**New Multi-Step Three-Term Conjugate Gradient Algorithms With Inexact Line Searches**

|  |  |
| --- | --- |
| **Abbas Y. Al-Bayati****Prof./ Department of Mathematics****College of Basic Education** **Telafer University, Iraq"****Email:** **profabbasalbayati@yahoo.com** | **Muna M. M. Ali****Lecture/ Department of Mathematics****College of Computers Sciences and Mathematics / Mosul University. Iraq** |

**Abstract**

This work suggests several multi-step three-term Conjugate Gradient (CG)-algorithms that satisfies their sufficient descent property and conjugacy conditions. First, we have considered a number of well-known three-term CG-method, and we have, therefore, suggested two new classes of this type of algorithms which was based on Hestenes and Stiefel (HS) and Polak-Ribière (PR) formulas with **four different versions**.

Both descent and conjugacy conditions for all the proposed algorithms are satisfied, at each iteration by using the strong Wolfe line search condition and it's accelerated version. These new suggested algorithms are some sort of modifications to the original HS and PR methods. These CG-algorithms are considered as a sort of the memoryless BFGS update.

 All of our new suggested methods are proved to be a global convergent and numerically, more efficient than the similar methods in same area based on our selected set of used numerical problems.

**Keywords:** Three-term Conjugate gradient, Scaled conjugate gradient, Multi-step conjugate gradient, Sufficient descent property, Inexact line searches, Global convergence property.

**1.Introduction.**

This paper considers the calculation of a local minimizer x\* say, for the problem:

 ; where  (1)

is a nonlinear function and its gradient vector  is available are calculated but the Hessian matrix is not available. At the current iterative point , the CG-method has the following form:

 (2a)

 (2b)

where  is a step-length;  is a search direction;  is a parameter. Standard algorithms for solving this problem include CG-algorithms which are iterative algorithms and generate a sequence of approximations to minimize a function  and their very low memory requirements.

**1.1 Three-Term CG-Methods.**

The first three-term CG-method was proposed by (Beale, 1972) as :

  (3a)

Where

  (3b)

  (3c)

and *dt* is a restart direction.

Nazareth (Nazareth, 1977) proposed another three-term recurrence formula:

  (4a)

with . (4b)

Two new three-term CG-algorithms was considered by Zhang, (Zhang et al., 2006, 2007), that is,

  (5a)

  (5b)

  (6a)

  (6b)

Also, Zhang, (Zhang et al., 2009) introduced a three-term recurrence formula based on the "(Dai-Liao, 2001)" CG-method as follows:

  (7a)

where *d0=-g0* and . The sufficient descent condition also holds independent of the used line search procedure, i.e. for this method:

 for all k. (7b)

A specialization of (7) was developed by (Al-Bayati and Sharif, 2010) where the search direction is computed as:

  (8a)

  ; . (8b)

Furthermore, it is easy to see that (8) satisfies the sufficient descent condition independent of the line search used.

Moreover, another three-term CG-method was improved by (Al-Bayati and Al-Hassan, 2011), and their search direction, with inexact line search (ILS), is as follows :

  (9a)

  (9b)

Recently, a three-term CG-method was introduced by (Al-Bayati and Al-Khayat, 2014), and their search direction is as follows :

  (10a)

  (10b)

**1.2 Memoryless QN-Methods.**

Conjugate Gradient algorithms could be regarded as a sort of the memoryless QN-update, especially, for the BFGS update. This type of method was suggested for the first time by (Perry, 1978); he noted that the scalar  was chosen to make  and  conjugate using an exact line search (ELS). Perry relaxed this requirement where  is defined by HS formula in an equivalent form, but assuming inexact line search (ILS), thus he obtained, i.e.:

  (11a)

but this matrix is not of full rank; Perry modified it further as:

 (11b)

or

 (11c)

 Then (Shanno, 1978) addressed that (11) does not satisfy the actual QN-condition, so he modified it in order to make it do so, he then obtained:

  (12)

 This new form of the projection matrix  has a special relationship with the BFGS update formula which defined by (Dennis and More, 1983).

 (13)

It is easily seen that (12) is equivalent to (13) with  replaced by ( I ) i.e. If , where ( I ) is the identity matrix.

 The CG-method, which is referred to as **Memoryless BFGS** method is defined by:

 (14a)

then

 (14b)

The above equality can be rewritten in the following equivalent form:

  (14c)

 Finally, (Al-Bayati, et al., 2019) introduced a new CG-algorithm with different parameters, namely; for

 ;  (15a)

 (15b)

**2. New Multi-Step Three-Term CG-Algorithms.**

 Nazareth (Nazareth and Nocedal, 1978) developed a multi-step CG-method which does not need ELS; by defining the following matrices:

 ****; ****; ****

 where B is an (nxn) upper triangular matrix with,

****

 Assuming the Hussain matrix G=I (Identity matrix), to get a set of mutually orthogonal vectors ****. Based on this idea let us construct a new set of mutually vectors **** where they are a linear combination of the normal gradient terms **** and defined as follows":

 (16a)

iterate for k = 2,3,4, ...... with

 (16b)

 (16c)

 (16d)

Clearly, (16) for the first iterate is equivalent for the normal set of gradients, while it gives better approximations for the next iterates. However, in the first part of this section, we describe our new CG-algorithms N1; N2 and N3. Descent and conjugacy conditions for these new CG-algorithms have been fully described.

