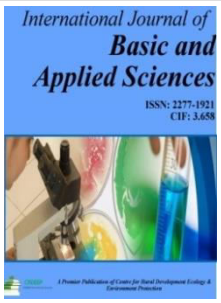
Vol. 10. No.1. 2021

Contents available at:

www.crdeepjournal.org

International Journal of Basic and Applied Sciences (ISSN: 2277-1921) (CIF:3.658 ; SJIF: 6.823)
(A Peer Reviewed quarterly Journal)



Full Length Research Paper

Quasi-Newton Line Search Algorithm for solving unconstrained non-linear least square Optimization Problem

Geleta Kinkino and Vasudeva Rao Kota

Lecturer, Department of Mathematics, Ambo University, Ethiopia
Assistant Professor Department Of Mathematics, Eritrea Institute Of Technology, Eritrea

ARTICLE INFORMATION

Corresponding Author:
Vasudeva Rao Kota

Article history:
Received:-04-01-2021
Revised: 30-01-2021
Accepted: 28-02-2021
Published: 02-03-2021

Key words:
Quasi-Newton-equation, Line Search, Wolfe conditions, Step length,

ABSTRACT

Quasi-Newton methods are among the most practical and efficient iterative methods for solving unconstrained optimization problem . These methods accelerate the steepest-descent technique for function minimization by using computational history to generate a sequence of approximations to the inverse of the Hessian matrix. Among those Quasi-Newton methods we focus on two different update formulas, which use line search strategy to find step length in each iteration. We call these methods Quasi-Newton line search methods, namely DFP and BFGS and applied this method over unconstrained non -linear least square problem. BFGS is well-liked for its efficiency and self-correcting properties to solve unconstrained non-linear least square optimization problem

Introduction

The most well-known minimization technique for unconstrained problems is Newton's Method. It is effective, robust and quadratic ally convergent [2, p.40-46]. In each iteration the step update is $x_{k+1} = xk - \nabla 2\, f\,(xk\,)^{-1} \wedge \nabla\, f\,(xk\,)$. The second derivative need to be calculated analytically and supplied to the algorithm by the user. However, Newton's 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 second derivatives may be unavailable. One fix to the problem is to use a finite difference approximation to the Hessian [1, p.115-116]. The other fix, which is more widely used, is quasi-Newton Methods, where approximate Hessian or inverse Hessian updates are updated in each iteration, while the gradients are supplied. In this paper we study such methods which use line strategy to find step length, and we call these method quasi-Newton line search methods. We consider solving the unconstrained non-linear least square minimization problem

$$min\, f\,(x) : x \in Rn \qquad (1)$$

The basic requirement for the updating formula is that the quasi-Newton equation or secant condition is satisfied in each iteration. A variety of formulas have been found to update *Bk* that satisfies secant condition. In this study we focus on two popular Quasi-Newton line search methods, namely DFP and BFGS methods. The properties and practical implementation of these methods are studied. The line search strategy is used to find the update step. Finally, numerical example over non-linear least square problem is taken and results are given.

Line Search

To compute the new step update in Quasi- Newton Methods we use the Line search strategy. In multivariable optimization algorithms, for given xk, the iterative scheme is

$$xk+1 = xk + \alpha k\, pk.$$

The key is to find the direction vector p_k and suitable step size αk. Let $\varphi(\alpha) = f\,(xk + \alpha pk\,)$.

So, the problem that departs from xk and finds a step size in the direction pk such that $\varphi(\alpha k) < \varphi(0)$ is just line search about α. If we find αk such that the objective function in the direction pk is minimized, i.e.
$f(xk + \alpha k\, pk) = \min f(xk + \alpha pk)$,
$\alpha > 0$

Then such line search is called exact(or optimal) line search [8, p.71]. If we choose α_k such that the objective functions has acceptable descent amount, i.e., such that the descent $(x_k) - f(x_k + \alpha_k p_k) > 0$ is acceptable by users, such a line is called in exact or approximate line search [8, p.71]. Inexact line search condition stipulates that αk should first of all give sufficient decrease in the objective function f ,as measured by the following inequality

$$f(x_k + \alpha p_k) \leq f(x_k) + c1\alpha\nabla f(x_k)^T$$

for some constant c1 ∈ (0, 1).

The sufficient decrease is not enough by itself to ensure that the algorithm makes reasonable progress, since it is satisfied for all sufficiently small value of α[11,p 38]. To rule out unacceptable short steps we introduce a second requirement called curvature condition, which requires αk to satisfy

$$\nabla f(x_k + ap_k)^T P_k \geq c_2 \nabla f_k^T p_k$$

for some constant c2 ∈ (c1, 1). Note that the left-hand side is simply the derivative $\varphi'(\alpha k)$, so that the curvature condition ensures that the slope of $\varphi(\alpha k)$ is greater than $c2$ times the gradient $\varphi'(0)$. This makes sense because if the slope $\varphi'(\alpha)$ is strongly negative, we have an indication that we can reduce f sufficiently by moving further along chosen direction. The sufficient decrease and curvature conditions are known collectively as the Wolfe conditions.

