A New CG-Algorithm with Self-Scaling VM-Update for Unconstraint Optimization

Abbas Y. Al-Bayati  
College of Basic Education Telafer  
University of Mosul, Mosul-Iraq  
profabbaslbayati@yahoo.com

Ivan S. Latif  
Department of Mathematics  
University of Salahaddin, Erbil-Iraq  
ivansubhi2001@yahoo.com

Received: February 09, 2011; Accepted: February 23, 2012

Abstract

In this paper, a new combined extended Conjugate-Gradient (CG) and Variable-Metric (VM) methods is proposed for solving unconstrained large-scale numerical optimization problems. The basic idea is to choose a combination of the current gradient and some pervious search directions as a new search direction updated by Al-Bayati's SCVM-method to fit a new step-size parameter using Armijo Inexact Line Searches (ILS). This method is based on the ILS and its numerical properties are discussed using different non-linear test functions with various dimensions. The global convergence property of the new algorithm is investigated under few weak conditions. Numerical experiments show that the new algorithm seems to converge faster and is superior to some other similar methods in many situations.

Keywords: Unconstrained Optimization, Gradient Related Method, Self-Scaling VM-Method, Inexact Line Searches

MSC 2010: 49M07, 49M10, 90C06, 65K
1. Introduction

Consider an unconstrained optimization problem:

$$\min f(x), \ x \in \mathbb{R}^n,$$

where \( f : \mathbb{R}^n \to \mathbb{R}^1 \) is a continuously differentiable function in \( \mathbb{R}^n \) an n-dimensional Euclidean space; \( n \) may be very large in some sense. Most of the well-known iterative algorithms for solving (1) take the form:

$$x_{k+1} = x_k + \alpha_k d_k,$$

where \( d_k \) is a search direction and \( \alpha_k \) is a positive step-size along the search direction. This class of methods is called a line search gradient method. If \( x_k \) is the current iterative point, then we denote \( \nabla f(x_k) \) by \( g_k \), \( f(x_k) \) by \( f_k \) and \( f(x^*) \) by \( f^* \), respectively. If we take \( d_k = -g_k \), then the corresponding method is called the Steepest Descent (SD) method; one of the simpler gradient methods. That has wide applications in large scale optimization; see Nocedal and Wright (1999). Generally the CG-method is a useful technique for solving large-scale nonlinear problems because it avoids the computation and storage of some matrices associated with the Hessian of objective functions. The CG-method has the form:

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + \beta_{k-1}d_{k-1}, & \text{if } k > 0, \end{cases}$$

where \( \beta_k \) is a parameter that determines the different CG-methods; see for example the following references: Crowder & Wolfe (1972); Dai & Yuan (1996, 1999) and Fletcher-Reeves (1964). Well known choices of \( \beta_k \) satisfy:

$$\rho_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \quad \rho_k^{PR} = \frac{g^T_k(g_k - g_{k-1})}{\|g_{k-1}\|^2}, \quad \rho_k^{HS} = \frac{g^T_k(g_k - g_{k-1})}{d^T_{k-1}g_{k-1}},$$

which respectively, correspond to the FR (Fletcher-Reeves, 1964), PR (Polak-Ribiere, 1969) and HS (Hestenes Stiefel, 1952). CG-method with Exact Line Search (ELS) has finite convergence when they are used to minimize strictly convex quadratic function; see for example Al-Bayati and Al-Assady (1986). However, if the objective function is not quadratic or ELS is not used then a CG-method has no finite convergence. Also a CG-method has no global convergence if the objective function is non-quadratic. Similarly, Miele and Cantrell (1969) studied the memory gradient method for (1); namely, if \( x_0 \) is an initial point and \( d_0 = g_0 \), the method can be stated as follows:

$$x_{k+1} = x_k + v_k, \quad v_k = -\alpha g_k + \beta v_{k-1},$$
where $\alpha$ and $\beta$ are scalars chosen at each step so as to yield the greatest decrease in the function $f$. Cantrell (1969) showed that the memory gradient method and the FR-CG method are identical in the particular case of a quadratic function. Cragg and Levy (1969) proposed a super-memory gradient method whose search direction is defined by:

$$d_k = -\alpha g_k + \sum_{i=1}^{k} \beta_i d_{i-1},$$

where $x_0$ is an initial point and $d_0 = g_0$. Wolfe and Viazminsky (1976) also investigated a super-memory descent method for (1) in which the objective takes the form:

$$f(x_k - \alpha_k p_k + \sum_{i=1}^{m} \beta_i^{(k)} v_{k-i}) = \min_{x, \beta_1, \ldots, \beta_{i-1}} f(x_k - \alpha_k p_k + \sum_{i=1}^{m} \beta_i^{(k)} v_{k-i}),$$

where

$$v_k = -\alpha_k p_k + \sum_{i=1}^{m} \beta_i^{(k)} \delta_{k-i},$$

$m$ is a fixed positive integer; with

$$p_k^T g_k \neq 0.$$

Both the memory and super-memory gradient methods are more efficient than the CG and SD methods by considering the amount of computation and storages required in the latter. Shi-Shen (2004) combined the CG-method and supper-memory descent method to form a new gradient method that may be more effective than the standard CG-method for solving large scale optimization problems as follows:

