British Journal of Mathematics & Computer Science
10(2): 1-13, 2015, Article no.BJMCS.18694
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

A Fourth-Order Nonlinear Conjugate Gradient Method in Large Scale Optimization

Nwaeze Emmanuel[1*]

[1]*Department of Maths/Comp. Sc./Stat./Info, Federal University Ndufu-Alike, Ikwo, Nigeria.*

Article Information

DOI: 10.9734/BJMCS/2015/18694
Editor(s):
(1) Sheng Zhang, Department of Mathematics, Bohai University, Jinzhou, China.
Reviewers:
(1) Ette Harrison Etuk, Rivers State University of Science and Technology, Nigeria.
(2) Ahmed Salhoumi, Université Hassan II, Morocco.
(3) Anonymous, Zaozhuang University, China.
(4) Anonymous, Hunan University of Humanities, China.
Complete Peer review History: http://sciencedomain.org/review-history/10076

Original Research Article

Received: 05 May 2015
Accepted: 13 June 2015
Published: 07 July 2015

Abstract

This paper is presenting a fourth-order nonlinear conjugate gradient method in large scale optimization. This method solves unconstrained optimization problems. It is based on a nonlinear polynomial approximation of the objective function. The idea is to approximate the minimizing function by Taylor series development using fourth-order terms. The algorithm is presented in steps and some properties of the gradients are proved, using classical results. Also, the convergence analysis has been proved under known assumptions. Some numerical results have been compared to existing data. The analysis of these results confirms that the new method is accurate, since the computed results are very close to the exact solutions.

Keywords: Fourth-order nonlinear conjugate gradient method; unconstrained optimization; objective function; nonlinear polynomial approximation; large scale optimization.

Mathematical subject classification (2010): 65K10.

1 Introduction

The unconstrained minimization of a smooth function, f, in many variables remains an important problem in optimization theory. Many scientists, in various fields of science and engineering, seek to solve this class of problems in real life applications. The general approach is to solve the zeros of the function gradient since the local minima occur at stationary points. In order to achieve fast global convergence, I develop and present a fourth-order nonlinear conjugate gradient method in large scale optimization. Consider the unconstrained optimization problem.

**Corresponding author: nwaezeema@yahoo.com;*

$$\min_{x\in\Re^n} f(x) \tag{1}$$

Where f is a differentiable function. In order to solve this problem, we need to design a special algorithm that reduces the high storage and computation cost of some computed matrices [1]. Various types of conjugate gradient method have been used to solve large scale unconstrained minimization problems [2]. Usually, a function F is constructed to approximate f. If the objective function is not quadratic or the inexact line search is used, some of the conjugate gradient methods fail to converge globally [3,4]. The process of minimizing a non-quadratic objective function through the conjugate gradient method is called the nonlinear conjugate gradient method [5,6]. Many scholars have published their findings on this method as shown in ref. [7-9]. New algorithms on nonlinear conjugate gradient method are available in ref. [10-14]. Every conjugate gradient method is an iterative scheme of the form

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, ... \tag{2}$$

where x_0 is an initial point, α_k is a step size and the search direction is

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_k d_k, & k \geq 1. \end{cases} \tag{3}$$

$g_k = \nabla f(x_k)$ and β_k specifies the choice of conjugate gradient method [15]. It could take any of the following forms.

$$\beta^{FR} = \frac{\| g_k \|^2}{\| g_{k-1} \|^2} \text{ [4]}, \beta^{PRP} = \frac{g_k{}^T (g_k - g_{k-1})}{\| g_{k-1} \|^2} \text{ [16]}, \beta^{HS} = \frac{g_k{}^T (g_k - g_{k-1})}{d_{k-1}{}^T (g_k - g_{k-1})} \text{ [17,18]},$$

$$\beta^{CD} = -\frac{\| g_k \|^2}{d_{k-1}{}^T g_{k-1}} \text{ [14]}, \beta^{DY} = \frac{\| g_k \|^2}{d_{k-1}{}^T (g_k - g_{k-1})} \text{ [12] and } \beta^{LS} = -\frac{g_k{}^T (g_k - g_{k-1})}{d_{k-1}{}^T g_{k-1}} \text{ [19]}$$

where

FR: Fletcher Reeves, PRP: Polak-Ribiére-Polyak, HS: Hestenes-Stiefiel, DY: Dai Yuan and LS: Liu-Storey.