 Our N4 scaled multi-step three-term CG-algorithms can be described in the next section. Namely; we have the following new CG-algorithms:

 . (17a)

  (17b)

 . (17c)

**2.1 An Acceleration Scheme for Wolfe Line Search.**

Wolfe line search procedure is fully described by many researchers, see for example (Nocedal, 1996) and (Liu and Nocedal). This line search scheme has been modified by Andrei (2009). First, let us consider:

The standard Wolfe line search conditions can be defined as:

  (18a)

  (18b)

The strong Wolfe line search conditions can be defined as:

 (18c)

 (18d)

The accelerating Scheme for Wolfe line search technique is as follows:

;  (18e)

. (18f)

Hence, if . New estimation of the solution is computed as , else . For this reason, using the definitions of  and the above acceleration scheme can present the accelerated Wolfe line search procedure.

**NOTE: (For The Rest of This Paper and For Simplicity, We Denote For Every  By  and For Every  By ).**

**2.2 The First Multi-Step Three-Term CG-Method (N1).**

To compute the new search direction  , let us consider the QN-BFGS update with H=I.

  19)

 (20)

 Or, equivalently

  (21)

  (22a)

 (22b)

**2.3 Outlines of the Multi-Step Three-Term CG-Algorithm (N1).**

**St1.** Given  , let ,  and . Set .

**St2.** If stopping criteria () satisfied, then stop.

**St3.** Compute  by Wolfe line search and it's acceleration as in (18).

**St4.** The parameters  and  are computed from (21).

**St5.** The new search direction  is computed from (22).

**St6.** If  is satisfied then set 

**St7.** Set , go to St2.

**2.4 The Second Multi-Step Three-Term CG-Method (N2).**

To compute the new search direction , let us consider (Al-Bayati, 1991) QN-update with

 (25)

Since  (26)

 (27)  (28)

 (29a)

 (29b)

**2.5 Outlines of the Multi-Step Three-Term CG-Algorithm (N2).**

**St1.** Given  , let ,  and . Set .

**St2.** If stopping criteria () satisfied, then stop.

**St3.** Compute  by Wolfe line search and it's acceleration as in (18).

**St4.** The parameters  and  are computed from (28).

**St5.** The new search direction  is computed from (29).

**St6.** If  is satisfied then set 

**St7.** Set , go to St2.

**2.6 The Third Multi-Step Three-Term CG-Method (N3).**

To compute the new search direction , let us consider (Oren, 1974) QN-update with H=I.

 (30a)

 (30b)

 (31)

This implies

 (32)

 (33)

 (34a)

 (34b)

**2.7 Outlines of the Multi-Step Three-Term CG-Algorithm (N3).**

**St1.** Given  , let ,  and . Set .

**St2.** If stopping criteria () satisfied, then stop.

**St3.** Compute  by Wolfe line search and it's acceleration as in (18).

**St4.** The parameters ; and  are computed from (33).

**St5.** The new search direction  is computed from (34).

**St6.** If  is satisfied then set 

**St7.** Set , go to St2.

**2.8 The Fourth Scaled Multi-Step Three-Term CG-Method (N4).**

Here, we describe our new CG-method (N4) for which, it is independent of the line search, at every step. Both the descent and the conjugacy condition are satisfied. The direction  is computed as:

  (35)

 , (36)

  (37)

  (38)

 , (39)

  (40)

**2.9 Outlines of the Multi-Step Three-Term CG-Algorithm (N4).**

**St1.** Given  , let ,  and . Set .

**St2.** If stopping criteria () satisfied, then stop.

**St3.** Compute  by Wolfe line search and it's acceleration as in (18).

**St4.** The parameters; ;  and  are computed from (36), (37) and (38) respectively.

**St5.**  and the new search direction are computed from (39) and (40) respectively.

**St6.** If  is satisfied then set 

**St7.** Set , go to St2.

**3. Conjugacy and Sufficient Descent Conditions.**

 Sufficient descent and conjugacy conditions properties of HS and PR methods can be found directly in (Hestenes and Stiefel, 1952) and (Polak and Ribière, 1969). To show that our new proposed multi-step three-term CG-algorithms have global convergence property by using Wolfe conditions (18). Let us use the following propositions:

**3.1 Proposition.**

 Suppose that Wolfe conditions defined in (18) are satisfied, then N1; N2 and N3 defined by (22), (29) and (34) are descent directions, i.e.

. (41)

**Proof.**

**I:** For  and from Wolfe conditions (18), we have , then:

  (42)

**II:** For  and from Wolfe condition (18), we have , then:

  (43)

 **III:** For  and from Wolfe condition (18), we have , then:

  (44)

Therefore, we conclude that our new algorithms N1; N2 and N3 are descent.

