Issues on the use of a modified Bunch and Kaufman decomposition for large scale Newton's equation

Andrea Caliciotti, Giovanni Fasano, Florian Potra & Massimo Roma

Computational Optimization and Applications
An International Journal

ISSN 0926-6003

Comput Optim Appl
DOI 10.1007/s10589-020-00225-8

ONLINE FIRST

Springer

Computational Optimization and Applications
https://doi.org/10.1007/s10589-020-00225-8

Issues on the use of a modified Bunch and Kaufman decomposition for large scale Newton's equation

Andrea Caliciotti[1] · Giovanni Fasano[2] · Florian Potra[3] · Massimo Roma[1]

Received: 3 May 2019 / Accepted: 29 August 2020

Abstract
In this work, we deal with Truncated Newton methods for solving large scale (possibly nonconvex) unconstrained optimization problems. In particular, we consider the use of a modified Bunch and Kaufman factorization for solving the Newton equation, at each (outer) iteration of the method. The Bunch and Kaufman factorization of a tridiagonal matrix is an effective and stable matrix decomposition, which is well exploited in the widely adopted SYMMBK (Bunch and Kaufman in Math Comput 31:163–179, 1977; Chandra in Conjugate gradient methods for partial differential equations, vol 129, 1978; Conn et al. in Trust-region methods. MPS-SIAM series on optimization, Society for Industrial Mathematics, Philadelphia, 2000; HSL, A collection of Fortran codes for large scale scientific computation, http://www.hsl.rl.ac.uk/; Marcia in Appl Numer Math 58:449–458, 2008) routine. It can be used to provide conjugate directions, both in the case of 1×1 and 2×2 pivoting steps. The main drawback is that the resulting solution of Newton's equation might not be gradient–related, in the case the objective function is nonconvex. Here we first focus on some theoretical properties, in order to ensure that at each iteration of the Truncated Newton method, the search direction obtained by using an adapted Bunch and Kaufman factorization is gradient–related. This allows to perform a standard Armijo-type linesearch procedure, using a bounded descent direction. Furthermore, the results of an extended numerical experience using large scale CUTEst problems is reported, showing the reliability and the efficiency of the proposed approach, both on convex and nonconvex problems.

Keywords Large scale optimization · Truncated Newton method · Bunch and Kaufman decomposition · Gradient–related directions

1 Introduction

In this paper we consider the unconstrained optimization problem

$$\min f(x), \tag{1.1}$$

Extended author information available on the last page of the article where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a possibly *nonconvex* real valued function, n is large and we solve (1.1) by means of an iterative method which generates the sequence $\{x_h\}$. We consider the standard assumptions that f is twice continuously differentiable, and that $\{x_h\} \subset \Omega$, where $\Omega \subset \mathbb{R}^n$ is compact. Furthermore, no sparsity pattern is assumed for the Hessian matrix $\nabla^2 f(x)$. Then, we approach the solution of (1.1) by means of the iterative process

$$x_{h+1} = x_h + \alpha_h p_h, \qquad h \geq 1, \tag{1.2}$$

being $p_h \in \mathbb{R}^n$ a search direction and $\alpha_h > 0$ a steplength. Dealing with large scale problems, Truncated Newton methods are often considered a good choice for their solution. Truncated Newton methods are also called Newton–Krylov methods, since a Krylov subspace method is usually used, for iteratively (approximately) solving the Newton equation at each iteration. In [21] a general description of a Truncated Newton method is reported. As well known, given an initial guess x_h, a Truncated Newton method for problem (1.1) is based on two nested loops:

- The *outer iterations*, where the current estimate of the solution x_{h+1} is updated using x_h;
- The *inner iterations*, where an iterative algorithm is used for computing, at each outer iteration h, the approximate solution p_h of the Newton equation

$$\nabla^2 f(x_h) p = -\nabla f(x_h). \tag{1.3}$$

The Conjugate Gradient (CG) method is among the most commonly used iterative algorithms (see e.g. [1,17]) for solving (1.3). The main drawback of CG method is that it is well-posed only in case $\nabla^2 f(x_h)$ is positive definite. Conversely, if matrix $\nabla^2 f(x_h)$ is indefinite, the CG algorithm could break down and some difficulties may arise to define an approximate solution p_h of (1.3). In order to cope also with indefinite linear systems, some modified CG algorithms have been proposed in [8,9,11,12,14]. In particular, if the CG breaks down when solving (1.3), we can take as Newton–type direction p_h in the recursion (1.2):

- A scaled steepest descent direction (i.e., $p_h = -C\nabla f(x_h)$, with C positive definite);
- A suitable gradient–related adaptation of the current best approximate solution of (1.3);
- A suitable (descent) negative curvature direction.

Moreover, in [18] a non-standard approach is proposed in order to build a gradient–related direction, by separately accumulating positive and negative curvature directions computed by the CG.

Considering a different standpoint, the CG algorithm may be also used in the indefinite case, as proposed in [13,18], where a suitable combination of conjugate directions proves to yield a gradient–related direction.

In this work we are interested to compare the quality of the search direction obtained by both CG and SYMMBK algorithms (the latter being proposed by Chandra in [5]), when solving (1.3), in order to provide eventually the gradient–related direction p_h in (1.2). On this guideline, we first need to prove that both these solvers yield a *gradient–related direction* to the optimization framework. I.e., the direction p_h in (1.2) is eventually *bounded* and of *sufficient descent*, namely for any $h \geq \bar{h} \geq 1$ the next definition holds.

Definition 1.1 Given the direction p_h in (1.2), which approximately solves (1.3), let $\bar{h}$, σ_1, σ_2 and M be finite positive values independent of h, with $\bar{h} \geq 1$. Then, we say that p_h is gradient–related at x_h if for any $h \geq \bar{h}$ the next couple of conditions hold:

$$\|p_h\| \leq M, \tag{1.4}$$

$$\nabla f(x_h)^T p_h \leq -\sigma_1 \|\nabla f(x_h)\|^{\sigma_2}. \tag{1.5}$$

For this purpose, as regards the CG, the reader may refer to [11,12,18]. On the other hand, we also need to analyze the Lanczos process, adopted inside SYMMBK algorithm. As well known, one of the advantages of using the Lanczos process in place of CG algorithm is that it does not break down in the indefinite case. We recall that in the case the matrix $\nabla^2 f(x_h)$ is positive definite, from a theoretical point of view, using the Lanczos process or CG algorithm is to a large extent equivalent [23,24].

The paper is organized as follows. In Sect. 2 some preliminaries on Truncated Newton method and Lanczos process are reported; then, the Bunch and Kaufman decomposition, as well as some basics on SYMMBK (see also [6]), are detailed. In Sect. 3 we show how to compute a gradient–related direction by using the Bunch and Kaufman decomposition. Section 4 details numerical experiments. Finally, Sect. 4 reports some concluding remarks.

We indicate by $\|\cdot\|$ the Eclidean norm of real vectors and matrices. Moreover, $\lambda_\ell(C)$ represents the ℓth eigenvalue of the real symmetric matrix C. Finally, e_i is the ith real unit vector.

2 Preliminaries

In this section we report some basics we will use in the sequel. In particular, first we recall a general scheme of a linesearch–based Truncated Newton algorithm. Afterwards, we report the well known Lanczos process. Finally, we detail the Bunch and Kaufman factorization in order to decompose the tridiagonal matrix provided by the Lanczos process.

2.1 Truncated Newton method

A practical linesearch–based Truncated Newton algorithm can be described as follows (see e.g. [21]).

```
Linesearch–based Truncated Newton algorithm

Data: Choose $x_1 \in \mathbb{R}^n$ and compute $\nabla f(x_1)$. Set $\nu_h \in [0, 1)$ for any $h \geq 1$,
with $\{\nu_h\} \to 0$;
OUTER ITERATIONS
Do $h = 1, 2, \ldots$
        If $\|\nabla f(x_h)\|$ is small then STOP.
            INNER ITERATIONS
            Compute $p_h$ which approximately solves equation (1.3) and
            satisfies the truncation rule $\|\nabla^2 f(x_h) p_h + \nabla f(x_h)\| <$
            $\nu_h \|\nabla f(x_h)\|$.
        Compute the steplength $\alpha_h$ by using a linesearch procedure.
        Set $x_{h+1} = x_h + \alpha_h p_h$.
End Do
```

In large scale settings, iterative methods like the CG algorithm can be used in the inner iterations. In the case the Hessian matrix in (1.3) is indefinite, then the Lanczos algorithm may successfully be used as an alternative to the CG method, since it does not suffer for a possible pivot breakdown. We remark that within the Truncated Newton methods, the importance of an efficient truncation criterion for the inner iterations was pointed out in [8,9,22]. More recently, an adaptive truncation rule, in order to enhance the residual–based criterion, was proposed in [3,4].

Assumption 2.1 Given the function $f : \mathbb{R}^n \to \mathbb{R}$ in (1.1), with $f \in C^2(\mathbb{R}^n)$, the sequence $\{x_h\}$ in (1.2) is such that $\{x_h\} \subset \Omega$, with $\Omega \subset \mathbb{R}^n$ compact.

For completeness we close this section by recalling a very general result of global convergence for Truncated Newton methods (see any textbook on Continuous Optimization).

