NEW SWITCHING ALGORITHM FOR COMBINING
NEW MULTI-STEP CG AND SELF-SCALING
VM ALGORITHMS FOR NONLINEAR
OPTIMIZATION

By
Abbas Y. Al Bayati,
Dept. of Mathematics, College of Science
University of Mosul, Iraq.

Muna M. Mohammad Ali,
Dept. of Mathematics, College of Science
University of Mosul, Iraq.

Abstract

New interleaved CG and self-scaling VM algorithm is presented in this research which combines Al Bayati’s self-scaling VM algorithm with another new multi-step CG algorithm with inexact line searches (ILS). In an interleaving algorithm a VM-update can be initiated between each CG-step and the technique improves the rate of convergence of the new proposed algorithms. The new algorithms are tested against the standard Hestenes-Stiefel and Buckley’s algorithms for a number of well-known test functions, with encouraging results.

Key Words: Switching algorithm, Monlinear Optimization.

205
1 - INTRODUCTION

Two of the most important classes of algorithms for function minimization are Variable Metric (VM) and Conjugate Gradient (CG) algorithm. It is well-known that VM algorithms require fewer iterations and do not require very accurate line searches. On the other hand, CG algorithms do not require matrix storage and have proved themselves appropriate for large problems where the matrix of second derivatives is not spare (see for example Fletcher, [1]). However, the class of interleaved algorithms which incorporate both a CG and a VM algorithms are "intermediate" in that they use only few storage locations and they accelerate the rate of convergence of CG algorithms. We now define explicitly the general VM and CG algorithms, and hence the new algorithms with their inexact line search.

2. WELL-KNOWN RESULTS:

Let \( f : \mathbb{R}^n \to \mathbb{R} \) be a twice differentiable function defined on an open set \( E \). We consider the problem of finding \( x \in E \) such that:

\[
\min_{x \in E} f(x) = f(x_{\text{min}})
\]

and \( v_k = x_{k+1} - x_k \) \( (3) \)

Then the Broyden [2] 9-class VM-family can be expressed as follows: given \( x_1 \) and an arbitrary \( n \times n \) positive definite matrix \( H_1 \), iterate for \( k = 1, 2, 3, \ldots \) with

\[
x_{k+1} = x_k + \alpha_k v_k
\]

where \( \alpha_k \) is a steplength and \( H_k \) is defined by:

\[
H_{k+1} = H_k - \frac{H_k y_k v_k^T}{v_k^T H_k y_k} + \theta_k \omega_k \omega_k^T + \rho_k \frac{v_k v_k^T}{v_k^T H_k y_k}
\]

where:

\[
\omega_k = (y_k^T H_k y_k)^{1/2} \left[ (v_k v_k^T) - (H_k y_k y_k^T H_k y_k) \right]
\]

and \( \theta_k, \rho_k \) are parameters \( \geq 0 \). The well known BFGS (Broyden, Fletcher, Goldfarb and Powell) update corresponds to \( \rho_k = 1; \ \theta_k = 1 \), while the DFP (David, Fletcher and Powell) update arises from \( \rho_k = 1; \ \theta_k = 0 \). In this research we are mainly concerned with Al Bayati's (1991) self-scaling VM update, for which \( \theta_k = 1, \rho_k = 1 \).

\[
\beta_k = \frac{y_k^T H_k y_k}{(y_k^T H_k y_k)^{1/2}} (v_k v_k^T) - (H_k y_k y_k^T H_k y_k)
\]

3. MULTI-STEP CG-METHODS:

Let \( f \) be a strictly convex quadratic function defined on \( \mathbb{R}^n \):

\[
f(x) = x^T A x + b^T x
\]

where \( A \) is a symmetric and strictly positive definite \( n \times n \) matrix and \( b \in \mathbb{R}^n \). A set of non-zero vectors \( (d_1, d_2, \ldots, d_n) \) is defined as mutually conjugate with respect to \( A \) if:

\[
d_i^T A d_k = 0 \quad \text{for all} \quad i \neq k
\]

Now, the CG-method was originally developed by Hestenes and Stiefel, (1952) to solve systems of linear equations; it has been updated to solve minimization problem (1) in the following form given arbitrary \( x_1 \), let \( d_1 = -g_1 \), iterate for \( k = 1, 2, 3, \ldots \) with

\[
d_{k+1} = -g_{k+1} + \beta_k d_k,
\]

\[
x_{k+1} = x_k + \lambda_k d_k
\]

where \( \lambda_k \) minimizes \( f \) in the direction \( d_k \). The scalar \( \lambda_k \) is called an exact minimizer (ILS) if:

\[
f(x_k + \lambda_k d_k) \leq f(x_k + \omega d_k) \quad \text{for all} \quad \omega \in (0, \omega_0).
\]

Different choice of the parameter \( \beta_k \) gives several different algorithms, including:

\[
\beta_k = \begin{cases} \frac{y_k^T H_k y_k}{(y_k^T H_k y_k)^{1/2}} \quad \text{Fletcher-Reeves (FR)} \\ \frac{y_k^T H_k y_k}{(y_k^T y_k)^{1/2}} \quad \text{Polak- Ribiere (PR)} \\ \frac{y_k^T y_k}{(y_k^T y_k)^{1/2}} \quad \text{Hestenes-Stiefel (HS)} 
\end{cases}
\]

We note that for a quadratic function the three formulae for \( \beta_k \) are the same provided ELS are used (see Fletcher, 1987).

