

Original Research Article

A FOURTH-ORDER NONLINEAR CONJUGATE GRADIENT METHOD IN EQUALITY CONSTRAINED OPTIMIZATION

Nwaeze Emmanuel^{1*}, Ozoigbo Gerald.²

^{1,2}Department of Maths/Comp. Sc./Stat./Info, Federal University Ndufu-Alike, Ikwo, Nigeria.

E-mail¹: nwaezeema@yahoo.com. E-mail²: geraldzoigbo@yahoo.co.uk.

ABSTRACT

This paper presents a fourth-order nonlinear conjugate gradient method in equality constrained optimization. The idea is to transform the constrained problem into unconstrained type through the Lagrange multipliers scheme. Using four terms of Taylor series development, we approximate the transformed function (augmented Lagrange function). Lastly, we employ the new fourth-order nonlinear conjugate gradient method in equality constrained optimization to solve the optimization problem. We present the algorithm in steps and some properties of the gradients are proved, using classical results. Also, the convergence analysis has been proved under classical and known assumptions. Furthermore, we present the obtained numerical results and compare them to some existing results. The analysis of results confirms that the new method is accurate.

Keywords: Fourth-order conjugate gradient method, equality constrained optimization, objective function, nonlinear polynomial approximation, Lagrange multipliers scheme.

Mathematical subject classification (2010): 65K10

1. INTRODUCTION

The equality constrained optimization of a smooth function, Q , in many variables remains an important problem in optimization theory. This is true since many scientists seek to solve this class of problems, in real life applications. Every equality constrained optimization problem [1] could be put in the form

$$\underset{x}{\text{Optimize}} \quad Q(x) \tag{1}$$

subject to

$$w_i(x) = b_i, \quad i = 1, 2, \dots, m,$$

where $Q(x) \in \mathfrak{R}$, $W_i(x) \in \mathfrak{R}^m$ and $b_i \in \mathfrak{R}$. The general approach is to transform the problem into unconstrained type by Lagrange multipliers scheme and solve the zeros of the function gradient since the local minima occur at stationary points. A fourth-order nonlinear conjugate gradient method in equality constrained optimization finds the global minimum of the transformed function. The new method is characterized by the following. Consider the transformed case of problem (1):

$$\min_{x \in \mathfrak{R}^n} f(x) \tag{2}$$

where f is a differentiable function. We note that $\max_{x \in \mathfrak{R}^n} f(x) = \min_{x \in \mathfrak{R}^n} (-f(x))$. In order to

solve this unconstrained problem, we need to design a special algorithm that reduces the high storage and computation cost of some computed matrices [2]. Various types of conjugate gradient method have been used to solve unconstrained minimization problems [3]. 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 [4, 5]. The process of minimizing a non-quadratic objective function through the conjugate gradient method is called the nonlinear conjugate gradient method [6, 7]. Many scholars have published their findings on this method [8, 9,10]. New algorithms on nonlinear conjugate gradient method are available[11, 12, 13, 14, 15]. Every conjugate gradient method is an iterative scheme of the form

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, \dots \quad (3)$$

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

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_k d_k, & k > 0. \end{cases} \quad (4)$$

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

$$\beta^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2} [5], \quad \beta^{PRP} = \frac{g_k^T (g_k - g_{k-1})}{\|g_{k-1}\|^2} [17], \quad \beta^{HS} = \frac{g_k^T (g_k - g_{k-1})}{d_{k-1}^T (g_k - g_{k-1})} [18, 19],$$

$$\beta^{CD} = -\frac{\|g_k\|^2}{d_{k-1}^T g_{k-1}} [15], \quad \beta^{DY} = \frac{\|g_k\|^2}{d_{k-1}^T (g_k - g_{k-1})} [19], \quad \beta^{LS} = -\frac{g_k^T (g_k - g_{k-1})}{d_{k-1}^T g_{k-1}} [20] \text{ and}$$

Many of these conjugate gradient methods use inexact line search technique [21]. Others use exact line search approach [22]. Stoer [23] 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 equality constrained optimization. This method is presented in seven sections. Sections (two and three) discuss the transcription of the equality constrained problem to unconstrained type and FONCGM, respectively. In section four, we give the convergence analysis. Section five presents some test problems. Section six explains the numerical results while section seven ends this work with a conclusion.