Proposition 2.2 *Let us consider the sequences $\{x_h\}$ and $\{p_h\}$ in (1.2). Assume $\{x_h\}$ satisfies Assumption 2.1 and $\{p_h\}$ fulfills Definition 1.1. If the steplength α_h in (1.2) is computed by an Armijo-type linesearch procedure, then*

- *The sequence $\{f(x_k)\}$ converges;*
- *The sequence $\{\nabla f(x_k)\}$ satisfies $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$.*

2.2 Basics of the Lanczos process

As well known, the Lanczos process (see [6]) is an efficient tool to transform the symmetric indefinite linear system (1.3) into a symmetric tridiagonal one. On this guideline, consider the Newton equation (1.3) and rewrite it as follows (here we drop for simplicity the dependency on the subscript h):

$$Ad = b, \tag{2.1}$$

where

- $A = \nabla^2 f(x_h)$ is a symmetric (possibly dense) *nonsingular* matrix;
- $b = -\nabla f(x_h)$ is a constant (bounded) vector.

A practical implementation of the Lanczos process applied to (2.1), which generates the orthogonal Lanczos vectors q_i, $i \geq 1$, is reported in the scheme below.

```
                    Lanczos process

Data: $\varepsilon \in (0,1)$. Set $k = 1$, $u_1 = b$, $q_1 = \frac{u_1}{\|u_1\|}$, $\delta_1 = q_1^T A q_1$ and $u_2 = A q_1 - \delta_1 q_1$.
Do
    $k = k + 1$;
    $\gamma_k = \|u_k\|$;
    If $\gamma_k < \varepsilon$ STOP.
    Else set
        $q_k = \frac{u_k}{\gamma_k}$;
        $\delta_k = q_k^T A q_k$;
        $u_{k+1} = A q_k - \delta_k q_k - \gamma_k q_{k-1}$;
    End If
End Do
```

Suppose the Lanczos process has performed k iterations; then,

- The k vectors $q_1, \ldots, q_k$, namely the *Lanczos vectors*, were generated;
- The k scalars $\delta_1, \ldots, \delta_k$ and the $k-1$ nonzero scalars $\gamma_2, \ldots, \gamma_k$ were generated.

Defining $Q_k \in \mathbb{R}^{n \times k}$ as the matrix whose columns are the Lanczos vectors, that is

$$Q_k = \begin{pmatrix} q_1 & \vdots & \vdots & q_k \end{pmatrix}, \tag{2.2}$$

and the symmetric tridiagonal matrix $T_k \in \mathbb{R}^{k \times k}$

$$T_k = \begin{pmatrix} \delta_1 & \gamma_2 & & & \\ \gamma_2 & \delta_2 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \delta_{k-1} & \gamma_k \\ & & & \gamma_k & \delta_k \end{pmatrix}, \tag{2.3}$$

the Lanczos process generates T_k from (2.1) in $k \le n$ steps. On the other hand, after $k \le n$ iterations the next additional relations also hold for the Lanczos process:

$$AQ_k = Q_k T_k + \gamma_{k+1} q_{k+1} e_k^T; \tag{2.4}$$

$$Q_k^T A Q_k = T_k; \tag{2.5}$$

$$Q_k^T Q_k = I; \tag{2.6}$$

$$Q_k^T q_{k+1} = 0; \tag{2.7}$$

$$\text{span}\,\{q_1, q_2, \ldots, q_k\} = \text{span}\left\{u_1, Au_1, \ldots, A^{k-1} u_1\right\}. \tag{2.8}$$

In particular, if $\gamma_{k+1} = 0$ in (2.4), then from (2.4)–(2.8) we have (see also [5])

$$\begin{cases} T_k y_k = \|b\| e_1 \\ d_k = Q_k y_k. \end{cases} \tag{2.9}$$

Relations (2.9) also reveal that, since T_k is nonsingular, as long as $\gamma_{k+1} \ne 0$ in (2.4) then computing the vector y_k from $T_k y_k = \|b\| e_1$ may easily yield $d_k = Q_k y_k$, being d_k an *approximate* solution of the linear system in (2.1).

2.3 Bunch and Kaufman decomposition

As reported in Sect. 2.1, unlike the CG method, the Lanczos process does not break down if matrix A in (2.1) is indefinite *nonsingular*. Furthermore, by (2.5)–(2.6) the nonsingularity of A also yields the nonsingularity of matrix T_k. On the other hand, observe that a suitable decomposition of the tridiagonal matrix T_k in (2.3) is mandatory, in order to easily compute y_k by solving the tridiagonal system $T_k z = \|b\| e_1$. However, in the case the matrix A is indefinite, the Cholesky factorization of T_k may not exist. In this regard, the Bunch and Kaufman decomposition of T_k may be adopted, as in SYMMBK algorithm. We detail now for T_k the Bunch and Kaufman decomposition (see [2])

$$T_k = S_k B_k S_k^T, \tag{2.10}$$

where B_k is a block diagonal matrix with 1×1 or 2×2 diagonal blocks, and S_k is a unit lower triangular matrix, such that its non-zeroes are restricted to three main diagonals. The diagonal blocks of B_k represent pivoting elements in the factorization (2.10). In particular, in order to control the growth factor when computing the decomposition (2.10), the scalar

$$\eta = \frac{\sqrt{5} - 1}{2 \max_i |\lambda_i(A)|}$$

is selected (see [5]), and the choice of 1×1 pivot or 2×2 pivot for B_k is based on the following rule (which only refers, without loss of generality, to the first step of the decomposition (2.10)):

- If $|\delta_1| > \eta(\gamma_2)^2$, then δ_1 in (2.3) is used as a 1×1 pivot to generate the factorization (2.10) of T_k. Namely, we have $T_k = S_k B_k S_k^T$ with

$$T_k = S_k B_k S_k^T = \left(\begin{array}{c|c} 1 & 0 \cdots 0 \\ \hline S_{2,1} & \\ 0 & I \\ \vdots & \\ 0 & \end{array}\right) \left(\begin{array}{c|c} \delta_1 & 0 \cdots 0 \\ \hline 0 & \\ 0 & \tilde{T}_{k-1} \\ \vdots & \\ 0 & \end{array}\right) \left(\begin{array}{c|c} 1 & S_{2,1} \; 0 \cdots 0 \\ \hline 0 & \\ 0 & I \\ \vdots & \\ 0 & \end{array}\right). \qquad (2.11)$$

After a brief computation we have

$$\begin{aligned} T_k &= \left(\begin{array}{c|c} 1 & 0 \cdots 0 \\ \hline S_{2,1} & \\ 0 & I \\ \vdots & \\ 0 & \end{array}\right) \left(\begin{array}{c|c} \delta_1 & \delta_1 S_{2,1} \; 0 \cdots 0 \\ \hline 0 & \\ 0 & \tilde{T}_{k-1} \\ \vdots & \\ 0 & \end{array}\right) \\ &= \left(\begin{array}{c|c} \delta_1 & \delta_1 S_{2,1} \; 0 \cdots\cdots\cdots\cdots 0 \\ \hline \delta_1 S_{2,1} & \\ \vdots & \left(\begin{array}{c|c} \delta_1 S_{2,1}^2 & 0 \cdots 0 \\ \hline 0 & \\ 0 & \emptyset \\ \vdots & \\ 0 & \end{array}\right) + \tilde{T}_{k-1} \\ \vdots & \\ 0 & \end{array}\right), \end{aligned}$$

so that $\tilde{T}_{k-1}$ is itself a tridiagonal matrix with

$$\tilde{T}_{k-1} = T_k[2:k, 2:k] - \left(\begin{array}{c|c} \delta_1 S_{2,1}^2 & 0 \cdots 0 \\ \hline 0 & \\ 0 & \emptyset \\ \vdots & \\ 0 & \end{array}\right) \qquad (2.12)$$

and $T_k[2:k, 2:k]$ is the $(k-1) \times (k-1)$ submatrix of T_k corresponding to the last $k-1$ rows and $k-1$ columns.

- If $|\delta_1| \le \eta(\gamma_2)^2$, then the 2×2 block

$$\begin{pmatrix} \delta_1 & \gamma_2 \\ \gamma_2 & \delta_2 \end{pmatrix} \qquad (2.13)$$

is used as a 2×2 pivot, to generate the factorization (2.10), so that

$$T_k = S_k B_k S_k^T = \left(\begin{array}{cc|c} 1 & 0 & 0\dots 0 \\ 0 & 1 & 0\dots 0 \\ \hline S_{3,1} & S_{3,2} & \\ 0 & 0 & I \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right) \left(\begin{array}{cc|c} \delta_1 & \gamma_2 & 0\dots 0 \\ \gamma_2 & \delta_2 & 0\dots 0 \\ \hline 0 & 0 & \\ 0 & 0 & \tilde{T}_{k-2} \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right) \left(\begin{array}{cc|c} 1 & 0 & S_{3,1}\ 0\dots 0 \\ 0 & 1 & S_{3,2}\ 0\dots 0 \\ \hline 0 & 0 & \\ 0 & 0 & \\ \vdots & \vdots & I \\ 0 & 0 & \end{array}\right), \tag{2.14}$$

being now $\tilde{T}_{k-2} \in \mathbb{R}^{(k-2)\times(k-2)}$ a tridiagonal matrix satisfying

$$\tilde{T}_{k-2} = T_k[3:k, 3:k] - \left(\begin{array}{c|c} \begin{pmatrix} S_{3,1} \\ S_{3,2} \end{pmatrix}^T \begin{pmatrix} \delta_1 & \gamma_2 \\ \gamma_2 & \delta_2 \end{pmatrix} \begin{pmatrix} S_{3,1} \\ S_{3,2} \end{pmatrix} & 0\cdots 0 \\ \hline 0 & \\ 0 & \emptyset \\ \vdots & \\ 0 & \end{array}\right). \tag{2.15}$$

Following a simple recursion, the tridiagonal matrices $\tilde{T}_{k-1}$ in (2.12) and $\tilde{T}_{k-2}$ in (2.15) can be similarly decomposed, so that the entries of S_k in (2.10) can be fully computed after m pivoting steps of the Bunch and Kaufman decomposition, with $m \le k$ (see [5] and relations (2.19)–(2.20) or (2.25)–(2.26)).

