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

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

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 \( \beta_k \); 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

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

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.

\[
F(x_k + t_k d_k) = f(x_k) + (g_k^T d_k) t_k + \frac{1}{2} (d_k^T H_k d_k) t_k^2, \quad k \geq 2,
\]

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

\[
\min_{t_k > 0} F(x_k + t_k d_k) - f(x_k) \leq F(x_k + d_k) - f(x_k).
\]

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

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

By a straightforward calculation, the coefficients \( \alpha_k \) and \( \beta_k \) of the search direction (1) are then given by

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

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

where

\[
u_k = g_k^T H_k g_k,
\]

\[v_k = d_{k-1}^T H_k d_{k-1},
\]

\[w_k = g_k^T H_k d_{k-1}, \quad D_k = u_k v_k - w_k^2 > 0.
\]

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

\[(f(x_k + t_k d_k) - f(x_k) \leq \sigma_1 t_k \nabla f(x_k)^T d_k, \quad 0 < \sigma_1 < 1/2),
\]

\[|\nabla f(x_k + t_k d_k)^T d_k| \leq -\sigma_2 \nabla f(x_k)^T d_k, \quad 0 < \sigma_1 < \sigma_2 < 1,
\]

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

\[u_k > 0, \quad v_k > 0,
\]

\[1 - \frac{w_k^2}{u_k v_k} \geq \frac{1}{4r_k}, \quad \infty > r_k > 0,
\]

\[
u_k \left( \frac{v_k}{||g_k||^2 (||d_{k-1}||^2)} \right)^{-1} \leq r_k, \quad \infty > r_k > 0.
\]

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

\[
u_k = \frac{1}{\gamma_k} g_k^T (\nabla f(x_k + \gamma_k g_k) - g_k).
\]
\( v_k = \frac{1}{\delta_k} d_k^T (\nabla f(x_k + \delta_k d_{k-1}) - g_k), \)

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

where \( \delta_k \) and \( \gamma_k \) 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

\[ H_k d_{k-1} \approx \frac{1}{t_{k-1}} (g_k - g_{k-1}), \]

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 \( Z_{k-1} = \text{span}\{-g_k, d_{k-1}\} \) 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

\[ \tilde{H}_k = Q_{k-1}^T H_k Q_{k-1}, \]

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

\[ d_k = -Q_{k-1}^{-1} \tilde{H}_k^{-1} Q_{k-1}^T g_k, \]

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

\[ \begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} = -\tilde{H}_k^{-1} \tilde{g}_k. \]

The interest of the analysis of Hu and Storey [8] is that it is possible to replace the true matrix \( \tilde{H}_k \) 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 \( \tilde{H}_k \), given by (17), is of the form

\[ \tilde{H}_k = \begin{pmatrix} u_k & -w_k \\ -w_k & v_k \end{pmatrix}, \]
and the condition (11) can be rewritten as

\[ 0 < u_kv_k/(4r_k) \leq u_kv_k - w_k^2 = \det \tilde{H}_k. \]

Thus, at each iteration \( k \), (11) gives a bound from below for the determinant of \( \tilde{H}_k \). The conditions (10)–(11) therefore ensure that \( \tilde{H}_k \) is positive definite.

Before replacing \( \tilde{H}_k \) 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 \( \tilde{H}_k = \begin{pmatrix} u_k & -w_k \\ -w_k & v_k \end{pmatrix} \) be a 2 × 2 matrix that satisfies (10)–(12), and \( Q^{-1} = (-g_k d_{k-1}) \). Then any LS type algorithm with search direction given by

\[ d_k = -(Q^{-1} \tilde{H}_k^{-1} Q^T g_k) \]

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 \( \tilde{H}_k \) satisfies (11) then \( w_k < \sqrt{u_kv_k} \), and therefore \( g^T d_k < 0 \).

