Optimization Methods and Software
Vol. 19, Nos. 3–4, June–August 2004, pp. 267–290

CONJUGATE GRADIENT (CG)-TYPE METHOD FOR THE SOLUTION OF NEWTON'S EQUATION WITHIN OPTIMIZATION FRAMEWORKS*

GIOVANNI FASANO[a,b,†]

[a]*Dipartimento di Informatica e Sistemistica 'A. Ruberti', Università di Roma 'La Sapienza', Rome, Italy;* [b]*Istituto di Analisi dei Sistemi ed Informatica 'A. Ruberti', CNR, Rome, Italy*

(Received 7 February 2003; Revised 5 February 2004; In final form 19 February 2004)

A conjugate gradient (CG)-type algorithm CG_Plan is introduced for calculating an approximate solution of Newton's equation within large-scale optimization frameworks. The approximate solution must satisfy suitable properties to ensure global convergence. In practice, the CG algorithm is widely used, but it is not suitable when the Hessian matrix is indefinite, as it can stop prematurely. CG_Plan is a symmetric variant of the composite step Bi-CG method of Bank and Chan, suitably adapted for optimization problems. It is an alternative to CG that copes with the indefinite case.

We show convergence for CG_Plan, then prove that the practical implementation always provides a gradient related direction within a truncated Newton method (algorithm TN_Plan). Some preliminary numerical results support the theory.

Keywords: Large-scale unconstrained optimization; Newton's equation; Conjugate Gradient method; Krylov subspace methods

AMS Subject Classification: 90C30

1 INTRODUCTION

This article deals with the definition of a conjugate gradient (CG)-based algorithm for the iterative solution of large-scale *indefinite* linear systems that arise in optimization schemes. In particular, we consider the optimization problem

$$\min_{y \in \mathbb{R}^n} f(y), \tag{1}$$

and the solution of the associated Newton equation

$$\nabla^2 f(y_h) d = -\nabla f(y_h), \tag{2}$$

---

* This work was published by MIUR, FIRB Research Program *Large Scale Nonlinear Optimization*, Rome, Italy

† Corresponding author. Tel.: +39-06-48299223; Fax: +39 06 4782 5618; E-mail: giovanni.fasano@dis_uniroma1.it

ISSN 1055-6788 print; ISSN 1029-4937 online
DOI: 10.1080/10556780410001689234

where $\nabla^2 f(y) \in \mathbb{R}^{n\times n}$ is the *symmetric* and *indefinite* Hessian matrix of the twice continuously differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}$, $\nabla f(y) \in \mathbb{R}^n$ is the gradient of $f(y)$, and n is *large*. The use of proper techniques for the solution of large-scale linear system (2) within optimization frameworks is indispensable. Now, we briefly examine the motivations of this conclusion.

Suppose we want to solve (1) with an iterative algorithm that generates a sequence of iterates $\{y_h\}$ according to

$$y_{h+1} = y_h + d_h. \tag{3}$$

For efficiency, Newton's method may be the method of choice: it starts from the guess y_h and at step h in place of Eq. (1), we tackle the problem

$$\min_{d\in\mathbb{R}^n} \frac{1}{2} d^{\mathrm{T}} \nabla^2 f(y_h)^{\mathrm{T}} d + \nabla f(y_h)^{\mathrm{T}} d + f(y_h). \tag{4}$$

If $\nabla^2 f(y_h)$ is *nonsingular*, the stationary point of Eq. (4) is the solution of Eq. (2). Of course, in general, we would like to retain the good convergence rate of Newton's method [3]. However, with large-scale problems, the exact solution of Eq. (2) could require excessive computation when the current iterate y_h is far from the solution y^*. In the latter case, it may be convenient to calculate an approximate solution $\tilde{d}_h$ of Newton's equation (2), and modify the iteration (3) to be

$$y_{h+1} = y_h + \alpha_h \tilde{d}_h, \tag{5}$$

where $\alpha_h \in \mathbb{R}$ is chosen by means of a suitable linesearch technique. We emphasize that the introduction of the stepsize α_h is essential to guarantee global convergence of the overall optimization algorithm. The truncated solution of Newton's equation (2), when n is large, implies some considerations.

- If the Hessian matrix $\nabla^2 f(y_h)$ is indefinite, the approximate solution $\tilde{d}_h$ of Eq. (2) may not be a descent direction for $f(y)$ at y_h; thus, the optimization method should carefully take into consideration the local information on $f(y)$, provided by $\tilde{d}_h$. In other words, the linesearch technique, we adopt for calculating α_h in Eq. (5), always requires specific properties on $\tilde{d}_h$. Consequently, the iterative method used for solving Eq. (2) must be properly chosen.
- The truncated solution $\tilde{d}_h$ of Eq. (2) may not satisfy an *angle condition* with the gradient $\nabla f(y_h)$, i.e., $\tilde{d}_h$ may not be a so-called *gradient related* direction [3]. We analyze this case in Section 5, where we focus on the importance of the latter property in an optimization framework. We remark that this issue specifically arises whenever we deal with optimization problems; it may be irrelevant when we solve a general linear system that is not the Newton equation.

Many algorithms have been proposed in the literature for the solution of Eq. (2). In particular, it is well known that for large-scale systems the use of iterative methods often reduces the total computation in comparison with direct methods. For the sake of simplicity, hereafter we adopt the notation A for $\nabla^2 f(y_h)$, x for d, and b for $\nabla f(y_h)$. Thus, Eq. (2) becomes

$$Ax = b \quad x \in \mathbb{R}^n. \tag{6}$$

Among the wide set of iterative methods for the solution of Eq. (6) [see Refs. 1,23,26], there are appealing *Krylov subspace methods* with short recurrences, which perform quite efficiently on large-scale problems. These methods start from the initial guess x_1 with the residual $r_1 = b - Ax_1$ and generate a sequence of iterates $\{x_k\}$ with the property that

$$x_k \in x_1 + \text{span}\{r_1, Ar_1, \ldots, A^{k-1}r_1\} \doteq x_1 + \mathcal{K}_k(r_1, A), \tag{7}$$

where $\mathcal{K}_k(r_1, A)$ is the Krylov subspace of order k associated with the pair (r_1, A). Some Krylov methods consider at step k also the subspaces $\mathcal{K}_k(r_1, A^{\mathrm{T}})$ or $\mathcal{K}_k(r_1, AA^{\mathrm{T}})$, depending on the nature of the linear system (2). Krylov subspace methods are commonly considered useful tools for linear algebra and within large-scale optimization frameworks, provided that eventually $k \ll n$, and a limited number of residuals from previous iterates are stored during the computation. A *restarting procedure* may be imposed (truncated iterative methods) in order to avoid excessive storage [see for instance Ref. 7]. In this article, we introduce and analyze a three-term recurrence scheme in the class of Krylov methods. Our method is mainly suitable for solving Newton's equation (2) in optimization frameworks, as its practical implementation always provides an approximate solution $\tilde{d}_h$ of Eq. (2), which is also gradient related.

Now, we briefly survey some related iterative methods. Even though all these methods (including our proposal) often work efficiently when a proper preconditioner is adopted, in the present article we avoid introducing this issue because it deserves specific attention.

When A is *positive definite*, the CG algorithm originally proposed by Hestenes and Stiefel [18] is a simple and appealing Krylov method. It is often an efficient method for solving Eq. (6). A very general CG kth iteration is the following:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \quad \alpha_k = \frac{r_k^{\mathrm{T}} p_k}{p_k^{\mathrm{T}} A p_k} \\ r_{k+1} &= r_k - \alpha_k A p_k \\ p_{k+1} &= r_{k+1} + \beta_k p_k \quad \beta_k = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}. \end{aligned} \tag{8}$$

However, many variants for CG have been proposed [see Refs. 24,27], aimed at improving the features of stability and accuracy.

When the symmetric matrix A is indefinite, CG can be unstable and may prematurely stop because the quantity $p_k^{\mathrm{T}} A p_k$ in coefficient α_k may approach zero. In place of CG, some specific Krylov schemes can be adopted [see Refs. 9,24,25]. The Lanczos expansion was successfully used within algorithms for solving Eq. (2) [14,15,19,21], even though it does not always provide solutions that are immediately useful in optimization schemes (see Secs. 5 and 6).

We are interested in describing a Krylov algorithm for the solution of indefinite Newton's equation (2). Our interest is twofold: as a linear solver we claim that this algorithm retains the satisfactory efficiency and accuracy of CG; moreover, despite the linear solvers cited above, it may guarantee the properties of the approximate solution $\tilde{d}_h$ that are required in the optimization framework. In particular, we considered the methods in Refs. [10,17,20], which have been addressed as *planar methods*. These algorithms inherit the structure of CG and aim to overcome its possible breakdown on *indefinite systems*. Indeed, they substantially coincide with CG as long as the quantity $|p_k^{\mathrm{T}} A p_k|$ (see coefficient α_k in Eq. (8)) is bounded away from zero. (In Ref. [17], a slightly different condition is adopted.) If the test fails, the kth CG step cannot be performed. A second direction p_{k+1} is then generated in such a way that the search for the solution of Eq. (6) over the linear manifold $x_k + \alpha p_k$, $\alpha \in \mathbb{R}$ (CG-step) is replaced by the search on the two-dimensional manifold (planar-step)

$$x_k + \text{span}\{p_k, p_{k+1}\}. \tag{9}$$

The scheme proposed by Hestenes [17] is more general and our numerical experience reveals that it is sometimes more precise than the algorithm of Luenberger [20] for solving an indefinite linear system; however, the computational complexity of Luenberger's method is preferable. Therefore, the latter method may be adopted, for instance when high precision is not required for x. In particular, at step k both the algorithms in Refs. [17,20] perform a composite step if a suitable test is not fulfilled. The test in Ref. [17] is more complex and involves the quantities p_k, p_{k+1}, Ap_k and Ap_{k+1}, while the test in Ref. [20] simply checks when $|p_k^{\mathrm{T}} Ap_k| < \varepsilon, k < n$, and ε is 'small'. The computational experience shows that a possible numerical instability may arise in Luenberger's method because of finite precision [13]. The latter aspect will be investigated in the next sections. Moreover, we shall see that by means of the CG-type method we propose, the numerical errors are not eliminated but they may be suitably bounded.

The algorithm in this article is specifically designed for solving the Newton equation (2) in optimization frameworks, while preserving the low computational cost of CG-like algorithms. Furthermore, a numerical comparison with SYMMLQ [22] (see Sec. 6.1) reveals that our algorithm may perform not poorly even on a general set of indefinite problems, where CG may be unstable.

In the sequel, we use the symbol $\|\cdot\|$ to denote the Euclidean norm for both a real n-dimensional vector and a real $n \times n$ matrix. Moreover, we adopt the symbol '$\doteq$' to indicate *by definition*. Finally, we introduce the notation $\lambda_M = \lambda_M(A) = \max_{1\le j\le n}\{|\lambda_j(A)|\}$ and $\lambda_m = \lambda_m(A) = \min_{1\le j\le n}\{|\lambda_j(A)|\}$, where $\lambda_j(A)$ is the jth eigenvalue of matrix $A \in \mathbb{R}^{n\times n}$.

