APPLICATIONES MATHEMATICAE
27,2 (2000), pp. 153–165

J. KOKO (Clermont-Ferrand)

A CONJUGATE GRADIENT METHOD WITH QUASI-NEWTON APPROXIMATION

Abstract. The conjugate gradient method of Liu and Storey is an efficient minimization algorithm which uses second derivatives information, without saving matrices, by finite difference approximation. It is shown that the finite difference scheme can be removed by using a quasi-Newton approximation for computing a search direction, without loss of convergence. A conjugate gradient method based on BFGS approximation is proposed and compared with existing methods of the same class.

1. Introduction. We are concerned with the unconstrained minimization problem

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

with f a twice continuously differentiable function. When the dimension of (P) is large, conjugate gradient (CG) methods are particularly useful thanks to their storage saving properties. The classical conjugate gradient methods aim to solve (P) by a sequence of *line searches*

$$x_{k+1} = x_k + t_k d_k, \quad k = 1, 2, \ldots,$$

where t_k is the step length and the *search direction* d_k is of the form

$$d_k = -g_k + \beta_k d_{k-1}$$

with $g_k = \nabla f(x_k)$. There are many formulas for computing the coefficient β_k; they can be found in [9], [3], [12] and [8].

Liu and Storey [9] propose a new CG method in which the search direction is of the form

$$d_k = -\alpha_k g_k + \beta_k d_{k-1}, \quad \alpha_k > 0, \tag{1}$$

1991 *Mathematics Subject Classification*: 65K10, 49M07.
Key words and phrases: unconstrained high-dimensional optimization, conjugate gradient methods, Newton and quasi-Newton methods.

by considering the effects of an inexact line search. First, they write the Newton approximation of $f(x_{k+1})$, i.e.

$$F(x_k + t_k d_k) = f(x_k) + (g_k^T d_k)t_k + \tfrac{1}{2}(d_k^T H_k d_k)t_k^2, \qquad k \geq 2,$$

where $H_k = \nabla^2 f(x_k)$ is the Hessian of f at x_k. If H_k is positive definite, then

$$\min_{t_k > 0} F(x_k + t_k d_k) - f(x_k) \leq F(x_k + d_k) - f(x_k). \tag{2}$$

Finally, in order to improve the line search, Liu and Storey propose to compute (α_k, β_k) in (1) as a minimizer of the right hand side of (2), i.e. of the function

$$\Phi(\alpha, \beta) = F(x_k + d_k) - f(x_k) = (g_k^T d_k)t_k + \tfrac{1}{2}(d_k^T H_k d_k)t_k^2.$$

By a straightforward calculation, the coefficients α_k and β_k of the search direction (1) are then given by

$$\alpha_k = \frac{1}{D_k}[\|g_k\|^2 v_k - (g_k^T d_{k-1})w_k], \tag{3}$$

$$\beta_k = \frac{1}{D_k}[\|g_k\|^2 w_k - (g_k^T d_{k-1})u_k], \tag{4}$$

where

$$u_k = g_k^T H_k g_k, \tag{5}$$

$$v_k = d_{k-1}^T H_k d_{k-1}, \tag{6}$$

$$w_k = g_k^T H_k d_{k-1}, \qquad D_k = u_k v_k - w_k^2 > 0. \tag{7}$$

Liu and Storey [9, Theorem 2.1] show that their CG algorithm is globally convergent under line search conditions

$$f(x_k + t_k d_k) - f(x_k) \leq \sigma_1 t_k \nabla f(x_k)^T d_k, \qquad 0 < \sigma_1 < 1/2, \tag{8}$$

$$|\nabla f(x_k + t_k d_k)^T d_k| \leq -\sigma_2 \nabla f(x_k)^T d_k, \qquad 0 < \sigma_1 < \sigma_2 < 1, \tag{9}$$

assuming that the level set $L = \{x \mid f(x) \leq f(x_0)\}$ is bounded. The main conditions of their convergence theorem are:

$$u_k > 0, \qquad v_k > 0, \tag{10}$$

$$1 - \frac{w_k^2}{u_k v_k} \geq \frac{1}{4r_k}, \qquad \infty > r_k > 0, \tag{11}$$

$$\frac{u_k}{\|g_k\|^2}\left(\frac{v_k}{\|d_{k-1}\|^2}\right)^{-1} \leq r_k, \qquad \infty > r_k > 0. \tag{12}$$