**3.2 Proposition.**

 If (18) is satisfies, then  ;  and  defined by (22), (29) and (34) are satisfy the conjugacy condition (Dai-Liao, 2001), namely:

  (45)

  for all k. (46)

**Proof.**

 **I:** If  is used then we get:

 , (47a)

 , since . (47b)

 **II:** If , is used then we get:

 , (48a)

 , since . (48b)

 **III:** If , is used then we get:

 , (49a)

 ;. (49b)

It is clear that, if *f* is satisfies (18), then  , and .

Therefore, CG-algorithms (22), (29) and (34) reduces to standard HSCG-method.

**3.3 Proposition.**

Suppose that (18) holds, then  defined in (40), is sufficient descent direction.

**Proof.**

When  is used and (18) is satisfied, then .

Therefore, we have from (36):

  (50)

Note that, since , then , for all .

Hence,  defined in (40) satisfies:

 , (51)

where  , is modified at every iteration. Therefore,  is sufficient descent direction.

**4. Convergence Analysis.**

Let us list the following mild assumptions, before going to proof the global convergence property of the new algorithms N1; N2; N3 and N4.

**Assumption (H):**

1. "The level set  is bounded, where  is the starting point" .
2. "In a neighborhood  of S , is continuously differentiable and its gradient is Lipchitz continuously, namely, there exists a constant  such that"

  (52)

"Under these assumptions on there exists a constant , such that":

 (53)

We know that the new search directions generated by (22), (29), (34) and (40) are always descent directions.

To ensure the global convergence property of these algorithms we need the following Lemmas:

**4.1 Lemma**

If (H, a) and (H, b) are hold. Then

  (54)

**Proof.**

From (18) and descent condition , we have



  (55)

"So, {} is a decreasing sequence. Since *f* is bounded below, there exists a constant  such that:"  . (56)

it follows that  (57a)

hence,  (57b)

**4.2**  **Lemma**

 If (H, a) and (H, b) hold, as well as the descent condition  , hold. Then

  (58a)

**Proof.**

Subtracting  from both sides of (18a) and using the Lipschitz condition, we get:

  (58b)

Therefore, using **Lemma (4.1)** we get (58a).

**4.3 Proposition**

Assume that (H, a) and (H, b) hold, and for any CG-algorithm defined in (2), where  is a descent direction and  is obtained by Wolfe condition (18). If

 , (59)

Then

 " (60)

Regularly for uniformly convex functions, we can prove that the norm of the search directions  ; ;  and  defined by (22), (29), (34) and (40) are bounded above. Therefore, by Proposition (4.3), we can prove the following result:

**4.4 Theorem**

Assume that (H, a) and (H, b) hold, and consider the algorithms (2), (22), (29) and (34), where N1; N2 and N3 are a descent directions and  is computed by (18). Suppose that *f* is a **uniformly convex function** on *S*, i.e. there exists a constant  such that:

 " " (61)

For all  then

 "" (62)

**Proof.**

Consider the algorithms (2), (22), (29) and (34). From "Lipschitz continuity", we have . Furthermore, from uniform convexity, . Using the Cauchy inequality, assumption (H, a) and (H, b) and the above inequalities, we have:

  (63)

  (64)

  (65)

  (66)

  (67)

 Therefore, using (63)-(67) in (22), we get:

 (68)

 (69)

 (70)

showing that (59) is true. By Proposition (4.3), it follows that (60) is true, which for uniformly convex functions is equivalent to (62).

**4.5 Theorem**

If (H, a) and (H, b) hold, for algorithms (2) and (40), where  defined in (40) is a sufficient descent direction and  is computed by Wolfe conditions (18) and if *f* is a **uniformly convex function** on *S*, then:

 

**Proof.**

For the algorithms (2) and (40), we have from "Lipchitz continuity", . From uniformly convexity, we have .

Choosing the Cauchy inequality, (H, a) and (H, b) and the above inequalities, we have:

  (71)

  (72)

  (73)

Therefore, using (71), (72) and (73) in the new algorithm (40), we get:

 (74)

Thus showing that (59) is true. By Proposition (4.3), it follows that (60) is true, which for uniformly convex functions is equivalent to (62).

**5. Numerical Results.**

 Here, we report the performance of the new proposed CG-algorithms, namely (N1; N2; N3 and N4) on a set of (39) large-scale nonlinear test problems (see the appendix for the details) using codes written in **Fortran** and compiled with F77. All the tests were performed on a PC. All the selected test problems are from (Bongartz, 1995); CUTE library.

 We have taken 10 numerical experiments with N = 1000, 4500, 10000 for each test function. In order to assess the reliability of our new proposed method, we have tested it against (Al-Bayati and Sharif; BS, 2010), using the same test problems and the same Wolfe line search conditions; namely, with ,  and the same "stopping criterion

  (75)

where  is the maximum absolute component of a vector. Also, this routine stopped if the iterations exceed 10000 or the number of function gradient evaluations reach 15000 without achieving convergence.