$$\min \{ g_k^T d_k (\beta_{k-m+1}, \ldots, 1 - \sum_{i=2}^{m} \beta_{k-i+1}) \left| \beta_{k-i+1} \in \left[ \frac{\delta_i}{2}, \delta_i^* \right] (i = 2, \ldots, m) \right\},$$

where

$$x_{k+1} = x_k + \alpha_k d_k (\beta_{k-m+1}, \ldots, \beta_{k+i}).$$

We denote $d_k (\beta_{k-m+1}, \ldots, \beta_{k+i})$ by $d_k$ throughout this paper, $m$ is a fixed positive integer and $\alpha_k$ is a scalar chosen by a line search procedure. The theoretical and practical merits of the Quasi Newton (QN) family of methods for unconstrained optimization have been systematically explored since the classic paper of Fletcher and Powell analyzed by Davidon’VM method. In 1970 the self-scaling VM algorithms were introduced, showing significant improvement in efficiency over earlier methods.
Recently, Al-Bayati and Latif (2008) proposed a new three terms preconditioned gradient memory method. Their method subsumes some other families of nonlinear preconditioned gradient memory methods as its subfamilies with Powell's restart criterion and inexact Armijo line searches. Their search direction was defined by:

\[ d_{k}^{b&l} = \begin{cases} -H_k g_k, & \text{if } k = 0 \\ -g_k + \beta H_k d_{k-1} - \alpha H_{k-1} d_{k-1}, & \text{if } k > 0, \end{cases} \]  

(10)

where \( \alpha \) is a step-size defined by inexact Armijo line search procedure and \( \beta \) is the conjugacy parameter. Al-Bayati et al. (2009) introduced two versions CG-algorithm. Their search directions are defined by:

\[ d_k^1 = \begin{cases} -g_k, & \text{if } k = 0 \\ -g_k + \beta_k^1 d_{k-1}, & \text{if } k > 0, \end{cases} \quad \text{and} \quad \beta_k^1 = (1 - \frac{s_k^T y_k}{y_k^T s_k}) \frac{(g_{k+1}^T y_k)}{y_k^T y_k}, \]

(11a)

\[ d_k^2 = \begin{cases} -g_k, & \text{if } k = 0 \\ -g_k + \beta_k^2 d_{k-1}, & \text{if } k > 0, \end{cases} \quad \text{and} \quad \beta_k^2 = (1 - \frac{s_k^T y_k}{y_k^T s_k}) \frac{(g_{k+1}^T y_k)}{y_k^T y_k} + \frac{s_k^T g_{k+1}}{d_k^T y_k}, \]

(11b)

where (11) has been proved to be a sufficiently descent directions.

Also, Zhang, et al. in (2009) had modified Dai-Liao DL-CG method with three terms search directions as follows:

\[ d_k = \begin{cases} -g_0, & \text{if } k = 0, \\ -g_k + \beta_k^DL d_{k-1} - \xi_k (y_{k-1} - t s_{k-1}), & \text{if } k > 0, \end{cases} \]

(12a)

where \( \xi_k = g_k^T d_{k-1} / d_{k-1}^T y_{k-1} \) and \( \beta_k^DL \) is defined by:

\[ \beta_k^DL = \frac{g_k^T (y_{k-1} - t s_{k-1})}{d_{k-1}^T y_{k-1}}, \quad t \geq 0. \]

(12b)

They show that the sufficient descent condition also holds true if no line search is used, that is,

\[ g_k^T d_k = -\|g_k\|^2. \]

(13)

In order to achieve the global convergence result, Grippo and Lucidi (1997) proposed the following new line search: for given constants \( \tau > 0, \delta > 0, \) and \( \lambda \in (0,1) \), let
\[ \alpha_k = \max \left\{ \lambda^j \frac{g_k^T d_k}{\|d_k\|^2}; j = 0, 1, \ldots \right\}, \]  

which satisfy

\[ f(x_k) \leq f(x_{k-1}) - \delta \alpha_k^2 \|d_{k-1}\|^2. \]  

This line search will be preferred to the classical Armijo one for the sake of a greater reduction of objective function. Introducing this line search rule. This may be taken as an open problem.

In this paper, a new gradient related algorithm combined with VM-update used for solving large scale unconstrained optimization problems, is proposed. The new algorithm is a kind of ILS method modified with VM-algorithm. The basic idea is to choose a combination of the current gradient and some previous search directions with Al-Bayati self-scaling VM-update which is based on two-parameter family of rank-two updating formulae. The algorithm is compared with similar published algorithms, which may be more effective than the standard conjugate related algorithm; namely, Nazareth (1977) and other VM-algorithm. The global rate of convergence is investigated under a diverse weak condition. Numerical experiment shows that the new algorithm seems to converge more stably and is superior to other similar methods.

2. Shi-Shen Algorithm

Shi-Shen (2004) proposed the following assumptions:

1. The objective function \( f \) has lower bound on the level set \( L_0 = \{ x \in \mathbb{R}^n | f(x) \leq f(x_0) \} \), where \( x_0 \) is an available initial point.