In this paper, we will refer to CG of Liu and Storey [9] as the *LS algorithm*. To avoid the computation and storage of H_k, Liu and Storey [9] propose computing u_k, v_k and w_k using some form of finite difference approximation

$$u_k = \frac{1}{\gamma_k} g_k^T (\nabla f(x_k + \gamma_k g_k) - g_k), \tag{13}$$

$$v_k = \frac{1}{\delta_k} d_{k-1}^T (\nabla f(x_k + \delta_k d_{k-1}) - g_k), \tag{14}$$

$$w_k = \frac{1}{\delta_k} g_k^T (\nabla f(x_k + \delta_k d_{k-1}) - g_k), \tag{15}$$

where δ_k and γ_k are suitable small positive numbers. To avoid some extra gradient evaluations, Hu and Storey [7] propose computing v_k and w_k using the relation

$$H_k d_{k-1} \approx \frac{1}{t_{k-1}} (g_k - g_{k-1}), \tag{16}$$

derived from the mean-value theorem.

Since conditions (10) must be satisfied, H_k must be positive definite. But it is well known that this is possible, in general, only in some neighborhood of a local minimum. In addition, if the function evaluation is costly in time, it is preferable to evaluate it as rarely as possible. In this paper, we propose computing u_k, v_k and w_k using a BFGS approximation formula so that (10) and extra gradient evaluations are removed. In the next section we derive a LS type algorithm using BFGS approximation. Numerical results on test problems are presented in Section 3.

2. LS-BFGS algorithm. Let $Z_{k-1} = \operatorname{span}\{-g_k, d_{k-1}\}$ and $Q_{k-1} = (-g_k \; d_{k-1})$. Hu and Storey [8] show that the LS method is a two-dimensional Newton method in the sense that it uses as new direction, at the current point x_k, the Newton direction of the restriction of f to Z_{k-1}. Indeed, on Z_{k-1} the Hessian of f, at the current point, is

$$\widetilde{H}_k = Q_{k-1}^T H_k Q_{k-1}, \tag{17}$$

where $H_k = \nabla^2 f(x_k)$; and the gradient is $\widetilde{g}_k = Q_{k-1}^T g_k$. Thus, the new direction is given by

$$d_k = -\big(Q_{k-1} \widetilde{H}_k^{-1} Q_{k-1}^T\big) g_k, \tag{18}$$

or, in extended form, $d_k = -\alpha_k g_k + \beta_k d_{k-1}$, where

$$\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} = -\widetilde{H}_k^{-1} \widetilde{g}_k. \tag{19}$$

The interest of the analysis of Hu and Storey [8] is that it is possible to replace the true matrix $\widetilde{H}_k$ given by (17) by another one computed by quasi-Newton techniques.

All quantities (vectors and matrices) in the transformed space Z_k will be marked by attaching a tilde to the untransformed ones.

The matrix $\widetilde{H}_k$, given by (17), is of the form

$$\widetilde{H}_k = \begin{pmatrix} u_k & -w_k \\ -w_k & v_k \end{pmatrix},$$

and the condition (11) can be rewritten as

$$0 < u_k v_k/(4r_k) \le u_k v_k - w_k^2 = \det \widetilde{H}_k.$$

Thus, at each iteration k, (11) gives a bound from below for the determinant of $\widetilde{H}_k$. The conditions (10)–(11) therefore ensure that $\widetilde{H}_k$ is positive definite. Before replacing $\widetilde{H}_k$ in (18) by another positive definite matrix it is necessary to know whether the corresponding algorithm converges.

COROLLARY 1. *Suppose that the level set L of f is bounded and the line search conditions are* (8)–(9). *Let*

$$\widetilde{H}_k = \begin{pmatrix} u_k & -w_k \\ -w_k & v_k \end{pmatrix}$$

be a 2×2 matrix that satisfies (10)–(12), *and $Q_{k-1} = (-g_k\ d_{k-1})$. Then any LS type algorithm with search direction given by*

$$d_k = -(Q_{k-1}\widetilde{H}_k^{-1}Q_{k-1}^T)g_k \tag{20}$$

converges.

Proof. Since Liu and Storey [9, Theorem 2.1] used the quantities u_k, v_k and w_k without replacing them by (5)–(7), the corollary is valid. Note also that if $\widetilde{H}_k$ satisfies (11) then $w_k < \sqrt{u_k v_k}$, and therefore $g_k^T d_k < 0$. ∎

Corollary 1 enables us to use in (18) or (19) any other 2×2 positive definite matrices satisfying (10)–(12), instead of $\widetilde{H}_k$ given by (17). Since $\widetilde{H}_k$ is the reduced Hessian, we can replace it by a reduced Hessian approximation using the BFGS correction formula. Details on the latter can be found, for example, in [1] and [2].

