Comput Optim Appl (2007) 38: 81–104
DOI 10.1007/s10589-007-9034-z

Iterative computation of negative curvature directions in large scale optimization

Giovanni Fasano · Massimo Roma

Published online: 17 May 2007

Abstract In this paper we deal with the iterative computation of negative curvature directions of an objective function, within large scale optimization frameworks. In particular, suitable directions of negative curvature of the objective function represent an essential tool, to guarantee convergence to second order critical points. However, an "adequate" negative curvature direction is often required to have a good resemblance to an eigenvector corresponding to the smallest eigenvalue of the Hessian matrix. Thus, its computation may be a very difficult task on large scale problems. Several strategies proposed in literature compute such a direction relying on matrix factorizations, so that they may be inefficient or even impracticable in a large scale setting. On the other hand, the iterative methods proposed either need to store a large matrix, or they need to rerun the recurrence.

On this guideline, in this paper we propose the use of an iterative method, based on a planar Conjugate Gradient scheme. Under mild assumptions, we provide theory for using the latter method to compute adequate negative curvature directions, within optimization frameworks. In our proposal any matrix storage is avoided, along with any additional rerun.

Keywords Conjugate gradient method · Large scale optimization · Negative curvature directions · Convergence to second order critical points

G. Fasano (✉) · M. Roma
Dipartimento di Informatica e Sistemistica "A. Ruberti", Università di Roma "La Sapienza",
via Buonarroti, 12, Roma, Italy
e-mail: fasano@dis.uniroma1.it

M. Roma
e-mail: roma@dis.uniroma1.it

G. Fasano
Istituto Nazionale per Studi ed Esperienze di Architettura Navale INSEAN,
via di Vallerano, 139, Roma, Italy
e-mail: g.fasano@insean.it

1 Introduction

In this paper we tackle a general problem which arises in the definition of many minimization methods: the computation of suitable negative curvature directions of the objective function, in large scale settings. Our interest is mainly motivated by the possibility of using negative curvature directions, within linesearch based truncated Newton methods, for large scale unconstrained optimization. However, the approach we propose may be exploited in many different contexts of nonlinear programming.

We recall that, given a twice continuously differentiable real valued function f and the point $x \in \mathbb{R}^n$, the vector $d \in \mathbb{R}^n$ is a direction of negative curvature at x if $d^T \nabla^2 f(x) d < 0$. The use of negative curvature directions in designing minimization methods has a twofold importance: from the theoretical point of view, convergence towards second order critical points, i.e. stationary points where the Hessian matrix is positive semidefinite, may be guaranteed. From the computational point of view, observe that negative curvature directions exploit the local nonconvexities of the objective function. Then, a descent negative curvature direction is such that the objective function and its directional derivative are (locally) decreasing along it. Therefore, a step along a negative curvature direction, can hasten the search for a region where the objective function is convex.

The use of negative curvature directions in unconstrained optimization goes back up to [22] and the two landmark papers [21, 23], where linesearch based modified Newton methods were proposed. Here, a curvilinear path obtained by combining a Newton-type direction and a negative curvature direction was considered. Moreover a "second order" Armjio-type rule was used, to ensure the convergence to a second order critical point. The second order convergence can be ensured by exploiting the local curvatures of the objective function, contained in the second order derivatives. In particular, the negative curvature direction is required to resemble the eigenvector corresponding to the smallest eigenvalue $\lambda^{\min}(\nabla^2 f(x))$ of the Hessian matrix. Formally, if x_j denotes the current iterate of the algorithm in hand, the negative curvature direction d^j must be nonascent and such that

$$(d^j)^T \nabla^2 f(x_j) d^j \to 0 \quad \text{implies} \quad \min\left[0, \lambda^{\min}\left(\nabla^2 f(x_j)\right)\right] \to 0. \tag{1.1}$$

Note that convergence to second order critical points is guaranteed also for the trust region methods in unconstrained optimization (see, e.g. [3, 24, 28]): the latter result strongly motivates the interest for such methods.

The use of appropriate directions of negative curvature plays a key role also in constrained optimization, when defining algorithms with convergence to points satisfying the second order KKT necessary optimality conditions (see, e.g. [6] and the references reported therein).

The computation of a nonascent direction of negative curvature d^j, which satisfies (1.1), is a very difficult task, and the difficulty increases with the dimension. Indeed, in principle the computation of d^j is equivalent to compute an eigenvector corresponding to the smallest eigenvalue of the Hessian matrix. In [23] the Bunch and Parlett factorization was proposed for determining a negative curvature direction which satisfies (1.1). It was proved that the Bunch and Parlett decomposition, for a symmetric indefinite matrix, can be used for computing both a Newton-type

direction and an adequate negative curvature direction. In [12] this approach was embedded in a nonmonotone framework, showing how its potentiality may be further exploited. In both cases, the computation of the negative curvature direction relies on matrix factorizations, which are impracticable when dealing with large scale problems. On the basis of this observation, in [19] a new strategy based on an iterative algorithm was proposed, within truncated Newton schemes. The computation was based on the well known ability of the Lanczos algorithm, to efficiently determine extreme eigenpairs of an indefinite matrix. In particular, the SYMMLQ algorithm [26] was successfully used. However, here at step $h \leq n$ the negative curvature direction d^j in (1.1) is computed as $d^j = V_h w_h$, where the h columns of V_h are the Lanczos vectors and $w_h \in \mathbb{R}^h$. Therefore, the storage of matrix V_h is required, and in order to handle the large scale case, only a limited number of Lanczos vectors are stored. Of course, this implies that d^j is only approximately evaluated, and the second order convergence is no longer guaranteed.

In [15] a new approach for managing negative curvature directions, in large scale unconstrained optimization, was introduced. In particular, a truncated Newton approach was considered, where the alternate use of a Newton-type direction and a negative curvature direction was proposed. Then, an appropriate linesearch was performed on the chosen direction. To ensure condition (1.1) the strict connection between the Conjugate Gradient (CG) and the Lanczos methods is exploited. Convergence to second order points is proved under mild assumptions, in such a way that the storage of any matrix is avoided. However, the price to pay is the necessity of rerunning the recurrence, in order to regenerate the Lanczos vectors whenever they are needed, similarly to the truncated Lanczos approach in [14].

A similar approach for iteratively computing a direction of negative curvature was proposed in [2], where the authors discuss the possibility of using Lanczos, Chebyshev and two-step Lanczos algorithms. As alternative to the storage of all the Lanczos vectors, they propose to store the two most recent vectors and to rerun the whole Lanczos process whenever necessary.

These considerations indicate the need of iterative methods, which compute nonascent negative curvature directions satisfying (1.1), *without storing any matrix*. In this paper we focus on a variant of one of the most popular Krylov-based method, the CG method. As well known, it plays a central role in many implementations of truncated Newton methods, due to its efficiency in determining a Newton-type direction. Unfortunately the CG algorithm could untimely stop in the nonconvex case. To overcome this drawback, the use of *planar conjugate gradient* algorithms has been proposed [1, 5, 9, 10, 17, 18, 22]. Planar schemes are an extension of the linear CG method to the indefinite case. They iteratively perform the search of critical points either on mutually conjugate directions or planes, so that they are effective both in the definite and in the indefinite case. Such methods can be naturally embedded within large scale unconstrained optimization, however they represent an important tool also within other nonlinear optimization frameworks.

In this paper we propose the use of the planar CG method FLR, described in [9], in order to iteratively compute, at each iteration of a truncated Newton method, a negative curvature direction which satisfies condition (1.1), without requiring to store any matrix and avoiding any rerun.

In particular, given an $n \times n$ real indefinite matrix A, we use the FLR algorithm to generate a tridiagonal decomposition of matrix A. We show that the tridiagonal matrix obtained is similar to the tridiagonal matrix generated by the Lanczos algorithm. Therefore, we prove a suitable relation between the eigenvalues of the tridiagonal matrix given by the FLR algorithm and the matrix A. Finally, the tridiagonal form is exploited for iteratively computing, under mild assumptions, a negative curvature direction satisfying condition (1.1).

The paper is organized as follows: in Sect. 2, we report the planar CG algorithm FLR for solving the indefinite linear system $As = b$. Then, we describe the reduction of A to tridiagonal form, and the relation between the eigenvalues of the tridiagonal matrix and the matrix A. In Sect. 3, we describe the details of the iterative computation of negative curvature directions for the matrix A, by means of the tridiagonal decomposition previously obtained. In Sect. 4 we report the results of a preliminary numerical testing. Finally, a section of concluding remarks completes the paper.

As regards the notations, given a vector $v \in \mathbb{R}^n$ we denote by $\|v\|$ the 2-norm of the vector v. Given a $n \times n$ symmetric matrix A, we denote by $\lambda^{\min}(A)$ the smallest eigenvalue of A and by $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$, the smallest and the largest *modulus* of an eigenvalue of A, respectively. Moreover, with $\mathcal{K}_k(A, r)$ we indicate the k-dimensional Krylov subspace associated to the pair (A, r), i.e. the set spanned by vectors $\{r, Ar, \ldots, A^{k-1}r\}$. With e_k we indicate the unit vector with 1 in the kth entry, and $\lfloor x \rfloor$ is the lower integer part of the real value x. In the end, $\kappa(B)$ indicates the condition number of matrix B.

