Int. J. Nonlinear Anal. Appl.
Volume 12, Special Issue, Winter and Spring 2021, 97-108
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/IJNAA.2021.4915

Optimizing The Modified Conjugate Gradient Algorithm

Amel Nashat Shakir[a*]

[a] *Master of Mathematics, Technical Institute of Kirkuk, Northern Technical University, Iraq.*

(Communicated by Dr. Ehsan Kozegar)

Abstract

In this paper, an efficient GV1-CG is developed to optimizing the modified conjugate gradient algorithm by using a new conjugate property. This is to increase the speed of the convergence and retain the characteristic mass convergence using the conjugate property. This used property is proposed to public functions as it is not necessary to be a quadratic and convex function.

The descent sharp property and comprehensive convergence for the proposed improved algorithm have been proved. Numerical results on some test function indicate that the new CG-method outperforms many of the similar methods in this field.

Keywords: Conjugate gradient, Developed conjugation property, Improved algorithm, Optimization, Global Convergence, Descent Direction.

1. Introduction

Our interest in the conjugate gradient methods is for two reasons: The first reason is that these methods are among the oldest and the best techniques to solve large dimensions of the linear equations systems. The second is that these methods can be adapted to solve problems of the nonlinear optimizations [13].

There are two types for these methods, the first type is the linear conjugation methods, and it is also known as quadratic conjugation methods. It is sometimes known as the pure conjugate gradient, and these methods are used to minimize convex quadratic function.

The second type is known as the nonlinear conjugate gradient method, known as the non-quadratic conjugate gradient, used to minimize general convex function or nonlinear functions.

The method of linear conjugation was first proposed by Hestenes and Stiefel in 1952 as an iterative method that was originally used as an alternative to the Gaussian deletion method. Its purpose was to solve large the linear systems with a matrix of positive parameters on the computer [10].

*Corresponding Author: Amel Nashat Shakir
Email address: Umayaa75@ntu.edu.com (Amel Nashat Shakir[a*])

Received: October 2020 *accepted:* February 2021

Both Fletcher and Reeves developed the method of Hestenes and Stiefel, and the method conjugate gradient was initially introduced in (1960). It is one of the first known techniques to solving nonlinear optimization of large dimensions. Over the years, many different methods have been proposed along the lines of the original formula of this method some of which are widely used in application [13].

2. Types of Conjugate Gradient Methods

2.1. Classical Conjugate Gradient Methods

The linear conjugate gradient method is an iterative method to solve the problem of miniaturization:

$$Min\ f(x) = \frac{1}{2}x^T Gx - b^T x + c \quad (1)$$

Here, b represents a constant vector, c is a constant value, G is a positive symmetric matrix of type $n \times n$.

Equation (1) can be obtained in an equivalent manner as a system of linear equations as follows:

$$Gx = b \quad (2)$$

This means the unique solution for equation (1) is the same solution for system (2).

Thus, we can prepare the method of gradient conjugation either in an algorithm to solve linear systems or to find the smallest value for the convex quadratic functions [14].

We note that the gradient of the function f equals the residual which is the negative gradient of the linear system, i.e.:

$$f(x) = Gx - b = g(x) \quad (3)$$

and when $x = x_k$

$$g_x = Gx_k - b = g(x) \quad (4)$$

Also, the conjugate gradient method has the ability to generate a set of vectors with the property of conjugacy, that it can calculate the new direction d_{k+1} by using the previous direction d_k and the current gradient g_k . It also selects a linearly composition of the steep descent direction (- $g_{k+1}$the previous direction d_k,we do not need t4o know all the previous directions $d_0, d_1, \cdots, d_{k-1}$ for the set of the conjugate; where d_{k+1} is conjugation with all the previous directions. This property makes this method require a strong space and calculation:

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k \quad (5)$$

β_{k+1} represents a numerical quantity and d_{k+1} and d_k conjugate of the matrix G.

The first search direction at the initial point$d_0 = -g_0$[13] is selected. Then, the convergence rate is linear unless retrieves the iteration [15].

