SOME NONLINEAR CONJUGATE GRADIENT METHODS USING WEAK WOLFE LINE SEARCH FOR LARGE-SCALE OPTIMIZATION FUNCTIONS

Supported by tetfund TERTIARY EDUCATION TRUST FUND

A. A. Ishaq[1], M. A. Ayinde[2]* and F. M. Jimoh[1]
[1]Department of Physical Sciences, Al-Hikmah University, Ilorin, Nigeria
[2]Department of Mathematics, Modibbo Adama University of Technology, Yola, Nigeria
*Corresponding author: abdullahim@mautech.edu.ng

Received: April 25, 2021 **Accepted:** June 30, 2021

Abstract: The nonlinear conjugate gradient method is an effective iterative scheme that is widely employed for the solution of unconstrained large-scale optimization problems. Key to any conjugate gradient algorithm is the calculation of an ideal step length for which numerous strategies have been postulated. In this work, we assessed and compared the execution of the weak. Wolfe line search technique on nine variants of non-linear conjugate gradient methods by carrying out a numerical test. Experiments revealed Dai-Yuan and Conjugate-Descent nonlinear conjugate gradient methods guaranteed faster convergence.

Keywords: Conjugate gradient, Non-linear conjugate gradient methods, optimization functions

Introduction

The nonlinear conjugate gradient method for solving optimization functions considers an unconstrained problem of the form;

$$\min f(x), x \in R^n \quad (1)$$

Where: R^n is n-dimensional Euclidean space and $f: R^n \to R$ is continuously differentiable.

The conjugate gradient method (CGM) was first made known by Hestenes & Stiefel (1952) and afterward expanded to the nonlinear CGM. Broad works have been done on nonlinear CGM by Daniel (1967), Beale (1971), Dixon *et al*. (1985), Yabe & Takano (2004), Hager & Zhang (2005), Yuan & Lu (2006), Andrei (2011), Ishaq *et al*. (2020) just to mention a few. The nonlinear CGMs constitute adiligent choice for effectively tackling (1), paticulally when the dimension n is huge, due to the uncompounded of its analysis, exceptionallymoo memory necessity, fast convergence and its great numerical execution, for engineers and mathematicians (Wang *et al*., 2020; Sun *et al*., 2018; Jinhong & Genjiao, 2013).

A nonlinear CGM creates a sequence of points $\{x^{(k)}\}$ with $k \geq 0$, by guessing astarting point $x^{(0)} \in R^n$, and utilizing the recurrence relation;

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)} \quad (2)$$

Where: α_k is the step-length that can be computed using various step-length techniques and $d^{(k)}$ is the search direction and can be generated using the relation:

$$d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}, d^{(0)} - g^{(0)} \quad (3)$$

Where: $g^{(k)} = -\nabla f(x^{(k)})$, which is the gradient of f at $x^{(k)}$ and $\beta_k$stipulates differentnonlinear CGM which corresponds to a different choice of β_k (Zhang *et al*., 2019; Hager & Zhang, 2006). The following are Some well-known β_k parameters:

$$\beta_k^{HS} = \frac{g^{(k+1)T}Y^{(k)}}{g^{(k)T}g^{(k)}}, \text{Hestenes \& } Stiefel(1952)(HS) \quad (4)$$

$$\beta_k^{FR} = \frac{\|g^{(k+1)}\|^2}{\|g^{(k)}\|^2}, \text{Fletcher \& } Reeves\ (1964)(FR) \quad (5)$$

$$\beta_k^{PR} = \frac{g^{(k+1)T}y^{(k)}}{\|g^{(k)}\|^2}, \text{Polak \& } Ribiere\ (1969)(PR) \quad (6)$$

$$\beta_k^{CD} = -\frac{\|g^{(k+1)}\|^2}{g^{(k)T}d^{(k)}}, \text{Fletcher (1987)(CD)} \quad (7)$$

$$\beta_k^{DY} = \frac{\|g^{(k+1)}\|^2}{y^{(k)T}d^{(k)}}, \text{Dai \& } Yuan\ (2000)(DY) \quad (8)$$

$$\beta_k^{LS} = -\frac{g^{(k+1)T}y^{(k)}}{g^{(k)T}d^{(k)}}, \text{Liu \& } Storey\ (1992)(LS) \quad (9)$$

$$\beta_k^{HZ} = \left(y^{(k)} - 2d^{(k)}\frac{\|y^{(k)}\|^2}{y^{(k)T}d^{(k)}}\right)^{(T)} \frac{g^{(k+1)T}d^{(k)}}{y^{(k)T}d^{(k)}},$$

$$\text{Hager \& } Zhang\ (2005)(\text{HZ}) \quad (10)$$

$$\beta_k^{BAN} = \frac{g^{(k+1)T}y^{(k)}}{g^{(k)T}y^{(k)}}, \text{Bamigbola etal. (2010)(BAN)} \quad (11)$$

$$\beta_k^{GSC} = \frac{g^{(k+1)T}g^{(k)}}{g^{(k)T}d^{(k)}}, \text{Gradient Search Conjugacy (GSC)} \quad (12)$$

Where: $y^{(k)} = (g^{(k+1)} - g^{(k)})$

Also one eminanthighlight of the nonlinear CGM is the inclusion of the line search procedures in its calculation. For anentirely quadratic function, the CG algorithm with exact or optimum line search merges limitedly (Rockafeller, 1970). Minimizing a non-quadratic (nonlinear) function, it is more suitable and cost-proficient to employ the inexact line search techniques in spite of the factthat in several cases, accuracy is yielded for global convergence.