2. TRANSCRIPTION OF EQUALITY CONSTRAINED OPTIMIZATION PROBLEM

We transform the constrained problem (1) into unconstrained problem of the form

$$\text{Optimize } f(x, \lambda) \quad (5)$$

$$x \in \mathfrak{R}^N$$

where $\lambda \in \mathfrak{R}^m$ and f is the Lagrange's function defined by

$$f(x, \lambda) = Q(x) + \sum_{i=1}^m \lambda_i (b_i - W_i(x)). \quad \lambda_i, \quad i = 1, 2, \dots, m \text{ are the Lagrange multipliers. A}$$

solution to problem (1) can then be found by solving problem (5) if there exists a vector

$\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)$ such that

$$\frac{\partial f(x, \lambda)}{\partial x_1} = 0, \quad \frac{\partial f(x, \lambda)}{\partial x_2} = 0, \quad \dots, \quad \frac{\partial f(x, \lambda)}{\partial x_N} = 0 \quad (6)$$

and

$$\frac{\partial f(x, \lambda)}{\partial \lambda_1} = 0, \quad \frac{\partial f(x, \lambda)}{\partial \lambda_2} = 0, \quad \dots, \quad \frac{\partial f(x, \lambda)}{\partial \lambda_m} = 0. \quad (7)$$

Equations (6) and (7) will generate a set of $N + m$ equations in $N + m$ unknowns to be solved. As the dimensionality of the problem increases, we have many equations to solve simultaneously. The problem computation becomes very tedious, if analytic method is

employed. Thus, we develop and apply the fourth-order nonlinear conjugate gradient method where the objective function is the augmented Lagrange function defined by

$$f(x, \lambda, \mu) = Q(x) + \sum_{i=1}^m \lambda_i (b_i - W_i(x)) + \frac{1}{2} \sum_{i=1}^m \mu (b_i - W_i(x))^2 \quad (8)$$

μ is a scalar called the penalty parameter. The multipliers updates usually take the form

$$\lambda_{i+1} = \lambda_i + \mu(b_i - W_i(x)) \quad (9)$$

Our approach is to choose initial vectors x_0, λ_0 , a parameter μ_k and use the new fourth-order nonlinear conjugate gradient method to optimize $f(x, \lambda, \mu_k)$ over $\mathfrak{R}^N$. The scalars $\mu_0, \mu_1, \dots, \mu_k$ could be determined on the basis of results obtained during iteration process. The new fourth-order nonlinear conjugate gradient method in equality constrained optimization follows.

3. THE FOURTH-ORDER NONLINEAR CONJUGATE GRADIENT METHOD (FONCGM)

The fourth-order nonlinear conjugate gradient method is based on four terms 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 .

$$F(x) = f(x_k) + df(x_k) + \frac{1}{2!} d^2 f(x_k) + \frac{1}{3!} d^3 f(x_k) + \frac{1}{4!} d^4 f(x_k), \quad (10)$$

$$\text{where } d^n f(x_k) = \sum_{i_1}^N \sum_{i_2}^N \dots \sum_{i_n}^N h_{i_1} h_{i_2} \dots h_{i_n} \frac{\partial^n f(x_k)}{\partial x_{i_1} \partial x_{i_2} \dots \partial x_{i_n}},$$

$$x, x_k, h_{i_j} \in \mathfrak{R}^N; h_{i_j} = x - x_k, \quad 2 \leq n \leq 4.$$

Using a vector $h = x - x_k$ and $A_i = \nabla^i f(x_k)$, in equation (10), we have

$$\begin{aligned} F(x) &= f(x_k) + h^T A_1 + \frac{1}{2!} h^T A_2 h + \frac{1}{3!} h^T (h^T A_3 h) + \frac{1}{4!} h^T (h^T A_4 h) h \\ &= f(x_k) + h^T A_1 + \frac{1}{2!} h^T \left[A_2 + \left\{ \frac{2}{3!} A_3 + \frac{2}{4!} A_4 h \right\} h \right] h. \end{aligned} \quad (11)$$

Using tensor notations presented in [24], we have

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

where $g(x)$ denotes the gradient of f , at point x , $\prod_{p=1}^2 h^{Z_p} = h^T h$, $\binom{n}{m} = \frac{n!}{(n-m)! m!}$,

$Z_p = T^{\frac{1}{2}[1 + (-1)^p]}$ and T denotes transpose. It follows that

$$F(x) = f(x_k) + h^T A_1 + \frac{1}{2} h^T H(x) h \quad (13)$$

where

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

Similarly,

$$\begin{aligned} \nabla F(x) &= A_1 + A_2 h^T + \frac{1}{2} h^T A_3 h + \frac{1}{3!} h^T (h^T A_4 h) \\ &= A_1 + \sum_{j=2}^4 \frac{1}{(j-1)!} A_j \prod_{p=1}^{j-1} h^{z_p} \\ &= A_1 + \sum_{j=2}^4 \frac{1}{(j-1)!} \sum_{m=0}^{j-1} (-1)^m \binom{j-1}{m} g[x_k + \{(j-1)-m\}h] \\ &= A_1 + \frac{1}{6} [g(x_k + 3h) + 3g(x_k + h) - 4g(x_k)] \\ &= A_1 + \frac{1}{6} [g(3x - 2x_k) + 3g(x) - 4g(x_k)] \end{aligned} \quad (14)$$

$$\nabla F(x_{k+1}) = A_1 + \frac{1}{6} [g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) - 4g(x_k)] \quad (15)$$

$$\nabla F(x_k) = A_1 = g(x_k). \quad (16)$$

$$\begin{aligned} \nabla F(x_{k+1}) &= \nabla F(x_k) + \frac{1}{6} [g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) - 4g(x_k)] \\ &= \frac{1}{6} [g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) - 4g(x_k) + 6g(x_k)] \\ &= \frac{1}{6} [g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) + 2g(x_k)]. \end{aligned} \quad (17)$$