In Section 2, we introduce our CG-like algorithm named CG_Plan, whose convergence and complexity properties are examined in Section 3. A full analysis of convergence is given. Sections 4 and 5 describe TN_Plan, the practical implementation of CG_Plan within optimization frameworks. Section 6 provides some preliminary numerical results. The Conclusions and Appendix complete the article.

2 PLANAR ALGORITHM: PRELIMINARIES

We are now concerned with describing and developing our scheme. On one hand, it approximately solves Newton's equation (2). On the other hand, we claim that it may ensure the properties necessary for global convergence of the overall optimization problem. We outline the algorithm CG_Plan where, for the sake of simplicity, we consider the solution of Eq. (6) in place of Eq. (2). In addition, we assume that the matrix A in Eq. (6) is nonsingular.

Algorithm CG_Plan

step 1:

$k = 1, x_1 \in \mathbb{R}^n,\ r_1 = b - A\,x_1,\ p_1 = r_1.$

step k **:**

If $\|r_k\| = 0$ **Then** STOP **Else**

Set $c_k = Ap_k,\ \delta_k = c_k^{\mathrm{T}} p_k$

If $\delta_k = 0$ **Then** go to **step** k_B **Else** go to **step** k_A

step k_A **(CG-step):**

$$x_{k+1} = x_k + \alpha_k p_k, \qquad \rho_k = r_k^{\mathrm{T}} p_k, \quad \alpha_k = \frac{\rho_k}{\delta_k} = \frac{r_1^{\mathrm{T}} p_k}{\delta_k}$$

$$r_{k+1} = r_k - \alpha_k c_k$$

$$p_{k+1} = r_{k+1} + \beta_k p_k, \qquad \beta_k = -\frac{r_{k+1}^{\mathrm{T}} c_k}{\delta_k} = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$$

$k \leftarrow k + 1$ repeat **step** k

step k_B (Planar-step):

$$p_{k+1} = \gamma_k\, c_k, \qquad \gamma_k \in \mathbb{R}$$
$$a_{k+1} = A p_{k+1}, \qquad \omega_{k+1} = a_{k+1}^{\mathrm{T}} p_{k+1}$$
$$x_{k+2} = x_k + \alpha_k p_k + \alpha_{k+1} p_{k+1}, \qquad \alpha_k = -\frac{\rho_k}{\gamma_k^2 \,\| c_k\|^4}\,\omega_{k+1}$$
$$r_{k+2} = r_k - \alpha_k c_k - \alpha_{k+1} a_{k+1}, \qquad \alpha_{k+1} = \frac{\rho_k}{\gamma_k \,\| c_k\|^2}$$
$$p_{k+2} = r_{k+2} + \sigma_k p_k, \qquad \sigma_k = -\frac{r_{k+2}^T a_{k+1}}{\gamma_k \,\| c_k\|^2}$$
$k \leftarrow k+2$ repeat **step** k

First observe that when δ_k at step k is not zero, then step k_A of algorithm CG_Plan reduces exactly to CG, i.e., the steplength α_k is chosen along the direction p_k.

On the other hand, if $\delta_k = 0$, then at step k_B we generate a vector p_{k+1}, which is *orthogonal* to the direction p_k (so that $\{p_k, p_{k+1}\}$ is an independent set), and we determine coefficients α_k and α_{k+1} such that the solution of Eq. (6) is determined on the two-dimensional manifold (9). We remark that the choice of the direction p_{k+1} may be inferred from Bank and Chan, in Section 3 of [2], where the problem we are considering is extensively studied in the unsymmetric case. Further, general references about the idea we adopt here are in Refs. [5,6].

In particular, suppose that algorithm CG_Plan stops when $k = n$. Observe that in case CG_Plan does not perform planar steps, i.e., it coincides with the standard CG, the following expression for A holds [13] (assuming exact arithmetic):

$$A = RTR^{\mathrm{T}}, \quad T = LDL^{\mathrm{T}},$$

where

$$R = \left(\frac{r_1}{\|r_1\|} \cdots \frac{r_n}{\|r_n\|}\right) \in \mathbb{R}^{n\times n}, \tag{10}$$

$$L = \begin{pmatrix} 1 & & & 0 \\ -\sqrt{\beta_1} & \vdots & & \\ & \vdots & \vdots & \\ 0 & & -\sqrt{\beta_{n-1}} & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}, \tag{11}$$

$$D = \operatorname{diag}\left(\frac{1}{\alpha_i}\right) \in \mathbb{R}^{n\times n}, \tag{12}$$

and T is tridiagonal. On the other hand, if CG_Plan does perform some planar steps, it can be proved [11] that the matrix D becomes *block* diagonal, with a 2×2 block corresponding to each planar step. Each 2×2 block prevents a *pivot breakdown* [2] when $\delta_k = 0$, i.e., a premature interruption of the CG process. Step k_B determines the new iterate x_{k+2} such that

$$x_{k+2} = x_k + \alpha_k p_k + \alpha_{k+1} p_{k+1}$$

and, equivalently,

$$r_{k+2} = r_k - \alpha_k A p_k - \alpha_{k+1} A p_{k+1}. \tag{13}$$

Now, observe that $\delta_k = 0$ straightforwardly implies $r_k^{\mathrm{T}} p_{k+1} = 0$. Thus, imposing the Galerkin conditions

$$r_{k+2}^{\mathrm{T}} p_k = 0 \quad \text{and} \quad r_{k+2}^{\mathrm{T}} p_{k+1} = 0, \tag{14}$$

from Eqs. (13) and (14) and the relation $r_k^{\mathrm{T}} p_{k+1} = 0$, we obtain the expressions for α_k and α_{k+1}:

$$\alpha_k = -\frac{\rho_k \omega_{k+1}}{\gamma_k^2 \|c_k\|^4}, \quad \alpha_{k+1} = \frac{\rho_k}{\gamma_k \|c_k\|^2}. \tag{15}$$

Similarly to Ref. [2], we now complete the definition of the planar step k_B in algorithm CG_Plan. As in the other Planar algorithms in Refs. [17,20], we calculate the direction

$$p_{k+2} = r_{k+2} + \sigma_k p_k + \sigma_{k+1} p_{k+1} \quad \sigma_k, \sigma_{k+1} \in \mathbb{R}, \tag{16}$$

where σ_k and σ_{k+1} are chosen in such a way that the conjugacy relations

$$p_{k+2}^{\mathrm{T}} A p_k = 0, \quad p_{k+2}^{\mathrm{T}} A p_{k+1} = 0 \tag{17}$$

hold. After a few calculations, considering Eq. (14) and relation $r_k^{\mathrm{T}} p_{k+1} = 0$, we obtain

$$\sigma_k = -\frac{r_{k+2}^{\mathrm{T}} a_{k+1}}{\gamma_k \|c_k\|^2}, \quad \sigma_{k+1} = 0. \tag{18}$$

Before analyzing the convergence properties of algorithm CG_Plan, we examine the choice of the scalar γ_k whenever $\delta_k = 0$. We propose for γ_k the following three choices:

(a) $\gamma_k = 1$,
(b) $\gamma_k = 1/\|c_k\|$,
(c) $\gamma_k = \|p_k\|/\|c_k\|$.

For all three possibilities we are able to prove convergence of algorithm CG_Plan, along with the features of conjugacy and orthogonality among the vectors it generates (see Sec. 3). Moreover, it is readily verified that the computational burden of step k is independent of the choice of γ_k, and the same iterates $\{x_k\}$ are generated. However, the parameter γ_k may affect the practical implementation (see Sec. 4). Indeed, our experience suggests the following conclusions:

- The exponent of the quantity $\|c_k\|$ in the denominator of α_k, α_{k+1}, and σ_k depends on the choice of γ_k.
- The choice $\gamma_k = 1/\|c_k\|$ may be interpreted as a *natural* scaling for vector p_{k+1} in step k_B.
- The choice $\gamma_k = \|p_k\|/\|c_k\|$ implies that the vector p_{k+1} in step k_B is simply rotated by an angle of $\pi/2$ radians with respect to p_k, i.e., $\|p_k\| = \|p_{k+1}\|$. In Section 4, we shall consider this choice for the practical implementation of algorithm CG_Plan, because of the stronger *theoretical* properties.

3 CONVERGENCE PROPERTIES

The following theorems summarize the convergence features of algorithm CG_Plan. In the sequel, we describe the direction p_{k+1} at step k_B as the '*fellow direction*' of p_k. Moreover, we shall refer to the '*singular direction*' p_k when $p_k^{\mathrm{T}} A p_k = 0$ (i.e., $\delta_k = 0$ in CG_Plan). Now, let us first highlight the following simple result.

THEOREM 3.1 *Let A be nonsingular and possibly indefinite. If at step $h \le n$ of algorithm CG_Plan we have $r_h \neq 0$, then the following relations hold* (*where* $1 \le j < k \le h \le n$):

(a) $r_h^{\mathrm{T}} p_j = 0$,
(b) $r_h^{\mathrm{T}} r_j = 0$,
(c) $p_k^{\mathrm{T}} A p_j \neq 0 \Longleftrightarrow p_j$ *is singular and* p_k *is the fellow direction of* p_j (i.e., $p_k = p_{j+1}$).

Proof The proof[1] is by induction on h. Let us consider (a); if either $h = 2$ (the first step is 1_A) or $h = 3$ (the first step is 1_B), then the statement (a) trivially holds because of the choice of α_1 or α_1, α_2. Thus, supposing (a) holds for index $h - 1$, let us prove it for the index h.

It must be shown that $r_h^{\mathrm{T}} p_j = 0, \forall j < h$; we distinguish two possibilities depending on the step where r_h is calculated:

- if r_h was calculated at step $(h-1)_A$, then a similar reasoning used for the standard CG yields $r_h^{\mathrm{T}} p_j = 0$, $j < h$.
- if r_h was calculated at step $(h-2)_B$, then

$$\begin{aligned} r_h^{\mathrm{T}} p_j &= (r_{h-2} - \alpha_{h-2} A p_{h-2} - \alpha_{h-1} A p_{h-1})^{\mathrm{T}} p_j \\ &= r_{h-2}^{\mathrm{T}} p_j - \alpha_{h-2} p_{h-2}^{\mathrm{T}} A p_j - \alpha_{h-1} p_{h-1}^{\mathrm{T}} A p_j \quad j < h, \end{aligned}$$

  and for $j < h - 2$, all three terms in the rightmost expression are zero (assuming complete induction holds). On the other hand, if either $j = h - 2$ or $j = h - 1$, then the last term on the right-hand side is zero for the choice of coefficients α_{h-2} and α_{h-1} (see Eqs. (14) and (15)).

As regards points (b) and (c), the proof follows directly from Theorem 4.4 in Ref. [2]. ■

The following conclusion summarizes one convergence property of algorithm CG_Plan.