ALGORITHM I (Hestenes and Stiefel):

Now, the CG-method was originally developed by Hestenes and Stiefel, (1952) to solve systems of linear equations; it has been updated to solve minimization problem (1) in the following form given arbitrary \( x_1 \), let \( d_1 = -g_1 \), iterate for \( k = 1, 2, 3, \ldots \) with

\[
d_{k+1} = -g_{k+1} + \beta_k d_k,
\]

\[
x_{k+1} = x_k + \lambda_k d_k
\]

where \( \lambda_k \) minimizes \( f \) in the direction \( d_k \). The scalar \( \lambda_k \) is called an exact minimizer (ILS) if:

\[
f(x_k + \lambda_k d_k) \leq f(x_k + \omega d_k) \quad \text{for all} \quad \omega \in (0, \omega_0).
\]
For the conjugacy condition to hold for the set d with respect to A then
\[ \beta_{ik} = (\gamma_i^T g_{k+1}) / k_i^T d_i; \quad i = 1, 2, 3, ..., k. \quad (16) \]

Particular CG-algorithms that do not require ELS have been introduced by Dixon (1975); Nazareth (1977); Shanno (1978). Nazareth-Nocedal (1978) developed a multistep CG-method which does not need ELS by defining matrices
\[ D = (d_1, d_2, \ldots, d_n) \]
and \[ G = (g_1, g_2, \ldots, g_n) \] and expressing (15) by
\[ -G = DB, \quad (17) \]
where B is an nxn upper triangular matrix with \( j_{3ii} = 1 \) for \( i = 1, 2, \ldots, k \). Using ELS in (15) yields:
\[ y_{i}^T g_{k+1} = 0 \quad \text{for} \quad i = 1, 2, \ldots, k-1, \quad (18) \]
and hence
\[ \beta_{ik} = 0 \quad \text{for} \quad i = 1, 2, 3, \ldots, k-1. \quad (19) \]

Hence, the conjugate search directions of the multi-step CG-method can be summarized as follows:

**ALGORITHM II ( Nazareth-Nocedal):**

\[ d_i = -g_i, \]

iterate for \( k = 1, 2, 3, \ldots, \) with
\[ \rho_{k+1} = -g_{k+1} + (\gamma_{k}^T g_{k+1}) / (\gamma_{k}^T d_{k}) d_{k}, \quad (20a) \]
\[ d_{k+1} = -\rho_{k+1} + c_k, \quad (20b) \]
\[ (c_{k+1} + [ (\gamma_{k}^T g_{k+1}) / (\gamma_{k}^T d_{k}) ] d_{k+1} : \quad k > 1, \quad c_k = 10; \quad k = 1 \quad (20c) \]

In the above algorithm using ILS not all the coefficients of the Gram-Schmidt process have to be computed at every iteration. For the \((k+1)\)th step, when computing \( d_{k+1} \) the coefficients for \( d_1, d_2, d_3, \ldots, d_{k-2} \) are already known since (see Nazareth-Nocedal 1978) it is established that:
\[ y_{i}^T y_{i+k} = 0 \quad \text{for} \quad k \geq 2. \quad (21) \]

Hence
\[ y_{i}^T y_{i+2} = 0 \quad (22a) \]
\[ y_{i}^T y_{i+3} = 0 \quad (22b) \]
and so
\[ y_{i}^T E_{i+2} = y_{i}^T E_{i+3} = \ldots \quad (22c) \]

Hence, only two new coefficients have to be computed and only two previous search directions must be stored; the contribution of the components along \( d_1, d_2, \ldots, d_{k-2} \) to the new direction \( d_{k+1} \) can be accumulated in a single vector \( c_k \). Similarly, the correction vector \( e_k \) to the current iterate \( x_k \) can also be accumulated in a single vector such that for the quadratic function (12) the correction term is defined by:
\[ e_{k+1} = e_k + c_k d_k \quad \text{for} \quad k \geq 1; \quad (23) \]

and so
\[ x_{\min} = x_{n+1} + e_{n+1}. \quad (24) \]

where
\[ e_k = -y_k \left( g_{k+1}^T d_k \right) / (y_k^T d_k) \], with \( e_1 = 0 \) initially (25) for further details see Nazareth (1977).

