newton's method converges at second iteration, DFP converges at seventh iterations and BFGS converges at fifth iterations.

Table 13.2 shows that the value of objective function for Newton's method converges at fourth iteration, DFP converges at thirteenth iterations and BFGS converges at seventh iterations .

Table 13.3 shows that the value of objective function for Newton's method converges at sixth iteration, DFP converges at twenty-one iterations and BFGS converges at seventh iterations.

Table 13.4 shows that the value of objective function for Newton's method converges at eleventh iteration, DFP converges at first iterations and BFGS converges at first iterations.

DISCUSSION

It was observed from the computational numerical result of Example 1 that Newton's method converges faster and performs better with less iteration, DFP has a low convergence rate with more iterations and BFGS performs better than DFP with less iterations. In Example 2, Newton's method performs better, follow by BFGS

Table 1: Extended trigonometric function: The Newton's method

k	χ_1	χ_1	χ_3	χ_3	f_k
	0.2000000000	0.2000000000	0.2000000000	0.2000000000	27.6778698800
	0.1055160013	0.1991396520	0.2893401672	0.3763019149	27.1634476800
	0.1069562346	0.1994316721	0.2890115884	0.3746123800	27.1633907800

Table 2: The numerical results of the DFP method for Example 1

k	χ_1	χ_2	χ_3	χ_4	f_k
	0.2000000000	0.2000000000	0.2000000000	0.2000000000	27.6778698800
	0.1204550379	0.1962411452	0.2720272524	0.3478133597	27.1764965200
	0.1212910535	0.2067633012	0.2838112726	0.3489617117	27.1739117600
	0.1110658064	0.2005876461	0.2878116694	0.3765732707	27.1636353800
	0.1069859837	0.1993955673	0.2890661108	0.3746124666	27.1633908600
	0.1069559806	0.1994424114	0.2890246451	0.3746173398	27.1633907900
	0.1052934127	0.2261453371	0.3221888209	0.3746121142	27.1839062000
	0.1069566240	0.1994317490	0.2890114121	0.3746121142	27.1633907800

Table 3: The numerical results of the BFGS method for Example 1

k	χ_1	χ_2	χ_3	χ_4	f_k
	0.2000000000	0.2000000000	0.2000000000	0.2000000000	27.6778698800
	0.1056307284	0.1955406303	0.2854505322	0.3753604340	27.1637019700
	0.1067621258	0.1993884909	0.2890649238	0.3744683505	27.1633914600
	0.1069591606	0.1994341937	0.2890059483	0.3746051425	27.1633908100
	0.1069603619	0.1994318498	0.2890081060	0.3746052126	27.1633908000
	0.1069565399	0.1994319409	0.2890112574	0.3746121053	27.1633907900

Table 4: Hager Function numerical results: The Newton's method

k	χ_1	χ_2	χ_3	χ_4	χ_5	
	1.0000000000	1.0000000000	1.0000000000	1.0000000000	1.0000000000	1.0000000000
	0.2526714251	0.3452855039	0.4122236412	0.4660872106	0.5117174457	0.5515785075
	0.1627524339	0.2790301739	0.3609495810	0.4253343393	0.4787870510	0.5246785214
	0.1753197844	0.2835400506	0.3629267760	0.4263020738	0.4792930356	0.5249544142
	0.1758659486	0.2835716434	0.3629306788	0.4263027512	0.4792931783	0.5249544475

Table 4: The numerical results of the Newton's method for Example 2 (contd...)

k	χ_1	χ_2	χ_3	χ_4	f_k
	1.0000000000	1.0000000000	1.0000000000	1.0000000000	4.7145401000
	0.5871236924	0.6192933373	0.6487362076	0.6759215712	0.2441202210
	0.5649901142	0.6009959294	0.6335683654	0.6633332581	0.2145827530
	0.5651446550	0.6010839344	0.6336189069	0.6633623321	0.2141315630
	0.5651446634	0.6010839368	0.6336189072	0.6633623324	0.2141308660

and then DFP. In Example 3, Newton's method still performs better, DFP has a very low convergence rate with more iterations while BFGS diverges at 6th iterations. Also in Example 4, DFP and BFGS performs better with just one iteration but newton's method performs poorly with more iterations.

