

# Planar Conjugate Gradient Algorithm for Large-Scale Unconstrained Optimization, Part 2: Application<sup>1,2</sup>

G. FASANO<sup>3</sup>

Communicated by L. C. W. Dixon

**Abstract.** In this paper, we describe an application of the planar conjugate gradient method introduced in Part 1 (Ref. 1) and aimed at solving indefinite nonsingular sets of linear equations. We prove that it can be used fruitfully within optimization frameworks; in particular, we present a globally convergent truncated Newton scheme, which uses the above planar method for solving the Newton equation. Finally, our approach is tested over several problems from the CUTE collection (Ref. 2).

**Key Words.** Large-scale unconstrained optimization, truncated Newton method, nonconvex problems, planar conjugate gradient method.

## 1. Introduction

In this paper, we consider the application of a conjugate gradient (CG) based algorithm (namely, Algorithm FLR) described in Part 1 (Ref. 1) within optimization frameworks. Algorithm FLR is a Krylov subspace method (Ref. 3) for the iterative solution of indefinite and nonsingular linear systems. In particular, we adopt it for solving the Newton equation

$$H_k d + g_k = 0, \tag{1}$$

---

<sup>1</sup>This work was supported by MIUR, FIRB Research Program on Large-Scale Nonlinear Optimization, Rome, Italy.

<sup>2</sup>The author acknowledges Luigi Grippo and Stefano Lucidi, who contributed considerably to the elaboration of this paper. The exchange of experiences with Massimo Roma was a constant help in the investigation. The author expresses his gratitude to the Associate Editor and the referees for suggestions and corrections.

<sup>3</sup>Postdoctoral Fellow, Department of Computer Science and Systems A. Ruberti, University of Rome-La Sapienza, Rome, Italy.

where  $H_k = \nabla^2 f(x_k)$  is the Hessian matrix and  $g_k = \nabla f(x_k)$  is the gradient of a nonconvex function  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ , with  $f \in \mathcal{C}^2(\mathbb{R}^n)$  and  $n$  large. Equation (1) arises frequently within large-scale optimization and in a wide range of gradient methods (Ref. 4). In particular, we focus on the iterative application of Newton-type methods, which are very popular algorithms. The success of Newton-type methods is due mainly to their fast convergence near the solution, along with reasonable use of resources (Ref. 4); indeed, unlike the pure Newton method, they do not require second derivatives.

The above schemes attempt at solving iteratively the general optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \quad n \text{ large}; \quad (2)$$

then update the current iterate  $x_k$  according to one of the relations

$$x_{k+1} = x_k + \alpha_k d_k, \quad (3a)$$

$$x_{k+1} = x_k + \alpha_k d_k + \beta_k s_k, \quad (3b)$$

where the direction  $d_k$  is an approximate solution (Newton-type direction) of (1), while  $s_k$  is a nonascent negative curvature direction (Ref. 5); i.e.,

$$\begin{aligned} s_k^T H_k s_k &\leq 0, \\ g_k^T s_k &\leq 0, \end{aligned}$$

which summarize the knowledge of  $f(x)$  on a concavity region near  $x_k$ . Newton-type algorithms employing relation (3a) can guarantee weaker properties for the solution with respect to the use of relation (3b). More exactly, Eq. (3b) is used in case we claim for the solution point  $x^*$  to verify the second-order necessary conditions of optimality, i.e.  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*) \succeq 0$ . On the other hand, by means of Eq. (3a), it is only possible to converge to stationary points, which are not maximizers of  $f(x)$ ; see Refs. 6–7. In this paper, we consider in Section 3 an iterative scheme corresponding to Eq. (3a).

The coefficient  $\alpha_k, \beta_k$  in (3) are chosen according to a stabilization technique [either a monotone one (Ref. 5) or a nonmonotone one (Ref. 8)], which ensures the global convergence of the overall algorithm. Algorithms which employ iteratively Eq. (3a) have been studied in Refs. 9–11; on the other hand, methods designed according to the Eq. (3b) have been studied in Refs. 12–14.

When  $n$  is large, the solution of (1) is often investigated by means of a conjugate gradient based iterative method (Ref. 10); the efficiency in

Table 1. Truncated Newton Algorithm TN for solving the Problem (2).

---

For  $k=0, 1, \dots$ , repeat the steps below until convergence:

- (a) Compute  $f(x_k), \nabla f(x_k)$ .
  - (b) Verify a test for convergence on  $\|\nabla f(x_k)\|$ .
  - (c) Solve approximately (1), compute  $d_k$  (gradient related) and possibly  $s_k$ .
  - (d) Compute  $\alpha_k$  and possibly  $\beta_k$  in (3) with a line search procedure.
  - (e) Update the iterate  $x_{k+1}$  according to (3).
- 

solving (1) relies substantially on the possibility of using the same iterative method for computing rapidly the directions  $d_k$  and  $s_k$ . Also, when  $n$  is large, the convergence of the overall optimization method see for an approximate solution of equation (1); see Refs. 9, 15. This is a well-known result for Newton-type methods and a relevant issue within the so-called truncated Newton methods. These are very general efficient schemes for approaching the iterative solution of large-scale unconstrained optimization problems (Ref. 10). In particular, truncated Newton methods consider the problem (2) and work according to Table 1. Observe that at Step (c), a choice of the iterative method for solving approximately Eq. (1) is required.