THEOREM 3.2 *Let A be nonsingular and possibly indefinite. Algorithm CG_Plan determines the solution of problem* (6) *in at most n steps.*

Proof The result is inferred from Ref. [2]. ■

Now, let us point out some further properties of algorithm CG_Plan, related to the other planar schemes in the literature:

(1) In a practical implementation of algorithm CG_Plan, when at step k the quantity $|\delta_k|$ is 'numerically small' though not zero, the step k_B is performed. (The same unavoidable numerical shortcoming occurs in Ref. [20] too, see Secs. 4 and 5.) However, this alters only partially the conjugacy properties in the set $\{p_1, \ldots, p_{k-1}, p_k, p_{k+1}\}$. Indeed, p_{k+1} at step k_B is orthogonal to the directions $p_1, \ldots, p_{k-1}$, regardless of the value of δ_k; furthermore, even though $\delta_k \neq 0$, the direction p_{k+1} is still conjugate to the directions $p_1, \ldots, p_{k-2}$ (see Theorem 3.1).
(2) Step k_B is always *well posed*, inasmuch as the quantity $\|c_k\|$ may approach zero *if and only if A* is nearly singular.
(3) Unlike in the Hestenes algorithm [17] and Luenberger's method [20], the denominators at steps k_A and k_B of algorithm CG_Plan can be computed without any additional computational burden: this allows numerical comparisons in advance, in order to detect potential instability at each step.

[1] We remark that if $p_{j+1}^{\mathrm{T}} A p_{j+1} = 0$ (i.e., the fellow direction p_{j+1} is singular too), the statement of the present theorem still holds, and in algorithm CG_Plan we simply have $\alpha_j = 0$. Moreover in the latter case, the direction p_{j+2} is *conjugate* to the manifold span$\{p_j, p_{j+1}\}$.

(4) We can readily verify that the proposed algorithm has a slightly lower computational complexity than the algorithms in Refs. [17,20]. Indeed, the planar step k_B in the latter references requires, respectively, two and one additional inner products, with respect to step k_B of CG_Plan.
(5) Since the planar methods cope with indefinite matrices, they may be fruitfully considered also for generating *negative curvature* directions, in the context of nonlinear optimization.

4 PRACTICAL IMPLEMENTATION OF ALGORITHM CG_PLAN

In this section and the next, we investigate the practical implementation of algorithm CG_Plan within optimization frameworks. We study the case when at step k the test $p_k^{\mathrm{T}} A p_k = 0$ is replaced by the more reliable test [see Refs. 13,16]

$$|p_k^{\mathrm{T}} A p_k| \leq \varepsilon_k \|p_k\|^2 \quad \varepsilon_k > 0 \text{ `small,'} \tag{19}$$

and even though step k_A is *theoretically* allowed, it might not be numerically advisable. Equivalently, we have to assess parameter ε_k properly in Eq. (19) (still an unsolved question in Ref. [20]).

We remarked in the previous item 1 that if $p_k^{\mathrm{T}} A p_k \neq 0$, then

$$\begin{aligned} p_{k+1}^{\mathrm{T}} A p_i &= 0, \quad i = 1, \ldots, k-2 \\ p_{k+1}^{\mathrm{T}} A p_{k-1} &\neq 0, \end{aligned} \tag{20}$$

i.e., the conjugacy between direction p_{k+1} in step k_B and direction p_{k-1} can slightly fail.

Here, we calculate a bound for the quantity $|p_{k+1}^{\mathrm{T}} A p_{k-1}|$ whenever the latter situation occurs. More specifically, we prove that if we apply algorithm CG_Plan with $\gamma_k = \|p_k\|/\|c_k\|$ and with the practical test (19) at step k, then $p_{k+1}^{\mathrm{T}} A p_{k-1}$ approaches zero whenever r_k approaches zero too. This ensures that, if we adopt the practical test (19) in place of the test $p_k^{\mathrm{T}} A p_k = 0$, the closer we are to the solution of Eq. (6), the smaller is the conjugacy error $p_{k+1}^{\mathrm{T}} A p_{k-1}$. The following theorem summarizes the latter result.

THEOREM 4.1 *Consider algorithm CG_Plan where A is nonsingular and possibly indefinite. Replace the test* $p_k^{\mathrm{T}} A p_k = 0$ *at step k with the test* (19). *If we choose* $\gamma_i = \|p_i\|/\|Ap_i\|$, $i \leq k$, *and*[2]

$$\varepsilon_i \leq \frac{\lambda_m}{2} \min \left\{ \left(\frac{\lambda_m}{\lambda_M} \right)^3, 2^{1/2} \frac{\|r_i\|}{\|p_i\|} \right\}, \quad i \leq k, \tag{21}$$

then at step k_B, *the conjugacy error* $p_{k+1}^{\mathrm{T}} A p_{k-1}$ *is bounded as follows*:

$$|p_{k+1}^{\mathrm{T}} A p_{k-1}| \leq \begin{cases} \rho 1_k \|r_k\|^2 & \text{if step } k_B \text{was preceded by step } (k-1)_A \\ \rho 2_k \|r_k\|^2 & \text{if step } k_B \text{was preceded by step } (k-2)_B, \end{cases} \tag{22}$$

where $\rho 1_k$ *and* $\rho 2_k$ *are bounded positive constants.*

Proof See Appendix. ■

[2] Observe that $\|r_i\|/\|p_i\| \leq 1$, $i \leq k$.

We conclude this section by remarking that relation (21) provides only a theoretical criterion for the choice of parameter ε_i; indeed, $\rho 1_k$ and $\rho 2_k$ in Eq. (22) are unrealistic high to be used in practice (see Appendix). Of course, several other criteria may be adopted for estimating ε_i at step i. Nevertheless, our choice for ε_i suggests that algorithm CG_Plan is theoretically more accurate when the iterates are close to the solution of Eq. (6), i.e., when $\|r_i\| \to 0$. Indeed, in this way, the conjugacy error $|p_{k+1}^{\mathrm{T}} A p_{k-1}|$ at step k_B can be bounded as in Eq. (22). Observe that often the quantity $\|r_i\|$ decreases with the iterations; moreover, even though algorithm CG_Plan converges slowly, relation (34) in Appendix indicates that small values of ε_k *force* a bound on the quantity $|p_{k+1}^{\mathrm{T}} A p_{k-1}|$.

5 APPLICATION OF ALGORITHM CG_PLAN WITHIN OPTIMIZATION

We already recalled that the use of iterative methods within large-scale optimization frameworks is often advisable for solving linear systems, especially for the Newton equation (2). However, the approximate solution $\tilde{d}_h$ of Eq. (2) is required to have certain additional properties within the optimization framework, in order to preserve the global convergence of the optimization method in hand. In particular, in a linesearch approach, the approximate solution $\tilde{d}_h$ of Eq. (2) may need to be modified in order to be a gradient-related direction [3]. That is, $\tilde{d}_h$ must satisfy the relations

$$\begin{aligned} \tilde{d}_h^{\mathrm{T}} \nabla f(y_h) &\le -q_1 \|\nabla f(y_h)\|^{h_1} \quad q_1, h_1 > 0 \\ \|\tilde{d}_h\| &\le q_2 \|\nabla f(y_h)\|^{h_2} \quad q_2, h_2 > 0. \end{aligned} \tag{23}$$

As before, let A denote the Hessian matrix $\nabla^2 f(y_h)$. We consider the practical implementation of algorithm CG_Plan, where the test on $p_k^{\mathrm{T}} A p_k$ at step k is based on the results in Refs. [8,16] and Section 4. In particular in Ref. [8], the authors proposed to solve Eq. (2) approximately by means of CG, where the CG iterations are stopped if $p_k^{\mathrm{T}} A p_k \le \varepsilon_k \|p_k\|^2$. This implies that only the positive curvature directions of A contributed to form $\tilde{d}_h$. In Ref. [16], the test had the more general form $|p_k^{\mathrm{T}} A p_k| \le \varepsilon_k \|p_k\|^2$, in order to retain also the strong contribution of negative curvature of A to the direction $\tilde{d}_h$. The latter strategy, which properly considers the general valuable role of negative curvatures within optimization, suggests that the test $p_k^{\mathrm{T}} A p_k = 0$ at step k of CG_Plan could be replaced by the practical test [see also Ref. 20]:

$$|p_k^{\mathrm{T}} A p_k| \le \varepsilon_k \|p_k\|^2 \quad \varepsilon_k > 0. \tag{24}$$

Of course, this implies that the theoretical properties proved in Section 3 for CG_Plan hold as described in item 1 of page 7, i.e., relations (20) hold. In addition, the proposed algorithm with the test (24) may be suitable for the solution of Newton's equation in optimization frameworks. Indeed, consider Eq. (2) and suppose y_h is the current iterate and $\tilde{d}_h$ the approximate solution of Eq. (2). We will prove that the proper application of algorithm CG_Plan with test (24) at step k can always provide a sequence of gradient related directions $\{\tilde{d}_h\}$ to the stabilization scheme of an optimization framework.

For this purpose, consider the algorithm CG_Plan and the direction $\tilde{d}_h \in \mathbb{R}^n$:

$$\tilde{d}_h = d^{PN} + d^{\mathrm{Pla}}, \tag{25}$$

where (see the new test (24) for step k of CG_Plan)[3]

$$\begin{aligned} d^{PN} &= \sum_{k \in I^P \cup I^N} \mathrm{sgn}[\delta_k] \alpha_k p_k \\ d^{\mathrm{Pla}} &= - \sum_{k \in I^{\mathrm{Pla}}} \mathrm{sgn}[\omega_{k+1}] \alpha_k p_k \end{aligned} \tag{26}$$

$$\begin{aligned} I^P &= \{k \geq 1 \colon k = k_A,\, p_k^{\mathrm{T}} A p_k \geq \varepsilon_k \|p_k\|^2\} \\ I^N &= \{k \geq 1 \colon k = k_A,\, p_k^{\mathrm{T}} A p_k \leq -\varepsilon_k \|p_k\|^2\} \\ I^{\mathrm{Pla}} &= \{k \geq 1 \colon k = k_B,\, |p_k^{\mathrm{T}} A p_k| < \varepsilon_k \|p_k\|^2\}. \end{aligned}$$

In general, suppose that the algorithm CG_Plan stops at step $l \leq n$. Then, the Newton direction that solves Eq. (2) is given by (see CG_Plan)

$$\sum_{k<l} \alpha_k p_k. \tag{27}$$

In Ref. [8], Dembo and Steihaug proved that the direction $d^P = \sum_{k \in I^P} \alpha_k p_k$ is gradient related; equivalently they proved that the standard CG method for solving Newton's equation (2), with A positive definite, generates the direction d^P, which is gradient related. In Ref. [16], Grippo *et al.* obtained an analogous result for the direction d^{PN} in Eq. (26) (with A nonsingular and indefinite), which is a slight alteration of the Newton direction (27) in order to obtain a gradient related direction. In particular, the authors in Ref. [16] simply reverse the negative curvatures of A that contribute to the Newton direction. Apart from the sign, they do not alter the value of the coefficient α_k, $k \in I^N$. Therefore we can conclude that, in some sense, they modify the pure Newton direction as little as possible. This helps to preserve the fast convergence that can be expected when Newton's direction is not harmfully perturbed.