It is more ideal in large scale issues to utilize β_k that does not need the assessment of the Hessian matrix which regularly require high computer storage. In the case of the optimum line search technique, similarities in the strategies can be instituted for strongly convex quadratic functions (Ishaq *et al*., 2020; Adeleke *et al*., 2013). For a distinctive class of functions, this importance highlight is misplaced. In such occurence, the inexact line search strategies are utilized (Liu *et al*., 2020).

Line search

This is a vital step in the conjugate gradient algorithm when solving unconstrained optimization problems. In every line search, the decision of procedure for determining α_k influences both the convergence and the speed of convergence of the algorithm. The two types of line search procedures basically in use are exact or optimum and Inexact line search.

Exact line search

Every line search rule aims to obtain a positive step-length α_k along the search direction $d^{(k)}$ to ensure an improving rate of convergence. Then to accomplish this, we first set $\alpha_k = \alpha^*$, such that,

$$\alpha^* = argmin_{\alpha>0} f(x^{(k)} + \alpha d^{(k)}) \quad (13)$$

i.e., α^* is the value of α_k which minimizes$f$ along $d^{(k)}$.

Therefore, α^* in (13) can be calculated by obtaining the solution of the following equation,

$$\frac{d}{d\alpha} f(x^{(k)} + \alpha d^{(k)}) = 0 \quad (14)$$

The idealadopted in (14) gives an exact or optimum value for α_k and is called an exact or optimum line search. However, Sun & Yuan (2006) stated that the exact or optimum line search is cost expensive, particulally when an initial point is distant from the solution of the problem during the actual computation.

Inexact line search

It is vital to note that exact or optimum line search is very costly to carry out, as a result of the limitation of the exact or optimum line search there is a need for a line search technique that can recognize a step-length that produces a significant decrease in the value of f at a minimal cost. And for nonlinear unconstrained optimization problems, inexact line search rules are cost-efficient and more accurate to work with. Below are the framework of inexact line search rules:

•create a measure that guarantees the step-length α is neither as well long nor as wellbrief;

• pick a decent starting step-length to begin the algorithm; and

• develop a grouping of overhauls that fulfil the model characterized in the first framework after each step.

A lot has been done on the detailing of diverse criteria by several researchers. Among these are Goldstein (1965), Armijo (1966), Wolfe (1969), Wolfe (1971), Powell (1975), Dennis & Schnabel (1983), Boggs & Schnabel (1987), Fletcher (1987), Potra & Shi (1995), Hager & Zhang (2005), Andrei (2011), etc. The most widely used inexact line search techniques are Wolfe line search procedures (Andrei, 2011). In what takes after, we provide more considerations to Wolfe line searches.

Weak Wolfe inexact line search rule

Wolfe (1969) first proposed that the step-size α_k is considered optimal if it fulfills the following criteria

$f(x^{(k)}+\alpha_k d^{(k)}) \le f(x^{(k)}) + \nu\alpha_k g^{(k)T} d^{(k)}$ (15)

$g(x^{(k)}+\alpha_k d^{(k)})^T \ge \vartheta g^{(k)T} d^k$ (16)

and $\varphi(\alpha_k) = f(x^{(k)}+\alpha_k d^{(k)})$ from where $0 \le \nu \le \vartheta \le 1$. The first inequality in (15) guarantees that the function decreased sufficiently whereas (16) avoids the steps from being as well little.

Algorithm: Weak Wolfe Inexact Line Search;

• Choose $\nu \in (0,1)$ and $\vartheta = 0.75, \gamma = 0.5$, set $\alpha = 1$

• If $f(x^{(k)}+\alpha_k d^{(k)}) \ge f(x^{(k)}) + \nu\alpha_k g^{(k)T} d^{(k)}$ and $|g(x^{(k)}+\alpha+kd^{(k)})^T| \le \vartheta|g^{(k)T}d^{(k)}|$, take $\alpha = \gamma\alpha$

• Terminate loop with $\alpha_k = \alpha$

Strong Wolfe inexact line search rule

Wolfe (1971) noticed that there are cases where α_k may fulfil the general Wolfe condition without necessarily minimizing the function $\varphi(\alpha_k)$. As such, a stronger strict two-sided measure is placed on the gradient of φ. This forces α_k to lie at last in the neighbourhood of a local minimizer of φ. This measure is called the strong Wolfe conditions and can be obtained by the following equations