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

$$D_{k+1}^T H_{k+1} D_k = 0 \quad (18)$$

From the classical results, it follows that

$$G_{k+1} = G_k + \alpha_k H_{k+1} D_k \quad (19)$$

With a given x_0 , α_k is computed such that

$$F(x_k + \alpha_k D_k) < F(x_k), \quad (20)$$

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

and

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

From equations (12) and (15),

$$\begin{aligned} [-G_{k+1} + \beta_k D_k]^T H_{k+1} D_k &= 0; \quad \beta_k D_k^T H_{k+1} D_k = G_{k+1}^T H_{k+1} D_k. \text{ From equation (19),} \\ \beta_k &= \frac{G_{k+1}^T (G_{k+1} - G_k)}{D_k^T (G_{k+1} - G_k)} \end{aligned} \quad (21)$$

The algorithm is described below.

129

130 **Algorithm 1. (FONCGM)**

131 Step 1: Select $x_0 \in \mathfrak{R}^N$, $N \geq 2$, $\|\cdot\|$ is Euclidean norm, λ_0, μ_k and $\varepsilon > 0$ (a small number:
132 0.000001).
133 Set $G_0 = \nabla f_\mu(x_0, \lambda_0, \mu_k)$, $D_0 = -G_0$ and $k = 0$.
134 Step 2: If $\|G_k\| \leq \varepsilon$, stop. Choose x_k , otherwise go to step 3.
135 Step 3: Compute α_k such that $F(x_k + \alpha_k D_k) < F(x_k)$ and go to step 4.
136 Step 4: Compute
137
$$G_{k+1} = \frac{1}{6}[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) + 2g(x_k)], \beta_k = \frac{G_{k+1}^T(G_{k+1} - G_k)}{G_k^T(G_{k+1} - G_k)}, D_{k+1} = -G_k + \beta_k D_k.$$

138 $x_{k+1} = x_k + \alpha_k D_k$.
139 Go to step 5.
140 Step 5: Check for optimality of G .
141 If $G_{i+1} = 0$ and $|\lambda_{i+1} - \lambda_i| > \varepsilon$, go to step 6. If $G_{i+1} = 0$ and $|\lambda_{i+1} - \lambda_i| \leq \varepsilon$, go to step
142 7.
143 Step 6: Update λ as follows.
144 $\lambda_{i+1} = \lambda_i + \mu_k(b_i - W_i(x_i))$
145 Choose $\mu_{k+1} > \mu_k$ such as $\mu_{k+1} = \mu_k + 0.1 * 0.2^j$; $j \leq 1000$; $\mu_0 = 0$
146 Set $k = k + 1$. Go to step 2.
147 Step 7: Stop iteration. x_{i+1} is the final optimal point found
148 **Remark:** Dai and Yuan [12] presented a nonlinear conjugate gradient algorithm for solving
149 unconstrained optimization problems. Below is Dai-Yuan's algorithm for problem (1).
150 **Algorithm 2. (Nonlinear conjugate gradient method)**
151 Step 1: Select $x_0 \in \mathfrak{R}^N$, $N \geq 2$ and $\varepsilon > 0$. Set $d_0 = -g_0$ and $k = 0$.
152 Step 2: If $\|g_k\| \leq \varepsilon$, stop. Take x_k . Otherwise go to step 3.
153 Step 3: Compute α_k such that $f(x_k + \alpha_k d_k) < f(x_k)$, go to step 4.
154 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 $x_{k+1} = x_k + \alpha_k d_k$.
155 Step 5: Set $k = k + 1$. Go to step 2.

157 4. CONVERGENCE ANALYSIS

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

161 A. Assumptions:

- 162 i. f is bounded below in $\mathfrak{R}^N$ and is four times continuously differentiable in a
163 neighborhood Z of the level set
164 $L = \{x \in \mathfrak{R}^N : f(x) \leq f(x_0)\}$
- 165 ii. The gradient, $g(x)$, is Lipschitz continuous in Z , namely, there exists a constant
166 $Lc > 0$ such that
167 $\|\nabla f(x) - \nabla f(y)\| \leq Lc \|x - y\|$, $x, y \in Z$.
- 168 iii. The extended hessian matrix $H(x)$ is positive definite.