Many of these conjugate gradient methods use inexact line search technique [20]. Others use exact line search approach [21]. Stoer et al. [22] studied the conjugate gradient method on a subspace and obtained a variant of the method with an inexact line search approach. The search for a reliable and accurate scheme motivated this work on a fourth-order nonlinear conjugate gradient method (FONCGM) in large scale optimization. FONCGM is presented in six sections. Sections (two and three) discuss the fourth-order nonlinear conjugate gradient method and its convergence analysis respectively. Section four presents some test problems. Section five explains the numerical results while section six ends this work with a conclusion.

2 The Fourth-Order Nonlinear Conjugate Gradient Method (FONCGM)

The fourth-order nonlinear conjugate gradient method is based on Taylor series representation of f by F. This representation is expected to be a better approximation of f than the usual representation. The following is the representation of F at point x_k.

$$F(x) = f(x_k) + df(x_k) + \frac{1}{2!}d^2 f(x_k) + \frac{1}{3!}d^3 f(x_k) + \frac{1}{4!}d^4 f(x_k), \tag{4}$$

where

$$d^{\,n} f(x_k) = \sum_{i_1}^{N}\sum_{i_2}^{N}\ldots\sum_{i_n}^{N} h_{i_1}h_{i_2}\ldots h_{i_n}\frac{\partial^n f(x_k)}{\partial x_{i_1}\partial x_{i_2}\ldots\partial x_{i_N}},\ x, x_k, h_{i_j} \in \Re^N; h_{i_j} = x - x_k,\ 2 \le n \le 4.$$

Using a vector $h = x - x_k$ and $A_i = \nabla^i f(x_k)$, in equation (4), I have

$$\begin{aligned} F(x) &= f(x_k) + h^T A_1 + \frac{1}{2!}h^T A_2 h + \frac{1}{3!}h^T\left(h^T A_3 h\right) + \frac{1}{4!}h^T\left(h^T A_4 h\right)h \\ &= f(x_k) + h^T A_1 + \frac{1}{2!}h^T\left[A_2 + \left\{\frac{2}{3!}A_3 + \frac{2}{4!}A_4 h\right\}h\right]h. \end{aligned} \tag{5}$$

Using tensor notations presented in [23], I have

$$\begin{aligned} \left[\frac{2}{3!}A_3 + \frac{2}{4!}A_4 h\right]h &= \left[\sum_{j=3}^{4}\frac{2}{j!}A_j\prod_{p=1}^{j-3}h^{Z_p}\right]h \\ &= \left[\sum_{j=3}^{4}\frac{2}{j!}\sum_{m=0}^{j-3}(-1)^m\binom{j-3}{m}g\left(x_k + \{(j-3)-m\}h\right)\right]^T h \\ &= \frac{1}{12}\left[g(x) + 3g(x_k)\right]^T h \end{aligned} \tag{6}$$

where $g(x)$ denotes the gradient of f, at point x, $\prod_{p=1}^{2}h^{Z_p} = h^T h$, $\binom{n}{m} = \frac{n!}{(n-m)!\,m!}$, Z_p represents $T^{\frac{1}{2}\left[1+(-1)^p\right]}$ and T denotes transpose. It follows that

$$F(x) = f(x_k) + h^T A_1 + \frac{1}{2}h^T H(x)h \tag{7}$$

Where

$$H(x) = A_2 + \frac{1}{12}\left[g(x) + 3g(x_k)\right]h^T.$$

Similarly,

$$\begin{aligned} \nabla F(x) &= A_1 + A_2 h^T + \frac{1}{2}h^T A_3 h + \frac{1}{3!}h^T(h^T A_4 h) \\ &= A_1 + \sum_{j=2}^{4}\frac{1}{(j-1)!}A_j\prod_{p=1}^{j-1}h^{Z_p} \\ &= A_1 + \sum_{j=2}^{4}\frac{1}{(j-1)!}\sum_{m=0}^{j-1}(-1)^m\binom{j-1}{m}g\left[x_k + \{(j-1)-m\}h\right] \\ &= A_1 + \frac{1}{6}\left[g(x_k + 3h) + 3g(x_k + h) - 4g(x_k)\right] \\ &= A_1 + \frac{1}{6}\left[g(3x - 2x_k) + 3g(x) - 4g(x_k)\right] \end{aligned} \tag{8}$$

$$\nabla F(x_{k+1}) = A_1 + \frac{1}{6}\left[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) - 4g(x_k)\right] \tag{9}$$

$$\nabla F(x_k) = A_1 = g(x_k). \tag{10}$$

$$\begin{aligned}\nabla F(x_{k+1}) &= \nabla F(x_k) + \frac{1}{6}\left[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) - 4g(x_k)\right] \\ &= \frac{1}{6}\left[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) - 4g(x_k) + 6g(x_k)\right] \\ &= \frac{1}{6}\left[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) + 2g(x_k)\right].\end{aligned} \tag{11}$$