Let $\Delta x_k = x_{k+1} - x_k$ and $\Delta g_k = g_{k+1} - g_k$ with $\Delta x_k^T \Delta g_k > 0$. Then the BFGS correction formula, which constructs an approximation to the Hessian matrix of f, is defined by

$$H_{k+1} = U_{\mathrm{BFGS}}(\Delta x_k, \Delta g_k, H_k) \tag{21}$$

$$\Updownarrow$$

$$H_{k+1} = H_k + \frac{\Delta g_k \Delta g_k^T}{\Delta x_k^T \Delta g_k} - \frac{H_k \Delta x_k \Delta x_k^T H_k}{\Delta x_k^T H_k \Delta x_k}. \tag{22}$$

We will use the update function (21), introduced by Dennis and Moré [2], to write (22) with suitable arguments. As in Nazareth's SAR methods [10], [11], the general scheme for updating $\widetilde{H}_k$ at each iteration is as follows:

(i) $\overline{H}_k = Q_k^T H_k Q_k$, the projection of H_k onto $Z_k = \mathrm{span}\{-g_{k+1}, d_k\}$.
(ii) $\Delta\widetilde{x}_k = Q_k^T \Delta x_k$, $\Delta\widetilde{g}_k = Q_k^T \Delta g_k$.
(iii) If $\Delta\widetilde{x}_k^T \Delta\widetilde{g}_k > 0$ then use the BFGS correction formula

$$\widetilde{H}_{k+1} = U_{\mathrm{BFGS}}(\Delta\widetilde{x}_k, \Delta\widetilde{g}_k, \overline{H}_k).$$

(iv) Extend the approximation to the whole space $\mathbb{R}^n$.

In this scheme, the crucial points are (iii) and (iv). The relation $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > 0$ is needed to ensure that $\widetilde{H}_{k+1}$ is positive definite. Note that the line search (8)–(9) will only ensure that $\Delta x_k^T\Delta g_k > 0$. Then we have to find a relation between the line search (8)–(9) and the inner product $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k$. The theorem below gives such a relation.

THEOREM 1. *Suppose that in the line search the stopping conditions are* (8)–(9). *Then* $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > 0$ *if and only if*

$$-\|d_k\|^2/\sigma_2 < g_{k+1}^T\Delta g_k < (1-\sigma_2)\|d_k\|^2/\sigma_2. \tag{23}$$

Proof. From $Q_k = (-g_{k+1}\ d_k)$, we have

$$\Delta\widetilde{x}_k = t_k\begin{pmatrix} -g_{k+1}^T d_k \\ d_k^T d_k\end{pmatrix} \quad\text{and}\quad \Delta\widetilde{g}_k = \begin{pmatrix} -g_{k+1}^T\Delta g_k \\ d_k^T\Delta g_k\end{pmatrix}.$$

Then

$$\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k = t_k[(g_{k+1}^T d_k)(g_{k+1}^T\Delta g_k) + (d_k^T d_k)d_k^T\Delta g_k]. \tag{24}$$

Note that $\Delta x_k^T\Delta g_k > 0$ implies $d_k^T\Delta g_k > 0$. The troublesome term in (24) is the first term on the right. But from (8)–(9) we know that

$$g_{k+1}^T d_k \in [\sigma_2 g_k^T d_k, -\sigma_2 g_k^T d_k].$$

Sufficiency. If $g_{k+1}^T\Delta g_k > 0$ then

$$\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k \ge t_k[\sigma_2(g_k^T d_k)(g_{k+1}^T\Delta g_k) + \|d_k\|^2(\sigma_2-1)g_k^T d_k].$$

Taking $\sigma_2(g_k^T d_k)$ as a factor, it follows that

$$\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > t_k\sigma_2(g_k^T d_k)[(g_{k+1}^T\Delta g_k) + (\sigma_2-1)\|d_k\|^2/\sigma_2] > 0.$$

In the same way, one shows that if $g_{k+1}^T\Delta g_k < 0$ then

$$\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > -t_k\sigma_2(g_k^T d_k)[(g_{k+1}^T\Delta g_k) + \|d_k\|^2/\sigma_2] > 0.$$