169 B. Lemma:

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

$$173 \quad \sum_{k \geq 0} \frac{(G_k^T D_k)^2}{\|D_k\|^2} < \infty$$

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

176 $\{x_k, k = 1, 2, \dots\}$ be generated by algorithm (1). Then, the algorithm either
 177 terminates at a stationary point or converges in the sense that

$$178 \quad \liminf_{k \rightarrow \infty} \|G(x_k)\| = 0$$

179 **Theorem 1.** Suppose that f is continuously differentiable, bounded below and the norm of
 180 the Hessian matrix is bounded. The iteration $\{x_k\}$ is generated by algorithm (2) satisfies
 181 $x_k \rightarrow x^*$ as $k \rightarrow \infty$ and the Hessian matrix of f is positive definite. Let ε_k be the relative
 182 error in the truncated conjugate gradient method and the algorithm. If $\varepsilon_k \rightarrow \infty$ then $\{x_k\}$
 183 converges, that is,

$$184 \quad \lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

185 **Proof (Lemma (i)):**

186 Dai and Yuan proved this lemma for algorithm (2): $\sum_{k \geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty$. Similar proof follows,

187 since the search directions d_k and D_k have same definition. That is,

$$188 \quad \sum_{k \geq 0} \frac{(G_k^T D_k)^2}{\|D_k\|^2} < \infty. \text{ This is true since}$$

$$\begin{aligned} 189 \quad \sum_{k \geq 0} \frac{(G_{k+1}^T D_{k+1})^2}{\|D_{k+1}\|^2} &= \frac{1}{36} \frac{\{[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) + 2g(x_k)]^T D_{k+1}\}^2}{\|D_{k+1}\|^2} \\ 190 \quad &= \frac{\|D_{k+1}\|^2}{36} \left(\frac{[g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) + 2g(x_k)]^T D_{k+1}}{\|D_{k+1}\|^2} \right)^2 \\ 191 \quad &\leq \frac{\|D_{k+1}\|^2}{36} \left[\frac{(g(3x_{k+1} - 2x_k)^T D_{k+1})^2}{\|D_{k+1}\|^2} + \frac{(3g(x_{k+1})^T D_{k+1})^2}{\|D_{k+1}\|^2} + \frac{(2g(x_k)^T D_{k+1})^2}{\|D_{k+1}\|^2} \right]^2 \\ 192 \quad &= \frac{\|D_{k+1}\|^2}{36} \left[\frac{(g(3x_{k+1} - 2x_k)^T D_{k+1})^2}{\|D_{k+1}\|^2} + \frac{(3g(x_{k+1})^T D_{k+1})^2}{\|D_{k+1}\|^2} + \frac{(2g(x_k)^T D_{k+1})^2}{\|D_{k+1}\|^2} \right]^2 < \infty. \end{aligned}$$

193 Thus,

$$194 \quad \sum_{k \geq 0} \frac{(G_k^T D_k)^2}{\|D_k\|^2} < \infty \text{ as required.}$$

195 **Proof (Lemma (ii)):**

196 Dai and Yuan proved this lemma for algorithm (2). The proof is same since, from equation
197 (16),
198 $G_k = G(x_k) = \nabla F(x_k) = g(x_k)$. It follows that
199 $\liminf_{k \rightarrow \infty} \|G_k\| = \lim_{k \rightarrow \infty} \|g_k\| = 0$. This is true since
200 $\liminf_{k \rightarrow \infty} \|G_{k+1}\| = \liminf_{k \rightarrow \infty} \frac{1}{6} \| \{g(3x_{k+1} - 2x_k) + 3g(x_{k+1}) + 2g(x_k)\} \|$
201 $\leq \liminf_{k \rightarrow \infty} \frac{1}{6} \|g(3x_{k+1} - 2x_k)\| + \liminf_{k \rightarrow \infty} \frac{1}{6} \|3g(x_{k+1})\| + \liminf_{k \rightarrow \infty} \frac{1}{6} \|2g(x_k)\| = 0$.
202 $\liminf_{k \rightarrow \infty} \|G_{k+1}\| \leq 0$. But $\|G_{k+1}\| \geq 0$ implies that $\liminf_{k \rightarrow \infty} \|G_{k+1}\| = 0$ or $\liminf_{k \rightarrow \infty} \|G_k\| = 0$
203 as required.
204 **Proof of theorem (1):** The proof is available in many literatures. Noting that
205 $G_k = G(x_k) = \nabla F(x_k) = g(x_k)$, the proof is same since the assumptions on algorithm (1)
206 meet the requirements of this theorem. Using $M \geq m \geq 0$, $\varepsilon_k = x_k - x^*$ and the results
207 from NMC [25], we have
208 $\|x_{k+1} - x^*\|^2 = (x_{k+1} - x^*)^T (x_{k+1} - x^*)$
209 $= (x_{k+1} - x^*)^T (x_k + \alpha_k d_k - x^*)$
210 $\leq \frac{M}{n} \left(\frac{1-R}{1+R} \right)^{2(k+1)} \|x_k - x^*\|^2 \leq \frac{M}{n} \left(\frac{1-R}{1+R} \right)^{2(k+1)} \|x_0 - x^*\|^2$; $R = \frac{m}{M}$.
211 $\|x_{k+1} - x^*\| \leq \sqrt{\frac{M}{n} \left(\frac{1-R}{1+R} \right)^{(k+1)}} \|x_k - x^*\|$
212 $\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0$, since $\lim_{k \rightarrow \infty} \sqrt{\frac{M}{n} \left(\frac{1-R}{1+R} \right)^{(k+1)}} = 0$.
213 End of proof.
214 **5. NUMERICAL CONSIDERATION**
215 To illustrate the behavior of the algorithm proposed in this paper, we wrote MATLAB codes
216 for solving the following problems and ran them on a PC with Windows 7. The gradient
217 tolerance is 0.000001. The problems are of the form (1) with the following expressions for
218 $Q(x)$ and the constraints.
219 Problem 1 (HS48[26])
220 $Q(x) = (x_1 - 1)^2 + (x_2 - x_3)^2 + (x_4 - x_5)^2$
221 such that $x_1 + x_2 + x_3 + x_4 + x_5 - 5 = 0$
222 $x_3 + 2(x_4 + x_5) + 3 = 0$
223 $x_0 = (3, 5, -3, 2, -2)$; $x^* = (1, 11/3, 11/3, -5/3, -5/3)$; $f(x^*) = 0$
224 Problem 2 (HS51[26])
225 $Q(x) = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2$
226 such that $x_1 + 3x_2 - 4 = 0$
227 $x_3 + x_4 - 2x_5 = 0$
228 $x_2 - x_5 = 0$
229 $x_0 = (2.5, 0.5, 2, -1, 0.5)$; $x^* = (1, 1, 1, 1, 1)$; $f(x^*) = 0$
230 Problem 3 (HS50[26])