2 Tridiagonalizations and current representations

Let us consider the *indefinite* and *nonsingular* matrix $A \in \mathbb{R}^{n \times n}$, where n is *large*. We are concerned with determining a matrix $R_h \in \mathbb{R}^{n \times h}$ and a tridiagonal matrix $T_h \in \mathbb{R}^{h \times h}$, such that

$$AR_h = R_h T_h, \quad h \leq n. \tag{2.1}$$

To this aim, since n is large, direct methods are not appropriate and iterative methods are usually used. In particular, let $b \in \mathbb{R}^n$, then several methods based on the Krylov subspace $\mathcal{K}_h(A, b)$ (see e.g., [13]) can be considered. After $h \leq n$ steps, some of the latter methods provide h orthogonal vectors, say $r_1, \ldots, r_h$, to be used to form the columns of matrix R_h, i.e. $R_h \equiv (r_1/\|r_1\| \ \ldots \ r_h/\|r_h\|)$. Assuming $r_i \neq 0$, $i = 1, \ldots, h$, $r_{h+1} = 0$, the latter methods give (2.1), and since A is nonsingular, T_h is a tridiagonal *irreducible* matrix. If $r_{h+1} \neq 0$, in place of (2.1) the following relation holds [13]:

$$AR_h = R_h T_h + \rho_{h+1} r_{h+1} e_h^T \tag{2.2}$$

where $\rho_{h+1} \in \mathbb{R}$. In other words, (2.2) is obtained from (2.1), by adding the rank-one update $\rho_{h+1} r_{h+1} e_h^T$. Of course, since $r_1, \ldots, r_{h+1}$ are orthogonal, (2.2) leads to

$$R_h^T A R_h = T_h. \tag{2.3}$$

Observe that if $h = n$, R_h is an orthogonal matrix, and (2.3) represents a *tridiagonal decomposition* of A (it is well known the importance of reducing the symmetric

matrix A to the tridiagonal form T_h). Relation (2.2) will be referred in the sequel as *current representation* of the symmetric matrix A.

The *CG* and the *Lanczos* methods are among the most commonly used Krylov subspace methods to give (2.3). In the case of the CG method, the vector r_i, $i \leq n$, is the residual generated at iteration $i-1$. The Lanczos method directly calculates orthonormal vectors (the *Lanczos vectors*) which can be used as columns of the matrix R_h. The two methods, along with their equivalence, have been deeply studied (see, e.g., [3, Sect. 5.2], [4, 13, 29]). We only recall that if A is positive definite, then T_h is positive definite too, so that the tridiagonal matrix T_h can be *stably factorized* in the form

$$T_h = L_h D_h L_h^T \tag{2.4}$$

where L_h is a unit lower bidiagonal matrix and D_h is diagonal. Furthermore, the entries of the matrix L_h and the diagonal elements of the matrix D_h can be easily recovered from the quantities generated by the CG or the Lanczos algorithms (see, e.g., [29]).

If A is indefinite, the factorization (2.4) may fail—in the sense that it may not exist or may be very unstable—and a stable *indefinite factorization* must be alternatively considered. Such a decomposition is based on factorizing the tridiagonal matrix T_h in the form

$$T_h = L_h B_h L_h^T \tag{2.5}$$

where L_h is a unit lower bidiagonal matrix, while B_h is a *block diagonal*—and no longer a diagonal matrix. Each diagonal block of B_h is 1×1 or 2×2, hence the procedure to obtain the decomposition (2.5) is definitely more cumbersome.

In this paper, in order to obtain relations (2.2) and (2.3), we propose the use of the *planar-CG algorithm* FLR [9], which is a modification of the planar CG algorithm proposed in [17]. It is an extension of the standard CG algorithm to the indefinite case; moreover it enables to overcome the well known drawbacks due to possible pivot breakdowns.

2.1 A planar-CG algorithm

In this section we briefly describe the planar CG algorithm FLR proposed in [9] and reported in Table 1, which will be used to obtain relations (2.1–2.3). This scheme is a modification of the algorithm proposed by Hestenes in [17]. An extensive description of the FLR algorithm can be found in [9, 10]; here we report some new related results, which arise from the application of the FLR algorithm within optimization frameworks. (We highlight that the role of the logical variable *CR*, introduced in Table 1, will be clarified in Sect. 3.3.)

Firstly, suppose the matrix A is positive definite; as long as at Step k we have $\epsilon_k \leq \lambda^{\min}(A)$, the planar CG Step k_B is never performed, therefore the algorithm reduces to the standard CG. On indefinite linear systems the FLR algorithm overcomes the well known drawback of the standard CG, which may untimely stop. The latter result is accomplished by detecting the critical point of the associated quadratic function $f(x) = 1/2x^T Ax - b^T x$ on mutually conjugate directions or planes. This is a common feature of all the *planar CG methods* (see [9] and the references reported

Table 1 Algorithm FLR for solving the linear system $As = b$

Algorithm FLR

Step 1: $k = 1$, $s_1 = 0$, $r_1 = b$, $CR = false$. If $r_1 = 0$ then $CR = true$ and STOP,
else compute $p_1 = r_1$.

Step k: Compute $\sigma_k = p_k^T A p_k$.
If $|\sigma_k| \geq \epsilon_k \|p_k\|^2$ then go to *Step* k_A else go to *Step* k_B

– **Step** k_A *(standard CG step)*:
Set $s_{k+1} = s_k + a_k p_k$, $r_{k+1} = r_k - a_k A p_k$, where $a_k = \frac{r_k^T p_k}{\sigma_k}$.
If $r_{k+1} = 0$ then $CR = true$ and STOP
else compute $p_{k+1} = r_{k+1} + \beta_k p_k$ with $\beta_k = \frac{-p_k^T A r_{k+1}}{\sigma_k} = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$.
Set $k = k + 1$ and go to *Step* k.

– **Step** k_B *(planar CG step)*:
If $k = 1$ then compute the vector $q_k = A p_k$,
else compute the vector

$$q_k = \begin{cases} A p_k + b_{k-1} p_{k-1}, & \text{if the previous step is } \textit{Step } (k-1)_A \\ A p_k + \frac{\hat{b}_{k-2}}{\Delta_{k-2}}(\sigma_{k-2} q_{k-2} - \delta_{k-2} p_{k-2}), & \\ \quad \text{if the previous step is } \textit{Step } (k-2)_B & \end{cases}$$

where $b_{k-1} = -(A p_{k-1})^T A p_k / \sigma_{k-1}$ and $\hat{b}_{k-2} = -(A q_{k-2})^T A p_k$.
Compute $c_k = r_k^T p_k$, $\delta_k = p_k^T A q_k$, $e_k = q_k^T A q_k$, $\Delta_k = \sigma_k e_k - \delta_k^2$
and $\hat{c}_k = (c_k e_k - \delta_k q_k^T r_k)/\Delta_k$, $\hat{\sigma}_k = (\sigma_k q_k^T r_k - \delta_k c_k)/\Delta_k$.
Set $s_{k+2} = s_k + \hat{c}_k p_k + \hat{\sigma}_k q_k$, $r_{k+2} = r_k - \hat{c}_k A p_k - \hat{\sigma}_k A q_k$.
If $r_{k+2} = 0$ then $CR = true$ and STOP
else compute $p_{k+2} = r_{k+2} + \frac{\hat{\beta}_k}{\Delta_k}(\sigma_k q_k - \delta_k p_k)$ with $\hat{\beta}_k = -q_k^T A r_{k+2}$.
Set $k = k + 2$ and go to *Step* k.

therein). In particular, assuming that the matrix A is indefinite and nonsingular, by applying the standard CG, pivot breakdown occurs at Step k if $p_k^T A p_k = 0$, so that the iterates are terminated prematurely. On the contrary, planar CG methods generate another direction at Step k_B (denoted by q_k). Then, instead of detecting the critical point along the line $x_k + \alpha p_k$, $\alpha \in \mathbb{R}$ (which is carried on at Step k_A), they perform a search on the 2-dimensional linear manifold $x_k + \text{span}\{p_k, q_k\}$. More specifically, as concerns the FLR algorithm, it can be easily proved (see Lemma 2.2 in [9]) that, if the matrix A is nonsingular and at Step k we have $r_k \neq 0$, the FLR algorithm can always perform either Step k_A or Step k_B. For further properties of the sequences $\{p_k\}$ and $\{q_k\}$ we refer to Theorem 2.1 in [9]. As regards the parameter ϵ_k at the Step k

(see also [9]), let $\bar{\epsilon} \in \mathbb{R}$ be positive, then the choice

$$\begin{aligned} &\bar{\epsilon} < \epsilon_k, \quad \text{at Step } k_A, \\ &\bar{\epsilon} < \epsilon_k \leq \hat{\epsilon}_k = \min\left\{ \frac{2}{3}\lambda_{\min}(A)\frac{\|r_k\|}{\|p_k\|},\ \lambda^2_{\max}(A)\frac{\|p_k\|}{\|q_k\|},\ \frac{\lambda^4_{\min}(A)}{2\lambda_{\max}(A)}\frac{\|p_k\|^2}{\|q_k\|^2} \right\}, \quad (2.6) \\ &\text{at Step } k_B, \end{aligned}$$

guarantees the denominators at Step k_A and Step k_B to be sufficiently bounded away from zero. We highlight that the choice (2.6) is slightly stronger than the choice of the parameter ϵ_k at Step k of the FLR algorithm in [9]. As regards the apparently cumbersome computation of the quantity $\|q_k\|$ in (2.6), no additional cost is needed (see also [9]). In the sequel we assume that at Step k the parameter ϵ_k satisfies condition (2.6).

2.2 Tridiagonalizations and current representations via the FLR algorithm

In this section we describe how to obtain a tridiagonal decomposition and a current representation of the matrix A, by means of the FLR algorithm. As regards the sequence of the directions generated by this algorithm, we adopt the following notation: if at Step k the condition $|p_k^T A p_k| \geq \epsilon_k \|p_k\|^2$ is satisfied, then set $w_k = p_k$ (standard CG step) otherwise set $w_k = p_k$ and $w_{k+1} = q_k$ (planar CG step). With the latter convention, $\{w_i\}$ represents the sequence of directions generated by the FLR algorithm.