Necessity. If $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > 0$, then we have

$$(g_{k+1}^T d_k)(g_{k+1}^T\Delta g_k) + (d_k^T d_k)d_k^T\Delta g_k > 0.$$

If $g_{k+1}^T d_k > 0$, then

$$g_{k+1}^T\Delta g_k > -\|d_k\|^2(d_k^T\Delta g_k)/(g_{k+1}^T d_k).$$

Since $d_k^T\Delta g_k = d_k^T g_{k+1} - d_k^T g_k > -g_k^T d_k$ and $g_{k+1}^T d_k \le -\sigma_2 g_k^T d_k$, we get

$$g_{k+1}^T\Delta g_k > -\|d_k\|^2/\sigma_2.$$

If $g_{k+1}^T d_k < 0$, then

$$g_{k+1}^T\Delta g_k < -\|d_k\|^2(d_k^T\Delta g_k)/(g_{k+1}^T d_k).$$

Since $-d_k^T\Delta g_k = -d_k^T g_{k+1} + d_k^T g_k < (1-\sigma_2)g_k^T d_k$ and $g_{k+1}^T d_k > \sigma_2 g_k^T d_k$, we have

$$g_{k+1}^T d_k < (1-\sigma_2)\|d_k\|^2/\sigma_2. \ \blacksquare$$

Inequalities (23) show the relation between the line search parameter σ_2 and the inner product $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k$. Greater values of σ_2 will reduce the interval defined by (23) for $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > 0$. Note that if an exact line search is used to determine the step length t_k then $\Delta x_k^T\Delta g_k > 0$ implies $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > 0$.

Suppose that $\Delta\widetilde{x}_k^T\Delta\widetilde{g}_k > 0$ with $\overline{H}_k = Q_k^T H_k Q_k$, the projection of the Hessian H_k onto Z_k. We compute $\widetilde{H}_{k+1}$ using the BFGS correction formula (22). To extend this Hessian approximation to the whole space $\mathbb{R}^n$, we have to define $\overline{Q}_k = (p_k\ q_k)$, the orthonormalized form of Q_k , with

$$p_k = -\frac{1}{\|g_{k+1}\|}g_{k+1}, \qquad q_k = \frac{1}{s_k}\left(d_k - \frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2}g_{k+1}\right),$$

where

$$s_k = \left(\|d_k\|^2 - \frac{(g_{k+1}^T d_k)^2}{\|g_{k+1}\|^2}\right)^{1/2}. \tag{25}$$

Note that the main property of $\overline{Q}_k$ is $\overline{Q}_k\overline{Q}_k^T z = z$ for all $z \in Z_k$. Therefore,

$$(I_n - \overline{Q}_k\overline{Q}_k^T)g_{k+1} = 0, \qquad (I_n - \overline{Q}_k\overline{Q}_k^T)d_k = 0.$$

The columns of $(I_n - Q_k\overline{Q}_k^T)^T$ span $Z_k^\perp$ and

$$H_{k+1} = Q_k\widetilde{H}_{k+1}^{-1}Q_k^T + (I_n - \overline{Q}_k\overline{Q}_k^T) \tag{26}$$

gives the extension of the approximate Hessian inverse $\widetilde{H}_{k+1}^{-1}$ to the whole $\mathbb{R}^n$. The new search direction is then given by

$$d_{k+1} = -H_{k+1}g_{k+1} = -(Q_k\widetilde{H}_{k+1}^{-1}Q_k^T)g_{k+1}.$$

The formula (26) is only used to compute the projection $\overline{H}_k$ of the Hessian H_k onto the subspace $Z_k = \operatorname{span}\{-g_{k+1}, d_k\}$. It will appear implicitly in the formula

$$\overline{H}_k = (Q_k^T Q_{k-1})\widetilde{H}_k(Q_k^T Q_{k-1})^T + Q_k^T Q_k - (Q_k^T\overline{Q}_{k-1})(Q_k^T\overline{Q}_{k-1})^T, \tag{27}$$

the 2×2 matrix used as the previous approximation to the Hessian in the BFGS correction formula. Note that $\overline{H}_k$ can be computed efficiently by inner products using the vectors g_{k+1}, d_k, g_k and d_{k-1} only.

LS-BFGS Algorithm

0. $k \leftarrow 0$, $d_0 \leftarrow -g_0$.
   Line search (8)–(9): $x_1 = x_0 + t_0 d_0$.
   $Q_0 = (-g_1\ d_0)$; $\widetilde{H}_1 \leftarrow I_2$.
1. If $\|g_{k+1}\| < \varepsilon$ then STOP otherwise $k \leftarrow k+1$.
2. If $k > n$ then go to 7.
3. $d_k = -\alpha_k g_k + \beta_k d_{k-1}$.
   Line search (8)–(9): $x_{k+1} = x_k + t_k d_k$.
   $\Delta x_k = x_{k+1} - x_k$; $\Delta g_k = g_{k+1} - g_k$.
4. If $\sigma_2(g_{k+1}^T\Delta g_k) \le -\|d_k\|^2$ or $\sigma_2(g_{k+1}^T\Delta g_k) \ge (1-\sigma_2)\|d_k\|^2$ then go to 7.

