Caliphate Journal of Science & Technology (CaJoST)

ISSN: 2705-313X (PRINT); 2705-3121 (ONLINE) Research Article

Open Access Journal available at: https://cajost.com.ng/index.php/files and https://www.ajol.info/index.php/cajost/index

DOI: https://dx.doi.org/10.4314/cajost.v6i1.1

Article Info

Received: 6th March 2023

Revised: 25th December 2023

Accepted: 30th December 2023

[1]Department of Physical Sciences, Al-Hikmah University, Ilorin, Nigeria.
[2*]Department of Mathematics, University of Abuja, Abuja, Nigeria.
[3]Federal College of Dental Technology and Therapy, Enugu, Enugu, Nigeria
[4]Department of Mathematics, University of Abuja, Abuja, Nigeria.

***Corresponding author's email:**

ayinde.abdullahi@uniabuja.edu.ng

Cite this: *CaJoST*, 2024, 1, 1-9

Nonlinear Conjugate Gradient Methods for Unconstrained Test Functions through an Approximate Wolfe Line Search

Ishaq A. Adam[1], Ayinde M. Abdullahi[2*], Oyedepo Taiye[3], and Ibrahim Salihu[4]

The nonlinear conjugate gradient method stands out as a potent iterative approach for tackling unconstrained large-scale optimization problems. A crucial aspect of any conjugate gradient algorithm lies in determining an optimal step length, a task for which various strategies have been put forth. To assess and contrast the performance of the approximate Wolfe line search technique, we conducted a numerical test across nine variants of nonlinear conjugate gradient methods. Through our experiments, a notable finding emerged: the Dai-Yuan nonlinear conjugate gradient method demonstrated a swifter convergence compared to its counterparts. The utilization of the approximate Wolfe line search technique, coupled with the distinctive features of the Dai-Yuan variant, contributed to its enhanced efficiency in navigating the optimization landscape. This empirical exploration sheds light on the nuanced dynamics within nonlinear conjugate gradient methods and underscores the significance of the selected strategy for approximating the Wolfe line search. The observed faster convergence of the Dai-Yuan method not only validates its efficacy but also suggests its potential applicability in scenarios where rapid and effective optimization is paramount.

Keywords: Conjugate Gradient Method, Large Scale Problem, Optimal step-length, Unconstrained Optimization, Wolfe line search.

1. Introduction

The nonlinear conjugate gradient optimization method addresses the challenge of minimizing unconstrained problems formulated as:

$$\min f(z), z \in R^m \quad (1)$$

where R^m is an n-dimensional Euclidean space and $f{:}R^m \to R$ is a continuously differentiable function. [17] introduced the conjugate gradient method (CGM), which was later expanded to include nonlinear CGM. To name a few, [2, 4, 6, 9, 11, 14, 15, 18, 20, 32, 33] have done extensive work on nonlinear CGM. Nonlinear CGMs are a good choice for effectively tackling (1), particularly when the dimension n is huge, for engineers and mathematicians because of the uncompoundedness of their analysis, their exceptionally high memory requirement, fast convergence, and excellent numerical execution in [8, 20, 21, 29].

A nonlinear CGM creates a sequence of points $\{z^{(i)}\}$ with $i \geq 0$, by guessing an initial point $z^{(0)} \in R^m$, and utilizing the recurrence relation

$$z^{(i+1)} = z^{(i)} + \alpha_i d^{(i)} \quad (2)$$

where α_i is the step-length that can be computed using various step-length rules and $d^{(i)}$ is the search direction and can be generated by using the relation:

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

where $g^{(i)} = -\nabla f(z^{(i)})$, which is the gradient of f at $z^{(i)}$ and β_i stipulates different nonlinear CGM which corresponds to a different choice of β_i [12, 34]. The following are Some well-known β_i parameters:

$$\beta_i^{HS} = \frac{g^{(i+1)T} y^{(i)}}{g^{(i)T} g^{(i)}}, [17] \quad (4)$$

$$\beta_i^{FR} = \frac{\|g^{(i+1)}\|^2}{\|g^{(i)}\|^2}, [12] \quad (5)$$

$$\beta_i^{PR} = \frac{g^{(i+1)T} y^{(i)}}{\|g^{(k)}\|^2}, [24] \quad (6)$$

$$\beta_i^{CD} = -\frac{\|g^{(i+1)}\|^2}{g^{(i)T}d^{(i)}}, [13] \quad (7)$$

$$\beta_i^{DY} = \frac{\|g^{(i+1)}\|^2}{y^{(i)T}d^{(i)}}, [8] \quad (8)$$

$$\beta_i^{LS} = -\frac{g^{(i+1)T}y^{(i)}}{g^{(i)T}d^{(i)}}, [22] \quad (9)$$

$$\beta_i^{HZ} = \left(y^{(i)} - 2d^{(i)}\frac{\|y^{(i)}\|^2}{y^{(i)T}d^{(i)}}\right)^{(T)} \frac{g^{(i+1)T}d^{(i)}}{y^{(i)T}d^{(i)}}, [34] \quad (10)$$

$$\beta_i^{BAN} = \frac{g^{(i+1)T}y^{(i)}}{g^{(i)T}y^{(i)}}, [5] \quad (11)$$

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

where $y^{(i)} = (g^{(i+1)} - g^{(i)})$.