Finally, the behaviour and performances of NM, DFP and BFGS algorithm depends on the nature of the objective function.

CONCLUSION

This paper presented the solution of nonlinear unconstrained optimization problem using the Newton's method, DFP and BFGS and the following observation were obtained:

1. DFP was developed to eradicate the computation of the inverse of the Hessian thereby keeping the positive definiteness of the approximate Hessian inverse.

Table 5: The numerical results of the DFP method for Example 2

k	χ_1	χ_2	χ_3	χ_4	χ_5
	1.0000000000	1.0000000000	1.0000000000	1.0000000000	1.0000000000	1.0000000000
	0.2520320401	0.3218650233	0.3754497557	0.4206237746	0.4604228840	0.4964040299
	0.1683311834	0.2814378064	0.3601413186	0.4222712062	0.4742864987	0.5193528516
	0.1772846235	0.2786857626	0.3585370696	0.4227416731	0.4764548452	0.5226910597
	0.1762492975	0.2841581938	0.3627157776	0.4260350762	0.4791306740	0.5248784026
	0.1761193130	0.2836283858	0.3630836241	0.4263740171	0.4792824897	0.5248825964
	0.1702242710	0.2823673037	0.3592033771	0.4249802759	0.4796527015	0.5264691152
	0.1699814321	0.2823160515	0.3590436354	0.4249224525	0.4796673386	0.5265337381
	0.1702250184	0.2823677170	0.3592054637	0.4249782675	0.4796513600	0.5264688989
	0.1742964897	0.2848842885	0.3541494564	0.4341058144	0.4826680833	0.5224650151
	0.1756993138	0.2837250123	0.3628284596	0.4261121486	0.4791420358	0.5247202558
	0.1758306001	0.2835915904	0.3628667940	0.4263058769	0.4792383296	0.5250262155
	0.1758439968	0.2835904773	0.3629156311	0.4262710582	0.4792797536	0.5249304879
	0.1758688707	0.2835702362	0.3629352640	0.4263015119	0.4792991893	0.5249460224

Table 5: The numerical results of the DFP method for Example 2 (contd..)

k	χ_1	χ_2	χ_3	χ_4	f_k
	1.0000000000	1.0000000000	1.0000000000	1.0000000000	4.7145401000
	0.5294921079	0.5602897398	0.5892155085	0.6165741805	0.2427247540
	0.5592868351	0.5952394006	0.6279915038	0.6581004486	0.2146198010
	0.5633209084	0.5995829221	0.6323413620	0.6622249759	0.2142696830
	0.5650964580	0.6010069293	0.6334702093	0.6631145707	0.2141322410
	0.5650324280	0.6009495207	0.6334777286	0.6632270271	0.2141311950
	0.5675180483	0.6040632691	0.6369268554	0.6369462083	0.2154729980
	0.5676195969	0.6041905780	0.6370678743	0.6370957258	0.2154726670
	0.5675179978	0.6040624789	0.6369249193	0.6369524391	0.2154723640
	0.5612161724	0.5999115674	0.6372876539	0.6541311494	0.2145980200
	0.5649508052	0.6009611918	0.6334690781	0.6633806887	0.2141312980
	0.5651578208	0.6009821313	0.6335369016	0.6633690997	0.2141309140
	0.5651331911	0.6010069891	0.6336191750	0.6633667513	0.2141308770
	0.5651440848	0.6010092481	0.6336247925	0.6633621237	0.2141308730