2.4 Basics of SYMMBK

Here we report some basics of the SYMMBK algorithm (first proposed by Chandra in [5]), that uses the Bunch and Kaufman factorization in (2.10) to compute d_k in (2.9). Technicalities in this section are needed to provide formulae (2.22)–(2.24) and (2.28)–(2.30), which are essential to prove properties for our proposal in Sect. 3. To this aim, recalling (2.10) we introduce the $n \times k$ matrix $W_k = \left[w_1 \ \vdots \ \vdots \ w_k \right]$ and the vector $\zeta^{(k)} = (\zeta_1, \dots, \zeta_k)^T$ such that

$$W_k S_k^T = Q_k \tag{2.16}$$

$$\zeta^{(k)} = S_k^T y_k. \tag{2.17}$$

By (2.10), (2.16) and (2.17), the equations in (2.9) can be rewritten as follows:

$$\begin{cases} S_k B_k \zeta^{(k)} = \|b\| e_1 \\ W_k S_k^T = Q_k \\ d_k = W_k \zeta^{(k)}. \end{cases} \tag{2.18}$$

Then, to calculate the vector d_k of (2.9), we equivalently need to compute both W_k and $\zeta^{(k)}$. In this regard, we must provide in (2.18) the matrices S_k and B_k, distinguishing between the cases where either a 1×1 or a 2×2 pivoting step was performed for their computation. This can be easily done by considering that the first step is performed as in (2.11) (1×1 pivot) or (2.14) (2×2 pivot). Then, for any successive step we use information from the previous pivot, as reported hereafter (we assume that applying the rule (j)–(jj) in Sect. 3 the Bunch and Kaufman decomposition performed m steps, with $m \le k$). To preserve the notation used by Chandra [5], here we simply focus on the last pivot; then, we infer results for the intermediate pivots, too.

If the last pivot performed is 1×1 (i.e. according with the rule (j) we have δ_k large enough), then from Sect. 2.3 the first relation $S_k B_k \zeta^{(k)} = \|b\| e_1$ in (2.18) is equivalent to

$$\left(\begin{array}{ccc|c} & S_{k-1} & & 0 \\ & & & 0 \\ & & & \vdots \\ & & & 0 \\ \hline 0\dots0 & s_1 & s_2 & 1 \end{array}\right)\left(\begin{array}{ccc|c} B_{k-3} & & & 0 \\ & & & 0 \\ & b_1 & b_3 & \vdots \\ & b_2 & b_4 & 0 \\ \hline 0\dots0 & 0 & 0 & \delta_k \end{array}\right)\begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-1} \\ \zeta_k \end{pmatrix} = \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \tag{2.19}$$

where $s_1, s_2, b_1, b_2, b_3, b_4$ are computed from the second last step. In particular, if the second last pivot was 1×1, then $s_1 = b_2 = b_3 = 0$. From (2.19) we have

$$\left(\begin{array}{ccc|c} & S_{k-1}B_{k-1} & & 0 \\ & & & 0 \\ & & & \vdots \\ & & & 0 \\ \hline 0\dots0 & (s_1b_1 + s_2b_2) & (s_1b_3 + s_2b_4) & \delta_k \end{array}\right)\begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-1} \\ \zeta_k \end{pmatrix} = \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \tag{2.20}$$

where

$$B_{k-1} = \left(\begin{array}{c|cc} B_{k-3} & & \\ \hline & b_1 & b_3 \\ & b_2 & b_4 \end{array}\right). \tag{2.21}$$

After some computations, replacing the expressions of B_k and S_k in (2.18) we obtain the following formulae

$$w_k = q_k - s_1 w_{k-2} - s_2 w_{k-1} \tag{2.22}$$

$$\zeta_k = -\frac{(s_1b_1 + s_2b_2)\zeta_{k-2} + (s_1b_3 + s_2b_4)\zeta_{k-1}}{\delta_k} \tag{2.23}$$

$$z_k = z_{k-1} + \zeta_k w_k. \tag{2.24}$$

With an obvious generalization the formulae (2.22)–(2.24) hold replacing the step k with the step $i \le k$, whenever a 1×1 pivot is performed by the Bunch and Kaufman decomposition.

Similarly, if the last pivot performed is now 2×2 (i.e. according with the rule (jj) we have δ_{k-1} relatively small and $|\delta_{k-1}\delta_k - \gamma_k^2|$ large enough), then $S_k B_k \zeta^{(k)} = \|b\| e_1$ in (2.18) is equivalent to

$$\left(\begin{array}{ccc|cc} & & & 0 & 0 \\ & & & 0 & 0 \\ & S_{k-2} & & \vdots & \vdots \\ & & & 0 & 0 \\ \hline 0\dots0 & s_1 & s_2 & 1 & 0 \\ 0\dots0 & 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{ccc|cc} B_{k-4} & & & 0 & 0 \\ & & & 0 & 0 \\ & b_1 & b_3 & \vdots & \vdots \\ & b_2 & b_4 & 0 & 0 \\ \hline 0\dots0 & 0 & 0 & \delta_{k-1} & \gamma_k \\ 0\dots0 & 0 & 0 & \gamma_k & \delta_k \end{array}\right) \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-2} \\ \zeta_{k-1} \\ \zeta_k \end{pmatrix} = \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{2.25}$$

where s_1, s_2, b_1, b_2, b_3, b_4 are computed from the second last step. In particular, if the second last pivot was 1×1, then $s_1 = b_2 = b_3 = 0$. By multiplying in (2.25) we have

$$\left(\begin{array}{ccc|cc} & & & 0 & 0 \\ & & & 0 & 0 \\ & S_{k-2}B_{k-2} & & \vdots & \vdots \\ & & & 0 & 0 \\ \hline 0\dots0 & (s_1b_1 + s_2b_2) & (s_1b_3 + s_2b_4) & \delta_{k-1} & \gamma_k \\ 0\dots0 & 0 & 0 & \gamma_k & \delta_k \end{array}\right) \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-2} \\ \zeta_{k-1} \\ \zeta_k \end{pmatrix} = \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{2.26}$$

where now

$$B_{k-2} = \left(\begin{array}{c|cc} B_{k-4} & & \\ \hline & b_1 & b_3 \\ & b_2 & b_4 \end{array}\right). \tag{2.27}$$

After some computations, replacing again the expressions of B_k and S_k in (2.18), we obtain the following formulae

$$\begin{aligned} w_{k-1} &= q_{k-1} - s_1 w_{k-3} - s_2 w_{k-2} \\ w_k &= q_k \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (2.28) \\ \zeta_{k-1} &= -\delta_k \frac{(s_1b_1 + s_2b_2)\zeta_{k-3} + (s_1b_3 + s_2b_4)\zeta_{k-2}}{\delta_{k-1}\delta_k - (\gamma_k)^2} \\ \zeta_k &= \gamma_k \frac{(s_1b_1 + s_2b_2)\zeta_{k-3} + (s_1b_3 + s_2b_4)\zeta_{k-2}}{\delta_{k-1}\delta_k - (\gamma_k)^2} \qquad (2.29) \\ z_k &= z_{k-2} + \zeta_{k-1} w_{k-1} + \zeta_k w_k. \qquad (2.30) \end{aligned}$$

As for (2.22)–(2.24), the formulae (2.28)–(2.30) hold replacing the step k with the step $i \le k$, as long as at step i a 2×2 pivot is used. In the end, we conclude that SYMMBK algorithm generates the vectors z_i, $i = 1, \dots, k$, which represent a sequence of approximations of the vector d_k satisfying (2.18). Moreover, the direction $d_k = z_k$ satisfies (2.18) (or equivalently (2.9)).

3 Our proposal

As detailed in Sect. 2.3, the Bunch and Kaufman decomposition used within the SYMMBK algorithm is able to cope with Newton's equation (2.1), also in the indefinite case. However, the search direction provided by (2.9) might not be gradient–related, at the hth outer iteration. In this section we aim at suitably adapting SYMMBK algorithm, in order to guarantee that, using the computed approximate solution of (1.3), we can yield a gradient–related direction. For this purpose, first of all we need to slightly modify the partial pivoting rule used for choosing 1×1 or 2×2 pivot in the Bunch and Kaufman decomposition. In particular, given $\omega \in (0, 1)$ the choice for 1×1 pivot or 2×2 pivot in Sect. 2.3, in order to obtain (2.11) or (2.14), is replaced by the following rule (again for simplicity we refer only to the first step of the decomposition):

(j) if $|\delta_1| > \omega\eta(\gamma_2)^2$, then δ_1 is used as a 1×1 pivot;
(jj) if $|\delta_1| \le \omega\eta(\gamma_2)^2$, then the 2×2 block (2.13) is used as a 2×2 pivot.