In Section 3, we propose a truncated Newton scheme, which implements a globally convergent modified Newton method (Ref. 4) for the solution of the general optimization problem (2), where  $f$  is a nonconvex function. For the approximate solution of Eq. (1), we adopt the planar CG Algorithm FLR of Ref. 1, which is effective and inexpensive with respect to other planar algorithms (Refs. 16–18); see Section 2. In Section 2.1, we prove that the vector  $d_k$  provided at Step (c) by Algorithm FLR can be worthwhile when used within truncated Newton schemes.

For a complete list of practical problems solved by means of truncated Newton schemes, we suggest Ref. 10 and the cited references.

In this paper, we use the symbol  $\|\cdot\|$  for indicating the Euclidean norm of either a real vector or a real matrix. With  $x^T y$ , we denote the Euclidean inner product between the vectors  $x, y \in \mathbb{R}^n$ . For a positive-definite [semidefinite] matrix  $A \in \mathbb{R}^{n \times n}$ , we use the notation  $A > 0 [A \geq 0]$ . The symbol  $\equiv$  stands for equivalent;  $\lambda_M, \lambda_m$  represent the largest and the smallest absolute value of the eigenvalues of the Hessian matrix  $H_k = \nabla^2 f(x_k)$ . Quantities calculated Step  $k$  are denoted with the subscript  $k$ .

In Section 2, we describe the use of Algorithm FLR for approximately solving the Newton Eq. (1). Section 3 describes a truncated Newton method based on Algorithm FLR. Section 4 deals with the numerical results. Finally, Section 5 contains the conclusions.

## 2. Algorithm FLR for the Newton Equation

As observed in the previous section, the solution of the large-scale Newton equation (1) requires the application of a suitable iterative method. In this section, we consider the use of Algorithm FLR, introduced in Part 1 (Ref. 1). The latter method is suitable in case the Newton equation is indefinite and nonsingular; therefore, it may be a natural tool within algorithms for solving iteratively the nonconvex problem (2). In Table 2, the computational complexity of Algorithm FLR is compared with that of the planar methods proposed in Ref. 17 (Algorithm Lue) Ref. 18 (Algorithm Fas), and Ref. 16 (Algorithm Hes). In particular, for these algorithms, we report in Table 2 the following three features, related to the complexity of computation:

- (i) the engaging memory (Mem), including the machine registers used in each step;
- (ii) the number of floating point operations (Flops);
- (iii) the number of matrix  $\times$  vector products (H\*p).

For each planar algorithm reported, we considered features (i), (ii), (iii) for three different steps: the initialization, the standard CG step (Step  $i_A$ ), and the planar CG step (Step  $i_B$ ). In particular, Flops represents the number of floating point operations (multiplications) required by the step: a matrix  $\times$  vector product requires  $n^2$  operations, while an inner product requires  $n$  operations. Observe that Step  $i_B$  in Algorithms Hes and FLR requires the same computational burden; on the other hand, Step  $i_A$  in Algorithm FLR is significantly cheaper with respect to Algorithm Hes.

### 2.1. Using Algorithm FLR for Generating Gradient Related Directions.

We consider the solution of the Newton equation (1) within a truncated Newton scheme; in particular, the solution is to be achieved by means of Algorithm FLR.

In Sections 1 and 2, we discussed the motivation for using a specific iterative method for solving efficiently the Newton equation (1) in the case where  $H_k \neq 0$ . We described the importance of using a proper iterative method for exploring the regions of convexity and concavity of  $f(x)$  in (2). Now, suppose that the iterative method has detected a vector  $d \in \mathbb{R}^n$  such that

$$d^T H_k d < 0; \tag{4}$$

then (see page 9 of Ref. 10) “such a direction can be used as part of a search direction, since either  $d$  or  $-d$  is a direction of nonascent.” This idea is discussed in Refs. 13, 11, 19.

Table 2. Comparison among complexities of planar CG methods.

Algorithm	Step	Mem	Flops	H* <sub>p</sub>
CG	Initialization	$2n + \mathcal{O}(1)$	$n^2$	1
	Step $i_A$	$4n + \mathcal{O}(1)$	$n^2 + 6n + \mathcal{O}(1)$	1
Hes	Initialization	$3n + \mathcal{O}(1)$	$2n^2$	2
	Step $i_A$	$5n + \mathcal{O}(1)$	$2n^2 + 8n + \mathcal{O}(1)$	2
	Step $i_B$	$7n + \mathcal{O}(1)$	$2n^2 + 14n + \mathcal{O}(1)$	2
Lue	Initialization	$2n + \mathcal{O}(1)$	$n^2$	1
	Step $i_A$	$4n + \mathcal{O}(1)$	$n^2 + 6n + \mathcal{O}(1)$	1
	Step $i_B$	$6n + \mathcal{O}(1)$	$2n^2 + 13n + \mathcal{O}(1)$	2
Fas	Initialization	$2n + \mathcal{O}(1)$	$n^2$	1
	Step $i_A$	$4n + \mathcal{O}(1)$	$n^2 + 6n + \mathcal{O}(1)$	1
	Step $i_B$	$6n + \mathcal{O}(1)$	$2n^2 + 12n + \mathcal{O}(1)$	2
FLR	Initialization	$2n + \mathcal{O}(1)$	$n^2$	1
	Step $i_A$	$5n + \mathcal{O}(1)$	$n^2 + 7n + \mathcal{O}(1)$	1
	Step $i_B$	$7n + \mathcal{O}(1)$	$2n^2 + 14n + \mathcal{O}(1)$	2