A step length may satisfy the Wolfe conditions without being particularly close to a minimizer of φ. However, we can modify the curvature condition to force α_k to lie in at least a broad neighborhood of local minimizer or stationary point of φ. The reason is that we have no longer allowed the derivative $\varphi'(\alpha_k)$ to be too positive. Hence, we exclude points that are far from stationary points of φ. Thus the strong Wolfe conditions requires α_k to satisfy

$$f(x_k + ap_k) \leq f(x_k) + c_1\alpha\nabla f_k^T \quad \nabla f(x_k + ap_k)^T P_k \leq |\nabla f_k^T p_k| \text{ with } 0 < c_1 < c_2 < 1.$$

Sufficient decrease and Backtracking
We presented that the sufficient decrease condition alone is not sufficient to ensure that the algorithm makes reasonable progress along the given search direction. However, if the line search algorithm chooses its candidate step lengths appropriately, by using backtracking approach, we can dispense with the second requirement called curvature condition and use just the sufficient decrease condition to terminate the line search procedure. Backtracking proceeds as follows

Procedure 1(Backtracking line Search)
Choose $\alpha_k > 0$, $\rho, c \in (0,1)$; set $\alpha \leftarrow \alpha_k$; repeat until $f(xk + \alpha pk) < f(xk) + c\alpha\nabla f T pk$ $\alpha \leftarrow \rho\alpha$ end repeat
Terminate with $\alpha_k = \alpha$. The backtracking approach ensures either that the selected step length αk is some fixed value(the initial choice α^-), or else that it is short enough to satisfy the sufficient decrease condition but not too short, see e.g.[11, p.42].

Quasi-Newton line search methods for Solving
Unconstrained non-linear optimization problem
We know that Newton's method $x_{k+1} = x_k - [\nabla^2 f_k]^{-1}\nabla f_k$ is successful because, it uses the Hessian which offers the useful curvature information. However, for various practical problems, the computing efforts of the Hessian matrices are very expensive, or the evaluation of the Hessian is difficult, even the Hessian is not available analytically. These lead to a class of methods that only uses the function values and the gradients 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 generates a series of Hessian approximations, and at the same time maintains a fast rate of convergence [8,203][1,116][18, p.249-287]. In this paper we study these Quasi-Newton methods which use line search strategy to find update step in each iteration

Formulation and criteria
The formulation of Quasi-Newton is to be presented, as well as some important criteria. search direction *pk* is obtained by solving following linear system

$$B_k P_k = -\nabla f(xk),$$

where Bk is a symmetric and positive definite matrix. During the iterations the matrix Bk is building up so it approximates the Hessian $\nabla 2 f(xk)$, while the objective function is while the objective function is minimized. Another way to present the Quasi-Newton method is by approximating the inverse Hessian Matrix formula i.e H_k approximates $\nabla^2 f(x_k)^{-1}$ [11, p.201]. The aim is to avoid solving a linear equation system each iteration. The Hessian approximation matrix B_{k+1} is updated according to

$$Bk+1 = Bk + Uk$$

There are several strategies for updating the Uk matrix. However, we focus on two update formula namely Davidon-Fletcher-Powell(DFP) and Broyden-Fletcher-Goldfarb-Shano(BFGS) Updating formula. These are presented in the next section. A condition to define the matrix *Bk* is

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

This is known as the Quasi-Newton condition (or secant condition). This condition is based on a generalization of $f'' \approx \frac{f'(x_k)-f'(x_{k-1})}{x_k-x_{k-1}}$. Where the formula first is made multidimensional and then the Hessian term $\nabla^2 f(x_k)$ is replaced by an approximation term B_k. Two vectors definitions follow below; these are introduced for simplicity and are used repeatedly further on,

$$sk = xk + 1 - x_k \; y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

These give a simplified expression of the secant condition.

$$Bk + 1sk = y_K \qquad (2)$$

This is known as the curvature condition. In fact, the secant condition always has a solution if the curvature condition is satisfied [11, p.195- 196][18]

Update method of U_k

Now we are looking for suitable updating formula for B_k which meet the secant condition.

DFP update

Various quasi-Newton updates obey the least change principle which refers to the $B_{k+!}$ $(or\, H_{k+1})$ being the minimum change to B_k $(or\, H_k)$ consistent with the secant condition if the change $B_{k+1} - B_k$ $(or\, H_{k+1} - H_k)$ is measured under some norm. This principle is helpful to maintain some information of the last iteration. Now, by use of the least change principle, we can derive DFP update.

Least change principile

Let us temporarily introduce for fixed $k \in N_0$ a different notation

$$s_k = x_{k+1} - x_k, \;\; s_k \mapsto s \;\; y_k = \nabla f(x_{k+1}) - \nabla f(x_k), \;\; y_k \mapsto y$$
$$B_k \mapsto B, \;\; B_{k+1} \mapsto B'$$

The secant condition (2.1) then reads $B's = y$. Donate by M_n the set of all real-valued (n,n)

Definition .1 *for fixed symmetric and positive definite* $W \in M_{n,l}$

$$\|A\|_w = \left\| W^{\frac{1}{2}} A W^{\frac{1}{2}} \right\|_F$$

where $W^{\frac{1}{2}}$ is square root of positive definite matrix W and $\| \;\; \|_F$ *is the well-known FROBENIUS norm with*

$$\|A\|^2_w = \sum_{i,j=1}^{n} trace\,(A^T A)$$

for any matrix, $A \in M_n$.