$f(x^{(k)}+\alpha_k d^{(k)}) \le f(x^{(k)}) + \nu\alpha_k g^{(k)T} d^{(k)}$ (17)

$|g(x^{(k)}+\alpha_k d^{(k)})^T| \le \vartheta|g^{(k)T}d^{(k)}|$ (18)

$0 \le \nu \le \vartheta \le 1$

Approximate Wolfe inexact line search rule

Hager & Zhang (2005) developed a new inexact line search rule called an approximate Wolfe line search. This accepts any step-size $\alpha_k > 0$ if and only if it satisfies the following conditions:

$(2\xi-1)\varphi'(0) \ge \varphi'(\alpha_k) \ge \upsilon\varphi'(0)$ (19)

Where $\varphi(\alpha_k) = f(x^{(k)}+\alpha_k d^{(k)})$ and $0 < \gamma < \frac{1}{2} < \upsilon < 1$.

Numerical experiments

Nonlinear CGM Algorithm

The algorithm below is a summary of the steps necessary for the implementation of the nonlinear CGM:

Algorithm: CGM Algorithm

• Pick the starting point, $x^{(0)} \in \mathbb{R}^n, \epsilon \ge 0$ (this is a small number called tolerance) and set $d^{(0)} = -g^{(0)}$.

• Terminate process if $||g^{(0)}|| \le \epsilon$, otherwise, go to the next step.

• Compute step-size $\alpha_k$by an efficient step size rule:

• Set $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$; if $||g^{(k+1)}|| \le \epsilon$, then stop, otherwise, go to the next step

• Compute the search direction $d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}$. **Where** β_k is given by equation $(4-12)$.

• Set $k = k+1$, and go to step 3.

Computational details

We intend to execute the experiments using nonlinear CGMs to minimize large-scale unconstrained optimization problems. To achieve this, thirty nonlinear unconstrained optimization test functions by Andrei (2008) were used as numerical examples. Algorithm 3.1 was implemented with Algorithms 2.1 for a weak Wolfe inexact line search algorithm. It sufficient to say here that the dimension of f was generally taken to be very huge (5000 and 10000). Also, we assumed $\epsilon = 10^{-6}$ forg^* ($g^*$is the gradient at the optimum value of the objective function f).

Computational examples

The following optimization test functions and initial values obtained from Andrei (1) are used as computational examples:

1. **Diagonal 1 function**

$$f(x) = \sum_{i=1}^{n} (\exp(x_i) - ix_i), x_0 = [1/n, 1/n, \dots, 1/n].$$

2. **Full Hessian FH2 function**

$$f(x) = (x_1-5)^2 + \sum_{i=2}^{n} (x_1+x_2+\cdots+x_i-1)^2, x_0 = [0.01, 0.01, \dots, 0.01].$$

3. **TRIDIA function(cute)**

$$f(x) = \gamma(\delta x_1 - 1)^2 + \sum_{i=2}^{n} i(\alpha x_i - \beta x_{i-1})^2,$$
$$x_0 = [1,1,\dots,1],\ \ \alpha = 2,\ \ \beta = 1,\ \ \gamma = 1,\ \ \delta = 1.$$

4. **Partial perturbed quadratic function**

$$f(x) = x_1^2 + \sum_{i=1}^{n} \left(ix_i^2 + \frac{1}{100}(x_1+x_2+\cdots+x_i)^2\right), x_0 = [0.5, 0.5, \dots, 0.5].$$

5. **Power function(cute)**

$$f(x) = \sum_{i=1}^{n} (ix_i)^2, x_0 = [1,1,\dots,1].$$

6. **EXPLIN2 function(cute)**

$$f(x) = \sum_{i=1}^{m} \exp\left(\frac{ix_ix_{i+1}}{10m}\right) - 10\sum_{i=1}^{n} (ix_i), x_0 = [0,0,\dots,0].$$

7. **VARDIM function(cute)**

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

8. **Variably dimensioned function**

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

9. INDEF function

$$f(x) = \sum_{i=1}^{n} (x_i) + \sum_{i=2}^{n-1} (\alpha\cos(2x_i - x_n - x_1)),$$
$$x_0 = \left[\frac{1}{n+1}, \frac{2}{n+1}, \dots, \frac{n}{n+1}\right], \quad \alpha = 0.5.$$

10. Liarwhd function

$$f(x) = \sum_{i=1}^{n} \left(4(x_i^2 - x_1)^2\right) + \sum_{i=1}^{n} (x_i - 1)^2, x_0 = [4,4, \dots, 4].$$

11. McCormick function

$$f(x) = \sum_{i=1}^{n-1} (-1.5x_i + 2.5x_{i+1} + 1 + (x_i - x_{i+1})^2 + \sin(x_i + x_{i+1})), x_0 = [1,1, \dots, 1].$$