To maintain the global convergence properties of the line search based truncated Newton method in Table 1, the search direction  $d_k$  satisfies the relations (Ref. 10)

$$d_k^T g_k \leq -c_1 \|g_k\|^{h_1}, \quad c_1, h_1 > 0, \tag{5a}$$

$$\|d_k\| \leq c_2 \|g_k\|^{h_2}, \quad c_2, h_2 > 0; \tag{5b}$$

i.e.,  $d_k$  is gradient related. Here, we prove that the use of Algorithm FLR for solving the Newton equation (1), can provide iteratively a gradient-related direction  $d_k$ . More specifically, we calculate  $d_k$  as

$$d_k = d_k^P + d_k^N + d_k^{Pla}, \tag{6}$$

where (see the test on the quantity  $p_i^T H_k p_i$  at Step  $i$  of Algorithm FLR)

$$d_k^P = \sum_{i \in I^P} (p_i^T r_i / p_i^T H_k p_i) p_i,$$

$$I^P = \{i \geq 1 : p_i^T H_k p_i \geq \epsilon_i \|P_i\|\};$$

$$d_k^N = - \sum_{i \in I^N} (p_i^T r_i / p_i^T H_k p_i) p_i,$$

$$I^N = \{i \geq 1 : p_i^T H_k p_i \leq -\epsilon_i \|P_i\|^2\};$$

$$d_k^{Pla} = \sum_{i \in I^{Pla}} \left[ (p_i^T r_i / \|H_k p_i\|^2) p_i + (q_i^T r_i / \|H_k q_i\|^2) q_i \right],$$

$$I^{Pla} = \{i \geq 1 : |p_i^T H_k p_i| < \epsilon_i \|P_i\|^2\}.$$

In Ref. 7, Dembo and Steihaug proved that the direction  $d_k^P$  is gradient related (equivalently, they proved that the CG applied to the linear system (1), with  $H_k$  positive definite, generates the gradient-related direction  $d_k^P$ ). Similarly, in Ref. 6, Grippo, Lampariello, and Lucidi obtained the same result with the direction  $d_k^P + d_k^N$ , which better resembles the Newton direction in respect to  $d_k^P$ , since it considers also the contribution of vectors with the property (4). Here, we give evidence that the definition of the new direction  $d_k^{Pla}$ , introduced with the application of the planar Algorithm FLR, improves the likelihood of  $d_k$  with Newton direction in respect to  $d_k^P + d_k^N$  (see also Section 3) and yields the properties (5). This aims at completing the evolution drawn by Refs. 7, 6 when dealing with the indefinite and nonsingular matrix  $H_k$ .

**Theorem 2.1.** Consider the Newton equation (1) and let the symmetric matrix  $H_k$  be indefinite and nonsingular. Suppose that for all  $k \geq 1$ , the quantity  $\lambda_m$  is uniformly bounded away from zero. Then, the direction  $d_k$  defined in (6) and (7), calculated with Algorithm FLR, is gradient related i.e., the relations (5) hold.

**Proof.** After few arrangements and considering that  $r_1 = -g_k$ , for (5a) we have

$$\begin{aligned} d_k^T g_k &= (d_k^P + d_k^N + d_k^{Pla})^T g_k \\ &= \sum_{i \in I^P} -(p_i^T r_i / p_i^T H_k p_i) p_i^T r_1 + \left[ - \sum_{i \in I^N} -(p_i^T r_i / p_i^T H_k p_i) p_i^T r_1 \right] \\ &\quad + \sum_{i \in I^{Pla}} - \left[ (p_i^T r_i / \|H_k p_i\|^2) p_i^T r_1 + (q_i^T r_i / \|H_k q_i\|^2) q_i^T r_1 \right], \end{aligned}$$

and from Theorem 2.1 of Ref. 1 the relation

$$p_i^T r_i = p_i^T r_1, \quad i \geq 1,$$

holds. Moreover, we prove that

$$q_i^T r_i = q_i^T r_1, \quad i \geq 1.$$

Let  $t_i$  and  $\alpha_i, i \geq 1$ , be defined as (see Ref. 1 and Algorithm FLR)

$$\begin{aligned} \text{if } |p_i^T A p_i| \geq \epsilon_i \|p_i\|^2, & \quad \text{then } \alpha_i = a_i \text{ and } t_i = p_i, \\ \text{if } |p_i^T A p_i| < \epsilon_i \|p_i\|^2, & \quad \text{then } \begin{cases} \alpha_i = \hat{c}_i, & t_i = p_i, \\ \alpha_{i+1} = \hat{d}_i, & t_{i+1} = q_i. \end{cases} \end{aligned}$$

Then, since

$$r_i = r_1 - \sum_{j=1}^{i-1} \alpha_j H_k t_j,$$

from (B2) of Theorem 2.1 in Ref. 1, we obtain

$$q_i^T r_i = q_i^T \left[ r_1 - \sum_{j=1}^{i-1} \alpha_j H_k t_j \right] = q_i^T r_1.$$