One eminent highlight of the nonlinear CGM is the inclusion of line search procedures in its calculation. The CG algorithm with exact or optimum line search merges only partially for an entirely quadratic function [27]. When minimizing a non-quadratic (nonlinear) function, inexact line search techniques are more appropriate and cost-effective, despite the fact that global convergence yields accuracy in some cases.

In large-scale problems, it is preferable to use β_i, which does not require the evaluation of the Hessian matrix, which frequently necessitates a large amount of computer storage. Similarities in strategies can be instituted for strongly convex quadratic functions using the optimum line search technique [1, 19]. This emphasis on importance is misplaced for a distinct class of functions. In such cases, inexact line search strategies are used [23].

2. 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_i influences both the convergence and the speed of convergence of the algorithm. The two types of line search procedures basically in use are exact and inexact line search.

2.1 Exact Line Search

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

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

i.e., α^* is the value of α_i which minimizes the objective function f along the search direction $d^{(i)}$. Therefore, α^* in (13) can be calculated by obtaining the solution of the following

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

The ideal used in (14) produces an exact or optimum value for α_i and is referred to as an exact or optimum line search. However, [28] stated that the exact or optimum line search is costly, especially when an initial point is far from the solution of the problem during the computation.

2.2 Inexact Line Search

It is critical to note that exact or optimum line search is very expensive to perform. As a result of this limitation, there is a need for a line search technique that can recognize a step-length that results in a significant decrease in the value of the objective function at a low cost. In addition, for nonlinear unconstrained optimization problems, inexact line search rules are more cost-effective and accurate to use. The framework of inexact line search rules is shown below:

• create a measure that guarantees the step-length α is neither too long nor too short;

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

• construct a sequence of updates that fulfil the model defined in (i) after every few steps.

A lot has been done on the detailing of diverse criteria by several researchers. Among these are [3, 4, 7, 10, 13, 14, 16, 25, 26, 30, 31], etc. The most widely used inexact line search techniques are Wolfe line search procedures [3]. In what takes after, we provide more consideration to Wolfe line searches.

2.3 Weak Wolfe Inexact Line Search Rule

[30] first proposed that the step-size α_i is considered optimal if it fulfills the following condition

$$f(z^{(i)} + \alpha_i d^{(i)}) \leq f(z^{(i)}) + \nu\alpha_i g^{(i)T} d^{(i)} \quad (15)$$

$$g(z^{(i)} + \alpha_i d^{(i)})^T \geq \vartheta g^{(i)T} d^i \quad (16)$$

and $\varphi(\alpha_i) = f(z^{(i)} + \alpha_i d^{(i)})$ from where $0 \leq \nu \leq \vartheta \leq 1$. The first inequality in (15) ensures that the function reduced sufficiently while the second in (16) prevents the steps from being too small.

2.4 Strong Wolfe Inexact Line Search Rule

[31] discovered that a step-size can satisfy the general Wolfe condition without necessarily minimizing the function $\varphi(\alpha_i)$. As a result, the gradient of φ is subjected to a more stringent two-sided test. This forces Alpha to end up in the vicinity of a local minimizer of φ. The following equations can be used to calculate the strong Wolfe conditions.

$$f(z^{(i)} + \alpha_i d^{(i)}) \leq f(z^{(i)}) + \nu\alpha_i g^{(i)T} d^{(i)} \quad (17)$$

$$|g(z^{(i)} + \alpha_i d^{(i)})^T| \leq \vartheta |g^{(i)T} d^{(i)}| \quad (18)$$

$$0 \leq \nu \leq \vartheta \leq 1$$

2.5 Approximate Wolfe Inexact Line Search Rule

[29] devised a novel inexact line search rule termed the approximate Wolfe line search. It approves a step size $\alpha_i > 0$ if and only if the following conditions satisfied:

$$(2\xi - 1)\varphi'(0) \geq \varphi'(\alpha_i) \geq \nu\varphi'(0) \quad (19)$$

where $\varphi(\alpha_i) = f(z^{(i)} + \alpha_i d^{(i)})$ and $0 < \gamma < \frac{1}{2} < \nu < 1$

ALGORITHM 2.1: Approximate Wolfe Inexact Line Search}

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

• If $(2\xi - 1)\varphi'(0) \geq \varphi'(\alpha_i)$ and $\varphi'(\alpha_i) \geq \nu\varphi'(0)$, take $\alpha = \gamma\alpha$

• Terminate loop with $\alpha_i = \alpha$

3. Numerical Experiments

3.1 Nonlinear CGM Algorithm

The following algorithm is a summary of the steps required to implement the CGM for a general nonlinear objective function:

ALGORITHM 3.1: CGM Algorithm

• Select the initial point, $z^{(0)} \in \mathbb{R}^m$, $\epsilon \geq 0$ (a small number called tolerance) and set $d^{(0)} = -g^{(0)} = -\nabla f(z^{(0)}), i = 0$.