Now we show that the novel condition $|\delta_1| > \omega\eta(\gamma_2)^2$ in (j) (which yields (2.11)), along with the novel condition $|\delta_1| \le \omega\eta(\gamma_2)^2$ in (jj) (which yields (2.14)), allow to use an adapted SYMMBK algorithm for the construction of a *gradient–related* direction. To this aim, we recall that repeatedly performing the factorizations (2.11) and (2.14) we finally obtain for T_k the decomposition (2.10), where $(m \le k)$

$$B_k = \mathrm{diag}_{1\le j\le m}\{B_k^j\}, \qquad B_k^j \text{ is a } 1 \times 1 \text{ or } 2 \times 2 \text{ block.} \tag{3.1}$$

Moreover, by (2.6) the columns of Q_k are orthogonal unit vectors, and from Assumption 2.1 then b in (2.1) has a bounded norm. Thus, according to (1.4) and (1.5) of Definition 1.1, p_h in (1.2) is gradient–related at x_h as long as the next two conditions are satisfied:

(i) The matrix T_k in (2.9) is nonsingular with the vector $y_k = T_k^{-1}(\|b\|e_1)$ bounded;
(ii) p_h is a descent direction at x_h and (1.5) holds.

Of course, since the matrix A in (2.1) is possibly indefinite, the direction d_k in (2.9) might be of ascent and/or not gradient–related. However, we show how to possibly modify d_k in order to obtain the gradient–related direction p_h, to be used in (1.2).

3.1 How SYMMBK can provide a gradient–related direction

Here we prove that both adopting the rule (j)–(jj) in the Bunch and Kaufman decomposition, and slightly adapting the SYMMBK algorithm for the solution of Newton's equation (2.1), we can provide a gradient–related direction p_h at x_h for the optimization problem (1.1). In this regard, we first refer the reader to Sect. 2.4, which shows that pairing (j)–(jj) with SYMMBK, to iteratively compute the vector d_k in (2.9), we generate the sequence of vectors z_i, $i \le k$, such that

- If at step i a 1×1 pivot is performed by the Bunch and Kaufman decomposition in SYMMBK, then the vector $z_i = z_{i-1} + \zeta_i w_i$ is generated,
- If at step i a 2×2 pivot is performed by the Bunch and Kaufman decomposition in SYMMBK, then the vector $z_i = z_{i-2} + \zeta_{i-1} w_{i-1} + \zeta_i w_i$ is generated,

- When $i = k$ we have $d_k = z_k$,

and the quantities $\{\zeta_i\}$, $\{w_i\}$ are computed using the entries of matrices S_k and B_k in (2.10). As regards the vectors $\{z_i\}$, the next result gives a relevant property of these vectors.

Proposition 3.1 *Let the matrix A in (2.1) be nonsingular, and let z_i, $i \le k$ be the directions generated by the SYMMBK algorithm when solving (2.9). Assume that the Bunch and Kaufman decomposition (2.10) is adopted according with the rule (j)–(jj). Then, for proper values of the parameter ω in (j)–(jj) we have that*

- *Any direction in the finite sequences $\zeta_1 w_1, \dots, \zeta_k w_k$ and $z_1, \dots, z_k$ is bounded;*
- *The vector d_k in (2.9) coincides with z_k (and is bounded).*

Proof In order to obtain the overall proof, without loss of generality it suffices to assume $k > 1$ and separately consider only the first step in the cases (2.11) and (2.14), since a similar reasoning holds for any step i (see Sect. 2.4). Indeed, since $\tilde{T}_{k-1}$ in (2.12) and $\tilde{T}_{k-2}$ in (2.15) are tridiagonal, then we can apply for them the same reasoning adopted, at the first step, for T_k. In this regard we have:

- If the first step is a 1×1 pivot step, comparing T_k in (2.11) and T_k in (2.3) we get

$$\delta_1 S_{2,1} = \gamma_2, \tag{3.2}$$

  that is

$$S_{2,1} = \frac{\gamma_2}{\delta_1}. \tag{3.3}$$

  Since $\gamma_2 \ge \varepsilon$ (see the Lanczos process) and $|\delta_1| > \omega\eta(\gamma_2)^2$ we have

$$|S_{2,1}| = \frac{|\gamma_2|}{|\delta_1|} < \frac{1}{\omega\eta|\gamma_2|} \le \frac{1}{\omega\eta\varepsilon}, \tag{3.4}$$

  i.e. in case of 1×1 pivot, we obtain in (2.11) that $|\delta_1| = |q_1^T A q_1| \le \max_\ell\{|\lambda_\ell(A)|\}$ and $|S_{2,1}|$ is bounded;
- If the first step is a 2×2 pivot step, comparing T_k in (2.14) and T_k in (2.3) we get

$$\begin{cases} S_{3,1}\delta_1 + S_{3,2}\gamma_2 = 0 \\ S_{3,1}\gamma_2 + S_{3,2}\delta_2 = \gamma_3, \end{cases} \tag{3.5}$$

  that is

$$\begin{cases} S_{3,1} = -\dfrac{\gamma_3\gamma_2}{\delta_1\delta_2 - (\gamma_2)^2} \\ S_{3,2} = \dfrac{\gamma_3\delta_1}{\delta_1\delta_2 - (\gamma_2)^2}. \end{cases} \tag{3.6}$$

  Now observe that

$$\gamma_2 = \|Aq_1 - \delta_1 q_1\| \ \le \ \|Aq_1\| + \|(q_1^T A q_1)q_1\|$$

$$
\begin{aligned}
&\le \max_\ell\{|\lambda_\ell(A)|\}\|q_1\| + \left(\max_\ell\{|\lambda_\ell(A)|\}\right)\|q_1\|^3 \;\le\; 2\left[\max_\ell\{|\lambda_\ell(A)|\}\right],\\
\gamma_3 &= \|Aq_2 - \delta_2 q_2 - \gamma_2 q_1\| \;\le\; 4\left[\max_\ell\{|\lambda_\ell(A)|\}\right],\\
&\vdots\\
\gamma_{k+1} &= \|Aq_k - \delta_k q_k - \gamma_k q_{k-1}\| \;\le\; 2k\cdot\left[\max_\ell\{|\lambda_\ell(A)|\}\right],
\end{aligned}
\tag{3.7}
$$

which implies that since $k \le n$ then γ_j is bounded, for any $j \ge 1$. Moreover, in case $\delta_1 = 0$ or $\delta_2 = 0$ then $|\delta_1\delta_2 - (\gamma_2)^2| = |-(\gamma_2)^2| > \varepsilon^2$, and by (3.6)–(3.7), along with $|\delta_1| \le \omega\eta(\gamma_2)^2$, we obtain that both $S_{3,1}$ and $S_{3,2}$ are bounded by a constant,

$$
\begin{cases}
|S_{3,1}| \le \dfrac{|\gamma_3\gamma_2|}{(\gamma_2)^2} = \dfrac{|\gamma_3|}{\gamma_2} \le \dfrac{4\max_\ell\{|\lambda_\ell(A)|\}}{\varepsilon}\\
|S_{3,2}| \le \gamma_3\omega\eta \le 4\omega\eta\max\limits_\ell\{|\lambda_\ell(A)|\}.
\end{cases}
\tag{3.8}
$$

On the other hand, in case $\delta_1 \ne 0$ and $\delta_2 \ne 0$ in (3.6), recalling that $\gamma_2 \ge \varepsilon$ and using the condition $|\delta_1| \le \omega\eta(\gamma_2)^2$ we distinguish two subcases:

- $\delta_2 > 0$ which implies $\delta_1\delta_2 - (\gamma_2)^2 \le -(\gamma_2)^2(1 - \omega\eta\delta_2)$, so that imposing $1 > 1 - \omega\eta\delta_2 > \xi$, for a given $0 < \xi < 1$, we obtain $\omega < (1-\xi)/(\eta\delta_2)$ and $|\delta_1\delta_2 - (\gamma_2)^2| > \varepsilon^2\xi$. Thus, if in the test (j)–(jj) we finally set

$$
0 < \omega < \min\left\{1\,,\;\frac{1-\xi}{\eta\delta_2}\right\}, \qquad \text{for some } 0 < \xi < 1,
$$

  then, $S_{3,1}$ and $S_{3,2}$ in (3.5)–(3.6) are bounded;
- $\delta_2 < 0$ which implies $\delta_1\delta_2 - (\gamma_2)^2 \le -(\gamma_2)^2(1 + \omega\eta\delta_2)$, so that imposing now $1 > 1 + \omega\eta\delta_2 > \xi$, for a given $0 < \xi < 1$, we obtain $\omega < (\xi - 1)/(\eta\delta_2)$ and $|\delta_1\delta_2 - (\gamma_2)^2| > \varepsilon^2\xi$. Thus, if in the test (j)–(jj) we finally set

$$
0 < \omega < \min\left\{1\,,\;\frac{\xi-1}{\eta\delta_2}\right\}, \qquad \text{for some } 0 < \xi < 1,
$$

  then, again $S_{3,1}$ and $S_{3,2}$ in (3.5)–(3.6) are bounded.

Thus, in both the above subcases $S_{3,1}$ and $S_{3,2}$ in (3.6) are evidently bounded by a constant value, with

$$
\begin{cases}
|S_{3,1}| \le \dfrac{8\max_\ell\{|\lambda_\ell(A)|\}^2}{\varepsilon^2\xi}\\[2ex]
|S_{3,2}| \le \dfrac{4\max_\ell\{|\lambda_\ell(A)|\}^2}{\varepsilon^2\xi}.
\end{cases}
\tag{3.9}
$$

Finally, iterating the above procedure for any step, by (3.2), (3.6), (3.8), (3.9) and using the test (j)–(jj) in the Bunch and Kaufman decomposition to decide pivoting 1×1 or 2×2, we obtain that

(a) By (2.11)–(2.12) and (2.14)–(2.15), any pivoting element B_k^j in (3.1) has a determinant which is bounded away from zero;

(b) All the entries of the matrix S_k in (2.11) and (2.14) are bounded by constant values independent of k.