Algorithm II generates mutually conjugate search directions with respect to the Hessian matrix A for the system of equations defined in (20). For more details see Nazareth and Nocedal (1978).

**ALGORITHM III (NEW1):**

However, in this research we have assumed that \( A = I \), the identity matrix. Eq. (20) generates I-conjugate vectors, i.e. they are also mutually orthogonal. Based on the above idea we have constructed a new set of mutually orthogonal vectors \( g_1, g_2, \ldots, g_n \) which are a linear combination of the normal gradient terms \( g_1, g_2, \ldots, g_n \) defined as follows:
\[ g_1^* = g_1, \quad (26a) \]

iterate for \( k = 2, 3, 4, \ldots \) with
\[ g_k^* = g_k - \left( \frac{\left( g_k g_{k-1}^* \right) g_{k-1}^*}{g_k^* g_{k-1}^*} \right) g_k^*; \quad (26b) \]
\[ g_k = g_k^* + c_k^*; \quad (26c) \]
\[ c_k^* = \begin{cases} 0 & \text{if } k = 1, 2 \quad (26d) \end{cases} \]

We now will prove that the set of orthogonal vectors defined in (26) will estimate the set of normal gradient vectors as follows:

The mutually conjugate search directions for algorithm III can be expressed as follows:
\[ d_i = -g_i^*, \quad (27a) \]

iterate for \( k = 1, 2, 3, \ldots \) with
New Switching Algorithm for Combining

\[ d_{k+1} = -g^*_{k+1} + \beta_k^* d_k \]  
(27b)

where

\[ \beta_k^* = \left( g^*_{k+1} \mathbf{T} y_*^k \right) / \left( d_k^T y_*^k \right) \]  
(27c)

and

\[ y_*^k = g^*_{k+1} - g_k^* \]  
(27d)

where \( g^*_{k+1} \) is a vector defined in (26).

However, this algorithm also generates mutually conjugate directions for the quadratic function (12) and the set \( \{g^*, g_2^*, ..., g_n^*\} \) provided that ELS are used (see theorem 1).

**THEOREM 1**: The set of orthogonal vectors \( \{g_i^* : i=1, 2, ..., n\} \) defined in (26) coincide with the set \( \{g_i : i=1, 2, ..., n\} \) in the case of the quadratic function (12) using ELS.

**Proof**: Proceed by induction:

\[ g_1^* = g_1, \text{ by assumption} \]

For \( i = 2, \) we have

\[ g_i^* = g_i - \left( g_2^T g_1^* \right) / \left( g_1^T g_1 \right) g_1^* \]  
(28)

since \( g_1^* = g_1 \) and \( g_2^T g_1 = 0 \) (orthogonality property is satisfied for the quadratic function with ELS); hence

\[ g_2^* = g_2. \]

Suppose now that the theorem is true for \( i, \) i.e.

\[ g_j^* = g_j, \text{ for } j = 1, 2, 3, ..., 1. \]  
(29)

To complete the proof of the theorem we must show it is true for \( i+1. \) Consider

\[ 2\delta g_i = g_{i+1} - \frac{g_{i+1}^T g_i^*}{g_i^T g_i} g_i + \frac{g_{i+1}^T g_i^*}{g_i^T g_i} g_i - \frac{g_{i+1}^T g_i g_i}{g_i^T g_i} g_i - ... \frac{g_{i+1}^T g_i g_i}{g_i^T g_i} g_i. \]  
(30)

Since all the gradients are mutually conjugate with respect to A in the set \( \{g_i : i = 1, 2, 3, ..., n\} \) and (29) is true, then clearly

\[ g_{i+1}^* = g_{i+1}. \]

For ILS and for general functions, the set \( \{g_i^* : i = 1, 2, 3, ..., n\} \) will only estimate the normal set of gradient vectors \( \{g_i : i = 1, 2, 3, ..., n\} \). For the quadratic function (12) the search directions defined in (27) are identical to the standard CO-method provided that ELS are used. But for general functions, algorithm III restarts with the stepest descent direction every \( n \) iteration or whenever Powell’s (1977) restarting criterion is satisfied, i.e. if either of the inequalities:

\[ g^T_{k+1} g_k^* \geq 0.2 \]  
(31a)

\[ g^T_{k+1} g_{k+1} \]  
(31b)

are satisfied.

4. NEW INTERLEAVING VM-CG METHOD

**ALGORITHM (IV)**:

In this section we have to present another new algorithm which uses both VM and CG-steps. Algorithm (IV) is particularly suitable for cases where less than \( nxn \) storage locations are conveniently available to store the VM matrix, and they perform VM-steps and CG-steps sequentially. This type of algorithms was originally studied by Buckley (1978).