• Terminate process if $\|g^{(0)}\| \leq \epsilon$, otherwise, go to Step 3.

• Compute step-size α_i by an efficient step size rule:

• Set $z^{(i+1)} = z^{(i)} + \alpha_i d^{(i)}$; if $\|g^{(i+1)}\| \leq \epsilon$, then stop, otherwise, go to the next step

• Compute the search direction $d^{(i+1)} = -g^{(i+1)} + \beta_i d^{(i)}$. Where β_i is given by equation $(4-12)$.

• Set $j = j + 1$, and go to step 3

3.2 Computational Details

The experiments will be carried out using nonlinear CGMs to solve large-scale unconstrained optimization problems. As numerical examples, thirty nonlinear unconstrained optimization test functions from [2] were used. Algorithms 3.1 and 2.1 were used to create an approximate 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 have assumed $\epsilon = 10^{-6}$ for g^* (g^* is the optimum value for the gradient of the objective function f).

3.3 Computational Examples

The following test functions and initial values obtained from [2] are used as computational examples:

1. **Diagonal 1 Function**

$$f(z) = \sum_{j=1}^{m} \left(\exp(z_j) - jz_j\right), z_0 = [1/m, 1/m, \dots, 1/m].$$

2. **Full Hessian FH2 Function**

$$f(z) = (z_1 - 5)^2 + \sum_{j=2}^{m} \left(z_1 + z_2 + \cdots + z_j - 1\right)^2,$$

$$z_0 = [0.01, 0.01, \dots, 0.01].$$

3. **TRIDIA Function(cute)**

$$f(z) = \gamma(\delta z_1 - 1)^2 + \sum_{i=2}^{m} j(\alpha z_j - \beta z_{j-1})^2,$$

$$z_0 = [1,1,\dots,1], \quad \alpha = 2, \quad \beta = 1, \quad \gamma = 1, \quad \delta = 1.$$

4. **Partial Perturbed Quadratic Function**

$$f(z) = z_1^2 + \sum_{j=1}^{m} \left(jz_j^2 + \frac{1}{100}(z_1 + z_2 + \cdots + z_j)^2\right),$$

$$z_0 = [0.5, 0.5, \dots, 0.5].$$

5. **POWER Function(cute)**

$$f(z) = \sum_{j=1}^{m} (jz_j)^2, z_0 = [1,1,\dots,1].$$

6. **EXPLIN2 Function(cute)**

$$f(z) = \sum_{j=1}^{m} \exp\left(\frac{jz_j z_{j+1}}{10m}\right)$$

$$-10 \sum_{j=1}^{m} (jz_j), z_0 = [0,0,\dots,0].$$

7. **VARDIM Function(cute)**

$$f(z) = \sum_{j=1}^{m} (z_j - 1)^2$$

$$+ \left(\sum_{j=1}^{m} jz_j - \frac{m(m+1)}{2}\right)^2 + \left(\sum_{j=1}^{m} jz_j - \frac{m(m+1)}{2}\right)^4,$$

$$z_0 = \left[1 - \frac{1}{m}, 1 - \frac{2}{m}, \dots, 1 - \frac{m}{m}\right].$$

8. Variably Dimensioned Function

$$f(z) = \sum_{j=1}^{m} (z_j - 1)^2 + \left(\sum_{j=1}^{m} j(z_j - 1)\right)^2 + \left(\sum_{j=1}^{m} j(z_j - 1)\right)^4,$$

$$z_0 = \left[1 - \frac{1}{m}, 1 - \frac{2}{m}, \dots, 1 - \frac{m}{m}\right].$$

INDEF Function

$$f(z) = \sum_{j=1}^{m} (z_j) + \sum_{j=2}^{m-1} \left(\alpha\cos(2z_j - z_m - z_1)\right),$$

$$z_0 = \left[\frac{1}{m+1}, \frac{2}{m+1}, \dots, \frac{m}{m+1}\right], \quad \alpha = 0.5.$$

9. Liarwhd Function

$$f(z) = \sum_{j=1}^{m} \left(4(z_j^2 - z_1)^2\right) + \sum_{j=1}^{m} (z_j - 1)^2, z_0 = [4,4, \dots, 4].$$

10. McCormick Function

$$f(z) = \sum_{j=1}^{m-1} \left(-1.5z_j + 2.5z_{j+1} + 1 + (z_j - z_{j+1})^2 + \sin(z_j + z_{j+1})\right),$$

$$z_0 = [1,1, \dots, 1].$$

11. NONDIA Function

$$f(z) = (z_1 - 1)^2 + \sum_{j=2}^{m} 100(z_1 - z_{j-1}^2)^2,$$

$$z_0 = [-1, -1, \dots, -1].$$

12. NONDQUAR Function

$$f(z) = \sum_{j=1}^{m-2} (z_j + z_{j+1} + z_m)^4 + (z_1 - z_2)^2 + (z_{m-1} + z_m)^2, z_0 = [-1, -1, \dots, -1].$$