Recall that from the last chapter if a given matrix is positive definite, then the matrix is non-singular. Assuming that the average hessian $\bar{G}$ *defined*

$$\bar{G}_k = \int_0^1 \nabla^2 f(x_k + \tau s_k) d\tau \qquad (4)$$

is positive definite, for concreteness the readers can assume that $W = \bar{G}_k^{-1}$. With this choice of matrix W the above norm is dimensionless, which is desirable property, since we do not wish the solution of the following convex optimization problem as to depend on the units of the problem. Before considering these optimization problem note that from fundamental theorem of calculus, that is if f is continuous at every point in $[a,b]$ and F is ant derivative of f on $[a,b]$, then $\int_a^b f(t)dt = F(b) - F(a)$. Similarly for two times continuously differentiable function f the property

$$y_k = \bar{G}_k s_k \qquad (5)$$

follows from fundamental theorem of calculus, since $y_k = \nabla f(x_k + s_k) - \nabla f(x_k)$.

Proposition .1 Suppose $W \in M_n$ *symmetric and positive definite* $s, y \in R^n$ *with,* $s \neq 0$, $c = W^{-1}s$ *and* $B \in M_n$ *symmetric. Then we obtain the following rank 2 update of B, that is a matrix of (at most) rank two added to B,*

$$B' = B + \frac{(y - Bs)c^T + c(y - Bs)^T}{\langle c, s\rangle} - \frac{\langle y - Bs, s\rangle cc^T}{\langle c, s\rangle^2}$$

as the unique solution of the convex optimization problem

$$min\{\|A - B\|_w : A \in M_n, with\ A^T = A\ and\ As = y\}.$$

This is called a principle of least change see, e.g.[3, p.133

proof: The existence of a minimizer A of this problem is as follows:

For the moment we only consider the case $W = I$. For $A \in M_n$ we have to minimize the term $\frac{1}{2}\|A - B\|_F^2$ subject to constraint $As = y$ and $A = A^T$ with the corresponding lagrangian

$$L(A, \rho, \sigma) = \frac{1}{2}\sum_{i,j=1}^{n}(a_{ij} - b_{ij})^2 + \sum_{i=1}^{n}\rho\left(\sum_{j=1}^{n}a_{ij}s_j - y_i\right) + \sum_{i,j=1}^{n}\sigma_{ij}(a_{ij} - a_{ji})$$

$(\rho \in R^n\ and\ A, \sigma \in M_n)$. It holds for the minimizer A and $1 \le i, j \le n$ that

$$\frac{\partial L}{\partial a_{ij}} = a_{ij} - b_{ij} + \rho_i s_j + \sigma_{ij} - \sigma_{ji} = 0 \quad (7)$$

$$\sum_{j=1}^{n}a_{ij}s_j = y_i \quad (8)$$

$$a_{ij} = a_{ji} \quad (9)$$

Exchanging the indices in (7) yields

$$a_{ji} - b_{ji} + \rho_{ji} + \sigma_{ji} - \sigma_{ij} = 0$$

Addition to (7) and consideration of (9) leads to the relation

$$2a_{ij} - 2b_{ij} + \rho_i s_j + \sigma_{ij} - \sigma_{ji} = 0$$

respectively

$$A = B - \frac{1}{2}(\rho s^T + s\rho^T) \quad (10)$$

It follows from 7 that

$$y = As = Bs - \frac{1}{2}(\rho s^T s + s\rho^T s)$$
$$= Bs - \frac{1}{2}(\rho\langle s, s\rangle + s\langle \rho, s\rangle)$$

setting $w := y - Bs$ we obtain from that:

$$\langle s, w\rangle = \langle s, \rho\rangle\ or\ \langle s, \rho\rangle = -\frac{\langle s, w\rangle}{\langle s, s\rangle}$$

with (8) we get

$$\langle s, s\rangle\rho = -2w - \langle \rho, s\rangle \text{ or } \rho = -2\frac{w}{\langle s,s\rangle} + \frac{\langle s,w\rangle}{\langle s,s\rangle^2}s$$

Hence

$$A = B - \frac{1}{2}(\rho s^T + s\rho^T)$$
$$= B - \frac{1}{2}\{\left(-2\frac{w}{\langle s,s\rangle} + \frac{\langle s,w\rangle}{\langle s,s\rangle^2}s\right)s^T + s\left(-2\frac{w^T}{\langle s,s\rangle} + \frac{\langle s,w\rangle}{\langle s,s\rangle^2}s\right)s^T\}$$
$$= B + \frac{ws^T + sw^T}{\langle s,s\rangle} + \frac{\langle s,w\rangle}{\langle s,s\rangle^2}ss^T.$$