For the outlines of the this new algorithm we consider the general iteration:

\[ d_i = -H g_i, \]  
(32a)

where \( g_i \) is arbitrary, and \( H \) is any symmetric and positive definite matrix, iterate for \( k = 1, 2, 3, ... \) with

\[ d_{k+1} = -H g_{k+1} + \beta_{k+1}^* d_k, \]  
(33a)

\[ x_{k+1} = x_k + \lambda_k d_k, \]  
(33b)

\[ \beta_{k+1}^* = (g_{k+1}^* H y_k) / (d_k^T y_k), \]  
(33c)

where \( \lambda_k \) is a steplength determined by ILS.

The new interleaved algorithm uses iteration (33) and Al-Bayati’s (1991) self-scaling VM updates as follows:

Given arbitrary \( x_i \) and matrix \( H_i \) (usually \( H_i = I \)), set \( t = 1 \).

**step 1:** set \( d_1 = -H g_1 \).

**step 2:** for \( k = t, t+1, t+2, ... \) iterate with (33)

**step 3:** check if

\[ \left| d_k^T y_k \right| > 0.0015 \]  
(34)

is satisfied then reset \( t \) to the current \( k \). Eq (34) was given by Dixon (1985).

Update \( H_i \) by

\[ H_{i+1} = H_i - \frac{H_i y_i v_i^T + v_i (2 y_i^T H_i y_i - H_i y_i v_i^T) y_i}{v_i^T y_i} \]  
(35)

**step 4:** replace \( i \) by \( i+1 \) and repeat from step 1.

For the VM-steps it is necessary that \( v_i^T y_i > 0 \) to ensure that \( H_{i+1} \) will be positive definite. Replace this condition by the equivalent one:
\[ d_{k+1}^T g_{k+1} < \mu_k d_k^T g_k \]  

(36)

where \( \mu_k \) is constant less than 1. Hence, we adopt the condition (36) in implementing our new algorithm but use it explicitly with \( \mu_k = 0.9 \) which corresponds to the best value for ILS in this case. Buckley (1978b) also proved that his combined QN-CG algorithm and the standard CG method generate identical sequences of points \( x_k \), if the two algorithms start from the same starting point. This means that the change in the metric along the QN-step does not prevent the mixed algorithm from terminating in \( n \) steps or less in the case of quadratic function, provided that the BFGS update is used.

For this new algorithm it is clear that Al-Bayati’s (1991) self-scaling VM update (7) will not effect the quadratic termination property.

**ALGORITHM (V) BUCKLEY:**

To measure the performance of the new proposed interleaving algorithm (IV) we have compared it with Buckley’s (1978a) interleaving algorithm. Buckley’s algorithm differs than the new algorithm in two major steps:

First: Buckley’s algorithm uses the BFGS update while the new algorithm uses Al-Bayati’s self-scaling VM-update.

Second: Buckley’s switching criterion is

\[ \| g_k^T g_k \| > 0.2 \| g_{k+1}^T g_{k+1} \| \]  

(37)

while the new algorithm uses eq (34) as new switching criterion.

**5. NUMERICAL COMPARISONS.**

Thirty standard test functions (see Appendix) were tried with a range of dimensions in order to examine the effectiveness of the new proposed algorithms. The numerical experiments were performed on the IBM-Elonex personal computer with double precision arithmetic with programs written in FORTRAN. The line search routine used was a cubic interpolation which uses function and gradient values and it is an adaptation of the routine published in (12).

The following five algorithms were tested: the first (HS) corresponds to the standard Hestenes and Stiefel CG-method, the second (NEW1) is the new multi-step CG-method, the third is Buckley’s method, (BUCKLEY), the fourth is the second new interleaving algorithm (NEW2) and the fifth is Al-Bayati’s self-scaling VM-method (BAYATI). The numerical results for these five algorithms are presented in four tables. The performance indicators employed are; total number of function calls (NOF), and total number of iterations (NOI) required to solve each test function, using the following stopping criterion:

\[ (g_k^T g_k) < 1 \times 10^{-5} \]  

(38)

Finally, the computational results documented in this section show that the new algorithms generally performs more efficiently than the other algorithms, in terms of NOF, for most of our test functions. The new algorithms almost always require a lower number of iterations and function calls than the well-know standard HS-method with overall savings in the range (12-25)% for NOF and (11-39)% for NOI for small dimensionality test functions (see Table-1-). Now for moderate dimensionality test functions there are (10-35)% for NOF and (27-50)% for NOI. For large dimensionality test functions there are improvements in the NOF about (35-38)% and (44-66)% for NOI. However, the new interleaved algorithms uses only moderate storage and they generally accelerate the CG-method. However, CG-methods remain valuable for the very large problems since only few vector storage locations are required.
New Switching Algorithm for Combining