Table 6: The numerical results of the BFGS method for Example 2

k	χ_1	χ_2	χ_3	χ_4	χ_5
	1.0000000000	1.0000000000	1.0000000000	1.0000000000	1.0000000000	1.0000000000
	0.2520320401	0.3218650233	0.3754497557	0.4206237746	0.4604228840	0.4964040299
	0.1683311835	0.2814378065	0.3601413187	0.4222712062	0.4742864987	0.5193528516
	0.1772846233	0.2786857626	0.3585370691	0.4227416732	0.4764548451	0.5226910596
	0.1762492973	0.2841581941	0.3627157771	0.4260350761	0.4791306740	0.5248784026
	0.1761193130	0.2836283861	0.3630836243	0.4263740167	0.4792824896	0.5248825964
	0.1758711812	0.2835753062	0.3629202964	0.4263153520	0.4792980733	0.5249493756
	0.1758689590	0.2835713036	0.3629307853	0.4263030799	0.4792940156	0.5249555260

Table 6: The numerical results of the BFGS method for Example 2 (contd…)

k	χ_1	χ_2	χ_3	χ_4	f_k
	1.0000000000	1.0000000000	1.0000000000	1.0000000000	4.7145401000
	0.5294921079	0.5602897398	0.5892155085	0.6165741805	0.2427247540
	0.5592868350	0.5952394005	0.6279915037	0.6581004485	0.2146198030
	0.5633209081	0.5995829227	0.6323413616	0.6622249756	0.2142696850
	0.5650946581	0.6010069298	0.6334702093	0.6631145703	0.2141322400
	0.5650324283	0.6009495203	0.6334777287	0.6632270270	0.2141311850
	0.5651370509	0.6010805838	0.6336229082	0.6633730208	0.2141308630
	0.5651459428	0.6010853209	0.6336201937	0.6633633168	0.2141308710

Table 6: The numerical results of the BFGS method for Example 2 (contd…)

k	χ_1	χ_2	χ_3	χ_4	f_k
	1.0000000000	1.0000000000	1.0000000000	1.0000000000	4.7145401000
	0.5294921079	0.5602897398	0.5892155085	0.6165741805	0.2427247540
	0.5592868350	0.5952394005	0.6279915037	0.6581004485	0.2146198030
	0.5633209081	0.5995829227	0.6323413616	0.6622249756	0.2142696850
	0.5650946581	0.6010069298	0.6334702093	0.6631145703	0.2141322400
	0.5650324283	0.6009495203	0.6334777287	0.6632270270	0.2141311850
	0.5651370509	0.6010805838	0.6336229082	0.6633730208	0.2141308630
	0.5651459428	0.6010853209	0.6336201937	0.6633633168	0.2141308710

Table 7: Quadratic QF2 function results: The Newton's method

k	χ_1	χ_2	χ_3	χ_4	χ_5	χ_6
	0.5000000000	0.5000000000	0.5000000000	0.5000000000	0.5000000000	0.5000000000
	-1.0000000000	-1.0000000000	-1.0000000000	-1.000000000	-1.0000000000	-1.0000000000
	-1.0000000000	-1.0000000000	-1.0000000000	-1.000000000	-1.0000000000	-1.0000000000
	-1.0000000000	-1.0000000000	-1.0000000000	-1.000000000	-1.0000000000	-1.0000000000
	-1.0000000000	-1.0000000000	-1.0000000000	-1.000000000	-1.0000000000	-1.0000000000
	-1.0000000000	-1.0000000000	-1.0000000000	-1.000000000	-1.0000000000	-1.0000000000

Table 7: The numerical results of the Newton's method for Example 3 (contd…)

k	χ_1	χ_2	χ_3	χ_4	f_k
	0.5000000000	0.5000000000	0.5000000000	0.5000000000	14.9687500000
	-1.0000000000	-1.0000000000	-1.0000000000	-1.2000000000	2.1680000000
	-1.0000000000	-1.0000000000	-1.0000000000	-1.0259036140	1.0396734340
	-1.0000000000	-1.0000000000	-1.0000000000	-0.9777734889	0.9874354592
	-1.0000000000	-1.0000000000	-1.0000000000	-0.9740167497	0.9871707736
	-1.0000000000	-1.0000000000	-1.0000000000	-0.9739943540	0.9871707643

