Qatar Univ. Sci. J. (1996), 16(2): 183-192

INVESTIGATION OF SINGLE-UPDATE MEMORYLESS CG-METHODS

ABBAS Y. AL-BAYATI* and HUDA ISSAM AHMED**
*Dept. of Mathematics, College of Science, University of Mosul, Iraq.
**Dept. of Statistic, College of Administration & Economics, University of Mosul, Iraq

دراسة في الخوارزميات المستغنيات عن الذاكرة للمشتقات المترافقة ذات التحديث الفردي

الخلاصة

في هذا البحث تم أشتقاق خوارزميتين لدوال لاخطية في الأمثليه غير المقيدة. الخوارزميات الجديدة تستعمل خطوط بحث غير تامة وقد قورنت عددياً مع خوارزميات مشهورة لـ PERRY, SHANNO أثبتت النتائج الحسابية بأن الخوارزميتين الجديدتين تتفوق حسابياً على نظيراتها من الخوارزميات المشهورة في هذا الحقل.

ABSTRACT

Two new CG-methods are proposed in this work for nonlinear unconstrained optimization problems which are considered as memoryless Variable Metric (VM) methods. They are derived for the inexact line searches and evaluated numerically against Shanno's two memoryless VM-methods. The results indicate that, the respective new methods have practical improvements on the Shanno's methods. Moreover, they are clearly more efficient than the Perry's extended CG-method.

INTRODUCTION

Conjugate Gradient (CG) algorithms are iterative techniques which generate a sequence of approximations to the minimizer x* of a scalar function f(x) of a vector variable x. The sequence x_k is defined by:-

$$x_{k+1} = x_k + \lambda_k d_k \qquad (1\text{-}1)$$

$$d_{k+1} = - g_{k+1} + \beta_k d_k \qquad (1\text{-}2)$$

where g_k is the gradient of f(x), λ_k is a positive scalar chosen to minimize f(x) along the search direction d_k, and β_k is a scalar, defined by one of the following expressions:-

$$\beta_k = \frac{y_k^T g_{k+1}}{g_k^T g_k} \qquad (1\text{-}3a)$$

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \qquad (1\text{-}3b)$$

$$\beta_k = \frac{y_k^T g_{k+1}}{g_k^T g_k} \qquad (1\text{-}3c)$$

$$\beta_k = \frac{- g_{k+1}^T g_{k+1}}{g_k^T d_k} \qquad (1\text{-}3d)$$

$$\beta_k = \frac{-g_{k+1}^T g_{k+1}}{d_{k+1}^T g_k} \qquad (1\text{-}3c)$$

where $y_k = g_{k+1} - g_k$, and the definition of β_k in (1-3a) is that due to Hestenes and Stiefel [1]; β_k in (1-3b) is that due to Fletcher and Reeves, [2]; β_k in (1-3c) is that due to Polak and Ribere, [3]; β_k in (1-3d) is that due to Dixon, [4]; and β_k in (1-3e) is that due to Al-Bayati and Al-Assady [5].

Properties of CG-methods

There are a number of common properties for all the CG-methods:

1. $d_i^T G d_j = 0$, $j<i$ (The conjugacy condition)
2. $g_i^T g_j = 0$, $j<i$ (The orthogonality condition)
3. $d_i^T g_i = - g_i^T g_i = - \| g_i \|^2$ (The descent condition).
4. They have the quadratic termination property.
5. They require o(n) multiplication per iteration for convergence to the minimum solution
   (i.e., they are global convergence algorithms).

where G is the Hessian matrix.

Rate of convergence

It is useful to examine the rate of convergence for each method. Rate of convergence can be expressed in various ways, but common classification is as follows, see for example Edgar and Himmelelbau [6]

(1) Linear convergence:

$$\frac{\| x_{k+1} - x^* \|}{\| x_k - x^* \|} \le C, \quad 0 \le C \le 1 \qquad (1\text{-}2\text{-}1)$$

(usually slow in practice)

(2) Order P convergence:

$$\frac{\| x_{k+1} - x^* \|}{\| x_k - x^* \|^p} \le C, \quad C \ge 0, P \ge 1 \qquad (1\text{-}2\text{-}2)$$

(faster in practice)

If P=2, the order of convergence is said to be quadratic.