231 $Q(x) = (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6$
 232 such that $x_1 + x_2 + x_3 + 4x_4 - 7 = 0$
 233 $x_3 + 5x_5 - 6 = 0$
 234 $x_0 = (10, 7, 2, -3, 0.8)$; $x^* = (1, 1, 1, 1, 1)$; $f(x^*) = 0$.
 235
 236 Problem 4 [27, problem(11)]
 237 $Q(x) = \sum_{i=1}^{100} ix_i^4$
 238 such that $c_i(x) = \sum_{j=1}^{i+1} x_j - \frac{i}{10} = 0, i = 1, 2, \dots, 20$.
 239 $x_0 = (0.25, 0.25, \dots, 0.25)$; $x^* = (0.0557, 0.0443, \dots, -0.0011, -0.0011)$; $f(x^*) = 0.0228$.
 240
 241
 242 Problem 5 [28, problem(394)]
 243 $Q(x) = \sum_{i=1}^{20} i(x_i^2 + x_i^4)$
 244 such that $c_1(x) = \sum_{i=1}^{20} x_i^2 - 1 = 0$
 245 $x_0 = (2, 2, 2, \dots, 2)$; $x^* = (0.91287, 0.408268, -0.000017, \dots, -0.0000014)$;
 246 $f(x^*) = 1.91667$.
 247
 248 Problem 6 [29, problem 8]
 249 $Q(x) = \frac{1}{2} \sum_{i=1}^{k-2} (x_{k+i+1} - x_{k+i})^4$
 250 such that $x_{k+i} - x_{i+1} + x_i = i, i = 1, 2, \dots, k-1$.
 251
 252 $x_0 = (1, 2, \dots, k, 2, 3, \dots, k)$; $k = 3$; $f(x^*) = 0$.
 253
 254 Problem 7 [29, problem 8]
 255 $Q(x) = \frac{1}{2} \sum_{i=1}^{k-2} (x_{k+i+1} - x_{k+i})^4$
 256 such that $x_{k+i} - x_{i+1} + x_i = i, i = 1, 2, \dots, k-1$.
 257
 258 $x_0 = (1, 2, \dots, k, 2, 3, \dots, k)$; $k = 10$; $f(x^*) = 0$.
 259
 260 Problem 8 [29, problem 10]
 261 $Q(x) = \sum_{i=1}^n \cos^2(2\pi x_i \sin(\pi/20))$
 262 such that $x_i - x_{i+1} = 0.4, i = 1, 2, \dots, n-1$.
 263