2. The gradient \( g(x) \) of \( f(x) \) is Lipschitz continuous in an open convex set \( B \) which contains \( L_0 \) i.e., there exist a constant \( L > 0 \) such that:

\[ \| g(x) - g(y) \| \leq L \| x - y \|, \quad \forall x, y \in B. \]

3. The gradient \( g(x) \) is uniformly continuous in an open convex set \( B \) containing \( L_0 \).

Obviously Assumption \( (S_2) \) implies \( (S_3) \).

As we know, a key to devise an algorithm for unconstrained optimization problems is to choose an available search direction \( d_k \) and a suitable step-size \( \alpha_k \) at each iteration. Certainly if we choose a search direction \( d_k \) satisfying:

\[ -g_k^T d_k < 0, \]
then we can devise a descent direction generally, we demand that:

$$\mathbf{- g}_k^T \mathbf{d}_k \geq \eta \| \mathbf{g}_k \|^2,$$  \hspace{1cm} (17a)

which is called sufficient descent condition where $\eta > 0$.

Furthermore, if

$$\mathbf{- g}_k^T \mathbf{d}_k \geq \eta \| \mathbf{g}_k \| \| \mathbf{d}_k \|,$$  \hspace{1cm} (17b)

then many descent algorithms have their convergence under the above condition. It is called an angle condition or a gradient–related conception.

**Definition 2.1. Berstekas (1982)**

Let $\{x_k\}$ be a sequence generated by the gradient method (2). We say that the sequence $\{d_k\}$ is uniformly gradient related to $\{x_k\}$ if for every convergent subsequence $\{x_k\}$ for which

$$\lim_{k \to \infty} g_k \neq 0,$$  \hspace{1cm} (18a)

we have

$$0 < \lim_{k \to \infty} \inf_{k \in K} \| g_k^T d_k \|, \lim_{k \to \infty} \sup_{k \in K} \| d_k \| < +\infty.$$  \hspace{1cm} (18b)

Equivalently, $\{d_k\}$ is uniformly gradient related if whenever a subsequence $\{g_k\}$ tends to a non-zero vector, the corresponding subsequence of direction $d_k$ is bounded and does not tend to be orthogonal to $g_k$. Moreover, we must choose a line search rule to find the step-size a long search direction at each iteration.

**Lemma 2.1. Berstekas (1982)**

Let $\{x_k\}$ be a sequence generated by a gradient method and assume that $\{d_k\}$ is uniformly gradient related and $\alpha_k$ is chosen by the minimization rule or the limited minimization rule, then every limited point of $\{x_k\}$ is a critical point $x^*$, i.e. $g(x^*) = 0$. As to the parameters in the algorithm, we seem to choose $\beta_{k-i+1}^{(k)} \in [\delta_k^i, \delta_k^i] (i = 2, \ldots, m)$ for solving large scale optimization problems as defined in (9). To get the algorithm to converge more quickly, we take:
\[ \beta_{k-1}^{(k)} = \begin{cases} \frac{\delta_k^i}{2}, & \text{if } \|g_k\|^2 \geq g_k^T d_{k-i+1}, \\ \delta_k^i, & \text{if } \|g_k\|^2 < g_k^T d_{k-i+1}. \end{cases} \quad \text{for } i = 1,2,\ldots,m. \quad (19) \]

Now, it is easy to prove that:

\[ 0 < \sum_{i=2}^{m} \beta_{k-1}^{(k)} \leq \sum_{i=2}^{m} \delta_k^i \leq \rho < 1 \]

and

\[ 1 > \beta_k^{(k)} = 1 - \sum_{i=2}^{m} \beta_{k-1}^{(k)} \geq 1 - \sum_{i=2}^{m} \delta_k^i \geq 1 - \rho > 0. \]

The details may be found in Crowder & Wolfe (1972).

**Algorithm 2.1. Shi-Shen**

Let \( 0 < \rho < 1 \), \( 0 < \mu_1 < \mu_2 < 1 \), a fixed integer \( m \geq 2 \), \( x_1 \in \mathbb{R}^n \), \( k = 1 \) and \( \varepsilon \) is a small parameter, then:

**Step 1.** If \( \|g_k\| < \varepsilon \), then stop.

**Step 2.** Set \( x_{k+1} = x_k + \alpha_k d_k (\beta_{k-m+1}^{(k)}, \ldots, \beta_k^{(k)}) \), where

\[ d_k (\beta_{k-m+1}^{(k)}, \ldots, \beta_k^{(k)}) = \begin{cases} -g_k, & \text{if } k = m-1, \\ -\beta_k^{(k)} g_k - \sum_{i=2}^{m} \beta_{k-i+1}^{(k)} d_{k-i+1}, & \text{if } k \geq m, \end{cases} \]

\[ \beta_{k-i+1}^{(k)} \in \left[ \frac{\delta_k^i}{2}, \delta_k^i \right], i = 2,\ldots,m, \]

\[ \beta_{k-1}^{(k)} = \begin{cases} \frac{\delta_k^i}{2}, & \text{if } \|g_k\|^2 \geq g_k^T d_{k-i+1}, \\ \delta_k^i, & \text{if } \|g_k\|^2 < g_k^T d_{k-i+1}, \end{cases} \quad \text{for } i = 1,2,\ldots,m, \]

and
\[
\delta^i_k = \frac{\rho}{m} \frac{\|g_k\|^2}{\|g_k\|^2 + \|g_k^T d_{k-i+1}\|^2}, \quad (i = 2, \ldots, m), \quad \beta^{(k)}_k = 1 - \sum_{i=2}^{m} \beta^{(k)}_{k-i+1},
\]

scalar \( \alpha_k \) in Step (2) is chosen by a cubic line search; Bunday (1984).