The conjugate gradient method can be extended to include general nonlinear functions using Tyler´s series in approximation of the function. This use is to show the rank of the objective function. Near the solution, these functions behave similarly to quadratic functions [9].

Algorithm (1) (Classical Conjugate Gradient Method) There are many steps for algorithm (1) and they are summarized as follows:

Step 1: Selecting x_0, set $d_0 = -g_0, \epsilon > 0$ and $k = 0$.
Step 2: Computing the step length $\alpha_k > 0$ satisfying the Wolfe line search
$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha g_k^T d_k$ and $g_{k+1}^T d_k \geq c_2 g_k^T d_k$ Since $0 < c_1 < c_2 < 1$.
Step 3: Computing $x_{k+1} = x_k + \alpha_k d_k$ if$\| g_{k+1} \| < 0$, then stop.
Step 4: Computing β_{k+1} and generate trend $d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$.
Step 5: Put $k = k + 1$ and then go to step 2.

The associated gradient algorithms differ according to the choices of the different parameter β_{k+1} in step 4, and the most popular formulas are summarized in the following table:

Table 1: Parameters of the estimation model by (OLS) method

$\beta_{k+1}^{HS} = \frac{g_{k+1}^T y_k}{g_k^T y_k}$	Hestenes – Stiefel (HS) 1952
$\beta_{k+1}^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$	Fletcher – Reeves (RE) 1064
$\beta_{k+1}^{PR} = \frac{g_{k+1}^T y_k}{g_k^T g_k}$	Polak – Ribiere (PR) 1969 – Reeves (RE) 1064
$\beta_{k+1}^{DY} = \frac{g_{k+1}^T g_{k+1}}{g_k^T y_k}$	Dai – Yuan (DY) 1999

2.2. Properties of Classical Conjugate Gradient Algorithms

Conjugate gradient methods have the following properties when the target function is a quadratic function, and linear search is exact:

1. Conjugacy $d_i^T G d_j = 0, J = 0, 1, 2. \cdots, i - 1$. . 2. Orthogonally $g_i^T g_i = 0, J = 0, 1, 2. \cdots, i - 1$.
3. Descent $d_i^T g_i = -g_i^T g_i = - \| g_i \|^2 < 0$.
4. Exact Line Search, ELS $g_{k+1}^T d_k = 0, k \geq 0$.
5. Quadratic Termination Property [18].
6. Multipliers requiring O(n) of the calculations for each recurrence of the approximation to the minimum meaning that there is a global convergence algorithms) [9].
7. $Spand_0, d_1, \cdots, d_{k+1} = spang_0, g_1, \ldots, g_{k+1} = spang_0, Gg_1, \ldots, G^k g_{k+1}$ making conjugate gradient methods (Krylov Subspace) [5].

Global Convergence of Descent Method

Let $f(x)$ is continuous function whose continuously is differentiable and regular in the convex set : $L = \{x \in R^n : f(x) \leq f(x_1)\}$. Thus, θ is the angle between the search direction d_k and $-g_k$. Then $cos\theta = \frac{-g_{k+1}^T d_k}{\| d_k \| \| g_k \|}$ where d_k satisfies the descentproperty. Also, the sequence is generated by $x_{k+1} = x_k + \alpha_k d_k$ converges to the critical point $g_k = 0$ or $f(x_k) \to -\infty$ or $g_k \to 0$, if $\theta \leq \frac{\pi}{2} - \mu$ and the linear search is Set [12].

2.3. Parametric Conjugate Gradient Method

Conjugate gradient algorithm for the parameter was defined in the same way as the Quasi Newton methods were collected to get on the Broyden or Huang families. These algorithms were defined as $x_{k+1} = x_k + \alpha_k d_k$, and $d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$ where parametric β_k is found for example: [4]

$$\beta_k^{DA} = \frac{g_{k+1}^T (y_k - T s_k)}{s_k^T - y_k}, t > 0 \; constant \quad (6)$$

or [6].

$$\beta_k = \frac{\| g_{k+1} \|_2^2}{\lambda \| g_k \|_2^2 + (1-\lambda) d_k^T y_k}, \lambda \in (0,1) \quad (7)$$

This method has been extended recently by Dai and Yuan for the interval $(-\infty, \infty)$ [7].