Observe that the sequence of the residuals $\{r_i\}$ is necessary to form the matrix R_h in (2.3); however, note that in the planar CG Step k_B, the vector r_{k+1} is not generated. The latter shortcoming may be overcome by introducing a "dummy" residual r_{k+1}, which completes the sequence of orthogonal vectors $r_1, \ldots, r_k, r_{k+1}, r_{k+2}, \ldots, r_h$ [1, 7]. It is easy to see that the only possible choice (apart from a scale factor) for the dummy residual r_{k+1}, such that $r_{k+1} \in \mathcal{K}_k(A, r_1)$ and $r_{k+1} \notin \mathcal{K}_{k-1}(A, r_1)$, is the following [7]:

$$r_{k+1} = \hat{\alpha}_k r_k + (1 + \hat{\alpha}_k)\,\mathrm{sgn}(\sigma_k) A p_k, \quad \hat{\alpha}_k = -\frac{|\sigma_k|}{\|r_k\|^2 + |\sigma_k|} \tag{2.7}$$

where the coefficient $\hat{\alpha}_k$ is computed by imposing the orthogonality condition $r_{k+1}^T p_k = r_{k+1}^T r_k = 0$. We highlight that since $\|r_k\|$ and $\|Ap_k\|$ are bounded, $\|r_{k+1}\|$ is bounded too. Moreover, from (2.7) and [9, Theorem 2.1], it can be readily seen that the dummy residual r_{k+1} satisfies also the required orthogonality properties

$$r_{k+1}^T r_i = 0, \quad i \leq k, \quad \text{and} \quad r_i^T r_{k+1} = 0, \quad i > k+1.$$

Now, we show that the FLR algorithm yields both the tridiagonalization (2.3) and the current representation (2.2), when matrix A is indefinite. To this aim suppose that the FLR algorithm performs up to step h and, for the sake of simplicity, *the only one planar CG step is Step* $k_B < h$. Then, considering (2.7) and the instructions at Step k_B, along with the position

$$R_h = \left(\frac{r_1}{\|r_1\|} \cdots \frac{r_h}{\|r_h\|} \right) \in \mathbb{R}^{n \times h}, \qquad P_h = \left(\frac{w_1}{\|r_1\|} \cdots \frac{w_h}{\|r_h\|} \right) \in \mathbb{R}^{n \times h},$$

the following result can be obtained:

Theorem 2.1 *Consider the* FLR *algorithm and let* $\|r_i\| \neq 0$, $i \leq h$. *Suppose the only one planar CG step is Step* $k_B < h$. *Then the following relations hold*:

$$P_h \tilde{L}_h^T = R_h, \tag{2.8}$$

$$AP_h = \left(R_h \quad \vdots \quad \frac{r_{h+1}}{\|r_{h+1}\|} \right) \begin{pmatrix} \bar{L}_h \\ \bar{l}_{h+1,h} e_h^T \end{pmatrix} \bar{D}_h, \quad h < n, \tag{2.9}$$

$$AR_h = \left(R_h \quad \vdots \quad \frac{r_{h+1}}{\|r_{h+1}\|} \right) \begin{pmatrix} \bar{T}_h \\ \bar{t}_{h+1,h} e_h^T \end{pmatrix}, \qquad h < n, \tag{2.10}$$

where

$$\tilde{L}_h = \begin{pmatrix} 1 & & & & & & & & \\ -\sqrt{\beta_1} & \cdot & & & & & & & \\ & \cdot & 1 & & & & & & \\ & & -\sqrt{\beta_{k-1}} & 1 & & & 0 & & \\ & & & \tilde{\alpha}_1 & \tilde{\alpha}_2 & & & & \\ & & & \tilde{\alpha}_3 & \tilde{\alpha}_4 & 1 & & & \\ & 0 & & & & -\sqrt{\beta_{k+2}} & \cdot & & \\ & & & & & & \cdot & 1 & \\ & & & & & & & -\sqrt{\beta_{h-1}} & 1 \end{pmatrix} \tag{2.11}$$

with ($\beta_k = \|r_{k+1}\|^2/\|r_k\|^2$, $\beta_{k+1} = \|r_{k+2}\|^2/\|r_{k+1}\|^2$)

$$\begin{aligned} \tilde{\alpha}_1 &= \frac{\hat{\alpha}_k}{\sqrt{\beta_k}}, & \tilde{\alpha}_2 &= (1 + \hat{\alpha}_k)\,\mathrm{sgn}(\sigma_k), \\ \tilde{\alpha}_3 &= \frac{\hat{\beta}_k \delta_k}{\Delta_k \sqrt{\beta_{k+1}\beta_k}}, & \tilde{\alpha}_4 &= -\frac{\hat{\beta}_k\, \sigma_k}{\Delta_k \sqrt{\beta_{k+1}}}, \end{aligned} \tag{2.12}$$

$$\bar{D}_h = \begin{pmatrix} \frac{1}{a_1} & & & & & & & \\ & \cdot & & & & & & \\ & & \frac{1}{a_{k-1}} & & & & 0 & \\ & & & \frac{1}{\xi_k} & & & & \\ & & & & \frac{1}{\xi_{k+1}} & & & \\ & 0 & & & & \frac{1}{a_{k+2}} & & \\ & & & & & & \cdot & \\ & & & & & & & \frac{1}{a_h} \end{pmatrix}, \tag{2.13}$$

$$\bar{L}_h = \begin{pmatrix} 1 & & & & & & & & \\ -\sqrt{\beta_1} & \cdot & & & & & & & \\ & \cdot & 1 & & & 0 & & & \\ & & -\sqrt{\beta_{k-1}} & \bar{\alpha}_1 & \bar{\alpha}_3 & & & & \\ & & & \bar{\alpha}_2 & \bar{\alpha}_4 & & & & \\ & & & & \bar{\alpha}_5 & 1 & & & \\ & & 0 & & & -\sqrt{\beta_{k+2}} & \cdot & & \\ & & & & & & \cdot & 1 & \\ & & & & & & & -\sqrt{\beta_{h-1}} & 1 \end{pmatrix}. \tag{2.14}$$

The coefficients ξ_k and ξ_{k+1} are independent arbitrary non-zero parameters[1]*, and $\bar{\alpha}_i$, $i = 1, \ldots, 5$, have the following values*:

$$\begin{aligned} \bar{\alpha}_1 &= \frac{\sigma_k}{\|r_k\|^2}\xi_k, & \bar{\alpha}_2 &= \sqrt{\beta_k}\left[\operatorname{sgn}(\sigma_k) + \frac{\sigma_k}{\|r_k\|^2}\right]\xi_k, \\ \bar{\alpha}_3 &= \frac{\xi_{k+1}}{\sqrt{\beta_k}\hat{\sigma}_k}\left[1 - \frac{\sigma_k}{\|r_k\|^2}\hat{c}_k\right], & \bar{\alpha}_4 &= -\frac{\hat{c}_k\xi_{k+1}}{\hat{\sigma}_k(1+\hat{\alpha}_k)\operatorname{sgn}(\sigma_k)}, \\ \bar{\alpha}_5 &= -\frac{\xi_{k+1}}{\hat{\sigma}_k}\sqrt{\beta_{k+1}}. \end{aligned} \tag{2.15}$$

Finally, $\bar{T}_h$ is an irreducible symmetric tridiagonal matrix defined by

$$\begin{pmatrix} & & \bar{T}_h & \\ 0 & \cdots & 0 & \bar{t}_{h+1,h} \end{pmatrix} = \begin{pmatrix} & & \bar{L}_h & \\ 0 & \cdots & 0 & \bar{l}_{h+1,h} \end{pmatrix} \bar{D}_h \tilde{L}_h^T \tag{2.16}$$

where $\bar{l}_{h+1,h}$ and $\bar{t}_{h+1,h}$ are the element $(h+1, h)$ of the matrix $\bar{L}_{h+1}$ and $\bar{T}_{h+1}$, respectively.

Proof From the FLR algorithm, [9, Theorem 2.1] and (2.7), after some calculations and assuming that the only one planar CG step is Step $k_B < h$, we obtain (2.8, 2.9) and the expression of $\tilde{L}_h$, $\bar{D}_h$ and $\bar{L}_h$. Moreover, we have

$$\begin{aligned} AR_h &= \begin{pmatrix} R_h & \vdots & \dfrac{r_{h+1}}{\|r_{h+1}\|} \end{pmatrix} \begin{pmatrix} & & \bar{T}_h & \\ 0 & \cdots & 0 & \bar{t}_{h+1,h} \end{pmatrix} \\ &= \begin{pmatrix} R_h & \vdots & \dfrac{r_{h+1}}{\|r_{h+1}\|} \end{pmatrix} \begin{pmatrix} \bar{T}_h \\ \bar{t}_{h+1,h} e_h^T \end{pmatrix}. \end{aligned}$$

□

Note that (2.10) can be rewritten in the form

$$AR_h = R_h\bar{T}_h + \frac{\bar{t}_{h+1,h}}{\|r_{h+1}\|} r_{h+1} e_h^T, \tag{2.17}$$

[1] We remark that $\hat{\sigma}_k \neq 0$ in the FLR algorithm, thus the parameters ξ_k and ξ_{k+1} may respectively be set to 1 and $\hat{\sigma}_k$, in order to simplify (2.15).

which is a current representation of the indefinite nonsingular matrix A. The following result provides further theoretical properties of the matrix $\tilde{L}_h$, which will be used later on.

Proposition 2.2 *Consider the* FLR *algorithm and let* $\|r_i\| \neq 0$, $i \leq h$, *where* $h < n$. *Suppose that* $\lambda_{\min}(A) \geq \gamma > 0$, $\gamma \in \mathbb{R}$, *and that* $\nu \geq 0$ *planar CG steps are performed. Let the parameter* ϵ_k *satisfy condition* (2.6), *with* $k \leq h$. *Then, the matrix* $\tilde{L}_h$ *in* (2.11) *is nonsingular and* $\|\tilde{L}_h\|$ *is bounded. Moreover, the following relations hold*

$$\left(\frac{1}{1+4\lambda_{\min}^2(A)/(9\bar{\epsilon})}\right)^{\nu} \leq |\det(\tilde{L}_h)| \leq 1, \tag{2.18}$$

$$\|\tilde{L}_h\| \leq \tilde{\ell}_{\max}\left[(2h-1)+\frac{h}{2}\right]^{1/2} \quad \textit{where} \tag{2.19}$$

$$\tilde{\ell}_{\max} = \max_{\substack{k\in\{j_1,\ldots,j_\nu\} \\ i=1,\ldots,h-1,\ i\neq k,k+1}} \left\{\sqrt{\beta_i}, 2, 2\sqrt{\beta_{k+1}\beta_k}, 2\frac{\hat{\epsilon}_k}{\lambda_{\min}^2(A)}\beta_k\sqrt{\beta_{k+1}}\right\}, \tag{2.20}$$

$\tilde{\ell}_{\max}$ *is bounded,* $j_1, \ldots, j_\nu$ *are the indices of the planar CG steps and* $j_p \leq h-1$, $p = 1, \ldots, \nu$.

