Available online at www.cuijca.com

INTERNATIONAL JOURNAL OF COMPUTATIONAL ANALYSIS
Vol (3), No. (1), pp. 41-48
DOI: 10.33959/cuijca.v3i1.12

CITY UNIVERSITY INTERNATIONAL JOURNAL OF COMPUTATIONAL ANALYSIS

QUASI NEWTON MULTI STEP n/4 SKIPPING TECHNIQUE FOR OPTIMIZATION OF UNCONSTRAINED NONLINEAR PROBLEMS

Nudrat Aamir, Tazkia Anwar, Ayesha Nawaz, Aneesa Zeb, Lubna Zeb

Department of Electrical Engineering, City University of Science and Information Technology, Peshawar

Keywords:

Hessian

Lagrangian polynomial

A B S T R A C T

Ford and Moghrabi [2] introduced the multi-step quasi-Newton method for optimization, Where Hessian matrix is updated for each iteration, which outperformed the single step method. Jhon and Nudrat [4] introduced a multi-step skipping technique with and without modified search direction,, which shows a better experimental results. In Ford and Moghrabi, papers series, they introduced some techniques (Accumulative and Fixed-point approaches) to describe the parametric values needed to figure out the distances between many sets of iterations being involved in the recent interpolation curve. In this paper, we extended the existing three-step Fixed-point skipping approach and examine the sensitivity of this technique. Evidently, experimental result provides, positive improvement with comparison to standard Broyden-Fletcher-Goldfrab-Shanno (BFGS) method. Result shows comparatively, three-step Fixed-point n/4 skipping technique computational saves time than existing single-step BFGS method.

1. Introduction

Ford and Mograbi introduced multi-step quasi-newton methods for optimization, where current approximate Hessian is updated by utilizing data from more than one step. Where they present the idea, that employing interpolating curves, we can construct such methods. The standard quasi-newton method is also used in these methods, but in secant equation new vectors (r, w) are utilize instead of the normally used vectors (s, y). Where the quasi-Newton condition is defined as:

$$B_{k+1}s_k = y_k, \quad (1)$$

(where the Hessian approximation B_{k+1} is needed to fulfill the above condition), is replaced by a similar condition,

$$B_{k+1}r_k = w_k , \quad (2)$$

where the vectors r_k and w_k are derived from the multi-step method under discussion.

In single-step quasi-Newton methods, the new approximation B_{k+1} in single-step quasi-Newton methods, is essentially required to fulfill the requirement of quasi-Newton equation eq (1), which is an appropriate estimation to the equation of Newton method [1], [3]:

* Corresponding author: E-mail address: nudrathafeez@yahoo.com

$G(x_{k+1})\frac{dx(\tau^*)}{d\tau} = \frac{dg(x(\tau^*))}{d\tau}$, (3)

Where the relation defined in above equation eq (3) is a clearly straight forward utilization of the Chain Rule application for the vector function $g(x(\tau))$: Where

$x(\tau)$ is restricted to satisfy $x(\tau^*) = x_{i+1}$ and is defined to be any differentiable curve (indicate by X) in R^n: the quasi-Newton equation eq(1) can be derived from

the Newton equation eq (3). If $(x(\tau)$) is assumed to be defined by,

$x(\tau) = x_k + \tau s_k$ (4)

From eq(4), we get,

$x(0) = x_k$

$x(1) = x_{k+1}$

And $\frac{dx(\tau)}{d\tau} = s_i, \quad \forall \qquad \tau,$

Using interpolatory linear polynomial, we can estimate the curve $g(x(\tau))$ such that,

$g\big(x(0)\big) + \tau\big[g\big(x(1)\big) - g\big(x(0)\big)\big],$

then $g(x(\tau^*))$ the derivative of $g(x(\tau))$ at $\tau^* = 1$ can be approximated by

$\frac{dg(x(1))}{d\tau} \approx g\big(x(1)\big) - g\big(x(0)\big)$

$= g(x_{k+1}) - g(x_k)$

$= y_k,$

thus eq (3), becomes

$M(x_{k+1})s_k \approx y_k.$ (5)

The approximation B_{k+1} to the Hessian $M(x_{k+1})$ In the standard quasi-Newton method, is expected to fulfill the relation defined in eq(5) as an equality:

$B_{k+1}s_k = y_k$ (6)

Now we briefly discuss the concept of multi-step quasi-Newton methods ($m > 1$), in which the information is used from the m most frequently utilized steps, where

the path X is defined as the polynomial $x(\tau)$ of degree m interpolating the known point