Using $G_{k+1} = \nabla F(x_{k+1})$, we present a fourth-order nonlinear conjugate gradient algorithm in which the directions of search, $D_0, D_1, ..., D_k$ are H conjugate. That is,

$$D_{k+1}^T H_{k+1} D_k = 0 \tag{12}$$

From the classical results, it follows that

$$G_{k+1} = G_k + \alpha_k H_{k+1} D_k \tag{13}$$

With a given x_0, α_k is computed such that

$$F(x_k + \alpha_k D_k) < F(x_k), \tag{14}$$

$$D_{k+1} = \begin{cases} -G_{k+1}, & k = 0 \\ -G_{k+1} + \beta_k D_k, & k > 0 \end{cases} \tag{15}$$

and

$$x_{k+1} = x_k + \alpha_k D_k, k = 0, 1, ...$$

From equations (12) and (15),

$\left[-G_{k+1} + \beta_k D_k\right]^T H_{k+1} D_k = 0$; $\beta_k D_k^T H_{k+1} D_k = G_{k+1}^T H_{k+1} D_k$. From equation (13),

$$\beta_k = \frac{G_{k+1}^T (G_{k+1} - G_k)}{D_k^T (G_{k+1} - G_k)} \tag{16}$$

The algorithm is described below.

Algorithm 1. (FONCGM)

Step 1: Select $x_0 \in \Re^N$, $N \geq 2$, $\|.\|$ is Euclidean norm and $\varepsilon > 0$ (a small number: 0.000001). Set $D_0 = -G_0 = -\nabla F(x_0)$ and $k = 0$.

Step 2: If $\| G_k \| \leq \varepsilon,$ stop. Choose x_k, otherwise go to step 3.

Step 3: Compute α_k such that $F(x_k + \alpha_k D_k) < F(x_k)$ and go to step 4.

Step 4: Compute

$$G_{k+1} = \frac{1}{6}\left[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) + 2g(x_k)\right] \quad \beta_k = \frac{G_{k+1}^T(G_{k+1} - G_k)}{G_K^T(G_{k+1} - G_k)}, \quad D_{k+1} = -G_k + \beta_k D_k.$$

$$x_{k+1} = x_k + \alpha_k D_k.$$

Go to step 5.

Step 5: Set $k = k + 1.$ Go to step 2.

Remark: Dai and Yuan [12] presented a nonlinear conjugate gradient algorithm for solving unconstrained optimization problems. Below is Dai-Yuan's algorithm for problem (1).

Algorithm 2. (Nonlinear conjugate gradient method)

Step 1: Select $x_0 \in \Re^N$, $N \geq 2$ and $\varepsilon > 0$. Set $d_0 = -g_0$ and $k = 0.$

Step 2: If $\| g_k \| \leq \varepsilon,$, stop. Take x_k. Otherwise go to step 3.

Step 3: Compute α_k such that $f(x_k + \alpha_k d_k) < f(x_k)$, go to step 4.

Step 4: Compute $\beta_k = \frac{\| g_k \|^2}{d_{k-1}{}^T(g_k - g_{k-1})}$, $d_{k+1} = -g_k + \beta_k d_k$ *and* $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Set $k = k + 1.$ Go to step 2.

3 Convergence Analysis

I employ the convergence results of algorithm (2), as contained in the following lemma and theorem, to establish the convergence of algorithm (1). Also, I assume that the objective function satisfies the following conditions.

3.1 Assumptions

i. f is bounded below in $\Re^N$ and is four times continuously differentiable in a neighborhood Z of the level set

$$L = \left\{x \in \Re^N : f(x) \leq f(x_0)\right\}$$

a.

ii. The gradient, $g(x)$, is Lipschitz continuous in Z, namely, there exists a constant $Lc > 0$ such that

$$\| \nabla f(x) - \nabla f(y) \| \leq Lc \| x - y \|, \quad x, y \in Z.$$

a.

iii. The extended hessian matrix $H(x)$ is positive definite.

3.2 Lemma

i. Suppose that x_0 is a starting point for which the above assumptions are satisfied. Consider any method of the form (2), where D_k, a vector, is the descent direction and α_k satisfies the standard Wolfe conditions [18], then

$$\sum_{k\geq 0}\frac{\left(G_k^T D_k\right)^2}{\| D_k \|^2} < \infty$$

ii. Suppose that x_0 is a starting point for which the above assumptions are satisfied. Let

iii. $\{x_k, k = 1, 2, ...\}$ be generated by algorithm (1). Then, the algorithm either terminates at a stationary point or converges in the sense that

$$\liminf_{k\to\infty} \| G(x_k) \| = 0$$

iv.