2.4. Modified Conjugate Gradient Methods

Conjugate gradient methods are important in this field because they have an implicit relationship with Quasi Newton methods. These methods have a square approximate velocity. If the objective function is quadratic and a linear, the search is exact. Thus, the velocity top linearly in general function. It needs a matrix and a large arithmetic operation to overcome the Quasi Newton methods problems. Also, increasing the convergence velocity of the vectors methods conjugate is required. Thus, it has been a modified version of the conjugation algorithms, for example. Shanno uses the BFGS formula that is Quasi – Newton as in (8). Here, it is $H_k = I_{n\times n}$ and calculated as follows: [17].

$$d_{k+1} = -g_{k+1} + \frac{s_k^T g_{k+1}}{s_k^T y_k} y_k + \frac{y_k^T g_{k+1}}{s_k^T y_k} - \left(1 + \frac{y_k^T y_k}{s_k^T y_k}\right) \frac{s_k^T g_{k+1}}{s_k^T y_k} s_k \quad (8)$$

This method is called Memory Less BFGS.

In 2010 Abbo proposed in his thesis [1] another method of conjugate gradient methods called $V1$–CG :

$$\beta^{v1} = \left(1 - \frac{s_k^T y_k}{y_k^T y_k}\right) \frac{s_k^T g_{k+1}}{s_k^T y_k} \quad (9)$$

$$d_{k+1} = -g_{k+1} + \beta^{v1} s_k \quad (10)$$

Shaker (2015) generalized the modified conjugate gradient [16] which was suggested by Abbo in 2010 called $V1$–CG and reached the following result:

$$\beta^{Gv1} = \frac{y_k^T g_{k+1}}{s_k^T y_k} \frac{y_k^T g_{k+1}}{(1 + \frac{\theta_k}{s_k^T y_k}) y_k^T y_k} \quad (11)$$

$$d_{k+1} = -g_{k+1} + \beta^{v1} s_k \quad (12)$$

2.5. Improved GV1 – CG Algorithm

The objective of improving β^{Gv_1} is derived from a convex quadratic function (conditions imposed on the objective function). In this section, we focus on the GV1– CG Method, where we improve this it and increase the convergence velocity while maintaining global convergence property [1]. This is by generalizing a common function which does not have to be a quadratic function and convex. This generalization is to improve this method and we use developed conjugate property by the world's Dai, Liao [8].

$$d_{k+1}^T y_k = -t g_{k+1}^T s_k \quad (13)$$

Here, the Lipchitz condition $y_k \leq L s_k$ and with $L = 1$, where ^{Gv_2}can be written as following:

$$\beta^{Gv2} = (1-\lambda)\beta^{HS} - \lambda \frac{y_k^T g_{k+1}}{y_k^T y_k} \quad (14)$$

where the descent directions of the algorithm is

$$d_{k+1}^T = g_{k+1}^T + \beta^{Gv2} s_k^T \quad (15$$

Here we multiply both side by y_k and using equation (13) we find λ as following:

$$\begin{aligned}
-tg_{k+1}^T y_k &= -g_{k+1}^T y_k + \beta^{Gv2} s_k^T y_k \\
-tg_{k+1}^T y_k &= -g_{k+1}^T y_k + \left[(1-\lambda)\beta^{HS} - \lambda \frac{y_k^T g_{k+1}}{y_{k+1}^T y_k}\right] s_k^T y_k \\
-tg_{k+1}^T y_k &= -g_{k+1}^T y_k + \left[(1-\lambda)\frac{y_k^T g_{k+1}}{s_k^T y_k} - \lambda \frac{y_k^T g_{k+1}}{y_k^T y_k}\right] s_k^T y_k \\
-tg_{k+1}^T y_k &= -g_{k+1}^T y_k + y_k^T g_{k+1} - \lambda g_{k+1}^T y_k - \lambda \frac{g_{k+1}^T y_k s_k^T y_k}{y_k^T y_k}
\end{aligned}$$

divide both sides by $g_{k+1}^T y_k$ to obtain:

$$\begin{aligned}
-t &= -\lambda - \lambda \frac{s_k^T y_k}{y_k^T y_k} \\
-t &= -\lambda(1 + \frac{s_k^T y_k}{y_k^T y_k}) \\
t &= \lambda(\frac{y_k^T y_k + s_k^T y_k}{y_k^T y_k}) \\
\lambda &= \frac{t y_k^T y_k}{y_k^T y_k + s_k^T y_k} \\
1 - \lambda &= 1 - \frac{t y_k^T y_k}{y_k^T y_k + s_k^T y_k} = \frac{y_k^T y_k + s_k^T y_k - t y_k^T y_k}{y_k^T y_k + s_k^T y_k} = \frac{(1-t) y_k^T y_k + s_k^T y_k}{y_k^T y_k + s_k^T y_k}
\end{aligned}$$

to compensate λ and $1-\lambda$ in equation (14) we find β^{Gv2} as follows:

$$\begin{aligned}
\beta^{Gv2} &= \frac{(1-t) y_k^T y_k + s_k^T y_k}{y_k^T y_k + s_k^T y_k} \frac{y_k^T g_{k+1}}{s_k^T y_k} - \frac{t y_k^T y_k}{y_k^T y_k + s_k^T y_k} \frac{y_k^T g_{k+1}}{y_k^T y_k} \\
\beta^{Gv2} &= y_k^T g_{k+1} \left[\frac{(1-t) y_k^T y_k + s_k^T y_k}{s_k^T y_k (y_k^T y_k + s_k^T y_k)} - \frac{t}{y_k^T y_k + s_k^T y_k}\right] \\
\beta^{Gv2} &= y_k^T g_{k+1} \left[\frac{(1-t) y_k^T y_k + (1-t) s_k^T y_k}{s_k^T y_k (y_k^T y_k + s_k^T y_k)}\right] \\
\beta^{Gv2} &= (1-t)\beta^{HS} \quad (16)
\end{aligned}$$

Here, t is a positive quantity defined as follows:

$$t = \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k} \; and \; that \; 0 < t < 1$$

Therefore, the final value of β^{Gv2} and after compensation for t is as follows:

$$\beta^{Gv2} = \left(1 - \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k}\right) \frac{g_{k+1}^T s_k}{s_k^T y_k} \quad (17)$$

Hint, equation (17) indicates a relationship between β^{Gv2} and $\beta^{HS}$and β^{Gv2} is a quantity multiplied by β^{HS}.

Therefore, we can define the search direction of the new improved algorithm as follows:

$$d_{k+1} = -g_{k+1} + \beta^{Gv2} s_k \quad (18)$$

2.5.1. Descent property for GV2 – CG algorithm

Theorem1. Theorem 1 can be explained as follows. In theorem, x satisfies the Wolfe condition $f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha g_k^T d_k$ and $g_{k+1}^T d_k \geq c_2 g_k^T d_k$ where $0 < c_1 < c_2 < 1$ and g satisfies Lipchitz condition $\forall x$ and that $0 < L < 1$ while $L^2(1-t)\dfrac{(s_k^T g_{k+1})^2}{y_k^T y_k} < g_{k+1}^T g_{k+1}$ true. Then GV2 – CG algorithm fulfills descent property.

Proof . To prove the theorem, we follow the induction. For the initial direction ($k = 1$), there are

$$d_1 = -g_1 \rightarrow d_k^T g_k = -g_1^T g_1 = -\|g_1\|_2^2 < 0$$

Suppose $d_k^T g_k < 0 \ \ \forall k$

Proof $d_{k+1}^T g_{k+1} < 0$ then

$$\begin{aligned} d_k + 1^T g_{k+1} &= -g_{k+1}^T g_{k+1} + \beta^{Gv2} s_k^T g_{k+1} \\ &= -g_{k+1}^T g_{k+1} + \left((1 - \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k}) \frac{g_{k+1}^T y_k}{s_k^T y_k}) \right) s_k^T g_{k+1} \\ &= -g_{k+1}^T g_{k+1} + (y_k^T g_{k+1})(s_k^T g_{k+1}) \left(\frac{1}{s_k^T y_k} - \frac{(g_{k+1}^T s_k)^2}{(s_k^T y_k)^2} \right) \\ &= -g_{k+1}^T g_{k+1} + (y_k^T g_{k+1})(s_k^T g_{k+1}) \left(\frac{s_k^T y_k - (g_{k+1}^T s_k)^2}{(s_k^T y_k)^2} \right) \end{aligned}$$