$x(\tau_i) = x_{k-m+i+1}, \quad for\ i = 0, 1, 2, \dots, m.$ (7)

Ford [2] found, that it is more comfortable to use the Lagrangian form of the interpolating curve X

$x(\tau) = \sum_{i=0}^{m} L_i(\tau_m)x_{k-m+1}$ (8)

where $L_k(\tau)$ presenting the Lagrangian polynomial of degree m and is defined as

$L_i(\tau) = \prod_{j=0, j\neq i}^{m} \frac{\tau-\tau_j}{\tau_i-\tau_j}$ (9)

If the gradient g is prescribed to the interpolating curve X then by the comparatively interpolatory polynomial, it can be approximate

$g(x(\tau)) \approx \sum_{i=0}^{m} L_i(\tau)g(x_{k-m+i+1})$ (10)

In order to execute the Newton equation eq (3) with $\tau^* = \tau^m$, we use the relation defined in eq (7) to find $\frac{dg(x(\tau_m))}{d\tau}$ and eq (9) to guess $\frac{dg(x(\tau_m))}{d\tau}$:

By these values in eq (3) we then found the condition defined in eq (1) expected to be fulfill by the new Hessian approximation B_{k+1}, where

$\frac{dx(\tau_m)}{d\tau} = \sum_{i=0}^{m} L_i'(\tau_m)x_{k-m+1}$

$\cong r_k$ (11)

$\frac{dg(x(\tau_m))}{d\tau} \approx \sum_{i=0}^{m} L_i'(\tau)g(x_{k-m+i+1})$

$\cong w_k$ (12)

$$L_i'(\tau_m) = (\tau_i - \tau_m)^{-1} \prod_{j=0, j\neq i}^{m} \frac{\tau-\tau_j}{\tau_i-\tau_j},\ (i \neq m) \quad (13)$$

$$L_i'(\tau_m) = \sum_{j=0}^{m-1} (\tau_i - \tau_m)^{-1} \quad (14)$$

If we clarify eq (12) and eq (13) we can substitute r_k and w_k in charge of the most fresh employed step vectors $\{s_{k-j}\}_{j=0}^{m-1}$ and $\{y_{k-j}\}_{j=0}^{m-1}$:

$$r_k = \sum_{j=0}^{m-1} s_{k-j} \sum_{i=m-j}^{m} L_i'(\tau_m) \quad (15)$$

$$w_k = \sum_{j=0}^{m-1} y_{k-j} \sum_{i=m-j}^{m} L_i'(\tau_m) \quad (16)$$

The new Hessian approximation B_{k+1}, in multi-step procedure can be derived by utilizing any BFGS formula [2], where the vectors s_k and y_k are recouped by r_k and w_k, consequently:

$$B_{k+1} = B_k - \frac{B_k r_k r_k^T B_k}{r_k^T B_k r_k} + \frac{w_k w_k^T}{r_k^T w_k}. \quad (17)$$

correspondingly, to update the reverse Hessian approximation H_k the BFGS-type method is given by

$$H_{k+1} = H_k + \left(1 + \frac{w_k^T H_k w_k}{r_k^T} w_k\right) \frac{r_k r_k^T}{r_k^T w_k} - \frac{(r_k w_k^T H_k + H_k w_k r_k^T)}{r_k^T w_k} \quad (18)$$

and from the updated inverse Hessian approximation H_{k+1} will be expected to fulfill the following condition

$$w_k = r_k \quad (19)$$

The updating conditions for two-step and three-step methods are given as:

2. Multi-step methods

2.1. Two-Step

$$r_k = s_k - \frac{\delta^2}{(2\delta+1)} s_{k-1}\ , \quad (20)$$

$$w_k = y_k - \frac{\delta^2}{(2\delta+1)} y_{k-1}\ , \quad (21)$$

Where,

$$\delta = \frac{(\tau_2-\tau_1)}{(\tau_1-\tau_0)}.$$

2.2 Three-Step

$$r_k = s_k + \left(\frac{-\delta_1^2(\delta_2+1)^3}{(\delta_1-\delta_2)(3\delta_1\delta_2+\delta_1+\delta_2)} + 1\right) s_{k-1} + \left(\frac{(\delta_1\delta_2)^2}{(3\delta_1\delta_2+\delta_1+\delta_2)}\right) s_{k-2}, \quad (22)$$

$$w = y_k + \left(\frac{-\delta_1^2(\delta_2+1)^3}{(\delta_1-\delta_2)(3\delta_1\delta_2+\delta_1+\delta_2)} + 1\right) y_{k-1} + \left(\frac{(\delta_1\delta_2)^2}{(3\delta_1\delta_2+\delta_1+\delta_2)}\right) y_{k-2}, \quad (23)$$

$$B_{k+1} r_k = w_k$$

Where,

$$\delta_1 = \frac{(\tau_3-\tau_1)}{(\tau_1-\tau_0)} \quad \text{and} \quad \delta_2 = \frac{(\tau_3-\tau_2)}{(\tau_2-\tau_0)}$$