265 $x_0 = (1, 0.6, 0.2, \dots, 1 - 0.4((i - 1))); n = 3; f(x^*) = 0.2640.$
 266 Problem 9 [29, problem 10]
 267 $Q(x) = \sum_{i=1}^n \cos^2(2\pi x_i \sin(\pi / 20))$
 268 such that $x_i - x_{i+1} = 0.4, i = 1, 2, \dots, n - 1.$
 269 $x_0 = (1, 0.6, 0.2, \dots, 1 - 0.4((i - 1))); n = 10; f(x^*) = 0.2700.$
 271 Problem 10 [29, problem 9]
 272 $Q(x) = \sum_{i=1}^{k-2} 100(x_{k+i+1} - x_{k+i})^2 + (1 - x_{k+i})^2$
 273 such that $x_{k+i} - x_{i+1} + x_i = i, i = 1, 2, \dots, k - 1.$
 274 $x_0 = (1, 2, \dots, k, 2, 3, \dots, k); k = 3; f(x^*) = 0.$
 275 Problem 11 [29, problem 9]
 276 $Q(x) = \sum_{i=1}^{k-2} 100(x_{k+i+1} - x_{k+i})^2 + (1 - x_{k+i})^2$
 277 such that $x_{k+i} - x_{i+1} + x_i = i, i = 1, 2, \dots, k - 1.$
 278 $x_0 = (1, 2, \dots, k, 2, 3, \dots, k); k = 10; f(x^*) = 0.$
 279 Problem 12 [30, Problem 4]
 280 $Q(x) = e^{x_1 x_2 x_3 x_4 x_5}$
 281 such that $x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$
 282 $x_2 x_3 - 5x_4 x_5 = 0$
 283 $x_1^3 + x_2^3 + 1 = 0$
 284 $x_0 = (-2, 2, 2, -1, -1); f(x^*) = 0.05395$
 285 The numerical results obtained for the new method vis-a-vis some classical methods
 286 (FOCGM, FR, DY, PRP) are presented in table (1), below (P: Problem; ITE: Number of
 287 iterations; TIME: Computer execution time (s); FN.: Function value at the end of iterations).
 288 The stopping criterion is $\|g(x_k)\| < 0.000001$ while the maximum number of iterations is
 289 1000.
 290
 291
 292
 293

P	FONCGM			FR			DY			PRP		
	ITE	TIME	FN.	ITE	TIME	FN.	ITE	TIME	FN.	ITE	TIME	FN.
1	10	0.07	1.15E-12	9	0.06	1.97E-12	3	0.04	2.02E-15	3	0.04	1.33E-12
2	5	0.04	6.59E-16	20	0.05	1.57E-15	12	0.05	2.69E-14	20	0.04	5.16E-16
3	5	0.24	1.17E-9	20	0.35	8.86E-11	2	0.3	6.35E-10	2	0.29	4.61E-9
4	330	20.10	0.022817	336	20.56	0.022817	336	20.58	0.022817	336	20.57	0.022817
5	138	2.15	1.916666	180	2.95	1.916666	180	3.08	1.916666	175	2.55	1.916666
6	2	0.09	6.89E-12	4	0.13	2.39E-10	4	0.11	2.39E-10	19	0.16	1.12E-10
7	600	3.17	2.9E-9	900	4.26	8.69E-9	900	3.34	1.78E-9	900	3.5	6.14E-10
8	761	0.86	0.293549	100	1.03	0.293549	100	0.94	0.293549	100	1.1	0.293549
9	100	46	1.939985	100	141	4.300656	100	27.38	1.676617	100	28	4.113811
10	171	0.30	1.58E-13	900	0.37	2.54E-14	501	0.08	9.55E-13	900	0.34	2.33E-15
11	202	2.74	2.47E-14	304	2.43	3.08E-13	323	2.81	2.39E-13	120	1.67	1.43E-12
12	9	0.35	0.05395	10	0.21	0.05395	10	0.21	0.05395	10	0.20	0.05395

TABLE 1: Number of iterations and CPU time in seconds.

Iteration	$f(x)$	Sum of constraints
1	0.900335462627903	4.000000000000000
2	0.059832098605313	0.068911047490848
3	0.054326438636889	0.004569764836374
4	0.053979357005701	0.000052498245927
5	0.053952294603960	0.000093989215249
6	0.053950054179953	0.000036636879952
7	0.053949865266059	0.000011504805313
8	0.053949849268435	0.000004014424548
9	0.053949847897123	0.000000713781098

Table 2: Solution of problem (12) by algorithm (1).

Performance profiles have been introduced by Dolan and Moré [31]. The main idea is to show, graphically, the **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

$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}$. 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

$$y_s : [1, w] \rightarrow [0, 1] \quad \text{such that} \quad y_s(t) = \frac{n(\{p \in P : r_{p,s} \leq t\})}{n(P)} \quad \text{or}$$

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

where $n(\cdot)$ denotes the number of elements of a set. The performance profiles of the methods discussed in this paper are shown below.