13. NONSCOMP Function

$$f(z) = (z_1 - 1)^2 + \sum_{j=2}^{m} 4(z_j - z_{j-1}^2)^2,$$

$$z_0 = [3,3, \dots, 3].$$

14. maratosb Extended Function

$$f(z) = z_1 + \sum_{j=1}^{m-1} \frac{(z_j^2 + z_{j+1}^2 - 1)^2}{0.000001}, z_0 = [0,0, \dots, 0].$$

15. MDHOLE Extended Function

$$f(z) = z_1 + \sum_{j=1}^{m} \frac{(\sin z_j - 10)^2}{0.01}, z_0 = [10,10, \dots, 10].$$

16. Logros Extended Function

$$f(z) = \sum_{j=1}^{m} \log\left(1 + 10000(1 - z_j^2)^2 + (1 - z_j)^2\right), z_0 = [-1.2, -1.2, \dots, -1.2].$$

17. mexhat Extended Function

$$f(z) = -2(z_1 - 1)^2 + 10000,$$

$$\sum_{j=1}^{m-1} \left((z_j - 1)^2 + \frac{(z_{j+1} - z_j^2)^2}{10000} - 0.02\right)^2$$

$$z_0 = [1 - 0.4 \times 1, 1 - 0.4 \times 2, \dots, 1 - 0.4 \times m].$$

18. nasty Extended Function

$$f(z) = 0.5 \sum_{j=1}^{m-1} \left((1.0e^{10} z_j)^2 + z_{j+1}^2\right),$$

$$z_0 = [e^{-30}, 1, \dots, (2 - m)e^{-30} - 1 + m].$$

19. Biggs B1 Function

$$f(z) = (z_1 - 1)^2 + \sum_{j=1}^{m-1} (z_{j+1} - z_j)^2 + (1 - z_m)^2$$

$$z_0 = [0,0, \dots, 0].$$

Extended Beale Function

$$f(z) = \sum_{j=1}^{m/2} \left(1.5 - z_{2j-1}(1 - z_{2j})\right)^2$$

$$+(2.25 - z_{2j-1}(1 - z_{2j}^2))^2 + (2.625 - z_{2j-1}(1 - z_{2j}^3))^2,$$

$$z_0 = [1,0.8, \dots, 1,0.8].$$

20. Extended Cliff Function

$$f(z) = \sum_{j=1}^{m/2} ((z_{2j-1} - 3)0.01)^2 - (z_{2j-1} z_{2j}) + \exp(20(z_{2j-1} - z_{2j})),$$

$$z_0 = [0, -1, \dots, 0, -1].$$

21. Extended DenschnA Function

$$f(z) = \sum_{j=1}^{m/2} (z_{2j-1}^4) + (z_{2j-1} + z_{2j})^2 + (-1 + \exp(z_{2j}))^2,$$

$$z_0 = [1,1, \dots, 1].$$

22. Extended DenschnB Function

$$f(z) = \sum_{j=1}^{m/2} (z_{2j-1} - 2)^2 + (z_{2j-1} - 2)^2 z_{2j}^2 + (z_{2j} + 1)^2, z_0 = [1,1, \dots, 1].$$

23. Extended DenschnF Function

$$f(z) = \sum_{j=1}^{m/2} (2(z_{2j-1} + z_{2j})^2 + (z_{2j-1} - z_{2j})^2 - 8)^2 + (5z_{2j-1}^2 + (z_{2j} - 3)^2 - 9)^2,$$

$$z_0 = [2,0, \dots, 2,0].$$

24. Diagonal 7 Function

$$f(z) = \sum_{j=1}^{m} \exp(z_j) - 2z_j - z_j^2,$$

$$z_0 = [1,1, \dots, 1].$$

25. Diagonal 8 Function

$$f(z) = \sum_{j=1}^{m} z_j \exp(z_j) - 2z_j - z_j^2,$$

$$z_0 = [1,1, \dots, 1].$$

26. Brownbs Function

$$f(z) = \sum_{j=1}^{m-1} (z_j - 1000000)^2$$

$$+ \sum_{j=1}^{m-1} (z_{j+1} - 0.000002)^2 + \sum_{j=1}^{m-1} (z_j z_{j+1} - 2)^2, z_0 = [1,1, \dots, 1].$$

27. DQDRTIC Function

$$f(z) = \sum_{j=1}^{m} (z_j^2 + 100 z_{j+1}^2 + 100 z_{j+2}^2), z_0 = [3,3, \dots, 3].$$

28. HimmelBG Function

$$f(z) = \sum_{j=1}^{m/2} (2z_{2j-1}^2 + 3z_{2j}^2) \exp(-z_{2j-1} - z_{2j}),$$

$$z_0 = [1.5,1.5, \dots, 1.5].$$

3.4 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 denoted by the following: Dim (dimension), ITR (number of iterations), CPU (time taken), AWLS (strong Wolfe line search), AVE (average).