3. Unit-Spaced Method

We can determine (without failure of generality) the values $\{\tau_i\}_{i=0}^{m}$ as $\tau_0 = 0$ and $\tau_1 = 1$ in single-step methods ($m = 1$): From correlation, in multi-step methods, the simplest preferred choice for the needed values of $\{\tau_i\}_{i=0}^{m}$ is a unit-spacing between the τ –values

$$\tau_i = i - m + 1, \quad for\ i = 0, 1, 2, \dots, m\ (say) \quad (24)$$

From eq (21) and eq (22) the vectors r_k and w_k are presented as,

$r_k = s_k,\ w_k = y_k\ , B_{k+1} r_k = w_k\ .$

3.1. Unit-Spaced Two-Step Method

In this method, where $m = 2$, the data utilizes from the current last three iterates $x_{k-1};\ x_k;\ x_{k+1}$ and analogous gradient

values. The values of $\{\tau_i\}_{i=0}^{2}$ are given as:
$\tau_0 = -1, \tau_1 = 0,\ \tau_2 = 1.\ \delta = 1$ and, from eq (19) and eq (20), the values of vectors r_k and w_k are given by
$r_k = s_k - \frac{1}{3}s_{k-1}$, $w_k = y_k - \frac{1}{3}y_{k-1}$, $B_{k+1}r_k = w_k$.

3.2. Unit-spaced Three-step Method

In this method, $m = 3$, and the information of data used is obtained from latest four iterates $x_{k-2};\ x_{k-1};\ x_k;\ x_{k+1}$ and their interrelated gradient values. The values of $\{\tau_i\}_{i=0}^{3}$
$\tau_0 = -2,\ \tau_1 = -1,\ \tau_2 = 0,\ \tau_3 = 1,$

Taking the values of $\delta_1 = 2$ and $\delta_2 = \frac{1}{2}$, then we have from the relation defined in eq (21) and eq (23), the values of vectors r_k and w_k are given by
$r_k = s_k - \frac{7}{11}s_{k-1} + \frac{2}{11}s_{k-2}$, $w_k = y_k - \frac{7}{11}y_{k-1} + \frac{2}{11}y_{k-2}$, $B_{k+1}r_k = w_k$.

4. Metric based approaches

In metric-based approaches the values of $\{\tau_i\}_{i=0}^{m}$ are derived by measuring the distances between the mostly fresh recent m iterates $\{x_{k-m+i+1}\}_{k=0}^{m}$. To figure out the distances between iterates $\{x_{k-m+i+1}\}_{k=0}^{m}$, through metric based procedure the matric $\emptyset_M$ is used. For example, to calculate the distance between iterates
x_{k+1} and x_k we calculate

$$\emptyset_M(x_{k+1}, x_k) = \left((x_{k+1} - x_k)^T M (x_{k+1} - x_k)\right)^{\frac{1}{2}}$$
$$= (s_k^T M s_k)^{\frac{1}{2}}$$

where M is a positive-definite matrix. Some of the choices considered for M are as follows.

4.1. The Identity Matrix I

The most simplest choice matrix for M is the identity matrix. Therefore, we can figure out the distance among the points z_2 and z_1 by

$$\emptyset_I(z_2, z_1) = \left((z_2 - z_1)^T (z_2 - z_1)\right)^{\frac{1}{2}}$$
$$= \|z_2 - z_1\| = \|s_k\|.$$

4.2. The Current Hessian Approximation $\mathbf{B_k}$

The choice $M = B_k$ look like a natural one. In this case,

$$\emptyset_{B_k}(z_2,\ z_1) = ((z_2 - z_1)^T B_k (z_2 - z_1))^{\frac{1}{2}} = (s_k^T B_k s_k)^{\frac{1}{2}}$$

