High Performance Algorithms and Software for Nonlinear Optimization, pp. 238-258
G. Di Pillo and A. Murli, Editors

Planar-CG Methods and Matrix Tridiagonalization in Large Scale Unconstrained Optimization[1]

Giovanni Fasano (fasano@dis.uniroma1.it)
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "La Sapienza"
via Buonarroti 12 - 00185 Roma, Italy

Abstract

In this paper we aim at carrying out and describing some issues for real eigenvalue computation via iterative methods. More specifically we work out new techniques for iteratively developing specific tridiagonalizations of a *symmetric* and *indefinite* matrix $A \in \mathbb{R}^{n\times n}$, by means of suitable Krylov subspace algorithms defined in [16], [26]. These schemes represent extensions of the well known Conjugate Gradient (CG) method to the indefinite case. We briefly recall these algorithms and we suggest a comparison with the method in [22], along with a discussion on the practical application of the proposed results for eigenvalue computation. Furthermore, we focus on motivating the fruitful use of these tridiagonalizations for ensuring the convergence to second order points, within an optimization framework.

Keywords: unconstrained optimization, eigenvalue computation, matrix tridiagonalization, Conjugate Gradient, Krylov subspace methods.

1 Introduction and preliminaries

The efficient computation of both eigenvalues and eigenvectors, in a standard symmetric eigenproblem, often provides a useful tool in an optimization framework, for studying several real life problems. The use of eigenelements ranges from the capability of giving a better insight into the evolution of linear systems (e.g. the structure resonance), to the assessment of stability under small perturbations. The problem we consider deals with the solution of the eigenproblem

$$Ax = \lambda x \qquad A \in \mathbb{R}^{n\times n}, \lambda \in \mathbb{R}, x \in \mathbb{R}^n \tag{1}$$

[1]This work was supported by MIUR, National Research Program *Algorithms for Complex Systems Optimization*

where matrix A is *symmetric*. In the last decades, a wide variety of approaches and algorithms has been considered for problem (1): this is a direct consequence of the fact that the order n of matrix A, along with the accuracy in the calculation and the computational cost for solving (1), jointly determine the method employed. Here we point out a brief taxonomy of the main approaches, in order to motivate our results and explain the context of their applicability.

As long as n is modest, *direct algorithms* appear the methods of choice and need an $\mathcal{O}(n^3)$ flops along with an $\mathcal{O}(n^2)$ storage. In general, at the end of the computation, they all can provide both eigenvalues and (with an additional cost) eigenvectors of matrix A; however, now we highlight several distinctions, in order to decide the best method for the problem in hand. Apart from *Rayleigh quotient iteration* [31] and *Jacobi's method* [11], which deal with general symmetric matrices, *Bisection* [2], *Tridiagonal QR iteration* [6] and the efficient *Divide and Conquer* [8] solve a tridiagonal eigenproblem, i.e. they assume that matrix A in (1) is endowed with a tridiagonal structure. In case $n \geq 25$, Divide and Conquer (LAPACK routine `sstevd`) proved to be definitely the fastest one when all the eigenpairs of A are required [10], since it usually performs $\mathcal{O}(n^{2.3})$ flops on average. However, in case the eigenvectors are not sought, the Tridiagonal QR iteration is the algorithm of choice [10] (LAPACK routines `ssyev` and `sstev`). Moreover, both Rayleigh quotient iteration and QR iteration converge cubically, thus the number of correct digits triples step by step. On the other hand, Bisection (LAPACK routine `sstebz`) has the remarkable feature of detecting the eigenvalues of A in the range $[a, b] \subseteq \mathbb{R}$, and it requires only $\mathcal{O}(hn)$ flops for their calculation, as long as h eigenvalues are sought. This implies that Bisection is faster than QR iteration whenever $h < n$, since the latter one requires $\mathcal{O}(n^2)$ for the eigenvalues computation. Finally Jacobi's method is significantly slower than all the others, taking $\mathcal{O}(n^3)$ flops on the overall, with a large constant. However, it preserves interesting properties of accuracy in case tiny eigenvalues need to be computed, i.e. in case A is near singular [11]. Other references for direct methods, aiming at solving problem (1), are provided by [1],[3],[4],[5],[17],[23],[27],[33], where finer techniques are reported along with more recent results and further references. In addition, a significant numerical comparison among the cited approaches is available in [10], where some real life examples, involving the solution of differential equations, are figured out.

Another standpoint for the computation of eigenelements in (1) is "via optimization". This approach namely applies when either n is *very large* [27], or the computational cost involved in accurately solving problem (1) becomes prohibitive [14]. The rationale behind this approach relies on the possibility of minimizing suitable functions, whose global minima coincide with eigenvectors of A. For instance, when matrix A is positive definite, the Rayleigh quotient

$$\rho(x) = \frac{x^T A x}{x^T x}$$

could be minimized by means of suitable iterative methods, in order to detect the

eigenvector associated to the least eigenvalue. This idea was originally introduced by Bradbury and Fletcher in [7]; in [27] a set of these functions is proposed and some variants of Newton method, for detecting the critical points of these functions, are compared. Finally in [12] a real application from chemistry is detailed.

There is a further approach in iteratively solving the symmetric eigenproblem (1): namely it is advisable in case n is large and only an approximation of the extreme eigenvalues of matrix A is required [10].
In this paper we shall focus on the latter approach, since in large scale optimization frames it is quite worthwhile. In particular we shall approximately solve (1) by means of a suitable class of *iterative methods*, which retain both a satisfactory effectiveness and a competitive computational burden.
At iteration k-th ($k < n$) these methods provide a set of k orthogonal vectors $\{r_1, \ldots, r_k\}$ (see [10] chapter 7), such that the following representation for matrix A can be generated:

$$AR_k \;=\; R_k T_k \;+\; \eta_k r_{k+1} e_k^T \tag{2}$$

where $R_k = (r_1 \cdots r_k) \in \mathbb{R}^{n\times k}$ with $\{r_1, \ldots, r_{k+1}\}$ orthonormal set, $T_k \in \mathbb{R}^{k\times k}$ *symmetric* and *tridiagonal*, $\eta_k \in \mathbb{R}$, $e_k = (0, \ldots, 0, 1) \in \mathbb{R}^k$. We remark that since the set $\{r_1, \ldots, r_{k+1}\}$ is orthogonal, relation (2) can alternatively be written in the following form (T_k factorization):

$$R_k^T A R_k \;=\; T_k \;+\; \eta_k R_k^T r_{k+1} e_k^T \;=\; T_k. \tag{3}$$

Relation (2) will be addressed as *current representation* of the symmetric matrix A, since it is not properly a matrix factorization. In the remainder of this introductory section and in Section 2 we shall point out the importance of iteratively calculating relations (2) and (3), within optimization frameworks. Moreover, we can see how relations (2) and (3) definitely become a useful tool, in case we aim at determining information on the extreme eigenvalues of matrix A (see also [14], [33] and [13]). Indeed, suppose matrix T_k in (2) is irreducible [2], then it can be proved that the real eigenvalues $\lambda_1(T_k), \ldots, \lambda_k(T_k)$ of matrix T_k are *all distinct* [18],[30]. In addition, in case $k = n$ we have the following relation between the eigenvalues of A and T_n:

$$\begin{array}{rcl} \min_i \{\lambda_i(A)\} & = & \min_i \{\lambda_i(T_n)\} \\ \max_i \{\lambda_i(A)\} & = & \max_i \{\lambda_i(T_n)\}\,. \end{array} \tag{4}$$