5. $Q_k = (-g_{k+1}\ d_k)$; $\Delta\widetilde{x}_k = Q_k^T \Delta x_k$; $\Delta\widetilde{g}_k = Q_k^T \Delta g_k$; $\widetilde{g}_{k+1} = Q_k^T g_{k+1}$;
   $V_k = Q_k^T Q_{k-1}$; $W_k = Q_k^T \overline{Q}_{k-1}$ and $\overline{H}_k = V_k^T \widetilde{H}_k V_k + Q_k^T Q_k - W_k^T W_k$.
   BFGS update: $\widetilde{H}_{k+1} = U_{\mathrm{BFGS}}(\Delta\widetilde{x}_k, \Delta\widetilde{g}_k, \overline{H}_k)$ with formula (22).
6. If $1 - w_{k+1}^2/(u_{k+1}v_{k+1}) \ge 1/(4r_{k+1})$, and
   $u_{k+1}\|d_k\|^2/(v_{k+1}\|g_{k+1}\|^2) \le r_{k+1}$, $r_{k+1} > 0$,
   then $(\alpha_{k+1}\ \beta_{k+1})^T = -\widetilde{H}_{k+1}^{-1}\widetilde{g}_{k+1}$ and go to 1.
7. $x_0 \leftarrow x_k$ and go to 0.

Instead of $\widetilde{H}_k^{-1}$, we can work directly with the *inverse reduced Hessian* approximation of f. But, for this, we have to "reverse" the conditions of the convergence theorem of Liu and Storey [9, Theorem 2.1].

COROLLARY 2. *Let*

$$\widetilde{H}_k = \begin{pmatrix} \overline{u}_k & \overline{w}_k \\ \overline{w}_k & \overline{v}_k \end{pmatrix}$$

be a 2×2 *matrix such that*:

(i) $\widetilde{H}_k$ *is positive definite*,

(ii) $1 - \dfrac{\overline{w}_k^2}{\overline{u}_k \overline{v}_k} \ge \dfrac{1}{4r_k}, \qquad \infty > r_k > 0,$

(iii) $\dfrac{\overline{v}_k}{\|d_{k-1}\|^2}\left(\dfrac{\overline{u}_k}{\|g_k\|^2}\right)^{-1} \le r_k, \qquad \infty > r_k > 0.$

Then, *under the line search conditions* (8)–(9), *any LS type algorithm with the search direction given by*

$$d_k = -(Q_{k-1}\widetilde{H}_k Q_{k-1}^T) g_k \tag{28}$$

converges.

The BFGS correction formula (22) is then replaced by

$$\widetilde{H}_{k+1} = \overline{H}_k + \left(1 + \frac{\Delta\widetilde{g}_k^T \overline{H}_k \Delta\widetilde{g}_k}{\Delta\widetilde{x}_k^T \Delta\widetilde{g}_k}\right)\frac{\Delta\widetilde{x}_k \Delta\widetilde{x}_k^T}{\Delta\widetilde{x}_k^T \Delta\widetilde{g}_k} - \frac{\Delta\widetilde{x}_k \Delta\widetilde{g}_k^T \overline{H}_k + \overline{H}_k \Delta\widetilde{g}_k \Delta\widetilde{x}_k^T}{\Delta\widetilde{x}_k^T \Delta\widetilde{g}_k}$$

which constructs an *inverse* Hessian approximation.

To compute the new approximation to the reduced Hessian $\widetilde{H}_k$ in step 5 of the LS-BFGS algorithm, we need at worst ten inner products; and at best seven inner products, if $\|d_{k-1}\|^2$, $\|g_k\|^2$ and s_k (given by (25)) are computed in the previous iteration and saved. The SAR algorithm requires the same number of operations for computing $\widetilde{H}_k$.

The most economical version of the LS method is obtained using (13) and (16) for computing α_k and β_k. Then in the LS method, in addition to one gradient evaluation, we need at worst six inner products.

It appears therefore that the LS-BFGS algorithm (or SAR algorithm) can be profitable if evaluating ∇f (or f) is more time-consuming than computing six inner products.

3. Algorithms and implementation. We have tested the new algorithm outlined in Section 2, the LS algorithm of Hu and Storey [7] and the SAR algorithm of Nazareth [10] on the collection of test problems given in Section 3.2.

We have used the line search given by Gilbert and Lemarechal [4] with initial step length

$$t_0 = \min\{2, 2(f(x_k) - f^*)/g_k^T d_k\},$$

where f^* is an estimate of the optimal function value. For all the test problems considered, we set $f^* = 0$, since the optimal function values are all nonnegative. The line search parameters in (8)–(9) are $\sigma_1 = 0.0001$ and $\sigma_2 = 0.1$.