In this section, we show that on one hand, the direction $d^{PN} + d^{\mathrm{Pla}}$ in Eq. (25) improves the similarity of $\tilde{d}_h$ to the Newton direction (i.e., the solution of Eq. (2)), with respect to d^{PN}. Indeed, d^{Pla} takes into account the contribution of p_k and p_{k+1}, $k \in I^{\mathrm{Pla}}$, to Newton's direction (27). In particular, observe that in Eq. (26), the coefficient α_k, $k \in I^{\mathrm{Pla}}$, of p_k is possibly altered in the sign ($\mathrm{sgn}[\omega_{k+1}]$), in order to preserve (in some sense) as much as possible relation (27).

On the other hand, the direction $\tilde{d}_h$ defined by Eqs. (25) and (26) satisfies properties (23) as shown below in Theorem 5.1. This completes the evolution drawn by Refs. [8,16], dealing with the case of a nonsingular indefinite Hessian A in Eq. (2). We remark that in the definition of the direction d^{Pla}, only the vector p_k gives a contribution; however, the coefficient of p_k relies on p_{k+1} (see Eq. (26)) and therefore the information contained in both p_k and p_{k+1}, $k \in I^{\mathrm{Pla}}$, contributes to building the direction d^{Pla}. Furthermore, if $p_{k+1}^{\mathrm{T}} A p_{k+1} = 0$, the step k_B does not contribute to d^{Pla}. This implies that formula (26) for d^{Pla} may not be adopted for the algorithm in Ref. [20], where p_{k+1}, $k \in I^{\mathrm{Pla}}$, is selected in such a way that $p_{k+1}^{\mathrm{T}} A p_{k+1} = 0$. A numerical comparison between our method and the approach in Ref. [16] is described in Ref. [12], where several unconstrained problems from the CUTE collection [4] are considered, and both a monotone and a nonmonotone stabilization scheme is adopted. Unfortunately, in very few cases the planar steps occur in such problems; therefore, at this stage of the research, we think that any conclusions would be premature. This motivates the choice of testing, in Section 6, the effectiveness of our approach with respect to the use of the standard routine SYMMLQ [22] in optimization frameworks.

[3] We define $sgn[x] = -1$ for $x < 0$ and $sgn[x] = +1$ if $x \geq 0$.

Let us now prove that the direction $\tilde{d}_h$ calculated according to Eqs. (25) and (26) by CG_Plan satisfies Eq. (23).

THEOREM 5.1 *Consider iteration* (5), *where* $\tilde{d}_h$ *approximately solves Eq.* (2). *Suppose that* $\lambda_m[\nabla^2 f(y_h)]$ *is uniformly bounded away from zero. Set* $x_1 = 0$ i.e. $r_1 = -\nabla f(y_h)$ *in CG_Plan and choose any* $\gamma_k \neq 0$, $k \geq 1$; *then the direction* $\tilde{d}_h$ *in Eqs.* (25) *and* (26) *is gradient related to* $\{y_h\}$; i.e. *relations* (23) *hold.*

Proof Consider the Newton equation (2) with $A = \nabla^2 f(y_h)$. After some rearrangements for the first relation in Eq. (23), we have

$$\begin{aligned}\tilde{d}_h^{\mathrm{T}} \nabla f(y_h) &= (d^{PN} + d^{\mathrm{Pla}})^{\mathrm{T}} \nabla f(y_h)\\ &= -\sum_{k \in I^P \cup I^N} \frac{p_k^{\mathrm{T}} r_k}{|p_k^{\mathrm{T}} A p_k|} p_k^{\mathrm{T}} r_1 + \sum_{k \in I^{\mathrm{Pla}}} \mathrm{sgn}[p_{k+1}^{\mathrm{T}} A p_{k+1}] \alpha_k p_k^{\mathrm{T}} r_1.\end{aligned}$$

Now, note that the relations $p_1 = r_1$, $p_i^{\mathrm{T}} r_i = p_i^{\mathrm{T}} r_1$, and $p_i^{\mathrm{T}} A r_1 = p_i^{\mathrm{T}} A p_1$ hold (see Theorem 3.1). We consider two cases: the first step was 1_A, and then

$$\tilde{d}_h^{\mathrm{T}} \nabla f(y_h) \leq -\frac{(p_1^{\mathrm{T}} r_1)^2}{|p_1^{\mathrm{T}} A p_1|} \leq -\frac{\|r_1\|^4}{\lambda_M \|r_1\|^2} = -\frac{1}{\lambda_M} \|r_1\|^2;$$

otherwise the first step was 1_B, and since at the k_Bth planar step $p_{k+1} = \gamma_k A p_k$ and $p_k^{\mathrm{T}} A r_1 = p_k^{\mathrm{T}} A p_1 = 0$, we have

$$\begin{aligned}\tilde{d}_h^{\mathrm{T}} \nabla f(y_h) &\leq \sum_{k \in I^{\mathrm{Pla}}} \mathrm{sgn}[p_{k+1}^{\mathrm{T}} A p_{k+1}] \alpha_k p_k^{\mathrm{T}} r_1\\ &= -\sum_{k \in I^{\mathrm{Pla}}} \mathrm{sgn}[p_{k+1}^{\mathrm{T}} A p_{k+1}] \frac{(p_k^{\mathrm{T}} r_1)^2}{\|A p_k\|^2} \left(\frac{A p_k}{\|A p_k\|}\right)^{\mathrm{T}} A \left(\frac{A p_k}{\|A p_k\|}\right)\\ &\leq -\sum_{k \in I^{\mathrm{Pla}}} \frac{(p_k^{\mathrm{T}} r_1)^2}{\|A p_k\|^2} \lambda_m \leq -\frac{(p_1^{\mathrm{T}} r_1)^2}{\|A p_1\|^2} \lambda_m \leq -\frac{\lambda_m}{\lambda_M^2} \|r_1\|^2.\end{aligned}$$

Therefore, we have the final relation

$$\tilde{d}_h^{\mathrm{T}} \nabla f(y_h) \leq -q_1 \|\nabla f(y_h)\|^{h_1}$$

with

$$q_1 = \min\left\{\frac{1}{\lambda_M}, \frac{\lambda_m}{\lambda_M^2}\right\} = \frac{\lambda_m}{\lambda_M^2}, \quad h_1 = 2,$$

which proves the first part of Eq. (23). For the second part of Eq. (23), in Ref. [16] the following relations were already proved (where $\varepsilon = \min_{k \geq 1}\{\varepsilon_k\}$ and ε_k is defined at step k of algorithm CG_Plan, as in Eq. (24)):

$$\|d^{PN}\| \leq 2\frac{n}{\varepsilon} \|\nabla f(y_h)\|.$$

Now, we give evidence that a similar relation holds for the direction d^{Pla}. Since $p_k^{\text{T}} r_k = p_k^{\text{T}} r_1$, we simply have from Eq. (26)

$$\begin{aligned}
\|d^{\text{Pla}}\| &\le \sum_{k\in I^{\text{Pla}}} \left\| \frac{p_k^{\text{T}} r_k}{\gamma_k^2 \|Ap_k\|^4} (p_{k+1}^{\text{T}} A p_{k+1}) p_k \right\| \\
&\le \sum_{k\in I^{\text{Pla}}} \left[\frac{\|p_k p_k^{\text{T}} r_1\|}{\|Ap_k\|^2} \cdot \left| \left(\frac{Ap_k}{\|Ap_k\|} \right)^{\text{T}} A \left(\frac{Ap_k}{\|Ap_k\|} \right) \right| \right] \\
&\le \sum_{k\in I^{\text{Pla}}} \frac{\|p_k\|^2 \lambda_M}{\|Ap_k\|^2} \|r_1\| \le \sum_{k\in I^{\text{Pla}}} \left[\frac{\lambda_M}{\lambda_m^2} \|r_1\| \right] \le \frac{n}{2} \frac{\lambda_M}{\lambda_m^2} \|r_1\|.
\end{aligned}$$

Therefore, for the second relation in Eq. (23), we have

$$\|\tilde{d}_h\| \le q_2 \|\nabla f(y_h)\|^{h_2}$$

with

$$q_2 = 2n \max\left\{ \frac{2}{\varepsilon}, \frac{\lambda_M}{2\lambda_m^2} \right\}, \qquad h_2 = 1.$$

■

Algorithm TN_Plan

Data: $A, g, \eta > 0, q_1 > 0, q_2 > 0, h_1 > 0, h_2 > 0.$
Step 1:
$k = 1,\ \varepsilon_1 > 0,\ r_1 = -g,\ p_1 = r_1$
$d_1 = 0,\ d_1^{PN} = 0,\ d_1^{\text{Pla}} = 0.$
Step 2:
Set $c_k = Ap_k$, $\delta_k = c_k^{\text{T}} p_k$
If $c_k = 0$ **Then go to Step 3**.
Elseif $|\delta_k| \ge \varepsilon_k \|p_k\|^2$ **Then**
$d_{k+1} = d_k + \alpha_k p_k, \quad \rho_k = r_k^{\text{T}} p_k, \quad \alpha_k = \frac{\rho_k}{\delta_k}$
$r_{k+1} = r_k - \alpha_k c_k$
$d_{k+1}^{PN} = d_k^{PN} + \text{sgn}\,[\delta_k]\, \alpha_k\, p_k$
$d_{k+1}^{\text{Pla}} = d_k^{\text{Pla}}$
If $\|r_{k+1}\| > \eta \|g\|$ **Then**
$p_{k+1} = r_{k+1} + \beta_k p_k, \quad \beta_k = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$
$k \leftarrow k+1$
set $\varepsilon_k > 0$, **go to Step 2**.
Else go to Step 3.
Else
$p_{k+1} = \gamma_k\, c_k, \qquad \gamma_k \in \mathbb{R}$
$a_{k+1} = Ap_{k+1}, \qquad \omega_{k+1} = p_{k+1}^{\text{T}} a_{k+1}$
$d_{k+2} = d_k + \alpha_k p_k + \alpha_{k+1} p_{k+1}, \qquad \alpha_k = -\frac{\rho_k\, \omega_{k+1}}{\gamma_k^2 \|c_k\|^4}$
$r_{k+2} = r_k - \alpha_k\, c_k - \alpha_{k+1}\, a_{k+1}, \qquad \alpha_{k+1} = \frac{\rho_k}{\gamma_k \|c_k\|^2}$
$d_{k+2}^{PN} = d_k^{PN}$
$d_{k+2}^{\text{Pla}} = d_k^{\text{Pla}} - sgn\,[\omega_{k+1}]\, \alpha_k\, p_k$

If $\|r_{k+2}\| > \eta \|g\|$ **Then**

$p_{k+2} = r_{k+2} + \sigma_k p_k, \qquad \sigma_k = -\frac{r_{k+2}^{\mathrm{T}} a_{k+1}}{\gamma_k \|c_k\|^2}$

$k \leftarrow k+2$

set $\varepsilon_k > 0$, **go to Step 2**.

Else go to Step 3.