For proving (5a), two cases must be analyzed: either the first step of Algorithm FLR is Step 1<sub>A</sub> and we have

$$\begin{aligned} (d_k^P + d_k^N + d_k^{Pla})^T g_k &\leq -(p_1^T r_1)^2 / |p_1^T H_k p_1| \\ &\leq -\|r_1\|^4 / \lambda_M \|r_1\|^2 \\ &= -(1/\lambda_M) \|r_1\|^2, \end{aligned}$$

or the first step is Step 1<sub>B</sub> and we have

$$\begin{aligned} (d_k^P + d_k^N + d_k^{Pla})^T g_k &\leq -(p_1^T r_1)^2 / \|H_k p_1\|^2 \\ &\leq -\|r_1\|^4 / \lambda_M^2 \|r_1\|^2 \\ &= -(1/\lambda_M^2) \|r_1\|^2. \end{aligned} \tag{7}$$

Therefore, we obtain the final relation

$$d_k^T g_k \leq -\min \{1/\lambda_M^2, 1/\lambda_M\} \|g_k\|^2, \tag{8}$$

which proves (5a).

For proving (5b) [again  $p_i^T r_i = p_i^T r_1$  and let  $\epsilon = \min_i \{\epsilon_i\}$ , with  $\epsilon_i$  defined at Step  $i$  of algorithm FLR], recalling that  $r_1 = -g_k$ ,

$$\begin{aligned} \|d_k^P\| &\leq \left\| \sum_{i \in I^P} (p_i p_i^T / p_i^T H_k p_i) r_1 \right\| \\ &\leq \sum_{i \in I^P} (\|p_i p_i^T\| \epsilon_i \|p_i\|^2) \|r_1\| \\ &\leq (n/\epsilon) \|g_k\| \end{aligned}$$

and similarly,

$$\|d_k^N\| \leq (n/\epsilon) \|g_k\|.$$

Thus, for proving (5b) it suffices to give evidence that a similar inequality holds for the direction  $d_k^{\text{Pla}}$ . Since again

$$p_i^T r_i = p_i^T r_1 \quad \text{and} \quad q_i^T r_i = q_i^T r_1,$$

we simply have

$$\begin{aligned} \|d_k^{\text{Pla}}\| &\leq \sum_{i \in I^{\text{Pla}}} \left[ \|(p_i^T r_i / \|H_k p_i\|^2) p_i\| + \|(q_i^T r_i / \|H_k q_i\|^2) q_i\| \right] \\ &\leq \sum_{i \in I^{\text{Pla}}} \left[ \|p_i p_i^T\| \|r_1\| / \|H_k p_i\|^2 + \|q_i q_i^T\| \|r_1\| / \|H_k q_i\|^2 \right] \\ &\leq \sum_{i \in I^{\text{Pla}}} \left[ \|p_i\|^2 / \lambda_m^2 \|p_i\|^2 + \|q_i\|^2 / \lambda_m^2 \|q_i\|^2 \right] \|r_1\| \\ &\leq (n/2)(2/\lambda_m^2) \|r_1\| \\ &\leq (n/\lambda_m^2) \|r_1\|; \end{aligned} \tag{9}$$

therefore, (5b) holds with

$$\|d_k\| \leq 3n \max\{1/\epsilon, 1/\lambda_m^2\} \|g_k\|. \tag{10}$$

Observe that the inequalities (8) and (10) imply the standard uniform descent condition

$$d_k^T g_k \leq -\epsilon_0 \|d_k\| \|g_k\|, \quad k \geq 1,$$

where  $\epsilon_0$  depends on  $\epsilon$ ,  $\lambda_m$ ,  $\lambda_M$ . We conclude this section remarking that the computation of the vector  $d_k^{\text{Pla}}$  does not need further calculation with respect to the computational burden involved in performing planar steps. In particular, both the matrix  $\times$  vector products  $H_k p_i$  and  $H_k q_i$  are available already at the outset of Step  $i_B$ .

### 3. Truncated Newton Method Based on Algorithm FLR

In Section 2, we have highlighted that, in the case where the Newton equation (1) is solved by means of Algorithm FLR, a gradient-related direction  $d_k$  is available when the iterative method stops. This suggests a natural embedding of Algorithm FLR within the truncated Newton algorithm introduced in Table 1, where the global convergence of the optimization method requires the generation of descent directions which are gradient related too. Here, we present the scheme TNFLR, which uses Algorithm FLR at step (c); i.e., we use Algorithm FLR to generate a gradient-related direction for the line search procedure. In Table 3, we

Table 3. Algorithm TNFLR providing the gradient related direction  $d_k$ .