In all cases the stopping condition is

$$\|g_k\| < 10^{-5} \max(1, \|x_k\|).$$

The sequence $\{r_k\}$ needed to ensure global convergence is given by $r_k \equiv 10^{10}$ for $k \geq 1$. The sequence $\{\gamma_k\}$ used in the finite difference scheme (13) is

$$\gamma_k = 4\|g_k\|^{-1} 10^{-10}.$$

This choice is better for large scale problems and can affect the performance of the LS algorithm in low-dimensional problems. On the other hand, a very small value in the numerator of γ_k can cause numerical difficulties for high-dimensional problems.

All the calculations were performed on a Sun Ultra 1 workstation, in double precision arithmetic.

3.1. *Algorithms.* We now detail the algorithms used in our tests; they differ mainly in computing the coefficients α_k and β_k of the search direction (1).

LS: The *Generalized Conjugate Gradient* algorithm of Liu and Storey, using (13) and (16), outlined in Hu and Storey [7]. Storage requirement: $6n$.

SAR: The *Successive Affine Reduction* algorithm of Nazareth [10], [11], the two-dimensional case ($z_k = \{\Delta x_k, \Delta g_k\}$). Restart is made with the LS algorithm, i.e. $\widetilde{H}_0$ is computed with (13)–(16). Storage requirement: $6n$.

LSB: The LS-BFGS algorithm outlined in Section 2. Restart is made with the LS algorithm as in the SAR algorithm. Storage requirement: $6n$.

Each algorithm was run in two versions:

1. The natural version.
2. The line search (8)–(9) is carried out if the conditions

$$f(x_k + d_k) - f(x_k) \leq \beta' g_k^T d_k, \qquad \beta' = 0.0001, \tag{29}$$

$$d_k^T \nabla f(x_k + d_k) \geq \beta g_k^T d_k, \qquad \beta = 0.9, \tag{30}$$

are not satisfied.

In the tables, the version corresponds to the number at the end of the algorithm name, e.g. LS1 is the natural LS algorithm and LS2 is the LS algorithm with unit step length strategy (29)–(30).

3.2. *Test problems*

PROBLEM 1. The extended Beale function

$$f(x) = \sum_{i=1}^{n/2}[(1.5 - x_{2i-1}(1 - x_{2i}^3))^2 + (2.25 - x_{2i-1}(1 - x_{2i}^2))^2 + (2.625 - x_{2i-1}(1 - x_{2i}^3))^2], \qquad n = 2, 4, 6, \ldots,$$

with $x_0 = (1, 1, \ldots, 1)^T$.

PROBLEM 2. The extended Miele and Cantrell function

$$f(x) = \sum_{i=1}^{n/4}[(\exp(x_{4i-3}) - x_{4i-2})^2 + 100(x_{4i-2} - x_{4i-1})^6 + (\tan(x_{4i-1} - x_{4i}))^4 + x_{4i-3}^8], \qquad n = 4, 20, 40, 60, \ldots,$$

with $x_0 = (1, 2, 2, 2, 1, 2, 2, 2, \ldots, 1, 2, 2, 2)^T$.

PROBLEM 3. The penalty 1 function

$$f(x) = 10^{-5}\sum_{i=1}^{n}(x_i - 1)^2 + \Big(\sum_{i=1}^{n} x_i^2 - 0.25\Big)^2, \qquad n = 1, 2, \ldots,$$

with $x_i^0 = i$, $i = 1, \ldots, n$.

PROBLEM 4. The penalty 2 function

$$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 + 10^{-3}\Big(\sum_{i=1}^{n} x_i^2 - 0.25\Big)^2, \qquad n = 1, 2, \ldots,$$

with $x_i^0 = i$, $i = 1, \ldots, n$.

PROBLEM 5. The extended Rosenbrock function

$$f(x) = \sum_{i=1}^{n/2}[100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2], \quad n = 2, 4, 6, \ldots,$$

with

$$\begin{cases} x_{2i}^0 = 1.0, \\ x_{2i-1}^0 = -1.2 + 0.4i/n, \end{cases} \qquad i = 1, \ldots, n/2.$$

The choice of this starting point is justified in [10].

Problem 6. The trigonometric function

$$f(x) = \sum_{i=1}^{n} \Big[n + i - \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j)\Big]^2, \qquad n = 1, 2, \ldots,$$

where $a_{ij} = \delta_{ij}$, $b_{ij} = i\delta_{ij} + 1$ and δ_{ij} is the Kronecker delta, with $x^0 = (1/n, \ldots, 1/n)^T$.