For the sake of brevity, we do not report here the proof of this result for which we refer to [11].

As we already discussed, several methods may be used to reduce the matrix A to a tridiagonal form. Furthermore, it can be worthwhile to point out to what extent the tridiagonal matrix generated depends on the method. On this guideline the Theorem 7-2-2 in [27] allows us to immediately derive the following proposition, concerning the FLR algorithm.

Proposition 2.3 *Let* $A \in \mathbb{R}^{n\times n}$ *be symmetric and indefinite. Suppose to perform the* FLR *algorithm and generate the orthogonal matrix* $R_n = (r_1/\|r_1\| \ \ldots \ r_n/\|r_n\|)$, *along with the irreducible tridiagonal matrix* $\bar{T}_n$, *such that* $AR_n = R_n\bar{T}_n$. *Then,* $\bar{T}_n$ *and* R_n *are uniquely determined by* A *and* $r_1/\|r_1\|$ *or, alternatively, by* A *and* $r_n/\|r_n\|$.

2.3 Relationship between the eigenvalues of matrices $\bar{T}_h$ and A

Suppose now to apply the Lanczos algorithm to the linear system $As = b$, with $b \in \mathbb{R}^n$. Then, after n steps the Lanczos algorithm generates the orthogonal matrix $V_n = (v_1, \ldots, v_n)$ and the tridiagonal matrix T_n, such that $AV_n = V_nT_n$. Similarly to Proposition 2.3, by Theorem 7-2-2 in [27] we have that T_n and V_n are uniquely determined by A and v_1, or alternatively by A and v_n. However, this does not imply that the tridiagonal matrices T_n (from the Lanczos process) and $\bar{T}_n$ (from the FLR algorithm) coincide. It rather suggests that the vectors $r_i/\|r_i\|$ and v_i, $i \leq n$, are parallel. Thus, the sequences $\{v_i\}$ and $\{r_i/\|r_i\|\}$ are orthonormal bases of the same Krylov subspace, and $v_i = \theta_i\ r_i/\|r_i\|$, with $\theta_i \in \{-1, +1\}$. The sequence $\{\theta_i\}$ is uniquely defined. In particular, in case no planar CG steps are performed (the FLR algorithm reduces to the standard CG), the detailed calculus of θ_i can be found, e.g., in [3, Sect. 5.2.2].

When planar CG steps are performed in the FLR algorithm, the expression of θ_i is reported in [8]. This implies that the following relation holds

$$T_h = V_h^T A V_h = \Theta_h^T R_h^T A R_h \Theta_h = \Theta_h \bar{T}_h \Theta_h, \quad h \le n, \tag{2.21}$$

where $\Theta_h = \mathrm{diag}\{\theta_i,\ i = 1, \ldots, h\}$ is nonsingular. Due to the similarity (2.21), some important relations between the eigenvalues of matrices $\bar{T}_h$ and A hold from the relation between T_h and A. Therefore, properties on the eigenvalues of $\bar{T}_h$ can be directly inherited from the properties of the eigenvalues of T_h. We now briefly describe the relation between the smallest eigenvalue of the tridiagonal matrix $\bar{T}_h$ in (2.16), and the eigenvalues of matrix A. This will provide a tool for estimating the eigenvector of A associated to the smallest eigenvalue of matrix A. Then, the latter vector will be used to compute a suitable negative curvature direction in optimization frameworks.

Theorem 2.4 *Let $\lambda_1(A) \le \cdots \le \lambda_n(A)$ be the ordered eigenvalues of the matrix A. Moreover, let $\mu_1(\bar{T}_h) \le \cdots \le \mu_h(\bar{T}_h)$ be the ordered eigenvalues of the matrix $\bar{T}_h$. Then the following statements hold.*

(i) *The inequality $\lambda_1(A) \le \mu_1(\bar{T}_h) \le \lambda_1(A) + \frac{C}{\phi_{h-1}^2(1+2\rho)}$, holds where C is a positive scalar, ϕ_{h-1} is the $h-1$ degree Chebyshev polynomial and $\rho = (\lambda_2(A) - \lambda_1(A))/(\lambda_n(A) - \lambda_2(A))$.*

(ii) *Let $r_h \neq 0$ and $r_{h+1} = 0$, then the eigenvalues $\mu_1(\bar{T}_h) \le \cdots \le \mu_h(\bar{T}_h)$ of the matrix $\bar{T}_h$ are also eigenvalues of matrix A, and $\mu_1(\bar{T}_h) < \mu_1(\bar{T}_{h-1}) < \cdots < \mu_{h-1}(\bar{T}_{h-1}) < \mu_h(\bar{T}_h)$. Moreover if $u_i^{(h)}$ is the eigenvector of $\bar{T}_h$ corresponding to the eigenvalue $\mu_i(\bar{T}_h)$, then the vector $R_h u_i^{(h)}$ is an eigenvector of the matrix A.*

(iii) *There exists an eigenvalue $\lambda(A) \in \mathbb{R}$ of matrix A such that*

$$\left|\lambda(A) - \mu_i(\bar{T}_h)\right| \le |\bar{t}_{h+1,h}|.$$

Proof It is well known that properties (i) and (ii) hold if we consider the tridiagonal matrix T_h generated by the Lanczos algorithm (see e.g. [13]). Hence, from the similarity (2.21) they hold for the tridiagonal matrix $\bar{T}_h$.

As regards (iii), suppose $(\mu_i(\bar{T}_h), u_i^{(h)})$ is an eigenpair of matrix $\bar{T}_h$, $i \le h$ with $\|u_i^{(h)}\| = 1$. Then, by Theorem 4-5-1 in [27] and (2.17), an eigenvalue $\lambda(A)$ of A exists such that:

$$\begin{aligned} |\lambda(A) - \mu_i(\bar{T}_h)| &\le \frac{\|A R_h u_i^{(h)} - \mu_i(\bar{T}_h) R_h u_i^{(h)}\|}{\|R_h u_i^{(h)}\|} = \left\|(A R_h - R_h \bar{T}_h) u_i^{(h)}\right\| \\ &= \left\| \bar{t}_{h+1,h} \frac{r_{h+1}}{\|r_{h+1}\|} e_h^T u_i^{(h)} \right\| \le |\bar{t}_{h+1,h}|. \end{aligned} \tag{2.22}$$

□

Observe that from property (i), the smallest eigenvalue of the tridiagonal matrix $\bar{T}_h$ approaches very quickly the smallest eigenvalue of the matrix A (see also [30, p. 279]).

Property (ii) yields $\mu_1(\bar{T}_h) < \mu_1(\bar{T}_{h-1})$ therefore, at each iteration the FLR algorithm improves the estimate of the smallest eigenvalue of the matrix A, by means of (2.17) and (2.22).

Finally, property (iii) shows that if $|\bar{t}_{h+1,h}|$ is sufficiently small, the quantity $\mu_i(\bar{T}_h)$ is a good approximation of the eigenvalue $\lambda(A)$ of matrix A (we recall that when $\bar{t}_{h+1,h} = 0$ the FLR algorithm stops).

3 Iterative computation of negative curvature directions

In this section we use the theory developed in the previous sections within truncated Newton frameworks. We recall that the latter iterative schemes have proved to be effective tools for large scale optimization problems (see, e.g., [25]). In particular, in unconstrained optimization, convergence towards second order stationary points where the Hessian is positive semidefinite, can be ensured by computing, at each iteration j, a pair of directions (s^j, d^j). The direction s^j is a Newton-type direction obtained by approximately solving the Newton's system $\nabla^2 f(x_j)\ s = -\nabla f(x_j)$, whose purpose is to guarantee the convergence to stationary points. The direction d^j is a suitable negative curvature direction used to force convergence to second order stationary points [19, 23]. This requires that the sequence $\{d^j\}$ satisfies the following assumption.

Condition A The directions $\{d^j\}$ are bounded and such that

$$(d^j)^T \nabla^2 f(x_j) d^j < 0, \tag{3.1}$$

$$(d^j)^T \nabla^2 f(x_j) d^j \to 0 \quad \text{implies} \quad \min\left[0, \lambda^{\min}(\nabla^2 f(x_j))\right] \to 0, \tag{3.2}$$

where $\lambda^{\min}(\nabla^2 f(x_j))$ is the smallest eigenvalue of the Hessian matrix $\nabla^2 f(x_j)$.

Here we describe how to compute such directions $\{d^j\}$ by using the theory developed in the previous sections. More specifically, we describe how to exploit relation (2.17), obtained in Sect. 2 via the FLR algorithm, for iteratively computing direction d^j in order to satisfy Condition A.

To this aim, we focus on a generic iteration j of a truncated Newton method and we consider the matrix A as the system matrix at the jth iteration of the method (in the unconstrained optimization case $As = b$ is the Newton's system at the jth iteration with $A = \nabla^2 f(x_j)$). Now, suppose to apply the FLR algorithm and that it terminates after n steps, i.e. the condition $r_{n+1} = 0$ is fulfilled and formulae (2.8) and (2.9) hold with $h = n$. Observe that in a large scale setting, when far from the solution, the FLR algorithm should possibly avoid to completely investigate an n-dimensional Krylov subspace. Thus, in general it may be impossible to guarantee condition (3.2) because (see [15, Sect. 4]) "*... the Krylov subspace investigated may not contain any eigenvector corresponding to the leftmost eigenvalue. However* [in a truncated Newton scheme, when close to a stationary point] *this happens with probability zero in exact arithmetic, and we don't expect it to happen in presence of rounding*." Therefore, we carry on a theoretical analysis under the assumption that the FLR algorithm performs exactly n steps, while in practice, we will compute the direction d^j after

a smaller number of steps. The numerical experience in Sect. 4 gives evidence that the latter approach is effective and efficient.