**Table1** compares N1, N2 and N3 against three-term BS;

**Table2** compares N1, N2 and N3 against three-term HS;

**Table3** compares N4 against three-term SPR.

In all these tables:

 **N** = "Dimension of the problem"; NOI = "Number of iterations".

 **NOFG** = "Number of function and gradient evaluations".

**TIME** = " Total time required to complete the evaluation process for each test Problem".

**These Tables the percentage performance of new algorithms against similar published Ones:**

**Tables4**, compares N1, N2 and N3 against BS;

**Tables5**, compares N1, N2 and N3 against TTHS;

**Tables6**, compares N4 against TTSPR.

All these comparisons are with respect to NOI; NOFG and TIME taking over all the Tools as 100%..

**Table (1)**

**Comparison of new algorithms against BS with N= 1000, 4500, 10000.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prob.** | **N1** | **N2** | **N3** | **BS (2010)** |
| **NOI** | **NOFG** | **TIME** | **NOI** | **NOFG** | **TIME** | **NOI** | **NOFG** | **TIME** | **NOI** | **NOFG** | **TIME** |
| 1 | 240 | 619 | 0.21 | 224 | 760 | 0.021 | 241 | 630 | 0.41 | 249 | 310 | 0.59 |
| 2 | 40 | 120 | 0.07 | 40 | 120 | 0.07 | 40 | 120 | 0.07 | 49 | 83 | 0.04 |
| 3 | 7 | 41 | 0.01 | 7 | 41 | 0.09 | 7 | 41 | 0.09 | 12 | 95 | 0.1 |
| 4 | 30 | 92 | 0.09 | 25 | 77 | 0.01 | 25 | 277 | 0.01 | 10 | 61 | 0.3 |
| 5 | 10 | 32 | 0.19 | 10 | 32 | 0.31 | 10 | 32 | 0.031 | 32 | 20 | 0.09 |
| 6 | 31 | 101 | 0.31 | 31 | 101 | 0.81 | 31 | 101 | 0.03 | 32 | 74 | 0.01 |
| 7 | 38 | 118 | 0.04 | 40 | 128 | 0.8 | 40 | 128 | 0.81 | 60 | 103 | 0.05 |
| 8 | 77 | 206 | 0.05 | 77 | 206 | 0.08 | 77 | 206 | 0.8 | 37 | 32 | 0.33 |
| 9 | 109 | 610 | 0.11 | 110 | 480 | 0.09 | 118 | 676 | 0.12 | 484 | 12566 | 2.00 |
| 10 | 95 | 308 | 0.01 | 100 | 300 | 0.0 | 99 | 410 | 0.08 | 199 | 3520 | 0.04 |
| 11 | 20 | 65 | 0 | 20 | 65 | 0.0 | 20 | 65 | 0.09 | 31 | 52 | 0.41 |
| 12 | 5 | 10 | 0 | 5 | 10 | 0.0 | 5 | 10 | 0 | 25 | 510 | 0.08 |
| 13 | 55 | 231 | 0.01 | 42 | 219 | 0.02 | 71 | 322 | 0 | 160 | 231 | 0.05 |
| 14 | 43 | 124 | 0.12 | 43 | 124 | 0.31 | 43 | 124 | 0.02 | 52 | 92 | 0.09 |