Step 3:

Choose the gradient related direction $\tilde{d}_h$:

$$\tilde{d}_h = \begin{cases} -g & \text{if } k = 1 \\ d_k & \text{if } d_k^{\mathrm{T}} g \le -q_1 \|g\|^{h_1} \textbf{ AND } \|d_k\| \le q_2 \|g\|^{h_2} \\ d_k^{PN} + d_k^{\mathrm{Pla}} & \text{otherwise} \end{cases}$$

and **STOP**.

We note that the computation of the vector d^{Pla} does not need further calculations. Indeed, observe that both products Ap_k and Ap_{k+1} are available at the outset of step k_B.

Consider also that the previous proof relies on relations $p_i^{\mathrm{T}} r_i = p_i^{\mathrm{T}} r_1, i \ge 1$. When i is large the loss of conjugacy due to planar steps might determine condition $\mathrm{sgn}\{p_k^{\mathrm{T}} r_k\} \ne \mathrm{sgn}\{p_k^{\mathrm{T}} r_1\}$. However, we did not observe this drawback over the test problems we considered. Further, we checked the sequence $\{\nabla f(y_h)^{\mathrm{T}} d_k^{PN}\}$, $k \ge 1$ and verified that, as expected, it was monotonically nonincreasing.

We conclude this section by explicitly displaying a truncated Newton scheme [21], namely Algorithm TN_Plan, which uses algorithm CG_Plan with test (24) to solve the Newton equation (2) in an optimization framework. In Section 6, we compare the performance of algorithm TN_Plan with a truncated Newton scheme, where the routine SYMMLQ [22] is used. In algorithm TN_Plan, we set $A = \nabla^2 f(y_h)$ and $g = \nabla f(y_h)$ to simplify the notation. Observe that, since the test $p_k^{\mathrm{T}} A p_k = 0$ of CG_Plan is replaced by the test $|p_k^{\mathrm{T}} A p_k| \le \varepsilon_k \|p_k\|^2$, Step 3 of TN_Plan always provides a direction $\{\tilde{d}_h\}$ that is gradient related. In particular, if the approximate solution d_k of Eq. (2) is not gradient related, then the choice $\tilde{d}_h = d_k^{PN} + d_k^{\mathrm{Pla}}$ is gradient related to $\{y_h\}$.

6 PRELIMINARY NUMERICAL RESULTS

In this section, we include some preliminary numerical results that partially illustrate the performance of the proposed algorithm, when applied to optimization problems. Because of the tight test on the quantity $p_k^{\mathrm{T}} A p_k$ at step k of CG_Plan, we expect that our proposal cannot fruitfully be employed for general purposes too. Therefore, as remarked in Section 1, its application should be considered for suitable classes of problems. In order to investigate more accurately the fields of interest, where algorithm CG_Plan could be usefully applied, we tested it in the following cases.

(1) We randomly generated indefinite Hessian matrices A in Eq. (6), and compared the behavior of algorithm CG_Plan with the SYMMLQ routine by Paige and Saunders [22] for simply solving the linear system $Ax = b$.
(2) We used algorithm TN_Plan as a truncated method for approximately solving the Newton equation (2) in an optimization framework [see also Refs. [19,21] and references cited therein].

We compare the performance of CG_Plan (and TN_Plan) with SYMMLQ (and SYMMLQ-based truncated Newton method), because SYMMLQ is used in optimization frameworks for dealing with the solution of indefinite linear systems [see Ref. [21] and references cited therein].

6.1 Algorithm CG_Plan for Indefinite Linear Systems

For case (1) (i.e., to apply CG_Plan for solving indefinite linear systems), we first built a problem generator that provides triples of the form (A, x^*, b), where $A \in \mathbb{R}^{n\times n}$ is a symmetric and nonsingular matrix, $x^*, b \in \mathbb{R}^n$, and $Ax^* = b$. More precisely, matrix A is explicitly calculated by means of relation

$$A = HDH,$$

where $D = \text{diag}\{\lambda_1, \dots, \lambda_n\}$ and H is the Householder orthogonal transformation: $H = I - 2zz^{\mathrm{T}}/\|z\|^2$, $z \in \mathbb{R}^n$, $z \neq 0$. According to the user's specifications, the generator provides the sequence of real diagonal elements λ_i, $i \leq n$, which become the eigenvalues of A. Then the components of z and x^* are randomly chosen in the interval $\{-1, 1\}$. Finally, vector b is calculated as $b = Ax^*$. The sequence $\{\lambda_i\}$ may be generated in such a way that the user can independently assign: the *dimension* 'n' of A, the *condition number* 'cond' of A, a *scale factor* 'λ_0' for the eigenvalues of A, whatever *clustering* for the eigenvalues $\lambda_2, \dots, \lambda_{n-1}$ of A, and the *inertia* of A (the number of positive and negative eigenvalues).

For preliminary testing, we generated indefinite Hessian matrices A, where blocks of either negative and positive eigenvalues are separately constructed according to the following parameters:

- $n = 500$ (though the experience revealed that similar results hold with $n = 100$ or $n = 1000$);
- full rank, i.e., A is nonsingular;
- condition number (for each block of positive and negative eigenvalues) cond, where cond $= e^i$, $i = 0, 2, 4, 6$;
- inertia equal to $(n/2, n/2)$;
- scale factor of eigenvalues $\lambda_0 = 10^{-4}$;
- clustering around $\lambda_m^{-/+}$ and $\lambda_M^{-/+}$, where $\lambda_m^{-/+}[\lambda_M^{-/+}]$ indicates the smallest [largest] absolute value of eigenvalues in either the negative or the positive block.

Tables I and II summarize the results obtained by applying CG_Plan and SYMMLQ to the solution of $Ax = b$ with $n = 500$. The parameter frac ≤ 1 indicates the width of the clusters, expressed as a percentage of the positive distance $\lambda_M^{-/+} - \lambda_m^{-/+}$ (i.e., $\lambda_i^{-/+} - \lambda_m^{-/+} \leq \text{frac} \cdot (\lambda_M^{-/+} - \lambda_m^{-/+})$, $i = 2, \dots, (n/2 - 1)$, and $\lambda_M^{-/+} - \lambda_i^{-/+} \leq \text{frac} \cdot (\lambda_M^{-/+} - \lambda_m^{-/+})$, $i = 2, \dots, (n/2 - 1)$).

Finally, each row gives the average results over 10 instances, randomly generated with the specifications above; 'it' is the average number of iterations (we allowed up to $2n$ iterations as in Ref. [15]), and 'pla' the average number of planar steps. The algorithms used a *random starting point* and stopped when the current iterate x_k satisfied the following simple test:

$$\|r_k\| = \|b - Ax_k\| \leq \text{tol} \cdot |A| \cdot \|x_k\|,$$

where tol $= 10^{-8}$ and $|A|$ is an estimate of $\|A\|$. This test is currently performed within SYMMLQ and proved to be quite effective within optimization frameworks, since the quantities

TABLE I Algorithm CG_Plan, stopping criterion $\|r_k\| \leq 10^{-8}\|A\|\|x_k\|$

n = 500	$\lambda_i \in cluster\{\lambda_m\}$				$\lambda_i \in cluster\{\lambda_M\}$			
(cond)	$\|x^* - x_I\|$	$\|r\|/\|r_I\|$	*it*	*pla*	$\|x^* - x_I\|$	$\|r\|/\|r_I\|$	*it*	*pla*
			frac = 1.0					
e^0	3.5E−16	2.0E−13	1.0	0.0	1.6E−11	8.7E−09	1.2	0.2
e^2	1.1E−07	1.3E−05	75.0	0.6	1.1E−07	1.3E−05	74.1	0.2
e^4	1.1E−07	2.0E−06	437.6	0.4	1.1E−07	1.9E−06	425.4	0.3
e^6	2.7E−05	6.8E−05	812.7	0.5	1.2E−06	2.7E−06	829.3	0.6
			frac = 0.8					
e^0	1.2E−15	6.5E−13	1.0	0.0	2.9E−16	1.6E−13	1.0	0.0
e^2	1.0E−07	1.5E−05	64.0	0.3	9.6E−08	1.0E−05	43.4	0.2
e^4	1.1E−07	2.5E−06	386.3	0.5	1.1E−07	1.7E−06	85.0	0.1
e^6	1.8E−06	5.4E−06	881.3	0.8	1.0E−07	2.1E−07	117.9	0.0
			frac = 0.6					
e^0	1.0E−12	5.7E−10	1.2	0.2	2.6E−15	1.4E−12	1.0	0.0
e^2	1.0E−07	1.7E−05	52.5	0.4	8.4E−08	8.1E−06	29.8	0.2
e^4	1.1E−07	3.2E−06	322.4	0.4	1.0E−07	1.4E−06	47.0	0.2
e^6	2.1E−06	7.6E−06	842.6	0.7	8.2E−08	1.5E−07	62.6	0.0
			frac = 0.4					
e^0	1.7E−11	9.7E−09	1.2	0.2	9.2E−12	4.9E−09	1.2	0.2
e^2	8.7E−08	1.9E−05	39.4	0.2	6.7E−08	5.9E−06	21.0	0.1
e^4	1.1E−07	4.6E−06	241.1	0.4	9.1E−08	1.1E−06	31.0	0.0
e^6	6.0E−07	3.4E−06	889.6	1.1	4.3E−08	7.2E−08	41.0	0.0
			frac = 0.2					
e^0	1.5E−15	8.3E−13	1.0	0.0	9.0E−12	5.2E−09	1.4	0.4
e^2	8.4E−08	2.6E−05	25.4	0.3	4.2E−08	3.4E−06	15.0	0.0
e^4	1.1E−07	8.0E−06	139.0	0.2	5.8E−08	6.4E−07	21.0	0.0
e^6	1.1E−07	1.1E−06	683.2	0.8	3.5E−08	5.4E−08	27.0	0.0

$|A|$ and $\|x_k\|$ properly take into account the scale of the problem. We emphasize that $|A|$ is the Frobenius norm of a tridiagonal approximation of matrix A. In this preliminary setting, the algorithm CG_Plan performed planar step k_B when $|p_k^{\mathrm{T}} A p_k| \leq 0.5 \cdot 10^{-6} \|p_k\|^2$, therefore we set $\varepsilon_k = 0.5 \cdot 10^{-6}$, $k \geq 1$, and provisionally avoided the use of relation (21). On one hand, the results reveal that algorithm CG_Plan may be competitive as a linear solver only in terms of the number of iterations, provided that the condition number of matrix A is not large. Furthermore, we observe that the clustering of the eigenvalues improves the performance of the algorithm CG_Plan, as for the standard CG method with positive definite matrix A. These results are not surprising since SYMMLQ is specifically designed for the indefinite problem (6). In addition, it is well known that the CG/Lanczos method implemented in SYMMLQ, though more expensive, is definitely competitive with the standard CG in terms of the precision of the solution. Anyway, we recall that the solution of Newton's equation may be effective even though it is only approximately calculated. Indeed, a higher precision in solving Newton's equation may not justify the increase of the overall time of computation. To summarize, if the condition number of A is not large and we are not concerned with getting a precise solution, then CG_Plan may be an inexpensive solver of indefinite linear systems. Otherwise, SYMMLQ seems preferable.