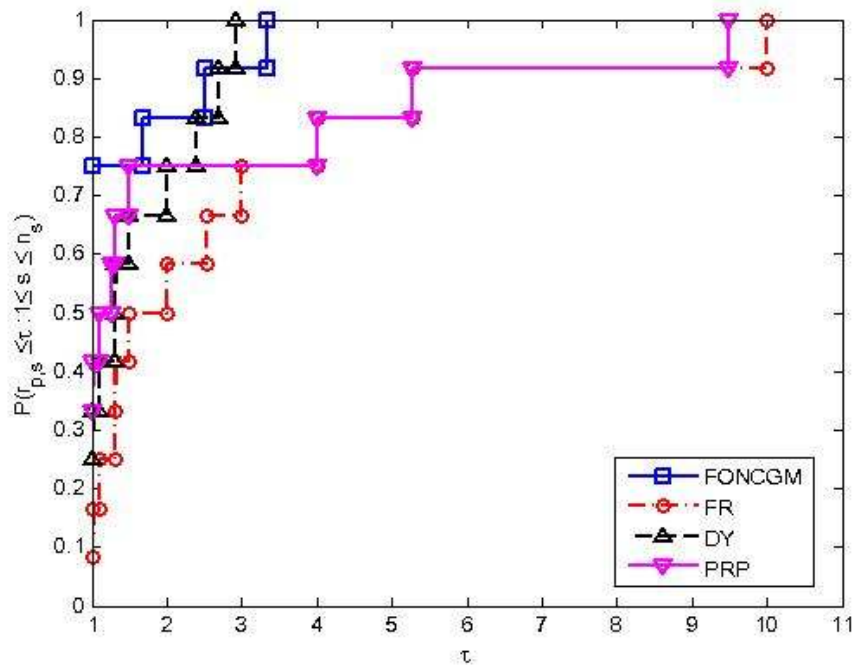


FIGURE 1: Performance profiles on number of iterations

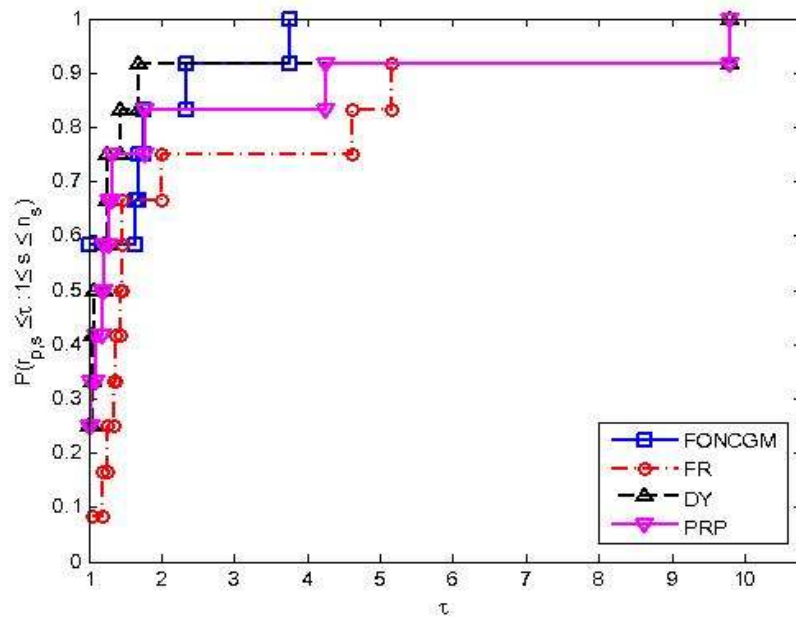


FIGURE 2: Performance profiles on execution time.

Nwaeze et al's [32] line search method was used in all the computations since it satisfies the standard Wolfe conditions [19].

E-mail address: nwaezeema@yahoo.com.

6. DISCUSSION 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 algorithm (1) on problem (12). 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. We observed that the new method is relatively faster in some of the iterations recorded for the tested problems. In confirmation, figures (1) and (2) show that the new method is fast and less costly as the number of function iterations per computed problem is relatively low. Finally, we saw that the results are accurate.

7. CONCLUSION

We hereby present a fourth-order nonlinear conjugate gradient method in equality constrained optimization to scientists and engineers. Some of the basic properties of the method have been explored and exploited. Numerical results show that the method is highly efficient and reliable.

ACKNOWLEDGEMENTS

We wish to acknowledge the timely effort of Mr. Chukwuebuka Nwaeze who helped in the arrangement of data in this manuscript. We also acknowledge God whose presence made it possible to produce this manuscript.

COMPETING INTERESTS

Authors state that no competing interests exist.

AUTHORS' CONTRIBUTIONS

Nwaeze Emmanuel designed the study, performed the algebraic analysis and wrote the first draft of this manuscript. Mr Ozoigbo Gerald did the literature searches. All authors read and approved the final manuscript.