From Lipchitz Condition $y_k^T g_{k+1} \leq L s_k^T g_{k+1}$

$$d_{k+1}^T g_{k+1} \leq -g_{k+1}^T g_{k+1} + L(s_k^T g_{k+1})^2 \left(\frac{s_k^T y_k - (g_{k+1}^T s_k)^2}{(s_k^T y_k)^2} \right)$$

This, we find $s_k^T y_k > 0$ from Wolfe condition, $y_k^T y_k > 0$ and $y_k^T y_k s_k^T y_k$ and to compensate for them in the above equation we obtain :

$$d_{k+1}^T g_{k+1} \leq -g_{k+1}^T g_{k+1} + L(s_k^T g_{k+1})^2 \left(\frac{L\left(1 - \dfrac{(g_{k+1}^T s_k)^2}{s_k^T y_k}\right)}{y_k^T y_k} \right) < 0 \ \ \forall L \in (0,1)$$

Le that

$$d_{k+1}^T g_{k+1} \leq -g_{k+1}^T g_{k+1} + L(s_k^T g_{k+1})^2 \left(\frac{L(1-t)}{y_k^T y_k} \right) < 0 \ \ \forall L \in (0,1)$$

Where $t = \dfrac{(g_{k+1}^T s_k)^2}{s_k^T y_k}$ and hence $d_{k+1}^T g_{k+1} \leq 0$

Therefore, the search direction resulting from the algorithm GV2 – CG is the descent direction $\forall k$.

2.5.2. Global Convergence of GV2 – CG

Theorem2. In theorem 2, α_k satisfies the Wolfe conditions and f bounded below while d_k is a descent direction $\forall k$i.e. $g_k^T d_k < 0$. Also, g_k satisfies Lipchitz condition in an open set N containing the level set L so $L \equiv \{x; f(x) \leq f(x_k)\}$, where x_k is the starting point then the algorithm (GV2 – CG) is could stop at stationary point i.e. $\|g_k\|_{2=0}$ or $lim_{k\to\infty} inf \|g_k\|_{2=0}$.

Proof . This theorem is proved by contradiction i.e. if theory is not correct $\|g_k\| \neq 0$, there exists a positive constant $\lambda > 0$:

$$\|g_k\| \geq \lambda \quad (19)$$

then

$$\begin{aligned} d_{k+1}^T g_{k+1} &= -g_{k+1}^T g_{k+1} + \beta^{Gv2} s_k^T g_{k+1} \\ &= -g_{k+1}^T g_{k+1} + \left((1 - \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k}) \frac{y_k^T g_{k+1}}{s_k^T y_k}) \right) s_k^T g_{k+1} \\ &= -g_{k+1}^T g_{k+1} + \frac{y_k^T g_{k+1}}{s_k^T y_k} s_k^T g_{k+1} - \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k} \frac{y_k^T g_{k+1}}{s_k^T y_k} s_k^T g_{k+1} \end{aligned}$$

The second (standard) Wolfe condition $g_{k+1}^T d_k \geq c_2 g_k^T d_k$ for $s_k^T g_{k+1}$ and Lipchitz condition are used for:

$$y_k^T y_k \leq L s_k^T y_k \to \frac{L}{y_k^T y_k} \geq \frac{1}{s_k^T y_k}$$

Therefore:

$$\begin{aligned} d_{k+1}^T g_{k+1} &\geq -g_{k+1}^T g_{k+1} + \sigma L \frac{y_k^T g_{k+1}}{y_k^T y_k} s_k^T g_k \\ &\quad - \sigma L \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k} \frac{y_k^T g_{k+1}}{y_k^T y_k} s_k^T g_k \\ &= -g_{k+1}^T g_{k+1} + \sigma L \left(1 - \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k} \right) \frac{y_k^T g_{k+1}}{y_k^T y_k} s_k^T g_k \end{aligned}$$