Table 8: The numerical results of the DFP method for Example 3

k	χ_1	χ_2	χ_3	χ_4	χ_5	χ_6
	0.5000000000	0.5000000000	0.5000000000	0.5000000000	0.5000000000	0.5000000000
	0.3474637443	0.1949274887	0.0423912330	-0.1101450226	-0.2626812783	-0.4152175339
	0.3608059450	0.2174661699	0.0685464909	-0.0858908899	-0.2442873854	-0.4035880223
	-0.0998934335	-0.5733705184	-0.8478373212	-0.8889935058	-0.7368793302	-0.5265856627
	-0.1546243247	-0.6685247509	-0.9575834156	-0.9851787747	-0.7980453466	-0.5450182250
	-0.5657402525	-1.1801200890	-1.0276997110	-0.8803303471	-1.0753294380	-1.0016473150
	-0.6647962879	-1.1218939450	-1.0198239020	-0.9823904609	-1.0264617610	-1.0585498310
	-0.6739293004	-0.9798662991	-0.9884384091	-1.0400928570	-1.0382253780	-1.0360457970
	-0.6721114152	-0.8918412415	-0.9831647128	-1.0377766720	-1.0907370290	-1.0741897360
	-0.6779442421	-0.7782808955	-1.0761542780	-0.9353472967	-1.0212945240	-0.9865736045
	-0.7457470733	-0.8074374043	-0.9703081505	-1.0053485070	-0.9161641634	-1.0104070850
	-0.8322563557	-0.8375906439	-0.9065271381	-1.0224654730	-0.9627863194	-0.9492305072
	-1.0800642680	-0.8559941983	-0.9233649445	-1.0037315670	-0.9666725184	-0.9761162411
	-0.9603601368	-0.9881113201	-0.9864522350	-0.9957541261	-1.0031139820	-1.0143433340
	-1.0223405160	-1.0016757510	-1.0033678980	-1.0068077030	-1.0019197420	-1.0052394570
	-1.0010528520	-1.0001332170	-0.9997535100	-0.9985660424	-1.0005684640	-0.9976098620
	-1.0000260780	-0.9998219153	-1.0005022110	-1.0000802800	-0.9996064566	-0.9997526262
	-1.0002063490	-1.0001046120	-0.9999437789	-0.9999374522	-0.9999646884	-1.0000124080
	-1.0000156260	-0.9999808721	-0.9999956688	-1.0000195630	-1.0000200110	-1.0000087030
	-0.9999997861	-0.9999961959	-0.9999987256	-0.9999984477	-0.9999980525	-1.0000009730
	-0.9999992665	-1.0000003300	-0.9999996920	-1.0000003340	-0.9999998487	-0.9999998655

Table 8: The numerical results of the DFP method for Example 3 (contd...)

k	χ_1	χ_2	χ_3	χ_4	f_k
	0.5000000000	0.5000000000	0.5000000000	0.5000000000	14.9687500000
	-0.5677537900	-0.7202900450	-0.8728263010	-1.2287442310	14.2969853400
	-0.5592414430	-0.7051999016	-0.8339192707	-1.0195278900	13.1639494400
	-0.4620036585	-0.7165330886	-1.2578834540	-1.0273270970	8.9169499870
	-0.4547080598	-0.7225419690	-1.2931456870	-1.0293081160	8.8029024100
	-0.8112220478	-1.0248814620	-1.2816879940	-1.0283058270	3.8755340170
	-1.0762871970	-1.0247958220	-1.2375289080	-1.0287735740	2.6930310780
	-1.0402298720	-1.1052037460	-1.2054908130	-1.0265161100	2.3810825180
	-1.0251563520	-1.0551586470	-1.1900817300	-1.0238711920	2.2425860240
	-1.0339937970	-1.0161667230	-1.1453840460	-1.0004450060	1.8362548120
	-0.9934322106	-0.9790557034	-1.1140658310	-0.9813301813	1.5481525870
	-1.0163519400	-1.0163519400	-1.1020099320	-0.9740713274	1.4443837870
	-1.0189597860	-1.0338055390	-1.0823089370	-0.9631215203	1.2810724530
	-1.0051828320	-1.0250678060	-1.0031331780	-0.9539882156	1.0126768390
	-0.9994828273	-1.0052816610	-0.9987617971	-0.9622647979	0.9920007111
	-0.9988590833	-1.0012045900	-0.9998844943	-0.9730196375	0.9873206759
	-1.0001199810	-1.0003034510	-0.9999244205	-0.9740710793	0.9871766272
	-1.0000533050	-1.0000763030	-0.9999408113	-0.9740521798	0.9871712155
	-0.9999850669	-1.0000220500	-0.9999915075	-0.9740119490	0.9871707922
	-0.9999984100	-0.9999997134	-0.9999986367	-0.9739942934	0.9871707645
	-0.9999999133	-0.9999999438	-0.9999996406	-0.9739943068	0.9871707643