12. NONDIA function

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} 100(x_1 - x_{i-1}^2)^2, x_0 = [-1, -1, \dots, -1].$$

13. NONDQUAR function

$$f(x) = \sum_{i=1}^{n-2} (x_i + x_{i+1} + x_n)^4 + (x_1 - x_2)^2 + (x_{n-1} + x_n)^2, x_0 = [-1, -1, \dots, -1].$$

14. NONSCOMP function

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} 4(x_i - x_{i-1}^2)^2, x_0 = [3,3, \dots, 3].$$

15. maratosb extended function

$$f(x) = x_1 + \sum_{i=1}^{n-1} \frac{(x_i^2 + x_{i+1}^2 - 1)^2}{0.000001}, x_0 = [0,0, \dots, 0].$$

16. MDHOLE extended function

$$f(x) = x_1 + \sum_{i=1}^{n} \frac{(\sin x_i - 10)^2}{0.01}, x_0 = [10,10, \dots, 10].$$

17. Logros extended function

$$f(x) = \sum_{i=1}^{n} \log\left(1 + 10000(1 - x_i^2)^2 + (1 - x_i)^2\right), x_0 = [-1.2, -1.2, \dots, -1.2].$$

18. mexhat extended function

$$f(x) = -2(x_1 - 1)^2 + 10000,$$
$$\sum_{i=1}^{n-1} \left((x_i - 1)^2 + \frac{(x_{i+1} - x_i^2)^2}{10000} - 0.02\right)^2$$
$$x_0 = [1 - 0.4 \times 1, 1 - 0.4 \times 2, \dots, 1 - 0.4 \times n].$$

19. nasty extended function

$$f(x) = 0.5 \sum_{i=1}^{n-1} \left((1.0e^{10}x_i)^2 + x_{i+1}^2\right),$$
$$x_0 = [e^{-30}, 1, \dots, (2 - n)e^{-30} - 1 + n].$$

20. Biggs B1 function

$$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 + (1 - x_n)^2, x_0 = [0,0, \dots, 0].$$

21. Extended Beale function

$$f(x) = \sum_{i=1}^{n/2} (1.5 - x_{2i-1}(1 - x_{2i}))^2 + (2.25 - x_{2i-1}(1 - x_{2i}^2))^2 + (2.625 - x_{2i-1}(1 - x_{2i}^3))^2,$$
$$x_0 = [1, 0.8, \dots, 1, 0.8].$$

22. Extended Cliff function

$$f(x) = \sum_{i=1}^{n/2} ((x_{2i-1} - 3)0.01)^2 - (x_{2i-1}x_{2i}) + \exp(20(x_{2i-1} - x_{2i})),$$
$$x_0 = [0, -1, \dots, 0, -1].$$

23. Extended DenschnA function

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1}^4) + (x_{2i-1} + x_{2i})^2 + (-1 + \exp(x_{2i}))^2, x_0 = [1,1, \dots, 1].$$

24. Extended DenschnB function

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1} - 2)^2 + (x_{2i-1} - 2)^2 x_{2i}^2 + (x_{2i} + 1)^2, x_0 = [1,1, \dots, 1].$$

25. Extended DenschnF function

$$f(x) = \sum_{i=1}^{n/2} (2(x_{2i-1} + x_{2i})^2 + (x_{2i-1} - x_{2i})^2 - 8)^2 + (5x_{2i-1}^2 + (x_{2i} - 3)^2 - 9)^2,$$
$$x_0 = [2, 0, \dots, 2, 0].$$

26. Diagonal 7 function

$$f(x) = \sum_{i=1}^{n} \exp(x_i) - 2x_i - x_i^2, x_0 = [1,1, \dots, 1].$$

27. Diagonal 8 function

$$f(x) = \sum_{i=1}^{n} x_i \exp(x_i) - 2x_i - x_i^2, x_0 = [1,1, \dots, 1].$$

28. Brownbs function

$$f(x) = \sum_{i=1}^{n-1} (x_i - 1000000)^2 + \sum_{i=1}^{n-1} (x_{i+1} - 0.000002)^2 + \sum_{i=1}^{n-1} (x_i x_{i+1} - 2)^2, x_0 = [1,1, \dots, 1].$$

29. DQDRTIC function

$$f(x) = \sum_{i=1}^{n-2} (x_i^2 + 100x_{i+1}^2 + 100x_{i+2}^2), x_0 = [3,3, \dots, 3].$$

30. HimmelBG function

$$f(x) = \sum_{i=1}^{n/2} (2x_{2i-1}^2 + 3x_{2i}^2)\exp(-x_{2i-1} - x_{2i}), x_0 = [1.5, 1.5, \dots, 1.5].$$

Computational results

The CGM Algorithm 3.1 was implemented with the Algorithms 2.1 for nine variant of nonlinear CGMs stated in (4)-(12) using MATLAB 1.8.0347 [R2009a] on an HP laptop computer 620 with processor Pentium (R) Dual-core CPU T4500 @2.30GB to solve all the thirty computational examples above, and the results obtained were given in Tables 1 – 2 denotedby the following:

Dim (dimension), ITR (number of iterations), CPU (time taken), AWLS (strong Wolfe line search), AVE (average).