This gives the result in this case (taking into account s = c)

b) For arbitrary W we set $M := W^{\frac{1}{2}}$ and obtained the desired result by considering

$$M(A - B)M = \underbrace{MAM}_{:=\hat{A}} - \underbrace{MBM}_{:=\hat{B}}$$

and

$$As = y \Leftrightarrow (MAM)\underbrace{M^{-1}s}_{:=\hat{s}} = \underbrace{My}_{:=\hat{y}} \Leftrightarrow \hat{A}\hat{s} = \hat{y}$$

The optimization problem is equivalent to

$$min\{\|\hat{A} - \hat{B}\|_w : \hat{A} \in M_n, with\ \hat{A}^T = \hat{A}\ and\ \hat{A}\hat{s} = \hat{y}\}.$$

Following a) we obtain with $\hat{w} := \hat{y} - \hat{B}\,\hat{s}$ the solution

$$A = \hat{B} + \frac{\hat{w}\hat{s}^T + \hat{s}\hat{w}^T}{\langle \hat{s}, \hat{s} \rangle} + \frac{\langle \hat{s}, \hat{w} \rangle}{\langle \hat{s}, \hat{s} \rangle^2} \hat{s}\hat{s}^T$$

Because of $\hat{s} = Mc$, $\hat{w} = Mw$, $\langle \hat{s}, \hat{s} \rangle = \langle M^{-1}s, M^{-1}s \rangle = \langle s, M^{-2}s \rangle = \langle s, W^{-1}s \rangle = \langle s, c \rangle$
$\langle \hat{s}, \hat{w} \rangle = \langle M^{-1}s, Mw \rangle = \langle M M^{-1}s, w \rangle = \langle s, w \rangle$ we have as claimed

$$A = B + \frac{wc^T + cw^T}{\langle c, s \rangle} + \frac{\langle c, w \rangle}{\langle c, s \rangle^2} cc^T$$

Hence, with $c = y$ we obtain the updating formula of Davidon-Fletcher-Powell(DFP)

$$B' = B + \frac{(y - Bs)c^T + c(y - Bs)^T}{\langle c, s \rangle} - \frac{\langle y - Bs, s \rangle cc^T}{\langle c, s \rangle^2}$$

This is called DFP update.
When updating a symmetric positive definite $B' := B_k$ in quasi-Newton method, it is desirable that $B' = B_{k+1}$ symmetric positive definite too.

Preposition If B is symmetric positive definite, then with assumption $sy^T > 0$. B'_{DFP} is also symmetric positive definite. For $H'_{DFP} := B'^{-1}_{DFP}$ and $H := B^{-1}$ it holds that

$$H'_{DFP} = H + \frac{ss^T}{\langle s, y \rangle} - \frac{Hyy^T H}{\langle y, Hy \rangle} \qquad (11)$$

and we get the original DFP update , see, e.g,.[3, p.131-137]
proof For $x \in R^n$ we have

$$\langle x, B'_{DFp} x \rangle = \langle x, Q^T B Q x + \frac{yy^T}{\langle s, y \rangle} x \rangle = \underbrace{\langle Qx, B(Qx) \rangle}_{\geq 0} + \langle \frac{\langle y, x \rangle}{\langle s, y \rangle} \rangle^2 \geq 0$$

where $Q = I - \frac{ys^T}{\langle s, y \rangle}$. From $\langle x, B'_{DFP} x \rangle = 0$ it thus follows $\langle Qx, B(Qx) \rangle = 0$ and $\langle y, x \rangle = 0$
Hence, $Qx = 0$ and $\langle y, x \rangle = 0$ and finally $x = 0$. Note that since $Q^2 = Q$ the matrix Q is a projection with $Qs = 0$, [3, p.136]. If we donate the right hand side of (11) by $\hat{H}$, $B'_{DFP}\hat{H} = I$ follows after some transformations from

$$(Q^T B Q + \frac{yy^T}{\langle s, y \rangle})(H + \frac{ss^T}{\langle s, y \rangle} - \frac{Hyy^T H}{\langle y, Hy \rangle})$$

Hence, $H'_{DFP} = \hat{H}'$

Algorithm 1 (Quasi-Newton line search algorithm)
Initial step: Given $x_0 \in R^n$ an initial point, $H_0 \in R_n$ a symmetric and positive definite matrix, $\varepsilon > 0$, a termination scalar, $k := 0$, $k - th$ step: For $K = 0,1,2, \ldots,$

1. If $\|\nabla f(x_k)\| \leq \varepsilon$, stop.
2. Compute $p_k = -H_k \nabla f(x_k)$
3. Compute the step size α_k using line search
4. Set $s_k = \alpha_k p_k$, $x_{k+1} = x_k + s_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
   $and\ H_{k+1} = H_k + \frac{s_k s_k^T}{\langle s_k, y_k \rangle} - \frac{H_k y_k y_k^T H_k}{\langle y_k, H y_k \rangle}$.
5. $k := k + 1$ go to step 1

The DFP updating formula is quite effective, but it was soon superseded by the BROYDEN–FLETCHER GOLDFARB-SHANO updating formula (BFGS),which is presently considered to be the most effective of all quasi-Newton updating formulae,[11, p.197][5, p.194].

ii. BFGS update

BFGS updating can be derived by making simple change in the argument that led to (2). Instead of imposing conditions on the Hessian approximations B_k, we impose similar condi-tions on their inverse H_k. If we exchange $y \leftrightarrow s$ as well as $B \leftrightarrow H$, it follows with proposition 1.