Problem 7. The Brown function

$$f(x) = \Big[\sum_{i=1}^{n/2} (x_{2i-1} - 3)\Big]^2 + 0.0001 \sum_{i=1}^{n/2} [(x_{2i-1} - 3)^2 - (x_{2i-1} - x_{2i}) + \exp(20(x_{2i-1} - x_{2i}))], \qquad n = 2, 4, 6, \ldots,$$

with $x_0 = (0, -1, 0, -1, \ldots, 0, -1)^T$.

Problem 8. The extended Powell function

$$f(x) = \sum_{i=1}^{n/4} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4], \qquad n = 4, 8, \ldots,$$

with $x_0 = (3, -1, 0, 3, 3, -1, 0, 3, \ldots, 3, -1, 0, 3)^T$.

Problem 9. The tridiagonal function

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

with $x^0 = (1, 1, \ldots, 1)^T$.

Problem 10. The extended Wood function

$$f(x) = \sum_{i=1}^{n/4} [100(x_{4i-2} - x_{4i-3}^2)^2 + (1 - x_{4i-3})^2 + 90(x_{4i} - x_{4i-1}^2)^2 + (1 - x_{4i-1})^2 + 10(x_{4i-2} + x_{4i} - 2)^2 + 0.1(x_{4i-2} - x_{4i})^2], \qquad n = 4, 8, \ldots,$$

with $x^0 = (-3, -1, -3, -1, \ldots, -3, -1)^T$.

3.3. *Tables.* In the tables, No is the number of the problem and n the number of variables. Since the conjugate gradient type methods are mainly useful for large problems, in our test problems n is very large (except in problem 6). NI is the number of iterations, NF/NG the number of function/gradient calls and CPU the Central Processor Unit time in seconds. The symbol "$*$", in a table, means that the run of the corresponding algorithm was stopped because the limit of 1500 function or gradient evaluations was exceeded $(\max(\mathrm{NF}, \mathrm{NG}) > 1500)$.

TABLE 1. Performance of SAR algorithms

Problems		SAR1			SAR2		
No	n	NI	NF/NG	CPU	NI	NF/NG	CPU
1	1000	20	26/27	1.43	29	33/34	2.10
1	10000	19	24/25	13.21	28	35/36	20.05
2	1000	47	135/137	27.19	132	273/322	43.83
2	10000	49	140/142	39.31	217	382/489	195.83
3	1000	60	129/130	35.07	97	132/134	46.89
3	10000	66	141/142	46.28	98	134/135	70.26
4	1000	20	47/48	1.63	36	49/70	2.47
4	10000	23	66/67	19.90	37	76/77	38.36
5	1000	29	69/70	1.95	36	59/60	2.57
5	10000	29	69/70	119.30	36	59/60	24.95
6	100	25	126/134	7.17	26	132/140	8.22
6	1000	35	169/171	423.47	27	134/141	395.72
7	1000	20	90/91	1.62	13	91/92	1.27
7	10000	23	93/94	18.09	22	97/98	16.96
8	1000	120	204/209	16.94	213	421/429	17.68
8	10000	229	360/373	190.49	248	490/504	203.73
9	1000	329	647/649	24.28	343	951/952	29.52
9	10000	$*$	$*$	557.66	$*$	$*$	462.97
10	1000	68	117/120	6.32	260	666/672	21.98
10	10000	85	160/163	49.04	261	686/690	220.83

TABLE 2. Performance of LS algorithms

Problems		LS1			LS2		
No	n	NI	NF/NG	CPU	NI	NF/NG	CPU
1	1000	16	36/28	0.57	18	32/33	0.47
1	10000	17	36/28	5.87	18	32/33	4.76
2	1000	85	165/249	4.69	134	136/269	4.52
2	10000	83	158/240	44.95	34	137/269	45.20
3	1000	85	179/263	11.38	229	183/311	14.55
3	10000	102	248/349	51.00	239	413/651	105.09
4	1000	12	47/48	0.69	36	49/70	0.67
4	10000	17	49/51	9.84	12	47/68	8.75
5	1000	26	76/91	0.99	31	77/97	1.11
5	10000	26	76/91	9.98	31	77/97	11.30
6	100	21	67/97	4.27	22	65/86	3.99
6	1000	23	72/102	362.72	25	71/92	341.57
7	1000	20	89/108	0.75	21	81/92	0.83
7	10000	22	89/110	17.18	24	81/95	21.32
8	1000	67	103/169	2.92	$*$	$*$	28.85
8	10000	91	149/239	41.36	$*$	$*$	293.39
9	1000	288	576/862	12.48	288	288/575	10.05
9	10000	$*$	$*$	220.06	$*$	$*$	270.35
10	1000	105	179/283	4.46	148	154/301	5.40
10	10000	158	188/145	23.76	164	170/333	60.05