Table 1: CPU results for WWL

S/N	Test Function	DIM	BAN	FR	PR	HS	CD	DY	LS	HZ	GSC
1	Diagonal 1 Function	5000	0.0689	0.0652	0.0425	0.0491	0.0465	0.0427	0.0424	0.0576	0.0458
		10000	0.1190	0.1147	0.1700	0.1378	0.1174	0.1227	0.1266	0.1670	0.1113
2	Full Hessian FH2 Function	5000	69.854	61.070	57.622	87.899	68.403	87.755	54.567	55.738	54.614
		10000	161.93	169.11	170.02	187.82	181.41	154.36	153.25	148.84	149.44
3	TRIDIA Function (cute)	5000	0.0232	0.0219	0.0262	0.0266	0.0262	0.0224	0.0208	0.0196	0.0197
		10000	0.0170	0.0192	0.0186	0.0225	0.0193	0.0223	0.0197	0.0163	0.0132
4	Partial Perturbed Quadratic Function	5000	39.973	40.822	43.305	40.436	38.894	39.584	38.009	37.570	37.213
		10000	163.94	163.21	160.90	162.46	180.44	152.76	150.58	150.99	150.36
5	POWER Function (cute)	5000	0.0268	0.0091	0.0119	0.0124	0.0136	0.0127	0.0136	0.0152	0.0134
		10000	0.0062	0.0063	0.0080	0.0066	0.0077	0.0079	0.0080	0.0082	0.0072
6	EXPLIN2 Function (cute)	5000	0.0485	0.0511	0.0550	0.0556	0.0639	0.0507	0.0504	0.0927	0.0502
		10000	0.1700	0.1284	0.1384	0.1287	0.1408	0.1658	0.1386	0.2051	0.1365
7	VARDIM Function (cute)	5000	0.0146	0.0122	0.0148	0.0109	0.0130	0.0160	0.0159	0.0173	0.0163
		10000	0.0103	0.0094	0.0093	0.0104	0.0120	0.0115	0.0115	0.0095	0.0084
8	Variably Dimensioned Function	5000	0.0135	0.0135	0.0156	0.0162	0.0146	0.0160	0.0121	0.0146	0.0156
		10000	0.0100	0.0104	0.0099	0.0109	0.0088	0.0115	0.0100	0.0112	0.0092
9	INDEF Function	5000	39.863	38.387	15.835	15.762	15.700	18.603	15.372	41.115	16.217
		10000	98.284	33.790	26.718	21.555	36.009	40.098	21.882	61.369	22.562
10	Liarwhd Function	5000	0.0317	0.0313	0.0258	0.0233	0.0259	0.0218	0.0259	0.0251	0.0229
		10000	0.0265	0.0282	0.0261	0.0238	0.0230	0.0211	0.0164	0.0197	0.0189
11	mccormck Function	5000	0.2198	0.1081	0.1023	0.3740	0.1345	0.3241	0.1499	0.2150	0.2063
		10000	0.2625	0.2041	0.1709	0.6315	0.1693	0.4811	0.2234	0.2946	0.1813
12	NONDIA Function	5000	0.0274	0.0265	0.0252	0.0248	0.0266	0.0213	0.0217	0.0255	0.0236
		10000	0.0239	0.0190	0.0242	0.0202	0.0215	0.0208	0.0155	0.0192	0.0160
13	NONDQUAR Function	5000	0.0208	0.0266	0.0484	0.0180	0.0166	0.0191	0.0177	0.0104	0.0202
		10000	0.0165	0.0147	0.0153	0.0263	0.0166	0.0193	0.0174	0.0210	0.0173
14	NONSCOMP Function	5000	0.0284	0.0269	0.0260	0.0275	0.0269	0.0268	0.0234	0.0206	0.0199
		10000	0.0223	0.0218	0.0208	0.0181	0.0222	0.0241	0.0226	0.0201	0.0277
15	maratosb Extended Function	5000	0.0430	0.0366	0.0523	0.0229	0.0250	0.0229	0.0421	0.0287	0.0320
		10000	0.0208	0.0377	0.0420	0.0231	0.0231	0.0219	0.0454	0.0255	0.0319