we have :

$$\begin{aligned} & y_k^T y_k = g_{k+1}^T g_{k+1} - 2 g_{k+1}^T g_{k+1} + g_{k+1}^T g_{k+1} \geq g_{k+1}^T g_{k+1} - g_{k+1}^T g_{k+1} = y_{k+1}^T g_{k+1} \\ & 1 \geq \frac{y_k^T g_{k+1}}{y_k^T y_k} \\ & d_{k+1}^T g_{k+1} \geq -\|g_{k+1}\|^2 + \omega s_k^T g_k \quad (20) \end{aligned}$$

when $\omega = \sigma L \left(1 - \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k} \right)$ divided both side of (20) by $\|g_{k+1}\|_2^2$ and square we get :

$$\frac{1}{\omega^2} \left[\frac{d_{k+1}^T g_{k+1}}{\|g_{k+1}\|_2^2} + 1 \right]^2 \geq \frac{(g_k^T s_k)^2}{\|g_{k+1}\|_2^4}$$

We then multiply both sides by $\|g_{k+1}\|_2^4$ and use the fact

$$(g_k^T s_k)^2 = \|s_k\|_2^2 \|g_{k+1}\|_2^2 cos^2 \theta_k$$

then

$$\frac{1}{\omega^2}\frac{\|g_{k+1}\|_2^4}{\|s_k\|_2^2}\left[\frac{d_{k+1}^T g_{k+1}}{\|g_{k+1}\|_2^2}+1\right]^2 \geq \frac{(g_k^T s_k)^2}{(g_k^T s_k)^2} = \|g_k\|_2^2 cos^2\theta_k \geq \lambda cos^2\theta_k$$

Then the sum for $k \geq 1$ is caclculated:

$$\frac{1}{\omega^2}\sum_{k=1}^{\infty}\frac{\|g_{k+1}\|_2^2}{\|s_k\|_2^2}\left[\frac{d_{k+1}^T g_{k+1}}{\|g_{k+1}\|_2^2}+1\right]^2 \sum_{k=1}^{\infty}\frac{(g_k^T s_k)^2}{(g_k^T s_k)^2}\sum_{k=1}^{\infty}\|g_k\|_2^2 cos^2\theta_k \sum_{k=1}^{\infty}\lambda^2 cos^2\theta_k = \infty$$

Contradiction with Zoutendijk condition [19], therefore$\|g_k\|_2 = 0$ or $\lim_{k\to\infty} inf\|g_k\|_2 = 0$.

3. Numerical Experiments

In this paragraph, the performed FORTRAN implement for the new term algorithm GV2-CG is performed in a set of unconstructive optimizing test problems [3]. Then we selected (10) large-scale test problems forms (see the Appendix). For each function, there is a number of variables for numerical experiments when $n = 100$ and $n = 1000$. Tis is followed by comparing the performance of GV2 – CG algorithm mentioned in equation (17) with the GV1 – CG algorithm mentioned in equation (11) [11].

This algorithms in the Wolfe line search conditions are implemented with $c_1 = 0.001$ and $c_2 = 0.09$ where the initial step size $\alpha_1 = \frac{1}{\|g_1\|}$ and initial guess for other iterations i.e. $(k > 1x); \alpha_k = \alpha_{k-1} * \frac{\|d_{k+1}\|}{\|d_k\|}$.

All cases stopping criterion are $\|d_{k+1}\| < 10^{-6} * max[1, |f_{k+1}|]$ and the maximum number of iteration is 2000.

Our comparison includes the following:

1- NOI: the number of iteration.

2- FGN: number of function and gradient evaluations which are same in these algorithms.

3- LINS: number of calling line search subroutine.

Table (2) shows numerical results for employing (10) test functions of $n = 100$, and **Table (3)** reveals numeric results using (10) test functions when n is1000.