Corollary 1 enables us to use in (18) or (19) any other 2 × 2 positive definite matrices satisfying (10)–(12), instead of \( \tilde{H}_k \) given by (17). Since \( \tilde{H}_k \) 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

\[ H_{k+1} = U_{BFGS}(\Delta x_k, \Delta g_k, H_k) \]

\[ \Downarrow \]

\[ H_{k+1} = H_k + \frac{\Delta g_k \Delta g_k^T}{\Delta x_k^T \Delta g_k} - \frac{H_k \Delta x_k \Delta x_k^T H_k}{\Delta x_k^T H_k \Delta x_k}. \]

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 \( \tilde{H}_k \) at each iteration is as follows:

(i) \( \Pi_k = Q_k^T H_k Q_k \), the projection of \( H_k \) onto \( Z_k = \text{span}\{-g_{k+1}, d_k\} \).

(ii) \( \Delta \tilde{x}_k = Q_k^T \Delta x_k, \Delta \tilde{g}_k = Q_k^T \Delta g_k \).

(iii) If \( \Delta \tilde{x}_k^T \Delta \tilde{g}_k > 0 \) then use the BFGS correction formula

\[ \tilde{H}_{k+1} = U_{BFGS}(\Delta \tilde{x}_k, \Delta \tilde{g}_k, \Pi_k) \].

(iv) Extend the approximation to the whole space \( \mathbb{R}^n \).
Theorem 1. Suppose that in the line search the stopping conditions are (8)–(9). Then \( \Delta \tilde{x}_k^T \Delta \tilde{g}_k > 0 \) if and only if
\[
-\|d_k\|^2/\sigma_2 < g_{k+1}^T \Delta g_k < (1 - \sigma_2)\|d_k\|^2/\sigma_2.
\]

Proof. From \( Q_k = (-g_{k+1}^T d_k) \), we have
\[
\Delta \tilde{x}_k = t_k \left( -g_{k+1}^T d_k \right) \quad \text{and} \quad \Delta \tilde{g}_k = \left( -g_{k+1}^T \Delta g_k \right).
\]
Then
\[
\Delta \tilde{x}_k^T \Delta \tilde{g}_k = t_k \left( (g_{k+1}^T d_k)(g_{k+1}^T \Delta g_k) + (d_k^T d_k) \right) \Delta g_k).
\]
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
\[
\Delta \tilde{x}_k^T \Delta \tilde{g}_k \geq t_k \left[ \sigma_2 (g_k^T d_k)(g_{k+1}^T \Delta g_k) + \|d_k\|^2 (\sigma_2 - 1) g_k^T d_k \right].
\]
Taking \( \sigma_2 (g_k^T d_k) \) as a factor, it follows that
\[
\Delta \tilde{x}_k^T \Delta \tilde{g}_k > t_k \left( g_k^T d_k \right) \left[ (g_{k+1}^T \Delta g_k) + (\sigma_2 - 1) \|d_k\|^2/\sigma_2 \right] > 0.
\]

 Necessity. If \( \Delta \tilde{x}_k^T \Delta \tilde{g}_k > 0 \), then we have
\[
g_{k+1}^T (g_{k+1}^T \Delta g_k) + (d_k^T d_k) \Delta \tilde{g}_k > 0.
\]
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 \( g_{k+1}^T d_k \leq -\sigma_2 g_k^T d_k \), 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
\[
g_{k+1}^T d_k < (1 - \sigma_2)\|d_k\|^2/\sigma_2. \]
Inequalities (23) show the relation between the line search parameter \( \sigma_2 \) and the inner product \( \Delta \tilde{x}_k^T \Delta g_k \). Greater values of \( \sigma_2 \) will reduce the interval defined by (23) for \( \Delta \tilde{x}_k^T \Delta g_k > 0 \). Note that if an exact line search is used to determine the step length \( t_k \) then \( \Delta \tilde{x}_k^T \Delta g_k > 0 \) implies \( \Delta \tilde{x}_k^T \Delta g_k > 0 \).