16	MDHOLE Extended Function	5000	17.697	35.614	43.064	18.570	19.475	19.045	23.509	140.716	28.393
		10000	22.985	67.271	69.441	63.042	25.865	70.391	28.440	24.074	215.707
17	Logros Extended Function	5000	0.0954	32.2777	62.5864	0.0347	16.3779	16.1946	16.8946	16.7731	16.4521
		10000	0.6724	74.8414	21.8720	0.0950	21.8957	21.8149	22.0654	22.6362	21.7901
18	mexhat Extended Function	5000	0.0183	0.0201	0.0183	0.0178	0.0211	0.0209	0.0173	0.0153	0.0710
		10000	0.0153	0.0154	0.0178	0.0164	0.0171	0.0168	0.0161	0.0282	0.0150
19	nasty Extended Function	5000	0.0119	0.0169	0.0171	0.0168	0.0141	0.0127	0.0168	0.0164	0.0126
		10000	0.0110	0.0117	0.0118	0.0084	0.0177	0.0116	0.0100	0.0103	0.0110
20	Biggs B1 Function	5000	0.4311	0.1008	0.0831	0.3210	0.1376	0.3103	0.0760	0.1083	0.1373
		10000	0.3477	0.0951	0.0904	0.3623	0.1438	0.3513	0.0836	0.1320	0.1494
21	Extended Beale Function	5000	0.0335	0.0664	0.0630	0.0361	0.0374	0.0398	0.0373	0.0414	0.0345
		10000	0.0338	0.0349	0.0796	0.0430	0.0481	0.0429	0.0419	0.0454	0.0438
22	Extended Cliff Function	5000	0.6448	0.1772	0.3905	0.0672	1.4385	0.0800	0.2709	0.0695	0.0521
		10000	0.6329	0.3058	0.4897	0.0676	2.2862	0.1829	0.4248	0.0868	0.0321
23	Extended DenschnA Function	5000	0.0423	0.0407	0.0375	0.0530	0.0550	0.0541	0.0546	0.0636	0.0456
		10000	0.0505	0.0439	0.0431	0.0567	0.0961	0.0579	0.1004	0.1382	0.0478
24	Extended DenschnB Function	5000	0.1298	0.0946	0.1010	0.8269	0.0623	0.0583	0.1051	0.1452	0.0586
		10000	0.1936	0.1257	0.1351	0.2784	0.0722	0.0623	0.1495	0.1244	0.0730
25	Extended DenschnF Function	5000	0.0262	0.0556	0.0295	0.0316	0.0303	0.0303	0.0315	0.0358	0.0350
		10000	0.0253	0.0306	0.0482	0.0311	0.0324	0.0349	0.0328	0.0359	0.0380
26	Diagonal 7 Function	5000	0.3199	0.0792	0.0766	76.3290	0.9540	6.3645	0.3648	1.0721	50.5751
		10000	0.3751	0.1746	0.1531	18.9577	1.0303	6.7061	0.2907	0.9493	123.660
27	Diagonal 8 Function	5000	30.394	0.142	0.220	0.750	0.390	0.640	0.101	0.196	0.250
		10000	9.1197	0.2254	0.3246	3.9503	1.7793	0.5708	0.1167	0.1790	0.2909
28	Brownbs Function	5000	0.0285	0.0264	0.0254	0.0245	0.0252	0.0245	0.0291	0.0266	0.0266
		10000	0.0244	0.0243	0.0271	0.0209	0.0258	0.0219	0.0274	0.0214	0.0218
29	DQDRTIC Function	5000	0.0303	0.0423	0.0670	0.0493	0.0380	0.0617	0.0341	0.0353	0.0421
		10000	0.0340	0.0556	0.0601	0.0444	0.0391	0.0493	0.0329	0.0312	0.0436
30	HimmelBG Function	5000	16.731	16.9740	16.5350	16.9583	0.0363	0.4110	79.6854	0.0859	0.0540
		10000	23.255	22.6883	23.2200	23.0485	0.0352	0.6261	58.0938	0.0792	36.5200