**TABLE (1)**

<table>
<thead>
<tr>
<th>TEST FUNCTIONS</th>
<th>N</th>
<th>HS NOI (NOF)</th>
<th>NEW1 NOI (NOF)</th>
<th>BAYATI NOI (NOF)</th>
<th>BUCKLEY NOI (NOF)</th>
<th>NEW2 NOI (NOF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBIC</td>
<td>2</td>
<td>14 (53)</td>
<td>19 (52)</td>
<td>13 (36)</td>
<td>14 (47)</td>
<td>14 (48)</td>
</tr>
<tr>
<td>BEALE</td>
<td>2</td>
<td>10 (26)</td>
<td>10 (26)</td>
<td>9 (23)</td>
<td>20 (70)</td>
<td>9 (30)</td>
</tr>
<tr>
<td>EDGAR</td>
<td>2</td>
<td>6 (20)</td>
<td>6 (18)</td>
<td>7 (16)</td>
<td>7 (17)</td>
<td>5 (18)</td>
</tr>
<tr>
<td>RECIPE</td>
<td>3</td>
<td>5 (16)</td>
<td>5 (16)</td>
<td>7 (16)</td>
<td>6 (16)</td>
<td>6 (21)</td>
</tr>
<tr>
<td>POWLL3</td>
<td>3</td>
<td>14 (30)</td>
<td>10 (24)</td>
<td>11 (22)</td>
<td>10 (30)</td>
<td>11 (33)</td>
</tr>
<tr>
<td>BIGGS</td>
<td>3</td>
<td>14 (42)</td>
<td>14 (52)</td>
<td>12 (30)</td>
<td>15 (48)</td>
<td>10 (40)</td>
</tr>
<tr>
<td>HELICAL</td>
<td>3</td>
<td>32 (74)</td>
<td>32 (67)</td>
<td>20 (67)</td>
<td>19 (66)</td>
<td>19 (62)</td>
</tr>
<tr>
<td>POWELL</td>
<td>4</td>
<td>65 (170)</td>
<td>46 (122)</td>
<td>32 (72)</td>
<td>17 (73)</td>
<td>10 (57)</td>
</tr>
<tr>
<td>WOOD</td>
<td>4</td>
<td>26 (60)</td>
<td>29 (70)</td>
<td>22 (51)</td>
<td>38 (114)</td>
<td>22 (72)</td>
</tr>
<tr>
<td>CANTERLL</td>
<td>4</td>
<td>25 (148)</td>
<td>16 (115)</td>
<td>14 (61)</td>
<td>19 (93)</td>
<td>16 (96)</td>
</tr>
<tr>
<td>TOTAL NOI (NOF)</td>
<td></td>
<td>211 (639)</td>
<td>187 (562)</td>
<td>147 (394)</td>
<td>165 (574)</td>
<td>128 (477)</td>
</tr>
</tbody>
</table>

**PERFORMANCE OF THE NEW ALGORITHMS IN RELATION TO STANDARD HS-CG METHOD**

<table>
<thead>
<tr>
<th>NO1</th>
<th>HS</th>
<th>NEW1</th>
<th>BAYATI</th>
<th>BUCKLEY</th>
<th>NEW2</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOF</td>
<td>100</td>
<td>88.6</td>
<td>69.6</td>
<td>78.1</td>
<td>60.6</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>87.9</td>
<td>61.6</td>
<td>89.8</td>
<td>71.6</td>
</tr>
</tbody>
</table>