---

Step 0.	Data. $H_k, g_k, 0 < \eta_k < 1, \gamma_1 > 0, \gamma_2 > 0, h_1 > 0, h_2 > 0$ .
Step 1.	Set $i = 1, r_1 = -g_k, p_1 = r_1, d_1 = 0, \bar{d}_1 = 0$ .
Step 2.	Compute $v_i = p_i^T H_k p_i$ , set $\epsilon_i > 0$ . If $H_k p_i = 0$ , then go to Step 3. If $ v_i  \geq \epsilon_i \ p_i\ ^2$ , then go to CG- $i_A$ . If $ v_i  < \epsilon_i \ p_i\ ^2$ , then go to PL- $i_B$ .
CG- $i_A$	Compute $\alpha_i = r_i^T p_i / v_i$ , set $d_{i+1} = d_i + \alpha_i p_i, r_{i+1} = r_i - \alpha_i H_k p_i$ . Set $\bar{d}_{i+1} = \begin{cases} \bar{d}_i - \alpha_i p_i, & \text{if } v_i \leq -\epsilon_i \ p_i\ ^2, \\ \bar{d}_i + \alpha_i p_i, & \text{if } v_i \geq \epsilon_i \ p_i\ ^2. \end{cases}$ If $\ r_{i+1}\  > \eta_k \ r_1\ $ , then compute $\beta_i = \ r_{i+1}\ ^2 / \ r_i\ ^2$ , set $p_{i+1} = r_{i+1} + \beta_i p_i$ , set $i = i + 1$ , go to Step 2. If $\ r_{i+1}\  \leq \eta_k \ r_1\ $ , then set $i = i + 1$ , go to Step 3.
PL- $i_B$ .	If $i = 1$ , then set $q_i = H_k p_i$ . If $i > 1$ and the previous step is CG- $(i - 1)_A$ , then set $\beta_{i-1} = -(H_k p_{i-1})^T H_k p_i / v_{i-1}$ and $q_i = H_k p_i + \beta_{i-1} p_{i-1}$ . If $i > 1$ and the previous step is PL- $(i - 2)_B$ , then set $\hat{\beta}_{i-2} = -(H_k q_{i-2})^T H_k p_i$ and $q_i = H_k p_i + \hat{\beta}_{i-2} (v_{i-2} q_{i-2} - \delta_{i-2} p_{i-2}) / \Delta_{i-2}$ . Compute $c_i = r_i^T p_i, f_i = r_i^T q_i, \delta_i = p_i^T H_k q_i, e_i = q_i^T H_k q_i, \Delta_i = v_i e_i - \delta_i^2$ . Compute $\hat{c}_i = (c_i e_i - \delta_i f_i) / \Delta_i, \hat{d}_i = (v_i f_i - \delta_i c_i) / \Delta_i$ . Set $d_{i+2} = d_i + \hat{c}_i p_i + \hat{d}_i q_i, r_{i+2} = r_i - \hat{c}_i H_k p_i - \hat{d}_i H_k q_i$ . Set $\bar{d}_{i+2} = \bar{d}_i + c_i / \ H_k p_i\ ^2 p_i + f_i / \ H_k q_i\ ^2 q_i$ . If $\ r_{i+2}\  > \eta_k \ r_1\ $ , then compute $\hat{b}_i = -q_i^T H_k r_{i+2}$ , Set $p_{i+2} = r_{i+2} + \hat{b}_i (v_i q_i - \delta_i p_i) / \Delta_i$ , set $i = i + 2$ , go to Step 2. If $\ r_{i+1}\  \leq \eta_k \ r_1\ $ , then set $i = i + 2$ , go to Step 3.
Step 3.	Compute the gradient-related direction $d_k$ as $d_k = \begin{cases} -g_k, & \text{if } i = 1, \\ d_i, & \text{if } d_i^T g_k \leq -\gamma_1 \ g_k\ ^{h_1} \text{ and } \ d_i\  \leq \gamma_2 \ g_k\ ^{h_2}, \\ \bar{d}_i, & \text{otherwise.} \end{cases}$

---

report this scheme, which stops at Step 3 after generating the direction  $d_k$ . According to Theorem 2.3 of Ref. 1, Algorithm TNFLR of Table 3 cannot cycle infinitely.

Substantially, after the initialization in Step 0 and Step 1, at Step 2 a test decides whether Algorithm FLR performs a conjugate gradient (CG- $i_A$ ) iteration or a planar (PL- $i_B$ ) iteration. Then, within each iteration, the vector  $d_i$  and  $\bar{d}_i$  are updated iteratively. In the end, when the algorithm stops, a gradient-related direction  $d_k$  is available at Step 3, using the information contained in either the current Newton direction  $d_i$  or the vector  $\bar{d}_i$ . This guarantees (see Section 2.1) that, in any case,  $d_k$  is gradient related. Observe that, in contrast to the standard CG, the iterative procedure that we adopt does not stop prematurely, provided that the Hessian matrix  $H_k$  is nonsingular. This aims at approximating the Newton direction as much as possible, even in the case where  $H_k \neq 0$ . In other words, we trust that the efficiency of the overall optimization algorithm may be improved if we take into account also the concavity region of  $f(x)$ , exploited by our planar-CG algorithm. We shall see in Section 4 that this idea has a positive effect when the iterate  $x_k$  is still far from the solution point  $x^*$ , i.e., when the Hessian matrix  $H_k$  may be indefinite.