Table 9: The numerical results of the BFGS method for Example 3

k	χ_1	χ_2	χ_3	χ_4	χ_5	χ_6
	0.5000000000	0.5000000000	0.5000000000	0.5000000000	0.5000000000	0.5000000000
	0.3474637443	0.1949274887	0.0423912330	-0.1101450226	-0.2626812783	-0.4152175339
	0.4886845863	0.4334886986	0.3192321855	0.1465734246	-0.0679906793	-0.2921246909
	0.1568125629	-0.2622527873	-0.7003166565	-0.9748244184	-0.9225479603	-0.6530707801
	0.1974262137	-0.2232684829	-0.6481975545	-0.8151537101	-0.7852498322	-0.6383650504
	-0.3104254072	-1.0892767380	-1.6944841040	-1.0694826350	-0.9284010795	-1.2330861560
	-0.2701634763	-0.8970325643	-1.9686565530	-1.1601076100	-1.0073273070	-1.3916062550

Table 9: The numerical results of the BFGS method for Example 3 (contd...)

k	χ_1	χ_2	χ_3	χ_4	f_k
	0.5000000000	0.5000000000	0.5000000000	0.5000000000	14.9687500000
	-0.5677537900	-0.7202900450	-0.8728263010	-1.2287442310	14.2969853400
	-0.4776546489	-0.5605680618	-0.4610139095	0.9857111250	14.8357695400
	-0.4881598402	-0.4105607808	-0.1103472429	0.3862736893	15.1993068100
	-0.5029229781	-0.4429235171	-0.0764954112	0.4758432537	15.0162460700
	-0.6985117769	-0.2424946061	-0.5656686270	0.0162412793	14.8859353900
	-0.5384481189	-0.1492258759	-0.7068143033	-0.2176190549	27.2134277900

Table 10: Quartc Function (CUTE): The Newton's method

k	χ_1	χ_2	χ_3	χ_4	χ_5	χ_6
	2.0000000000	2.0000000000	2.0000000000	2.0000000000	2.0000000000	2.0000000000
	1.6666666670	1.6666666670	1.6666666670	1.6666666670	1.6666666670	1.6666666670
	1.4444444450	1.4444444450	1.4444444450	1.4444444450	1.4444444450	1.4444444450
	1.2962962970	1.2962962970	1.2962962970	1.2962962970	1.2962962970	1.2962962970
	1.1975308650	1.1975308650	1.1975308650	1.1975308650	1.1975308650	1.1975308650
	1.1316872430	1.1316872430	1.1316872430	1.1316872430	1.1316872430	1.1316872430
	1.0877914950	1.0877914950	1.0877914950	1.0877914950	1.0877914950	1.0877914950
	1.0585276630	1.0585276630	1.0585276630	1.0585276630	1.0585276630	1.0585276630
	1.0390184420	1.0390184420	1.0390184420	1.0390184420	1.0390184420	1.0390184420
	1.0260122950	1.0260122950	1.0260122950	1.0260122950	1.0260122950	1.0260122950
	1.0173415300	1.0173415300	1.0173415300	1.0173415300	1.0173415300	1.0173415300
	1.0115610200	1.0115610200	1.0115610200	1.0115610200	1.0115610200	1.0115610200