ALL the algorithms terminate when $\| g^* \| < 1 \times 10^{-5}$
<table>
<thead>
<tr>
<th>TEST FUNCTIONS</th>
<th>N</th>
<th>HS NO1 (NIF)</th>
<th>NEW1 NO1 (NOF)</th>
<th>BAYATI NO1 (NOF)</th>
<th>BUCKLEY NO1 (NOF)</th>
<th>NEW2 NO1 (NOF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIXON</td>
<td>10</td>
<td>22 (50)</td>
<td>30 (62)</td>
<td>15 (31)</td>
<td>17 (40)</td>
<td>16 (49)</td>
</tr>
<tr>
<td>SHALLOW</td>
<td>20</td>
<td>8 (19)</td>
<td>7 (19)</td>
<td>8 (18)</td>
<td>9 (28)</td>
<td>8 (27)</td>
</tr>
<tr>
<td>MIELE</td>
<td>20</td>
<td>54 (141)</td>
<td>38 (122)</td>
<td>32 (89)</td>
<td>25 (84)</td>
<td>29 (102)</td>
</tr>
<tr>
<td>EX-CANTRL</td>
<td>20</td>
<td>20 (132)</td>
<td>17 (120)</td>
<td>15 (74)</td>
<td>19 (93)</td>
<td>16 (96)</td>
</tr>
<tr>
<td>EX-POWELL</td>
<td>20</td>
<td>60 (162)</td>
<td>47 (129)</td>
<td>38 (79)</td>
<td>39 (119)</td>
<td>17 (60)</td>
</tr>
<tr>
<td>PN2</td>
<td>30</td>
<td>16 (40)</td>
<td>18 (146)</td>
<td>29 (64)</td>
<td>38 (109)</td>
<td>16 (50)</td>
</tr>
<tr>
<td>EX-MIELE</td>
<td>40</td>
<td>82 (197)</td>
<td>43 (141)</td>
<td>34 (94)</td>
<td>28 (90)</td>
<td>29 (102)</td>
</tr>
<tr>
<td>EX-CANTRL</td>
<td>40</td>
<td>20 (132)</td>
<td>17 (126)</td>
<td>15 (74)</td>
<td>19 (93)</td>
<td>16 (96)</td>
</tr>
<tr>
<td>EX-POWELL</td>
<td>40</td>
<td>85 (213)</td>
<td>47 (129)</td>
<td>41 (85)</td>
<td>66 (198)</td>
<td>18 (65)</td>
</tr>
<tr>
<td>FULL</td>
<td>40</td>
<td>50 (100)</td>
<td>39 (79)</td>
<td>41 (81)</td>
<td>53 (107)</td>
<td>40 (120)</td>
</tr>
<tr>
<td>TOTAL NO1 (NOF)</td>
<td></td>
<td>417 (1185)</td>
<td>303 (1073)</td>
<td>268 (689)</td>
<td>313 (961)</td>
<td>205 (767)</td>
</tr>
</tbody>
</table>

Performance of the new algorithms in relation to standard HS-CG method

<table>
<thead>
<tr>
<th></th>
<th>HS</th>
<th>NEW1</th>
<th>BAYATI</th>
<th>BOCKLEY</th>
<th>NEW2</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO1</td>
<td>100</td>
<td>72.6</td>
<td>64.2</td>
<td>75.1</td>
<td>49.1</td>
</tr>
<tr>
<td>NOF</td>
<td>100</td>
<td>90.5</td>
<td>58.1</td>
<td>81.1</td>
<td>64.7</td>
</tr>
</tbody>
</table>

All the algorithms terminate when $\| g^* \| < 1 \times 10^{-5}$
New Switching Algorithm for Combining

**TABLE (3)**

<table>
<thead>
<tr>
<th>TEST FUNCTIONS</th>
<th>N</th>
<th>HS NO1 (NIF)</th>
<th>NEW1 NO1 (NOF)</th>
<th>BAYATI NO1 (NOF)</th>
<th>BUCKLEY NO1 (NOF)</th>
<th>NEW2 NO1 (NOF)</th>
<th>BFGS NO1 (NOF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX-POWELL</td>
<td>60</td>
<td>125 (303)</td>
<td>47 (129)</td>
<td>45 (96)</td>
<td>72 (206)</td>
<td>21 (70)</td>
<td>71 (203)</td>
</tr>
<tr>
<td>FREUD</td>
<td>60</td>
<td>6 (18)</td>
<td>10 (25)</td>
<td>7 (28)</td>
<td>16 (51)</td>
<td>6 (22)</td>
<td>15 (49)</td>
</tr>
<tr>
<td>STRAIT</td>
<td>70</td>
<td>6 (20)</td>
<td>6 (18)</td>
<td>6 (15)</td>
<td>5 (20)</td>
<td>5 (18)</td>
<td>5 (20)</td>
</tr>
<tr>
<td>EX-POWELL</td>
<td>80</td>
<td>112 (303)</td>
<td>47 (129)</td>
<td>40 (83)</td>
<td>63 (169)</td>
<td>18 (63)</td>
<td>60 (165)</td>
</tr>
<tr>
<td>EX-CANTRLL</td>
<td>80</td>
<td>20 (132)</td>
<td>18 (137)</td>
<td>15 (74)</td>
<td>19 (93)</td>
<td>17 (110)</td>
<td>22 (95)</td>
</tr>
<tr>
<td>WOLFE</td>
<td>80</td>
<td>50 (99)</td>
<td>47 (95)</td>
<td>42 (83)</td>
<td>63 (127)</td>
<td>41 (123)</td>
<td>65 (130)</td>
</tr>
<tr>
<td>EX-RECIPE</td>
<td>90</td>
<td>12 (33)</td>
<td>6 (18)</td>
<td>7 (16)</td>
<td>6 (16)</td>
<td>6 (21)</td>
<td>6 (16)</td>
</tr>
<tr>
<td>PNI</td>
<td>90</td>
<td>8 (25)</td>
<td>10 (41)</td>
<td>8 (21)</td>
<td>10 (31)</td>
<td>8 (29)</td>
<td>10 (31)</td>
</tr>
<tr>
<td>EX-POWELL</td>
<td>100</td>
<td>105 (276)</td>
<td>48 (140)</td>
<td>44 (91)</td>
<td>62 (175)</td>
<td>18 (63)</td>
<td>60 (170)</td>
</tr>
<tr>
<td>EX-CUBIC</td>
<td>100</td>
<td>14 (40)</td>
<td>19 (52)</td>
<td>13 (36)</td>
<td>47 (118)</td>
<td>13 (43)</td>
<td>43 (115)</td>
</tr>
<tr>
<td>TOTAL NO1 (NOF)</td>
<td>458</td>
<td>258</td>
<td>227</td>
<td>363</td>
<td>153</td>
<td>966</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1249)</td>
<td>(784)</td>
<td>(543)</td>
<td>(1006)</td>
<td>(562)</td>
<td>(1010)</td>
</tr>
</tbody>
</table>