On this guideline, condition $r_{n+1} = 0$ in (2.17) yields $AR_n = R_n \bar{T}_n$. So that from (2.16), we have the following factorization for the symmetric tridiagonal matrix $\bar{T}_n$

$$\bar{T}_n = \bar{L}_n \bar{D}_n \tilde{L}_n^T \tag{3.3}$$

where $\tilde{L}_n$ and $\bar{L}_n$ are given in (2.11) and (2.14) (with $h = n$). Note that in general $\bar{L}_n \neq \tilde{L}_n$, unless no planar CG steps were performed by the FLR algorithm. Since both $\bar{L}_n$ and $\tilde{L}_n$ are nonsingular, the nonsingular matrix $\Pi_n \in \mathbb{R}^{n\times n}$ exists such that

$$\bar{L}_n = \tilde{L}_n \Pi_n. \tag{3.4}$$

Now, w.l.o.g. assume that only one planar CG step was performed and it was Step k_B; then, the explicit expression of Π_n is given by

$$\Pi_n = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & 0 & \\ & & 1 & & & \\ & & & \pi_{k,k} & \pi_{k,k+1} & \\ & & & \pi_{k+1,k} & \pi_{k+1,k+1} & \\ & & & & & 1 \\ & 0 & & & & \ddots \\ & & & & & & 1 \end{pmatrix}, \tag{3.5}$$

with

$$\begin{aligned} \pi_{k,k} &= \bar{\alpha}_1, & \pi_{k,k+1} &= \bar{\alpha}_3, \\ \pi_{k+1,k} &= \frac{\bar{\alpha}_2 - \bar{\alpha}_1\tilde{\alpha}_1}{\tilde{\alpha}_2}, & \pi_{k+1,k+1} &= \frac{\bar{\alpha}_4 - \bar{\alpha}_3\tilde{\alpha}_1}{\tilde{\alpha}_2} \end{aligned} \tag{3.6}$$

where $\tilde{\alpha}_i$ and $\bar{\alpha}_i$, $i \leq 4$, are given in (2.12) and (2.15). Hence, by (3.3) and (3.4)

$$\bar{T}_n = \tilde{L}_n (\Pi_n \bar{D}_n) \tilde{L}_n^T, \tag{3.7}$$

and it is easy to verify that the block diagonal matrix $\Pi_n \bar{D}_n$ is symmetric.

The importance of the factorization (3.7) in computing the negative curvature direction d^j, for matrix A, relies on the following facts:

- the computation of the eigenpairs of $\Pi_n \bar{D}_n$ is trivial. Furthermore it is possible to establish a simple relation between the eigenvalues of $\bar{T}_n$ and the eigenvalues of $\Pi_n \bar{D}_n$;
- since $r_i \neq 0$, $i = 1, \ldots, n$ and $r_{n+1} = 0$, the matrix $\bar{T}_n$ is irreducible and the eigenvalues $\mu_1(\bar{T}_n) < \cdots < \mu_n(\bar{T}_n)$ of $\bar{T}_n$ are also eigenvalues of matrix A (Theorem 2.4).

Now let us state the following technical proposition.

Proposition 3.1 *Consider the solution $\bar{y}$ of the linear system $\tilde{L}_n^T y = z_1$, where $\tilde{L}_n$ is given in (2.11) and z_1 is the normalized eigenvector corresponding to the smallest eigenvalue of $\Pi_n \bar{D}_n$ (see (2.13) and (3.5)). Let $\nu \geq 0$ be the number of the planar CG steps performed by the* FLR *algorithm. Let $\tilde{\ell}_{\max}$ be an upper bound for the modulus of the entries of matrix $\tilde{L}_n$. Assume that ϵ_k in the* FLR *algorithm satisfies (2.6). Then the vector $\bar{y}$ is bounded and satisfies*

$$\|\bar{y}\| \leq n 2^{\nu} (\hat{\ell}_{\max})^{n-1} \sqrt{(n-\nu) + \nu \left(1 + \frac{4}{9} \frac{\lambda_{\min}^2(A)}{\bar{\epsilon}}\right)^2}$$

where

$$\hat{\ell}_{\max} = \left(1 + \frac{4}{9} \frac{\lambda_{\min}^2(A)}{\bar{\epsilon}}\right) \tilde{\ell}_{\max}.$$

For the sake of brevity, we do not report here the proof of this result for which we refer to [11].

The next theorem, which follows from [23, Lemma 4.3], shows how to determine a direction of negative curvature by means of the decomposition (3.7).

Theorem 3.2 *Let us consider the decomposition (3.7), where $\bar{T}_n$ has at least one negative eigenvalue. Let z_1 be the normalized eigenvector corresponding to the smallest eigenvalue of $\Pi_n \bar{D}_n$, and $\bar{y} \in \mathbb{R}^n$ the solution of the linear system*

$$\tilde{L}_n^T y = z_1. \tag{3.8}$$

Then the vector

$$d^j = R_n \bar{y} \tag{3.9}$$

is a bounded negative curvature direction satisfying (3.1) and (3.2).

Proof From relation (2.3) with $h = n$ and considering that $\mu^{\min}(\bar{T}_n)$ is the smallest eigenvalue of $\bar{T}_n$, Lemma 4.3 in [23] yields

$$\begin{aligned} 0 > \mu^{\min}(\bar{T}_n) &\geq \left(\bar{y}^T \bar{T}_n \bar{y}\right) \|\tilde{L}_n\|^2 \geq \left[\kappa(\tilde{L}_n)\right]^2 \frac{(\bar{y}^T \bar{T}_n \bar{y})}{\|\bar{y}\|^2} \\ &= \left[\kappa(\tilde{L}_n)\right]^2 \frac{[(R_n \bar{y})^T A (R_n \bar{y})]}{\|R_n \bar{y}\|^2}. \end{aligned} \tag{3.10}$$

Finally, from Theorem 2.4 the smallest eigenvalue $\lambda^{\min}(A)$ of A satisfies $\lambda^{\min}(A) = \mu^{\min}(\bar{T}_n)$, then

$$0 > \lambda^{\min}(A) \geq \left[\kappa(\tilde{L}_n)\right]^2 \frac{[(R_n \bar{y})^T A (R_n \bar{y})]}{\|R_n \bar{y}\|^2}, \tag{3.11}$$

i.e. d^j is a negative curvature direction and (3.2) holds. Now we show that the direction d^j defined in (3.9) is bounded. On this purpose, considering that R_n is orthogonal, d^j is bounded as long as the solution $\bar{y}$ of the linear system (3.8) is bounded. The latter result is stated in Proposition 3.1. □

Now observe that, according to (3.9), the computation of d^j requires the storage of both the vector $\bar{y}$ and the full rank matrix R_n. Unfortunately this is unpractical for large scale problems. Here we show that both the computation of $\bar{y}$ and the storage of R_n may be avoided, so that the FLR algorithm iteratively provides the vector d^j. We distinguish between the (simpler) case in which no planar CG steps are performed, from the case where planar CG steps are performed.

3.1 No planar CG steps are performed in the FLR algorithm

If no planar CG steps are performed, in Proposition 3.1 we have $z_1 = e_m$, for a certain $m \leq n$. Moreover, the decomposition (3.3) becomes $\bar{T}_n = L_n D_n L_n^T$, where [29]

$$L_n = \begin{pmatrix} 1 & & & 0 \\ -\sqrt{\beta_1} & 1 & & \\ & \cdot & \cdot & \\ & & \cdot & 1 \\ 0 & & & -\sqrt{\beta_{n-1}} \quad 1 \end{pmatrix}$$

and $D_n = \text{diag}\{1/a_1, \ldots, 1/a_n\}$. Therefore, recalling relation $\beta_i = \|r_{i+1}\|^2/\|r_i\|^2$, $i \leq n-1$, the solution $\bar{y} = (\bar{y}_1 \ldots \bar{y}_n)^T$ of the linear system (3.8) can be obtained by backtracking from $\bar{y}_n$ to $\bar{y}_1$. In particular, we have

$$\bar{y}_n = \cdots = \bar{y}_{m+1} = 0, \qquad \bar{y}_m = 1, \quad \text{and}$$

$$\bar{y}_{m-1} = \frac{\|r_m\|}{\|r_{m-1}\|}, \qquad \bar{y}_{m-2} = \frac{\|r_m\|}{\|r_{m-2}\|}, \qquad \ldots, \qquad \bar{y}_1 = \frac{\|r_m\|}{\|r_1\|}.$$

From (3.9), recalling that $p_i = r_i + \beta_{i-1} p_{i-1}$, $i \leq n$, we simply obtain

$$d^j = \|r_m\| \sum_{i=1}^{m} \frac{r_i}{\|r_i\|^2} = \frac{p_m}{\|r_m\|}. \tag{3.12}$$

The difficulty that m is not known "a priori," can be easily overcome, since m is the index of the least negative diagonal entry of matrix D_n, i.e. $1/a_m \leq 1/a_i$, for any $a_i < 0$, $i \leq n$. Hence, from (3.12) the calculation of d^j may be easily carried on iteratively and requires the *storage of only one additional vector*.