All calculations were performed on a PC Pentium II 850 MHz. Algorithm CG_Plan was implemented in Fortran 90 and compiled with Compaq Visual Fortran, with double precision used throughout.

TABLE II Algorithm SYMMLQ, stopping criterion $\|r_k\| \le 10^{-8}\|A\|\|x_k\|$

$n = 500$	$\lambda_i \in cluster\{\lambda_m\}$			$\lambda_i \in cluster\{\lambda_M\}$		
(cond)	$\|x^* - x_I\|$	$\|r\|/\|r_I\|$	*it*	$\|x^* - x_I\|$	$\|r\|/\|r_I\|$	*it*
			frac = 1.0			
e^0	2.3E−18	1.2E−15	1.0	2.4E−18	1.3E−15	1.0
e^2	5.8E−10	7.0E−08	110.2	5.6E−10	6.7E−08	109.6
e^4	9.3E−09	1.6E−07	479.1	8.9E−09	1.5E−07	478.6
e^6	7.2E−08	1.7E−07	710.0	7.5E−08	1.7E−07	696.9
			frac = 0.8			
e^0	2.2E−18	1.2E−15	1.0	2.2E−18	1.2E−15	1.0
e^2	4.8E−10	6.9E−08	93.6	4.3E−10	4.6E−08	59.0
e^4	7.7E−09	1.6E−07	442.2	4.0E−09	6.2E−08	98.3
e^6	6.4E−08	1.8E−07	728.0	3.3E−08	7.1E−08	121.3
			frac = 0.6			
e^0	2.2E−18	1.2E−15	1.0	2.3E−18	1.2E−15	1.0
e^2	3.1E−10	5.5E−08	78.7	3.0E−10	3.0E−08	39.0
e^4	5.4E−09	1.5E−07	382.7	2.2E−09	3.1E−08	55.0
e^6	5.0E−08	1.9E−07	692.4	2.1E−08	3.9E−08	65.0
			frac = 0.4			
e^0	2.3E−18	1.2E−15	1.0	2.2E−18	1.2E−15	1.0
e^2	2.9E−10	6.5E−08	59.0	1.4E−10	1.3E−08	28.6
e^4	4.0E−09	1.6E−07	290.6	9.8E−10	1.2E−08	37.0
e^6	4.2E−08	2.4E−07	715.2	7.7E−09	1.2E−08	43.0
			frac = 0.2			
e^0	2.4E−18	1.3E−15	1.0	2.4E−18	1.3E−15	1.0
e^2	2.0E−10	6.6E−08	37.0	2.7E−10	2.2E−08	19.0
e^4	2.8E−09	2.0E−07	174.2	4.9E−10	5.4E−09	25.0
e^6	4.2E−08	4.2E−07	599.0	5.0E−09	7.5E−09	28.8

6.2 Algorithm TN_Plan for Optimization Problems

For case (2) (i.e., to test TN_Plan within optimization frameworks), we apply a truncated Newton method to the general problem

$$\min_{y\in\mathbb{R}^n} f(y) \quad f\colon \mathbb{R}^n \longrightarrow \mathbb{R}, \ \ n \text{ large},$$

where $f(y)$ is allowed to be *nonconvex* and 'min' stands for a local minimum. At the hth iteration, we apply both algorithms TN_Plan ($A = \nabla^2 f(y_h)$, $g = \nabla f(y_h)$, $q_1 = 10^{-8}$, $q_2 = 10^8$, $h_1 = 2$, $h_2 = 1$, and $\gamma_k = \|p_k\|/\|Ap_k\|$) and SYMMLQ to determine direction $\tilde{d}_h \in \mathbb{R}^n$ as an approximate solution of Newton's equation (2)

$$\nabla^2 f(y_h)d + \nabla f(y_h) = 0 \quad d \in \mathbb{R}^n. \tag{28}$$

Then, a monotone Armijo-type linesearch is performed along $\tilde{d}_h$ and a steplength α_h is calculated to obtain the next iterate $y_{h+1} = y_h + \alpha_h\tilde{d}_h$. Under standard assumptions, the globalization scheme ensures convergence. As stated in Section 5, if $\tilde{d}_h$ is calculated by means of TN_Plan, it is always gradient related to $\{y_h\}$. On the other hand, SYMMLQ often does not provide a gradient related direction; thus some arrangements are necessary, which may diminish the efficiency of the overall algorithm. In particular, if SYMMLQ calculates d_h^{LQ} as a solution of Eq. (28), then the following direction $\tilde{d}_h$ is provided to the linesearch:

$$\tilde{d}_h = \begin{cases} d_h^{\mathrm{LQ}} & \text{if } d_h^{\mathrm{LQ}} \text{ is gradient related to } \{y_h\} \\ -D_h \nabla f(y_h) & \text{otherwise,} \end{cases} \tag{29}$$

where D_h is a suitable positive definite matrix. Frequent use of the modified steepest descent direction generally leads to intolerably slow progress. In our preliminary tests, we simply set $D_h = I$, $h \geq 1$, even though a better choice of D_h seems to deserve further investigation. The choice (29) was successfully adopted in Ref. [19], within a curvilinear stabilization framework.

We tested the scheme over a set of 78 large-scale nonlinear functions from the CUTE collection [4]; this test set contains both *convex* and *nonconvex* functions. We implemented a truncated Newton scheme with standard settings and a monotone linesearch, with Newton's equation (28) being approximately solved by both TN_Plan and SYMMLQ. The results are summarized in Table III (convex problems) and Table IV (nonconvex problems). More specifically, in Table III, we give the results on the test problems where SYMMLQ always generated a gradient related direction d_h^{LQ} that approximately solved (28). In Table IV, we report results where SYMMLQ generated *nongradient related* Newton type directions, i.e., those problems, where for some h we had to choose $\tilde{d}_h = -\nabla f(y_h)$ for SYMMLQ (see Eq. (29)). We used an IBM RISC System/6000, and the stopping criterion for the overall optimization method was simply

$$\|\nabla f(y_h)\| < 10^{-5}. \tag{30}$$

For each optimization step, the truncation criterion for approximately solving Eq. (28) was the original test of the SYMMLQ routine, i.e.,

$$\|\nabla^2 f(y_h) d_h + \nabla f(y_h)\| \leq \eta_h$$

with $\eta_h = \mathrm{tol}_h \cdot |A| \cdot \|d_h\|$. Similarly to SYMMLQ, $|A|$ is the norm of a tridiagonal approximation of $\nabla^2 f(y_h)$, and [see Ref. 21]

$$\mathrm{tol}_h = 10^{-2} \min\left\{\frac{1}{h}, \|\nabla f(y_h)\|\right\} \|\nabla f(y_h)\|.$$

The acronyms in Tables III and IV represent: the size of the problem (n), the number of outer iterations (iter) (i.e., the number of points generated by the truncated Newton scheme), the function evaluations (func), the number of CG_Plan iterations (CG-it), the number of SYMMLQ iterations (SYM-it), the counter of Newton-type directions that are not gradient related (viol) (i.e., here we choose, respectively, $\tilde{d}_h = d_k^{PN} + d_k^{\mathrm{Pla}}$ for TN_Plan or $\tilde{d}_h = -\nabla f(y_h)$ for SYMMLQ), the function value at the solution (f) (in Table III it coincides for TN_Plan and SYMMLQ, so we did not report it). On six test problems (EIGENBLS, GENHUMPS, MSQRTALS, NONCVXUN, NONCVXU2, and SPARSINE, with $n = 10{,}000$) both the algorithms fail, so the relative results are not reported. Finally, we considered a failure if either iter > 5000 or the time of computation exceeded 0.5 h (MAX time). At the bottom of Tables III and IV, for each algorithm we report the number of wins, in terms of function evaluations and inner iterations (since the latter seem the most significant parameters in a comparison on large-scale problems).

The results confirm (Table III) that in these settings, as long as no negative curvatures are encountered by the iterative method, or the solution of Eq. (28) is gradient related, SYMMLQ seems to be more precise and competitive than TN_Plan. Indeed, even though in general,

TABLE III CUTE test problems: convex case

		Algorithm TN_Plan				*Algorithm SYMMLQ*			
Problem	*n*	*iter*	*func*	*CG−it*	*viol*	*iter*	*func*	*SYM−it*	*viol*
ARWHEAD	1000	6	7	2	0	6	7	6	0
ARWHEAD	5000	5	6	1	0	6	7	6	0
BDQRTIC	1000	36	37	60	0	27	28	71	0
BRYBND	1000	17	24	123	0	25	26	113	0
BRYBND	5000	19	25	138	0	19	21	65	0
CRAGGLVY	1000	19	20	179	0	20	21	105	0
DIXMAANA	3000	6	7	3	0	6	8	8	0
DIXMAANB	3000	7	8	3	0	7	8	7	0
DIXMAANC	3000	8	9	3	0	8	9	9	0
DIXMAAND	3000	8	9	3	0	9	10	10	0
DIXMAANE	3000	8	9	306	0	15	16	295	0
DIXMAANF	3000	20	42	1939	0	14	15	385	0
DIXMAANG	3000	19	30	1997	1	15	16	299	0
DIXMAANH	3000	15	22	1383	0	15	16	280	0
DIXMAANI	3000	9	10	5572	0	27	28	4366	0
DQRTIC	1000	30	31	0	0	30	31	57	0
DQRTIC	5000	35	36	0	0	35	36	63	0
EDENSCH	2000	14	16	50	0	13	14	26	0
FMINSURF	1024	22	131	1812	0	50	177	470	0
FMINSURF	5625	33	211	4418	0	99	392	1311	0
FMINSURF	10,000	28	188	4329	0	129	542	2001	0
LIARWHD	1000	39	40	8	0	12	13	12	0
LIARWHD	10,000	17	19	7	0	12	13	12	0
MOREBV	1000	1	2	1999	0	3	4	965	0
MOREBV	5000	1	2	9999	0	2	3	858	0
NONDIA	1000	6	7	3	0	6	7	6	0
NONDIA	10,000	4	5	1	0	5	6	5	0
PENALTY1	1000	40	43	14	0	40	43	40	0
POWELLSG	1000	22	23	47	0	20	21	47	0
POWELLSG	10,000	24	25	51	0	22	23	51	0
POWER	1000	43	44	1092	0	40	41	258	0
QUARTC	1000	30	31	0	0	30	31	57	0
QUARTC	10,000	37	38	0	0	37	38	65	0
SCHMVETT	1000	5	6	72	0	7	8	46	0
SCHMVETT	10,000	6	7	89	0	7	8	47	0
SROSENBR	1000	7	8	3	0	8	10	8	0
SROSENBR	10,000	7	8	3	0	8	10	8	0
TESTQUAD	1000	275	276	3367	0	456	457	3405	0
TOINTGSS	1000	3	4	12	0	4	5	13	0
TOINTGSS	10,000	2	3	1	0	4	5	8	0
TQUARTIC	1000	9	15	5	1	1	2	1	0
TQUARTIC	10,000	8	13	4	0	1	2	1	0
TRIDIA	1000	45	46	638	0	86	87	34	0
TRIDIA	5000	97	98	2095	0	237	238	2534	0
VAREIGVL	1000	14	15	2468	0	14	15	311	0
VAREIGVL	5000	22	32	3912	0	15	16	304	0
Total wins			21	22			15	22	