**PERFORMANCE OF THE NEW ALGORITHMS IN RELATION TO STANDARD HS-CG METHOD**

<table>
<thead>
<tr>
<th></th>
<th>HS</th>
<th>NEW1</th>
<th>BAYATI</th>
<th>BUCKLEY</th>
<th>NEW2</th>
<th>BFGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO1</td>
<td>100</td>
<td>56.3</td>
<td>49.5</td>
<td>79.2</td>
<td>33.4</td>
<td>78.4</td>
</tr>
<tr>
<td>NOF</td>
<td>100</td>
<td>62.7</td>
<td>43.4</td>
<td>80.5</td>
<td>44.9</td>
<td>80.9</td>
</tr>
</tbody>
</table>

ALL the algorithms terminate when \[ \| g^* \| < 1 \times 10^{-5} \]
TABLE (4)

A statistical comparison between the new algorithm (NEW2) with [HS; BAYATI; and NEW1 algorithm by using paired t-test.

<table>
<thead>
<tr>
<th>N</th>
<th>ALGORITHMS</th>
<th>NO1 / NOF</th>
<th>SIGNIFICANCE LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NO1</td>
<td>NOF</td>
</tr>
<tr>
<td></td>
<td>HS</td>
<td>0.112614</td>
<td>0.214132</td>
</tr>
<tr>
<td>2 - 4</td>
<td>NEW1</td>
<td>0.081546</td>
<td>0.243499</td>
</tr>
<tr>
<td></td>
<td>BAYATI</td>
<td>0.269737</td>
<td>0.0861184</td>
</tr>
<tr>
<td></td>
<td>BUCKLEY</td>
<td>0.0654013</td>
<td>0.116255</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NO1</td>
<td>NOF</td>
</tr>
<tr>
<td></td>
<td>HS</td>
<td>0.0233992</td>
<td>0.0428506</td>
</tr>
<tr>
<td>10 - 40</td>
<td>NEW1</td>
<td>0.0279254</td>
<td>0.0366562</td>
</tr>
<tr>
<td></td>
<td>BAYATI</td>
<td>0.0610346</td>
<td>0.241029</td>
</tr>
<tr>
<td></td>
<td>BUCKLEY</td>
<td>0.0619183</td>
<td>0.242282</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NO1</td>
<td>NOF</td>
</tr>
<tr>
<td></td>
<td>HS</td>
<td>0.0596574</td>
<td>0.9825033</td>
</tr>
<tr>
<td>60 - 100</td>
<td>NEW1</td>
<td>0.0262952</td>
<td>0.00710468</td>
</tr>
<tr>
<td></td>
<td>BAYATI</td>
<td>0.0729792</td>
<td>0.802641</td>
</tr>
<tr>
<td></td>
<td>BUCKLEY</td>
<td>0.011004</td>
<td>0.0362787</td>
</tr>
</tbody>
</table>

For the above Table one sided test is considered when the significance level is less than 0.05 and we have considered two sided tests otherwise, conventionally the levels are (see Robert and James 1980)

0.05 and less (significant)
0.01 and less (highly significant)
0.001 and less (very highly significant).
6. REFERENCES:


APPENDIX

All the presented test functions are from the general literature:

1- Cubic Function:
\[ f = 100(x_2 - x_1^3)^2 + (1-x_1)^2, \quad x_0 = (-1.2, 1)^T. \]

2- Strait Function:
\[ f = [(x_{2i-1} - x_2)^2 + 100(1 - x_{2i-1})^2 \]

3- Generalized Wood Function:
\[ f = \sum_{i=1}^{d} 100[(x_{2i-1} - x_{4i+1}^3)^2 + (1-x_{4i+1})^2 + (1-x_{4i})^2 + 10.1 [(x_{2i-1} - x_{4i+1})^2 + (x_{4i+1} - x_{4i})^2] + 19.8 (x_{4i+1} - x_{4i}) \]
\[ x_{2i-1} = (-3, -1, -3, -1; ...)^T. \]