Now, if without lost of generality we order the eigenvalues of T_k as $\lambda_1(T_k) < \cdots < \lambda_k(T_k)$, then the following interlacing rule between the spectra of tridiagonal irreducible matrices T_{k-1} and T_k holds (*Sturm sequence*) [34]:

$$\lambda_1(T_k) < \lambda_1(T_{k-1}) < \lambda_2(T_k) < \lambda_2(T_{k-1}) < \cdots < \lambda_{k-1}(T_{k-1}) < \lambda_k(T_k). \tag{5}$$

The properties summarized above, about symmetric tridiagonal irreducible matrices, provide a clue for investigating the spectrum of matrix A in case representation (2)

[2] I.e. all the entries on the first subdiagonal (superdiagonal) of tridiagonal matrix T_k are not zero.

is available. Indeed, from the properties just outlined we deduce that the sequence $\{\lambda_1(T_k)\}$ $[\{\lambda_k(T_k)\}]$ represents a monotonically decreasing [increasing] sequence of estimates for the smallest [greatest] eigenvalue of A. Hence, in case n is large and only an approximation of either $\lambda_1(A)$ or $\lambda_n(A)$ are sought, a suitable iterative method might be employed for estimating them from (3), (4) and (5) [3].

To this aim, this paper will point out the key role of some CG-based methods, which match the latter requirement. In addition, by means of the same algorithms, it will be possible to evaluate the error $|\lambda_1(A) - \lambda_1(T_k)|$ at step k-th on the smallest eigenvalue of A. More precisely, if $(\lambda_1(T_k), u_1^{(k)})$ is the eigenpair of T_k with $\|u_1^{(k)}\| = 1$, then the so called *Ritz pair* $(\lambda_1(T_k), R_k u_1^{(k)})$ may be considered an approximation of the eigenpair $(\lambda_1(A), u_1)$ of matrix A, inasmuch as [18]:

$$\left\| AR_k u_1^{(k)} - \lambda_1(T_k) R_k u_1^{(k)} \right\| = \left\| AR_k u_1^{(k)} - R_k T_k u_1^{(k)} \right\| \leq |\eta_k| \tag{6}$$

and this bound is available *without explicitly calculating* the eigenpair $(\lambda_1(T_k), R_k u_1^{(k)})$ at iteration k-th. When tridiagonal matrix T_k in (2) becomes reducible or near so, i.e. $\eta_k \approx 0$, the results above are, to a large extent, a powerful tool for partially investigating the structural properties of matrix A.

The remainder of this section sets the terminology and the notation used. Throughout the paper the symbol $\|\cdot\|$ denotes the Euclidean norm of a real vector. Moreover, we shall use the notation $x^T y$ for the Euclidean inner product between vectors $x, y \in \mathbb{R}^n$, and $x \perp y$ for their orthogonality. Finally the symbol "$\doteq$" stands for "...by definition...", and the quantities generated at step k-th of an iterative scheme will be denoted with the same subscript k.
A brief outline of the paper is the following: Section 2 introduces the optimization framework we deal with. Section 3 is devoted to recall and describe the potentialities of the Lanczos method, which proved to be effective for our purposes, while Section 4 includes a similar description regarding the CG method. In Sections 5 and 5.1 we work out a new approach where the same results of Sections 3 and 4 are obtained by means of some CG-based methods. The results of Section 5.1 will provide suitable generalizations of Section 4. Finally Section 6 will summarize the conclusions along with the forthcoming work.

2 The optimization framework

Suppose we deal with the solution of the optimization problem:

$$\min_{x \in \mathbb{R}^n} \; f(x) \tag{7}$$

[3] We recall that the calculation of the spectrum of a tridiagonal matrix, can be straightforwardly obtained by means of direct algorithms.

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and n is *large*. A truncated Newton method (see [9]) for the solution of (7) may be a successful choice, because of its good Newton-like convergence property for large n. Now suppose function $f(x)$ is nonconvex over large regions and we aim at detecting a stationary point x^* of $f(x)$, which verifies second order optimality conditions. The most efficient truncated Newton schemes based on a linesearch approach [25], [20], [28] deeply explore the local behaviour of $f(x)$, by means of generating a pair of suitable directions. In particular, these methods consider the second order local expansion of function $f(x)$ at current point x_k. Then they determine both a Newton-type direction s_k, which approximately solves the linear system (Newton equation)

$$\nabla^2 f(x_k)s + \nabla f(x_k) \;=\; 0, \tag{8}$$

and a direction d_k such that

$$d_k^T \nabla^2 f(x_k) d_k \;\leq\; a\lambda_m \left(\nabla^2 f(x_k)\right) \tag{9}$$

where $a > 0$, $\lambda_m\left(\nabla^2 f(x_k)\right)$ is the smallest eigenvalue of the Hessian $\nabla^2 f(x_k)$ at x_k, and $\nabla f(x_k)$ is the gradient at x_k. Finally, a suitable steplength α_k is computed and the subsequent point x_{k+1} results from the relation

$$x_{k+1} \;=\; x_k + \alpha_k^2 s_k + \alpha_k d_k.$$

By roughly speaking vector s_k preserves the performances of the method, while direction d_k both resembles the least eigenvector u_m of $\nabla^2 f(x_k)$ and under suitable assumptions, ensures the convergence towards a region where the Hessian $\nabla^2 f(x_k)$ is positive semidefinite (convergence to a second order stationary point). Anyway, since n is large, these truncated Newton schemes are effective as long as both s_k and d_k are efficiently computed. In other words s_k and d_k have to be computed by means of the same iterative method, which has to provide an estimate of $\lambda_m\left(\nabla^2 f(x_k)\right)$ as well. On this purpose, firstly we shall survey some Krylov subspace methods, which are widely used inside truncated Newton schemes: it will be shown how they accomplish the latter requirement by means of calculating relation (2) (see [10] [18]). Then we will recall and rearrange suitable inexpensive CG-based algorithms introduced in [16] [26], for obtaining some improved results: this will be the original contribution of this paper.