Observe that, for the indefiniteness of  $H_k$ , we are not ensured that, at steps CG- $i_A$  and PL- $i_B$ , the directions  $p_i$  and  $q_i$  are descent directions. Hence, it may happen that the current approximation of the Newton direction  $d_i$  is not a descent too. This motivates the choice of the alternative direction  $\bar{d}_i$ : it represents a compromise, between the necessity of resembling as much as possible the Newton direction (see also page 117 of Ref. 20) and the requirement on  $d_k$  to be a gradient-related direction. In particular, consider the expression for updating the vector  $\bar{d}_i$  in a planar iteration,

$$\bar{d}_i = \bar{d}_{i-2} + (p_{i-2}^T r_{i-2} / \|H_k p_{i-2}\|^2) p_{i-2} + (q_{i-2}^T r_{i-2} / \|H_k q_{i-2}\|^2) q_{i-2}. \quad (11)$$

The quantities  $p_{i-2}^T r_{i-2} / \|H_k p_{i-2}\|^2$  and  $q_{i-2}^T r_{i-2} / \|H_k q_{i-2}\|^2$  are chosen with a twin purpose: on the one hand, they resemble as much as possible the structure of the coefficient  $\alpha_{i-2}$  at iteration CG- $(i-2)_A$ ; i.e., the relation (11) aims at formally altering as less as possible a double CG iteration. On the other hand, they are designed in such a way that relations (7) and (9) hold; i.e.,  $\bar{d}_i$  is gradient related whenever  $\bar{d}_i \neq 0$ . Finally, notice that, in the case where Algorithm TNFLR performs only CG iterations, with the quantity  $p_i^T H_k p_i$  positive, then  $d_i \equiv \bar{d}_i$ .

Moreover, due to the fast convergence of the pure Newton method, the decision rule at Step 3 of Algorithm TNFLR is aimed at using as much as possible the Newton direction  $d_i$ , as long as  $d_i$  is gradient related.

Nevertheless, observe that, from Section 2.1, a different algorithm may be considered, where the direction  $\bar{d}_i$  is always taken in the selection strategy at Step 3.

The following result summarizes the convergence properties of the algorithm which uses the scheme TNFLR of Table 3. We remark that Algorithm TNFLR considers only the first relation (3); i.e. we set  $s_k = 0$  in relation (3). We are still investigating the possibility of defining a truncated Newton scheme which includes both the directions  $d_k$  and  $s_k$ .

**Theorem 3.1.** Consider the problem (2), where  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  and  $f \in \mathcal{C}^2(\mathbb{R}^n)$ . Let  $x_i \in \mathbb{R}^n$  such that the level set  $\mathcal{L}_1 = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$  is compact; consider the truncated Newton method in Table 1, where  $d_k$  is computed by means of Algorithm TNFLR and  $\alpha_k$  at Step (d) is the step-length from the nonmonotone, Armijo-type line search in Ref. 8. Then, we have:

- (i) every limit point of  $\{x_k\}$  is a stationary point of  $f(x)$ ;
- (ii) every limit point of  $\{x_k\}$  cannot be a maximum of  $f(x)$ ;
- (iii) if  $\{x_k\}$  converges to  $x^*$ , where  $x^*$  is a stationary point with  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*) > 0$ , then  $\{x_k\}$  converges superlinearly.

**Proof.** This is a standard result, which follows from Refs. 8, 6. □

#### 4. Preliminary Numerical Results

A truncated Newton algorithm (see Table 1), which uses Algorithm TNFLR at Step (c), was applied for solving several large-scale optimization problems. In the general setting of this approach, we generate the new iterate

$$x_{k+1} = x_k + \alpha_k d_k \tag{12}$$

in such a way that  $\{x_k\} \rightarrow x^*$ , where  $x^*$  is solution of the problem

$$\min_{x \in \mathbb{R}^n} f(x), \quad f : \mathbb{R}^n \rightarrow \mathbb{R}. \tag{13}$$

The vector  $d_k$  (computed by means of Algorithm TNFLR in Table 3) is a Newton-type direction which solves approximately

$$\nabla^2 f(x_k) d + \nabla f(x_k) = 0. \tag{14}$$

Moreover,  $\alpha_k$  is the steplength provided by a suitable Armijo-type line search. In order to preserve the efficiency of the method, the general strategy of a truncated scheme reduces to the following: as long as  $x_k$  is far

from the stationary points  $x^*$ ,  $d_k$  may be even a coarse approximate solution of (14). Conversely, whenever  $x_k$  approaches  $x^*$ ,  $d_k$  is required to solve (14) more accurately and this implies a larger computational burden. We ensure the efficiency of the algorithm (i.e., the q-superlinear rate of convergence) by solving Eq. (14) according to the relation

$$\|\nabla^2 f(x_k)d_k + \nabla f(x_k)\| \leq \eta_k,$$

where in our implementation  $\eta_k$  has the expression (see Refs. 15, 13, 21)

$$\eta_k = 10^{-1} \min\{1, \|\nabla f(x_k)\| \max\{1/k, 1/\exp(10k/n)\}\} \|\nabla^2 f(x_k)\| \|d_k\|. \quad (15)$$

In this section, we compare the following choices:

- (C1)  $d_k$  is calculated at Step 3 of Algorithm TNFLR; i.e., in general  $d_k$  contains information on both the positive and the negative curvature directions generated by the iterative method TNFLR; see Tables 5 and 6 of Ref. 22.
- (C2)  $d_k$  is calculated by means of the routine SYMMLQ (Ref. 21); i.e.,  $d_k$  contains information on both the positive and the negative curvature directions generated by a Lanczos-CG iterative method; see Tables 7 and 8 of Ref. 22.