The secant condition (2), now written as $H_{k+1} y_k = s_k$ and the condition closeness to H_k be now specified by the following an analogue of proposition 1:

$$min\{\|H - H_k\|_w : H \in M_n, with\ H^T = H\ and\ Hy = s\} \qquad (12)$$

The norm is again Frobenious norm described above, where the weight matrix W is now any matrix $Ws = y$. Then the solution is

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T \qquad (13)$$

where $\rho_k = \frac{1}{(y_k s_k^T)}$.

The update formula for B_k is obtained by apply-ing the Sherman-Morrison-Wood bury formula (A.2) to (2.14), see e.g. [1, p.119] [11, p.198][6]. Then

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k{}^T)}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \qquad (14)$$

Note that the minimization problem (12) that give rise to the BFGS update formula does not explicitly require the updated hessian approximation to be positive definite. It is to show however, that H_{k+1} will be positive definite whenever H_k is positive definite, by using the following preposition.

Proposition .3 *(Positive definitiveness of BFGS update)*

Given H_k symmetric and positive definite, then the BFGS update

$$H_{k+1} = \left(I - \frac{s_k y_k{}^T}{s_k^T y_k}\right) \mathrm{H_k} (I - \frac{y_k s_k{}^T}{s_k^T y_k}) + \frac{s_k s_k^T}{s_k^T y_k})$$

is positive definite if and only if $s_k^T y_k > 0$. Expanding $s_k^T y_k > 0$, we have $\nabla f(x_{k+1}) s_k > \nabla f(x_k) s_k$. Remember that in a minimum search algorithm we have $s_k = a_k p_k$ with $\alpha_k > 0$. But the second Wolfe condition (i.e. curvature condition) for line search is

$$\nabla f(x_k + a_k p_k)^T P_k \geq c_2 \nabla f_k^T p_k$$

with $0 < c_2 < 1$. But this implies

$$\nabla f(x_{k+1}) s_k > c_2 \nabla f(x_k) s_k > \nabla f(x_k) s_k \Longrightarrow$$

$s_k^T y_k > 0$. Let $s_k^T y_k > 0$, consider for any vector $z \neq 0$ we have

$$z^T H_{k+1} z = w^T H_k w + \frac{(z^T s_k)^2}{s_k^T y_k} > 0$$

where $z - \frac{y_k (s_k^T z}{s_k^T y_k}$. In order to have $z^T H_{k+1} z = 0$ we must have $w = 0$ and $Z^T s_k = 0$.

But, $Z^T s_k = 0 \Longrightarrow w = z$ and this implies $z = 0$.

Let $z^T H_{k+1} z > 0$ for all $z \neq 0$: choosing $Z = y_k$ we have

$$0 < y_k^T H_{k+1} y_k = \frac{(s_k^T y_k)^2}{s_k^T y_k} = s_k^T y_k \text{ and thus } s_k^T y_k > 0.$$

Algorithm 1 (Quasi-Newton line search algorithm using BFGS update)

Initial step: Given $x_0 \in R^n$ an initial point, $H_0 \in R_n$ a symmetric and positive definite matrix, $\varepsilon > 0$, a termination scalar, $k \coloneqq 0$, $k - th$ step: For $k = 0,1,2, \ldots$,

1. If $\|\nabla f(x_k)\| \leq \varepsilon$, stop.
2. Compute $p_k = -H_k \nabla f(x_k)$
3. Compute the step size α_k using appropriate line search
4. Set $s_k = \alpha_k p_k$, $x_{k+1} = x_k + s_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
   $and\ H_{k+1} = \left(I - \frac{s_k y_k{}^T}{s_k^T y_k}\right) \mathrm{H_k} (I - \frac{y_k s_k{}^T}{s_k^T y_k}) + \frac{s_k s_k^T}{s_k^T y_k})$
   .
5. $k \coloneqq k + 1$ go to step 1

Practical implementation

A few practical implementations need to be added to algorithm 2 to produce an efficient implementation. The line search which should satisfy either the Wolfe conditions or strong Wolfe conditions should always try step length $\alpha_k = 1$ first, because this step length will eventually always accepted, there by producing super-linear convergence of the overall algorithm of BFGS [11, p.200]. The value $c_1 = 10^{-4}$ and $c_2 = 0.9$ are commonly used in Wolfe conditions.

The initial matrix H_0 is often is set to some multiple βI of the identity, where β is scaling factor, but there is no good general strategy for choosing. If β is too large, so that the first step $p_0 = -\beta g_0$ is too long, many functions evaluation may be required to find a suitable value for the step length α_0.

A heuristic that is often quite effective is to scale the starting matrix after the step has been computed, but before the first BFGS update is performed. We change the provisional value $H_0 = I$ by setting

$$H_0 \leftarrow \frac{y_k^T s_k}{y_k^T y_k} I$$

before applying the update (13), (14) to obtain H_1. This formula attempts to make the size H_0 similar to that of $[\nabla^2 f(x_0)]^{-1}$ in the following sense. Assuming that average Hessian defined in (2) is positive definite, there exists a square root $\bar{G}_k^{\frac{1}{2}}$, satisfying $\bar{G}_k = \bar{G}_k^{\frac{1}{2}}\bar{G}_k^{\frac{1}{2}}$ (by positive definiteness). Therefore, by defining $z_k = \bar{G}_k^{\frac{1}{2}} s_k$ and using the relation (5), we have

$$\frac{y_k^T s_k}{y_k^T y_k} = \frac{(\bar{G}_k^{\frac{1}{2}} s_k)^T (\bar{G}_k^{\frac{1}{2}} s_k)}{(\bar{G}_k^{\frac{1}{2}} s_k)^T \bar{G}_k (\bar{G}_k^{\frac{1}{2}} \bar{G}_k^{\frac{1}{2}})^T} = \frac{z_k^T z_k}{z_k \bar{G}_k z_k} \qquad (15)$$

The reciprocal of (15) is an approximation to one of the Eigen values of $\bar{G}_k$ which in turn is close to an Eigen value of $\nabla^2 f(x_k)$, see [11, p.201][10]). Hence, the quotient (15) itself approximates an eigenvalue of $[\nabla^2 f(x_k)]^{-1}$. Other scaling factor can be used in above, but the one presented here appears to be the most successful in practice.

Self-scaling quasi-Newton methods

The standard quasi-Newton methods that we considered so far may have difficulties in solving some ill-conditioned problems. In this section we consider self- scaling hessian approximation matrix B_k can be updated according to a self-scaling BFGS update of the form

$$B_{k+1} = \rho_k \left(B_k - \frac{(B_k s_k)(B_k s_k{}^T)}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \right) \qquad (16)$$

where $\rho_k = \frac{y_k^T s_k}{s_k^T B_k s_k}$ is self-scaling factor. A suggestion of Al-Baali see e.g.,[7], is to modify the scaling factor to

$$\rho_k = \min\{1, \frac{y_k^T s_k}{s_k^T B_k s_k}\}$$

If applying this self-scaling quasi-Newton method on the DFP method, it takes the form

$B_{k+1} = \rho_k (I - \gamma_k y_k s_k^T) B_k (I - \gamma_k s_k y_k^T) + \gamma_k y_k y_k^T$

where $\gamma_k = \frac{1}{y_k^T s_k}$ and $\rho_k = \min\{1, \frac{y_k^T s_k}{s_k^T B_k s_k}\}$.

Solving non-linear least square problem

This paper devoted to solve unconstrained non-linear least squares optimization problems

$$\min_{x \in R^n} f(x) = \frac{1}{2} r(x)^T r(x) = \frac{1}{2} \sum_{i=1}^{m} [r_i(x)]^2,$$

$$m \geq n \qquad (16)$$

where $r: R^n \rightarrow R^m$ is non –linear function. Non-linear least squares problem can be regarded as unconstrained minimization with special structure. Non-linear leas square problems have wide application in data fitting, parameter estimation, function approximations and others, see e.g. [8, p.247-271][17, p273-275]

Numerical examples and results

In the following problem, we need to solve non-linear least square problem and then discuss over the result. First, we want to minimize using our matlab code which is written on appendix B. Finally, we want to see with result's obtained using minimization toolbox function fminunc which uses a line search based DFP and BFGS, see e.g.[12][15, p.350][16, p.8].

Example .1 $min\ f(x_1, x_2) = \frac{1}{2}(x_2 - x_1^2)^2 + (1 - x_1)^2$

Solution: Table 1: Matlab's result obtained using DFP algorithm

Algorithm :
Line search : backtracking

Iteration	x_ current	alpha	‖ g (xc)‖
0 . 0000	[0 . 6 0 0 0	0 . 0 0 0 0]	1 . 0 0 0 0	5. 1 4 8 0 e−001
1 . 0 0 0 0	[0 . 9 6 8 0	0 . 3 6 0 0]	1 . 0 0 0 0	1 . 2 0 0 8 e+000
2 . 0 0 0 0	[0 . 9 6 8 0	0 . 3 6 0 0]	0 . 0 0 0 0	1 . 2 0 1 1 e+000
3 . 0 0 0 0	[1 . 0 6 7 4	1 . 1 9 9 8]	1 . 0 0 0 0	6 . 0 7 0 1 e−002
4 . 0 0 0 0	[1 . 0 1 3 4	1 . 0 3 0 6]	1 . 0 0 0 0	2 . 0 0 1 7 e−002
5 . 0 0 0 0	[0 . 9 9 9 6	0 . 9 9 8 0]	1 . 0 0 0 0	1. 8 3 7 1 e−003
6 . 0 0 0 0	[1 . 0 0 0 0	1 . 0 0 0 0]	1 . 0 0 0 0	1 . 0 7 5 3 e−004
7 . 0 0 0 0	[1 . 0 0 0 0	1 . 0 0 0 0]	1 . 0 0 0 0	2 . 2 9 2 0 e−006

Table 2: Matlab's result obtained using BFGS algorithm
Algorithm :
Line search : backtracking

Iteration	x_current	alpha	‖ g (xc)‖
0 . 0000	[0 . 6 0 0 0	0 . 0 0 0 0]	1 . 0000 5 . 1480 e−001
1 . 0000	[0 . 9 6 8 0	0 . 3 6 0 0]	1 . 0000 1 . 2008 e+000
2 . 0000	[0 . 9 6 8 0	0 . 3 6 0 0]	0 . 0000 1 . 2010 e+000
3 . 0000	[1 . 0 6 7 4	1 . 2 5 0 7]	1 . 0000 1 . 5170 e−001
4 . 0000	[1 . 0 1 9 3	1 . 0 4 0 6]	1 . 0000 3 . 5032 e−002
5 . 0000	[1 . 0 0 1 6	0 . 9 9 9 7]	1 . 0000 1 . 0735 e−002
6 . 0000	[1 . 0 0 0 1	0 . 9 9 9 8]	1 . 0000 1 . 0269 e−003
7 . 0000	[1 . 0 0 0 0	1 . 0 0 0 0]	1 . 0000 1 . 2009 e−006

These results are unsurprising as we expected the algorithms to converge to the minimizer of f. We also get convergence in 7 steps. This is very good given the non-linear least square problem. The following table is summary of the number of iteration needed to converge to the minimum point x=(1,1) with different starting point using our matlab code and optimization toolbox function fminunc. Consider

Column A represents number of iteration required to converge to $x = (1,1)$ using mat- lab code in appendix B for DFP method. [see table 1].
Column B represents number of iteration required to converge to $x = (1,1)$ using mat- lab code in appendix B for BFGS method, [see table 2].
Column C represents number of iteration required to converge to $x = (1,1)$ using fminunc which uses DFP update.
Column D represents number of iteration required to converge to $x = (1,1)$ using fminunc which uses BFGS update.

Table 3: Comparison of the DFP and the BFGS methods

Starting	**A**	**B**	**C**	**D**
(10, −8)	14	13	98	24
(−9, 7)	21	22	30	20
(0.6, 0)	7	7	14	8
(0, 0)	7	7	8	8
(1, −1)	11	8	11	10
(−1, 1)	14	9	21	9
(−1, −1)	11	9	13	10
(1, 1)	0	0	0	0
(0.8, 0.6)	5	6	6	6
(6, 6)	17	14	101	21

Discussion

Our matlab code uses backtracking line search procedure to obtain step length which satisfies sufficient decrease condition. fminuc uses mixed quadratic and cubic interpolation. However, as we see from above table the number of iteration required to converge to the minimizer $x = (1,1)$ is varies. This is due to computational efficiency of line search procedure to find appropriate step length that satisfy sufficient decrease condition without being not too small. In addition to this, we can see that the efficiency of BFGS method over DFP method from above table. Thus, BFGS method is well- liked for its efficiency to solve non-linear least square optimization problem.

References

[1] Philip E. Gill Walter Murray and Margaret H. Wright: Practical optimization, second edition, 1982.
[2] R. Fletcher: Practical Methods of Opti- mization, Second edition, John Wiley & Sons, Chichester,2000.
[3] Wilhelm Forst -Dieter Hoffmann: Opti- mization theory and practice, springer, New York, 2010.
[4] M.S.Bazaraa, H.D. Sherall, C.M.Shetty: Non-linear programming-Theory and al- gorithms, second edition, 1993.
[5] J.E. Dennis and R.B. Schnabel, Numerical methods for unconstrained optimization, Prentice-Hall, Englewood Cliffs, NJ, 1983
[6] G.R.WALSH, Methods of optimazation,
John Wiley & Sons, New york, 1975
[7] M. Al-Baali and H. Khalfan, An over view of some practical quasi-Newton method for unconstrained Optimization, SQU Journal For Science, 12 (2):199–209,
2007.

[8] Wenyu Sun, YA-Xing Yuan: Optimization theory and methods, springer, New York, 2006.
[9] C.T. Kelley: iterative method for opti- mization, SIAM at http://www.ec-secure host.com/SIAM/FR18.html,1999.
[10] Fletcher, R., Powell, M. J. D.A rapidly con- vergent descent method for minimization, 1963/1964,p.163–168.
[11] J. Nocedal, S.J. Wright: Numerical op- timization, springer, Berlin, Heidelberg, New york, 2006
[12] MATLAB (2006): Optimization Toolbox, User's guide, The Math works Inc.
[13] Ronald Schoenberg, Optimization with the quasi-Newton method, September 5, 2001
[14] J.E.Dennis and H.Wolkowicz, Sizing and least change methods, march 1990
[15] Won Young Yang, W. Cao, T.S. Chung and J.Morris: Applied Numerical method us- ing matlab, 2005
[16] Thomas F. Coleman, Wei Xu and Gang Liu, a Secant Method for Nonlinear Least- Squares Minimization, 2004.
[17] Singresu S. Rao, Enginering optimization theory and practice , Fifth edition,2019
[18] Mishra, Shashi & Ram, Bhagwat. (2019). Introduction to Unconstrained Optimization with R. 10.1007/978-981-15-0894-3.

APPENDIX

Appendix A: SHERMAN-MORRISON- WOODBURY FORMULA

If the square non-singular matrix A undergoes a rank-one update to become

$$\bar{A} = A + ab^T,$$

where $a, b \in R^n$, then if $\bar{A}$ is non-singular, we have

$$\bar{A}^{-1} = A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1 + b^TA^{-1}a} \qquad (A.1)$$

This formula can be extended to higher-rank updates [5] [11]. Let *U* and *V* be matrices in $R^{n\times p}$ for some *p* between 1 and *n*. If we define

$$\bar{A} = A + UV^T,$$

then

$$\bar{A}^{-1} = A^{-1} - A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1} \qquad (A.2)$$

Appendix B: *Quasi-Newton Methods MATLAB Code*

```
function [xn,i]=Qnewt(f,xc,option,a)
%Qnewt , Performs the quasi-Newton method algorithms for finding
% a minimum of function f, with initial position xc with optional,
% The function returns the location of the minimum xn and the number of
% iterations. Qnewt requires inputs of the function to be minimized f,
% the starting location (xc) and can be run with a specified
% step size (a), the option input allows a selection of quasi-Newton algorithms
% OPTION=1 -DFP Method
% OPTION=2 -BFGS Method
% EXAMPLES:
% >> syms x y
% >> xinit=[0.6,0];
% >> [xfin,n]=Qnewt(f,xinit,1,1); %This is DFP Method and take f for intance(example 2)
% xfin =[1.0000,1.0000] , n=8
clc
% Calculate Gradient syms x y S=symvar(f).'; g=jacobian(f).';
% quasi-Newton method Hc=eye(length(S));
i=0; alpha=1; gc=subs(g,S,xc);% First Iteration Table
if option==1, str2=0DFP0; elseif option==2, str2=0BFGS0;
else str2=0 0; display(0Improper Variant Selec-tion nn0);
end
% Output Table Header and Initial Values fprintf(0Algorithm: %s \n Linesearch: %s \n',str2,linstr);
fprintf(0Iteration \t x-current \t\t\t alpha \t\t ‖ g(xc)‖ \n',);
d1=num2str(i,0%10.4 f '); d2=num2str(xc(1),'%10.4 f '); d3=num2str(xc(2),'%10.4 f ');
d4=num2str(alpha,'%10.4 f ');
d5=num2str(norm(gc),'%10.4e');
```

```
fprintf(0%s  \t [%s  %s] \t %s \t %s \t
%s\n',d1,d2,d3,d4,d6);
while i>=0,%  Iterate until stopping criteria
breaks the loop.
gc=subs(g,S,xc);
if norm(gc) <= 0.0001,break;  end
dc=-Hc*gc';% Decide which line search to use
alpha=linesearch(g,xc,dc);
%al-pha = a; Fixed  Step-Size  or al-pha=blinesearch(xc,dc,alpham,ro)  xn=xc+alpha*dc;
% Calculate new x
i=i+1;

gn=subs(g,S,xn);dx=alpha*dc; % Calculate Difference  Parameters for Hc
dg=gn-gc;
if option == 1, % DFP method
Hc=Hc+((dx*dx.')/(dx.'*dg))-
(((Hc*dg)*(Hc*dg).')/(dg.'*Hc*dg));
else if option == 2,%  BFGS  method
Hc=Hc+((1+((dg.'* Hc*dg)/(dg.'*dx)))*((dx*dx.')/(dx.'*dg)))-((Hc*dg*dx.'+(Hc*dg*dx.').')/(dx.'*dg));
else
display('Improper Variant  Selection nn') ;
end
xc=xn;
d1=num2str(i,0%10.4f0);
d2=num2str(xc(1),0%10.4f0);d3=num2str(xc(2),0%10.4f0);  d4=num2str(alpha,0%10.4f0);
d5=num2str(norm(gn),0%10.4e0);
fprintf(0    %s\t
%s\n',d1,d2,d3,d4,d5);
if i>100,  break;
end
end
```

Appendix C: Backtracking Line Search Matlab Code.

```
f u n c t i o n

% alpham =1 ; ro = 0 .33 , or choose ro number between 0 and 1 , alpham > 0 . alpha=alpham ;
c =0 . 0001 ;
f k = f(xk ) ;
% f o r  i n s t a n c e  savef =1/2*( x1^3−x2  −1)^2+1/2*(  x1^2−x2  ) ^ 2  as f .m f i l e .
gx0=grad  ( xk )  d ' ;
%grad =[3  x1  ^2 * ( x1^3-x2  ) ,
x c c =xk+alpha  *d ;
c f k = f ( x c c ) ;
while  ( cfk > f k +c  alpha  gx0 )
%while  s u f f i c i e n t  d e c r e a s e  c o n d i t i o n
alpha=ro  alpha ;
x c c =xk+alpha  d ;
c f k = f ( x c c ) ;
end
```