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

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

where

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

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

\[
(I_n - \tilde{Q}_k \tilde{Q}_k^T)g_{k+1} = 0, \quad (I_n - \tilde{Q}_k \tilde{Q}_k^T)d_k = 0.
\]

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

\[
H_{k+1} = Q_k \tilde{H}_{k+1}^{-1} Q_k^T + (I_n - \tilde{Q}_k \tilde{Q}_k^T)
\]
gives the extension of the approximate Hessian inverse \( \tilde{H}_{k+1}^{-1} \) to the whole \( \mathbb{R}^n \). The new search direction is then given by

\[
d_{k+1} = -H_{k+1} g_{k+1} = -(Q_k \tilde{H}_{k+1}^{-1} Q_k^T) g_{k+1}.
\]

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

\[
\tilde{H}_k = (Q_k^T Q_{k-1}) \tilde{H}_k (Q_k^T Q_{k-1})^T + Q_k^T Q_k - (Q_k^T \tilde{Q}_{k-1})(Q_k^T \tilde{Q}_{k-1})^T,
\]

the 2 \times 2 matrix used as the previous approximation to the Hessian in the BFGS correction formula. Note that \( \tilde{H}_k \) 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): \( x_1 = x_0 + t_0 d_0 \).

\( Q_0 = (-g_1, d_0); \tilde{H}_1 \leftarrow I_2 \).

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 \( \sigma_2(g_{k+1}^T \Delta g_k) \leq -\|d_k\|^2 \) or \( \sigma_2(g_{k+1}^T \Delta g_k) \geq (1 - \sigma_2)\|d_k\|^2 \) then go to 7.
5. $Q_k = (-g_{k+1} d_k)$; $\Delta \tilde{x}_k = Q_k^T \Delta x_k$; $\Delta \tilde{g}_k = Q_k^T \Delta g_k$; $\tilde{g}_{k+1} = Q_k^T g_{k+1}$; $V_k = Q_k^T Q_{k-1}$; $W_k = Q_k^T Q_{k-1}$ and $H_k = V_k^T H_k V_k + Q_k^T Q_k - W_k^T W_k$.

BFGS update: $H_{k+1} = U_{\text{BFGS}}(\Delta \tilde{x}_k, \Delta \tilde{g}_k, H_k)$ with formula (22).

6. If $1 - \frac{u_{k+1}^2}{w_{k+1}^2} / (u_{k+1} v_{k+1}) \geq 1/(4r_{k+1})$, and $u_{k+1} \|d_k\|^2 / (v_{k+1} \|g_{k+1}\|^2) \leq r_{k+1}$, $r_{k+1} > 0$,
then $(\alpha_{k+1} \beta_{k+1})^T = -H_{k+1}^{-1} \tilde{g}_{k+1}$ and go to 1.

7. $x_0 \leftarrow x_k$ and go to 0.

Instead of $\tilde{H}_k^{-1}$, 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

$$\tilde{H}_k = \begin{pmatrix} \boldsymbol{m}_k & \boldsymbol{m}_k \\ \overline{w}_k & \overline{v}_k \end{pmatrix}$$

be a $2 \times 2$ matrix such that:

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

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

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

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

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

converges.

The BFGS correction formula (22) is then replaced by

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

which constructs an inverse Hessian approximation.

To compute the new approximation to the reduced Hessian $\tilde{H}_k$ 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 $\tilde{H}_k$.

The most economical version of the LS method is obtained using (13) and (16) for computing $\alpha_k$ and $\beta_k$. 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 $\nabla f$ (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 $\gamma_k$ 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 $\alpha_k$ and $\beta_k$ 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. $\tilde{H}_0$ 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

$$f(x_k + d_k) - f(x_k) \leq \beta' g_k^T d_k, \quad \beta' = 0.0001,$$

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 $\gamma_k$ 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 $\alpha_k$ and $\beta_k$ 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. $\tilde{H}_0$ 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

$$f(x_k + d_k) - f(x_k) \leq \beta' g_k^T d_k, \quad \beta' = 0.0001,$$
Conjugate gradient method

(30) \[ d_k^T \nabla f(x_k + d_k) \geq \beta g_k^T d_k, \quad \beta = 0.9, \]
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

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

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

PROBLEM 2. The extended Miele and Cantrell function

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

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

PROBLEM 3. The penalty 1 function

\[
f(x) = 10^{-5} \sum_{i=1}^{n} (x_i - 1)^2 + \left( \sum_{i=1}^{n} x_i^2 - 0.25 \right)^2, \quad n = 1, 2, \ldots,
\]

with \( x_i^n = i, i = 1, \ldots, n \).