In **Table (4)**, there is a comparison between the percentage of the new algorithm with the GV1 – CG algorithm when $n = 100$ and when NOI; FGN, and LINS are taking over the GV1-CG tools as 100%.

Table (5) depicts the same comparison as table (5) but when n equals 1000.

Table 2: Comparison between (NOI; FGN & LINS) methods for the total of (10) Problems with $n = 100$

Prop.	GV1 – CG			GV2 – CG		
	NOI	FGN	LINS	NOI	FGN	LINS
1	9	24	8	6	19	6
2	34	53	18	27	50	22
3	7	14	6	4	8	3
4	60	100	39	55	101	45
5	13	23	9	10	19	8
6	63	118	54	30	57	25
7	58	87	28	37	57	19
8	219	438	218	42	74	31
9	14	26	11	13	24	10
10	75	106	30	71	110	38
Total	552	989	421	297	519	207

Table 3: Comparison between (NOI; FGN & LINS) methods for the total of (10) Problems with $n = 1000$

Prop.	GV1 – CG			GV2 – CG		
	NOI	FGN	LINS	NOI	FGN	LINS
1	2	5	2	2	5	2
2	69	105	32	14	59	21
3	7	14	6	4	8	3
4	80	148	67	69	127	57
5	10	20	9	19	31	11
6	78	139	60	32	58	24
7	52	82	28	62	97	33
8	519	1024	504	141	247	105
9	15	27	11	14	25	10
10	75	501	195	254	411	156
Total	305	2065	914	611	1068	422

Table 4: $N = 100$

Tools	GV1 – CG	GV2 – CG
NOI	100	53.44
FGN	100	52.24
LINS	100	49.16

Table 5: $N = 1000$

Tools	GV1 – CG	GV2 – CG
NOI	100	53.73
FGN	100	51.71
LINS	100	46.17

4. Discussions

Table (4) shows that when $n = 100$, we take a 100% for the GV1 – CG method. The new method (GV2 – CG) is an improved version for (47) %NOI; (48) % FGN; (51) % LINS.

Also, in $Table(5)$, n is1000 with a 100% for the GV1 – CG method. The proposed method (GV2 – CG) is an improvement for (47) %NOI; (49) % FGN; (54) % LINS.

In general, the results indicate that the new algorithm generally improved largely.

Appendix.

1 - Extended Penalty Function

$$f(x) = \sum_{i=1}^{n-1}(x_{i-1})^2 + (\sum_{j=1}^{n} x_j^2 - 0.25)^2, x_0 = [1, 2, \cdots, n]^T$$

2 - DENSCHNA Function

$$f(x) = \sum_{i=1}^{\frac{n}{2}} x_{2i-1}^4 + \sum_{j=1}^{n}(x_{2i-1} + x_{2i})^2 + (-1 - exp(x_{2i}))^2, x_0 = [8, 8, \cdots, 8]$$

3 - Diagonal 3

$$f(x) = \sum_{i=1}^{n}(exp(x_i) - isin(x_i)), x_0 = [1, 1, \cdots, 1]^T$$

4 – Extending Powell Function

$$f(x) = \sum_{i=1}^{\frac{n}{2}}(x_{2i-1}^2 + x_{2i}^2 + x_{2i-1}x_{2i})^2 + sin^2(x_{2i-1}) + cos^2(x_{2i}), x_0 = [3, -1, 0, 1, \cdots, 3, -1, 0, 1]$$

5 - Extended Himmelblan Function

$$f(x) = (x_i - 5)^2 \sum_{i=1}^{n}(x_1 + x_1 + \cdots + x_{i-1})^2, x_0 = [0.01, 0.01, \cdots, 0.01]$$

6 – Quadratic Diagonal Perturbed

$$f(x) = (\sum_{i=1}^{n} x_i)^2 + \sum_{i=1}^{n} \frac{i}{100} x_i^2, x_0 = [0, 5, 0, 5, \cdots, 0, 5]$$

7 – Generalized Tridiagonal 2 Function

$$f(x) =((5-3x_i-x_i^2)x_i-3x_2+1)^2+\sum_{i=1}^{n-1}(5-3x_i-x_i^2)x_i-x_{i-1}-3x_{i+1}+1)^2$$
$$+(5-3x_i-x_i^2)x_i-x_{n-1}+1)^2, x_i=[-1,-1,\cdots,-1]$$