| 15 | 580 | 120 | 0.05 | 611 | 1200 | 0.05 | 699 | 1390 | 0.05 | 123 | 200 | 0.03 |
| 16 | 77 | 351 | 0.31 | 76 | 301 | 0.21 | 76 | 380 | 0.21 | 120 | 1843 | 0.03 |
| 17 | 11 | 33 | 0.07 | 11 | 33 | 0.01 | 11 | 33 | 0.3 | 15 | 29 | 0.01 |
| 18 | 172 | 521 | 0.11 | 151 | 561 | 0.1 | 141 | 421 | 0.3 | 100 | 191 | 0.04 |
| 19 | 22 | 142 | 0.03 | 25 | 140 | 0.09 | 25 | 169 | 0.04 | 172 | 4732 | 0.05 |
| 20 | 84 | 270 | 0.21 | 84 | 220 | 0.2 | 84 | 249 | 0.41 | 58 | 509 | 0.06 |
| 21 | 5 | 19 | 0.3 | 5 | 19 | 0.5 | 5 | 19 | 0.4 | 10 | 25 | 0.01 |
| 22 | 17 | 70 | 0.0 | 14 | 60 | 0.0 | 14 | 60 | 0.5 | 17 | 39 | 0.04 |
| 23 | 23 | 77 | 0.2 | 23 | 77 | 0.4 | 23 | 71 | 0.2 | 19 | 56 | 0.25 |
| 24 | 27 | 73 | 0.2 | 27 | 73 | 0.4 | 27 | 73 | 0.09 | 47 | 75 | 0.75 |
| 25 | 10 | 31 | 0.07 | 10 | 31 | 0.04 | 10 | 32 | 0.1 | 13 | 25 | 0.75 |
| 26 | 29 | 80 | 0.04 | 29 | 30 | 0.3 | 29 | 81 | 0.01 | 24 | 53 | 0.41 |
| 27 | 140 | 420 | 0.04 | 109 | 321 | 0.3 | 120 | 331 | 0.21 | 59 | 101 | 0.75 |
| 28 | 29 | 89 | 0.21 | 29 | 89 | 0.09 | 29 | 89 | 0.41 | 50 | 160 | 0.4 |
| 29 | 262 | 741 | 0.07 | 292 | 874 | 0.07 | 249 | 720 | 0.81 | 261 | 402 | 0.3 |
| 30 | 8 | 25 | 0.02 | 8 | 25 | 0.02 | 8 | 25 | 0.21 | 11 | 24 | 0.01 |
| 31 | 121 | 481 | 0.2 | 127 | 471 | 0.04 | 127 | 520 | 0.0 | 299 | 3918 | 0.01 |
| 32 | 89 | 580 | 0.3 | 90 | 430 | 0.06 | 111 | 620 | 0.02 | 400 | 12277 | 0.02 |
| 33 | 29 | 89 | 0.04 | 29 | 89 | 0.9 | 29 | 89 | 0.05 | 51 | 92 | 0.04 |
| 34 | 22 | 47 | 0.11 | 22 | 47 | 0.01 | 22 | 47 | 0.02 | 26 | 53 | 0.5 |
| 35 | 107 | 391 | 0.01 | 109 | 372 | 0.04 | 109 | 382 | 0.01 | 179 | 2171 | 0.5 |
| 36 | 35 | 113 | 0 | 35 | 113 | 0.2 | 35 | 113 | 0.01 | 50 | 91 | 0.2 |
| 37 | 76 | 210 | 0 | 70 | 201 | 0.1 | 90 | 289 | 0.02 | 201 | 179 | 0.2 |
| 38 | 21 | 67 | 0 | 21 | 67 | 0.2 | 21 | 67 | 0.01 | 60 | 83 | 0.1 |
| 39 | 101 | 403 | 0.07 | 121 | 401 | 0.07 | 113 | 402 | 0.09 | 232 | 3210 | 0.0 |
| **Total** | **2897** | **9114** | **3.88** | **2902** | **8902** | **5.73** | **3034** | **9814** | **7.041** | **4029** | **48287** | **9.64** |