Theorem 1. Suppose that f is continuously differentiable, bounded below and the norm of the Hessian matrix is bounded. The iteration, $\{x_k\}$, generated by algorithm (2) satisfies $x_k \to x^*$ as $k \to \infty$ and the Hessian matrix of f is positive definite. Let ε_k be the relative error in the truncated conjugate gradient method and the algorithm. If $\varepsilon_k \to \infty$ then $\{x_k\}$ converges, that is,

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

Proof (Lemma (i)):

Dai and Yuan proved this lemma for algorithm (2): $\sum_{k\geq 0}\frac{\left(g_k^T d_k\right)^2}{\| d_k \|^2} < \infty$. A Similar proof is presented for algorithm (1), since $G_k = G(x_k) = \nabla F(x_k) = g(x_k))$ and the search directions d_k and D_k have same definition. This is obvious on using the assumptions of lemma (i), Dai and Yuan's proof and k in place of $k+1$ in equation (11).

$$\frac{\left(G_k^T D_k\right)^2}{\| D_k \|^2} = \frac{\frac{1}{36}\left\{\left[g(3x_k - 2x_{k-1}) + 3g(x_k) + 2g(x_{k-1})\right]^T D_k\right\}^2}{\| D_k \|^2}$$

$$= \frac{\frac{1}{36}\left\{\left[g(3x_k - 2x_{k-1}) + 3g(x_k) + 2g(x_{k-1})\right]^T d_k\right\}^2}{\| d_k \|^2}, \; d_k = D_k$$

$$= \frac{\frac{1}{36}\left\{\left(g(3x_k - 2x_{k-1})^T d_k\right) + 3\left(g(x_k)^T d_k\right)\right\}^2}{\| d_k \|^2}, \; g(x_{k-1})^T d_k = 0 \; (by \; choice \; of \; \alpha_k)$$

$$\leq \frac{\frac{1}{36}\left\{\left(g(3x_k - 2x_{k-1})^T d_k\right)^2 + 9\left(g(x_k)^T d_k\right)^2 + 6\left(g(3x_k - 2x_{k-1})^T d_k\right)^2 + 6\left(g(x_k)^T d_k\right)^2\right\}}{\| d_k \|^2}$$

$$= \frac{1}{36}\left\{\frac{\left(g(3x_k - 2x_{k-1})^T d_k\right)^2}{\| d_k \|^2} + 9\frac{\left(g(x_k)^T d_k\right)^2}{\| d_k \|^2} + 6\frac{\left(g(3x_k - 2x_{k-1})^T d_k\right)^2}{\| d_k \|^2} + 6\frac{\left(g(x_k)^T d_k\right)^2}{\| d_k \|^2}\right\}$$

$$\sum_{k\geq 0}\frac{\left(G_k^T D_k\right)^2}{\| D_k \|^2} \leq \frac{1}{36}\left\{\sum_{k\geq 0}\frac{\left(g(3x_k - 2x_{k-1})^T d_k\right)^2}{\| d_k \|^2} + 9\sum_{k\geq 0}\frac{\left(g(x_k)^T d_k\right)^2}{\| d_k \|^2} + 6\sum_{k\geq 0}\frac{\left(g(3x_k - 2x_{k-1})^T d_k\right)^2}{\| d_k \|^2} + 6\sum_{k\geq 0}\frac{\left(g(x_k)^T d_k\right)^2}{\| d_k \|^2}\right\} < \infty$$

Thus,

$\sum_{k\geq 0}\frac{\left(G_k^T D_k\right)^2}{\| D_k \|^2} < \infty$ as required.

Proof (Lemma (ii)):

Dai and Yuan proved this lemma for algorithm (2). He proved that $\liminf_{k\to\infty} \| g(x_k) \| = 0.$ It follows that $\liminf_{k\to\infty} \| g(3x_k - 2x_{k-1}) \| = 0$ and $\liminf_{k\to\infty} \| g(x_{k-1}) \| = 0$ at points of convergence of algorithm (2).