**Step 3.** If the available storage is exceeded, then employ a restart option [Zoutendijk (1970)] either with

\[ k = n \quad \text{or} \quad g_{k+1}^T g_{k+1} > g_k^T g_k. \]

**Step 4.** Set \( k = k + 1 \), go to Step 1.

### 3. A New Proposed Algorithm for Solving Problem (1)

In this section we want to choose a line search rule to find the best step-size parameter along the search direction at each iteration. In fact, we can use the generalized Armijo’ line search rule implemented in Luenberger (1989):

Set scalar \( S, \beta, \mu_i \) with \( \mu_i \in (0,1) \), \( \beta \in (0,1) \) and \( S > 0 \). Let \( \alpha_k \) be the largest \( \alpha \) in \( \{S, S\beta, S\beta^2, \ldots\} \) such that:

\[
f_k - f(x_k + \alpha d_k) \geq -\mu_i \alpha g_k^T d_k. \tag{20}
\]

Choosing the parameter \( \beta \) is important for the implementation of the line search method. If \( \beta \) is too large then the line search process may be too slow while if \( \beta \) is too small then the line search process may be too fast so as to lose the available step size we should choose a suitable step size at each iteration.

### 4. New Method

In order to increase the efficiency of Algorithm 2.1, an extended Armijo line search rule given in Cantrell (1969) is used to find the best value of the step-size in order to locate the new hybrid line search which combines the search direction of Shi-Shen Algorithm 2.1 with Al-Bayati (1991) self-scaling update and as shown below:

Let \( 0 < \rho < 1, \frac{1}{2} < \mu_i < 1, \) a fixed integer \( m \geq 2 \), \( x_i \in \mathbb{R}^n \), \( k = 1 \) and \( H_1 \) is any positive definite matrix usually \( H_1 = I \) and \( \varepsilon \) a small parameter.

**Algorithm 4.1.**

**Step 1.** If \( \|g_k\| = 0 \), then stop.
Step 2. Set $x_{k+1} = x_k + \alpha_k d_k(\beta^{(k)}_{k-m+1}, \ldots, \beta^{(k)}_k)$, where

$$
\begin{align*}
d_k(\beta^{(k)}_{k-m+1}, \ldots, \beta^{(k)}_k) &= \begin{cases} 
-Hg_k, & \text{if } k \leq m-1, \\
-H\{\beta^{(k)}_k g_k - \sum_{i=2}^m \beta^{(k)}_{k-i+1} d_{k-i+1}\}, & \text{if } k \geq m,
\end{cases}
\end{align*}
$$

$\beta^{(k)}_{k-i+1} \in [\delta^i_k, \delta^i_k]$,

$\beta^{(k)}_k \in \begin{cases} 
\frac{\delta^i_k}{2}, & \text{if } \|g_k\|^2 \geq g_k^T d_{k-i+1}, \\
\delta^i_k, & \text{if } \|g_k\|^2 < g_k^T d_{k-i+1},
\end{cases}$

and

$$
\delta^i_k = \frac{\rho}{m - \|g_k\|^2 + \|g_k^T d_{k-i+1}\|}, \quad (i = 2, \ldots, m), \quad \beta^{(k)}_k = 1 - \sum_{i=2}^m \beta^{(k)}_{k-i+1}.
$$

Scalar $\alpha_k$ in Step (2) is chosen by Armijo line search rule defined in (20) and $H_k$ is defined by Al-Bayati (1991) VM-update defined in Step (3).

Step 3. Update $H_k$ by

$$
H_{k+1} = \left( H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + w_k w_k^T \right) + \mu_k \left( v_k v_k^T / y_k^T y_k \right)
$$

with

$$
v_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k,
$$

$$
w_k = (y_k^T H_k y_k)^{\frac{1}{2}} \left[ v_k / v_k^T y_k - H_k y_k / y_k^T H_k y_k \right],
$$

$$
\mu_k = y_k^T H_k y_k / v_k^T y_k.
$$

Step 4: If available storage is exceeded, then employ a restart option either with $k = n$ or

$$
g_{k+1}^T g_{k+1} > g_{k+1}^T g_k.
$$

Step 5. Set $k = k + 1$ and go to Step 2.

Now to ensure that the new algorithm has a super-linear convergence, let us consider the following theorems:
Theorem 4.1.

If $(S_1)$ and $(S_2)$ hold and if the new Algorithm 4.1 generates an infinite sequence $\{x_k\}$, then

$$\sum_{k=m}^{\infty} \frac{\|g_k\|^4}{\gamma_k} < +\infty,$$

(21a)

where

$$\gamma_k = \max_{2\leq i \leq m}(\|g_k\|^2, \|d_{k-i+1}\|^2).$$

(21b)

Proof:

Since $\{f_k\}$ is a decreasing sequence and has a lower bound on the level set $L_0$, it is a convergent sequence. Moreover, since $H_{k+1}$ is also has a global rate of convergence, see Al-Bayati (1991) for the details; therefore Lemma 2.1 shows that (21) holds and hence the new algorithm has a super linear convergence and hence the proof is complete.