Table 2: ITR results for WWL

S/N	Test Function	DIM	BAN	FR	PR	HS	CD	DY	LS	HZ	GSC
1	Diagonal 1	5000	2	2	2	2	2	2	2	2	2
	Function	10000	2	2	2	2	2	2	2	2	2
2	Full Hessian FH2	5000	4	3	3	4	3	4	3	3	3
	Function	10000	2	2	2	2	2	2	2	2	2
3	TRIDIA	5000	2	2	2	2	2	2	2	2	2
	Function(cute)	10000	2	2	2	2	2	2	2	2	2
4	Partial Perturbed	5000	2	2	2	2	2	2	2	2	2
	Quadratic Function	10000	2	2	2	2	2	2	2	2	2
5	POWER	5000	1	1	1	1	1	1	1	1	1
	Function(cute)	10000	1	1	1	1	1	1	1	1	1
6	EXPLIN2	5000	2	2	2	2	2	2	2	2	2
	Function (cute)	10000	2	2	2	2	2	2	2	2	2
7	VARDIM	5000	1	1	1	1	1	1	1	1	1
	Function (cute)	10000	1	1	1	1	1	1	1	1	1
8	Variably	5000	1	1	1	1	1	1	1	1	1
	Dimensioned Function	10000	1	1	1	1	1	1	1	1	1
9	INDEF Function	5000	2000	2000	2000	2000	2000	2000	2000	2000	2000
		10000	2000	2000	2000	2000	2000	2000	2000	2000	2000
10	Liarwhd Function	5000	2	2	2	2	2	2	2	2	2
		10000	2	2	2	2	2	2	2	2	2
11	mccormck	5000	11	7	7	22	13	22	9	13	13
	Function	10000	12	9	8	23	13	21	9	11	13
12	NONDIA	5000	2	2	2	2	2	2	2	2	2
	Function	10000	2	2	2	2	2	2	2	2	2
13	NONDQUAR	5000	1	1	1	1	1	1	1	1	1
	Function	10000	1	1	1	1	1	1	1	1	1
14	NONSCOMP	5000	2	2	2	2	2	2	2	2	2
	Function	10000	2	2	2	2	2	2	2	2	2
15	maratosb	5000	2	4	4	2	2	2	4	2	3
	Extended Function	10000	2	4	4	2	2	2	4	2	3
16	MDHOLE	5000	2000	2000	2000	2000	2000	2000	2000	2000	2000
	extended function	10000	2000	2000	1919	2000	2000	2000	2000	2000	2000
17	Logros Extended	5000	3	2000	2000	3	2000	2000	2000	2000	2000
	Function	10000	55	2000	2000	8	2000	2000	2000	2000	2000
18	mexhat Extended	5000	1	1	1	1	1	1	1	1	1
	Function	10000	1	1	1	1	1	1	1	1	1
19	nasty Extended	5000	1	1	1	1	1	1	1	1	1
	Function	10000	1	1	1	1	1	1	1	1	1
20	Biggs B1	5000	25	6	6	25	11	25	7	11	13
	Function	10000	25	6	6	25	11	25	7	11	13
21	Extended Beale	5000	2	2	2	2	2	2	2	2	2
	Function	10000	2	2	2	2	2	2	2	2	2
22	Extended Cliff	5000	46	9	25	4	115	4	25	8	3
	Function	10000	46	9	25	4	115	4	25	8	3
23	Extended	5000	4	3	3	4	3	4	3	3	3
	DenschnA Function	10000	4	3	3	4	3	4	3	3	3
24	Extended	5000	8	6	8	34	5	5	8	7	5
	DenschnB Function	10000	8	6	8	14	5	5	8	6	5
25	Extended	5000	2	2	2	2	2	2	2	2	2
	DenschnF Function	10000	2	2	2	2	2	2	2	2	2
26	Diagonal 7	5000	42	4	4	2000	13	86	4	14	2000
	Function	10000	42	4	4	2000	13	88	4	14	2000
27	Diagonal 8	5000	1959	5	6	36	20	7	7	15	28
	Function	10000	312	5	6	201	20	7	7	15	27
28	Brownbs	5000	2	2	2	2	2	2	2	2	2
	Function	10000	2	2	2	2	2	2	2	2	2
29	DQDRTIC	5000	4	3	3	4	3	5	3	3	3
	Function	10000	4	3	3	4	3	5	3	3	3
30	HimmelBG	5000	2000	2000	2000	2000	3	44	2000	2	6
	Function	10000	2000	2000	2000	2000	3	54	2000	2	2000

Table 3: Inference on CGMs with WWL by CPU

	BAN	FR	PRP	HS	CD	DY	LS	HZ	GSC
WWL-CPU-AVE	11.66	12.65	11.91	12.36	10.24	10.65	11.10	11.75	15.44
WWL-ITR-AVE	244.53	269.22	268.48	274.62	207.22	207.95	269.95	203.50	303.40

Remarks on computational results

For superior depiction and understanding of the Tables 1 – 2, the performance inference of all the nonlinear CGMs fo WWL are given in the Table 3.

It is clear from Table 3 that judging by CPU, the nonlinear CGMs of CD, DY and HZ perform better than other CG methods, while DY, BAN and CD perform better than other CG methods judging by ITR. Therefore, we can remark on the result that DY and CD performed better than all other CG methods judging by the CPU and ITR average.

Conclusion

In this work, nine (9) variants of CGMs namely BAN, FR, PR, HS, CD, DY, LS, HZ, GSC methods were considered in solving thirty unconstrained optimization test problems using weak Wolfe line search.DY and CDnonlinear CGMs gave better results compared to the rest of the nonlinear CGMs considered.

Conflict of Interest

Authors have declared that there is no conflict of interest reported in this work.

References

Adeleke OJ, Aderemi OA, Omoregbe NI & Adekunle RA 2013. Numerical comparison of line search criteria in nonlinear conjugate gradient algorithms. *J. Math. and Stat. Stud.,* 2(1): 14-15.