Table 13: Table of comparison of the objective functions

Table 13.1: Table of comparison for Example 1

	NM	*DFP*	*BFGS*
χ^*_1	0.1069562	0.1069566	0.1069565
χ^*_2	0.1994317	0.1994317	0.1994319
χ^*_3	0.2890116	0.2890114	0.2890113
χ^*_4	0.3746124	0.3746121	0.3746121
f	27.163391	27.163391	27.163391
k	2	7	5

Table 13.2: Table of comparison for Example 2

	NM	*DFP*	*BFGS*
χ^*_1	0.1758659	0.1758689	0.175869
χ^*_2	0.2835716	0.2835702	0.2835713
χ^*_3	0.3629307	0.3629353	0.3629308
χ^*_4	0.4263028	0.4263015	0.4263031
χ^*_5	0.4792932	0.4792992	0.479294
χ^*_6	0.5249544	0.524946	0.5249555
χ^*_7	0.5651447	0.5651441	0.5651459
χ^*_8	0.6010839	0.6010092	0.6010853
χ^*_9	0.6336189	0.6336248	0.6336202
χ^*_{10}	0.6633623	0.6633621	0.6633633
f^*	0.2141309	0.2141309	0.2141309
k	4	13	7

Table 13.3: Table of comparison for Example 3

	NM	*DFP*	*BFGS*
χ^*_1	-1	-0.9999993	-0.2701635
χ^*_1	-1	-1.0000003	-0.8970326
χ^*_2	-1	-0.9999997	-1.9686566
χ^*_3	-1	-1.0000003	-1.1601076
χ^*_4	-1	-0.9999998	-1.0073273
χ^*_5	-1	-0.9999999	-1.3916063
χ^*_6	-1	-0.9999999	-0.5384481
χ^*_7	-1	-0.9999999	-0.1492259
χ^*_8	-1	-0.9999996	-0.7068143
χ^*_9	-0.9739944	-0.9739943	-0.2176191
χ^*_{10}	0.9871708	0.9871708	27.213428
f^*	0.9871708	0.9871708	27.213428
k	6	21	7

2. The BFGS was developed when the positive definiteness is not satisfied.

Table 13.3: Table of comparison for Example 3

	NM	*DFP*	*BFGS*
χ^*_1	-1	-0.9999993	-0.2701635
χ^*_1	-1	-1.0000003	-0.8970326
χ^*_2	-1	-0.9999997	-1.9686566
χ^*_3	-1	-1.0000003	-1.1601076
χ^*_4	-1	-0.9999998	-1.0073273
χ^*_5	-1	-0.9999999	-1.3916063
χ^*_6	-1	-0.9999999	-0.5384481
χ^*_7	-1	-0.9999999	-0.1492259
χ^*_8	-1	-0.9999996	-0.7068143
χ^*_9	-0.9739944	-0.9739943	-0.2176191
χ^*_{10}	0.9871708	0.9871708	27.213428
f^*	0.9871708	0.9871708	27.213428
k	6	21	7

3. The DFP and BFGS solutions agreed strongly with the solutions obtained by Newton's method.

RECOMMENDATIONS

The researchers recommend that further studies be carried out on class of constrained optimization problems.

REFERENCES

Gill PE, Murray W 1974. Newton-type methods for unconstrained and linearly constrained optimization. *Mathematical Programming,* 7(1): 311-350.

Jiang XW, Yan YT 2010. A self-scaling Quasi-Newton method for large scale unconstrained optimization. *Journal of Information and Computational Science,* 7: 1739-1745.

Li DH, Fukushima M 2001. A modified BFGS method and its global convergence in nonconvex minimization. *Journal of Computational and Applied Mathematics,* 129(1): 15-35.

Martinez JM 2000. Practical Quasi-Newton methods for solving nonlinear systems. *Journal of Computational and Applied Mathematics,* 124(1): 97-121.

Yuan YX, Byrd RH 1995. Non-Quasi-Newton updates for unconstrained optimization. *Journal of Computational Mathematics-International Edition,* 13: 95-107.

Paper received for publication on July 2016
Paper accepted for publication on December 2016