PROBLEM 4. The penalty 2 function

\[
f(x) = \sum_{i=1}^{n} (x_i - 1)^2 + 10^{-3} \left( \sum_{i=1}^{n} x_i^2 - 0.25 \right)^2, \quad n = 1, 2, \ldots,
\]

with \( x_i^n = i, i = 1, \ldots, n \).

PROBLEM 5. The extended Rosenbrock function

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

with

\[
\begin{align*}
    x_{2i}^0 &= 1.0, \quad i = 1, \ldots, n/2, \\
    x_{2i-1}^0 &= -1.2 + 0.4i/n, \quad i = 1, \ldots, n/2.
\end{align*}
\]

The choice of this starting point is justified in [10].
**Problem 6.** The trigonometric function

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

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

**Problem 7.** The Brown function

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

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

**Problem 8.** The extended Powell function

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

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

**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

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

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 (\( \max(\text{NF}, \text{NG}) > 1500 \)).
### TABLE 1. Performance of SAR algorithms

<table>
<thead>
<tr>
<th>Problems</th>
<th>SAR1</th>
<th></th>
<th>SAR2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>n</td>
<td>NI</td>
<td>NF/NG</td>
<td>CPU</td>
<td>NI</td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
<td>20</td>
<td>26/27</td>
<td>1.43</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>47</td>
<td>135/137</td>
<td>27.19</td>
<td>132</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>60</td>
<td>129/130</td>
<td>35.07</td>
<td>97</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>20</td>
<td>47/48</td>
<td>1.63</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>29</td>
<td>69/70</td>
<td>1.95</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>25</td>
<td>126/134</td>
<td>7.17</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>35</td>
<td>169/171</td>
<td>423.47</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>12</td>
<td>47/48</td>
<td>0.69</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>26</td>
<td>76/91</td>
<td>0.99</td>
<td>31</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>23</td>
<td>93/94</td>
<td>18.09</td>
<td>22</td>
</tr>
</tbody>
</table>

### TABLE 2. Performance of LS algorithms

<table>
<thead>
<tr>
<th>Problems</th>
<th>LS1</th>
<th></th>
<th>LS2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>n</td>
<td>NI</td>
<td>NF/NG</td>
<td>CPU</td>
<td>NI</td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
<td>16</td>
<td>36/28</td>
<td>0.57</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>85</td>
<td>105/249</td>
<td>4.69</td>
<td>134</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>85</td>
<td>179/263</td>
<td>11.38</td>
<td>229</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>12</td>
<td>47/48</td>
<td>0.69</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>26</td>
<td>76/91</td>
<td>0.99</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>23</td>
<td>72/102</td>
<td>362.72</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>20</td>
<td>89/108</td>
<td>0.75</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>22</td>
<td>89/110</td>
<td>17.18</td>
<td>24</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>67</td>
<td>103/169</td>
<td>2.92</td>
<td>*</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>91</td>
<td>149/239</td>
<td>41.36</td>
<td>*</td>
</tr>
<tr>
<td>11</td>
<td>1000</td>
<td>288</td>
<td>576/862</td>
<td>12.48</td>
<td>288</td>
</tr>
<tr>
<td>12</td>
<td>1000</td>
<td>*</td>
<td>*</td>
<td>220.06</td>
<td>*</td>
</tr>
<tr>
<td>13</td>
<td>1000</td>
<td>105</td>
<td>179/283</td>
<td>4.46</td>
<td>146</td>
</tr>
<tr>
<td>14</td>
<td>10000</td>
<td>158</td>
<td>188/145</td>
<td>23.76</td>
<td>164</td>
</tr>
</tbody>
</table>
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 $\bar{Q}_{k-1}$. Preconditioning $\bar{H}_k$ 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

Conjugate gradient method


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