TABLE 3. Performance of LSB algorithms

Problems		LSB1			LSB2		
No	n	NI	NF/NG	CPU	NI	NF/NG	CPU
1	1000	10	44/51	1.52	17	44/48	1.45
1	10000	11	48/63	10.19	15	31/35	11.30
2	1000	42	110/125	2.83	34	116/129	3.31
2	10000	53	150/167	26.03	45	126/136	30.34
3	1000	25	85/97	1.92	28	126/139	2.87
3	10000	26	87/103	42.57	28	453/561	170.64
4	1000	18	40/49	1.00	36	49/70	1.47
4	10000	24	40/51	14.38	17	56/63	15.65
5	1000	27	65/77	1.20	27	79/91	1.68
5	10000	27	66/78	12.17	27	83/95	17.98
6	100	25	51/65	2.87	26	63/65	3.08
6	1000	27	55/68	110.72	27	67/72	128.43
7	1000	18	75/83	0.39	15	60/67	0.57
7	10000	27	85/93	14.78	20	75/83	20.68
8	1000	52	67/89	1.09	$*$	$*$	3.63
8	10000	57	112/131	21.81	$*$	$*$	41.70
9	1000	145	281/307	6.61	357	582/897	38.82
9	10000	$*$	$*$	268.43	$*$	$*$	376.35
10	1000	48	89/121	2.40	$*$	$*$	41.01
10	10000	57	103/117	15.35	$*$	$*$	411.76

4. Conclusions. We have investigated the behavior of our LS-BFGS algorithm, the original LS algorithm and the SAR algorithm. The numerical results are reported in Tables 1 to 3.

The LS algorithm appears to be better in terms of CPU time, for relatively fast evaluation functions (problems 1, 3, 4, 5 and 9). The SAR algorithms have the best rate NF/NG but require more function/gradient evaluations than the LS type algorithms. For costly evaluation functions (problems 2, 6, 7, 8 and 10) the saving time obtained with LSB1 is significant. The unit step test (29)–(30) does not work very well in the three algorithms. Numerical experiments have shown that failures of LSB1 and LSB2 are due to round-off errors in the computation of $\widetilde{H}_{k+1}$ in step 5 of LS-BFGS algorithm, because of using Q_k and the orthonormalized matrix $\overline{Q}_{k-1}$. Preconditioning $\widetilde{H}_k$ before projection would probably clear the round-off errors and improve the LS-BFGS algorithm. We are working in this direction, using the results of [8].

References

[1] C. G. Broyden, *The convergence of a class of double-rank minimization algorithms*, J. Inst. Math. Appl. 6 (1970), 60–76.

[2] J. E. Dennis and J. J. Moré, *Quasi-Newton methods, motivations and theory*, SIAM Rev. 19 (1977), 46–89.

[3] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradient*, Computer J. 7 (1964), 143–154.

[4] J. C. Gilbert and C. Lemarechal, *The modules M1QN2, N1QN2, M1QN3 and N1QN3*, INRIA, technical report, 1989.

[5] J. C. Gilbert and J. Nocedal, *Global convergence properties of conjugate gradient methods for optimization*, SIAM J. Optim. 2 (1992), 21–42.

[6] M. Hestenes and E. Stiefel, *Methods of conjugate gradient for solving linear systems*, J. Res. Nat. Bureau Standards B48 (1952), 409–436.

[7] Y. F. Hu and C. Storey, *Efficient generalized conjugate gradient algorithms, Part 2: Implementation*, J. Optim. Theory Appl. 69 (1991), 139–152.

[8] —, —, *Preconditioned low-order Newton methods*, ibid. 79 (1993), 311–331.

[9] Y. Liu and C. Storey, *Efficient generalized conjugate gradient algorithms, Part 1: Theory*, ibid. 69 (1991), 129–137.

[10] J. L. Nazareth, *The method of successive affine reduction for nonlinear minimization*, Math. Programming 35 (1985), 97–109.

[11] —, *Conjugate gradient methods less dependent on conjugacy*, SIAM Rev. 28 (1986), 501–511.

[12] E. Polak and G. Ribière, *Note sur la convergence des méthodes de directions conjuguées*, RAIRO Rech. Opér. 16 (1969), 35–43.

Jonas Koko
ISIMA–LIMOS, Université Clermont-Ferrand II
Campus des Cézeaux, BP 125
F-63173 Aubière Cedex, France
E-mail: koko@sp.isima.fr

Received on 29.1.1998;
revised version on 26.7.1999