4- Generalized Powell Function:
\[ f = \sum_{i=1}^{d} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i} + 2x_{4i-1})^2 + 10(x_{4i-3} - x_{4i})^4], \quad x_0 = (3, -1, 0, 1; ...)^T. \]

5- Beale Function:
\[ f = (1.5-x_1 + (1-x_1^2)^2 + (2.25-x_1)(1-x_1^2)^2 + (2.625- x_{1} (1-x_1^2)^2), \quad x_0 = (0, 0)^T. \]

6- Pen 1. (Penalty Function):
\[ f = \prod_{i=1}^{d} [1 + (100(x_i^2 - 0.25)^2)], \quad x_0 = (1, 2, ... ,n)^T. \]

7- Pen 2. (Generalized Penalty Function):
\[ f = \prod_{i=1}^{d} [\exp(10(x_i^2 - 0.25)^2) + (x_i^2 - 0.25)^2], \quad x_0 = (1, 2, ... ,n)^T. \]

8- Dixon Function:
\[ f = \sum_{i=1}^{d} [(1 - x_i)^2 + (1 - x_i)^2 + \sum_{i=1}^{d} (x_{2i}-x_{i+1})^2], \quad x_0 = (-1; ...)^T. \]

9- Generalized Edger & Himmel Function:
\[ f = \sum_{i=1}^{d} [(x_{4i+1} - 2x_{2i})^4 + (x_{2i}-2)^2 + x_{2i}^2 + (x_{2i}+1)^2], \quad x_0 = (1, 0; ...)^T. \]

10- Generalized Recip Function:
\[ f = \sum_{i=1}^{d} \left\{ \frac{(x_{3i+1} - 5)^2 + x_{3i+1}^2}{(x_{3i+1} - x_{3i+2})^2} \right\}, \quad x_0 = (2.5, 5, 1; ...)^T. \]

11- Wolfe Function:
\[ f = [-x_1 (3-x_1/2) + 2x_2 - 1]^2 + \sum_{i=1}^{d} (x_{4i+1} + x_{3i} (3-x_1/2) + 2x_{4i+1} - 1)^2 \]
\[ (x_{4i+1} + x_{3i} (3-x_1/2) - 1)^2], \quad x_0 = (-1, ...)^T. \]

12- Biggs Function:
\[ f = \prod_{i=2}^{d} \left\{ \exp(x_1x_2) - (x_3 \exp(x_2x_1) - \exp(-x_3) + 5 \exp(-10x_3))^2, \right\}, \quad x_0 = (1, 2, 1, ...)^T. \]

13- Generalized Freudenstien and Roth Function:
\[ f = \sum_{i=1}^{d} \left\{ (-3x_{2i-1} + (5-x_{2i})x_{2i-2})x_{2i}^2 + [29 + x_{2i-1} + (1 + x_{2i})x_{2i-1} - 14x_{2i}^2 \right\}, \quad x_0 = (30, 3; ...)^T. \]

14- Full Set of Distinct Eigen values Function:
\[ f = (x_1 - 1)^2 + \sum_{i=2}^{d} [2x_i - x_{i-1}]^2, \quad x_0 = (1; ...)^T. \]

15- Generalized Miele Function:
\[ f = \sum_{i=1}^{d} \left\{ \exp(5x_2) - (x_3 \exp(5x_2) - \exp(-5x_2) + 5 \exp(-10x_2))^2, \right\}, \quad x_0 = (1, 2, 2, 2; ...)^T. \]

16- Generalized Helical Valley Function:
\[ f = \sum_{i=1}^{d} \left\{ 100[(x_3i - 10)^2 + (r-1)^2] + x_3i \right\}, \quad x_0 = (1, 2, ... ,n-1)^T \]
\[ e = \begin{cases} \quad 1 \quad & \text{for } x_i > 0 \\ \frac{1}{2} + \frac{1}{2} \arctan (x_2 / x_1) & \text{for } x_i < 0 \end{cases} \]
\[ r = (x_1^2 + x_2^2)^{1/2} \quad x_0 = (-1, 0, 0; ...)^T. \]

17- Generalized Powell 3. (Powell Three Variable Function):
\[ f = \sum_{i=1}^{d} \left\{ \sum_{j=1}^{d} \left[ \frac{1}{2} \sin^2 \frac{\pi x_{2j} x_{3j}}{2} \right] \cdot \exp(-x_{2j} - x_{3j}) \right\}, \quad x_0 = (0, 1, 2, ...)^T. \]

18- Generalized Cantreal Function:
\[ f = \sum_{i=1}^{d} \left\{ \exp(x_{4i+1}x_{4i+2}) + 100(x_{4i+1} - x_{4i})^2 \right\}, \quad x_0 = (1, 2, 2; ...)^T. \]