REFERENCES

- [1] Bunday B.D and Garside G.R., Optimization methods in pascal, Edward Arnold publishers, London. 1987.
- [2] N. Andrei, 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.
- [3] Sanmtias S and Vercher E, A generalized conjugate gradient algorithm, Journal of optimization theory and Applications, 1988; 98; 489 – 502.
- [4] Dai Y.H. and Yuan Y. *Nonlinear* conjugate gradient methods. Shanghai Science and Technology Press, Shanghai, 2000.
- [5] Fletcher R. and Reeves C, *Function* Minimization by conjugate gradients. Computer J.,7, 1964; 149-154.
- [6] Shi Z.J., Nonlinear conjugate gradient method with exact line search. Acta Math. Sci. Ser. A Chin. Ed., 2004; 24(6); 675 – 682.
- [7] Hagger W.W., Zhang H, A new conjugate gradient method with guaranteed descent and an efficient line search, SIAM Journal on Optimization, 2005;16; 170-192.
- [8] Boland W.R., Kamgnia E.R., and Kowalik J.S., A CG optimization method invariant to nonlinear scaling, Journal of optimization theory and applications, 1979; 27(2); 221 – 230.
- [9] Fried I., N-step conjugate gradient minimization scheme for nonquadratic functions, AIAA Journal, 1971; 9; 2286-2287.
- [10] Goldfarb D. Variable metric and conjugate-direction methods in unconstrained optimization: Recent developments, ACM proceedings, Boston, Massachusetts, 1972.
- [11] Dai Y.H., Conjugate gradient methods with rmijo-type line searches. Act Mathematicae App, 2002; 18; 123-130.
- [12] Dai Y.H., Han J.Y., Liu G.H., Sun D.F., Yin H.X. and Yuan Y. Convergence properties of nonlinear conjugate gradient methods. SIAM J. Optim., 2000; 10; 345-358.
- [13] Dai Y.H. and Yuan Y., Convergence properties of the conjugate gradient descent method. Adv. Math., 1996; 25(6), 552-562.

375 [14] Dai Y.H. and Yuan Y, A nonlinear conjugate gradient method with a strong global convergence
 376 property. SIAM J. Optim., 1999; 10; 177-182.
 377 [15] Fletcher R., Practical methods of optimization, John Wiley & Sons, New York, 1987.
 378 [16] Shi Z.J. and Guo J., A new algorithm of nonlinear conjugate gradient method with strong
 379 convergence, Computational and Applied Mathematics, 2008; 27(1); 93-106.
 380 [17] Polak E. and Ribiere G., Note sur la convergence de directions conjugees. Rev.
 381 Francaise Informat Recherche Opertionelle, 3e Annee, 1969; 16; 35-43;
 382 [18] Polyak B.T, The conjugate gradient method in extreme problems. USSR Comp. Math. and Math.
 383 Phys.1969; 9; 94-112.
 384 [19] Hestenes M.R. and Stiefel E.L., Methods of conjugate gradients for solving linear systems. J. Res.
 385 Nat. Bur. Stds, 1952;49; 409- 436.
 386 [20] Liu Y. and Storey C., Efficient generalized conjugate gradient algorithms. J. Optim. Theory Appl.
 387 1991; 69; 129-137.
 388 [21] Tang M. and Yuan Y., Using truncated conjugate gradient method in trust-region method with
 389 two subprograms and backtracking line search, Computational and Applied Mathematics, 2010; 29(2);
 390 89-106.
 391 [22] Nocedal J. and Yuan Y., Combining trust-region and line search techniques. Advances in
 392 nonlinear programming, 1998; 153-175.
 393 [23] Yuan Y. and Stoer J., A subspace study on conjugate gradient algorithms. Z. Angew Math.
 394 Mech.,1995; 75; 69-77.
 395 [24] Smirnov A.V., Introduction to tensor calculus, <http://faculty.gg.uwyo.edu> (2004).
 396 [25] National Mathematical Centre (NMC), Abuja, Foundation Postgraduate course on computational
 397 methods and applications in optimization, Abuja, Nigeria, 1999; 12-18.
 398 [26] Hock W. and Schittkowski K., Test examples for nonlinear programming codes,lecture notes in
 399 Economics and Mathematical systems, Springer, New York, NY USA, 1981.
 400 [27] Bartholomew-Biggs M.C., Hernandez M.F.G., Using the KKTmatrix in an augmented lagrangian
 401 SQP method for sparse constrained optimization, J. Optim. Theory Appl. 1995; 85; 201-220.
 402 [28] Schittkowski K., More test examples for nonlinear programming codes, I in ecture notes in
 403 Economics and Mathematical systems, Springer, Berlin, 1987; 282.
 404 [29] Dai Z., Extension of modified Polak-Ribiere-Polyak conjugate gradient method to linear equality
 405 constraints minimization problems. Abstract and applied analysis journal,
 406 <http://dx.doi.org/10.1155/2014/921364>, 2014; 9.
 407 [30] Nwaeze E., A higher-order conjugate gradient method for solving nonlinear optimization problems,
 408 Ph.D. Thesis, 2008; 65-67.
 409 [31] Dolan E.D. and Moré J.J. Benchmarking optimization software with performance profiles,
 410 Mathematical programming, 2002; 91(2) 201-213.
 411 [32] Nwaeze E., Isienyi S.U., Zhengui L., An augmented cubic line search algorithm for solving high-
 412 dimensional nonlinear optimization problems. Journal of Nigerian Mathematical Society, 2013; 32, 185-
 413 191.
 414