(3) Superlinear convergence:

$$\lim_{k \to \infty} \frac{\| x_{k+1} - x^* \|}{\| x_k - x^* \|} \to 0 \qquad \text{as } k \to \infty \qquad (1\text{-}2\text{-}3)$$

(usually fast in practice)

Fletcher and Reeves CG-Algorithm (FRNAG)

Among the most efficient CG-algorithm is the FR-CG algorithm which was coded as a NAG routine in different versions. Among these different versions, the routine of Harwell library in 1972. This routine was written in Fortran language and it needs another subroutine, namely FMO2AD to compute the inner product of any two vectors.

We have used the FR-CG routine to confirm our numerical computations in this research. The routine is coded in MIZE, [7], and it restarts every n+1 iterations. For more detail of restarting techniques see Poewell [8].

CG-methods as memoryless quasi Newton methods

In this section we will present some established and new memoryless CG-algorithms.

The memoryless quasi-Newton methods

This type of CG-method was suggested, for the first time, by Perry [9] and was further analyzed by Shanno [10]. These algorithms generate descent directions even if inexact line searches are used since:-

$$d_k = - H_k g_k \qquad (2\text{-}1\text{-}1)$$

Multiplying eq. (2-1-1) by g_k^T, yields

$$d_k^T g_k = - g_k^T H_k g_k < 0 \qquad (2\text{-}1\text{-}2)$$

Since H_k is positive definite and $g_k^T H_k g_k > 0$

d_k is a descent direction.

Also this type of algorithm does not need to update the matrix H explicitly (i.e., This reduces to a vector of order n).

Perry's memoryless CG-method:

Perry [9] developed a memoryless CG-algorithm as follows: he noted that in eq(1-2) the scalar β_k was chosen to make d_k and d_{k+1} conjugate using an exact line search (ELS). Since, in general line search are not exact, Perry relaxed this requirement and he rewrote eq(1-2) where β_k is defined by (1-3a) in an equivalent form, but assuming inexact line search (ILS); thus he obtained

$$d_{k+1} = - [I - \frac{d_k y_k^T}{y_k^T d_k}] g_{k+1} \qquad (2\text{-}2\text{-}1)$$

But the projection matrix multiplying is not of full rank; hence he modified eq(2-2-1) as:

$$d_{k+1} = -[I - \frac{v_k y_k^T}{v_k^T y_k} + \frac{v_k v_k^T}{v_k^T y_k}] g_{k+1} \quad (2\text{-}2\text{-}2a)$$

$$= -Q_{k+1}\, g_{k+1} \quad (2\text{-}2\text{-}2b)$$

Perry gave other reasons to support his new choice: First, the matrix Q_{k+1} satisfies the form

$$Q_{k+1}^T \; y_k = v_k,$$

which is similar but not identical to the quasi-Newton (QN) condition. Second, eq(2-2-2) reduces to eq(2-2-1) if an (ELS) is carried out at this iteration since then $v_k^T g_{k+1} = 0$

Perry's limited experiments with his algorithm (six test functions with ≤ 4) showed that it performs only slightly better than the standard CG-method. But we have used this algorithm with (15 test function, for ≤ 2000) and we show that it performs better than the standard CG-method.

Algorithm (Perry):

We list below the outlines of Perry's algorithm.

For an initial point x

Step (1): set k = 1, $d_k = -g_k / \| g_k \|$

Step (2): set $x_{k+1} = x_k + \lambda_k d_k$, where λ_k is a scalar chosen in such a way that $f_{k+1} < f_k$ (ILS)

Step (3): check for convergence, $\| g_{k+1} \| < \epsilon$, where ϵ is small positive tolerance, stop.

Step (4): otherwise, if k = n

or $| g_{k+1}^T g_k | \geq 0.2 \; | g_{k+1}^T g_k |$

Compute the new search direction defined by

$d_{k+1} = -g_{k+1} (\lambda_k d_k^T d_k \; / \; g_{k+1}^T g_{k+1})$; set k = 1, and go to step 2. Else, set k = k + 1.

Step (5): compute the new search direction defined by

$$d_{k+1} = -g_{k+1} - (\frac{v_k^T g_{k+1}}{v_k^T y_k} - \frac{y_k^T g_{k+1}}{v_k^T y_k}) v_k$$

And go to step (2).