3.2 Some planar CG steps are performed in the FLR algorithm

In case some planar CG steps are performed by the FLR algorithm, the structure of matrix $\tilde{L}_n$ in (3.8) is given in (2.11) with $h = n$. Now, we manipulate (3.7) so that

$$\tilde{L}_n = \hat{L}_n \tilde{D}_n \quad \text{and} \quad T_n = \hat{L}_n B_n \hat{L}_n^T \tag{3.13}$$

where $\tilde{D}_n$ is a nonsingular diagonal matrix and $\hat{L}_n$ is a nonsingular unit lower triangular matrix (see [11] for the exact definition of $\tilde{D}_n$ and $\hat{L}_n$). Then, $B_n = \tilde{D}_n \Pi_n \tilde{D}_n \tilde{D}_n$

is a block diagonal matrix with 1×1 and 2×2 nonsingular blocks. From Proposition 3.1 and its proof we observe that w.l.o.g., in place of solving the linear system (3.8) in Theorem 3.2, we can consider equivalently from (3.13) the system

$$\hat{L}_n^T y = z_1 \tag{3.14}$$

where now z_1 is the normalized eigenvector corresponding to the least negative eigenvalue of B_n. The vector z_1 may have two different expressions:

(1) if the smallest negative eigenvalue of B_n is a 1×1 diagonal block of matrix B_n (corresponding to a *standard CG step* in the FLR algorithm), then $z_1 = e_m$ for a certain $m \leq n$;
(2) otherwise, let λ_m, λ_{m+1} be eigenvalues of B_n, corresponding to a 2×2 diagonal block (*planar CG step* in the FLR algorithm), with $\lambda_m \leq \lambda_{m+1}$. Then, if λ_m is the smallest eigenvalue of B_n, it results $z_1 = (0, \ldots, 0, \omega_m, \omega_{m+1}, 0, \ldots, 0)^T$ where $\omega_m, \omega_{m+1} \in \mathbb{R}$ and can be trivially calculated.

Therefore, as in the previous section we want to determine the solution $\bar{y}$ of (3.14) and the direction d^j of (3.9) in either the cases above, assuming that some planar CG steps were performed.

- We first consider the case (1) in which

$$z_1 = e_m, \tag{3.15}$$

i.e. the smallest eigenvalue of B_n corresponds to a standard CG step in the FLR algorithm. By simply backtracking from $\bar{y}_n$ to $\bar{y}_1$ we obtain (see (3.14))

$$\bar{y}_n = \cdots = \bar{y}_{m+1} = 0.$$

Then, we compute $\bar{y}_i$, $i = 1, \ldots, m$, by checking whether Step i, $i = 1, \ldots, m$, was a standard CG step or a planar CG step. In particular, suppose w.l.o.g. that the FLR algorithm performed the standard CG steps $m_A, (m-1)_A, \ldots, (h+2)_A$, $(h-1)_A, \ldots, 1_A$, and the only one planar CG step h_B. Then, it is possible to verify that by backtracking from index $i = m$, we have from (3.14)

$$\begin{aligned}
&\bar{y}_i = \frac{\|r_m\|}{\|r_i\|}, && i = m, \ldots, h+2, \\
&\bar{y}_{h+1} = \frac{\|r_m\|}{\|r_{h+2}\|} \Phi_1^{(h)}, && \\
&\bar{y}_h = \frac{\|r_m\|}{\|r_{h+2}\|} \Phi_2^{(h)}, && \\
&\bar{y}_i = \frac{\|r_m\|}{\|r_{h+2}\|} \Phi_2^{(h)} \frac{\|r_h\|}{\|r_i\|}, && i = h-1, \ldots, 1,
\end{aligned} \tag{3.16}$$

where

$$\Phi_1^{(h)} = -\frac{\tilde{\alpha}_4^{(h)}}{\tilde{\alpha}_2^{(h)}}, \qquad \Phi_2^{(h)} = \frac{\tilde{\alpha}_1^{(h)} \tilde{\alpha}_4^{(h)}}{\tilde{\alpha}_2^{(h)}} - \tilde{\alpha}_3^{(h)}. \tag{3.17}$$

As a consequence, after few calculations we conclude that the negative curvature direction $d^j = R_n \bar{y}$, can be computed similarly to (3.12). Note that the index m is known only at Step n, so that the storage of R_n is apparently required. However, observing the structure of vector $\bar{y}$ in (3.16), at the current step of the FLR algorithm we simply need to store only a *pair* of vectors. Roughly speaking, one corresponds to a "current" negative curvature direction, the others takes into account that the smallest negative eigenvalue of B_n might not have been yet computed (see the scheme in the next section).

- Let us consider now the case (2). Again the solution $\bar{y}$ is obtained by simply backtracking from $\bar{y}_n$ to $\bar{y}_1$. In particular, from (3.14) we have

$$\bar{y}_n = \cdots = \bar{y}_{m+2} = 0, \qquad \bar{y}_{m+1} = \Psi_1^{(m)}, \qquad \bar{y}_m = \Psi_2^{(m)} \tag{3.18}$$

where $\Psi_1^{(m)} = \omega_{m+1}$ and $\Psi_2^{(m)} = \omega_m - \tilde{\alpha}_1 \omega_{m+1}$ ($\tilde{\alpha}_1$ is defined in (2.12) with $k = m$). Then, assuming w.l.o.g. that before the planar CG step m_B the FLR algorithm performed the standard CG steps $(m-1)_A, \ldots, (h+2)_A, (h-1)_A, \ldots, 1_A$ and the planar CG Step h_B, by backtracking we have from (3.14)

$$\begin{aligned}
\bar{y}_i &= \frac{\|r_m\|}{\|r_i\|} \Psi_2^{(m)}, && i = m-1, \ldots, h+2, \\
\bar{y}_{h+1} &= \frac{\|r_m\|}{\|r_{h+2}\|} \Psi_2^{(m)} \Phi_1^{(h)}, \\
\bar{y}_h &= \frac{\|r_m\|}{\|r_{h+2}\|} \Psi_2^{(m)} \Phi_2^{(h)}, \\
\bar{y}_i &= \frac{\|r_m\|}{\|r_{h+2}\|} \Psi_2^{(m)} \Phi_2^{(h)} \frac{\|r_h\|}{\|r_i\|}, && i = h-1, \ldots, 1.
\end{aligned} \tag{3.19}$$

Again, from (3.19) it can be seen that d^j may be iteratively computed, as summarized in case (1) (see also the next section).

Finally, observe that the considerations above may be straightforwardly generalized, in case several planar CG steps are performed.

3.3 Explicit scheme for computing the negative curvature direction

In the previous section we described the rules for the iterative computation of the negative curvature direction d^j, *without requiring the storage of the full rank matrix* R_n. Now, we report in Table 2 a scheme which summarizes such computation, in the general case of a sequence of standard and planar CG steps. We remark that the algorithm in Table 2 performs as many iterations as the FLR algorithm (i.e. it can possibly perform $h < n$ steps). Thus, according with Sect. 3, the negative curvature direction which satisfies (3.1) and (3.2) can be fruitfully *approximately computed*. In particular, we denote by d_h^j the direction of negative curvature d^j computed after $h \leq n$ steps of the FLR algorithm. We give evidence in Sect. 4 that this approach is effective.

Before going into the details of this scheme, we describe both variables and parameters involved, in order to allow a straightforward consultation. Firstly, observe that since the approximated negative curvature direction d_h^j, $h \leq n$, satisfies

Table 2 The iterative computation of the approximate negative curvature direction d_h^j, after $h \le n$ iterations of the FLR algorithm. The logical variable *CR* (defined in the FLR algorithm) is used for the stopping criterion

Step 1: $k = 1, d_{\mathcal{A}} = d_{\mathrm{curr}_{\mathcal{A}}} = 0, d_{\mathcal{B}} = d_{\mathrm{curr}_{\mathcal{B}}} = 0, r_{\mathrm{norm}_{\mathcal{A}}} = 0, \lambda_{m_{\mathcal{A}}} = \lambda_{m_{\mathcal{B}}} = 0,$
$CR = false.$

Step k:
If $CR = true$ then go to the *Final step*.
If $|\sigma_k| \ge \epsilon_k \|p_k\|^2$ then
if $1/a_k \ge \lambda_{m_{\mathcal{A}}}$ then
$d_{\mathrm{curr}_{\mathcal{A}}} = d_{\mathrm{curr}_{\mathcal{A}}} + \frac{r_k}{\|r_k\|^2}, k = k+1,$ go to *Step k*
else
$d_{\mathrm{curr}_{\mathcal{A}}} = d_{\mathrm{curr}_{\mathcal{A}}} + \frac{r_k}{\|r_k\|^2}, d_{\mathcal{A}} = d_{\mathrm{curr}_{\mathcal{A}}}, \lambda_{m_{\mathcal{A}}} = 1/\alpha_k$
$r_{\mathrm{norm}_{\mathcal{A}}} = \|r_k\|, k = k+1,$ go to *Step k*
endif
if $1/a_k \ge \lambda_{m_{\mathcal{B}}}$ then
$d_{\mathrm{curr}_{\mathcal{B}}} = d_{\mathrm{curr}_{\mathcal{B}}} + \frac{r_k}{\|r_k\|^2}, k = k+1,$ go to *Step k*
else
$d_{\mathrm{curr}_{\mathcal{B}}} = 0, d_{\mathcal{B}} = 0, \lambda_{m_{\mathcal{B}}} = 0, k = k+1,$ go to *Step k*
endif
Else
compute $\lambda_k, \lambda_{k+1}, \omega_k$ and ω_{k+1}. Set $\lambda = \min\{\lambda_k, \lambda_{k+1}\}$
if $\lambda \ge \lambda_{m_{\mathcal{A}}}$ then
$d_{\mathrm{curr}_{\mathcal{A}}} = \big(d_{\mathrm{curr}_{\mathcal{A}}} + \frac{r_k}{\|r_k\|^2}\big)\Phi_2^{(k)} \frac{\|r_k\|}{\|r_{k+2}\|} + \frac{r_{k+1}}{\|r_{k+1}\|\|r_{k+2}\|}\Phi_1^{(k)}$
$k = k+2,$ go to *Step k*
else
$W = d_{\mathrm{curr}_{\mathcal{A}}}$
$d_{\mathrm{curr}_{\mathcal{A}}} = \big(d_{\mathrm{curr}_{\mathcal{A}}} + \frac{r_k}{\|r_k\|^2}\big)\|r_k\|\Psi_2^{(k)} + \frac{r_{k+1}}{\|r_{k+1}\|}\Psi_1^{(k)}$
$d_{\mathcal{A}} = d_{\mathrm{curr}_{\mathcal{A}}}, r_{\mathrm{norm}_{\mathcal{A}}} = 1, \lambda_{m_{\mathcal{A}}} = \lambda, k = k+2,$ go to *Step k*
endif
if $\lambda \ge \lambda_{m_{\mathcal{B}}}$ then
$d_{\mathrm{curr}_{\mathcal{B}}} = \big(d_{\mathrm{curr}_{\mathcal{B}}} + \frac{r_k}{\|r_k\|^2}\big)\Phi_2^{(k)} \frac{\|r_k\|}{\|r_{k+2}\|} + \frac{r_{k+1}}{\|r_{k+1}\|\|r_{k+2}\|}\Phi_1^{(k)}$
$k = k+2,$ go to *Step k*
else
$d_{\mathrm{curr}_{\mathcal{B}}} = \big(W + \frac{r_k}{\|r_k\|^2}\big)\|r_k\|\Psi_2^{(k)} + \frac{r_{k+1}}{\|r_{k+1}\|}\Psi_1^{(k)}$
$d_{\mathcal{B}} = d_{\mathrm{curr}_{\mathcal{B}}}, \lambda_{m_{\mathcal{B}}} = \lambda, k = k+2,$ go to *Step k*.
endif
Endif

Final step:
If $\lambda_{m_{\mathcal{A}}} < \lambda_{m_{\mathcal{B}}}$ then
$d_h^j = r_{\mathrm{norm}_{\mathcal{A}}} d_{\mathcal{A}}$
Else
$d_h^j = d_{\mathcal{B}}$
Endif
If $g^T d_h^j > 0$ then $d_h^j = -d_h^j$.