Using the assumptions of lemma (ii), equation (10) ($G_k = G(x_k) = \nabla F(x_k) = g(x_k)$) and Dai's proof, it follows that

$\liminf_{k\to\infty} \| G_k \| = \lim_{k\to\infty} \| g_k \| = 0.$ Alternatively, using k in place of $k+1$ in equation (11) yields

$$\liminf_{k\to\infty} \| G_k \| = \liminf_{k\to\infty} \frac{1}{6} \| \{g(3x_k - 2x_{k-1}) + 3g(x_k) + 2g(x_{k-1})\} \|$$
$$\leq \liminf_{k\to\infty} \frac{1}{6} \| g(3x_k - 2x_{k-1}) \| + \liminf_{k\to\infty} \frac{1}{6} \| 3g(x_k) \| + \liminf_{k\to\infty} \frac{1}{6} \| 2g(x_{k-1}) \| = 0.$$

$\liminf_{k\to\infty} \| G_k \| \leq 0$ and $\| G_k \| \geq 0$ implies that $\liminf_{k\to\infty} \| G_k \| = 0.$

Proof of theorem (1): The proof is available in many literatures. Noting that

$G_k = G(x_k) = \nabla F(x_k) = g(x_k),$ the proof is same since the assumptions on algorithm (1) meet the requirements of this theorem. Using $M \geq m \geq 0,$ $\varepsilon_k = x_k - x^*$ and the results from NMC [24], I have

$$\| x_{k+1} - x^* \|^2 = \left(x_{k+1} - x^*\right)^T \left(x_{k+1} - x^*\right)$$
$$= \left(x_{k+1} - x^*\right)^T \left(x_k + \alpha_k d_k - x^*\right)$$
$$\leq \frac{M}{n}\left(\frac{1-R}{1+R}\right)^{2(k+1)} \| x_k - x^* \|^2 \leq \frac{M}{n}\left(\frac{1-R}{1+R}\right)^{2(k+1)} \| x_0 - x^* \|^2; \; R = \frac{m}{M}.$$
$$\| x_{k+1} - x^* \| \leq \sqrt{\frac{M}{n}}\left(\frac{1-R}{1+R}\right)^{(k+1)} \| x_k - x^* \|$$

$\lim_{k\to\infty} \frac{\| x_{k+1} - x^* \|}{\| x_k - x^* \|} = 0,$ since $\lim_{k\to\infty} \sqrt{\frac{M}{n}}\left(\frac{1-R}{1+R}\right)^{(k+1)} = 0.$

4 Numerical Considerations

The motivation for this numerical test is to demonstrate the numerical performance of the new method comparatively with some existing classical methods. The following problems were solved by implementing algorithm (1) through MATLAB 7.3 codes.

Problem 1. Penalty function I (problem (1) in [25]

$$f(x)=\sum_{i=1}^{n}10^{-5}\left(x_i-1\right)^2+\left[\left(\sum_{i=1}^{n}x_i^2\right)-\frac{1}{4}\right]^2,\quad [x_0]_i=i.$$

Problem 2. Variable dimensioned function ([25,26]).

$$f(x)=\sum_{i=1}^{n}\left(x_i-1\right)^2+\left[\sum_{i=1}^{n}i(x_i-1)\right]^2+\left[\sum_{i=1}^{n}i(x_i-1)\right]^4,\quad [x_0]_i=1-\frac{i}{n}.$$

Problem 3. Trigonometric function ([25,26])

$$f(x)=\sum_{i=1}^{n}\left[n-\sum_{j=1}^{n}\cos(x_j)+i(1-\cos(x_i))-\sin(x_i)\right]^2,\quad [x_0]_i=1/n.$$

Problem 4. A penalty function [25]

$$f(x)=1+\sum_{i=1}^{n}x_i+10^3\left(1-\sum_{i=1}^{n}1/x_i\right)^2+10^3(\left(1-\sum_{i=1}^{n}i/x_i\right)^2,\quad [x_0]_i=1.$$

Problem 5. Extended Rosenbrock function [25]

$$f(x)=\sum_{i=1}^{n}\left[100\left(x_{2i}-x_{2i-1}^2\right)^2+\left(1-x_{2i-1}\right)^2\right]\quad [x_0]_{2i-1}=-1.2,[x_0]_{2i}=1.$$

Problem 6. Penalty function II (modification of problem [25]

$$f(x)=\sum_{i=1}^{n}\left(\exp(x(i))-x(i)\right),\ [x_0]_i=1.$$

Problem 7. Linear function-rank 1 ([25] with new initial values)

$$f(x)=\sum_{i=1}^{m}\left[i\left(\sum_{j=1}^{n}jx_j\right)-1\right]^2,\ m\geq n,\ [x_0]_i=\frac{1}{i}.$$

The numerical results obtained for the new method vis-a-vis some classical methods (FOCGM, PRP, FR, DY and HS) are presented in Table 1. The stopping criterion is $\| g(x_k)\|<0.000001$ while the maximum number of iterations is 1000.