5. The Fixed-Point Methods

In Fixed-point methods the metric $\emptyset_M$ is used to figure out the distance of every single iterative point from one particular selected iterate (consider as the fixed point, which thereby becomes the origin for values of). In this approach, the latest iterate x_{i+1} is taken to be the fixed-point giving, $(\tau_m = 0)$ and we then compute the other parametric values τ_i by measuring precisely the distance between $x_{k+j-m+1}$ and x_{i+1} :

$$\tau_j = -\emptyset_M\left(x_{k+1},\ x_{k+j-m+1}\right),\quad for\ j = 0, 1, 2, \dots, m.$$

Now, we will examine Three-step methods of fixed-point approach:

5.1. Three-Step Fixed-Point Method

To figure out the values of τ , we fix one of the iterative point as origin say x_{i+1}, This iterative point corresponds to τ_3 so we consider τ_3 as origin, such that $\tau_3 = 0$, and calculate the values of τ_2, τ_1 and τ_0 by measuring the distance from x_{i+1} to x_i and

x_{i-1}, x_{i-2} directly. Distance is measured by using the above defined metric.

$$\emptyset_N(z_1, z_2) = \sqrt{(z_1 - z_2)^T N (z_1 - z_2)}$$

$$\forall\, z_1, z_2 \in R^n$$

Now consider algorithms for $M = I$ and $M = B_k$ (where B_k is a single step implicit update of B_{k-2}).
First we define some relations which will be used in the algorithms defined below.
We know that

$$s_{k-1} = t_{k-1} p_{k-1},$$

and

$$s_k = t_k p_k,$$

where,

$$p_{k-1} = -B_{k-1}^{-1} g_{k-1},$$

$p_k = -B_{k-1}^{-1} g_k$, (because we skipped the update of B_{k-1}).

Therefore, we obtain

$$s_k = -t_k\, B_{k-1}^{-1} g_k.$$

Using the above relations, we therefore find that

$$B_{k-1} s_k = -t_k g_k,$$

and

$$B_{k-1} s_{k-1} = -t_{k-1} g_{k-1}.$$

As we define above,

$$z_i^{(j)} = B_j s_i \qquad (i\ \&\ j = k, k-1, k-2):$$

For $j = k-2\ and\ i = k-2$;

$$z_{k-2}^{(k-2)} = B_{k-2} s_{k-2}$$

We know that

$$s_{k-2} = t_{k-2} p_{k-2}$$

and

$$p_{k-2} = -B_{k-2}^{-1} g_{k-2}$$

which gives us

$$z_{k-2}^{(k-2)} = -t_{k-2} g_{k-2} \tag{5.1.1}$$

Similarly (because of skipping)

$$z_{k-1}^{(k-2)} = -t_{k-1} g_{k-1} \tag{5.1.2}$$

and

$$z_k^{(k-2)} = -t_k g_k \tag{5.1.3}$$

We define $B_k = BFGS(B_{k-2}, s_{k-1}, y_{k-1})$ which gives us

$$z_{k-2}^{(k)} = B_k s_{k-2}$$

$$= B_{k-2} s_{k-2} - \frac{B_{k-2} s_{k-1} s_{k-1}^T B_{k-2} s_{k-2}}{s_{k-1}^T B_{k-2} s_{k-1}} + \frac{y_{k-1} y_{y-1}^T s_{k-2}}{s_{k-1}^T y_{k-1}}$$

$$= z_{k-2}^{(k-2)} - \frac{z_{k-2}^{(k-2)} s_{k-1}^T z_{k-2}^{(k-2)}}{s_{k-1}^T z_{k-1}^{(k-2)}} + \frac{y_{k-1} y_{y-1}^T s_{k-2}}{s_{k-1}^T y_{k-1}} \quad (5.1.4)$$

Using eq (5.1.1) and eq (5.1.2)
also,

$$z_k^{(k)} = B_k s_k$$

$$= B_{k-2} s_k - \frac{B_{k-2} s_{k-1} s_{k-1}^T B_{k-2} s_k}{s_{k-1}^T B_{k-2} s_{k-1}} + \frac{y_{k-1} y_{y-1}^T s_k}{s_{k-1}^T y_{k-1}}$$

$$= z_k^{(k-2)} - \frac{z_{k-2}^{(k-2)} (s_{k-1}^T z_k^{(k-2)})}{s_{k-1}^T z_{k-1}^{(k-2)}} + \frac{y_{k-1} (y_{y-1}^T s_k)}{s_{k-1}^T y_{k-1}} \quad (5.1.5)$$

utilizing eq (5.1.2) and eq (5.1.3)
also,

$$z_{k-1}^{(k)} = B_k s_{k-1}$$
$$= y_{k-1} \quad (5.1.6)$$

Now, we discuss Fixed-point Algorithms.