$d_h^j = \sum_{i=1}^{h} \bar{y}_i r_i / \|r_i\|$, at each step of the FLR algorithm we refine the calculation of d_h^j, by adding a new term. The latter result is iteratively achieved in the scheme of Table 2 by updating contemporaneously the information referred to a couple of different scenarios ($\mathcal{A}$ and $\mathcal{B}$).

The index m in Sects. 3.1 and 3.2 corresponds either to the standard CG step $m_{\mathcal{A}}$ or the planar CG step $m_{\mathcal{B}}$. Thus, in the scenario with subscript $\mathcal{A}$ we store information related to the best standard CG step, performed by the FLR algorithm up to the current step. In the scenario with subscript $\mathcal{B}$ we store the information related to the best planar CG step, performed by the FLR algorithm up to the current step. Moreover, each step of the FLR algorithm may affect both scenarios, so that we need to store a pair of vectors for each scenarios. As regards unknowns and parameters we have:

- k is the index of the current iteration of the FLR algorithm;
- d_h^j is the negative curvature direction computed after $h \leq n$ iterations of the FLR algorithm;
- $r_{\text{norm}_{\mathcal{A}}}$ is a scalar which contains the norm $\|r_m\|$ where m is defined above;
- $d_{\mathcal{A}}$, $d_{\mathcal{B}}$, $d_{\text{curr}_{\mathcal{A}}}$, $d_{\text{curr}_{\mathcal{B}}}$ are four n real vectors with the following role:
  - $r_{\text{norm}_{\mathcal{A}}} d_{\mathcal{A}}$ and $d_{\mathcal{B}}$ represent the best negative curvature directions, in the respective scenarios, detected up to the current Step k,
  - $d_{\text{curr}_{\mathcal{A}}}$ and $d_{\text{curr}_{\mathcal{B}}}$ contain the current information which will be possibly used to update respectively $d_{\mathcal{A}}$ and $d_{\mathcal{B}}$;
- $\lambda_{m_{\mathcal{A}}}$ and $\lambda_{m_{\mathcal{B}}}$ contain the best current approximation of the least negative eigenvalue of the matrix A, respectively in scenario $\mathcal{A}$ and scenario $\mathcal{B}$. Alternatively $\lambda_{m_{\mathcal{A}}}$ and $\lambda_{m_{\mathcal{B}}}$ contain the current least negative eigenvalue of respectively 1×1 and 2×2 diagonal blocks of matrix B_n in (3.13);
- $\{r_k\}$, $\{p_k\}$, $\{\epsilon_k\}$, $\{a_k\}$ are defined in the FLR algorithm;
- g is defined as $g = -r_1$, and is used in the *Final step* to ensure that d_h^j is of descent (in an optimization framework where $As = b$ is the Newton equation);
- the sequences $\{\Phi_1^{(k)}\}$ and $\{\Phi_2^{(k)}\}$ are defined in (3.17);
- the sequences $\{\Psi_1^{(k)}\}$ and $\{\Psi_2^{(k)}\}$ are defined in (3.18);
- the real scalars λ_m, λ_{m+1}, ω_m, ω_{m+1} are defined in (2) of the previous section;
- W is an n real working vector;
- *CR* is the logical variable introduced in Table 1, to address the stopping condition. Thus, it is accordingly used in Table 2.

Observe that at Step k, either the quantity $1/a_k$ (eigenvalue of the current 1×1 block) or λ (the least eigenvalue of the current 2×2 block) is tested. In particular, the latter quantities are compared with the current least negative eigenvalues $\lambda_{m_{\mathcal{A}}}$ and $\lambda_{m_{\mathcal{B}}}$ in both scenarios. In case $1/a_k$ and λ are larger than $\lambda_{m_{\mathcal{A}}}$ and $\lambda_{m_{\mathcal{B}}}$, then only vectors $d_{\text{curr}_{\mathcal{A}}}$ and $d_{\text{curr}_{\mathcal{B}}}$ are updated. On the contrary, if $1/a_k$ or λ improve respectively $\lambda_{m_{\mathcal{A}}}$ or $\lambda_{m_{\mathcal{B}}}$, then also $\lambda_{m_{\mathcal{A}}}$ or $\lambda_{m_{\mathcal{B}}}$ will be updated, along with $d_{\mathcal{A}}$ or $d_{\mathcal{B}}$.

Finally, observing the iterative procedure in the scheme of Table 2, we remark that *at most* 4 *vectors* ($d_{\mathcal{A}}$, $d_{\mathcal{B}}$, $d_{\text{curr}_{\mathcal{A}}}$, $d_{\text{curr}_{\mathcal{B}}}$), additional with respect to the ones required by the FLR algorithm, *need to be stored* for computing d_h^j.

4 Preliminary numerical experience

In this section we aim at preliminarily assessing both the reliability and the effectiveness of the approach proposed in this paper to compute negative curvature directions. On this guideline we applied the algorithm proposed in Table 1 to solve:

(a) "stand alone" indefinite linear systems;
(b) linear systems arising in nonconvex unconstrained optimization.

As regards (a), the algorithm in Table 1 is experienced for solving a sequence of symmetric indefinite linear systems of 500 unknowns, in order to verify if in practice condition (3.11) is met. In other words, we give evidence that few iterations ($\ll n$) suffice to obtain a satisfactory negative curvature direction. In particular different values of the condition number (*cond*) are considered for each linear system in Table 3. The latter choice is evidently motivated by the well known strong dependency of the CG-type methods from *cond* [13]. In particular the values $cond = \exp(\alpha)$, $\alpha = 2, 4, 6, 8, 10$, were considered. Furthermore, the stopping criterion adopted at Step k of the FLR algorithm is $\|r_{k+1}\| \leq 10^{-8}\|r_1\|$, so that in Table 3 we can observe that the stopping criterion was met within 500 iterations only for $cond = \exp(2) \cong 7.39$.

As regards the other acronyms in Table 3, n represents the number of unknowns, CG_it is the number of iterations which are necessary to meet the stopping criterion. CG_it gives an average result over 10 randomly generated runs with n and *cond* fixed. Then, $\|r_{500}\|/\|r_1\|$ is the ratio between the norm of the residual computed at step 500 and the initial residual r_1. Finally, with $\kappa_L^2 Ray(d_k/\|d_k\|)$ we indicate the quantity (see relation (3.11))

$$\left[\kappa(\tilde{L}_k)\right]^2 \frac{[(R_k\bar{y})^T A(R_k\bar{y})]}{\|R_k\bar{y}\|^2}, \quad k \leq n,$$

computed after $k = 50, 100, \ldots, 450$ iterations. Observe that the results in Table 3 aim at giving evidence that (3.11) holds even in case $k \ll n$ in the FLR algorithm. In particular, the Table 3 confirms that (3.11) is largely satisfied, provided that at least one negative curvature direction was detected. We can see that the latter condition always occurs even after a small fraction of n iterations.

As regards (b), in order to assess if the iterative computation of negative curvature directions described in the previous sections is reliable within optimization frameworks, we performed a preliminary numerical study. We embedded the new computational scheme in the truncated Newton method for unconstrained optimization proposed in [19]. For the sake of brevity, we refer to [19] and [20] for a description of the method. We only recall that at the current iterate x_j of this method, a pair of directions is computed—a Newton-type direction s^j and a direction of negative curvature d^j—and the new point is computed by means of a curvilinear search along the path $x(\alpha) = x_j + \alpha^2 s^j + \alpha d^j$. Convergence towards stationary points where the Hessian matrix is positive semidefinite is guaranteed by a suitable assumption on the negative curvature direction (which is slightly weaker than *Condition A* stated in Sect. 3). In [19] both the search directions are computed by using a Lanczos based iterative truncated scheme. In particular, in determining the negative curvature direction, it is

Table 3 Relation (3.11) is tested on different symmetric indefinite linear systems: the condition $\lambda^{\min} > \kappa_L^2 Ray(d_k/\|d_k\|)$ is always fulfilled

n	CG_it	$\|r_{500}\|/\|r_1\|$	$cond$	k	$\lambda^{\min}$	$\kappa_L^2 Ray(\frac{d_k}{\|d_k\|})$
500	105.7	0.795E−08	0.739E+01	50	−0.739E+01	−0.517E+10
				100		−0.983E+09
500	510.9	0.918E−06	0.546E+02	50	−0.546E+02	−0.515E+13
				100		−0.300E+13
				150		−0.297E+12
				200		−0.297E+12
				250		−0.420E+12
				300		−0.419E+12
				350		−0.419E+12
				400		−0.419E+12
				450		−0.931E+11
500	1010.9	0.161E−01	0.403E+03	50	−0.403E+03	−0.745E+15
				100		−0.985E+14
				150		−0.165E+13
				200		−0.164E+13
				250		−0.175E+13
				300		−0.935E+12
				350		−0.927E+12
				400		−0.923E+12
				450		−0.562E+12
500	1510.9	0.471E−01	0.298E+04	50	−0.298E+04	−0.312E+16
				100		−0.247E+16
				150		−0.856E+15
				200		−0.272E+14
				250		−0.236E+14
				300		−0.224E+14
				350		−0.202E+14
				400		−0.201E+14
				450		−0.198E+14
500	2010.9	0.957E−01	0.220E+05	50	−0.220E+05	−0.405E+18
				100		−0.145E+17
				150		−0.129E+17
				200		−0.951E+14
				250		−0.897E+14
				300		−0.884E+14
				350		−0.861E+14
				400		−0.861E+14
				450		−0.792E+14

required to store n Lanczos vectors to ensure the convergence to second order critical points. Actually, due to the requirement of limited storage room, only a (fixed) small number of such vectors (say 50) is stored in practice.

We considered the monotone version (MonNC) of the method proposed in [19] where we replaced the computation of the negative curvature direction with the iterative scheme proposed in Table 2. Therefore the resulting algorithm uses substantially the same Newton-type direction as in the MonNC algorithm, and differs from MonNC only in the determination of the negative curvature direction. This is motivated by the need to assess only the new iterative scheme for computing the direction of negative curvature, the computation of the Newton direction being equal (of course, a more realistic algorithm could be considered, which uses the same iterative scheme for computing both the search directions). As regards all the parameters, we adopt the standard values reported in [19], while as termination criterion we use $\|\nabla f(x_j)\| \leq 10^{-5} \max\{1, \|x_j\|\}$.

The large scale unconstrained problems from the CUTEr collection [16] are used as test problems. We compare the results obtained by the original (Lanczos based) MonNC truncated algorithm and the one which uses the new iterative scheme (we remark that for each outer iteration of the truncated scheme, both the algorithms are forced to perform the same number of inner iterations). In Table 4, we report the re-

Table 4 Comparison on *nonconvex* large scale problems, between the Lanczos based iterative scheme (MonNC [19]), and a Truncated Newton method which uses the algorithm in Table 1

Problem	n	Lanczos-based scheme		New iterative scheme	
		it/ng	nf	it/ng	nf
BRYBND	10000	26	42	25	34
COSINE	10000	10	15	9	13
CURLY10	10000	3034	3042	2963	2971
DIXMAANE	1500	15	17	15	17
DIXMAANE	3000	16	18	16	18
DIXMAANG	3000	15	16	15	16
DIXMAANH	1500	16	17	16	17
DIXMAANI	1500	24	25	24	25
DIXMAANI	3000	27	28	27	28
FLETCHCR	1000	1604	2368	1613	2417
GENROSE	1000	660	1194	679	1151
GENROSE	10000	6782	12380	6916	11693
MSQRTALS	1024	49	50	46	47
MSQRTBLS	1024	45	46	45	46
SINQUAD	1000	17	24	19	24
SINQUAD	10000	27	35	31	39
SPMSRTLS	1000	15	16	15	16
SPMSRTLS	10000	18	19	18	19
TOINTGSS	1000	5	6	6	7
TOINTGSS	10000	5	6	5	6
WOODS	1000	65	95	56	71
WOODS	10000	109	123	107	129

sults obtained for all the problems coherently solved by both the algorithms, where convergence to the same point is achieved and negative curvature directions were encountered (if negative curvature directions are not encountered, the two algorithms coincide). The results are reported in terms of number of iterations and gradient evaluations (it/ng), number of function evaluations (nf). We highlight that this preliminary test does not aim at assessing the performance of two different algorithms; rather it tests the new approach in computing negative curvature directions in optimization frameworks. From these results it can be observed that the adoption of the iterative scheme based on the FLR algorithm provides "good" directions of negative curvature. In fact, our proposal is effective and, in some cases, even better than the Lanczos based scheme MonNC. Moreover, unlike algorithm MonNC, we recall that the new scheme does not require to store any matrix, in order to guarantee the convergence to second order critical points.

5 Conclusions

In this paper we introduce a new approach for iteratively computing directions of negative curvature, for an indefinite matrix. The latter result can be used within different contexts of large scale optimization. The aim of this work is to provide a general theoretical framework to ensure the convergence to second order critical points, in solving optimization problems. The resulting method allows to compute adequate negative curvature directions, avoiding any matrix storage in large scale settings. We performed a preliminary numerical experience where our proposal was effective.

However, to better evaluate the efficiency of the proposed method, an extensive numerical testing is certainly needed, and this deserves a separate work. In fact, a wide investigation is necessary on different optimization contexts, in order to assess the capability of our approach to take advantage from the nonconvexity of the objective function.

Acknowledgement This work was partially supported by the *Ministero delle Infrastrutture e dei Trasporti* in the framework of the research plan "*Programma di Ricerca sulla Sicurezza*," Decreto 17/04/2003 G.U. n. 123 del 29/05/2003. This work was also partially supported by Research Project FIRB RBNE01WBBB on "*Large Scale Nonlinear Optimization*," Rome, Italy.

We would like to thank the anonymous referee for the constructive comments and suggestions which led to improve the paper.

References

1. Bank, R., Chan, T.: A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems. Numer. Algorithms **7**, 1–16 (1994)
2. Boman, E., Murray, W.: An iterative approach to computing a direction of negative curvature. Presented at Copper Mountain conference, March 1998. Available at the url: www-sccm.stanford.edu/students/boman/papers.shtml
3. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-Region Methods. MPS–SIAM Series on Optimization. SIAM, Philadelphia (2000)
4. Cullum, J., Willoughby, R.: Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Birkhäuser, Boston (1985)
5. Dixon, L., Ducksbury, P., Singh, P.: A new three-term conjugate gradient method. Technical report 130, Numerical Optimization Centre, Hatfield Polytechnic, Hatfield, Hertfordshire, UK (1985)

6. Facchinei, F., Lucidi, S.: Convergence to second order stationary points in inequality constrained optimization. Math. Oper. Res. **93**, 746–766 (1998)
7. Fasano, G.: Use of conjugate directions inside Newton-type algorithms for large scale unconstrained optimization. PhD thesis, Università di Roma "La Sapienza", Roma, Italy (2001)
8. Fasano, G.: Lanczos-conjugate gradient method and pseudoinverse computation, on indefinite and singular systems. J. Optim. Theory Appl. DOI 10.1007/s10957-006-91193
9. Fasano, G.: Planar-conjugate gradient algorithm for large-scale unconstrained optimization, part 1: theory. J. Optim. Theory Appl. **125**, 523–541 (2005)
10. Fasano, G.: Planar-conjugate gradient algorithm for large-scale unconstrained optimization, part 2: application. J. Optim. Theory Appl. **125**, 543–558 (2005)
11. Fasano, G., Roma, M.: Iterative computation of negative curvature directions in large scale optimization: theory and preliminary numerical results, Technical report 12-05, Dipartimento di Informatica e Sistemistica "A. Ruberti", Roma, Italy (2005)
12. Ferris, M., Lucidi, S., Roma, M.: Nonmonotone curvilinear linesearch methods for unconstrained optimization. Comput. Optim. Appl. **6**, 117–136 (1996)
13. Golub, G., Van Loan, C.: Matrix Computations, 3rd edn. John Hopkins University Press, Baltimore (1996).
14. Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Solving the trust-region subproblem using the Lanczos method. SIAM J. Optim. **9**, 504–525 (1999)
15. Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Exploiting negative curvature directions in linesearch methods for unconstrained optimization. Optim. Methods Softw. **14**, 75–98 (2000)
16. Gould, N.I.M., Orban, D., Toint, P.: CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited. ACM Trans. Math. Softw. **29**, 373–394 (2003)
17. Hestenes, M.: Conjugate Direction Methods in Optimization. Springer, New York (1980)
18. Liu, Y., Storey, C.: Efficient generalized conjugate gradient algorithm, part 1. J. Optim. Theory Appl. **69**, 129–137 (1991)
19. Lucidi, S., Rochetich, F., Roma, M.: Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization. SIAM J. Optim. **8**, 916–939 (1998)
20. Lucidi, S., Roma, M.: Numerical experiences with new truncated Newton methods in large scale unconstrained optimization. Comput. Optim. Appl. **7**, 71–87 (1997)
21. McCormick, G.: A modification of Armijo's step-size rule for negative curvature. Math. Program. **13**, 111–115 (1977)
22. Miele, A., Cantrell, J.: Study on a memory gradient method for the minimization of functions. J. Optim. Theory Appl. **3**, 459–470 (1969)
23. Moré, J., Sorensen, D.: On the use of directions of negative curvature in a modified Newton method. Math. Program. **16**, 1–20 (1979)
24. Moré, J., Sorensen, D.: Computing a trust region step. SIAM J. Sci. Stat. Comput. **4**, 553–572 (1983)
25. Nash, S.: A survey of truncated-Newton methods. J. Comput. Appl. Math. **124**, 45–59 (2000)
26. Paige, C., Saunders, M.: Solution of sparse indefinite systems of linear equations. SIAM J. Numer. Anal. **12**, 617–629 (1975)
27. Parlett, B.: The Symmetric Eigenvalue Problem. Prentice-Hall Series in Computational Mathematics. Prentice-Hall, Englewood Cliffs (1980)
28. Shultz, G., Schnabel, R., Byrd, R.: A family of trust-region-based algorithms for unconstrained minimization. SIAM J. Numer. Anal. **22**, 47–67 (1985)
29. Stoer, J.: Solution of large linear systems of equations by conjugate gradient type methods. In: Bachem A., Grötschel M., Korte B. (eds.) Mathematical Programming. The State of the Art, pp. 540–565. Springer, Berlin/Heidelberg (1983)
30. Trefethen, L., Bau, D.: Numerical Linear Algebra. SIAM, Philadelphia (1997)