Table 1. CPU Results for SWR (Strong Wolfe Line Search)

S/N	Test Function	DIM	BAN	FR	PR	HS	CD	DY	LS	HZ	GSC
1.	Diagonal 1	5000	.21	86.07	.15	.55	.18	0.58	.16	.09	.07
		10000	0.12	0.26	0.24	0.14	0.52	0.13	0.10	0.16	0.08
2.	Full Hessian FH2	5000	737.71	804.65	875.6	776.30	849.07	895.08	897.06	838.07	803.79
		10000	986.68	957.57	946.8	946.69	986.87	957.46	968.36	945.47	954.48
3.	TRIDIA (cute)	5000	76.41	109.60	44.67	20.53	11.87	0.17	48.65	45.68	8.24
		10000	187.98	53.55	103.4	22.87	11.05	0.34	103.23	106.35	41.40
4.	Partial Perturbed Quadratic	5000	737.7	804.65	875.6	776.31	849.07	895.08	897.06	838.07	803.79
		10000	986.7	957.57	946.8	946.69	986.87	957.46	968.36	945.47	954.48
5	POWER (cute)	5000	0.02	0.02	0.02	0.02	0.04	0.02	0.03	0.02	0.01
		10000	0.02	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.06
6	EXPLIN2 (cute)	5000	0.27	10.91	315.8	0.11	0.26	0.40	312.69	0.33	0.19
		10000	0.49	3.12	76.84	0.21	0.61	1.37	603.61	0.64	0.36
7	VARDIM (cute)	5000	0.08	0.08	0.07	0.07	0.08	0.08	0.09	0.11	0.08
		10000	0.14	0.14	0.15	0.13	0.14	0.14	0.13	0.14	0.14
8	Variably Dimensioned	5000	0.08	0.07	0.08	0.08	0.07	0.08	0.07	0.08	0.09
		10000	0.15	0.14	0.14	0.13	0.14	0.14	0.15	0.13	0.16
9	INDEF	5000	3.04	16.00	24.56	0.22	8.65	8.36	23.48	43.06	48.09
		10000	56.35	34.76	41.60	0.23	47.56	7.38	37.96	60.96	81.84
10	Liarwhd	5000	1.08	117.74	15.28	0.45	116.35	0.33	94.44	67.45	60.29
		10000	3.95	140.76	8.77	1.20	53.79	1.67	144.49	152.68	250.95
11	McCormick	5000	195.17	206.99	112.3	23.58	27.05	698.66	106.88	43.14	225.76
		10000	206.14	516.10	184.8	46.70	42.31	238.57	199.87	59.41	71.03
12	NONDIA	5000	0.45	81.38	71.37	1.18	0.25	2.43	54.74	1.99	17.71
		10000	0.74	110.46	88.67	0.86	0.47	0.48	87.37	3.62	107.99
13	NONDQUAR	5000	0.10	0.11	0.10	0.10	0.12	0.10	0.10	0.22	0.14
		10000	0.20	0.20	0.204	0.20	0.20	0.20	0.20	0.20	0.20
14	NONSCOMP	5000	0.11	77.04	93.78	0.12	98.58	0.11	87.38	0.78	126.41
		10000	3.45	193.07	158.1	0.33	173.30	0.34	161.63	1.23	205.07
15	Maratosb Extended	5000	0.02	66.37	45.99	0.03	0.03	0.03	47.42	0.02	0.03
		10000	0.02	139.99	79.7	0.02	0.02	0.02	81.70	0.02	0.03
16	MDHOLE Extended	5000	3.31	83.27	00.0	0.23	97.92	11.03	57.29	78.67	98.34
		10000	39.28	154.56	10.63	3.41	72.67	64.38	119.13	3.71	385.78
17	Logros Extended	5000	0.04	36.31	16.50	0.06	17.04	0.11	16.55	43.42	48.55
		10000	1.13	69.92	22.46	0.09	24.23	0.18	22.40	22.79	73.50
18	mexhat Extended	5000	0.11	0.10	0.097	0.10	0.10	0.09	0.10	0.10	0.09
		10000	0.19	0.20	0.261	0.20	0.18	0.18	0.18	0.20	0.20
19	nasty Extended	5000	0.06	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.05
		10000	0.04	0.04	0.04	0.04	0.03	0.04	0.04	0.04	0.03
20	Biggs B1 Function	5000	0.61	2.94	101.9	0.65	0.64	0.64	97.25	0.61	0.67
		10000	1.15	5.47	178.0	1.14	1.18	1.18	170.68	1.17	1.28