8 - Extended Cliff Function

$$f(x)=\sum_{i=1}^{\frac{n}{2}}\left(\frac{x_{2i-1}-3}{100}\right)^2-(x_{2i-1}-x_{2i})+exp(20(x_{2i-1}-x_{2i})), x_0=[0,-1,\cdots,0,1]$$

9 – Almost Perturbed Quadratic

$$f(x)=\sum_{i=1}^{n}ix_i^2+\frac{x_{2i-1}-3}{100}(x_i+x_n)^2, x_1=[0.5,\cdots,0.5]$$

10 – ENGVALI Function (CUTE)

$$f(x)=\sum_{i=1}^{n-1}(x_i^2+x_{i+1}^2)^2+\sum_{i=1}^{n-1}(-4x_i+3), x_1=[2,2,\cdots,2]$$

References

[1] K.K. Abbo, Developing of Gradient Algorithm for Solving Unconstrained Non-Linear Problem with Artificial Neural Networks, Doctoral thesis Faculty of Computing and Mathematics University of Mosul Sciences, (2010).
[2] N. Andrei, Conjugate Gradient Algorithms for Molecular Formation under Pair Wise Potential Minimization, Center for Advanced Modeling and Optimization, (2007c).
[3] N. Andrei, An Unconstrained Optimization Test Function collection, Advance Model Optimization, Vol. (10), (2008) 147 -161 .
[4] A.G. Buckley, A combined Conjugate Gradient Quasi – Newton Minimization Algorithm. Math, Prog. 15, (1987) 200 – 210.
[5] E.K.P. Chong and S.H. Zak, An Introduction to Optimization, Jon Wiley & Sons, Inc., Canada (2001).
[6] Y. Dai and Y. Yuan,A three Parameter family of hybrid conjugate gradient method, Mathematics of Computation, 70, (2001).
[7] Y. Dai and Y. Yuan, An Extended Class of Non – Linear CG Methods, to appear, Fifth International Conference of Optimization, Hong – Kong (2010).
[8] Y. Dai and L.Z. Liao, New Conjugacy Conditions and Related Non- Linear Conjugate Gradient Methods, Applied Mathematics and Optimization, Springer – Verlag, 43, New York, USA (2001)87- 101.
[9] R. Fletcher, Practical methods of optimization, A Wiley Inter Science Publication, Johan-Wiley & Sons, Inc., New York (1987).
[10] M.R. Hestenes and E. Stiefel, Method of Conjugate gradient for solving linear systems, Journal of Research of the National Bureau of Standards, Vol. (5), No. (49), (1952) 409-436.
[11] Y. Hiroshi and S. Naoki, Anew Nonlinear Conjugate Gradient Method for Unconstrained Optimization, Journal of the Optimization Research, Vol. (48), No. (4), (2005) 284-296.
[12] J. Kinsella, Course Notes for MS4327 Optimization, http://Jkcray. Maths.ul.ie/ms4327/slides.pdf 2011. INT [1], (2009).
[13] J. Nocedal and J.S. Wright, Numerical optimization series in operation research, 2^{nd} edition, Springer-Verlag, New York (2006).
[14] P. Pedregal, Introduction to Optimization, Springer-Verlag, Inc., New York, USA (2004).
[15] M.J.D. Powell, Restart Procedure for conjugate gradient method, Mathematical Programming, 12, (1977) 241-254.

[16] A.N. Shaker, Development of Modified Conjugate Gradient Algorithm, (2015).
[17] D.F. Shanno, On the convergence of a Conjugate Gradient Algorithm, SIAM. J. Number. Anal, 15 (1978).
[18] W. Sun and y. Yuan, Optimization Theory and Methods, Nonlinear Programming, Springer Science, Business Media, LIC., New York (2006).
[19] G. Zoutendijk, Nonlinear Programming, Computational Methods, In: Integer and Nonlinear Programming, J. Abadie Ed., North – Holland (1970) 37-86.