Theorem 4.2.

If conditions in Theorem 4.1 hold, then either $\lim_{k\to\infty} \|g_k\| = 0$ or $\{x_k\}$ has no bound.

Proof:

If $\lim_{k\to\infty} \|g_k\| \neq 0$, then there exists an infinite subset $K_0 \subseteq \{m, m+1, \ldots\}$ and $\varepsilon > 0$ such that:

$$\|g_k\| > \varepsilon, \quad k \in K_0$$

(22)

Thus

$$\frac{\varepsilon^4}{\gamma_k} \leq \frac{\|g_k\|^4}{\gamma_k}, \quad \forall \quad k \in K_0$$

(23)

By Theorem 4.1 and for $k \geq 1$, we obtain

$$\|d_k\|^2 \leq \max_{1 \leq i \leq k} \|g_i\|^2$$

(24)

Now if $k \leq m$, then the conclusion is obvious. However, if $k > m$, then by induction process we obtain the conclusion; we have:
\[
\sum_{k \in K, \gamma_k} \frac{E^4}{K} \leq \sum_{k=m}^{+\infty} \|g_k\|^4 < +\infty.
\]  

(25)

Then there exists at least one \( i_0 \) : \( 2 \leq i_0 \leq m \) such that:
\[
\lim_{k \in K, k \to +\infty} \|d_{k-i_0}\| = +\infty
\]

(26)

and hence \( \{x_k\} \) has no bound.

5. Numerical Results.

Comparative tests are performed with seventy eight well-known test functions (Twenty six with three different versions) which are specified in the Appendix. All the results shown in Table 1 are obtained with newly-programmed Fortran routines which employ double precision. The Comparative performances of the algorithms are in the usual way by considering both the total number of function evaluations (NOF) and the total number of iterations (NOI). In each case the convergence criterion is that the value of \( \|g_k\| < 1 \times 10^{-5} \). The cubic fitting technique, published in its original form by Bunday (1984) is used as the common linear search subprogram for Shi-Shen algorithm while Armijo line search procedure defined in (21) is used for our new proposed algorithm.

Each test function was solved using the following two algorithms:

(1) The original algorithm published by Shi-Shen call it (Shi-Shen, Algorithm 2.1).

(2) The new proposed algorithm call it (New algorithm, Algorithm 4.1).