Shanno's single update memoryless CG-method

Shanno [10] addressed the issue that eq(2-2-2a) does not satisfy the actual QN-condition, which requires an update of the approximation to the inverse Hessian matrix H_{k+1} which satisfy $H_{k+1}^T y_k = v_k$.

He also pointed out that the matrix Q_{k+} is not necessarily symmetric positive definite. So that eq(2-2-2a) may not define downhill direction. Hence, he symmetrized Q_{k+1} by adding an appropriate term. Specifically, Shanno proposed:

$$Q_{k+1}^+ = [-\frac{v_k y_k^T}{v_k^T y_k} - \frac{y_k v_k^T}{v_k^T y_k} + \frac{v_k v_k^T}{v_k^T y_k}] \quad (2\text{-}3\text{-}2)$$

But this new symmetrix matrix does not satisfy neither (2-2-3) nor (2-3-1), so he again modified it in order to make it to do so. He then obtained:

$$Q_{k+1}' = I - [\frac{v_k y_k^T + y_k v_k^T}{v_k^T y_k}] + [1 + \frac{y_K^T y_k}{v_k^T y_k}] \frac{v_k v_k^T}{v_k^T y_k} \quad (2\text{-}3\text{-}3)$$

This new form of the projection matrix Q_{k+1} has a special relationship with the BFGS update formula. Indeed the actual BFGS formula is given by:

$$H_{k+1} = H_k - [\frac{H_k y_k^T v_k^T + v_k y_k^T H_k}{v_k^T y_k} + (1 + \frac{y_k^T H_k y_k}{v_k^T y_k}) \frac{v_k v_k^T}{v_K^T y_k}] \quad (2\text{-}3\text{-}4)$$

It is then easily seen that eq(2-3-3) is equivalent to eq(2-3-4) with H_k replaced by I.

The CG-method, which is referred to as a memoryless BFGS method is defined by

$$d_{k+1} = -Q_{k+1}^I \, g_{k+1} \quad (2\text{-}3\text{-}5)$$

then

$$d_{k+1} = -g_{k+1} - [(1 + \frac{y_k^T y_k}{v_k^T y_k}) \frac{v_k^T g_{k+1}}{v_k^T y_k} - \frac{y_k^T g_{k+1}}{v_k^T y_k}] v_k + \frac{v_k^T g_{k+1}}{v_k^T y_k} y_k \quad (2\text{-}3\text{-}6)$$

Properties of Shanno's method

First, for Shanno's method it is necessary to ensure that:

$$v_k^T y_k > 0, \text{ for } k > 1, \quad (2\text{-}3\text{-}1\text{-}1)$$

in order to maintain positive definite updating and hence that a downhill direction d_{k+1} is obtained. To explain that for the quadratic function, using the relation $H_{k+1}\, y_k = v_k$, since $(H_{k+1} = G^{-1})$ hence $y_k = G_k v_k$, substituting in eq(2-3-1-1) we get $v_k^T\, Gv_k > 0$ since G is positive definite and hence $v_k^T\, y_k > 0$. Second, eq(2-3-6) reduces again to eq(2-2-1) assuming ELS. Moreover, it does not require storage of the matrix Q_{k+1} in eq(2-3-3). Thus, no additional information is needed to compute d_{k+1} beyond the standard CG-method.

Self-scaling memoryless CG-methods

The idea of the self-scaling CG-method was originally developed in a series of papers by Oren, [11], Luenberger, [12]; Oren and Spedicato, [13]; Al-Bayati [14] and finally Al-Bayati and Al-Salih. [15] for any unconstrained nonlinear function f(x).