This implies (see Sect. 2.4) that any vector in the sequences $\zeta_1 w_1, \ldots, \zeta_k w_k$ and $z_1, \ldots, z_k$ is bounded. Moreover, the vector $d_k = z_k$ fulfilling (2.9) is unique and bounded.

Observe that by the previous result, the condition (i) in Sect. 3 is partly guaranteed to hold. However, using SYMMBK algorithm, the direction d_k computed in Proposition 3.1 might not be a direction of sufficient descent. To guarantee also the fulfillment of (ii), we propose to suitably use the information collected by the sequence $\{\zeta_i w_i\}$, in order to compute p_h endowed with descent property.

From Proposition 3.1 and Sect. 2.4, the real vector $\zeta_i w_i = z_i - z_{i-1}$ (respectively the vector $\zeta_{i-1} w_{i-1} + \zeta_i w_i = z_i - z_{i-2}$) is bounded. Thus, we propose to compute the search direction p_h, at the outer iteration h of the Truncated Newton method, in accordance with the next scheme:

```
                    Reverse–Scheme

Data: Set the initial vector $p_h = z_0 = 0$, along with the parameter $\phi > 0$;
Do $i \geq 1$
    If at step $i$ of the Bunch and Kaufman decomposition in SYMMBK a $1 \times 1$
    pivot is performed, when solving the linear system $T_k z = \|b\| e_1$, then
            If $\nabla f(x_h)^T (\zeta_i w_i) > 0$ then set $uu = -\zeta_i w_i$ else $uu = \zeta_i w_i$.
            Set $p_h = p_h + uu$.

    If at step $i$ of the Bunch and Kaufman decomposition in SYMMBK a $2 \times 2$
    pivot is performed, when solving the linear system $T_k z = \|b\| e_1$, then set
            $\tilde{\zeta}_{i-1} = \begin{cases} sgn(\zeta_{i-1}) \max\{|\zeta_{i-1}|, \phi\} & i = 2 \\ \zeta_{i-1} & i > 2 \end{cases}$
    so that
            If $\nabla f(x_h)^T (\tilde{\zeta}_{i-1} w_{i-1}) > 0$ then set $uu = -\tilde{\zeta}_{i-1} w_{i-1}$ else
            $uu = \tilde{\zeta}_{i-1} w_{i-1}$.
            If $\nabla f(x_h)^T (\zeta_i w_i) > 0$ then set $vv = -\zeta_i w_i$ else $vv = \zeta_i w_i$.
            Set $p_h = p_h + uu + vv$.

End Do
```

We remark that in case the matrix A in (2.1) is positive definite, and the parameter ω in (j)–(jj) is sufficiently small (i.e. no 2×2 pivot is performed), then the vector p_h computed by the *Reverse–Scheme* coincides with Newton's direction.

Now, in the next proposition we prove that the direction p_h, obtained by *Reverse–Scheme* at any iterate of the sequence $\{x_h\}$ of the Truncated Newton method, is gradient–related. This result is not an immediate consequence of Proposition 3.1, since the latter proposition merely guarantees the simple boundedness of the direction d_k, for a given $k \geq 1$. Conversely, the next result proves that, under mild additional assumptions, all the vectors in the sequence $\{p_h\}$ are of *sufficient descent* and eventually are *uniformly bounded* by a positive finite constant.

Proposition 3.2 *Let Assumption 2.1 hold. Assume that Proposition 3.1 holds with $A = \nabla^2 f(x_h)$ and $b = -\nabla f(x_h)$. Let the search direction p_h in (1.2) be computed as in* Reverse–Scheme. *Then, for any $k \geq 1$ the direction d_k in (2.9) is bounded, namely it satisfies $\|d_k\| < \mu$, for some positive value of μ, and p_h is a gradient–related direction as in (1.5).*

Proof The statement trivially holds if in Proposition 3.1 the Lanczos process performs just one iteration, being $\gamma_2 < \varepsilon$. In all the other cases (i.e. the Lanczos process has performed at least 2 iterations), we first prove that p_h is of sufficient descent for $f(x)$ at x_h. In this regard, it suffices to prove the sufficient descent property for p_h after the first step of the Bunch and Kaufman decomposition in SYMMBK, since the *Reverse–Scheme* accumulates only *non-ascent directions* in p_h. To this aim, we distinguish between two cases, which may occur at iterate x_h, when applying the Bunch and Kaufman decomposition to matrix T_k in (2.9):

- The first step corresponds to a 1×1 pivot. Then, by (2.16) and (2.22)–(2.24), we have for the vector uu in *Reverse–Scheme*

$$\begin{aligned} uu &= -sgn[\nabla f(x_h)^T(\zeta_1 w_1)]\zeta_1 w_1 = -sgn[\nabla f(x_h)^T(\zeta_1 q_1)]\zeta_1 q_1 \\ &= -sgn[-\zeta_1 \|\nabla f(x_h)\|]\zeta_1 \left(-\frac{\nabla f(x_h)}{\|\nabla f(x_h)\|}\right) = -sgn[\zeta_1]\zeta_1 \frac{\nabla f(x_h)}{\|\nabla f(x_h)\|} \\ &= -|\zeta_1| \frac{\nabla f(x_h)}{\|\nabla f(x_h)\|}, \end{aligned} \tag{3.10}$$

so that

$$\begin{aligned} \nabla f(x_h)^T p_h = &-\frac{|\zeta_1|}{\|\nabla f(x_h)\|}\|\nabla f(x_h)\|^2 + \\ &\sum_{i\geq 2} -sgn\left[\nabla f(x_h)^T(\zeta_i w_i)\right]\nabla f(x_h)^T(\zeta_i w_i) \\ \leq &-|\zeta_1|\|\nabla f(x_h)\|. \end{aligned} \tag{3.11}$$

More explicitly, by (2.17) and relations (2.9)–(2.11) we have

$$\zeta^{(k)} = S_k^T y_k = S_k^T T_k^{-1}\|b\|e_1 = B_k^{-1} S_k^{-1}\|b\|e_1$$

$$
= \left(\begin{array}{c|c} \delta_1 & 0\cdots 0 \\ \hline 0 & \\ 0 & \tilde{T}_{k-1} \\ \vdots & \\ 0 & \end{array}\right)^{-1} \left(\begin{array}{c|c} 1 & 0\cdots 0 \\ \hline S_{2,1} & \\ 0 & I \\ \vdots & \\ 0 & \end{array}\right)^{-1} \|b\|e_1
$$

$$
= \left(\begin{array}{c|c} 1/\delta_1 & 0\cdots 0 \\ \hline 0 & \\ 0 & \tilde{T}_{k-1}^{-1} \\ \vdots & \\ 0 & \end{array}\right) \left(\begin{array}{c|c} 1 & 0\cdots 0 \\ \hline -S_{2,1} & \\ 0 & I \\ \vdots & \\ 0 & \end{array}\right) \|b\|e_1 \; = \; \|b\| \begin{pmatrix} \frac{1}{\delta_1} \\ * \\ \vdots \\ * \end{pmatrix},
$$

where the asterisks represent irrelevant entries. Then, $\zeta_1 = \|b\|/\delta_1$. Moreover, by Assumption 2.1 the set Ω is compact, so that $|\zeta_1| = \|b\|/|\delta_1| \geq \|b\|/\max_\ell \left|\lambda_\ell[\nabla^2 f(x_h)]\right| \geq \|b\|/\lambda_M$, for some $0 < \lambda_M < +\infty$. Thus, relation (3.11) becomes

$$
\nabla f(x_h)^T p_h \leq -|\zeta_1|\|\nabla f(x_h)\| \leq -\frac{1}{\lambda_M}\|\nabla f(x_h)\|^2. \tag{3.12}
$$

- The first step corresponds to a 2×2 pivot. Then, recalling (2.28) we have $w_2 = q_2$, so that by (3.10) for the vectors uu and vv in *Reverse–Scheme* the next relations hold

$$
\begin{cases} uu = -sgn[\nabla f(x_h)^T(\zeta_1 w_1)]\zeta_1 w_1 = -|\zeta_1|\dfrac{\nabla f(x_h)}{\|\nabla f(x_h)\|}, \\ vv = -sgn[\nabla f(x_h)^T(\zeta_2 w_2)]\zeta_2 w_2 = -sgn[\nabla f(x_h)^T(\zeta_2 q_2)]\zeta_2 q_2. \end{cases} \tag{3.13}
$$