TABLE IV CUTE test problems: nonconvex case

		Algorithm TN_Plan					Algorithm SYMMLQ + steepest descent				
Problem	*n*	*iter*	*func*	*CG−it*	*f*	*viol*	*iter*	*func*	*SYM−it*	*f*	*viol*
BROYDN7D	1000	114	396	43,732	0.627127D+03	38	776	5016	19,272	0.385697D+03	722
CHAINWOO	1000	26	43	311	0.124217D+02	3	>5000				4330
CHAINWOO	10,000	35	44	324	0.279814D+02	2		****MAX time****			1703
COSINE	1000	7	15	32	−0.999000D+03	2	36	377	130	−0.999000D+03	3
COSINE	10,000	6	7	12	−0.999900D+04	1	42	480	101	−0.999900D+04	4
CURLY10	1000	59	60	9214	−0.100316D+06	11		****MAX time****			21
CURLY20	1000	86	88	7515	−0.100316D+06	10	651	5571	134,976	−0.100306D+06	14
CURLY30	1000	103	104	8939	−0.100316D+06	10		****MAX time****			11
EIGENALS	930	52	59	1032	0.128865D−11	1	243	990	2323	0.320360D−10	2
FLETCHCR	1000	1474	1682	21,564	0.257064D−16	1	3864	>10,000	32,135	0.192444D−14	2061
FREUROTH	1000	11	13	20	0.121470D+06	1	>5000				1
FREUROTH	5000	11	13	19	0.608159D+06	1		****MAX time****			1
GENROSE	1000	793	1409	18,586	0.10,0000D+01	55	1451	>10,000	16,503	0.10,0000D+01	668
MSQRTBLS	1024	22	33	8854	0.145126D−15	1	193	615	4906	0.762770D−11	14
NCB20B	1000	10	12	1326	0.167601D+04	0		****MAX time****			209
NONCVXUN	1000	212	436	217,165	0.235153D+04	35	607	4649	108,588	0.233723D+04	191
NONCVXU2	1000	116	387	10,536	0.234094D+04	32	450	2612	18,137	0.231858D+04	206
SINQUAD	1000	42	70	67	0.405267D−05	2	>5000	>10,000			4992
SINQUAD	10,000	97	159	114	0.296285D−04	4		****MAX time****			1825
SPARSINE	1000	33	34	3915	0.749965D−13	0	1778	>10,000	138,488	0.537392D−12	1490
SPMSRTLS	1000	14	21	543	0.566963D−15	1	17	24	246	0.469453D−15	1
SPMSRTLS	4999	25	46	1494	0.240442D−14	3	23	27	309	0.181290D−12	1
SPMSRTLS	10,000	27	62	1569	0.939622D−01	2	26	30	230	0.152396D−10	1
VARDIM	1000	36	37	0	0.254847D−19	30	18	177	18	0.000000D+00	11
WOODS	1000	55	117	111	0.204919D−20	10	166	886	339	0.894989D−15	61
WOODS	10,000	64	127	108	0.370726D−15	11	1059	9435	1518	0.401339D−14	971
Total wins			24	19				2	7		

a larger number of function evaluations are necessary, a smaller number of Lanczos iterations suffices for estimating the solution of Eq. (28). However, this conclusion is reversed when nonconvex problems are solved (see Table IV). In this case, the use of $-\nabla f(y_h)$ in place of d_h^{LQ} often proves harmful for the overall optimization algorithm. On the contrary, the proper manipulation of the conjugate directions generated by algorithm TN_Plan (vectors d^{PN} and d^{Pla} in Section 5) proved to be significantly useful in providing a gradient-related direction to the linesearch technique. Thus, we believe that the Lanczos process may be successful when a curvilinear framework is considered [19,15], or in a trust region approach [14]. In order to recover the effectiveness of a Lanczos-based method on nonconvex problems, specific treatment of negative curvature of the Hessian matrix must be considered. Our algorithm TN_Plan provides one such approach.

Unfortunately, only over a few test problems TN_Plan performed planar steps. Therefore, we were not able to experience TN_Plan on a significant test set, in order to give also a complete practical evidence of the theoretical robustness of our approach.

7 CONCLUSIONS AND PERSPECTIVES

In this article, we have proposed a CG-type method, namely CG_Plan, in the class of Krylov subspace methods, for the iterative solution of indefinite linear systems within optimization frameworks. This new algorithm overcomes the premature possible stopping of CG, in the case of indefinite linear systems. The practical implementation of CG_Plan is affected by the choice, at step k, of the test on quantity $p_k^{\mathrm{T}} A p_k$ (see Sec. 5).

The algorithm CG_Plan was suitably adapted to solve Newton's equation, and we proved that the resulting algorithm (TN_Plan) always provides gradient-related search directions within the optimization framework.

In our opinion, at step k of CG_Plan, a *careful choice* of the parameter ε_k within the test on the quantity $p_k^{\mathrm{T}} A p_k$ (Sec. 5), should be further investigated.

Other interesting issues for future work on algorithm CG_Plan are: the possibility of introducing specific preconditioners for CG_Plan, and the use of CG_Plan for the generation of negative curvature directions, in nonconvex optimization. On one hand, we believe that for large-scale problems, preconditioners often represent a fruitful tool to speed up the convergence of the iterative methods. On the other hand, negative curvature plays a key role within nonconvex optimization to ensure convergence to local minima that satisfy the second-order necessary conditions for optimality.

Acknowledgments

I feel deeply indebted towards Stefano Lucidi for having trusted me and for his spurs during my Ph.D. course. I thank Luigi Grippo too, for his huge respect in teaching me to be humble. Finally, the corrections proposed by Massimo Roma and the valuable observations of the two referees were really helpful.

References

[1] O. Axelsson (1996). *Iterative Solution Methods*. Cambridge University Press.
[2] R.E. Bank and T.F. Chan (1994). A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems. *Numerical Algorithms*, **7**, 1–16.
[3] D.P. Bertsekas (1995). *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, USA.
[4] I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint (1995). CUTE: constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, **21**, 123–160.

[5] J.R. Bunch and B.N. Parlett (1971). Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.*, **8**, 639–655.
[6] J.R. Bunch (1974). Partial pivoting strategies for symmetric matrices. *SIAM J. Numer. Anal.*, **11**(8), 521–528.
[7] A.T. Chronopoulos and S.K. Kim (1990). s-Step Orthomin and GMRES implemented on parallel computers. Technical Report 90/43R, UMSI, Minneapolis.
[8] R.S. Dembo and T. Steihaug (1983). Truncated-Newton methods algorithms for large scale unconstrained optimization. *Mathematical Programming*, **26**, 190–212.
[9] J.W. Demmel (1997). *Applied Numerical Linear Algebra*. SIAM, Philadelphia.
[10] G. Fasano (2000). *Conjugate directions inside Newton-type algorithms, for large-scale unconstrained optimization*. PhD thesis in Operations Research, University of Rome 'La Sapienza', XIII course, Italy.
[11] G. Fasano (2002). Planar-CG methods and matrix tridiagonalization in large-scale unconstrained optimization. *Proceedings of the 33rd Workshop on High Performance Algorithms and Software for Nonlinear Optimization*, Erice, Sicily, from June 30th to July 8th, 2001.
[12] G. Fasano (2001). A new CG-type method for the solution of indefinite linear systems in optimization frameworks. Technical Report 08-01, Dipartimento di Informatica e Sistemistica, University of Rome 'La Sapienza', Italy.
[13] R. Fletcher (1975). Conjugate gradient methods for indefinite systems. In: G.A. Watson (Ed.), *Proc. of the Dundee Biennal Conf. on Numerical Analysis*, Springer, Berlin, Heidelberg, New York.
[14] N.I.M. Gould, S. Lucidi, M. Roma and Ph.L. Toint (1999). Solving the trust-region subproblem using the Lanczos method. *SIAM J. Optim.*, **9**(2), 504–525.
[15] N.I.M. Gould, S. Lucidi, M. Roma and Ph.L. Toint (2000). Exploiting negative curvature directions in linesearch methods for unconstrained optimization. *Optim. Methods & Soft.*, **14**, 75–98.
[16] L. Grippo, F. Lampariello and S. Lucidi (1989). A truncated Newton method with nonmonotone linesearch for unconstrained optimization. *J. of Optimization Theory and Applications*, **60**, 401–419.
[17] M.R. Hestenes (1980). *Conjugate Direction Methods in Optimization*. Springer Verlag, New York, Heidelberg, Berlin.
[18] M.R. Hestenes and E. Stiefel (1952). Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards* **49**, 409–435.
[19] S. Lucidi, F. Rochetich and M. Roma (1999). Curvilinear stabilization techniques for Truncated Newton methods in large scale unconstrained optimization. *SIAM J. Optim.*, **8**(4), 916–939.
[20] D.G. Luenberger (1969). Hyperbolic pairs in the method of conjugate gradients. *SIAM J. Appl. Math.*, **17**(6), 1263–1267.
[21] S.G. Nash (2000). A survey of truncated-Newton methods. *J. Comp. and Appl. Math.*, **124**, 45–59.
[22] C.C. Paige and M.A. Saunders (1975). Solution of sparse indefinite systems of linear equations. *SIAM J. on Numerical Analysis*, **12**, 617–629.
[23] Y. Saad and H.A. Van Der Vorst (2000). Iterative solution of linear systems in the 20th century. *J. Comp. Appl. Math.*, **123**, 1–33.
[24] G.L.G. Sleijpen and H.A. Van Der Vorst (1993). Krylov subspace methods for large linear systems of equations. Preprint 803, Department of Mathematics, University of Utrecht.
[25] L.N. Trefethen and D. Bau III (1997). *Numerical Linear Algebra*. SIAM, Philadelphia.
[26] H.A. Van Der Vorst and T.F. Chan (1997). Linear system solvers: sparse iterative methods. In: D.E. Keyes, A.Samed and V. Venkatakrishnan (Eds.), *Parallel Numerical Algorithms. ICASE/LaRC Interdisciplinary Series in Science and Engineering*, Vol. 4, pp. 91–118, Kluwer Academic, Dodrecht.
[27] R. Weiss (1990). *Convergence behavior of generalized conjugate gradient methods*. PhD dissertation, University of Karlsruhe, Karlsruhe, Germany.