The numerical results in Table 1 give the comparison between the New and the Shi-Shen algorithms for different dimensions of test functions, while Table 2 gives the total overall the tools. The details of the percentage of improvements of NOI and NOF was given in Table 3. The important thing is that the new algorithm needs less iteration, fewer evaluations of \( f(x) \) and \( g(x) \) than the standard Shi-Shen algorithm in many situations especially for large-scale unconstrained optimization problems, when the iterative process reaches the same precision. The new proposed algorithm uses less CPU time than Shi-Shen method even we have not mention it. However, we can see that Shi-Shen algorithm may fail in some cases while the new method always converges for the minimum points. Moreover, the new algorithm seems to be suitable to solve ill-conditioned problems and suitable to solve large-scale minimization problems.
<table>
<thead>
<tr>
<th>NO. OF TEST</th>
<th>TEST FUNCTION</th>
<th>SHI-SHEN (3.1)</th>
<th>NOI(40F)</th>
<th>NEW (4.1)</th>
<th>NOI(40F)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>n=12</td>
<td>36</td>
<td>360</td>
<td>1080</td>
</tr>
<tr>
<td>1</td>
<td>EX-Tridiagonal-1</td>
<td>m&lt;n</td>
<td>57</td>
<td>54</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>142</td>
<td>118</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>113</td>
<td>112</td>
<td>112</td>
</tr>
<tr>
<td>2</td>
<td>EX-Three exponential</td>
<td>m&lt;n</td>
<td>26</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>61</td>
<td>59</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>55</td>
<td>43</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>Matrix Rom</td>
<td>m&lt;n</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>74</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>EX-Freud &amp; Roth</td>
<td>m&lt;n</td>
<td>2743</td>
<td>2781</td>
<td>1371</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>13715</td>
<td>13715</td>
<td>13715</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>9440</td>
<td>718975</td>
<td>10528</td>
</tr>
<tr>
<td>5</td>
<td>GEN-Tridiagonal-1</td>
<td>m&lt;n</td>
<td>58</td>
<td>71</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>145</td>
<td>123</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>99</td>
<td>89</td>
<td>127</td>
</tr>
<tr>
<td>6</td>
<td>Diagonal4</td>
<td>m&lt;n</td>
<td>7</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>15</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>15</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>Dqdrtic</td>
<td>m&lt;n</td>
<td>1260</td>
<td>1371</td>
<td>925</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>3036</td>
<td>3807</td>
<td>2509</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>2521</td>
<td>1371</td>
<td>925</td>
</tr>
<tr>
<td>8</td>
<td>Denschnb</td>
<td>m&lt;n</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>28</td>
<td>31</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>NO. OF TEST</td>
<td>TEST FUNCTION</td>
<td>$n$</td>
<td>SHI-SHEN (3.1)</td>
<td>NOI(NOOF)</td>
<td>NEW (4.1)</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>-----</td>
<td>---------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>36</td>
<td>360</td>
<td>1080</td>
</tr>
<tr>
<td>9</td>
<td>GEN-Quartic GQ1</td>
<td>$m&lt;n$</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m=n$</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m&gt;n$</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>Diagonal 7</td>
<td>$m&lt;n$</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m=n$</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m&gt;n$</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>Full Hessian</td>
<td>$m&lt;n$</td>
<td>25</td>
<td>31</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m=n$</td>
<td>25</td>
<td>31</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m&gt;n$</td>
<td>25</td>
<td>31</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>GEN-Powell</td>
<td>$m&lt;n$</td>
<td>85</td>
<td>88</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m=n$</td>
<td>109</td>
<td>107</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m&gt;n$</td>
<td>101</td>
<td>107</td>
<td>115</td>
</tr>
<tr>
<td>13</td>
<td>GEN-Rosen</td>
<td>$m&lt;n$</td>
<td>48</td>
<td>37</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m=n$</td>
<td>830</td>
<td>864</td>
<td>954</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m&gt;n$</td>
<td>820</td>
<td>864</td>
<td>954</td>
</tr>
<tr>
<td>14</td>
<td>Non-Diagonal</td>
<td>$m&lt;n$</td>
<td>23</td>
<td>44</td>
<td>73</td>
</tr>
<tr>
<td>15</td>
<td>GEN-Wolf</td>
<td>$m&lt;n$</td>
<td>102</td>
<td>197</td>
<td>206</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m=n$</td>
<td>182</td>
<td>197</td>
<td>206</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m&gt;n$</td>
<td>182</td>
<td>197</td>
<td>206</td>
</tr>
<tr>
<td>16</td>
<td>GEN-Strait</td>
<td>$m&lt;n$</td>
<td>9</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m=n$</td>
<td>9</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$m&gt;n$</td>
<td>9</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>NO. OF TEST</td>
<td>TEST FUNCTION</td>
<td>SHI-SHEN (3.1)</td>
<td>NOI(NO.F)</td>
<td>NEW (4.1)</td>
<td>NOI(NO.F)</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>17</td>
<td>GEN-Recipe</td>
<td>(m&lt;n)</td>
<td>69</td>
<td>12</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>69</td>
<td>14</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>69</td>
<td>14</td>
<td>73</td>
</tr>
<tr>
<td>18</td>
<td>Non-diagonal (Shanno-78)</td>
<td>(m&lt;n)</td>
<td>30</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>30</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>30</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>EX-Tridigonal-2</td>
<td>(m&lt;n)</td>
<td>83</td>
<td>19</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>83</td>
<td>19</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>83</td>
<td>19</td>
<td>102</td>
</tr>
<tr>
<td>20</td>
<td>GEN-Beale</td>
<td>(m&lt;n)</td>
<td>656</td>
<td>22</td>
<td>586</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>503</td>
<td>25</td>
<td>459</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>433</td>
<td>22</td>
<td>457</td>
</tr>
<tr>
<td>21</td>
<td>EX-Block-Diagonal BD2</td>
<td>(m&lt;n)</td>
<td>16</td>
<td>75</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>16</td>
<td>75</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>16</td>
<td>75</td>
<td>18</td>
</tr>
<tr>
<td>22</td>
<td>Diagonal 7</td>
<td>(m&lt;n)</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>Cosine (cute)</td>
<td>(m&lt;n)</td>
<td>9</td>
<td>75</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>9</td>
<td>75</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>9</td>
<td>75</td>
<td>10</td>
</tr>
<tr>
<td>24</td>
<td>EX-Himmelblau</td>
<td>(m&lt;n)</td>
<td>3</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m=n)</td>
<td>3</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(m&gt;n)</td>
<td>3</td>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>
### Table 2. Comparison between the New (4.1) and Shi-Shen (3.1) algorithms using the total of tools for each test function.