**Table (2)**

**Comparison of new algorithms against TTHS with N= 1000, 4500, 10000.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prob.** | **N1** | **N2** | **N3** | **TTHS** |
| **NOI** | **NOFG** | **TIME** | **NOI** | **NOFG** | **TIME** | **NOI** | **NOFG** | **TIME** | **NOI** | **NOFG** | **TIME** |
| 1 | 240 | 619 | 0.12 | 224 | 760 | 0.021 | 241 | 630 | 0.41 | 242 | 362 | 0.21 |
| 2 | 40 | 120 | 0.07 | 40 | 120 | 0.07 | 40 | 120 | 0.07 | 39 | 100 | 0.04 |
| 3 | 7 | 41 | 0.01 | 7 | 41 | 0.09 | 7 | 41 | 0.09 | 13 | 99 | 0.01 |
| 4 | 30 | 92 | 0.09 | 25 | 77 | 0.01 | 25 | 77 | 0.01 | 30 | 61 | 0.1 |
| 5 | 10 | 32 | 0.19 | 10 | 32 | 0.31 | 10 | 32 | 0.31 | 10 | 24 | 0.3 |
| 6 | 31 | 101 | 0.31 | 31 | 101 | 0.03 | 31 | 101 | 0.03 | 33 | 79 | 0.4 |
| 7 | 38 | 118 | 0.04 | 40 | 128 | 0.81 | 40 | 128 | 0.81 | 42 | 102 | 0.09 |
| 8 | 77 | 206 | 0.05 | 77 | 206 | 0.8 | 77 | 206 | 0.8 | 25 | 30 | 0.07 |
| 9 | 109 | 610 | 0.11 | 110 | 480 | 0.08 | 118 | 676 | 0.12 | 493 | 1270 | 0.01 |
| 10 | 95 | 308 | 0.01 | 100 | 300 | 0.09 | 99 | 410 | 0.08 | 200 | 3601 | 0.8 |
| 11 | 20 | 65 | 0 | 20 | 65 | 0.0 | 20 | 65 | 0.09 | 37 | 77 | 0.2 |
| 12 | 5 | 10 | 0 | 5 | 10 | 0.0 | 5 | 10 | 0 | 37 | 541 | 0.4 |
| 13 | 55 | 231 | 0.01 | 42 | 219 | 0.02 | 71 | 322 | 0 | 132 | 227 | 0.01 |
| 14 | 43 | 124 | 0.12 | 43 | 124 | 0.31 | 43 | 124 | 0.02 | 59 | 121 | 0.01 |
| 15 | 580 | 1120 | 0.05 | 611 | 1200 | 0.05 | 699 | 1390 | 0.05 | 182 | 248 | 0.03 |
| 16 | 77 | 351 | 0.31 | 76 | 301 | 0.21 | 76 | 380 | 0.21 | 176 | 1981 | 0.01 |
| 17 | 11 | 33 | 0.07 | 11 | 33 | 0.01 | 11 | 33 | 0.3 | 16 | 27 | 0.1 |
| 18 | 172 | 521 | 0.11 | 191 | 561 | 0.1 | 141 | 421 | 0.3 | 107 | 200 | 0.5 |
| 19 | 22 | 142 | 0.03 | 25 | 140 | 0.09 | 25 | 169 | 0.04 | 172 | 4832 | 0.5 |
| 20 | 84 | 270 | 0.21 | 84 | 220 | 0.2 | 84 | 249 | 0.41 | 41 | 139 | 0.5 |
| 21 | 5 | 19 | 0.3 | 5 | 19 | 0.5 | 5 | 19 | 0.4 | 12 | 22 | 0.4 |
| 22 | 17 | 70 | 0.0 | 14 | 60 | 0.0 | 14 | 60 | 0.5 | 17 | 35 | 0.03 |
| 23 | 23 | 71 | 0.2 | 23 | 71 | 0.4 | 23 | 71 | 0.2 | 18 | 53 | 0.03 |
| 24 | 27 | 73 | 0.2 | 27 | 73 | 0.4 | 27 | 73 | 0.09 | 50 | 75 | 0.04 |
| 25 | 10 | 31 | 0.07 | 10 | 31 | 0.04 | 10 | 32 | 0.1 | 13 | 25 | 0.05 |
| 26 | 29 | 80 | 0.04 | 29 | 80 | 0.3 | 29 | 81 | 0.01 | 24 | 53 | 0.04 |
| 27 | 140 | 420 | 0.04 | 109 | 321 | 0.3 | 120 | 331 | 0.21 | 71 | 110 | 0.02 |
| 28 | 29 | 89 | 0.21 | 29 | 89 | 0.09 | 29 | 89 | 0.41 | 60 | 171 | 0.08 |
| 29 | 262 | 741 | 0.07 | 292 | 874 | 0.07 | 249 | 720 | 0.81 | 261 | 400 | 0.2 |
| 30 | 8 | 25 | 0.02 | 8 | 25 | 0.2 | 8 | 25 | 0.21 | 14 | 26 | 0.01 |
| 31 | 121 | 481 | 0.2 | 127 | 471 | 0.04 | 127 | 520 | 0.0 | 281 | 2960 | 0.01 |
| 32 | 89 | 580 | 0.3 | 90 | 430 | 0.06 | 111 | 620 | 0.02 | 423 | 1309 | 0.02 |
| 33 | 29 | 89 | 0.04 | 29 | 89 | 0.9 | 29 | 89 | 0.05 | 60 | 98 | 0.03 |
| 34 | 22 | 47 | 0.11 | 22 | 47 | 0.01 | 22 | 47 | 0.02 | 24 | 50 | 0.07 |
| 35 | 107 | 391 | 0.01 | 109 | 372 | 0.04 | 109 | 382 | 0.01 | 179 | 2141 | 0.05 |
| 36 | 35 | 113 | 0 | 35 | 113 | 0.2 | 35 | 113 | 0.01 | 75 | 102 | 0.3 |
| 37 | 76 | 210 | 0 | 70 | 201 | 0.1 | 90 | 289 | 0.02 | 234 | 189 | 0.1 |
| 38 | 21 | 67 | 0 | 21 | 67 | 0.2 | 21 | 67 | 0.01 | 90 | 90 | 0.09 |
| 39 | 101 | 403 | 0.07 | 121 | 401 | 0.07 | 113 | 402 | 0.09 | 3410 | 3410 | 0.01 |
| **Total** | **2897** | **9114** | **3.88** | **2902** | **8902** | **5.73** | **3034** | **9814** | **7.04** | **3996** | **24687** | **8.74** |