Table 1. Number of iterations and CPU time in seconds

Problem	N	FONCGM	PRP	FR	DY	HS
1	2000	214/91.6	27/2.6	61/9.4	61/9.4	27/4.3
	1000	142/20.1	24/0.7	63/3.4	64/3.4	24/1.4
2	2000	214/171.2	5/0.97	6/2.2	5/1.93	5/1.9
	1000	100/27.1	4/0.8	9/1.1	8/0.96	3/0.5
3	2000	64/76.7	1000/358.3	1000/405.8	452/202.8	83/73.9
	1000	52/16.2	59/6.6	386/42	379/55.1	58/14.4
4	2000	43/36.2	Fail	Fail	Fail	Fail
	1000	41/11.9	Fail	Fail	Fail	Fail
5	2000	441/229.5	57/6.22	33/7.4	33/7.4	57/12.4
	1000	98/12.7	91/3.34	33/2.5	33/2.5	80/5.8
6	2000	9/5.6	1000/118.6	1000/238.9	Fail	1000/237.1
	1000	12/2.8	1000/38.1	1000/77.1	Fail	1000/76.2
7	2000	6/13.3	2/0.96	5/3.2	9/4.9	4/2.8
	1000	6/5.1	15/1.07	4/0.79	8/3.1	7/1.2

Performance profiles have been introduced by Dolan and More' [27]. The main idea is to show, graphically, the relative performance of various solvers on a given set of problems. That is, the curves are used to compare the efficiency of a set S of solvers on a set P of test problems. $t_{p,s}$ denotes the performance of a solver s (based on the number of iterations, function evaluations, gradient evaluations or execution time) on the problem p. $r_{p,s}$ denotes the relative performance of a solver s on a problem p and $r_{p,s} = \dfrac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}$. Our assumption is that $r_{p,s} \leq w$, $w \in \Re$, for all solvers s on the problems p. $r_{p,s} = w$ if solver s cannot solve problem p. The performance profile of the solver s is the function

$y_s : [1, w] \to [0, 1]$ such that

$$y_s(t) = \frac{n(\{p \in P : r_{p,s} \leq t\})}{n(P)}$$

or

$$y_s(t) = \frac{n(\{p \in P : \log_2(r_{p,s}) \leq t\})}{n(P)}. \; n(.)$$

Denotes the number of elements of a set. The performance profiles of the methods discussed in this paper are shown below.

Nwaeze et al's [28] line search method was used in all the computations since it satisfies the standard Wolfe conditions [18].

5 Discussions on Numerical Results

Table 1 contains the numerical results obtained through the new method vis-à-vis some existing methods. Table 2 displays the convergence trend of FONCGM on problem (5). These results indicate that the new method compares favorably well with the other methods. The execution time depends on various methods used for evaluating the step lengths and the speed of computer processing unit. I observed that the new

method is relatively faster in some of the iterations recorded for the tested problems. In confirmation, Figs. 1 and 2 shows that the new method is fast and less costly as the number of function iterations per computed problem is relatively low. Finally, I saw that the results are accurate.

Table 2. (Results of problem (5) with N=1000)

Iteration	$f(x)$	$\sqrt{\|\| g \|\|}$
1	12100	5207.0799171207021
2	97.345115883282858	8.6092401363759200
3	97.345115883282858	8.6092440243170521
4	97.100196984612197	25.331224458370258
5	97.100196965540889	25.323077747407670
91	1.340935157300257e-011	1.370669547082824e-005
92	1.266377812796263e-011	3.523846673981857e-005
93	8.809655111200542e-012	6.884192394861915e-005
94	2.666369056764479e-012	5.595801520356587e-005
95	5.690424496376759e-013	2.349487098812904e-005
96	2.516532960480889e-013	8.476983780570614e-006
97	2.111016849525381e-013	3.028277554699159e-006
98	2.058344114262070e-013	1.173173451545769e-006

$x_{optimal}$ = *(0.999999980348753, 0.999999960684452, 0.999999980348753,..., 0.999999979210695)) is the obtained optimal point from the above experiment*