Shanno's self-scaling memoryless CG-method (MSHN)

Oren, [11] established the self-scaling BFGS algorithm for which H_{k+1} was given by:

$$H_{k+1} = [H_k - \frac{H_k\, y_k\, y_k^T\, H_k}{y_k^T\, H_k\, y_k} + w_k\, w_k^T]\,\eta_K + \frac{v_K\, v_k^T}{v_k^T\, y_k} \qquad (3\text{-}1\text{-}1)$$

where

$$w_k = (y_k^T\, H_k\, y_k)^{0.5}\,[\frac{v_k}{v_k^T\, y_k} - \frac{H_k\, y_k}{y_k^T\, H_k\, y_k} \qquad (3\text{-}1\text{-}2)$$

η_k is the scaling factor defined by:-

$$\eta_k = \frac{v_k^T\, y_k}{y_k^T\, H_k\, y_k} \qquad (3\text{-}1\text{-}3)$$

However, it seems natural to scale the CG-method on every iteration by using eq(2-3-6): this amounts to substituting I for H_k in eq(3-1-1); thus Shanno defined a modified CG-method with H_k replaced by I and

$$d_{k+1} = -\frac{v_k^T\, y_k}{y_k^T\, y_k} g_{k+1} - [\frac{2v_k^T\, g_{k+1}}{v_k^T\, y_k} - \frac{y_k^T\, g_{k+1}}{y_k^T\, y_k}]\, v_k + \frac{v_k^T\, g_{k+1}}{y_k^T\, y_k}\, y_k \qquad (3\text{-}1\text{-}4)$$

Unfortunately, Shanno found that this modified CG-method did not produce as good results as the one just defined by eq(2-3-6).

Modified Shanno's algorithm (MSHN)

We list below the lines of modified Shanno algorithm for an initial point x_1

Step (1): set k = 1, $d_k = -g_k / \| g_k \|$

Step (2): set $x_{k+1} = x_k + \lambda_k d_k$, where λ is a scalar chosen in such a way that $f_{k+1} < f_k$ (ILS)

Step (3): check for convergence, $\| g_{k+1} \| < \epsilon$, where ϵ is small positive tolerance, stop.

Step (4): otherwise, if k = n

or $| g_k^T\, g_k | \geq 0.2\, | g_{k+1}^T\, g_{k+1} |$

compute the new search direction defined by

$$d_{k+1} = -g_{k+1}\,(\lambda_k\, d_k^T\, d_k \;/\; g_{k+1}^T\, g_{k+1});$$

set k = 1, and go to step (2). Else, set k = k + 1.

Step (5): compute the new search direction defined by

$$d_{k+1} = -\frac{v_k^T\, y_k}{y_k^T\, y_k} g_{k+1} - [\frac{2v_k^T\, g_{k+1}}{v_k^T\, y_k} - \frac{y_k^T\, g_{k+1}}{y_k^T\, y_k}]\, v_k + \frac{v_k^T\, g_{k+1}}{y_k^T\, y_k}\, y_k$$

and go to step (2).

New memoryless CG-methods

In this research we have investigated two new memoryless CG-methods which employ single VM-updates; namely NEW1 and NEW2 memoryless CG-methods.

The first new proposed algorithm (NEW1)

Al-Bayati, [14] investigated another family of VM-updates for which the updating matrix H_k was defined by

$$H_{k+1} = H_k + [\frac{2y_k^T\, H_k\, y_k}{v_k^T\, y_k}]\, v_k\, v_k^T - \frac{H_k\, y_k\, v_k^T + v_k\, y_{k_k}^T H_k}{v_k^T\, y_k} \qquad (3\text{-}2\text{-}1\text{-}1)$$

The above updating formula generates positive definite matrices, (see Al-Bayati, [14]).

Now since

$$d_{k+1} = - H_{k+1}\, g_{k+1} \qquad (3\text{-}2\text{-}1\text{-}2)$$

Hence

$$d_{k+1} = - g_{k+1} - [\frac{2y_k^T\, y_k}{v_k^T\, y_k} - \frac{v_k^T\, g_{k+1}}{v_k^T\, y_k} - \frac{y_k^T\, g_{k+1}}{v_k^T\, y_k}]\, v_k + \frac{v_k^T\, g_{k+1}}{v_k^T\, y_k}\, y_k \qquad (3\text{-}2\text{-}1\text{-}3)$$

We also note that if $v_k^T\, g_{k+1} = 0$ (using ELS) then eq(3-2-1-3) reduces to:

$$d_{k+1} = - g_{k+1} + (\frac{y_k^T\, g_{k+1}}{v_k^T\, y_k})\, v_k,$$

which is the standard Hestense and Stiefel CG-method and therefore has n-step convergence to the minimum defined precisely by the new VM-update (3-2-1-1), where the approximation to the inverse Hessian reset to the identity matrix at every step.