Andrei N 2008. An unconstrained optimizationtest functions collection. *J. Advan. Modeling and Optimization,* 10(1): 149-159.

Andrei N 2011. Open problems in non-linear conjugate gradient algorithms for unconstrained optimization. *Bull*. 34(2): 319-330.

Armijo L 1966. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific J. Math.,* 16: 1-3.

Bamigbola OM, Ali M & Nwaeze E 2010. An efficient and convergent method for unconstrained nonlinear optimization. *Proceedings of Int. Congress of Mathematicians*. Hyderabad, India.

Beale EML 1971. A derivation of conjugate gradients in numerical methods for nonlinear optimization. *Academic Press, London*, pp. 39-43.

Boggs PT & Schnabel RB 1987. Numerical Optimization. *Lecture Note for a Short Course.*

Dai Y & Yuan Y 2000. A nonlinear conjugate gradient with a strong global convergence properties. SIAM *Journal on Optimization*, 10: 177-182.

Daniel JW 1967. The conjugate gradient method for linear and nonlinear operator equations. SIAM *Journal on Numerical Analysis,* 4: 10-26.

Dennis JE & Schnabel RB 1983. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. *Prentice Hall Series in Computational Mathematics*, Englewood Cliffs, NJ.

Dixon LCW, Ducksbury PG & Singh P 1985. A New Three-term Conjugate Gradient Method. *J. Optimization Theory Applic.,* 47(3): 285-300.

Fletcher R & Reeves CM 1964. Function Minimization by Conjugate Gradients. *Computer Journal*, 7(2).

Fletcher R 1987. Practical Methods of Optimization. *John Wiley and Sons,* New York.

Goldstein AA 1965. On steepest descent. *Journal of Industrial Application,* 3: 147-151.

Hager WW & Zhang H 2005. A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM *Journal of Optimization,* 16(1): 170-192.

Hager W & Zhang H 2006. A survey of nonlinear conjugate gradient methods. *Pacific J. Optimization,* 2(1): 35-58.

Hestenes MR & Stiefel E 1952. Method of conjugate gradient for solving linear equations. *J. Res. Nat. Bur. Stand.* 49:

Ishaq AA, Latunde T & Akande KB 2020. A Step-Length Formulafor Conjugate Gradient Methods. *Malysian Journal of Computing,* 5(1): 403-413.

Ishaq AA, Latunde T & Jimoh FM 2020. An optimum line search for unconstrained non-polynomial test functions using nonlinear conjugate gradient methods. *Int. J. Math. Modelling & Comp.,* 10(40): 20-25.

Jinhong H & Genjiao Z 2013. A conjugate gradient method without line search and the convergence analysis. *Fourth Int. Conf. Emerg. Intel. Data and Web Techn.,* pp. 37-86.

Wang L, Cao M, Xing F & Yang Y 2020. The new spectral conjugate gradient method for large-scale unconstrained optimization. *J. Inequalities and Applic.,* 111.

Liu Y & Storey C 1992. Efficient generalized conjugate gradient algorithms. *J. Optimization Theory and Applic*., 69: 129-137.

Liu M, Ma G & Yin J 2020. Two new conjugate gradient methods for Unconstrained Optimization. *Hindawi Complexity*, Article ID 9720653.

Polak E & Ribiere G 1969. Note sur la Convergence de Directions Conjugees. *Rev. Francaise Informat Recherche Operationelle,* 3(16): 35-43.

Potra F A & Shi Y 1995. Efficient line search algorithm for unconstrained optimization. *J. Optimization Theory Applic.,* 85(3): 677-704.

Powell MJD 1975. Some Global Convergence Properties of a Variable Metric Algorithm for Minimization without Exact Line Searches in Nonlinear Programming, pp. 53-72.

Rockafellar R T 1970. Convex analysis. *Princeton University Press*, Princeton.

Sun W & Yuan Y 2006. *Optimization Theory and Methods*. Springer.

Sun Z, Li H, Wang J & Tian Y 2018. Two Modified Spectral Conjugate Gradient Methods and their Global Convergence for Unconstrained Optimization. *Int. J. Comp. Math.,* 95(10): 2082-2099.

Wolfe P 1969. Convergence conditions for ascent methods. *SIAM Rev.,* 11: 226-235.

Wolfe P 1971. Convergence condition for ascent methods. ii. Some corrections. *SIAM J. Optimization,* 226-235.

Yabe H & Takano M 2004. Global Convergence Properties of Nonlinear Conjugate Gradient Methods with Modified Secant Condition. *J. Comp. Optimization Applic.,* 28(2): 203-225.

Yuan G & Lu X 2009. A modified PRP conjugate gradient method. *Annual Operational Res.,*166: 73-90.

Zhang K Liu H & Liu Z 2019. A Dai-lio conjugate gradient method with optimal parameter cchoice. *Numerical Functional Analysis and Optimization,* 40(2): 194-215.