21	Extended Beale	5000	1.36	2.47	15.75	0.95	69.53	23.96	21.17	10.17	417.92
		10000	2.15	4.27	26.72	2.59	90.89	28.06	24.52	18.07	778.92
22	Extended Cliff	5000	2.94	2.08	5.07	0.09	0.37	0.58	0.88	1.77	110.57
		10000	17.53	3.62	3.71	0.09	0.88	0.73	1.23	2.66	184.29
23	Extended DenschnA	5000	0.13	110.26	288.0	0.16	3.18	1.26	266.67	0.50	0.78
		10000	0.20	187.00	405.9	0.22	4.09	1.42	418.04	0.53	7.38
24	Extended DenschnB	5000	0.39	85.78	77.57	0.18	1.79	116.61	78.29	0.46	0.30
		10000	0.66	147.56	141.2	0.27	10.26	207.31	136.94	0.59	0.38
25	Extended DenschnF	5000	0.16	191.63	193.3	0.15	0.60	0.19	163.08	0.29	1.65
		10000	0.22	344.36	305.6	0.26	4.93	0.30	297.39	0.67	3.22
26	Diagonal 7	5000	0.08	0.51	1.48	0.06	16.59	0.60	1.27	24.25	31.94
		10000	0.08	1.03	2.27	18.58	22.56	0.80	2.04	24.92	37.29
27	Diagonal 8	5000	0.27	0.43	2.35	0.14	1.68	1.39	4.71	69.32	18.16
		10000	0.25	1.76	3.74	50.24	1.78	0.39	5.15	41.34	3.34
28	Brownbs	5000	0.09	1.52	0.62	0.10	0.48	248.52	0.45	0.42	0.18
		10000	0.15	2.50	0.848	0.20	0.43	0.56	0.82	0.67	0.30
29	DQDRTIC	5000	2.15	1.22	6.89	2.79	59.88	0.70	2.44	1.68	0.91
		10000	3.41	19.32	5.04	3.33	59.97	1.70	3.09	2.50	7.70
30	HimmelBG	5000	16.59	17.17	16.78	16.90	0.26	0.38	17.39	0.03	0.27
		10000	23.27	23.66	22.99	23.17	0.47	0.57	22.72	0.03	0.45

Table 2. ITR Results for SWR (Strong Wolfe Line Search)

S/N	Test Function	DIM	BAN	FR	PR	HS	CD	DY	LS	HZ	GSC
1.	Diagonal 1 Function	5000	9	2000	4	33	6	155	4	5	4
		10000	3	3	3	3	8	3	3	6	4
2.	FullHessian FH2	5000	5	6	6	4	6	6	6	6	6
		10000	8	8	8	8	8	8	8	8	8
3.	TRIDIA (cute)	5000	2000	2000	2000	662	243	7	2000	2000	104
		10000	2000	805	2000	325	110	8	2000	2000	265
4.	Partial Perturbed Quadratic Function	5000	5	6	6	4	6	6	6	6	6
		10000	8	8	8	8	8	8	8	8	8
5	POWER (cute)	5000	1	1	1	1	1	1	1	1	1
		10000	1	1	1	1	1	1	1	1	1
6	EXPLIN2 (cute)	5000	5	73	2000	4	4	6	2000	5	3
		10000	5	15	2000	4	4	7	2000	5	3
7	VARDIM (cute)	5000	1	1	1	1	1	1	1	1	1
		10000	1	1	1	1	1	1	1	1	1
8	Variably Dimensioned	5000	1	1	1	1	1	1	1	1	1
		10000	1	1	1	1	1	1	1	1	1
9	INDEF	5000	279	2000	2000	14	370	212	2000	2000	2000
		10000	2000	2000	2000	4	1302	91	2000	2000	2000
10	Liarwhd	5000	65	2000	364	12	2000	13	2000	2000	2000
		10000	136	2000	165	18	2000	28	2000	2000	2000
11	McCormick	5000	2000	2000	2000	2000	2000	2000	2000	2000	770
		10000	2000	2000	2000	2000	2000	2000	2000	2000	122
12	NONDIA	5000	36	2000	2000	63	16	68	2000	129	414
		10000	40	2000	2000	28	16	17	2000	143	1375

13	NONDQUAR	5000	1	1	1	1	1	1	1	1	1
		10000	1	1	1	1	1	1	1	1	1
14	NONSCOMP	5000	7	2000	2000	8	2000	6	2000	60	2000
		10000	113	2000	2000	12	2000	6	2000	59	2000
15	maratosb	5000	2	2000	2000	2	2	2	2000	2	3
	Extended	10000	2	2000	2000	2	2	2	2000	2	3
16	MDHOLE	5000	139	2000	2000	16	2000	463	2000	2000	2000
	Extended	10000	645	2000	225	104	2000	2000	2000	99	2000
17	Logros	5000	3	2000	2000	3	2000	4	2000	2000	2000
	Extended	10000	87	2000	2000	8	2000	4	2000	2000	2000
18	mexhat	5000	1	1	1	1	1	1	1	1	1
	Extended	10000	1	1	1	1	1	1	1	1	1
19	nasty Extended	5000	1	1	1	1	1	1	1	1	1
		10000	1	1	1	1	1	1	1	1	1
20	BiggsB1	5000	3	40	2000	3	3	4	2000	3	4
		10000	3	40	2000	3	3	4	2000	3	4
21	Extended Beale	5000	56	83	193	11	2000	242	148	106	2000
		10000	58	102	204	23	2000	242	132	164	2000
22	Extended Cliff	5000	120	46	171	4	7	21	36	63	2000
		10000	480	46	90	4	9	21	36	83	2000
23	Extended DenschnA Function	5000	3	2000	2000	3	24	27	2000	11	12
		10000	3	2000	2000	3	22	17	2000	9	35
24	Extended DenschnB	5000	13	2000	2000	8	31	2000	2000	12	5
		10000	13	2000	2000	8	90	2000	2000	9	5
25	Extended DenschnF	5000	4	2000	2000	6	18	6	2000	10	25
		10000	4	2000	2000	6	54	6	2000	11	28
26	Diagonal 7	5000	9	67	91	7	2000	88	87	2000	2000
		10000	7	92	87	2000	2000	90	81	2000	2000
27	Diagonal 8	5000	29	39	82	8	12	16	97	2000	2000
		10000	21	84	82	2000	12	18	85	2000	208
28	Brownbs	5000	3	24	11	3	17	2000	11	14	6
		10000	3	20	11	3	7	11	11	16	6
29	DQDRTIC	5000	153	88	227	158	2000	75	132	165	29
		10000	166	723	146	146	2000	124	103	172	89
30	HimmelBG	5000	2000	2000	2000	2000	4	44	2000	2	9
		10000	2000	2000	2000	2000	4	54	2000	2	9