Algorithm (NEW1)

We list below the outlines of the first proposed algorithm (NEW1). For an initial point x_1

Step (1): set k = 1, $d_k = - = - g_k / \| g_k \|$

Step (2): set $x_{k+1} = x_k + \lambda_k d_k$, where λ_k is a scalar chosen in such a way that $f_{k+1} < f_k$ (ILS)

Step (3): check for convergence, $\| g_{k+1} \| < \epsilon$, where ϵ is small positive tolerance, stop.

Step (4): otherwise, if k = n

or $\left| g_k^T\, g_k \right| \geq 0.2 \left| g_{k+1}^T\, g_{k+1} \right|$

compute the new search direction defined by

$$d_{k+1} = - g_{k+1}\, (\lambda_k\, d_k^T\, d_k\, / g_{k+1}^T\, g_{k+1});$$

set k = 1, and go to step (2). Else, set k = k + 1.

Step (5): compute the new search direction defined by

$$d_{k+1} = - g_{k+1} - [\frac{2y_k^T\, y_k}{v_k^T\, y_k} - \frac{v_k^T\, g_{k+1}}{v_k^T\, y_k} - \frac{y_k^T\, g_{k+1}}{v_k^T\, y_k}]\, v_k + \frac{v_k^T\, g_{k+1}}{v_k^T\, y_k}\, y_k$$

and go to step (2).

The second proposed algorithm (NEW2)

Another alternative self-scaling VM-update was implemented by Al-Bayati and Al-Salih, [15] for which the updating formula H_k has the form:

$$H_{k+1} = H_k - [\frac{H_k\, y_k\, v_k^T}{v_k^T\, y_k} + \frac{y_k^T\, H_k\, y_k}{v_k^T\, y_k} - \frac{v_k\, v_k^T}{v_k^T\, y_k}, \qquad (3\text{-}2\text{-}2\text{-}1)$$

where this updating formula generates also positive definite matrices.

Using I for H_k in the RHS of eq(3-2-2-1) we get the following new proposed method:

$$d_{k+1} = - H_{k+1}\, g_{k+1}$$

Hence:

$$d_{k+1} = - g_{k+1} - [\frac{y_k^T\, y_k}{(v_k^T\, y_k)^2} - v_k^T\, g_{k+1}]\, v_k + \frac{y_k^T\, g_{k+1}}{v_k^T\, y_k}\, v_k, \qquad (3\text{-}2\text{-}2\text{-}2)$$

We also note that if $v_k^T\, g_{k+1} = 0$ (using ELS) then
eq(3-2-2-2) reduce to:

$$d_{k+1} = - g_{k+1} + (\frac{y_k^T\, g_{k+1}}{v_k^T\, y_k})\, v_k.$$

which is the standard Hestenes and Stiefel CG-method.

Algorithm (NEW2)

We list below the outlines of the second proposed algorithm (NEW2). For an initial point x_1

Step (1): set k = 1, $d_k = - g_k / \| g_k \|$

Step (2): set $x_{k+1} = x_k + \lambda_k d_k$, where λ_k is a scalar chosen in such a way that $f_{k+1} < f_k$ (ILS).

Step (3): check for convergence, $\| g_{k+1} \| < \epsilon$, where ϵ is small positive tolerance, stop.

Step (4): otherwise, if k = n

or $\left| g_{k+1}^T\, g_k \right| \geq 0.2 \left| g_{k+1}^T\, g_{k+1} \right|$

compute the new search direction defined by

$$d_{k+1} = - g_{k+1}\, (\lambda_k\, d_k^T\, d_k\, / g_{k+1}^T\, g_{k+1});$$

set k = 1, and go to step (2). Else, set k = k + 1.

Step (5): compute the new search direction defined by

$$d_{k+1} = -g_{k+1} - [-\frac{y_k^T y_k}{(v_k^T y_k)^2} . v_k^T g_{k+1}] v_k + \frac{y_k^T g_{k+1}}{v_k^T y_k} v_k$$

and go to step (2).