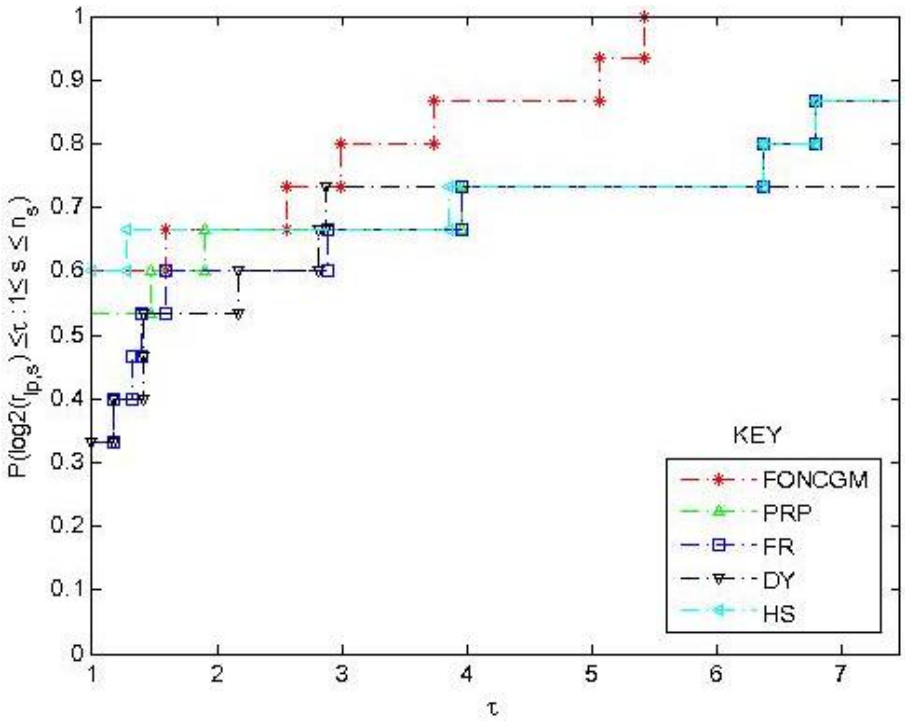

Fig. 1. Performance profiles on number of iterations

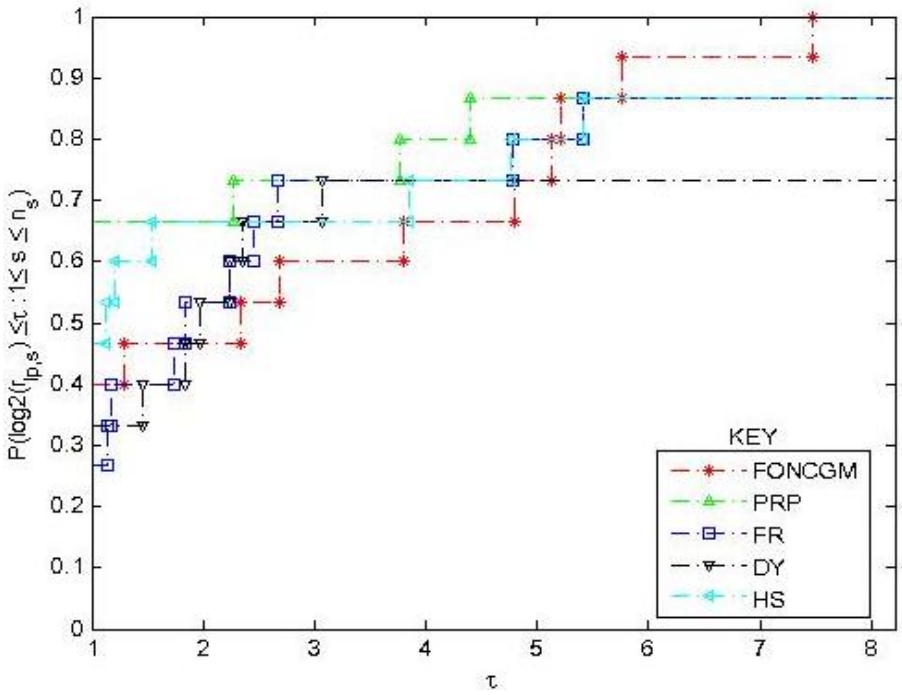

Fig. 2. Performance profiles on execution time

6 Conclusion

I hereby present a fourth-order nonlinear conjugate gradient method in large scale optimization to scientists and engineers. Some of the basic properties of the method have been explored and exploited. On comparison with some existing methods, FONCGM obtained better results. The numerical results show that the method is highly efficient and reliable. Table 1 shows that FONCGM is relatively faster in some of the iterations recorded from the tested problems. On comparison with known results, Figs. 1 and 2 show that the performance profile of this method is relatively better. It is less costly as the number of function iterations per computed problem is relatively low. Finally, the obtained results are very close to the exact solutions.