Table 3. Inference on CGMs with AWLS by considering the average of total number of CPU and ITR

	BAN	FR	RP	HS	CD	DY	LS	HZ	GSC
SWR-CPU-AVE	71.73	118.17	117.78	61.54	80.50	89.85	131.32	75.12	116.29
SWR-ITR-AVE	279.42	907.17	936.60	229.47	574.00	237.53	1016.77	523.50	626.40

3.5 Remarks on Computational Results

To provide a clearer description and enhance understanding of the aforementioned tables, the performance assessments of all the nonlinear Conjugate Gradient Methods (CGMs) under the approximate Wolfe's inexact line search are presented in Table 3.

According to Table 3, the nonlinear CGMs of HS, BAN, and HZ perform better than other CG methods in terms of CPU, while DY, BAN, and

HS perform better in terms of ITR. As a result, we can conclude that DY and BAN outperform all other CG methods in terms of CPU and ITR average.

4. Conclusion

In this paper, nine (9) CGM variants were considered, namely BAN, FR, PR, HS, CD, DY, LS, HZ, and GSC methods, to solve thirty unconstrained optimization test problems using approximate Wolfe line search. DY nonlinear CGM performed better than the other nonlinear CGMs considered.

Conflict of interest

The authors declare no conflict of interest.

Acknowledgements

The authors appreciate the reviewers and editor for their careful reading, and valuable suggestions.

References

[1] O. J. Adeleke, O. A. Aderemi, N. I. Omoregbe and R. A. Adekunle`` Numerical Comparison of Line Search Criteria in Nonlinear Conjugate Gradient Algorithms,'', *Journal of Mathematics and Statistics Studies*, vol. **2**, no. 1, p. 14-15, 2013. www.ea-journals.org

[2] N. Andrei ``An Unconstrained Optimization Test Functions Collection,'' *Journal of Advanced Modeling and Optimization*, vol. **10**, no. 1, p. 149-159, 2008. https://doi.org/10.1.1.665.3152

[3] N. Andrei `` Open Problems in Nonlinear Conjugate Gradient Algorithms for Unconstrained Optimization,'' *Bull*, vol. **30**, no. 2, p. 319-330, 2011. http://math.usm.my/bulletin

[4] L. Armijo `` Minimization of Functions having Lipschitz Continuous First Partial derivatives,''*Pacific Journal of Mathematics*, vol. **16**, p. 1-3, 1966. https://msp.org/pjm/1966/16-1/pjm-v16-n1-p01-

[5] O. M. Bamigbola, M. Ali and E. Nwaeze, `` An efficient and convergent method for unconstrained nonlinear optimization,'' *Proceedings of International Congress of Mathematicians*, Hyderabad, India, 2010.

[6] E. M. L. Beale, `` *A derivation of conjugate gradients in numerical methods for nonlinear optimization*,'' Academic Press, London. p. 39-43, 1971.

[7] P. T. Boggs and R. B. Schnabel, ``*Numerical Optimization*,'' Lecture Note for a short course, 1987

[8] Y. Dai and Y. Yuan, `` A nonlinear conjugate gradient with a strong global convergence properties,'' *SIAM Journal on Numerical Analysis*, vol. **10**, , p. 177-182, 2000. DOI: 10.1137/S1052623497318992

[9] J. W. Daniel, `` The conjugate gradient method for linear and nonlinear operator equations,'' *SIAM Journal on Optimization*, vol. **4**, ,p. 10-26, 1967. https://doi.org/10.1137/0704002

[10] J. E. Dennis and R. B. Schnabel, `` Numerical Methods for Unconstrained Optimization and Nonlinear Equations*,'' Prentice Hall Series in Computational Mathematics*, Englewood Cliffs, NJ, 1983.

[11] L. C. W. Dixon, P. G. Ducksbury and P. Singh, ``A New Three-term Conjugate Gradient Method'', *Journal on Optimization Theory Application,* vol. **47**, no. 32, p. 285-300, 1985.