Now, again by (2.17) and relations (2.9)–(2.10), (2.14) we get

$$
\zeta^{(k)} = S_k^T y_k \; = \; S_k^T T_k^{-1}\|b\|e_1 \; = \; B_k^{-1} S_k^{-1}\|b\|e_1
$$

$$
= \left(\begin{array}{cc|c} \delta_1 & \gamma_2 & 0\ldots 0 \\ \gamma_2 & \delta_2 & 0\ldots 0 \\ \hline 0 & 0 & \\ 0 & 0 & \tilde{T}_{k-2} \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right)^{-1} \left(\begin{array}{cc|c} 1 & 0 & 0\ldots 0 \\ 0 & 1 & 0\ldots 0 \\ \hline S_{3,1} & S_{3,2} & \\ 0 & 0 & I \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right)^{-1} \|b\|e_1
$$

$$
= \left(\begin{array}{cc|c} \multicolumn{2}{c|}{\begin{pmatrix} \delta_1 & \gamma_2 \\ \gamma_2 & \delta_2 \end{pmatrix}^{-1}} & \emptyset \\ \hline 0 & 0 & \\ 0 & 0 & \tilde{T}_{k-2}^{-1} \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right) \left(\begin{array}{cc|c} 1 & 0 & 0\ldots 0 \\ 0 & 1 & 0\ldots 0 \\ \hline -S_{3,1} & -S_{3,2} & \\ 0 & 0 & I \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right) \|b\|e_1
$$

$$= \left(\begin{array}{cc|c} \frac{1}{\delta_1\delta_2-(\gamma_2)^2} & \begin{pmatrix} \delta_2 & -\gamma_2 \\ -\gamma_2 & \delta_1 \end{pmatrix} & \emptyset \\ \hline * & * & \\ * & * & \tilde{T}_{k-2}^{-1} \\ \vdots & \vdots & \\ * & * & \end{array}\right) \|b\| e_1 \; = \; \|b\| \begin{pmatrix} \dfrac{\delta_2}{\delta_1\delta_2 - (\gamma_2)^2} \\ \dfrac{-\gamma_2}{\delta_1\delta_2 - (\gamma_2)^2} \\ * \\ \vdots \\ * \end{pmatrix},$$

where again the asterisks represent irrelevant entries. Thus, it results

$$\zeta_1 = \frac{\delta_2}{\delta_1\delta_2 - (\gamma_2)^2}\|b\|,$$
$$\zeta_2 = \frac{-\gamma_2}{\delta_1\delta_2 - (\gamma_2)^2}\|b\|.$$

Finally, similarly to (3.11), recalling that $\nabla f(x_h)^T q_2 = -\|\nabla f(x_h)\| q_1^T q_2 = 0$, we have by (3.13) and *Reverse–Scheme* the relation

$$\begin{aligned} \nabla f(x_h)^T p_h &= -sgn[\nabla f(x_h)^T(\tilde{\zeta}_1 q_1)]\nabla f(x_h)^T(\tilde{\zeta}_1 q_1) \\ &\quad + \sum_{i>2} -sgn\left[\nabla f(x_h)^T(\zeta_i w_i)\right]\nabla f(x_h)^T(\zeta_i w_i) \\ &\leq -|\tilde{\zeta}_1|\|\nabla f(x_h)\| \;\leq\; -\phi\|\nabla f(x_h)\|^2. \end{aligned} \tag{3.14}$$

Relations (3.12) and (3.14) prove that for any $h \geq 1$ the condition (1.5) is eventually fulfilled, with

$$\sigma_1 = \min\left\{\frac{1}{\lambda_M}, \phi\right\}, \qquad \sigma_2 = 2.$$

As regards the boundedness of d_k in (2.9) and p_h, for any $h \geq 1$, observe that by (2.9) we first have

$$d_k = Q_k y_k.$$

Then, $\|d_k\| = \|y_k\| \leq \|T_k^{-1}\| \cdot \|\nabla f(x_h)\| \leq \|S_k^{-T} B_k^{-1} S_k^{-1}\| \|\nabla f(x_h)\| \leq \|S_k^{-1}\|^2 \cdot \|B_k^{-1}\| \cdot \|\nabla f(x_h)\|$. Now, let m_1 be the number of steps where the test (j) is fulfilled (i.e. m_1 is the number of 1×1 pivot steps), so that $(k - m_1)/2$ is the number of 2×2 pivot steps. Also observe that the inverse S_k^{-1} of S_k in (2.11) is given by

$$S_k^{-1} = \left(\begin{array}{c|c} 1 & 0\cdots 0 \\ \hline -S_{2,1} & \\ 0 & I \\ \vdots & \\ 0 & \end{array}\right)$$

and the inverse S_k^{-1} of S_k in (2.14) is given by

$$S_k^{-1} = \left(\begin{array}{cc|c} 1 & 0 & 0\dots0 \\ 0 & 1 & 0\dots0 \\ \hline -S_{3,1} & -S_{3,2} & \\ 0 & 0 & I \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right).$$

Thus, the bound (3.4) on $S_{2,1}$, along with the bounds (3.8) and (3.9) on $S_{3,1}$ and $S_{3,2}$ yield

$$\|S_k^{-1}\| \le \beta, \tag{3.15}$$

where

$$\beta = \left(\frac{m_1}{\omega\eta\varepsilon}\right) + \left[\frac{k-m_1}{2}\max\left(4\max_\ell\{|\lambda_\ell(\nabla^2 f(x_h))|\}\left(\frac{1}{\varepsilon}+\omega\eta\right),\right.\right.$$
$$\left.\left.\frac{16}{\varepsilon^2\xi}\max_\ell\{|\lambda_\ell(\nabla^2 f(x_h))|\}^2\right)\right] + k.$$

Moreover, as regards the diagonal blocks of B_k^{-1} (see (2.11) and (2.14)), for any $i \ge 1$ we have by the proof of Proposition 3.1 and the compactness of Ω the following bounds:

$$1\times 1 \text{ pivot}: |\delta_i|^{-1} \le \frac{1}{\omega\eta\varepsilon^2}$$
$$2\times 2 \text{ pivot}: \left\|\begin{pmatrix} \delta_i & \gamma_{i+1} \\ \gamma_{i+1} & \delta_{i+1}\end{pmatrix}^{-1}\right\| = \frac{1}{|\delta_i\delta_{i+1} - (\gamma_{i+1})^2|}\left\|\begin{pmatrix} \delta_{i+1} & -\gamma_{i+1} \\ -\gamma_{i+1} & \delta_i\end{pmatrix}\right\|$$
$$\le \frac{1}{\varepsilon^2\xi}\left\{4i\cdot\max_\ell\left\{\left|\lambda_\ell(\nabla^2 f(x_h))\right|\right\} + 2\max_\ell\left\{\left|\lambda_\ell(\nabla^2 f(x_h))\right|\right\}\right\}.$$

Finally, by Assumption 2.1 we have that $\|\nabla f(x_h)\|$ is bounded on Ω by a constant value, showing that the direction d_k is bounded by a constant value independent of h. To complete the proof we show the boundedness of vectors $\{\zeta_i w_i\}$ in *Reverse–Scheme*, by a constant value independent of h. From relation $\|d_k\| = \|y_k\| \le \mu$, for any $k \ge 1$, and recalling (3.15) along with relation $S_k^T = [-S_k^{-1}]^T$, we immediately infer (using (2.17))

$$\|\zeta^{(k)}\| \le \|S_k^T\|\|y_k\| \le \beta\|y_k\| \le \beta\mu,$$

where β is given in (3.15). Similarly, by (2.18) we have $W_k = Q_k S_k^{-T}$, so that any column w_i of W_k satisfies (see (3.15))

$$\|w_i\| = \|Q_k S_k^{-T} e_i\| \le \|S_k^{-T}\| \le \beta, \qquad \text{for all} \quad i \ge 1. \tag{3.16}$$

This implies, by (3.16) and the boundedness of vectors $\zeta_1 w_1, \ldots, \zeta_k w_k$ (see Proposition 3.1), that also $\|uu\|$ and $\|vv\|$ in *Reverse–Scheme* are bounded by a positive constant, for any $x_h \in \Omega$. Thus, $\|p_h\|$ is bounded by a positive constant independent of h.

4 Numerical experiments

In order to assess the effectiveness of the approach proposed in this paper, we performed an extensive numerical testing. We used the `HSL_MI02` Fortran routine available from the HSL Mathematical Software Library [19] which implements the SYMMBK algorithm, where we modified the pivoting rule as described in Sect. 3, namely (j)–(jj) on page 9. Then, we embedded this routine within a standard implementation of a linesearch based Truncated Newton method, namely the one proposed in [15] (we refer to this paper for any implementation details), without considering any preconditioning strategy. Moreover, we replaced the CG used as iterative solver by the Truncated Newton method in [15] with the modified SYMMBK algorithm. It is worthwhile to note that in [15], in order to handle the indefinite case, the inner CG iterations are not terminated whenever a negative curvature direction is encountered. Indeed, by drawing inspiration from [18], positive and negative curvature CG directions $\{d_i\}$ are separately accumulated, and then suitably combined to obtain the search direction. We report the following scheme of the CG method adopted in [15], in order to briefly describe the latter strategy.