Acknowledgements

I acknowledge God for his kind direction.

Author's Contribution

The sole author designed, analyzed and interpreted and prepared the manuscript.

Competing Interests

Author has declared that no competing interests exist.

References

[1] Andrei N. 40 conjugate gradient algorithms for unconstrained optimization: A survey on their definition, ICI technical report No.13/08. Research Institute for Informatics, Centre for Advanced Modeling and optimization, Romania. 2008;13.

[2] Sanmtias S, Vercher E. A generalized conjugate gradient algorithm. Journal of Optimization Theory and Applications. 1988;98;489–502.

[3] Dai YH, Yuan Y. Nonlinear conjugate gradient methods. Shanghai Science and Technology Press, Shanghai; 2000.

[4] Fletcher R, Reeves C. Function minimization by conjugate gradients. Computer J. 1964;**7:**149-154.

[5] Shi ZJ. Nonlinear conjugate gradient method with exact line search. Acta Math. Sci. Ser. A Chin. Ed. 2004;24(6):675–682.

[6] Hagger WW, Zhang H. A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM Journal on Optimization. 2005;16:170-192.

[7] Boland WR, Kamgnia ER, Kowalik JS. A CG optimization method invariant to nonlinear scaling. Journal of Optimization Theory and Applications. 1979;27(2):221–230.

[8] Fried I. N-step conjugate gradient minimization scheme for nonquadratic functions. AIAA Journal. 1971;9:2286-2287.

[9] Goldfarb D. Variable metric and conjugate-direction methods in unconstrained optimization: Recent developments. ACM proceedings, Boston, Massachusetts; 1972.

[10] Dai YH. Conjugate gradient methods with rmijo-type line searches. Act Mathematicae App. 2002;18: 123-130.

[11] Dai YH, Han JY, Liu GH, Sun DF, Yin HX, Yuan Y. Convergence properties of nonlinear conjugate gradient methods. SIAM J. Optim. 2000;10:345-358.

[12] Dai YH, Yuan Y. Convergence properties of the conjugate gradient descent method. Adv. Math. 1996;25(6):552-562.

[13] Dai YH, Yuan Y. A nonlinear conjugate gradient method with a strong global convergence property. SIAM J. Optim. 1999;10:177-182.

[14] Fletcher R. Practical methods of optimization. John Wiley & Sons, New York; 1987.

[15] Shi ZJ, Guo J. A new algorithm of nonlinear conjugate gradient method with strong convergence. Computational and Applied Mathematics. 2008;27(1):93-106.

[16] Polak E, Ribiere G. Note sur la convergence de directions conjugees. Rev. Francaise Informat Recherche Opertionelle, 3e Annee. 1969;16:35-43.

[17] Polyak BT. The conjugate gradient method in extreme problems. USSR Comp. Math and Math. Phys. 1969;9:94-112.

[18] Hestenes MR, Stiefel EL. Methods of conjugate gradients for solving linear systems. J. Res. Nat. Bur. Stds. 1952;49:409-436.

[19] Liu Y, Storey C. Efficient generalized conjugate gradient algorithms. J. Optim. Theory Appl. 1991;69:129-137.

[20] Tang M, Yuan Y. Using truncated conjugate gradient method in trust-region method with two subprograms and backtracking line search. Computational and Applied Mathematics. 2010;29(2): 89-106.

[21] Nocedal J, Yuan Y. Combining trust-region and line search techniques. Advances in Nonlinear Programming. 1998;153-175.

[22] Yuan Y, Stoer J. A subspace study on conjugate gradient algorithms. Z. Angew Math. Mech. 1995; 75:69-77.

[23] Smirnov AV. Introduction to tensor calculus; 2004. Available: http://faculty.gg.uwyo.edu

[24] National Mathematical Centre (NMC). Abuja, foundation postgraduate course on computational methods and applications in optimization. Abuja, Nigeria. 1999;12-18.

[25] Conn AR, Gould NIM, Toint PL. Testing a class of methods for solving minimization problems with simple bounds on the variables. Math. Comp. 1998;50:3999-430.

[26] More J, Garbow B, Hillstrom K. Testing unconstrained optimization software. ACM Transactions on Mathematical Software. 1981;7:17-41.

[27] Dolan ED, More JJ. Benchmarking optimization software with performance profiles. Mathematical Programming. 2002;91(2):201-213.

[28] Nwaeze E, Isienyi SU, Zhengui L. An augmented cubic line search algorithm for solving high-dimensional nonlinear optimization problems. Journal of Nigerian Mathematical Society. 2013;32: 185-191.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/10076