[12] R. Fletcher and C. M. Reeves, `` Function Minimization by Conjugate Gradients, '' *Computer Journal,* vol.7, no. 2, 1964. doi.org/10.1093/comjnl/7.2.149

[13] R. Fletcher, `` *Practical Methods of Optimization*,'' John Wiley and Sons, New York, 1987. DOI:10.1002/9781118723203

[14] A. A. Goldstein, `` On steepest descent,'' *Journal of Industrial Application*, vol. **3**, , p. 147-151, 165. DOI:10.1137/0303013

[15] W. W. Hager and H. Zhang `` A new conjugate gradient method with guaranteed descent and an efficient line search,'' *SIAM Journal on Optimization*, vol. **16**,no. 1, p. 170-192, 2005. doi.org/10.1137/030601880

[16] W. W. Hager and H. Zhang ``A survey of nonlinear conjugate gradient methods,'' *Pacific Journal of Optimization*, vol. **2**,no. 1, p. 35-58, 2006. https://www.cmor-faculty.rice.edu/~yzhang/caam554/pdf/cgsurvey.pdf

[17] M. R. Hestenes and E. Stiefel,``Method of conjugate gradient for solving linear equations,'' *Journal on Res. Nat. Bur. Stand*, vol. **49**, 1952. https://nvlpubs.nist.gov/nistpubs/jres/049/jresv49n6p409_a1b.pdf

[18] A. A. Ishaq, T. Latunde and K. B. Akande,``A Step-Length Formulafor Conjugate Gradient Methods,''*Malysian Journal of Computing*, vol. **5**, no.1,p.403-413,2020. DOI:10.24191/mjoc.v5i1.7715

[19] A. A. Ishaq, T. Latunde and F. M. Jimoh,``An Optimum Line Search for Unconstrained non-polynomial Test Functions Using Nonlinear Conjugate Gradient Methods,''*Malysian Journal of Computing*, vol. **10**, no. 40 , p. 20-25, 2020. http://ijm2c.iauctb.ac.ir

[20] H. Jinhong and Z. Genjiao,`` A Conjugate Gradient Method without Line Search and the Convergence Analysis,'' *Fourth International Conference on Emerging Intelligent Data and Web Technologies*, p. 37-86, 2013. DOI:10.1109/EIDWT.2013.133

[21] L. Wang, M. Cao, F. Xing and Y. Yang, ``The new spectral conjugate gradient method for large-scale unconstrained optimization,'' *Journal of Inequalities and Applications*, vol. **2020**, p. 111, 2020. 10.1186/s13660-020-02375-z

[22] Y. Liu and C. Storey,``A Step-Length Formulafor Conjugate Gradient Methods,'' *Journal of Optimization Theory and Application*, vol. **69**, p. 129-137, 1992.
[23] M. Liu, G. Ma and J. Yin,`` Two New Conjugate Gradient Methods for Unconstrained Optimization,'' *Hindawi Complexity*, vol. **2020**, p. 1-13, 2020.
doi.org/10.1155/2020/9720653
[24] E. Polak and G. Ribiere,`` Note sur la Convergence de Directions Conjugees,'' *Rev. Francaise Informat Recherche Operationelle*, vol. **3**,no.16,p.35-43,1969.
http://eudml.org/doc/193115
[25] F. A. Potra and Y. Shi, ``Efficient Line Search Algorithm for Unconstrained Optimization,'' *Journal on Optimization Theory Application* vol. **85**, no. 3, p. 677-704, 1995. https://link.springer.com/article/10.1007/BF02193062
[26] M. J. D. Powell, ``*Some Global Convergence Properties of a Variable Metric Algorithm for Minimization without Exact Line Searches in Nonlinear Programming*, '' p. 53-72, 1975.
[27] R.T. Rockafellar, ``*Convex analysis*, ''Princeton University Press, Princeton, 1970. doi.org/10.1515/9781400873173
[28] W. Sun and Y. Yuan, ``*Optimization Theory and methods*, '' Springer-Verlag, New York, 2006.
https://link.springer.com/book/10.1007/b106451
[29] Z. Sun, H. Li, J. Wang and Y. Tian,`` Two Modified Spectral Conjugate Gradient Methods and their Global Convergence for Unconstrained Optimization,''*Malysian Journal of Computing*, vol. 9**5**, no. 10 , p. 2082-2099, 2018. DOI: 10.1080/00207160.2017.1366457
[30] P. Wolfe, `` *Convergence Conditions for Ascent Methods*, '' *SIAM Rev*, vol. 11, 223-265, 1969. http://www.siam.org/journals/ojsa.php
[31] W. Sun and Y. Yuan, `` Convergence Condition for Ascent Methods. ii. Some corrections, '' *SIAM Journal on Optimization*, vol. 13, no. 2, 1971. https://doi.org/10.1137/1013035
[32] H. Yabe and M. Takano`` Global Convergence Properties of Nonlinear Conjugate Gradient Methods with Modified Secant Condition,'' *Journal on Computational Optimization Application*, vol. **28**, no. 2 , p. 203-225, 2004. https://doi.org/10.1023/B:COAP.0000026885.81997.88
[33] G. Yuan and X. Lu,`` A Modified PRP Conjugate Gradient Method,'' *Annual Operational Research*, vol. **166**, p. 73-90, 2009. DOI: 10.1007/s10479-008-0420-4
[34] K. Zhang, H. Liu and Z. Liu `` A Dai-lio Conjugate Gradient Method with Optimal Parameter Cchoice,'' *Numerical Functional Analysis and Optimization*, vol. **40**, no. 2, p. 194-215, 2019.