5.1.1 Algorithm F

For $= I$, let us consider,

$$\tau_3 = 0:$$

Then,

$$\tau_2 = -\emptyset_I(x_{k+1}, x_k) \ ,$$

$$= -\|s_k\|,$$

also,

$$\tau_1 = -\emptyset_I(x_{k+1}, x_k) \ ,$$

$$= -\|s_k + s_{k-1}\|,$$

and

$$\tau_0 = -\emptyset_I(x_{k+1}, x_{k-2}),$$

$$= -\|s_k + s_{k-1} + s_{k-2}\|.$$

5.1.2. Algorithm $\mathbf{F_{B_k}}$

Taking $M = B_k$, we have,

$$\tau_3 = 0:$$

Then,

$$\tau_2 = -\emptyset_{B_k}(x_{k+1}, x_k) = -\sqrt{s_k^T B_k s_k} = -\sqrt{s_k^T z_k^{(k)}},$$

using eq (7.1.5).
Also,

$$\tau_1 = -\emptyset_{B_k}(x_{k+1}, x_{k-1}) = -\sqrt{(s_k + s_{k-1})^T B_k (s_k + s_{k-1})} = -\sqrt{s_k^T z_k^{(k)} + 2 s_k^T z_{k-1}^{(k)} + s_{k-1}^T z_{k-1}^{(k)}},$$

Using eq (7.1.5) and eq (7.1.6). Now finally we calculate,

$$\tau_0 = -\emptyset_{B_k}(x_{k+1}, x_{k-2}) = -\sqrt{(s_k + s_{k-1} + s_{k-2})^T B_k (s_k + s_{k-1} + s_{k-2})} = -\sqrt{\begin{matrix} s_k^T z_k^{(k)} + 2 s_k^T z_{k-1}^{(k)} + s_{k-1}^T z_{k-1}^{(k)} + 2 s_k^T z_{k-2}^{(k)} \\ + 2 s_{k-1}^T z_{k-2}^{(k)} + s_{k-2}^T z_{k-2}^{(k)} \end{matrix}}$$

From eq(7.1.4), eq (7.1.5) and eq (7.1.6).

6. Numerical results

The proposed idea in this paper is tested on existing single step, existing unite-spaced and on existing Three-step fixed point method and then the Three-step fixed point method with the $\frac{n}{4}$ skipping and is compared with single step and unite spaced BFGS method. All the algorithms in the experiments will utilize the BFGS formula in computation to update the inverse Hessian approximation

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

But (in the case of multi-step methods) with the usual vectors s_k and y_k replaced by r_k and w_k:

$$H_{k+1} = H_{k-1} + \left(1 + \frac{w_k^T H_k w_k}{r_k^T w_k}\right)\frac{r_k r_k^T}{r_k^T w_k} - \left(\frac{r_k w_k^T H_k + H_k w_k r_k^T}{r_k^T w_k}\right)$$

Total 25 test functions are tested in the experiments with different dimensions rang start from 2 to 200. These were selected from standard problems expressed in the literature [5]. Four distinct starting points were employed, for each function, which gives a total of 100 test problems. All these test functions were classified into the following different subsets (where n is the dimension of the vector x):

(1) Low : $(2 \leq n \leq 20)$;
(2) Medium : $(21 \leq n \leq 60)$;
(3) High : $(61 \leq n \leq 200)$;
(4) Combined : $(2 \leq n \leq 200)$.

According to classification in total their are 3, 7 and 15 problems in low, Medium and high respectively. Therefore, in each subset their are 12, 28 and 60 test problems located respectively. All these functions are selected from J.J. Mor, B.S. Garbow and K.E. Hillstrom [5] and Toint [6].

Table 6.1. Problems and dimensions

Problems	Dimension
Extended Rosenbrock	2, 20, 26, 40, 60, 80, 100, 120
Merged Quadratic [6]	30, 50, 70, 110, 136, 180
Discrete integral equation	20, 84, 100, 150, 175, 200
Freudenstein and Roth [6]	28, 52, 85, 118, 190

In this paper we have discussed existing standard BFGS method and quasi-newton methods. The proposed idea, skipping $\frac{n}{4}$ updates was implemented on BFGS and three-step fixed-point technique. The proposed idea were tested on four different functions with dimensions ranging from 2 to 200, as discussed in table (Table 6.1.).