Line search criterion

For all memoryless CG-algorithms discussed in this work it is necessary to ensure that:

$$v_k^T y_k > 0, \text{ for } k > t, \qquad (3\text{-}4\text{-}1)$$

in order to maintain the positive definite property and hence a downhill direction d will be obtained. However, condition (3-4-1) is in practice generally replaced by a slightly stronger line search criterion namely that line searches are terminated when both:

$$\left| d_k^T g_{k+1} \right| < \rho_1 \left| d_k^T g_{k+1} \right| \qquad (3\text{-}4\text{-}2)$$

and

$$f_{k+1} - f_k < \rho_2 v_k^T g_k, \qquad (3\text{-}4\text{-}3)$$

where these conditions are sufficient to ensure convergence of any descent method, as quoted by Shanno, [16]. Shanno found that $\rho_2 = 0.0001$ works satisfactorily, but, ρ_1 is the critical and sensitive parameter. He found that ρ_1 varies between 0.1 and 0.9, and therefore we have used the same two values for our first and second new algorithms defined in (3-2-1-3) and (3-2-2-2). For more details see Powell [17].

RESULTS AND CALCULATIONS

In order to assess the performance of the two new proposed algorithms (NEW1, NEW2), seven CG-algorithms are tested over (15) generalized selected well-known test functions with different dimensions where $100 \le n \le 2000$.

(I)	Shanno's method (Shanno)
(II)	Modified Shanno's method (MSHN).
(III)	Perry's method (Perry).
(IV)	Hestenes and Stiefel method (HS).
(V)	The new method (NEW1).
(VI)	The new method (NEW2).
(VII)	Fletcher and Reeves method (FRNAG).

For the purpose of a unique comparisons all these algorithms (except the FRNAG routine) are restarted every n iterations or

whenever

$$\left| g_{k+1}^T g_k \right| \ge 0.2 \left| g_{k+1}^T g_{k+1} \right|$$

is satisfied, with:

$$d_{k+1} = -g_{k+1} (\lambda_k d_k^T d_k / g_{k+1}^T g_{k+1}).$$

All the algorithms use exactly the same line search strategy which is the cubic fitting technique, directly adapted from Bunday, [18].

These algorithms are assumed to have convergence when each element of the gradient vector is less than 1.e-5; that is $\| g_{k+1} \| < 1 \times 10^{-5}$. All computations were performed on a 286 personal computer, with math co-processor.

The comparative performance for all these algorithms are evaluated by considering both the total number of function evaluations (NOF). Whereas NOF is the best measure of actual work done it is dependent on the linear search and the accuracy required; NOI is preferred by some others for this reason, but the requirement of higher accuracy (and so high NOF) can even reduce (NOI), both should therefore be taken into account.

Also we have computed the CPU time and the total minimum function value (F_{MIN}) required by each algorithm to satisfy the convergence criterion.

Indeed, all our numerical results are presented in the following tables:

Computation were carried out between all the considered seven CG-algorithms with dimensions 100, 1000 and 2000, respectively. Table 1 gives a very well-known study between the average of the NOI and NOF of the all selected test functions and for all selected dimensions 100, 1000 and 2000.

Table 1

The average number of NOI and NOF required by each test function for different dimensions

Test Fn.	FRNAG		PERRY		HS		SHANNO		MSHN		NEW1		NEW2	
	NOI	NOF	NOI	NOF	NOI	NOF	NOI	NOF	NOI	NOF	NOI	NOF	NOI	NOF
1-	51	249	11	30	11	30	11	32	11	48	11	32	11	32
2-	13	19	6	19	6	19	6	19	6	20	6	19	6	19
3-	77	205	27	67	27	67	27	67	28	193	42	58	19	46
4-	775	1498	43	132	49	138	63	193	48	230	40	107	44	115
5-	11	54	10	43	10	33	10	42	11	52	11	42	10	44
6-	151	340	25	79	25	79	26	83	24	96	26	81	25	82
7-	34	77	9	29	9	29	9	29	9	33	9	29	8	28
8-	18	51	6	20	6	20	6	20	6	28	6	20	6	20
9-	528	557	260	524	260	524	260	524	260	784	260	525	261	526
10-	334	640	37	88	38	89	37	88	38	146	36	86	36	84
11-	145	553	15	44	15	44	14	41	16	87	16	45	15	42
12-	441	782	26	58	26	58	26	58	26	82	26	58	26	58
13-	110	225	106	302	111	306	112	307	96	311	107	302	110	298
14-	81	182	29	76	29	76	27	69	30	128	32	78	30	76
15-	13	75	12	80	21	80	24	94	17	71	16	63	16	62