<table>
<thead>
<tr>
<th>NO. OF TEST</th>
<th>TEST FUNCTION</th>
<th>SHI-SHEN (3.1)</th>
<th>NOI(NOF)</th>
<th>NEW (4.1)</th>
<th>NOI(NOF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n</td>
<td>12</td>
<td>36</td>
<td>360</td>
<td>1080</td>
</tr>
<tr>
<td>25</td>
<td>Raydan2</td>
<td>m&lt;n</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>Diagonal6</td>
<td>m&lt;n</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total of each function</th>
<th>n</th>
<th>SHI-SHEN</th>
<th>NOI(NOF)</th>
<th>NEW</th>
<th>NOI(NOF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12</td>
<td>36</td>
<td>360</td>
<td>1080</td>
<td>4320</td>
</tr>
<tr>
<td>m&lt;n</td>
<td>9243</td>
<td>3083</td>
<td>4327</td>
<td>4636</td>
<td>5699</td>
</tr>
<tr>
<td></td>
<td>20286</td>
<td>8480</td>
<td>11616</td>
<td>13045</td>
<td>16365</td>
</tr>
<tr>
<td>m=n</td>
<td>4289</td>
<td>12825</td>
<td>14113</td>
<td>14885</td>
<td>16354</td>
</tr>
<tr>
<td></td>
<td>9493</td>
<td>26275</td>
<td>28576</td>
<td>30095</td>
<td>34206</td>
</tr>
<tr>
<td>m&gt;n</td>
<td>14238</td>
<td>13550</td>
<td>14141</td>
<td>14883</td>
<td>16354</td>
</tr>
<tr>
<td></td>
<td>725153</td>
<td>27428</td>
<td>28595</td>
<td>30081</td>
<td>33936</td>
</tr>
<tr>
<td>Total of 26x3=78 test functions</td>
<td>27770</td>
<td>29458</td>
<td>32581</td>
<td>34404</td>
<td>38407</td>
</tr>
<tr>
<td></td>
<td>754932</td>
<td>62183</td>
<td>68787</td>
<td>73221</td>
<td>84507</td>
</tr>
</tbody>
</table>
Table 3. Percentage performance of the new (4.1) proposed algorithm against Shi-Shen (3.1) algorithm for 100% in both NOI and NOF.

<table>
<thead>
<tr>
<th>n</th>
<th>Cost</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>m&lt;n</td>
</tr>
<tr>
<td>12</td>
<td>NOI</td>
<td>95.5</td>
</tr>
<tr>
<td></td>
<td>NOF</td>
<td>89.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m=n</td>
</tr>
<tr>
<td></td>
<td>NOI</td>
<td>96.96</td>
</tr>
<tr>
<td></td>
<td>NOF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
</tr>
<tr>
<td></td>
<td>NOI</td>
<td>95.3</td>
</tr>
<tr>
<td></td>
<td>NOF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>36</td>
<td>NOI</td>
<td>86.18</td>
</tr>
<tr>
<td></td>
<td>NOF</td>
<td>96.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
</tr>
<tr>
<td></td>
<td>NOI</td>
<td>96.88</td>
</tr>
<tr>
<td></td>
<td>NOF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>360</td>
<td>NOI</td>
<td>89.67</td>
</tr>
<tr>
<td></td>
<td>NOF</td>
<td>96.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m&gt;n</td>
</tr>
<tr>
<td></td>
<td>NOI</td>
<td>95.88</td>
</tr>
</tbody>
</table>

5. Conclusions and Discussions.

In this Paper, a new combined gradient related and VM-algorithm for solving large-scale unconstrained optimization problems is proposed. The new algorithm is a kind of Armijo line search method. The basic idea is to choose a combination of the current gradient and some previous search directions which are updated by Al-Bayati’s self scaling (1991) VM as a new search direction and to find a step-size by using Armijo ILS. Using more information at the current iterative step may improve the performance of the algorithm and accelerate the gradient relates which need a few iterations. The new algorithm concept is useful to analyze its global convergence property. Numerical experiments show that the new algorithm converges faster and is more efficient than the standard Shi-Shen algorithm in many situations. The new algorithm is expected to solve ill-conditioned problems. Clearly there are large ranges of the improving percentages against the standard Shi-Shen algorithm; namely, the new algorithm has about (53)% NOI and (75)%NOF improvements against Shi-Shen algorithm taking n=12. These
improvements are very clear for \( n=36; \ n = 360, 1080 \) and finally, for \( n = 4320 \) the new algorithm has about (64)% NOI and (80)% NOF improvements against Shi-Shen (2004) CG-algorithm.

REFERENCES


**APPENDIX**

All the test functions used in this paper are from general literature Nocedal (1980 , 2005).

1. **Extended Tridigonal-1 Function:**

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

\[ x_0 = [2, 2, ..., 2]. \]

2. **Extended Three Exponential Terms Function:**

\[ f(x) = \sum_{i=1}^{n/2} (\exp(x_{2i-1} + 3x_{2i} - 0.1) + \exp(x_{2i-1} - 3x_{2i} - 0.1) + \exp(-x_{2i-1} - 0.1)), \]

\[ x_0 = [0.1, 0.1, ..., 0.1]. \]

3. **Diagonal 5 Function (Matrix Rom):**

\[ f(x) = \sum_{i=1}^{n} \log(\exp(x_i) + \exp(-x_i)), \]

\[ x_0 = [1.1, ..., 1.1]. \]

4. **Extended Freud & Roth Function:**
5. Generalized Tridiagonal-1 Function:

\[
f(x) = \sum_{i=1}^{n/2} \left( -13 + x_{2i-1} + ((5 - x_{2i})x_{2i} - 2)x_{2i} \right)^2 + \left( -29 + x_{2i-1} + ((x_{2i} + 1)x_{2i} - 14)x_{2i} \right)^2,
\]
\[
x_0 = [0.5,-2.,0.5,-2.,...,0.5,-2.].
\]

6. Diagonal 4 Function:

\[
f(x) = \sum_{i=1}^{n/2} \left( x_{2i-1} + x_{2i} - 3 \right)^2 + (x_{2i-1} - x_{2i} + 1)^4,
\]
\[
x_0 = [2,2,...,2].
\]