We specify that, in (C2), the SYMMLQ routine is used for approximately solving the Newton equation (14) and providing a provisional solution  $\tilde{d}_k$ . Then, the direction  $d_k$  of Step (c) of Algorithm TN in Table 1 is chosen according to

$$d_k = \begin{cases} \tilde{d}_k, & \text{if } \tilde{d}_k \text{ is gradient related to } \{x_k\}, \\ -D_k \nabla f(x_k), & \text{otherwise } (D_k > 0). \end{cases} \quad (16)$$

Roughly speaking, the above two choices correspond to applying respectively Algorithm FLR and the SYMMLQ for solving equation (14), in order to provide a gradient-related direction to the scheme TN. In our test, we set  $D_k = I$  when we applied the SYMMLQ routine. However, other possible choices might be considered, which take into account the knowledge of the Hessian matrix.

The resulting method was tested over several problems from the CUTE collection (Ref. 2); the results are reported in Tables 5–8 of Ref. 22. In this preliminary setting, we adopted the efficient nonmonotone stabilization technique in Ref. 23, which was used successfully in Ref. 13 too with the SYMMLQ routine.

We considered 87 test problems from the CUTE collection, both convex and nonconvex; over two problems (BROYDN7D-1000 and NONCVXU2-1000), the two choices of  $d_k$  investigated converged to different points. For

the remaining 85 problems, Table 4 summarizes the comparative performances of the two choices for  $d_k$ , in terms of function evaluations and inner iterations (the costly parameters in a truncated Newton method). In particular, we report in two columns the successes for each of the two choices (not including the failures of the other algorithm); as a consequence, the results in any column may not add up to 85 problems. This is because, for some test problems, once a parameter is considered (either function evaluations or inner iterations), no algorithm is preferable to the others, with respect to that parameter.

As expected, the SYMMLQ performed quite well over convex problems; in particular, the Newton-type direction that it provides is well scaled (very few function evaluations). On this stream, we remark that in general the choice of the forcing term  $\eta_k$  is crucial for the overall performances of the optimization method. Therefore, to stress the performance of Algorithm TNFLR in a challenging setting, we chose the forcing term (15), which was designed specifically in Refs. 13, 21 for the SYMMLQ method.

Unfortunately, over the test problems that we considered, Algorithm TNFLR performed the planar Step PL- $i_B$  very rarely (see Pla in Tables 5, 6 of Ref. 22). Nevertheless, the possibility of performing a planar iteration, in case at Step 2 the quantity  $p_i^T H_k p_i$  is too small, ensures that our algorithm does not stop beforehand over a class of specific problems. On the other hand, for nonconvex problems, there was a contribution of the negative curvature to the final Newton-type direction  $d_k$ . To sum up, considering the number of successes and failures in the previous table, the choice (C1) of calculating the Newton-type direction  $d_k$  seems in general preferable for nonconvex problems, with respect to the choice (C2). However, numerical experimentation over a larger number of problems, including real applications, seems necessary to further emphasize the importance of our approach.

Observe that, when we consider the choice (C2) (SYMMLQ routine), over nonconvex problems, resorting to the use of  $-\nabla f(x_k)$  according to (16) turns out to be harmful. This may be interpreted in the following way: when the iterate  $x_k$  is far from the solution points  $x^*$  and we are in a region of nonconvexity for  $f(x)$ , then the choice (C2) for  $d_k$  seems to partially ignore the local knowledge on  $f(x)$ . This leads to slow convergence of the overall algorithm.

Within Algorithm TNFLR, the planar step PL- $i_B$  was performed when the quantity  $|p_i^T H_k p_i|$  was too small, as in the relation

$$|p_i^T H_k p_i| \leq 0.5 \cdot 10^{-6} \min\{\|p_i\|^2, 1\},$$

Table 4. Comparative results over 85 test problems.

Choice of $d_k$	Convex problems			Nonconvex problems		
	Successes		Failures	Successes		Failures
	Function evaluations	Inner iterations		Function evaluations	Inner iterations	
(C1)	7	24	0	11	10	2
(C2)	21	19	0	5	6	13

while we adopted for the overall method the strong stopping criterion  $\|\nabla f(x_k)\| \leq 10^{-5}$ . Finally we remark that the quantity  $\|\nabla^2 f(x_k)\|$  in relation (15) is evaluated approximately by calculating iteratively the norm of the matrix  $T_k$ , which is a tridiagonalization of  $\nabla^2 f(x_k)$ . For the calculation, we used IBM RISC System 6000; a Fortran 90 code was developed and double precision was considered for the variables.

## 5. Conclusions

Considering the results summarized in Table 4 of this paper and Table 2 in Ref. 1, we can conclude that, at the moment, the proposed Algorithm FLR is expected to be fruitfully employed in optimization frameworks, rather than as solver of indefinite linear systems. Indeed in preliminary numerical experience (not reported here), the SYMMLQ routine seems to be more stable as solver of linear systems, though it is slightly more expensive than our algorithm. However, there is not a clear evidence that a more accurate implementation of our method cannot be competitive.