In this section the experimental results of three-step skipping technique and the non-skipping single-step BFGS methods for 'low', 'medium' and 'high' dimensions are discussed. Experimental results were obtained from MATlAB R2009b and Windows7 in Shaheed Benazir Bhutto Women University Peshawar lab. All the skipping techniques are discussed in comparison with single-step BFGS methods.

Table 6.2. Low dimensions

Methods	Evaluation	Iteration	Time(sec)	Failure
BFGS	2041	1673	0.172065	0
*BFGS**	2144	1522	0.180345	0
$[\frac{n}{4}]F_I^3$	3350	2087	0.230485	0
$[\frac{n}{4}]F_{B_k}^3$	2624	1694	0.213625	0

Table 6.3. Medium dimensions

Methods	Evaluation	Iteration	Time(sec)	Failure
BFGS	4700	4351	0.51805	0
*BFGS**	4493	3664	0.398265	0
$[\frac{n}{4}]F_I^3$	6008	4903	0.530355	0
$[\frac{n}{4}]F_{B_k}^3$	5166	4204	0.50995	0

Table 6.4. High dimensions

Methods	Evaluation	Iteration	Time(sec)	Failure
BFGS	8956	8486	3.678605	0
*BFGS**	8246	7010	2.102285	0
$[\frac{n}{4}]F_I^3$	10353	8986	2.819045	0
$[\frac{n}{4}]F_{B_k}^3$	9455	8199	2.66202	0

Table 6.5. Combined dimensions

Methods	Evaluation	Iteration	Time(sec)	Failure
BFGS	15697	14510	4.363715	0
*BFGS**	14883	12196	2.68127	0
$[\frac{n}{4}]F_I^3$	19711	15976	3.270165	0
$[\frac{n}{4}]F_{B_k}^3$	17245	14097	3.38557	0

Investigating the results, we have observed that BFGS* (single step method with n/4 skipping) performing much better in computational time saving, function evaluations and iterations for medium, high and combined dimensions . But for low dimensions the non-skipping single-step BFGS method shows progress on all skipping techniques BFGS*, $[\frac{n}{4}]F_I^{(3)}$ and $[\frac{n}{4}]F_{B_k}^{(3)}$. Since we know that the solution for low and medium dimension is easy to implement. Therefore, in this thesis we are emphasize on high and combined dimensions. Observing results, we conclude that the skipping techniques BFGS*, $[\frac{n}{4}]F_I^3$ and $[\frac{n}{4}]F_{B_k}^3$ working much better than non-skipping BFGS method. Skipping techniques saves computational time than non-skipping BFGS method. But unfortunately, the number of iterations and evaluations are increasing in $[\frac{n}{4}]F_I^3$ and $[\frac{n}{4}]F_{B_k}^3$.

7. Conclusion

In this paper we have investigated the standard BFGS method and three-step fixed-point technique with skipping idea i.e skipping $\frac{n}{4}$ updates. From above discussion we have concluded that the new skipping idea performs better than non-skipping idea in case of computational time, although the functions evaluations are increasing. We have also noticed that for low and medium dimensions single-step standard BFGS method outperform the three-step fixed-point techniques with proposed skipping idea. Since the solution for low and medium dimensions is easy to implement, therefore, in this paper we have focus on high and combined dimensions. Evidently, from results we have noticed that the skipping techniques succeeded in saving computational time for high and combined dimensions as compared to standard BFGS method.

REFERENCES

[1]. J. A. Ford, "On the use of additional function evaluations in quasi-newton methods", Department of Computer Science, University of Essex, 1986.

[2]. J. A. Ford and I. A. Moghrabi, "Multi-step quasi-newton methods for optimization", Journal of Computational and Applied Mathematics, 1994.

[3]. J. A. Ford and A. F. Saadallah, "A rational function model for unconstrained optimization", Colloquia Mathematica Societatis Janos Bolyai, 1986.

[4]. J. A. Ford and N. Aamir, "Multi-step skipping methods for unconstrained optimization", Numerical Analysis and Applied Mathematics Icnaam 2011.

[5]. J. J. Mor*é*, B. S. Garbow and K. E. Hillstrom," Testing unconstrained optimization software", ACM Transactions on Mathematical Software, 1981.

[6]. Ph. L. Toint, "On large scale nonlinear least squares calculations", SIAM Journal on Scientific and Statistical Computing, 1987), no. 3, 416-435.