3 Matrix tridiagonalization and Current Representation with the Lanczos algorithm

Consider the solution of the *symmetric* linear system

$$Ax = b \tag{10}$$

where matrix A is *indefinite* and *nonsingular*; among the Krylov iterative methods for solving (10), *the Lanczos algorithm* [22] was one of the first to be proposed. The underpinning idea of this method is the following: given a *symmetric indefinite* matrix $A \in \mathbb{R}^{n\times n}$ and a non null vector $q_1 \in \mathbb{R}^n$ such that $\|q_1\| = 1$, the method generates a set of mutually orthogonal directions (*the Lanczos vectors*) starting from q_1. In particular at step k-th of the iterative procedure, the Lanczos algorithm (Alg_L) provides an orthonormal basis $\{q_1, \ldots, q_k\}$ for Krylov subspace $\mathcal{K}_k(q_1, A) \doteq span\{q_1, Aq_1, \ldots, A^{k-1}q_1\}$, then the method attempts to determine the solution of linear system (10) over this subspace. Here below is the method, in its original formulation (see [18]):

Alg_L

step 0**:**

$k = 0$, $v_0 = b \in \mathbb{R}^n$

$q_0 = 0$, $\delta_0 = \|b\| \neq 0$

step k**:**

if $\delta_k \neq 0$ then:

$q_{k+1} = \dfrac{v_k}{\delta_k}$

$k \leftarrow k + 1$

$\alpha_k = q_k^T A q_k$

$v_k = (A - \alpha_k I)q_k - \delta_{k-1}q_{k-1}$

$\delta_k = \|v_k\|$

repeat **step** k

else **STOP**.

△

An interesting aspect of this method, which deserves to be considered in the sequel, is that the application of Alg_L up to step k-th can be summarized by the following relation [18] [30], which resembles (2):

$$AQ_k \;=\; Q_k T_k \;+\; \delta_k q_{k+1} e_k^T \tag{11}$$

where $Q_k = (q_1, \ldots, q_k) \in \mathbb{R}^{n\times k}$ and whose columns (the Lanczos vectors) form a k-dimensional orthonormal basis of $\mathcal{K}_k(q_1, A)$; $\delta_k > 0$ while $T_k \in \mathbb{R}^{k\times k}$ is a *tridiagonal matrix* such that:

$$T_k \;=\; \begin{pmatrix} \alpha_1 & \delta_1 & & 0 \\ \delta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \delta_{k-1} \\ 0 & & \delta_{k-1} & \alpha_k \end{pmatrix}.$$

There are other remarkable properties of the Lanczos method that we should consider,

when it is used to solve system (10): among these features, the following results claim for a precise analogy with similar conclusions, related to some CG-based algorithms we are going to describe in Section 5:

(1) it solves linear system (10) in at most n steps and the computational burden is almost entirely devoted to perform the matrix-vector products;

(2) as in relation (3) matrix T_k verifies the important equality:

$$T_k \;=\; Q_k^T A Q_k \tag{12}$$

where the column rank of matrix Q_k is increased step by step. In case algorithm Alg_L performs n steps the square matrix Q_n turns to be orthogonal and its columns span $\mathbb{R}^n$;

(3) the iterative procedure can untimely stop after $m < n$ steps; as a consequence a reduced subset of orthogonal directions will be generated. This occurs when the following equality is fulfilled at step m-th

$$\mathcal{K}_m(q_1, A) \;=\; \mathcal{K}_{m+1}(q_1, A);$$

(4) at step $(k+1)$-th the matrix T_{k+1} can be drawn by simply "bordering" the matrix T_k, that is adding a new row and a new column to T_k. Furthermore all the necessary information for constructing matrix T_{k+1} may be recovered from the sequences $\{\alpha_1, \ldots, \alpha_{k+1}\}$ and $\{\delta_1, \ldots, \delta_k\}$ (see Alg_L);

(5) since matrix A is symmetric, an n-dimensional basis of orthogonal eigenvectors exists for A. Now, suppose the vector $q_1 \doteq b/\|b\|$ has a non null projection onto the subspace spanned by $\bar{k}$ eigenvectors of A. In case only $\tilde{k} \leq \bar{k}$ eigenvectors among these are associated to *distinct non null eigenvalues*, then the Lanczos algorithm terminates at step $\tilde{k}$-th [4].

On the other hand, from the viewpoint of optimization, problem (10) is equivalent [5] to determine the stationary point of the quadratic functional

$$q(x) \;=\; \frac{1}{2} x^T A x - b^T x. \tag{13}$$

Thus, the Lanczos method seems a natural candidate for solving a wide variety of optimization problems too. Indeed if $Q_m = (q_1, \ldots, q_m)$ is generated by Lanczos when it stops at m-th iteration (i.e. $\delta_m = 0$), then by means of the substitution $x = Q_m z$,

[4] We remark the complete analogy with CG algorithm properties (see [32]).

[5] Provided that matrix A is nonsingular.

$z \in \mathbb{R}^m$ into relation (13) and recalling that $q_1 = b/\|b\|$, we obtain:

$$\nabla q(z) \;=\; Q_m^T A Q_m z - Q_m^T b \;=\; T_m z - \|b\| e_1.$$

Consequently, whenever the solution z^* of the *tridiagonal* system

$$T_m z - \|b\| e_1 \;=\; 0\ , \qquad z \in \mathbb{R}^m \tag{14}$$

is available, point $x^* = Q_m z^*$ is a solution of the original system (10), or equivalently the stationary point of (13) over the Krylov subspace $\mathcal{K}_m(b, A) = span\{q_1, \ldots, q_m\} \subseteq \mathbb{R}^n$. Furthermore we highlight that in case the Lanczos algorithm stops at step m-th, relation (11) yields:

$$A Q_m \;=\; Q_m T_m \tag{15}$$

inasmuch as $\delta_m = 0$, i.e. the set $\{q_1, \ldots, q_m\}$ is an invariant subspace of the range of matrix A (under A transformation) (see [30]).
We conclude this section with the following two considerations about the Lanczos algorithm; we shall see that they play a key role within an optimization framework:

- it actually *does not solve* linear system (10); it rather *transforms* (10) into the simpler one (14): this means that some further calculations are necessary in order to provide the explicit solution of (14) for backtracking to (10). The CG-based methods proposed in Sections 5 and 5.1 do not suffer for such a drawback;

- it seems we need to *store matrix* Q_m in order to calculate the solution x^* of (10), according to relation:

  $$x^* \;=\; Q_m z^*. \tag{16}$$

  This could be a serious disadvantage whenever n becomes large; however, in case we are only interested about the solution x^*, the storage can be avoided (see algorithm SYMMLQ in [29]) by means of a suitable recursive calculation. On the other hand, as recalled in Section 1, we are dealing with proper optimization frameworks where the Lanczos method is also asked to provide a negative curvature direction d_k, which verifies (9). In case we use the Lanczos method and it stops at step m-th, the calculation of d_k inevitably requires the storage of full rank matrix Q_m [25],[24]. In alternative, we could avoid the direct calculation of the Lanczos vectors $q_1, \ldots, q_m$, by simply recovering them from the Conjugate Gradient iteration (see [20], [19]). However this implies that when matrix A is indefinite, the CG procedure might fail or be unstable. The CG-based schemes proposed in Section 5 and 5.1 do not suffer from this drawback, since they cope with the indefinite case too.

4 Matrix tridiagonalization and Current Representation with CG

Provided that matrix A in (10) is *positive definite*, in this section we recall how the use of Conjugate Gradient (CG) algorithm provides a tridiagonalization of matrix A. In addition, an expression of the form (12), as well as a current representation (11), will be straightforwardly derived (see [25], [10]). As well known, a general description of CG method [21] for solving (10), can be figured out by the following instructions:

CG

step 1**:**

$k = 1\ ,\quad x_1 \in I\!R^n$

$r_1 = b - Ax_1\ ,\quad p_1 = r_1$

step k**:**

if $r_k = 0$ then STOP else

$$x_{k+1} = x_k + \theta_k p_k \qquad \theta_k = \frac{r_k^T p_k}{p_k^T A p_k}$$

$$r_{k+1} = r_k - \theta_k A p_k$$

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

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

△

where $r_{k+1} = b - Ax_{k+1}$ and the sequences $\{r_i\}$ and $\{p_i\}$ are such that:

$$\begin{aligned} r_i^T r_j &= 0 \qquad i \neq j \leq k+1 \\ r_i^T p_j &= 0 \qquad j < i \leq k+1 \\ p_i^T A p_j &= 0 \qquad i \neq j \leq k+1. \end{aligned}$$

Now suppose the CG stops at m-th step and $r_{m+1} = 0$, then introducing matrices

$$R_m = \left(\frac{r_1}{\|r_1\|} \cdots \frac{r_m}{\|r_m\|}\right) \in I\!R^{n\times m}, \tag{17}$$

$$P_m = \left(\frac{p_1}{\|r_1\|} \cdots \frac{p_m}{\|r_m\|}\right) \in I\!R^{n\times m}, \tag{18}$$

$$L_m = \begin{pmatrix} 1 & & & 0 \\ -\sqrt{\beta_1} & \ddots & & \\ & \ddots & \ddots & \\ 0 & & -\sqrt{\beta_{m-1}} & 1 \end{pmatrix} \in I\!R^{m\times m}, \tag{19}$$

$$D_m \;=\; \begin{pmatrix} \frac{1}{\theta_1} & & & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & & & \frac{1}{\theta_m} \end{pmatrix} \in \mathbb{R}^{m\times m}, \tag{20}$$

the iterative relations which calculate r_{k+1} and p_{k+1} can equivalently be recasted resorting to the following *matrix equalities* [10]:

$$P_m L_m^T \;=\; R_m \tag{21}$$
$$AP_m \;=\; R_m L_m D_m. \tag{22}$$

Now post-multiplying both the sides of equality (22) for the invertible matrix L_m^T and substituting $P_m L_m^T$ from (21) we obtain:

$$AP_m L_m^T \;=\; R_m L_m D_m L_m^T \;\Longrightarrow\; AR_m \;=\; R_m T_m \tag{23}$$

where $T_m \in \mathbb{R}^{m\times m}$ is a symmetric tridiagonal matrix defined by:

$$T_m = \begin{pmatrix} \frac{1}{\theta_1} & -\frac{\sqrt{\beta_1}}{\theta_1} & & & 0 \\ -\frac{\sqrt{\beta_1}}{\theta_1} & \left(\frac{1}{\theta_2}+\frac{\beta_1}{\theta_1}\right) & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \left(\frac{1}{\theta_{m-1}}+\frac{\beta_{m-2}}{\theta_{m-2}}\right) & -\frac{\sqrt{\beta_{m-1}}}{\theta_{m-1}} \\ 0 & & & -\frac{\sqrt{\beta_{m-1}}}{\theta_{m-1}} & \left(\frac{1}{\theta_m}+\frac{\beta_{m-1}}{\theta_{m-1}}\right) \end{pmatrix} \in \mathbb{R}^{m\times m}. \tag{24}$$

Relations (23) and (24) prove that as long as matrix A is positive definite, the CG provides for matrix A the iterative tridiagonalization (3). Now, likewise algorithm Alg_L, we would like to obtain a current representation for matrix A, by means of algorithm CG. In other words, suppose $r_{k+1} \neq 0$ (i.e. algorithm CG was not arrested at step k-th) then we ought to derive an expression which resembles (2).
On this purpose consider matrix L_k defined as in (19) and introduce the new matrix:

$$\tilde{L}_k \;=\; \begin{pmatrix} L_k \\ 0\cdots 0 \;\; -\sqrt{\beta_k} \end{pmatrix} \in \mathbb{R}^{(k+1)\times k}$$

that allows to rearrange equalities (21) and (22) which become:

$$P_k L_k^T \;=\; R_k \tag{25}$$
$$AP_k \;=\; R_{k+1}\tilde{L}_k D_k. \tag{26}$$

Finally a combination of (25) and (26) yields:

$$AP_k L_k^T \;=\; AR_k \;= R_{k+1}\tilde{L}_k D_k L_k^T \;=\; R_{k+1}\tilde{T}_k \tag{27}$$

where the unsymmetric tridiagonal matrix $\tilde{T}_k \in \mathbb{R}^{(k+1)\times k}$ assumes the expression:

$$\tilde{T}_k \;=\; \begin{pmatrix} & T_k & \\ 0\cdots 0 & & -\frac{\sqrt{\beta_k}}{\theta_k} \end{pmatrix}$$

with T_k defined in (24). Eventually we can readily show that relation (27) recovers relation (2), since it can be rearranged as follows:

$$\begin{aligned} AR_k \;&=\; R_{k+1}\tilde{T}_k \;=\; \left(R_k \quad \frac{r_{k+1}}{\|r_{k+1}\|}\right)\begin{pmatrix} & T_k & \\ 0\cdots 0 & & -\frac{\sqrt{\beta_k}}{\theta_k} \end{pmatrix} = \\ &=\; \left(R_k \quad \frac{r_{k+1}}{\|r_{k+1}\|}\right)\begin{pmatrix} & T_k & \\ & -\frac{\sqrt{\beta_k}}{\theta_k}e_k^T & \end{pmatrix} \;=\; R_kT_k \;-\; \frac{\sqrt{\beta_k}}{\theta_k}\frac{r_{k+1}}{\|r_{k+1}\|}e_k^T \end{aligned} \tag{28}$$

which is evidently the *current representation* for matrix A we were looking for.
We complete the section with an utmost intuitive consideration about the relationship between algorithm Alg_L and algorithm CG. Formulas (15) and (23) formally coincide when we swap the sequences $\{q_i\}$ and $\{r_i/\|r_i\|\}$, thus a simple question seems to arise spontaneously: why the Lanczos algorithm and CG look so different in practice ? For our purposes the difference is "apparent" since at iteration k-th, they both provide a k-dimensional orthonormal basis. However, this is accomplished in a unique phase by the Lanczos algorithm (which requires the storage of the last two vectors generated), while CG algorithm needs an intermediate step (see (21) and (22)).

5 The introduction of *Planar-CG* algorithms

In this section some tridiagonalizations and current representations will be derived for the matrix A in problem (10); however, despite the previous section, here we assume more generally A *indefinite*. We will recall the guidelines provided by the application of the Lanczos and CG methods in Sections 3 and 4, however, for our purposes, a CG-based class of methods will be used.
More precisely we consider the category of *Planar Methods* [26], [21] [15], [16] for both iteratively solving linear system (10), and generating a negative curvature direction d_k for matrix A. The general approach of these CG-based schemes (the Planar methods in [16] and [26] are sketched in Section 5.1) may be roughly summarized as follows. Consider algorithm CG in Section 4 with A indefinite, and suppose that at step k-th the quantity $p_k^TAp_k$ approaches zero. Then, instead of stopping untimely (as CG does), a second direction q_k is generated and the search of the stationary point on the line $x_k + \alpha p_k \quad \alpha \in \mathbb{R}$ (CG-step), is replaced by the search on the 2-dimensional linear manifold (Planar-step)

$$x_k \;+\; span\{p_k, q_k\}. \tag{29}$$

The algorithm in [15] (namely Alg_FLR) accomplishes such a result in a more general way in respect to both the methods in [16] (Alg_F) and [26] (Alg_ML). Consequently algorithm Alg_FLR usually provides more precise solutions; on the other hand Alg_F and Alg_ML are *computationally cheaper*.
The substantial difference between Alg_FLR and both Alg_F and Alg_ML, relies on a test for quantity $p_k^T A p_k$ calculated at step k-th (see Section 5.1). Indeed, Alg_F and Alg_ML perform a Planar-step if and only if relation $p_k^T A p_k = 0$ holds; thus, they might cope inaccurately with the situation $0 < |p_k^T A p_k| < \varepsilon$, ε "small". On the contrary, Alg_FLR is more flexible and does not suffer from the latter drawback. Furthermore, it was proved [15] that both Alg_F and Alg_ML are particular cases of the more general scheme Alg_FLR. Nevertheless, in this context we consider that the use of the former algorithms is still meaningful. Indeed, we recall that we are dealing with linesearch-based truncated Newton schemes, introduced in Section 1. Therefore at step k-th these planar methods are expected to provide only an *approximation* of both a Newton-type direction s_k (which solves (8)) and a negative curvature direction d_k (which verifies (9)). Hence, the low computational burden associated to Alg_F and Alg_ML might be a winning feature[6]. In addition, since Planar-CG algorithms are Krylov subspace methods, they show exactly the same features (1)-(5) described in Section 3 for the Lanczos scheme. We avoid the verbatim report of those properties, however we highlight that this parallelism has remarkable implications inside an optimization framework. Moreover all Planar schemes are far cheaper than the Lanczos method, inasmuch as they skip the solution of the tridiagonal system (14) and do not require *any matrix storage* (see Section 3).

5.1 Matrix Tridiagonalization and Current Representation with algorithms Alg_F and Alg_ML

As reported at the outset of the previous section, here relations (2) and (3) will be obtained by means of Planar-CG algorithms in [16] and [26], which are respectively addressed as Alg_F and Alg_ML (the similitude of behaviour between these algorithms justifies a common treatment of the subject). Here we outline these methods:

Alg_F

step 1:

$k = 1$, $x_1 \in I\!R^n$

$r_1 = b - A x_1$, $p_1 = r_1$

step k**:**

[6] In a forthcoming paper we shall determine a matrix tridiagonalization with algorithm Alg_FLR, too.

if $r_k = 0$ then STOP else

if $p_k^T A p_k \neq 0$ go to **step** k_A else go to **step** k_B

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

$x_{k+1} = x_k + \alpha_k p_k \qquad \alpha_k = \frac{r_k^T p_k}{p_k^T A p_k} = \frac{r_1^T p_k}{p_k^T A p_k}$

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

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

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

step k_B **(Planar-step):**

$p_{k+1} = \dfrac{Ap_k}{\|Ap_k\|}$

$x_{k+2} = x_k + \alpha_k p_k + \alpha_{k+1} p_{k+1} \qquad \alpha_k = -\frac{r_k^T p_k}{\|Ap_k\|^2}(p_{k+1}^T A p_{k+1})$

$r_{k+2} = r_k - \alpha_k A p_k - \alpha_{k+1} A p_{k+1} \qquad \alpha_{k+1} = \frac{r_k^T p_k}{\|Ap_k\|}$

$p_{k+2} = r_{k+2} + \sigma_k p_k \qquad \sigma_k = -\frac{r_{k+2}^T A p_{k+1}}{\|Ap_k\|}$

$k \leftarrow k+2$ repeat **step** k

$\triangle$

Alg_ML

step 1:

$k = 1\ , \quad x_1 \in \mathbb{R}^n$

$r_1 = b - Ax_1\ , \quad p_1 = r_1$

step k:

if $r_k = 0$ then STOP else

if $p_k^T A p_k \neq 0$ go to **step** k_A else go to **step** k_B

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

$x_{k+1} = x_k + \alpha_k p_k \qquad \alpha_k = \frac{r_k^T p_k}{p_k^T A p_k} = \frac{r_1^T p_k}{p_k^T A p_k}$

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

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

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

step k_B **(Planar-step):**

$p_{k+1} = Ap_k - \dfrac{(Ap_k)^T A(Ap_k)}{2 \cdot \|Ap_k\|^2} p_k$

$x_{k+2} = x_k + \alpha_k p_k + \alpha_{k+1} p_{k+1} \qquad \alpha_k = -\frac{r_k^T p_{k+1}}{p_k^T A p_{k+1}}$

$r_{k+2} = r_k - \alpha_k A p_k - \alpha_{k+1} A p_{k+1} \qquad \alpha_{k+1} = \frac{r_k^T p_k}{p_k^T A p_{k+1}}$

$p_{k+2} = r_{k+2} + \sigma_k p_k \qquad \sigma_k = -\frac{r_{k+2}^T A p_{k+1}}{p_k^T A p_{k+1}}$

$k \leftarrow k+2$ repeat **step** k

$\triangle$

We remark that at step k-th, algorithm Alg_F performs exactly a CG step (see scheme at page 246), provided that quantity $d_k = p_k^T A p_k$ is not zero. The latter analytical condition allows the iterative generation of a new residual $r_{k+1} = r_k - \alpha_k A p_k$. Consequently algorithm Alg_F is able to extend the search for the solution of system (10), on the Krylov subspace $\mathcal{K}_{k+1}(r_1, A)$.
On the other hand whenever $d_k = p_k^T A p_k = 0$, step k_B-th of algorithm Alg_F "must" be performed: by roughly speaking this step is substantially equivalent to a double k_A-th step, where the algorithm does not generate an intermediate residual r_{k+1}, but "jumps" directly from the generation of residual r_k to the generation of residual r_{k+2}.
Now suppose $p_k^T A p_k = 0$ at step k-th of Alg_F; we set aside the nature of "residuals" attributed to vectors $r_1, \ldots, r_k$ of algorithm Alg_F, and define a new vector r_{k+1} which verifies:

$$\begin{aligned} r_{k+1}^T r_j &= 0 \qquad & j = 1, \ldots, k \\ r_{k+h}^T r_{k+1} &= 0 \qquad & h \geq 2. \end{aligned}$$

In this way, as well as for the Lanczos algorithm, an orthogonal basis $\{r_1, \ldots, r_{k+1}\}$ of $\mathbb{R}^{k+1}$ will be available.
With a slight deeper insight in algorithm Alg_F we can prove the following result, which defines the nature of the vectors generated by Alg_F [16]:

Theorem 5.1 *Let matrix $A \in \mathbb{R}^{n \times n}$ be symmetric and nonsingular, if at step h-th of algorithm Alg_F we have $r_h \neq 0$, then the following relations hold ($1 \leq j < k \leq h \leq n$):*

(1) $r_h^T p_j = 0$

(2) $r_h^T r_j = 0$

(3) $p_k^T A p_j \neq 0 \iff p_j^T A p_j = 0$ and $k = j + 1$.

□

Now, the condition $p_k^T A p_k = 0$ implies $A p_k \perp p_k$, therefore the conjugacy of direction p_k with the manifold $span\{p_1, \ldots, p_{k-1}\}$ along with the choice:

$$r_{k+1} \doteq A p_k = \|A p_k\| p_{k+1} \tag{30}$$

at step k-th, guarantee the following relations to hold (respectively for steps j_A-th and j_B-th):

$$r_{k+1}^T r_j = \begin{cases} r_{k+1}^T (p_j - \beta_{j-1} p_{j-1}) = (A p_k)^T (p_j - \beta_{j-1} p_{j-1}) = 0 & j \leq k \\ r_{k+1}^T (p_j - \sigma_{j-2} p_{j-2}) = (A p_k)^T (p_j - \sigma_{j-2} p_{j-2}) = 0 & j \leq k. \end{cases}$$

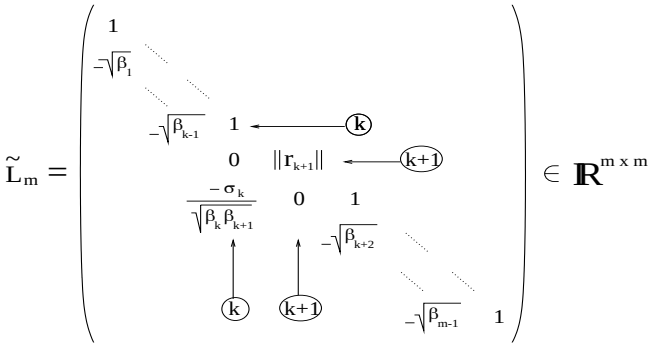

Figure 1: Matrix $\tilde{L}_m$ in (31).

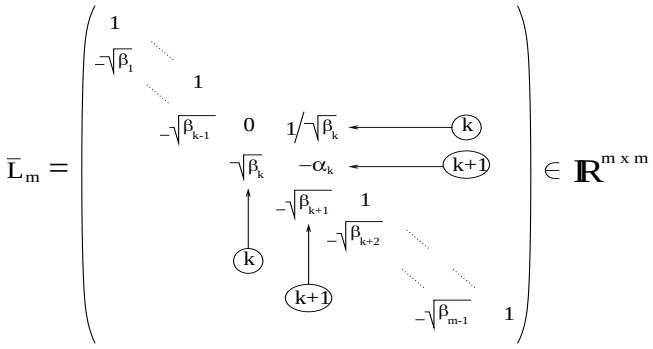

Figure 2: Matrix $\bar{L}_m$ in (32).

Moreover, relations (see Theorem 5.1)

$$r_{k+h}^T r_{k+1} \;=\; r_{k+h}^T(\|Ap_k\|p_{k+1}) \;=\; \|Ap_k\| r_{k+h}^T p_{k+1} \;=\; 0 \qquad h \geq 2$$

will be straightforwardly fulfilled. Thus, the set $\{r_1, \ldots, r_k, r_{k+1}, \ldots, r_{k+h}\}$ is orthogonal. Now suppose $r_{m+1} = 0$; with the position $\beta_j = \|r_{j+1}\|^2 / \|r_j\|^2$, $j \geq 1$ and since at step k-th $r_{k+1} \neq 0$ (see (30)), we could derive the equalities (compare with (21) and (22)):

$$P_m \tilde{L}_m^T \;=\; R_m \tag{31}$$

$$AP_m \;=\; R_m \bar{L}_m \bar{D}_m \tag{32}$$

where matrices P_m and R_m have been defined in (18) and (17). The entries of matrices $\tilde{L}_m$, $\bar{L}_m$ and $\bar{D}_m$ respectively coincide with the entries of L_m, L_m and D_m, as long as $p_k^T A p_k \neq 0$, $k \leq m$. On the contrary if at step k-th $p_k^T A p_k = 0$ holds, then matrices $\tilde{L}_m$, $\bar{L}_m$ and $\bar{D}_m$ will respectively assume the expression in Figures 1, 2 and 3. As a consequence, if algorithm Alg_F stops at step m-th and condition $r_{m+1} = 0$ occurs,

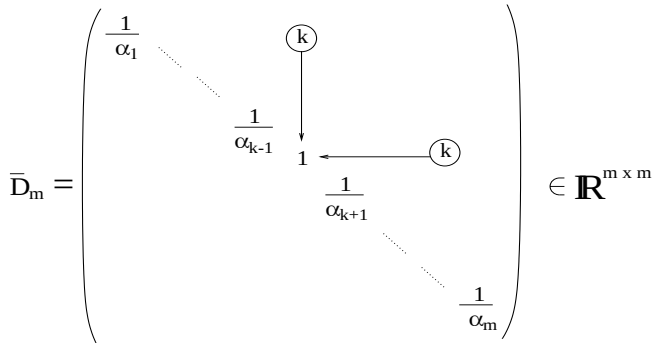

Figure 3: Matrix $\bar{D}_m$ in (32).

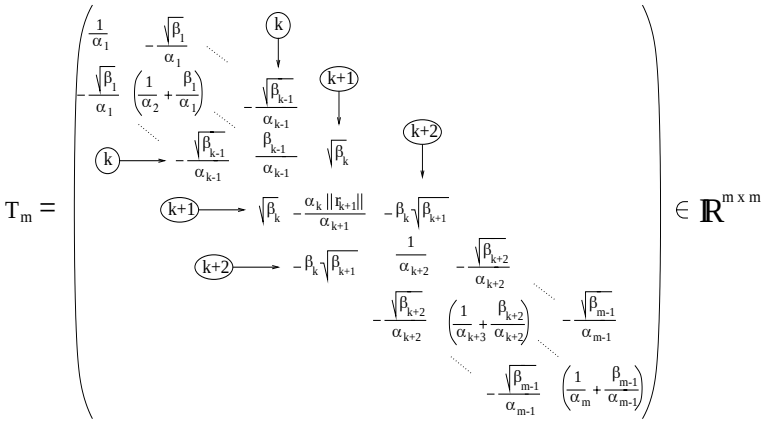

Figure 4: Matrix T_m in (33).

then combining relations (31) and (32) we obtain:

$$AP_m\tilde{L}_m^T \;=\; R_m\bar{L}_m\bar{D}_m\tilde{L}_m^T \;\Longrightarrow\; AR_m \;=\; R_mT_m \tag{33}$$

which is definitely similar to (15) and (23). Furthermore, by means of the definitions for α_k, α_{k+1} and σ_k in Alg_F, matrix T_m is *symmetric* [7] and explicitly depicted in Figure 4. We remark, in accordance with theory, that matrix T_m in (24) and matrix T_m in Figure 4 coincide if step k_B-th of algorithm Alg_F is not performed.
So far, we have iteratively determined tridiagonalization formula (33) for Alg_F likewise relation (23) for algorithm CG. However these formulas hold as long as condition

$$r_{m+1} \;=\; 0$$

[7] In order to justify the symmetry of matrix T_m, it suffices to say that for coefficient σ_k the following equivalence holds:

$$\sigma_k \;=\; -\frac{r_{k+2}^T Ap_{k+1}}{\|Ap_k\|} \;=\; -\frac{r_{k+2}^T\left[\frac{r_k - r_{k+2} - \alpha_k Ap_k}{\alpha_{k+1}}\right]}{\|Ap_k\|} \;=\; \beta_{k+1}\beta_k$$

where the last equality is due to Theorem 5.1 and relation (30).

holds; on the contrary we aim at generalizing (33) when the residual r_{m+1} is not null. In few words we would like to get a *current representation* for matrix A, in the same way we obtained relations (11) for algorithm Alg_L and (28) for CG. On this purpose we introduce the matrix $\tilde{L}_k \in \mathbb{R}^{k\times k}$, which denotes the sub-matrix of $\tilde{L}_m$, corresponding to the first k rows and k columns. Then likewise to relation (25), at step k-th of algorithm Alg_F we can summarize the relation between directions p_i, $i \leq k$ and residuals r_i, by means of the following matrix equivalence:

$$P_k \tilde{L}_k^T \;=\; R_k. \tag{34}$$

Moreover, we introduce another new matrix $\hat{L}_k$ such that:

$$\hat{L}_k \;=\; \begin{pmatrix} \bar{L}_k \\ 0 \cdots 0 \; \bar{l}_{k+1,k} \end{pmatrix} \in \mathbb{R}^{(k+1)\times k}$$

where $\bar{L}_k \in \mathbb{R}^{k\times k}$ is the sub-matrix of $\bar{L}_m$ corresponding to the first k rows and k columns, and $\bar{l}_{k+1,k}$ denotes entry $(k+1,k)$ of $\bar{L}_m$. Finally with $\bar{D}_k \in \mathbb{R}^{k\times k}$ we indicate the sub matrix corresponding to the first k rows and k columns of $\bar{D}_m$.
Likewise (26) the intermediate relation

$$AP_k \;=\; R_{k+1}\hat{L}_k\bar{D}_k \tag{35}$$

holds. Now, post multiplying (35) by $\tilde{L}_k^T$ and substituting (34) we obtain:

$$AP_k\tilde{L}_k^T \;=\; R_{k+1}\hat{L}_k\bar{D}_k\tilde{L}_k^T \;\Longrightarrow AR_k \;=\; R_{k+1}\bar{T}_k \tag{36}$$

where the unsymmetric matrix $\bar{T}_k$ is given by:

$$\bar{T}_k \;=\; \begin{pmatrix} T_k \\ 0 \cdots 0 \; t_{k+1,k} \end{pmatrix} \in \mathbb{R}^{(k+1)\times k}$$

and, as usual, T_k is the sub-matrix of T_m in Figure 4, corresponding to the first k rows and k columns, while $t_{k+1,k}$ is entry $(k+1,k)$ of T_m. Finally, relation (36) may be alternatively recasted in the following expression:

$$\begin{aligned} AR_k \;&=\; R_{k+1}\bar{T}_k \;=\; \begin{pmatrix} R_k & \dfrac{r_{k+1}}{\|r_{k+1}\|} \end{pmatrix} \begin{pmatrix} T_k \\ 0 \cdots 0 \; t_{k+1,k} \end{pmatrix} \;= \\ &=\; \begin{pmatrix} R_k & \dfrac{r_{k+1}}{\|r_{k+1}\|} \end{pmatrix} \begin{pmatrix} T_k \\ t_{k+1,k}e_k^T \end{pmatrix} \;=\; R_kT_k \;+\; t_{k+1,k}\frac{r_{k+1}}{\|r_{k+1}\|}e_k^T \end{aligned} \tag{37}$$

which provides a *current representation* of the form (2) for the symmetric indefinite matrix A.
We end up this section with deriving the same kind of results (*tridiagonalization* and *current representation*) for matrix A, by using algorithm Alg_ML in place of Alg_F.

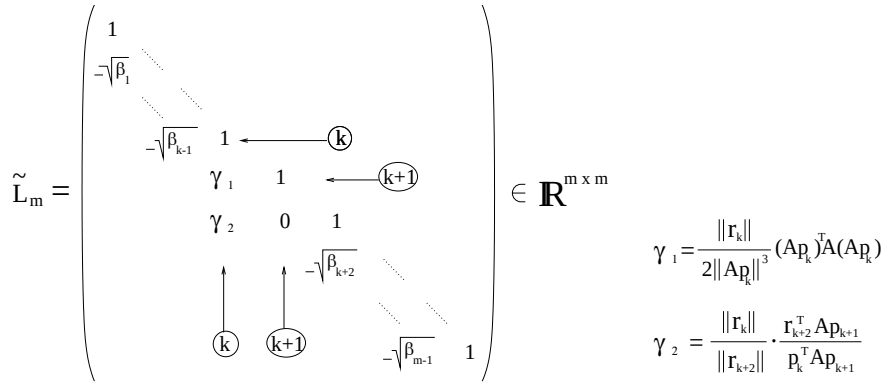

Figure 5: New matrix $\tilde{L}_m$ for algorithm Alg_ML.

We could simply apply algorithm Alg_ML to replicate exactly the same procedure leading to (33) (tridiagonalization of matrix A) and (37) (current representation for matrix A), following the guidelines of algorithm Alg_F. It can be readily seen that position (30) can be assumed as well [8]; consequently relations (31) and (32) still hold. In particular, equality (32) remains formally the same (i.e. matrices $\bar{L}_m$ and $\bar{D}_m$ have the same expression); however the reader should pay attention to the different definition of coefficient α_k of Alg_ML in $\bar{L}_m$. Finally in relation (31) matrix $\tilde{L}_m$ should be replaced by the expression in Figure 5, so that a tridiagonal matrix T_m for matrix A follows straightforwardly. As a consequence, relations (33) and (37) can be still derived by compounding and rearranging equalities (31) and (32): this completes the likeness between algorithm Alg_F and algorithm Alg_ML.

6 Conclusions

In this paper we have proposed the iterative tridiagonalization of a symmetric indefinite matrix, by means of proper CG-based algorithms. We have suitably applied this result inside large scale unconstrained optimization frameworks, within a truncated Newton scheme. Our approach is computationally cheaper in respect to the usual use of the Lanczos method [20], [25]; however, some unresolved questions still deserve to be considered.

Firstly, we did not deal with stability issues related to the new algorithms. On this stream, observe that at step k-th of Planar methods, either a CG-step or a Planar-step can be performed [15],[16]. Nevertheless some numerical shortcomings could arise in case the quantity $|p_k^T A p_k|$ is small; as a consequence, the careless application of the test on $p_k^T A p_k$, at step k-th, may provide misleading results.

In addition, the user's application in hand requires a careful evaluation of the accuracy of Alg_F and Alg_ML in solving linear system (10). In fact, in our experience there

[8] We remark that for algorithm Alg_ML we have a result similar to Theorem 5.1 [26].

is the following trade-off in evaluating the behaviour of Planar methods. Algorithm Alg_FLR is a more accurate solver of (10) in respect to both Alg_F and Alg_ML, however it is computationally more expensive than the latter algorithms. On one hand, this consideration justifies the use of tridiagonalizations through algorithms Alg_F and Alg_ML; on the other hand, it remarks our initial interest for iterative tridiagonalizations through Alg_FLR. In a future work we shall explicitly point out that the results contained in this paper are just a particularization of analogous results, obtained by means of applying Alg_FLR.

Acknowledgments

The author wants to pay his gratitude to Gianni Di Pillo e Massimo Roma for their careful help in reading this paper. Moreover, the valuable advice of the anonymous referee significantly contributed to improve the clearness.

References

[1] G.S.Ammar, L.Reichel, D.C.Sorensen (1992), "An implementation of a divide and conquer algorithm for the unitary eigenproblem", *ACM Transactions on Mathematical Software*, vol. 18, pp. 292-307.

[2] W.Barth, R.S.Martin, J.H.Wilkinson (1967), "Calculation of the eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection", *Numerische Mathematik*, vol. 9, pp. 379-404.

[3] C.H.Bischof, M.Marques, X.Sun (1993), "Parallel Bandreduction and Tridiagonalization", Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, R.Sincovec, Ed., pp.389-390, SIAM.

[4] C.H.Bischof, X.Sun (1992), "A Framework for Symmetric Band Reduction and Tridiagonalization", Preprint MCS-P298-0392, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4801.

[5] C.H.Bischof, X.Sun (1995), "On tridiagonalizing and Diagonalizing Symmetric Matrices with Repeated Eigenvalues", Argonne Preprint MCS-P545-1095, PRISM Working Note #25, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4801.

[6] H.Bowdler, R.S.Martin, C.Reinsch, J.H.Wilkinson (1968), "The QR and QL Algorithms for Symmetric Matrices", *Numerische Mathematik*, vol. 11, pp. 293-306.

[7] W.W.Bradbury, R.Fletcher (1966), "New iterative methods for solutions of the eigenproblem", *Numerische Mathematik*, vol. 9, pp. 259-267.

[8] J.J.M.Cuppen (1981), "A divide and conquer method for the symmetric tridiagonal eigenproblem", *Numer. Math.*, vol. 36, pp. 177-195.

[9] R.S.Dembo, T.Steihaug (1983), "Truncated-Newton Algorithms for large scale Matrix Methods", *Mathematical Programming*, vol. 26, pp. 190-212.

[10] J.W.Demmel (1997), *Applied Numerical Linear Algebra*, SIAM, Philadelphia.

[11] J.Demmel, K.Veselic' (1992), "Jacobi's method is more accurate than QR", *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 1204-1246 (LAPACK Working Note 15).

[12] A.Edelman, T.Arias, S.T.Smith (1994), "Curvature in Conjugate Gradient Eigenvalue Computation with Applications to Materials and Chemistry Calculations", Proceedings of the SIAM Applied Linear Algebra Conference, J.G.Lewis, ed., SIAM, Philadelphia, pp. 233-238.

[13] A.Edelman, T.Arias, S.T.Smith (to appear), "The geometry of algorithms with orthogonality constraints", *SIAM J. Matrix Anal. Appl.*

[14] A.Edelman, S.T.Smith (1996), "On Conjugate Gradient-like methods for eigen-like problems", *BIT*, vol. 36:1, pp. 494-508.

[15] G.Fasano (2002), "On Some Properties of Planar-CG algorithms for Large Scale Unconstrained Optimization - *Part A*" T.R. 03-02, Dipartimento di Informatica e Sistemistica 'A.Ruberti', Università "La Sapienza" Roma, Italy.

[16] G.Fasano (2001), "A new CG-based method for the solution of large scale indefinite linear systems", T.R. 08-01, Dipartimento di Informatica e Sistemistica 'A.Ruberti', Università "La Sapienza" Roma, Italy.

[17] W.N.Gansterer, D.F.Kvasnicka, C.W.Ueberhuber (1998), "Numerical Experiments with Symmetric Eigensolvers", AURORA TR1998-19, University of Technology, Vienna.

[18] G.H.Golub, C.F.Van Loan (1989), *Matrix computations - 3rd edition*, The John Hopkins Press, Baltimore.

[19] N.I.M.Gould, S.Lucidi, M.Roma, Ph.L.Toint (1999), "Solving the trust-region subproblem using the Lanczos method", *SIAM Journal on Optimization* vol. 9, pp. 504-525.

[20] N.I.M.Gould, S.Lucidi, M.Roma, Ph.L.Toint (2000), "Exploiting Negative Curvature Directions in Linesearch Methods for Unconstrained Optimization", *Optimization Methods and Software* vol. 14, pp. 75-98.

[21] M.R.Hestenes (1980), *Conjugate Direction Methods in Optimization*, Springer Verlag, New York Heidelberg Berlin.

[22] C.Lanczos (1950), "An Iterative Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators", *Journal of Research of the National Bureau of Standards* vol. 45, Research Paper 2133.

[23] B.Lang (1999), "Out-of-Core Solution of Large Symmetric Eigenproblems", Preprint RWTH-CS-SC-99-03, Aachen University of Technology, Institute for Scientific Computing.

[24] S.Lucidi, M.Roma (1997), "Numerical experiences with new truncated Newton methods in large scale unconstrained optimization", *Computational Optimization and Applications*, vol. 7, pp. 71-87.

[25] S.Lucidi, F.Rochetich, M.Roma (1999), "Curvilinear stabilization techniques for Truncated Newton methods in large scale unconstrained Optimization", *SIAM Journal on Optimization*, vol. 8, pp. 916-939.

[26] D.G.Luenberger (1969), "Hyperbolic pairs in the Method of Conjugate Gradients", *SIAM J. Appl. Math.*, vol. 17, pp. 1263-1267.

[27] M.Mongeau, M.Torki (1999), "Computing eigenelements of real symmetric matrices via optimization", T.R. MIP 99-54.

[28] J.J.More, D.C.Sorensen (1979), "On the use of directions of negative curvature in a modified Newton method", *Mathematical Programming*, vol. 16, pp. 1-20.

[29] C.C.Paige, M.A.Saunders (1975), "Solution of sparse indefinite systems of linear equations", *SIAM J. on Numerical Analysis*, vol. 12, pp. 617-629.

[30] B.Parlett (1980), *The symmetric eigenvalue problem*, Prentice-Hall series in Computational Mathematics, Englewood Cliffs.

[31] B.Parlett (1974), "Generalized Rayleigh Methods with Applications to Finding Eigenvalues of Large Matrices", *Lin. Alg. and its Applic.*, vol. 4, pp. 353-368.

[32] J.Stoer (1983), *Mathematical Programming, The State of the Art*, A.Bachem, M.Grotschel, B.Korte eds., Springer-Verlag, Berlin.

[33] H.Van Der Vorst (1996), "Subspace Iteration for Eigenproblems" *CWI Quarterly*, vol. 9, pp. 151-160.

[34] J.H.Wilkinson (1965), *The Algebraic Eigenvalue Problem*, Oxford: Oxford University (Clarendon).