**Table (3)**

**Comparison of new algorithms against TTSPR with N= 1000, 4500, 10000.**

|  |  |  |
| --- | --- | --- |
| **Prob.** | **N4** | **TTSPR** |
| **NOI** | **NOFG** | **TIME** | **NOI** | **NOFG** | **TIME** |
| **1** | 23 | 66 | 0.2 | 54 | 160 | 0.01 |
| **2** | 60 | 240 | 0.2 | 160 | 1401 | 0.32 |
| **3** | 161 | 430 | 0.1 | 171 | 292 | 0.11 |
| **4** | 30 | 91 | 0.01 | 29 | 51 | 0.01 |
| **5** | 19 | 55 | 0.04 | 19 | 34 | 0.01 |
| **6** | 19 | 42 | 0.1 | 19 | 27 | 0.03 |
| **7** | 5 | 34 | 0.06 | 10 | 84 | 0.04 |
| **8** | 77 | 212 | 0.2 | 87 | 181 | 0.02 |
| **9** | 19 | 67 | 0.24 | 18 | 40 | 0.05 |
| **10** | 5 | 20 | 0.1 | 8 | 20 | 0.03 |
| **11** | 9 | 29 | 0.18 | 9 | 20 | 0.08 |
| **12** | 21 | 80 | 0.07 | 19 | 55 | 0.02 |
| **13** | 21 | 70 | 0.26 | 31 | 61 | 0.3 |
| **14** | 20 | 61 | 0.06 | 55 | 90 | 0.04 |
| **15** | 16 | 51 | 0.01 | 14 | 35 | 0.05 |
| **16** | 24 | 71 | 0.25 | 28 | 50 | 0.01 |
| **17** | 57 | 258 | 0.21 | 588 | 1809 | 0.1 |
| **18** | 10 | 32 | 0.11 | 14 | 29 | 1.1 |
| **19** | 66 | 242 | 0.3 | 147 | 1809 | 1.2 |
| **20** | 12 | 45 | 0.2 | 39 | 61 | 0.02 |
| **21** | 19 | 57 | 0.16 | 35 | 58 | 0.01 |
| **22** | 15 | 45 | 0.13 | 8 | 36 | 0.02 |
| **23** | 4 | 14 | 0.01 | 27 | 530 | 0.3 |
| **24** | 26 | 91 | 0.3 | 160 | 241 | 0.3 |
| **25** | 20 | 61 | 0.1 | 20 | 40 | 0.05 |
| **26** | 13 | 45 | 0.01 | 51 | 82 | 0.01 |
| **27** | 6 | 21 | 0.25 | 12 | 27 | 0.09 |
| **28** | 11 | 29 | 0.02 | 15 | 30 | 0 |
| **29** | 13 | 41 | 0.2 | 39 | 70 | 0.01 |
| **30** | 51 | 191 | 0.2 | 89 | 971 | 0.2 |
| **31** | 8 | 23 | 0.09 | 11 | 19 | 0.05 |
| **32** | 27 | 79 | 0.08 | 39 | 110 | 0.02 |
| **33** | 131 | 306 | 0.1 | 90 | 161 | 0.07 |
| **34** | 17 | 50 | 0.6 | 22 | 50 | 0.04 |
| **35** | 81 | 207 | 0.01 | 99 | 170 | 0.3 |
| **Total** | **1116** | **3456** | **5.16** | **2236** | **23995** | **5.02** |

**Table (4)**

**Percentage Performance of new algorithms against BS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tools** | **BS** | **N1** | **N2** | **N3** |
| **NOI** | **100%** | **72.49%** | **72.62%** | **75.92%** |
| **NOFG** | **100%** | **36.91%** | **36.05%** | **39.75%** |
| **CPU** | **100%** | **44.39%** | **65.56%** | **80.54%** |

**Table (5)**

**Percentage Performance of new algorithms against TTHS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tools** | **TTHS** |  **N1** | **N2** | **N3** |
| **NOI** | **100%** | **72.49%** | **72.62%** | **75.92%** |
| **NOFG** | **100%** | **36.91%** | **36.05%** | **39.75%** |
| CPU | 100% | 44.39% | 65.56% | 80.54% |

**Table (6)**

**Percentage Performance (N4) against TTSPR**

|  |  |  |
| --- | --- | --- |
| **Tools** | **TTSPR**  | **N4** |
| **NOI** | **100%** | **49.91%** |
| **NOFG** | **100%** | **14.40%** |
| **CPU** | **100%** | **97.28%** |

**5.1 Result Discussions.**

**From Table (4) taking, the tools as a 100% for BS method we have:**

For N1 (27.51%) NOI, (63.91%) NOFG and (55.61%) TIME.

For N2 (27.39%) NOI, (63.95%) NOFG and (34.44%) TIME.

For N3 (24.08%) NOI, (60.25%) NOFG and (19.46%) TIME.

**From Table (5); taking, the tools for TTHS method as a 100%, we have:**

For N1 (27.51%) NOI, (63.91%) NOFG and (55.61%) TIME.

For N2 (27.39%) NOI, (63.95%) NOFG and (34.44%) TIME.

For N3 (24.08%) NOI, (60.25%) NOFG and (19.46%) TIME.

**From Table (6); taking, the tools for TTSPR method as a 100% , we have**

N4 has about (50.09%) NOI , (85.8%) NOFG and (2.75%) TIME.

**6. Conclusions.**

In this work , we have investigated new three-term multi-step search directions defined in (17). The goal of these new algorithms is a multi-step property that combines three-term CG-techniques with memoryless QN-updates.

Our theoretical implementation related to be satisfies the requirement of sufficient descent properties, and ensures the property of global convergence. In addition, we have presented four new three-term multi-step CG-algorithms, which they use Wolfe's and it's acceleration as a line search subprogram.

 Moreover, Our numerical results show that our new algorithms have robust numerical results as compared to other similar algorithms in the same field.

**Data Availability**

The used data are available from the corresponding author upon request.

**Conflicts of Interest**

The authors declare that there are no conflicts of interest regarding this work.

**Acknowledgments**

The research is supported by College of Computer Sciences and Mathematics, University of Mosul, Republic of Iraq,

**Appendix:** Details of the selected set of test functions.

**1- Extended Trigonometric Function.**



**2- Extended Penalty Function ز**

  

**3- Raydan2 Function.**

** **

**4- Hager Function.**

 ****

**5- Generalized Tri-diagonal Function.**

,.