On the other hand, the scheme in Table 4 reveals that our algorithm seems to be more efficient in using the negative curvature directions of the current Hessian matrix  $H_k$  in optimization frameworks. This feature needs further investigation. Indeed, in case we want to adopt the iterative scheme reported in relation (3b), then an effective negative curvature direction  $s_k$  has to be computed. We recall that, in large-scale optimization, the calculation of  $s_k$  is indispensable whenever the algorithm is asked to converge to second-order points (Refs. 14, 12), i.e., those minimum points which satisfy the second-order necessary conditions of optimality.

In the near future, we are going to test our algorithm over real problems, where the nonconvexity of the objective function forces the algorithm to perform a significant number of planar steps. Furthermore, the use of a suitable preconditioner is another relevant issue when dealing with large-scale problems. Though the introduction of a preconditioner at

Step CG- $i_A$  of Algorithm FLR simply yields a preconditioned CG, the use of a preconditioner within the planar Step PL- $i_B$  is not completely clear.

## References

1. FASANO, G., *Planar Conjugate Gradient Algorithm for Large-Scale Unconstrained Optimization, Part 1: Theory*, Journal of Optimization Theory and Applications, Vol. 125, pp. 523–541, 2005.
2. BONGARTZ, I., CONN, A. R., GOULD, N., and TOINT, P. L., *CUTE: Constrained and Unconstrained Test Environment*, ACM Transactions on Mathematical Software, Vol. 21, pp. 123–160, 1995.
3. AXELLSON, O., *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1996.
4. BERTSEKAS, D. P., *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts, 1995.
5. McCORMICK, G. P., *Nonlinear Programming: Theory, Algorithm, and Applications*, Wiley and Sons, New York, NY, 1983.
6. GRIPPO, L., LAMPARIELLO, F., and LUCIDI, S., *A Truncated Newton Method with Nonmonotone Line Search for Unconstrained Optimization*, Journal of Optimization Theory and Applications, Vol. 60, pp. 401–419, 1989.
7. DEMBO, R. S., and STEIHAUG, T., *Truncated Newton Algorithms for Large-Scale Matrix Methods*, Mathematical Programming, Vol. 26, pp. 190–212, 1983.
8. GRIPPO, L., LAMPARIELLO, F., and LUCIDI, S., *A Nonmonotone Line Search Technique for Newton's Method*, SIAM Journal on Numerical Analysis, Vol. 23, pp. 707–716, 1986.
9. DEMBO, R. S., EISENSTAT, S. C., and STEIHAUG, T., *Inexact Newton Methods*, SIAM Journal on Numerical Analysis, Vol. 19, pp. 400–408, 1982.
10. NASH, S. G., *A Survey of Truncated Newton Methods*, Journal of Computational and Applied Mathematics, Vol. 124, pp. 45–59, 2000.
11. LUCIDI, S., and ROMA, M., *Numerical Experiences with Truncated Newton Methods in Large-Scale Unconstrained Optimization*, Computational Optimization and Applications, Vol. 7, pp. 71–87, 1997.
12. MORE', J. J., and SORENSEN, D. C., *On the Use of Directions of Negative Curvature in a Modified Newton Method*, Mathematical Programming, Vol. 16, pp. 1–20, 1979.
13. LUCIDI, S., ROCHETICH, F., and ROMA, M., *Curvilinear Stabilization Techniques for Truncated Newton Method in Large-Scale Unconstrained Optimization*, SIAM Journal on Optimization, Vol. 8, pp. 916–939, 1999.
14. GOULD, N. I. M., LUCIDI, S., ROMA, M., and TOINT, P. L., *Exploiting Negative Curvature Directions in Line Search Methods for Unconstrained Optimization*, Optimization Methods and Software, Vol. 14, pp. 75–98, 2000.
15. EISENSTAT, S. C., and WALKER, H. F., *Choosing the Forcing Terms in an Inexact Newton Method*, Technical Report 94463, Center for Research on Parallel Computation, Rice University, Houston, Texas, 1994.

16. HESTENES, M. R., *Conjugate Direction Methods in Optimization*, Springer Verlag, New York, NY, 1980.
17. LUENBERGER, D. G., *Hyperbolic Pairs in the Method of Conjugate Gradients*, SIAM Journal on Applied Mathematics, Vol. 17, pp. 1263–1267, 1969.
18. FASANO, G., *Use of Conjugate Directions Inside Newton-Type Algorithms for Large-Scale Unconstrained Optimization*, PhD Dissertation, Rome, Italy, 2001.
19. SRINIVASAN, M., *Using Directions of Negative Curvature in Newton-Type Methods for Nonlinear Nonconvex Problems*, PhD Thesis, School of Information Technology and Engineering, George Mason University, Fairfax, Virginia, 1994.
20. DENNIS, J. E., and SCHNABEL, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
21. PAIGE, C. C., and SAUNDERS, M. A., *Solution of Sparse Indefinite Systems of Linear Equations*, SIAM Journal on Numerical Analysis, Vol. 12, pp. 617–629, 1975.
22. FASANO, G., *Planar Conjugate-Gradient Algorithm for Large-Scale Unconstrained Optimization Part 2: Applications*, Technical Report 2004-016, Istituto Nazionale per Studi ed Esperienze di Architettura Navale (INSEAN), Rome, Italy, 2004.
23. GRIPPO, L., LAMPARIELLO, F., and LUCIDI, S., *A Class of Nonmonotone Stabilization Methods in Unconstrained Optimization*, Numerische Mathematik, Vol. 59, pp. 779–805, 1991.