Taking the FRNAG routine as: 100% NOI; NOF and time yields:

Table 2

Tools	FRNAG	PERRY	HS	SHANNO	MSHN	NEW1	NEW2
NOI	100	33	33	34	32	32	31
NOF	100	41	40	43	57	39	37
Time	100	55	59	61	68	57	55

It is clear from the above table that both NEW1 and NEW2 are the most efficient algorithms according to our calculations and for our selected group of test functions. If we neglect the FRNAG routine: (It is coded in a Fortran language) we will give the following comparison:-

Table 3

Tools	NEW2	NEW1	HS	SHANNO	MSHN	PERRY
NOI	100	103	108	112	104	107
NOF	100	104	107	116	154	112
Time	100	104	107	112	124	100

It is obvious that the new proposed algorithms improve the other published CG-algorithms in (4-12)% NOI; (4-54)% NOF and (0-24)% Time.

Taking the FRNAG routine as: 100% NOI; NOF and time for n=1000 yields:-

Table 4

Tools	FRNAG	PERRY	HS	SHANNO	MSHN	NEW1	NEW2
NOI	100	92	22	21	20	20	21
NOF	100	27	27	29	38	26	26
Time	100	30	32	35	40	36	33

From the above table it is seen that both NEW1 and NEW2 are the most efficient algorithms according to our calculations and for our selected group of test functions. If we neglect the FRNAG routine we will give the following comparisons:-

Table 5

Tools	NEW2	NEW1	HS	SHANNO	MSHN	PERRY
NOI	100	89	100	104	95	99
NOF	100	101	104	109	143	101
Time	100	108	96	106	120	90

Clearly the new proposed algorithms improve the other published CG-algorithms in about 4% NOI (excluding Perry's algorithm); 43% NOF and about 20% time.

Finally, taking the FRNAG routine as: 100% NOI; NOF and time for n=2000 yields:-

Table 6

Tools	FRNAG	PERRY	HS	SHANNO	MSHN	NEW1	NEW2
NOI	100	24	25	25	26	24	24
NOF	100	30	30	31	51	30	30
Time	100	32	35	38	53	35	34

Appendix

All the presented test functions are from the general literature:

1. Generalized Beale Function:

$$f = [1.5-x_{2i-1}(1-x_{2i})]^2 + [2.25-x_{2i-1}(1-x^2_{2i})]^2 + [2.625-x_{2i-11}(1-x^3_{2i})]^2, \quad x_o = (0, 0)^T$$

2. Generalized Edgar & Himmel Function:

$$f = \sum_{i=1}^{n} (x_{2*i-1}-2)^4 + (x_{2*i-1}-2)^2 * x_{2*i}^2 + (x_{2*i}+1)^2, \quad x_o = (1, 0)^T$$

3. Non-Diagonal Variant of Rosenbrock Function:

$$f = \sum_{i=2}^{n} [100(x_1-x_i^2)^2 + (1-x_i)^2], \quad x_o = (-1; \ldots)^T$$

4. Generalized Powell Function:

$$f = \sum_{i=1}^{n/4} [(x_{4i-3}+10x_{4i-2})^2 + 5(x_{4i-1}-x_{4i})^2 + (x_{4i-2}-2x_{4i-1})^4 + 10(x_{4i-3}-x_{4i})^4], \quad x_o = (3, -1, 0, 1; \ldots)^T$$

5. Generalized Pen1. Function:

$$f = \sum_{i=1}^{n} [(x_i-1)^2 + \exp(x^2_i-0.25)^2], \quad x_o = i$$

6. Generalized Pen2. Function:

$$f = \sum_{i=1}^{n} [\exp(x_i-1)^2 + (x^2_i-0.25)^2], \quad x_o = i$$

7. Generalized Shallow Function:

$$f = \sum_{i=1}^{n/2} [(x^2_{2i-1}-x_{2i})^2 + (1-x_{2i-1})^2], \quad x_o = (-2; \ldots)^T$$

8. Generalized Strait Function:

$$f = \sum_{i=1}^{n} [(x^2_{2i-1}-x_{2i})^2 + 100(1-x_{2i-1})^2], \quad x_o = (-2; \ldots)^T$$