**6- Extended Three Exponential Function.**

**7- Diagonal-4 Function.**

****  ****

**8- Diagonal-5 Function.**

 

**9- Extended Himmelblau Function.**



**10- Generalized PSC1-Function.**

 

**11- Extended Block Diagonal BD1-Function.**

****

**12- Extended QP1-Function**.

  

**13- Extended EP1-Function**



**14- Extended Tri-diagonal-2 Function.**

****

 ****

**15- ARWHEAD Function.**



**16- DIXMAANA Function.**

****

**17- DIXMAANB Function.**

**18- DIXMAANC Function.**

**19- EDENSCH Function.**



**20- DIAGONA- 6 Function.**



**21- ENGVALI Function.**



**22- DENSCHNA Function.**



**23- DENSCHNC Function.**



**24- DENSCHNB Function.**



**25- DENSCHNF Function.**



**26- Extended Block–Diagonal BD2-Function.**



**27-Generalized quadratic GQ1-Function.**



**28- DIAGONAL-7 Function.**



**29- DIAGONAL-8 Function.**



**30- Full Hessian Function.**



**31- SINCOS Function.**



**32- Generalized quadratic GQ2-Function.**



**33- ARGLINB Function.**



**34- HIMMELBG Function.**



**35- HIMMELBH Function.**



**36- Extended Beale function.**

**37-Broyden Tri-diagonal function.  **

**38) Fletcher (CUTE) Function.**



**39) HIMMELBHA Function.**

****

**References:**

1. Al-Bayati, A.Y. (1991). A new family of self-scaling variable metric algorithms for unconstrained optimization, Journal of Educ. and Sci., University of Mosul, Iraq, 12, 25-54.
2. Al-Bayati, A. Y. and, Sharif, W. H. (2010). A new three-term conjugate gradient method for unconstrained optimization, Canadian. J. Sci. Eng. Math. 1 (5), 108–124.
3. Al-Bayati, A. Y. and Hassan, B. A. (2011), A Modified Three-Term Conjugate Gradient Method, Advances in Modeling and Analysis A, (AMSE), France, 48 (1-2), 1-14.
4. Al-Bayati, A. Y. and Al-Khayat, H. N. (2014) .Different Types of Three-Term CG-Methods with Sufficient Descent and Conjugacy Conditions, Journal of Zankoy Suleimani Part-A for Pure and Applied Sciences, Suleimani, Iraq, 16(2), 27-45.
5. Al-Bayati A. Y; Ahmed, H. I. and Hamed, E. T. (2019). A New Hybrid Algorithm for Convex Nonlinear Unconstrained Optimization, Journal of Applied Mathematics, Article ID 8728196, 1-6.
6. Andrei, N. (2009). Acceleration of conjugate gradient algorithms for unconstrained optimization, Applied Mathematical Computational, 213, 361–369.
7. Beale, E.M.L. (1972).A derivation of conjugate gradients", in numerical methods for nonlinear optimization, (Lootsma, F.A.) New York, 39-43.
8. Bongartz, K. E., Conn A. R., Gould N. I. M. and Toint, P. L., (1995). CUTE, constrained and unconstrained testing environments, ACM Trans. Math. Software, 21, 123-160.
9. Dai, Y. and Liao, L. (2001). New conjugacy conditions and related nonlinear conjugate gradient methods, Appl. Math Optim., 43. Spring-Verlag, New York.
10. Dennis, J. E., Jr., and Schnabel, R. E. (1983). Numerical methods for unconstrained optimization and nonlinear equations, Prentice-Hall, New Jersey.
11. Hestenes, M. R. and Stiefel, E. (1952), Methods of conjugate gradient for solving linear system, Journal of Research of the National Bureau of Standards , 49, 409–436.
12. Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization, Mathematical Program, 45, 503–528.
13. Nazareth, L. (1977). A conjugate direction algorithm without line search, Journal of Optimization Theory and Applications, 23, 373–387.
14. Nazareth L. and Nocedal J. (1978). Properties of CG methods with Inexact line searches, Report 780-1 system optimization laboratory, department of Operation Research, Stanford University, USA.
15. Nocedal, J. (1996). Conjugate gradient methods and nonlinear optimization, (Adams and Nazareth) ,Linear and Nonlinear Conjugate Gradient—Related Methods, 9–23. SIAM, Philadelphia.
16. Oren, S. S. (1974). On the selection of parameters in self-scaling variable metric algorithms, Mathematical Programming, 3, 351-367.
17. Perry, A. (1978). A modified conjugate gradient algorithm, Operations Research, 26, 1073–1078.
18. Polak, E. and Ribière, G.(1969). Note Sur la Convergence de Directions Conjuguée, Rev. Francaise Informat Recherche Operationelle, 3e Année, 16, 35–43.
19. Shanno, D. (1978). Conjugate gradient methods with inexact searches, Math. of operation Research, 3, 244-256.
20. Zhang, L. Zhou, W. Li. D. (2006). A descent modified Polak- Ribière-Polyak conjugate gradient method and its global convergence, IMA Journal of Numerical Analysis , 26, 629–640.
21. Zhang, L. Zhou, W. Li, D. (2007). Some descent three-term conjugate gradient methods and their global convergence, Optimization Methods and Software , 22, 697– 711.
22. Zhang, L. Xiao, Y. Wei. Z. (2009). Nonlinear conjugate gradient methods with sufficient descent condition for large-scale unconstrained optimization, Math. Prob. Eng. (2009), Article ID 243290, 16.

.