7. Dqudtric Function (CUTE):

\[
f(x) = \sum_{i=1}^{n/2} \left( x_i^2 + cx_{i+1}^2 + dx_{i+2}^2 \right),
\]
\[
x_0 = [3,3,...,3], c = 100, d = 100.
\]

8. Extended Denschnb Function (CUTE):

\[
f(x) = \sum_{i=1}^{n/2} \left( x_{2i-1} - 2 \right)^2 + (x_{2i-1} - 2)^2 x_{2i}^2 + (x_{2i} + 1)^2
\]
\[
x_0 = [0.1,0.1,...,0.1].
\]

9. Generalized quartic Function GQ1

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

10. Diagonal 8 Function:
\[ f(x) = \sum_{i=1}^{n} x_i \exp(x_i) - 2x_i - x_i^2, \]
\[ x_0 = [1., 1., ..., 1., 1.]. \]

11. Full Hessian Function:
\[ f(x) = \left( \sum_{i=1}^{n} x_i \right)^2 + \sum_{i=1}^{n} \left( x_i \exp(x_i) - 2x_i - x_i^2 \right), \]
\[ x_0 = [1., 1., ..., 1.]. \]

12. Generalized Powell function:
\[ f(x) = \sum_{i=1}^{n/3} \left\{ 3 - \left[ \frac{1}{1+(x_i-x_2i)^2} \right] - \sin\left( \frac{\pi x_2i x_{3i}}{2} \right) - \exp\left[ -\left( \frac{x_i + x_3i}{x_2i} - 2 \right)^2 \right] \right\}, \]
\[ x_0 = [0., 1., 2., ..., 0., 1., 2.]. \]

13. Generalized Rosen Brock Banana function:
\[ f(x) = \sum_{i=1}^{n/2} 100(x_{2i} - x_{2i-1})^2 + (1 - x_{2i-1})^2, \]
\[ x_0 = [-1.2, 1., ..., -1.2, 1]. \]

14. Generalized Non-diagonal function:
\[ f(x) = \sum_{i=2}^{n} [100(x_1 - x_i)^2 + (1 - x_i)^2, \]
\[ x_0 = [-1., ..., -1.]. \]

15. Generalized Wolfe Function:
\[ f(x) = (-x_1(3-x_1/2) + 2x_2 - 1)^2 + \sum_{i=1}^{n-1} (x_{i-1} - x_i(3-x_i/2 + 2x_{i+1}-1))^2 + (x_{n-1} - x_n(3-x_n/2)-1)^2, \]
\[ x_0 = [-1., ..., -1.]. \]
16. Generalized Strait Function:

\[ f(x) = \sum_{i=1}^{n/2} (x_{2i-1}^2 - x_{2i})^2 + 100(1 - x_{2i-1})^2 , \]

\[ x_0 = [-2,\ldots,-2]. \]

17. Generalized Recipe Function:

\[ f(x) = \sum_{i=1}^{n/3} \left( (x_{3i-1} - 5)^2 + x_{3i-1}^2 + \frac{x_{3i-1}^3}{(x_{3i-1} - x_{3i-2})^2} \right) , \]

\[ x_0 = [2,5,1,...,2,5,1]. \]

18. Non-diagonal (Shanno-78) Function (Cute):

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

\[ x_0 = [-1,-1,...,-1]. \]

19. Extended Tridiagonal-2 Function:

\[ f(x) = \sum_{i=1}^{n-1} (x_i x_{i+1} - 1)^2 + c(x_i + 1)(x_{i+1} + 1) , \]

\[ x_0 = [1,1,...,1] , c = 0.1. \]

20. Generalized Beale Function:

\[ f(x) = \sum_{i=1}^{n/2} \left[ 1.5 - x_{2i} + (1 - x_{2i})^3 \right] + \left[ 2.25 - x_{2i+1} (1 - x_{2i})^2 \right] + \left[ 2.625 - x_{2i+1} (1 - x_{2i})^2 \right] , \]

\[ x_0 = [-1,-1,...,-1,-1]. \]

21. Extended Block-Diagonal BD2 Function:

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

\[ x_0 = [1.5,2,...,1.5,2]. \]
22. Diagonal 7 Function:

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

\[ x_0 = [1.,1.,...,1.,1.]. \]

23. Cosine Function (CUTE):

\[ f(x) = \sum_{i=1}^{n} \cos(-0.5x_{i+1} + x_i^2), \]

\[ x_0 = [1.,1.,...,1.,1.]. \]

24. Extended Himmelblau Function:

\[ f(x) = \sum_{i=1}^{n/2} \left( x_{2i-1}^2 + x_{2i} - 11 \right)^2 + \left( x_{2i-1} + x_{2i}^2 - 7 \right)^2, \]

\[ x_0 = [1.1,1.1,...,1.1,1.1]. \]

25. Raydan 2 Function:

\[ f(x) = \sum_{i=1}^{n} (\exp(x_i) - x_i), \]

\[ x_0 = [1.,1.,...,1.,1.]. \]

26. Diagonal 6 Function:

\[ f(x) = \sum_{i=1}^{n} (\exp(x_i) - (1 + x_i)), \]

\[ x_0 = [1.,1.,...,1.,1.]. \]