9. Generalized Tri. Function:

$$f = \sum_{i=2}^{n} [2x_i-x_{i-1}]^2, \quad x_o = (1; \ldots)^T$$

10. Generalized Wood Function:

$$f = \sum_{i=1}^{n/4} 100[(x_{4i-2}-x^2_{4i-3})^2] + (1-x_{4i-3})^2 + 90(x_{4i}-x^2_{4i-1})^2 + (1-x_{4i-1})^2 + 10.1[(x_{4i-2}-1)^2 + (x_{4i}-1)^2] + 19.8(x_{4i-2}-1)(x_{4i}-1), \quad x_o = (-3, -1, -3 -1; \ldots)^T$$

11. Generalized Cubic Function:

$$f = \sum_{i=1}^{n/4} [100(x_{2i}-x^3_{2i-1})^2 + (1-x_{2i-1})^2, \quad x_o = (-1.2, 1)^T$$

12. Generalized Dixon Function:

$$f = \sum_{i=1}^{n} [(1-x_1)^2 + (1-x_2)^2 + \sum_{i=1}^{n-1} (x_i^2 - x_{i+1})^2], \quad x_o = (-1; \ldots)^T$$

13. OSP, Function:

$$f = [\sum_{i=1}^{n} ix^2_i]^2, \quad x_o = (1; \ldots)^T$$

14. Generalized Rosenbrok Function:

$$f = \sum_{i-1}^{n/2} [100(x_{2i} - x^2_{2i-1})^2 + (1-x_{2i-1})^2], \quad x_o = (-1.2,1; \ldots)^T$$

15. Sum Function:

$$f = \sum_{i}^{n} [x_i - 1]^4, \quad x_o = (2; \ldots)^T$$

REFERENCES

[1] **Hestenes, M.R.** and **Stiefel, E., 1952**. Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards, 49: 409-488.

[2] **Fletcher, R.** and **Reeves, C.M., 1964**. Function minimization by conjugate gradations, Computer Journal, 7: 149-154.

[3] **Polak, E., 1970**. Computational methods in optimization a unified approach. Academic Press, New York.

[4] **Dixon, L.C.W., 1975**. Conjugate direction without linear search, Journal of Institute of Mathematics, 11: 317-328.

[5] **Al-Bayati, A.,** and **Al-Assady, N., 1986**. Conjugate gradient methods, technical report, Leeds University.

[6] **Edgar, T.F.** and **Himmelelbau, D.M., 1989**. Optimization of chemical processes, Department of Chemical Engineering, University of Texas.

[7] **Joe, H. Mize,** and **J.L. Kuester, 1973**. Optimization techniques with Fortran, McGraw, Hill, New York.

[8] **Powell, M.J.D., 1977**. Restart procedures for the conjugate gradient method, Math. Programming, 12: 241-254.

[9] **Perry, A., 1978**. A modified conjugate gradient algorithm, Journal of Operations Research, 26: 1073-1978.

[10] **Shanno, D.F., 1978a**. Conjugate gradient methods with inexact search, Mathematics of Operations Research, 3: 244-256.

[11] **Oren, S.S., 1974**. Self-scaling variable metric algorithm, Part II, Management Science, 20: 863-874.

[12] **Oren, S.S.** and **Luenberger, D., 1974**. Self-scaling variable metric algorithm, Part I, Management Science, 20: 845-862.

[13] **Oren, S.S.** and **Spedicato, E., 1978**. Optimal conditioning of self-scaling variable metric algorithms, Mathematical Programming, 10: 70-90.

[14] **Al-Bayati, A., 1991**. A new family of self-scaling variable metric algorithms for unconstrained optimization, Journal of Education and Science, Mosul University, 12: 25-54.

[15] **Al-Bayati, A.** and **Al-Salih, M., 1994**. New VM-Methods for nonlinear unconstrained optimization, to appear in the education and science magazine, University of Mosul.

[16] **Shanno, D.F., 1978**. On the convergence of a new conjugate gradient algorithm, SIAM Journal of Numerical Analysis, 15: 1247-1257.

[17] **Powell, M.J.D., 1981**. Nonlinear optimization, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 17-23.

[18] **Bunday, B., 1984**. Basic optimization methods, Edward Arnold, London.