The Conjugate Gradient (CG) method

```
Data: $k = 1$, $y_1 = 0$, $r_1 = b - Ay_1$. If $r_1 = 0$, then STOP. Else, set $d_1 = r_1$;
Do $k = 1, 2, \ldots$
    Compute $a_k = \dfrac{r_k^T d_k}{d_k^T A d_k}$,  $y_{k+1} = y_k + a_k d_k$,  $r_{k+1} = r_k - a_k A d_k$
    If $r_{k+1} = 0$, then STOP. Else, set $\beta_k = -\dfrac{r_{k+1}^T A d_k}{d_k^T A d_k} = \dfrac{\|r_{k+1}\|^2}{\|r_k\|^2}$.
    Set $d_{k+1} = r_{k+1} + \beta_k d_k$.
End Do
```

In particular, at the kth CG inner iteration, the following vectors are defined

$$p_k^P = \sum_{i \in I_k^+} a_i d_i, \qquad p_k^N = -\sum_{i \in I_k^-} a_i d_i,$$

where for a given $\epsilon \in (0, 1)$

$$I_k^+ = \left\{ i \in \{1, \ldots, k\} \ : \ d_i^T \nabla^2 f(x_h) d_i > \epsilon \|d_i\|^2 \right\},$$
$$I_k^- = \left\{ i \in \{1, \ldots, k\} \ : \ d_i^T \nabla^2 f(x_h) d_i < -\epsilon \|d_i\|^2 \right\},$$

$|I_k^+| + |I_k^-| = k$. Then, the vector $p_k^P + p_k^N$ is used as search direction. This choice has several advantages for which we refer to [15]. In particular, it can be proved that $p_k^P + p_k^P$ is a gradient–related direction. However, as shown in [7], the use of p_k^P and p_k^N obtained by separating the contribution of negative and positive curvature directions actually corresponds to considering a modified positive definite linear system, in place of the original indefinite one. For this reason, in our numerical experiments, in order to avoid unfair conclusions, we prefer not to use the modified search direction $p_k^P + p_k^N$ and adopt a more standard approach in the literature. Therefore, in our implementation we use the approximate solution of the original linear system as search direction. An additional safeguard is needed whenever the Hessian matrix is nearly singular at the hth iterate of the Truncated Newton method. In this case, the steepest descent direction is considered as "alternative" direction (both when the CG or the modified SYMMBK algorithms are adopted). As regards the termination criterion of the inner iteration, the standard residual–based criterion is used (see e.g. [3,4,21]), which we make slightly tighter (with respect to that in [15]) by multiplying the tolerance of the criterion by 10^{-2}. In this manner, a more accurate solution of the Newton system at each outer iteration of the methods is required, and possibly a more significant comparison between the use of the modified SYMMBK in place of the standard CG can be obtained. Finally, the parameter ξ in the proof of Proposition 3.1 is set as $\xi = \max\{(1-\eta|\delta_2|), 10^{-1}\}$, so that we consequently chose $\omega = \min\{1, (1-\xi)/(\eta|\delta_2|)\}$ in (j)–(jj). In addition, the parameter ϕ in *Reverse–Scheme* was set as $\phi = 10^{-10}$: in this regard we strongly remark that in our numerical experience we always observed $\tilde{\zeta}_{i-1} \equiv \zeta_{i-1}$ in the *Reverse–Scheme*.

As regards the test problems, we selected all the large scale unconstrained test problems in the CUTEst collection [16], including both convex and nonconvex functions. In particular, we considered 112 problems whose dimension ranges from 1000 to 10,000. We report the results obtained in terms of number inner iterations, number of function evaluations and CPU time. The comparison between the use of the CG algorithm and the modified SYMMBK, when performing the inner iterations, is reported by using performance profiles [10]. Here the abscissa axis details values of the positive parameter $\tau \geq 1$. In particular, for a given value $\bar{\tau}$ of the parameter τ, each profile represents the percentage of problems solved by the corresponding method, using at most $\bar{\tau}$ times the resource (inner iterations/function evaluations/CPU time) used by the best method.

Figure 1 reports the comparison in terms of inner iterations. By observing these plots, the performance of the two approaches seems to be similar and the use of the modified SYMMBK is slightly preferable in terms of robustness. The profiles in terms of function evaluations are reported in Fig. 2. In this case, the superiority of using the modified SYMMBK algorithm is more evident. Finally, as regards the comparison in terms of CPU, Fig. 3 indicates that the use of the standard CG leads to a slight improvement in terms of efficiency, but it is definitely less performing in terms of robustness. We recall indeed that though the CG can easily provide a gradient–related search direction in convex problems, it might prematurely stop in nonconvex problems.

Now, in order to better assess the quality of the search direction obtained by using the modified SYMMBK algorithm, we performed further numerical experiments. In particular we computed the normalized directional derivative of the search direction

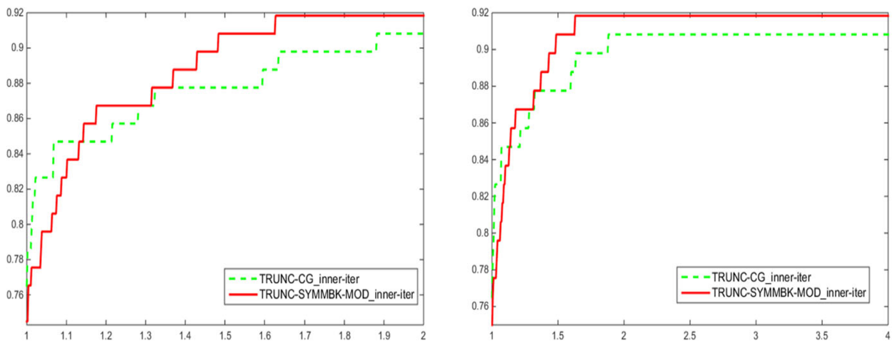

Fig. 1 Comparison between our approach based on the modified SYMMBK algorithm (continuous line) and the use of the CG algorithm (dashed line). Performance profiles are with respect to the number of inner iterations (a detail on the *left*, the full profile on the *right*)

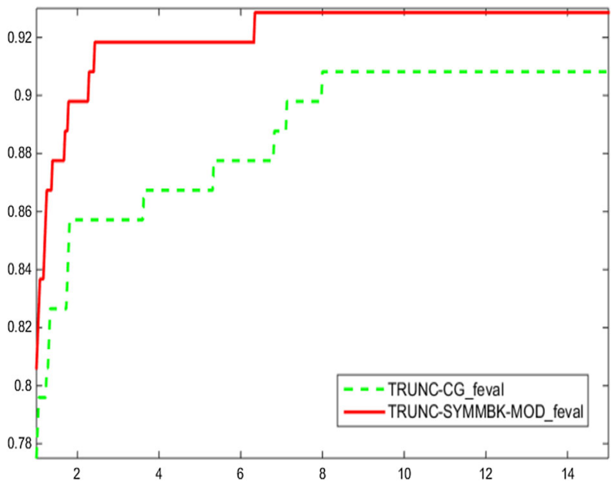

Fig. 2 Comparison between our approach based on the modified SYMMBK algorithm (continuous line) and the use of the CG algorithm (dashed line). Performance profiles are with respect to the number of function evaluations

p_h in (1.2), namely $\nabla f(x_h)^T p_h / \|\nabla f(x_h)\| \|p_h\|$, at each outer iteration of the CG and the modified SYMMBK method respectively, for some test problems. As an example, we report in Fig. 4 the normalized directional derivative obtained as the number of outer iterations increases, for the test problems DIXMAANJ-1500 (*upper-left*), DIXMAANK-3000 (*upper-right*), DIXMAANL-1500 (*mid-left*), DIXMAANL-3000 (*mid-right*), DIXMAANJ-3000 (*lower-left*), SPARSINE-1000 (*lower-right*), where the number indicates the dimension of the test problem.

On these problems the method based on the modified SYMMBK algorithm outperforms the one based on the standard CG (a much smaller number of iterations is required to achieve convergence). Plots reveal that this is due to the fact that the quality of the Newton–type direction, obtained by means of the modified SYMMBK algorithm, is definitely better. Indeed, on one hand the use of the modified SYMMBK

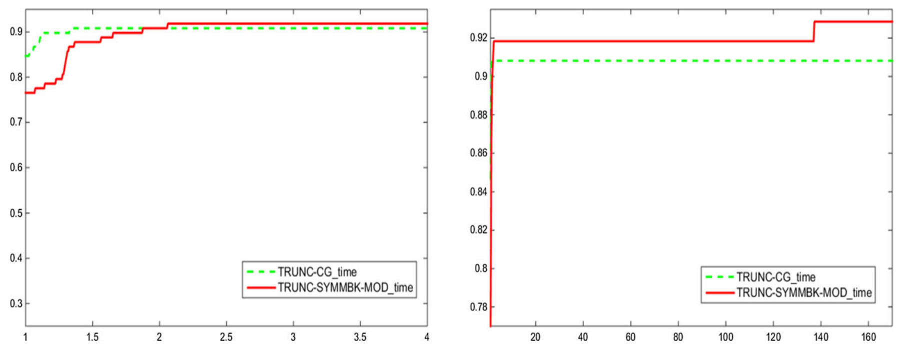

Fig. 3 Comparison between our approach based on the modified SYMMBK algorithm (continuous line) and the use of the CG algorithm (dashed line). Performance profiles are with respect to CPU time (in seconds) (a detail on the *left*, the full profile on the *right*)

algorithm for computing the search direction enables to discard the steepest descent direction more frequently than the use of the standard CG. On the other hand, the introduction of the parameter ω in the modified SYMMBK algorithm (see (j)–(jj)) allows to have values of the directional derivative which are more likely bounded away from zero. Similar behaviour is evidenced when solving many other test problems of the CUTEst collection. Note also that in the plots of Fig. 4, both the compared methods tend to initially select the steepest descent method as a search direction. This should not be surprising, inasmuch as in the early outer iterations the truncation criterion of the Truncated Newton method is less tight.

5 Conclusions

In this work, which deals with an efficient Truncated Newton method for solving large scale unconstrained optimization problems, we considered the use of an adapted Bunch and Kaufman decomposition for the solution of Newton's equation. By slightly modifying the test performed at each iteration of the Bunch and Kaufman decomposition, we were able to ensure that the Newton–type direction computed by our proposal is gradient–related. Extensive numerical testing, on both convex and nonconvex problems, highlights the effectiveness and robustness of a Truncated Newton framework, where our modified SYMMBK algorithm is used in place of the CG method.