A APPENDIX

Here, we prove Theorem 4.1 of Section 4. There are two cases to be examined. On one hand, if step k_B is preceded by the step $(k-1)_A$, then we have

$$\begin{aligned}
p_{k+1}^{\mathrm{T}} A p_{k-1} &= \left(\frac{\|p_k\| A p_k}{\|A p_k\|}\right)^{\mathrm{T}} (A p_{k-1}) = \left(\frac{\|p_k\| A p_k}{\|A p_k\|}\right)^{\mathrm{T}} \left(\frac{r_{k-1} - r_k}{\alpha_{k-1}}\right) \\
&= \left(\frac{\|p_k\| A p_k}{\|A p_k\|}\right)^{\mathrm{T}} \left[\frac{p_{k-1} - \omega(p_{k-2}, p_{k-3})}{\alpha_{k-1}} - \frac{r_k}{\alpha_{k-1}}\right] \\
&= -\left(\frac{\|p_k\| A p_k}{\|A p_k\|}\right)^{\mathrm{T}} \frac{r_k}{\alpha_{k-1}} = -\frac{\|p_k\|}{\|A p_k\| \alpha_{k-1}} p_k^{\mathrm{T}} A p_k, \qquad (31)
\end{aligned}$$

where (see algorithm CG_Plan)

$$\omega(p_{k-2}, p_{k-3}) = \begin{cases} \beta_{k-2} p_{k-2} & \text{if } p_{k-1} \text{was generated at step } (k-2)_A \\ \sigma_{k-3} p_{k-3} & \text{if } p_{k-1} \text{ was generated at step } (k-3)_B, \end{cases}$$

$\alpha_{k-1} = \|r_{k-1}\|^2 / p_{k-1}^{\mathrm{T}} A p_{k-1}$, and the last equality in Eq. (31) is a consequence of the conjugacy of vector p_k with the directions $p_{k-1}, p_{k-2}, p_{k-3}$.

On the other hand, if step k_B is preceded by step $(k-2)_B$, from Eq. (19) we cannot assume, in general, $p_{k-2}^{\mathrm{T}} A p_{k-2} = 0$; then we have, for the vector $A p_{k-1}$ at step $(k-2)_B$, the expression $A p_{k-1} = (r_{k-2} - r_k - \alpha_{k-2} A p_{k-2})/\alpha_{k-1}$, where now (after few calculations and using the reasoning that gave (15))

$$\alpha_{k-1} = \frac{\|Ap_{k-2}\|^3 \|r_{k-2}\|^2 - (p_{k-2}^{\mathrm{T}} A p_{k-2})^2 \|Ap_{k-2}\|}{\|p_{k-2}\| \|Ap_{k-2}\|^4 - \|p_{k-2}\| (p_{k-2}^{\mathrm{T}} A p_{k-2})(p_{k-2}^{\mathrm{T}} A^3 p_{k-2})}, \tag{32}$$

which yields, along with Theorem 3.1, the relation

$$p_{k+1}^{\mathrm{T}} A p_{k-1} = -\left(\frac{\|p_k\| A p_k}{\|Ap_k\|} \right)^{\mathrm{T}} \frac{r_k}{\alpha_{k-1}} = -\frac{\|p_k\|}{\|Ap_k\| \alpha_{k-1}} p_k^{\mathrm{T}} A p_k. \tag{33}$$

Therefore, from Eqs. (19), (31), and (33), regardless of the step that precedes the current step k_B in algorithm CG_Plan, we can ensure that

$$|p_{k+1}^{\mathrm{T}} A p_{k-1}| \le \frac{\varepsilon_k}{\lambda_m |\alpha_{k-1}|} \|p_k\|^2. \tag{34}$$

In addition, in order to calculate an upper bound for the quantity $|p_{k+1}^{\mathrm{T}} A p_{k-1}|$ we need some further results: we calculate a lower bound for the coefficient $|\alpha_{k-1}|$.

On one hand at general step $(i-1)_A$, we have $p_i = r_i + \beta_{i-1} p_{i-1}$ and $|p_{i-1}^{\mathrm{T}} A p_{i-1}| > \varepsilon_{i-1} \|p_{i-1}\|^2$; thus if we consider for the parameter ε_{i-1}, the expression

$$\varepsilon_{i-1}^A \le \frac{\lambda_m}{2} \left(\frac{\lambda_m}{\lambda_M} \right)^3, \tag{35}$$

where the exponent 'A' indicates the step, we obtain

$$\begin{aligned} \|p_i\| &\le \|r_i\| + \left\| \frac{r_i^{\mathrm{T}} A p_{i-1}}{p_{i-1}^{\mathrm{T}} A p_{i-1}} p_{i-1} \right\| = \|r_i\| + \left\| \frac{p_{i-1} p_{i-1}^{\mathrm{T}} A}{p_{i-1}^{\mathrm{T}} A p_{i-1}} r_i \right\| \\ &\le \left[1 + 2 \left(\frac{\lambda_M}{\lambda_m} \right)^4 \right] \|r_i\|. \end{aligned} \tag{36}$$

On the other hand at general step $(i-2)_B$, since $\gamma_{i-2} = \|p_{i-2}\| / \|Ap_{i-2}\|$ and in general $|p_{i-2}^{\mathrm{T}} A p_{i-2}| \neq 0$, the following relations hold for coefficients σ_{i-2} and σ_{i-1} (see Theorem 3.1):

$$\begin{aligned} \sigma_{i-2} &= \frac{\|Ap_{i-2}\|^2 r_i^{\mathrm{T}} A^2 p_{i-2}}{(p_{i-2}^{\mathrm{T}} A p_{i-2})(p_{i-2}^{\mathrm{T}} A^3 p_{i-2}) - \|Ap_{i-2}\|^4}, \\ \sigma_{i-1} &= \frac{p_{i-2}^{\mathrm{T}} A p_{i-2}}{\|p_{i-2}\|} \frac{\|Ap_{i-2}\| r_i^{\mathrm{T}} A^2 p_{i-2}}{\|Ap_{i-2}\|^4 - (p_{i-2}^{\mathrm{T}} A p_{i-2})(p_{i-2}^{\mathrm{T}} A^3 p_{i-2})}; \end{aligned} \tag{37}$$

moreover, similarly to Eq. (35) we consider now for ε_{i-2}, the expression

$$\varepsilon_{i-2}^{B} \leq \frac{\lambda_m}{2} \min\left\{ \left(\frac{\lambda_m}{\lambda_M}\right)^3, 2^{1/2}\frac{\|r_{i-2}\|}{\|p_{i-2}\|} \right\}, \tag{38}$$

where the exponent 'B' indicates the step, and similarly to Eq. (36) (with $|p_{i-2}^{\mathrm{T}} A p_{i-2}| < \varepsilon_{i-2}\|p_{i-2}\|^2$), we have

$$\begin{aligned} \|p_i\| &\leq \|r_i\| + \|\sigma_{i-1} p_{i-1} + \sigma_{i-2} p_{i-2}\| \\ &\leq \|r_i\| + \frac{\left\| p_{i-2}^{\mathrm{T}} A p_{i-2} A - \|A p_{i-2}\|^2 I \right\| \cdot \|p_{i-2} p_{i-2}^{\mathrm{T}} A^2\|}{|\|A p_{i-2}\|^4 - (p_{i-2}^{\mathrm{T}} A p_{i-2})(p_{i-2}^{\mathrm{T}} A^3 p_{i-2})|} \|r_i\|. \end{aligned} \tag{39}$$

Therefore, we obtain

$$\begin{aligned} \|p_i\| &\leq \|r_i\| + 2\lambda_M^2 \|p_{i-2}\|^2 \frac{\lambda_M^2 \|p_{i-2}\|^2}{\lambda_m^4 \|p_{i-2}\|^4 - (\lambda_m/2)(\lambda_m/\lambda_M)^3 \|p_{i-2}\|^2 \lambda_M^3 \|p_{i-2}\|^2} \|r_i\| \\ &= \left[1 + 4\left(\frac{\lambda_M}{\lambda_m}\right)^4\right] \|r_i\|. \end{aligned} \tag{40}$$

Finally, with Eq. (38), we can rearrange the expression of α_{k-1} in Eq. (32) as follows:

$$\begin{aligned} |\alpha_{k-1}| &\geq \frac{\|A p_{k-2}\|(\lambda_m^2 \|p_{k-2}\|^2 \|r_{k-2}\|^2 - \varepsilon_{k-2}^2 \|p_{k-2}\|^4)}{2\lambda_M^4 \|p_{k-2}\|^5} \\ &\geq \frac{\lambda_m}{2\lambda_M^4} \frac{(\lambda_m^2 \|r_{k-2}\|^2 - \varepsilon_{k-2}^2 \|p_{k-2}\|^2)}{\|p_{k-2}\|^2} \geq \frac{\lambda_m}{4\lambda_M^4} \lambda_m^2 \frac{\|r_{k-2}\|^2}{\|p_{k-2}\|^2}. \end{aligned} \tag{41}$$

Now, from Eqs. (36) and (40) we have, respectively,

$$\begin{aligned} \frac{\|r_i\|}{\|p_i\|} &\geq \frac{\lambda_m^4}{\lambda_m^4 + 2\lambda_M^4} \quad \text{step } (i-1)_A, \\ \frac{\|r_i\|}{\|p_i\|} &\geq \frac{\lambda_m^4}{\lambda_m^4 + 4\lambda_M^4} \quad \text{step } (i-2)_B. \end{aligned} \tag{42}$$

Therefore, if the step k_B is preceded, respectively, by step $(k-1)_A$ or step $(k-2)_B$, relations (34), (36), (40), and (41) yield

$$|p_{k+1}^{\mathrm{T}} A p_{k-1}| \leq \begin{cases} \dfrac{\varepsilon_k^A}{\lambda_m} \dfrac{|p_{k-1}^{\mathrm{T}} A p_{k-1}|}{\|r_{k-1}\|^2} \|p_k\|^2 \\ \dfrac{\varepsilon_k^B}{\lambda_m} \dfrac{4\lambda_M^4}{\lambda_m^3} \dfrac{\|p_{k-2}\|^2}{\|r_{k-2}\|^2} \|p_k\|^2, \end{cases}$$

or, equivalently, from Eqs. (35), (38) and (42)

$$|p_{k+1}^{\mathrm{T}} A p_{k-1}| \leq \begin{cases} \left(\dfrac{\lambda_m^4}{2\lambda_M^3}\right) \dfrac{\lambda_M}{\lambda_m} \dfrac{\|p_{k-1}\|^2}{\|r_{k-1}\|^2} \|p_k\|^2 \leq \rho 1_k \|r_k\|^2 \\ \rho 2_k \|r_k\|^2, \end{cases} \tag{43}$$

where coefficients $\rho 1_k$ and $\rho 2_k$ are defined by (recall that $\|r_k\|/\|p_k\| \le 1$ for any k)

$$\rho 1_k = \frac{\lambda_m^3}{2\lambda_M^2}\left(\frac{\lambda_m^4 + 4\lambda_M^4}{\lambda_m^4}\right)^2\left(\frac{\lambda_m^4 + 2\lambda_M^4}{\lambda_m^4}\right)^2$$

$$\rho 2_k = 2\min\left\{\left(\frac{\lambda_m}{\lambda_M}\right)^3, 2^{1/2}\frac{\|r_k\|}{\|p_k\|}\right\}\frac{\lambda_M^4}{\lambda_m^3}\left(\frac{\lambda_m^4 + 4\lambda_M^4}{\lambda_m^4}\right)^4.$$

This completes the proof of Theorem 4.1