We remark that the enhanced robustness of our proposal, with respect to the CG method, is not only the consequence of the possible premature stopping of the CG on nonconvex problems. On the contrary, we often tend to generate at the current outer iteration an approximate Newton's direction which yields a more effective directional derivative with respect to the CG.

Note that a comparison of the standard SYMMBK and our modified SYMMBK routines, within the same Truncated Newton method, yields to large extent, similar numerical results. However, the standard SYMMBK routine is unable to provide essential theoretical properties, which are sought for the search direction of the optimization framework.

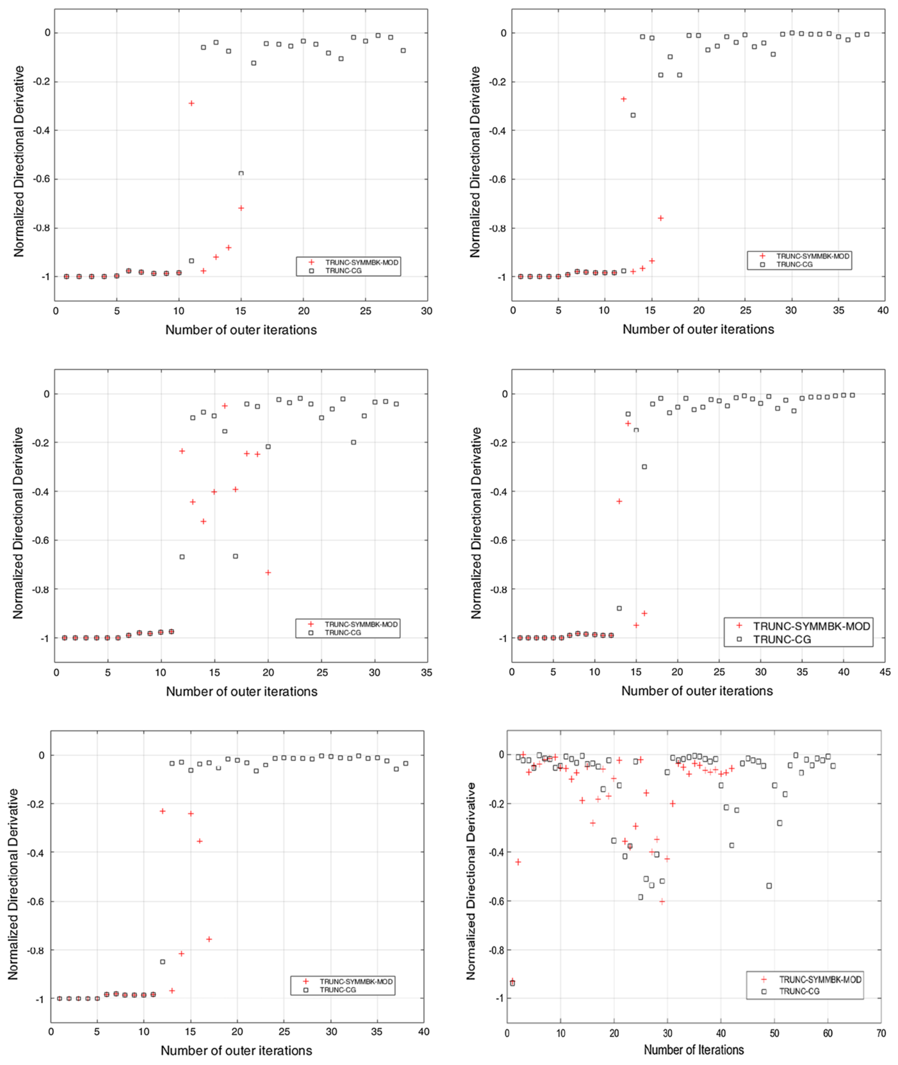

Fig. 4 Plot of the normalized directional derivative, when our proposal with the modified SYMMBK is used (+), and when the CG is used (*empty squares*). The plot refers to DIXMAANJ-1500 (*upper-left*), DIXMAANK-3000 (*upper-right*), DIXMAANL-1500 (*mid-left*), DIXMAANL-3000 (*mid-right*), DIXMAANJ-3000 (*lower-left*), SPARSINE-1000 (*lower-right*)

Finally, we are persuaded that the modified test (j)–(jj) in SYMMBK might be worth investigating, also to construct a suitable negative curvature direction (see [14]), in nonconvex problems. This issue plays a keynote role in case the adopted Truncated Newton method is asked to provide also convergence to stationary points, where second order optimality conditions hold.

Furthermore, there is the chance that adaptively updating the parameter ω in (j)–(jj) may further enhance the performance of our method, preserving the theory developed in Sect. 3.

Acknowledgements The authors wish to thank both the reviewers and the Associate Editor for their fruitful and constructive comments. G. Fasano thanks the National Research Council - Marine Technology Research Institute (CNR-INSEAN), Italy, along with the working group GNCS of INδAM (Istituto Nazionale di Alta Matematica), Italy, for the support received.

Funding Open access funding provided by Università Ca' Foscari Venezia within the CRUI-CARE Agreement.

References

1. Barret, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., Van der Vorst, H.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM, Philadelphia, PA (1994)
2. Bunch, J., Kaufman, L.: Some stable methods for calculating inertia and solving symmetric linear equations. Math. Comput. **31**, 163–179 (1977)
3. Caliciotti, A., Fasano, G., Nash, S.G., Roma, M.: An adaptive truncation criterion, for linesearch-based truncated Newton methods in large scale nonconvex optimization. Oper. Res. Lett. **46**, 7–12 (2018)
4. Caliciotti, A., Fasano, G., Nash, S.G., Roma, M.: Data and performance profiles applying an adaptive truncation criterion, within linesearch-based truncated Newton methods, in large scale nonconvex optimization. Data Brief **17**, 246–255 (2018)
5. Chandra, R.: Conjugate gradient methods for partial differential equations, Ph.D. thesis, Yale University, New Haven. Research Report 129 (1978)
6. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-Region Methods. MPS-SIAM Series on Optimization. Society for Industrial Mathematics, Philadelphia (2000)
7. De Leone, R., Fasano, G., Sergeyev, Y.: Planar methods and grossone for the conjugate gradient breakdown in nonlinear programming. Comput. Optim. Appl. **71**, 73–93 (2018)
8. Dembo, R., Eisenstat, S., Steihaug, T.: Inexact Newton methods. SIAM J. Numer. Anal. **19**, 400–408 (1982)
9. Dembo, R., Steihaug, T.: Truncated-Newton algorithms for large-scale unconstrained optimization. Math. Program. **26**, 190–212 (1983)
10. Dolan, E.D., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
11. Fasano, G.: Planar-conjugate gradient algorithm for large-scale unconstrained optimization, part 1: theory. J. Optim. Theory Appl. **125**, 523–541 (2005)
12. Fasano, G.: Planar-conjugate gradient algorithm for large-scale unconstrained optimization, part 2: application. J. Optim. Theory Appl. **125**, 543–558 (2005)
13. Fasano, G.: Lanczos-conjugate gradient method and pseudoinverse computation, in unconstrained optimization. J. Optim. Theory Appl. **132**, 267–285 (2006)
14. Fasano, G., Roma, M.: Iterative computation of negative curvature directions in large scale optimization. Comput. Optim. Appl. **38**, 81–104 (2007)
15. Fasano, G., Roma, M.: Preconditioning Newton–Krylov methods in nonconvex large scale optimization. Comput. Optim. Appl. **56**, 253–290 (2013)

16. Gould, N.I.M., Orban, D., Toint, P.L.: CUTEst: a constrained and unconstrained testing environment with safe threads. Comput. Optim. Appl. **60**, 545–557 (2015)
17. Greenbaum, A.: Iterative Methods for Solving Linear Systems. SIAM, Philadelphia (1997)
18. Grippo, L., Lampariello, F., Lucidi, S.: A truncated Newton method with nonmonotone linesearch for unconstrained optimization. J. Optim. Theory Appl. **60**, 401–419 (1989)
19. HSL, A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/
20. Marcia, R.: On solving sparse symmetric linear systems whose definiteness is unknown. Appl. Numer. Math. **58**, 449–458 (2008)
21. Nash, S.: A survey of truncated-Newton methods. J. Comput. Appl. Math. **124**, 45–59 (2000)
22. Nash, S., Sofer, A.: Assessing a search direction within a truncated-Newton method. Oper. Res. Lett. **9**, 219–221 (1990)
23. Nocedal, J., Wright, S.: Numerical Optimization, 2nd edn. Springer, New York (2006)
24. 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 (1983)

Affiliations

Andrea Caliciotti[1] · Giovanni Fasano[2] · Florian Potra[3] · Massimo Roma[1]

✉ Giovanni Fasano
fasano@unive.it

Andrea Caliciotti
caliciotti@diag.uniroma1.it

Florian Potra
potra@umbc.edu

Massimo Roma
roma@diag.uniroma1.it

1 Dipartimento di Ingegneria Informatica, Automatica e Gestionale "A. Ruberti" SAPIENZA, Università di Roma, Via Ariosto, 25, 00185 Roma, Italy

2 Department of Management, University Ca' Foscari of Venice, S. Giobbe, Cannaregio 873, 30121 Venice, Italy

3 Department of Mathematics and Statistics, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA