Iterative Grossone-Based Computation of Negative Curvature Directions in Large-Scale Optimization

Renato De Leone[1] · **Giovanni Fasano**[2] · **Massimo Roma**[3] ·
Yaroslav D. Sergeyev[4,5]

Abstract

We consider an iterative computation of negative curvature directions, in large-scale unconstrained optimization frameworks, needed for ensuring the convergence toward stationary points which satisfy second-order necessary optimality conditions. We show that to the latter purpose, we can fruitfully couple the conjugate gradient (CG) method with a recently introduced approach involving the use of the numeral called *Grossone*. In particular, recalling that in principle the CG method is well posed only when solving positive definite linear systems, our proposal exploits the use of grossone to enhance the performance of the CG, allowing the computation of negative curvature directions in the indefinite case, too. Our overall method could be used to significantly generalize the theory in state-of-the-art literature. Moreover, it straightforwardly allows the solution of Newton's equation in optimization frameworks, even in nonconvex problems. We remark that our iterative procedure to compute a negative curvature direction does not require the storage of any matrix, simply needing to store a couple of vectors. This definitely represents an advance with respect to current results in the literature.

1 Introduction

We consider the solution of the nonconvex unconstrained optimization problem $\min_{x \in \mathbb{R}^n} f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear smooth function and n is large.

 Springer

Despite the use of the term 'minimization' in the last problem, most of the methods proposed in the literature (for its solution) generate a sequence of points $\{x_k\}$, which is only guaranteed to converge to stationary points. Thus, specific methods need to be applied in case stationary points for the above problem, satisfying also second-order necessary optimality conditions, are sought (see, for instance, the seminal papers [1–7] in the framework of truncated Newton methods). Observe that additional care when using the latter methods is definitely mandatory, since imposing standard first-order stationarity conditions may not in general ensure convexity of the quadratic model of the objective function, in a neighborhood of the solution points. In this regard, the computation of so-called *negative curvature* directions for the objective function is an essential tool (see also the recent papers [4,8]), to guarantee convergence to stationary points which satisfy second-order necessary conditions.

Here, we want to provide a framework for the computation of negative curvature directions of quadratic functions, to be used within globally convergent iterative methods for large-scale nonlinear programming. Observe that the asymptotic convergence of iterative methods, toward second-order stationary points, implies that the Hessian matrix at limit points must be positive semidefinite. This fact requires that in principle the iterative methods adopted must be able to fully explore the eigenspaces of the Hessian matrix, at the current iterate, at least in a neighborhood of the stationary points. Equivalently, the optimization method adopted will have to efficiently cope also with nonconvexities of the objective function. The latter fact issues specific concerns in case n is large, since the computational effort to solve a nonlinear programming problem can be strongly affected by the scale.

In particular, as shown in [3,5,9], exploiting the nonconvexities of $f(x)$ can be accomplished by suitable Newton–Krylov methods (in the context of Hessian-free truncated Newton methods), such that at each outer iteration j, a pair of search directions (s_j, d_j) is computed, satisfying specific properties. Namely, the vector s_j must be a direction which approximately solves Newton's equation $\nabla^2 f(x_j) s = -\nabla f(x_j)$ at x_j. Its purpose is essentially that of ensuring the efficient convergence of the sequence $\{x_j\}$ to stationary points. On the other hand, the nonascent direction d_j is a negative curvature direction (if any) for the objective function at x_j. That is, d_j is a nonascent direction such that $d_j^T \nabla^2 f(x_j) d_j \leq 0$, satisfying suited conditions in order to force convergence to those stationary points where second-order necessary optimality conditions hold. In particular, d_j should resemble an eigenvector corresponding to the least negative eigenvalue of the Hessian matrix $\nabla^2 f(x_j)$. In [3,5] the direction d_j is obtained as by-product of the Krylov-subspace method applied for solving Newton's equation, though an expensive storage is required in [5] and a heavy computational burden is necessary in the approach proposed in [3].

To overcome these drawbacks, in [9] an important novelty is introduced, namely the *iterative computation* of the sequence $\{d_j\}$, not requiring neither expensive computations nor excessive storage. This novel approach is based on the use of the so-called *Planar-CG method* which is a modification of the conjugate gradient method in [11]. In particular, in [9], at any iterate $x_j \in \mathbb{R}^n$, the vector d_j is computed through the linear combination of a few n-real vectors, generated by the used Krylov subspace method. It was proved that on nonconvex problems, the overall exact computation of d_j simply

requires at iterate x_j the storage of at most four n-real vectors. Even if this approach revealed effective, it presents some drawbacks related to a cumbersome analysis.

In this paper, partially starting from the drawbacks of the proposal in [9], we aim at describing a strong simplification in the computation of the directions $\{d_j\}$, by using a novel approach which extends some ideas in [12]. Namely, we adopt the Krylov-based method $CG_①$ defined in [12], being ① the symbol for the *grossone* (see [13]), in order to generate a suitable matrix factorization which allows the computation of $\{d_j\}$. Similarly to [12], we first show that the CG is an ideal candidate to generate the latter matrix factorization. However, it may reveal serious disadvantages on nonconvex problems. In this regard, $CG_①$ represents a natural generalization of the CG, and with some care allows to extend CG properties on indefinite problems. Then, the $CG_①$ is used to generate directions in eigenspaces of the Hessian matrix associated with negative eigenvalues and to provide a suitable matrix factorization, which allows to exploit the results in [14].

We also propose a numerical experience, where we assess the effectiveness of the negative curvature directions computed in the current paper. We prefer to skip a numerical comparison between our proposal and those in [3,5,10], the latter requiring an expensive matrix storage or the need for recomputing some quantities/vectors. This risks to make the comparison unfair, inasmuch as in [9] and here we prove that an inexpensive iterative computation of d_j is obtained, by storing at most two (four in [9]) working vectors.

To sum up, considering [9] as a reference paper with respect to our analysis, the main enhancements of the approach in the current paper can be summarized as follows:

– in [9] the computation of the negative curvature direction d_j, at iterate x_j, requires the storage of up to *four vectors*, while here we propose a method requiring the storage of only *two vectors*;
– the theory in [9] heavily relies on complicate matrix factorizations, due to the structure of the Planar-CG method therein adopted. Conversely, here the analysis through grossone only indirectly uses matrix factorizations provided by a Planar-CG method. Moreover, the Planar-CG method indirectly adopted here is definitely simpler that the one adopted in [9]. Hence, here the theoretical analysis to prove convergence of the sequence $\{x_j\}$ to stationary points, satisfying second-order conditions, is drastically simplified;
– the strategy adopted in [9] to compute the search directions is definitely more computationally expensive than the one proposed here;
– as regards numerical results, we do not claim that our proposal is always more efficient than the one in [9], depending on the problem in hand.

The paper is organized as follows: Sect. 2 reports some preliminaries on the use of negative curvature directions within truncated Newton methods. In Sect. 3, we give some basics on grossone, in order to motivate its use within the algorithm $CG_①$. Section 4 emphasizes the importance of certain matrix factorizations, in order to iteratively compute the final negative curvature direction. Section 5 stresses the importance of pairing $CG_①$ with the approach in [14]. Moreover, the last section explicitly yields our formula for determining the negative curvature direction. Finally, Sect. 6 contains

a numerical experience on our proposal, and Sect. 7 reports some conclusions and future research perspectives.

In this paper, we use standard notations for vectors and matrices. With $\| \cdot \|$, we indicate the Euclidean norm. $\lambda[A]$ is a general eigenvalue of matrix $A \in \mathbb{R}^{n \times n}$, and $A \succ 0 \ [A \succeq 0]$ indicates that A is positive definite [semidefinite]. $e_k \in \mathbb{R}^n$ represents the kth unit vector, while the symbol ① represents the numeral *grossone* (see also [13]).

2 Negative Curvature Directions in Truncated Newton Methods

Hereafter, we will use the following scheme

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

with $f \in C^2(\mathbb{R}^n)$, as a general reference for an unconstrained optimization problem. Moreover, the equation

$$\nabla^2 f(x_j) \, s = -\nabla f(x_j) \tag{2}$$

represents Newton's equation associated with problem (1).

The use of negative curvature directions in the framework of truncated Newton methods was introduced in the early papers [6,7], in order to define algorithms converging toward second-order critical points, namely stationary points where the Hessian matrix is positive semidefinite. Following the approach in [7], the sequence of negative curvature directions $\{d_j\}$ is expected to satisfy the conditions in the next assumption.

Assumption 2.1 Given problem (1), with $f \in C^2(\mathbb{R}^n)$, the nonascent directions in the sequence $\{d_j\}$ are bounded and satisfy the conditions

(a) $\nabla f(x_j)^T d_j \leq 0, \quad d_j^T H_j d_j \leq 0,$
(b) if $\lim_{j \to \infty} d_j^T \nabla^2 f(x_j) d_j = 0$ then $\lim_{j \to \infty} \min \left\{ 0, \ \lambda_{\min} \left[\nabla^2 f(x_j) \right] \right\} = 0,$

where $\lambda_{\min} \left[\nabla^2 f(x_j) \right]$ is the smallest eigenvalue of the Hessian matrix $\nabla^2 f(x_j)$.

The approach adopted in [7] may be generalized to some extent (see, for instance, the proposal in [5]), by suitably weakening conditions the directions $\{d_j\}$ are subject to.

Roughly speaking, the condition (*a*) in Assumption 2.1 implies that at any iterate x_j, the nonascent vector d_j must be a nonpositive curvature direction. Moreover, as in condition (*b*), when the quantity $d_j^T \nabla^2 f(x_j) d_j$ approaches zero, then the sequence $\{x_j\}$ is approaching a region of *convexity* for the function $f(x)$. Indeed, in such a case there will be no more chances to compute a negative curvature direction satisfying $d_j^T \nabla^2 f(x_j) d_j < 0$, so that eventually the condition $d_j^T \nabla^2 f(x_j) d_j \to 0$ must hold. Of course, on convex problems, negative curvature directions are not present, so that points provided by Newton–Krylov methods eventually satisfy also second-order stationarity conditions.

We recall that the main purpose of Newton–Krylov methods is to compute the (possibly) infinite sequence $\{x_j\}$, such that at least one of its subsequences is convergent to a stationary point of $f(x)$. In this regard, Assumption 2.1 does not imply a unique choice of d_j at iteration j. In fact, in order to fulfill (b) of Assumption 2.1, it suffices to compute d_j so that $d_j^T \nabla^2 f(x_j)d_j \leq v_j^T \nabla^2 f(x_j)v_j$, being v_j an eigenvector associated with the smallest eigenvalue of $\nabla^2 f(x_j)$. In addition, d_j becomes essential only eventually; i.e., far from a stationary point, it might be unnecessary to force convergence toward regions of convexity for $f(x)$. Nevertheless, using information associated with the negative curvature direction d_j, also when far from a solution point, may considerably enhance efficiency. The latter fact was evidenced, for instance, in [3,5,10] and intuitively follows from the next reasoning. Given the local quadratic expansion

$$q_j(d) = f(x_j) + \nabla f(x_j)^T d + \frac{1}{2}d^T \nabla^2 f(x_j)d,$$

at x_j, the directional derivative of $q_j(d)$

$$\nabla q_j(d)^T d = \nabla f(x_j)^T d + d^T \nabla^2 f(x_j)d,$$

along d, may strongly decrease when d is not only a descent vector, but also a negative curvature direction. This fact is explicitly used in [2,3,5,10] when a negative curvature direction d_j is adopted, at any iteration j, even if it is possibly not associated with an eigenvector z corresponding to the smallest negative eigenvalue of $\nabla^2 f(x_j)$.

Further developments on the computation of negative directions, to be used within truncated Newton methods, have been introduced in the already mentioned papers [3,5,9]. In particular, in [3,5] the direction d_j is obtained as by-product, when applying a Krylov-subspace method to solve Newton's equation (2). However, since the Krylov method may perform, at iteration j, a number k of steps considerably smaller than n, not all the eigenspaces associated with the Hessian matrix $\nabla^2 f(x_j)$ will be explored, vanishing the search of z. Hence, only an approximation of z may be available after k steps of the Krylov-based method.

In [9], the iterative computation of the directions $\{d_j\}$ is proposed. The innovative contribution in [9] consisted of explicitly providing the *iterative computation* of the sequence $\{d_j\}$, without requiring burdensome re-computing (as in [3]) or any expensive storage (as in [5]). In particular, the Krylov-based procedure adopted in [9] to compute d_j involves the use of the *Planar-CG method*, which represents an extension of the CG to nonconvex quadratic functions, where the Hessian matrix is possibly indefinite. The approach using this Planar-CG method surely proved to be effective, but it has a major disadvantage, requiring a fairly complex analysis which involves considering different and articulated subcases.

3 A Brief Introduction to the ①-Based Computational Methodology

The numeral ① called *grossone* has been introduced (see a recent survey [13]) as a basic element of a powerful numeral system, allowing one to express not only finite but also infinite and infinitesimal quantities. (Analogously, the numeral 1 is a basic

element allowing one to express finite quantities.) From the foundational point of view, grossone has been introduced as an infinite unit of measure equal to the number of elements of the set $\mathbb{N}$ of natural numbers. (Notice that the noncontradictoriness of the ①-based computational methodology has been studied in depth in [15–17].) From the practical point of view, this methodology has given rise both to a new supercomputer patented in several countries (see [18]) and called *Infinity Computer* and to a variety of applications starting from optimization (see [12,19–24]) and going through infinite series (see [13,25–28]), fractals and cellular automata (see [25,29–32]), hyperbolic geometry and percolation (see [33,34]), the first Hilbert problem and Turing machines (see [13,35,36]), infinite decision making processes and probability (see [13,37–39]), numerical differentiation and ordinary differential equations (see [40–43]), etc.

This methodology does not contradict traditional views on infinity and infinitesimals (Cantor, Leibnitz, Robinson, etc.) and proposes just another, more computationally oriented, way to deal with these objects. In particular, in order to avoid misunderstanding it should be stressed that there exist several differences (see [44] for a detailed discussion) that distinguish the numerical ①-based methodology from symbolically oriented nonstandard analysis of Robinson. Another important preliminary remark is that symbols traditionally used to work with infinities and infinitesimals (∞ introduced by Wallis, Cantor's ω, $\aleph_0$, $\aleph_1$, ..., etc.) are not used together with ①. Similarly, when the positional numeral system and the numeral 0 expressing zero had been introduced, symbols V, X and other symbols from the Roman numeral system had not been used in the positional numeral system.

The numeral ① allows one to construct different numerals involving infinite, finite, and infinitesimal parts and to execute numerical computations with all of them in a unique computational framework. As a result, it becomes possible to execute arithmetical operations with a variety of different infinities and infinitesimals. As a remarkable result, indeterminate forms such as $\infty - \infty$ or $\infty \cdot 0$ are not present when one works with numbers expressed in the ①-based numeral system. Traditionally existing kinds of divergences do not appear, as well. They are substituted by expressions that can contain also finite, infinite and infinitesimal parts.

In order to give some examples of arithmetical operations that can be executed in the ①-based numeral system, let us consider the following numbers: ① and $①^{16.3}$ (that are examples of infinities), along with $①^{-1}$ and $①^{-16.3}$ (that are examples of infinitesimals). Then, we can compute, for instance, the following expressions:

$$0 \cdot ① = ① \cdot 0 = 0, \quad ① - ① = 0, \quad \frac{①}{①} = 1, \quad ①^0 = 1, \quad 1^① = 1, \quad 0^① = 0,$$

$$0 \cdot ①^{-1} = ①^{-1} \cdot 0 = 0, \quad ①^{16.3} > ①^1 > 1 > ①^{-1} > ①^{-16.3} > 0,$$

$$①^{-1} - ①^{-1} = 0, \quad \frac{①^{-1}}{①^{-1}} = 1, \quad \frac{6.7 + ①^{-16.3}}{①^{-16.3}} = 6.7①^{16.3} + 1, \quad (①^{-1})^0 = 1,$$

$$① \cdot ①^{-1} = 1, \quad ① \cdot ①^{-16.3} = ①^{-15.3}, \quad \frac{4.2①^{16.3} + 29.1①}{①} = 4.2①^{15.3} + 29.1,$$

$$\frac{①^{16.3}}{①^{-16.3}} = ①^{32.6}, \quad (①^{16.3})^0 = 1, \quad ①^{16.3} \cdot ①^{-1} = ①^{15.3}, \quad ①^{16.3} \cdot ①^{-16.3} = 1.$$

$$(3)$$

Table 1 A practical implementation of the CG algorithm for the symmetric positive definite linear system $Ax = b$, with $A \in \mathbb{R}^{n \times n}$

The Conjugate Gradient (CG) method

Data: Set $k = 0$, $x_0 = 0$, $r_0 = b - Ax_0$, $\varepsilon > 0$.
If $\|r_0\| < \varepsilon$, then STOP. Else, set $p_0 = r_0$.

Step k: Compute $a_k = r_k^T p_k / p_k^T A p_k$, $\quad x_{k+1} = x_k + a_k p_k$, $\quad r_{k+1} = r_k - a_k A p_k$.
If $\|r_{k+1}\| < \varepsilon$, then STOP.
Else, set $\beta_k = -r_{k+1}^T A p_k / p_k^T A p_k = \|r_{k+1}\|^2 / \|r_k\|^2$,
$p_{k+1} = r_{k+1} + \beta_k p_k$, and $k = k + 1$.
Go to **Step k**.

In general, in the ①-based numeral system the simplest infinitesimal numbers are represented by numerals having only negative finite powers of ① (e.g., $40.17①^{-13.26} + 87.32①^{-25.7}$, see also examples above). The simplest infinite numbers are represented by numerals having at least one positive power of ①. Then, it can be seen in (3) that $①^0 = 1$; therefore, a finite number a can be represented in the new numeral system simply as $a①^0 = a$, where the numeral a itself can be written down by any convenient numeral system used to express finite numbers. These numbers are called *purely finite* because they do not contain infinitesimal parts. For instance, number 5 is purely finite and $5 + 3①^{-16.3}$ is finite but not purely finite, because it contains the infinitesimal part $3①^{-16.3}$. Notice that all infinitesimals are not equal to zero. In particular, $\frac{1}{①} > 0$ because it is a result of division of two positive numbers.

4 The Matrix Factorizations We Need

In the current and the next section, we describe how to use some Krylov-subspace methods, in order to gain advantage of a suitable factorization for the (possibly) indefinite Hessian matrix $\nabla^2 f(x_j)$. We strongly remark that we *never explicitly* compute here any Hessian decomposition, since our final achievements definitely rely on *implicit decompositions*, induced by Krylov-based methods.

As a general result, we highlight that computing negative curvature directions for $f(x)$ at x_j, which match the requirements in Assumption 2.1, may reduce to a simple task when suitable factorizations of $\nabla^2 f(x_j)$ are available. To give an intuition of the latter fact, suppose both the relations

$$M_j^T \nabla^2 f(x_j) M_j = C_j, \qquad C_j = Q_j B_j Q_j^T, \qquad (4)$$

are available at iterate x_j, being $M_j \in \mathbb{R}^{n \times k}$, where $C_j, Q_j, B_j \in \mathbb{R}^{k \times k}$ are non-singular. In this regard, the CG (Table 1) is an example of a Krylov-based method, satisfying the following properties:

– it provides the decompositions in (4) (see also (6)) when $\nabla^2 f(x_j) \succ 0$, with $j \geq 1$;

– the matrices M_j, C_j, Q_j, B_j have a special structure, inasmuch as (see also [45]): The columns of M_j are unit orthogonal vectors, C_j is tridiagonal, Q_j is unit lower bidiagonal, and B_j is diagonal.

Note that the Lanczos process, which represents another renowned Krylov-based method in the literature, iteratively provides the *left* decomposition in (4), with C_j tridiagonal, but not the *right* one. It is indeed necessary to couple the Lanczos process with a suitable factorization of C_j, in order to obtain usable negative curvature directions or solvers for Newton's equation (see, e.g., SYMMLQ/MINRES [46], SYMMBK [47]).

Now, given (4) suppose the vector $w \in \mathbb{R}^k$ is an eigenvector of B_j, associated with its *negative eigenvalue* λ, whose computation is 'relatively simple.' Moreover, suppose the vector $y \in \mathbb{R}^k$ is easily available, such that $Q_j^T y = w$. Then, by (4) the equalities/inequalities

$$
\begin{aligned}
(M_j y)^T \nabla^2 f(x_j)(M_j y) &= y^T \left[M_j^T \nabla^2 f(x_j) M_j \right] y \\
&= y^T C_j y \;=\; (Q_j^T y)^T B_j (Q_j^T y) \\
&= w^T B_j w \;=\; \lambda \|w\|^2 \;<\; 0
\end{aligned}
$$

immediately show that the direction $d_j = M_j y$ is of negative curvature for $f(x)$ at x_j. In particular, thanks to the chain of equalities above, if λ is the *smallest* negative eigenvalue of B_j, then $M_j y$ is also an eigenvector of $\nabla^2 f(x_j)$, associated with the *smallest eigenvalue* of $\nabla^2 f(x_j)$. The most renowned Krylov-subspace methods for symmetric linear systems (i.e., SYMMLQ, SYMMBK, CG, Planar-CG methods [48–50]) can all provide the factorizations (4) when applied to solve Newton's equation at the iterate x_j. Hence, generating a negative curvature which satisfies (*a*) in Assumption 2.1, may not be a difficult goal. However, fulfilling also (*b*) and the *boundedness* of the latter negative curvature direction, is a less trivial task. Indeed, the counterexample in Sect. 4 of [7] issues such a drawback, when a modified Cholesky factorization of the Hessian matrix is possibly adopted.

We strongly remark this point, since our main effort here is that of coupling a Krylov-subspace method with the novel tool in the literature given by *grossone*. In particular, we want to show that by the use of a subset of properties which hold for grossone, we can yield an implicit matrix factorization as in (4), fulfilling also (*b*) and the boundedness of the final negative curvature direction d_j in Assumption 2.1.

On this purpose, let us first state a general formal result for Krylov-subspace methods, which possibly summarizes the above considerations. The proof of the next lemma easily follows from Lemma 4.3 in [7] and Theorem 3.2 in [9].

Lemma 4.1 *Let problem* (1) *be given with $f \in C^2(\mathbb{R}^n)$, and consider an iterative method for solving* (1)*, which generates the sequence $\{x_j\}$. Let the level set $\mathcal{L}_0 = \{x \in \mathbb{R}^n : f(x) \le f(x_0)\}$ be compact, being any limit point $\bar{x}$ of $\{x_j\}$ a stationary point for* (1)*, with $\left| \lambda[\nabla^2 f(\bar{x})] \right| > \bar{\lambda} > 0$. Suppose n iterations of a Newton–Krylov method are performed to solve Newton's equation* (2) *at iterate x_j, for a given $j \ge 0$, so that*

the decompositions

$$R_j^T \nabla^2 f(x_j) R_j = T_j, \qquad T_j = L_j B_j L_j^T \tag{5}$$

are available. Moreover, suppose $R_j \in \mathbb{R}^{n \times n}$ is orthogonal, $T_j \in \mathbb{R}^{n \times n}$ has the same eigenvalues of $\nabla^2 f(x_j)$, with at least one negative eigenvalue, and $L_j, B_j \in \mathbb{R}^{n \times n}$ are nonsingular. Let z be the unit eigenvector corresponding to the smallest eigenvalue of B_j, and let $\bar{y} \in \mathbb{R}^n$ be the (bounded) solution of the linear system $L_j^T y = z$. Then, the vector $d_j = R_j \bar{y}$ is bounded and satisfies Assumption 2.1.

The vector d_j computed in Lemma 4.1 may be used to guarantee the satisfaction of Assumption 2.1, i.e., the sequence $\{d_j\}$ can guarantee convergence to second-order critical points. However, three main drawbacks of the approach in Lemma 4.1 are that

(α) the eigenvector z of B_j and the solution of the linear system $L_j^T y = z$ should be of *easy computation*;
(β) the corresponding vector $\bar{y}$ should be provably *bounded*;
(γ) at iterate j the Newton–Krylov method adopted to solve (2) possibly does not perform n iterations.

Observe that according to the requirements in Assumption 2.1, after a careful consideration, the issue at item (γ) is not really so relevant. Indeed, in any case (see also [9]) when $j \to \infty$ the convergence of the Newton–Krylov method imposes that it eventually performs n iterations [51]. On the other hand, in case at iterate x_j, for a finite j, $\nabla^2 f(x_j) \succeq 0$ or a vector $v \in \mathbb{R}^n$ such that $v^T \nabla^2 f(x_j) v < 0$ is unavailable, then the factorization (5) yet exists and we can simply set $d_j = 0$, which satisfies (a) in Assumption 2.1 along with the boundedness requirement.

Though the CG is not well posed when $\nabla^2 f(x_j) \nsucceq 0$, in [9] the authors reported that, *in case n CG steps are performed without stopping when solving Newton's equation*, the above items (α) and (β) can be relatively easily fulfilled, even in case $\nabla^2 f(x_j)$ is indefinite. In particular, these results are obtained exploiting the factorizations in Lemma 4.1, for which the CG specifically yields (Table 1)

$$R_j = \begin{pmatrix} \dfrac{r_1}{\|r_1\|} & \cdots & \dfrac{r_n}{\|r_n\|} \end{pmatrix},$$

$$B_j = \begin{pmatrix} \frac{1}{a_1} & & & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & & & \frac{1}{a_n} \end{pmatrix}, \quad L_j = \begin{pmatrix} 1 & & & & & \\ -\sqrt{\beta_1} & 1 & & & 0 & \\ & \ddots & \ddots & & & \\ & & \ddots & \ddots & & \\ & & & -\sqrt{\beta_{n-2}} & 1 & \\ 0 & & & & -\sqrt{\beta_{n-1}} & 1 \end{pmatrix}. \tag{6}$$

Thanks to the above expressions of R_j, B_j and L_j, in [9] the authors proved that after n steps the CG straightforwardly yields also the bounded negative curvature direction

$$d_j = \frac{p_m}{\|r_m\|}, \tag{7}$$

being $1 \leq m \leq n$ an index such that

$$\frac{1}{a_m} = \min_i \left\{ \frac{1}{a_i} \right\},$$

i.e.,

$$d_j = \frac{p_m}{\|r_m\|} \in \operatorname{argmin}_i \left\{ \left(\frac{p_i}{\|r_i\|} \right)^T A \left(\frac{p_i}{\|r_i\|} \right) \right\}.$$

Moreover, d_j in (7) satisfies Lemma 4.1, thanks to the fact that B_j is diagonal (i.e., its eigenvectors coincide with the canonical basis), L_j is unit lower bidiagonal (so that the solution of $L_j^T y = e_m$ is straightforwardly available by backtracking) and d_j is provably bounded.

Our goal is that of possibly replicating an analogous reasoning, with other Krylov-based methods for indefinite linear systems, following similar guidelines. In this regard, observe that both the tasks (α) and (β) might be hardly guaranteed only using, for instance, the instruments in [14], essentially because comparing with the CG, the structure of the matrices L_j and B_j generated by the Planar-CG method in [14] is more cumbersome. Nevertheless, in the next sections, starting from the structure of matrices L_j and B_j, as computed by the algorithm in [14], we will show how to use CG$_①$ in [12] in order to fulfill the hypotheses of Lemma 4.1.

5 Our Proposal: Preliminaries

To fill the gap outlined in the previous section, and recalling that in Lemma 4.1 we focus on the case where $j \to +\infty$, let us set for the sake of simplicity $A = \nabla^2 f(x_j)$, $b = -\nabla f(x_j)$. This allows us to drop the dependency on the subscript j. Consider the method CG$_①$ in [12] (which is also reported in Table 2, for the sake of completeness). Observe that the practical implementation of Step k in CG$_①$ currently allows the test $p_k^T A p_k \neq 0$ to be replaced by the inequality $|p_k^T A p_k| \geq \eta \|p_k\|^2$, with $\eta > 0$ small).

The CG$_①$ substantially coincides with the CG, as long as $p_k^T A p_k \neq 0$. Moreover, in case at Step k we have $p_k^T A p_k = 0$, from Section 5.1 of [12] the CG$_①$ generates both the vectors r_{k+1} and p_{k+1}, such that they depend on ①. Furthermore, we have (after a simple computation, and using the standard Landau–Lifsits notation $O(\cdot)$)

$$\alpha_k \doteq \frac{\|r_k\|^2}{s①}$$

Table 2 The CG$_{①}$ algorithm for solving the symmetric indefinite linear system $Ax = b$, $A \in \mathbb{R}^{n \times n}$. In a practical implementation of Step k of CG$_{①}$, the test $p_k^T A p_k \neq 0$ may be replaced by the inequality $|p_k^T A p_k| \geq \eta \|p_k\|^2$, with $\eta > 0$ small

---

CG$_{①}$: Conjugate Gradient method coupled with grossone ①

Data: Set $k = 0$, $x_0 = 0$, $r_0 = b - Ax_0$, $\varepsilon > 0$, $s = O(①^{-2})$.
           If $\|r_0\| < \varepsilon$, then STOP. Else, set $p_0 = r_0$.

Step k: If $\|p_k\|$ is finite (bounded) and $p_k^T A p_k \neq 0$ then compute
              $\alpha_k = r_k^T p_k / p_k^T A p_k$, $x_{k+1} = x_k + \alpha_k p_k$, $r_{k+1} = r_k - \alpha_k A p_k$.
            If $\|r_{k+1}\| < \varepsilon$, then STOP.
            Elseif $\|p_k\|$ is finite (bounded) set $p_k^T A p_k = s①$ and compute
              $r_{k+1} = r_k - \|r_k\|^2 / (s①) A p_k$.
            Else compute $\alpha_k = r_k^T p_k / p_k^T A p_k$,
              $x_{k+1} = x_k + \alpha_k p_k$, $r_{k+1} = r_k - \alpha_k A p_k$.
              If the finite part of r_{k+1} satisfies $\|r_{k+1}\| < \varepsilon$, then STOP.
            Endif
            Set $\beta_k = -r_{k+1}^T A p_k / p_k^T A p_k = \|r_{k+1}\|^2 / \|r_k\|^2$, and
            $p_{k+1} = r_{k+1} + \beta_k p_k$, $k = k + 1$.
            Go to **Step k**.

$$
\alpha_{k+1} \doteq \frac{\|r_{k+1}\|^2}{p_{k+1}^T A p_{k+1}}
$$

$$
= \frac{-\|r_k\|^2 + \frac{\|r_k\|^4 \|A p_k\|^2}{s^2 ①^2}}{\frac{\|r_k\|^4}{s^2 ①^2} (A p_k)^T A (A p_k) - \frac{\|r_k\|^4 \|A p_k\|^4}{s^3 ①^3} + O(①)}
$$

$$
= \frac{-\|r_k\|^2 s^2 ①^2 + \|r_k\|^4 \|A p_k\|^2}{\|r_k\|^4 (A p_k)^T A (A p_k) - \frac{\|r_k\|^4 \|A p_k\|^4}{s①} + s^2 ①^2 O(①)}
$$

$$
= \frac{-\|r_k\|^2 s^3 ①^3 + \|r_k\|^4 \|A p_k\|^2 s①}{\|r_k\|^4 (A p_k)^T A (A p_k) s① - \|r_k\|^4 \|A p_k\|^4 + s^3 ①^3 O(①)}. \tag{8}
$$

Recalling that by definition $s = O(①^{-2})$, i.e., $s① = O(①^{-1})$, then neglecting in α_{k+1} the terms containing negative powers of ① (corresponding indeed to infinitesimals with respect to the value $①^0 = 1$—see Sect. 3) we have

$$
\begin{aligned}
\frac{1}{\alpha_k} &= \frac{s①}{\|r_k\|^2} \\
\frac{1}{\alpha_{k+1}} &= -\frac{\|A p_k\|^2}{s①}.
\end{aligned} \tag{9}
$$

This immediately implies that

$$
\alpha_k \cdot \alpha_{k+1} < 0. \tag{10}
$$

Note that by using CG$_{①}$, similarly to the CG to solve (2), we can recover the structure of B_j and L_j in (6), so that (7) formally applies. However, since $s①$ is infinitesimal in (9), after some computation we have (see also Sect. 5.1 of [12])

$$r_{k+1} = r_k - \frac{\|r_k\|^2}{s①} A p_k$$

$$\|r_{k+1}\|^2 = \frac{\|r_k\|^4 \|A p_k\|^2}{s^2①^2} - \|r_k\|^2$$

$$p_{k+1} = -\beta_{k-1} p_{k-1} - \frac{\|r_k\|^2}{s①} A p_k + \frac{\|r_k\|^2 \|A p_k\|^2}{s^2①^2} p_k.$$

Then, in case $m = k + 1$ in (7), to compute the direction d_j we would have

$$d_j = \frac{p_m}{\|r_m\|} = \frac{p_{k+1}}{\|r_{k+1}\|} = \frac{\|A p_k\|}{|s|①} p_k + O(①^0), \tag{11}$$

which implies that $\|d_j\|$ is *not bounded* (being $s① = O(①^{-1})$) and Lemma 4.1 cannot be fulfilled. This consideration should not be surprising. Indeed, it basically summarizes the fact that similarly to the CG, CG$_①$ is unable to provide the diagonal matrix (6) with finite entries, in case the tridiagonal matrix T_j in (5) is indefinite.

Nevertheless, to overcome the latter limitation, now we show how to properly couple CG$_①$ with the Planar-CG method in [14], in order to obtain a suitable *bounded* negative curvature direction which fulfills Lemma 4.1.

5.1 Coupling CG$_①$ with the Algorithm in [14]

Following the taxonomy in Sect. 4, assume without loss of generality that the Krylov-subspace method detailed in [14] is applied to solve Newton's equation

$$Au = b, \tag{12}$$

and n steps are performed.[1] Again this allows us to drop the dependency on the subscript j. After some computation, the following matrices are generated (see also [52])

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ 0 & L_{32} & L_{33} \end{pmatrix}, \qquad B = \begin{pmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{pmatrix},$$

where

$$L_{11} = \begin{pmatrix} 1 & & \\ -\sqrt{\beta_1} & \ddots & \\ & \ddots & 1 \end{pmatrix}, \quad L_{21} = \begin{pmatrix} 0 & \cdots & -\sqrt{\beta_{k-1}} \\ 0 & \cdots & 0 \end{pmatrix}, \quad L_{22} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{13}$$

---

[1] Note that with reference to the comments in Sect. 4, since n is large, any Krylov-based method used to solve Newton's equation $\nabla^2 f(x_j)u = -\nabla f(x_j)$ is usually expected to perform n steps only eventually (i.e., when $j \to +\infty$), being this computation typically expensive. Here, as in Lemma 4.1, the assumption of performing n steps with the procedure in [14] is uniquely motivated to preserve simplicity. Nevertheless, a few additional trivial modifications are necessary in case only $k < n$ steps are performed, following the guidelines in [52] and [5].

$$L_{32} = \begin{pmatrix} -\sqrt{\beta_k \beta_{k+1}} & 0 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix}, \quad L_{33} = \begin{pmatrix} 1 & & \\ -\sqrt{\beta_{k+2}} & \ddots & \\ & \ddots & 1 \\ & & -\sqrt{\beta_{n-1}} & 1 \end{pmatrix}, \tag{14}$$

and

$$B_{11} = \begin{pmatrix} \frac{1}{\alpha_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\alpha_{k-1}} \end{pmatrix}, \quad B_{22} = \begin{pmatrix} 0 & \sqrt{\beta_k} \\ \sqrt{\beta_k} & e_{k+1} \end{pmatrix}, \quad B_{33} = \begin{pmatrix} \frac{1}{\alpha_{k+2}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\alpha_n} \end{pmatrix}, \tag{15}$$

such that

$$AR = RT, \qquad T = LBL^T, \tag{16}$$

where (see also [52]) the matrix $R \in \mathbb{R}^{n \times n}$ has n unit orthogonal columns and is given by

$$R = \begin{pmatrix} \dfrac{r_1}{\|r_1\|} & \cdots & \dfrac{r_n}{\|r_n\|} \end{pmatrix}, \tag{17}$$

with $r_{k+1} = Ap_k$, while $T \in \mathbb{R}^{n \times n}$ is tridiagonal. Moreover, $\{\alpha_i\}$, $\{\beta_i\}$, e_{k+1} are suitable scalars (being $e_{k+1} = (Ap_k)^T A(Ap_k)/\|Ap_k\|^2$). We also recall that $\beta_i > 0$, for any $i \geq 1$. Finally, in the above matrices L and B we assume (for the sake of simplicity) that the Krylov-based method in [14] has performed all CG steps, with the exception of only one planar iteration (namely the kth iteration—see [14] and [48]), corresponding to have $p_k^T A p_k \approx 0$.

Then, our novel approach proposes to introduce the numeral *grossone*, as in [13,22–24], and follows some guidelines from [12], in order to exploit a suitable matrix factorization from (16), such that Lemma 4.1 is fulfilled. In this regard, consider matrix B in (16) and the next technical result.

Lemma 5.1 *Consider the matrix B in (16), and let $\beta_k > \bar{\sigma} > 0$, for any $k \geq 0$. Then, the 2×2 submatrix*

$$\begin{pmatrix} 0 & \sqrt{\beta_k} \\ \sqrt{\beta_k} & e_{k+1} \end{pmatrix}$$

has the Jordan factorization

$$\begin{pmatrix} 0 & \sqrt{\beta_k} \\ \sqrt{\beta_k} & e_{k+1} \end{pmatrix} = V_k \Lambda_k V_k^T, \tag{18}$$

with

$$V_k = \begin{pmatrix} \dfrac{\sqrt{\beta_k}}{\sqrt{\beta_k+\lambda_k^2}} & \dfrac{\sqrt{\beta_k}}{\sqrt{\beta_k+\lambda_{k+1}^2}} \\[3mm] \dfrac{\lambda_k}{\sqrt{\beta_k+\lambda_k^2}} & \dfrac{\lambda_{k+1}}{\sqrt{\beta_k+\lambda_{k+1}^2}} \end{pmatrix}, \tag{19}$$

where $\Lambda_k = diag\{\lambda_k, \lambda_{k+1}\}$.

Moreover, $\lambda_k \cdot \lambda_{k+1} < 0$, *with* $\lambda_k, \ \lambda_{k+1} \in \left\{ \dfrac{e_{k+1} \pm \sqrt{e_{k+1}^2 + 4\beta_k}}{2} \right\}$, *along with* $\lambda_k > 0$ *and* $\lambda_{k+1} < 0$. *Finally, if* $\|r_i\| \geq \varepsilon$, *for any* $i \leq k$, *then*

$$\sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} \leq \max_i \{|\lambda_i(A)|\}^{1/2} + \sqrt{2}\left(\frac{\|Ap_k\|}{\varepsilon}\right)^{1/2} < +\infty, \tag{20}$$

$$\left| \frac{\lambda_k}{\lambda_{k+1} - \lambda_k} \right| \leq 1. \tag{21}$$

Proof The first part of the proof follows after a short computation and observing that

$$\det \begin{pmatrix} 0 & \sqrt{\beta_k} \\ \sqrt{\beta_k} & e_{k+1} \end{pmatrix} = -\beta_k < 0. \tag{22}$$

As regards (20), since

$$\lambda_{k+1} = \frac{e_{k+1} - \sqrt{e_{k+1}^2 + 4\beta_k}}{2},$$

we have

$$\sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} = \sqrt{\frac{\beta_k + \frac{e_{k+1}^2}{4} + \frac{e_{k+1}^2 + 4\beta_k}{4} - \frac{e_{k+1}\sqrt{e_{k+1}^2 + 4\beta_k}}{2}}{\frac{\sqrt{e_{k+1}^2 + 4\beta_k} - e_{k+1}}{2}}}$$

$$= \sqrt{\frac{2\beta_k + e_{k+1}^2 + 2\beta_k - e_{k+1}\sqrt{e_{k+1}^2 + 4\beta_k}}{\sqrt{e_{k+1}^2 + 4\beta_k} - e_{k+1}}}$$

$$= \sqrt{\frac{-e_{k+1}\sqrt{e_{k+1}^2 + 4\beta_k} + e_{k+1}^2 + 4\beta_k}{\sqrt{e_{k+1}^2 + 4\beta_k} - e_{k+1}}}$$

$$= \sqrt{\frac{-e_{k+1}\left[(e_{k+1}^2 + 4\beta_k) - e_{k+1}^2\right] + 4\beta_k\left[\sqrt{e_{k+1}^2 + 4\beta_k} + e_{k+1}\right]}{(e_{k+1}^2 + 4\beta_k) - e_{k+1}^2}}$$

$$= \sqrt{\frac{4\beta_k\sqrt{e_{k+1}^2 + 4\beta_k}}{4\beta_k}} = (e_{k+1}^2 + 4\beta_k)^{1/4}$$

$$\leq |e_{k+1}|^{1/2} + \sqrt{2}\beta_k^{1/4},$$

and since $\beta_k = \|Ap_k\|^2/\|r_k\|^2$, with $\|r_k\| \geq \varepsilon$, the relation

$$e_{k+1} = (Ap_k)^T A(Ap_k)/\|Ap_k\|^2$$

yields the result.

Finally, as regards (21) note that

$$\left|\frac{\lambda_k}{\lambda_{k+1} - \lambda_k}\right| = \left|\frac{e_{k+1} + \sqrt{e_{k+1}^2 + 4\beta_k}}{-2\sqrt{e_{k+1}^2 + 4\beta_k}}\right|$$

$$= \left|-\frac{1}{2}\left(\frac{e_{k+1}}{\sqrt{e_{k+1}^2 + 4\beta_k}} + 1\right)\right| \leq \frac{1}{2}(1+1) = 1.$$

$\square$

Then, replacing the factorization (18) into the expression of B in (16), we obtain the equivalent factorization $T = LBL^T = \bar{L}\bar{B}\bar{L}^T$, where

$$\bar{L} = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & V_k & 0 \\ 0 & \bar{L}_{32} & L_{33} \end{pmatrix}, \qquad \bar{B} = \begin{pmatrix} B_{11} & 0 & 0 \\ 0 & \Lambda_k & 0 \\ 0 & 0 & B_{33} \end{pmatrix}, \tag{23}$$

where L_{11}, L_{21} are defined in (13), L_{33} in (14), B_{11}, B_{33} in (15) and

$$\bar{L}_{32} = \begin{pmatrix} (-\sqrt{\beta_k\beta_{k+1}} \quad 0) \cdot V_k \\ \vdots \\ 0 \end{pmatrix}.$$

We remark that unlike the matrix B, now $\bar{B}$ is a diagonal matrix, though $\bar{L}$ has now a slightly more complex structure than the matrix L. Note also that after an easy computation, we have in $\bar{L}$

Table 3 Correspondence between quantities/vectors computed by the algorithm in [14] and the algorithm $CG_①$ in [12]

Algorithm in [14]		$CG_①$ in [12]	
r_i,	$i = 1, \ldots, k$	r_i,	$i = 1, \ldots, k$
r_{k+1}		Ap_k	
r_i,	$i \geq k + 2$	r_i,	$i \geq k + 2$ (neglecting the terms with $s①$)
p_i,	$i = 1, \ldots, k$	p_i,	$i = 1, \ldots, k$
p_{k+1}		$\dfrac{Ap_k}{\|Ap_k\|}$	
p_i,	$i \geq k + 2$	p_i,	$i \geq k + 2$ (neglecting the terms with $s①$)
α_i,	$i = 1, \ldots, k$	α_i,	$i = 1, \ldots, k$
α_i,	$i \geq k + 2$	α_i,	$i \geq k + 2$ (neglecting the terms with $s①$)
β_i,	$i = 1, \ldots, k - 1$	β_i,	$i = 1, \ldots, k - 1$
β_k		$\dfrac{\|Ap_k\|^2}{\|r_k\|^2}$	
β_i,	$i \geq k + 1$	β_i,	$i \geq k + 1$ (neglecting the terms with $s①$)

$$\left(-\sqrt{\beta_k \beta_{k+1}} \quad 0 \right) \cdot V_k = -\beta_k \sqrt{\beta_{k+1}} \left(\frac{1}{\sqrt{\beta_k + \lambda_k^2}} \quad \frac{1}{\sqrt{\beta_k + \lambda_{k+1}^2}} \right), \qquad (24)$$

where (see [14])

$$\beta_k = \frac{\|r_{k+1}\|^2}{\|r_k\|^2} = \frac{\|Ap_k\|^2}{\|r_k\|^2}, \qquad \beta_{k+1} = \frac{\|r_{k+2}\|^2}{\|r_{k+1}\|^2} = \frac{\|r_{k+2}\|^2}{\|Ap_k\|^2}.$$

Now, let us consider again the algorithm $CG_①$ in [12], and assume that at Steps k and $k + 1$ it generated the coefficients α_k and α_{k+1} in (9), when solving the linear system (12), being $p_k^T Ap_k \approx 0$ at Step k. In [12], we have already detailed the one-to-one relationship between the quantities generated by the algorithms in [14] and in Table 2, showing how $CG_①$ can be considered, to large extent, an extension of the CG to the indefinite case. Table 3 specifically reports this relationship, showing how it can be possible to compute all the quantities in (23) using $CG_①$, in place of the algorithm in [14]. Thus, similarly to the result obtained in (6), applying the CG, after n steps of $CG_①$ we want to define an implicit matrix factorization for A as in (16), where now the 2×2 matrix on the left-hand side of (18) is suitably replaced by the matrix $diag\{1/\alpha_k, 1/\alpha_{k+1}\}$. Now we establish a full correspondence between the matrix Λ_k in (18) and (23), obtained by the algorithm in [14], and the matrix $diag\{1/\alpha_k, 1/\alpha_{k+1}\}$ from [12]. Since both $\alpha_k s① \neq 0$ and $\alpha_{k+1}/(s①) \neq 0$, and by Lemma 5.1 $\lambda_k > 0$ with $\lambda_{k+1} < 0$, we can always find the 2×2 (diagonal) positive definite matrix C_k such that

$$\begin{pmatrix} \frac{1}{\alpha_k s \textcircled{1}} & 0 \\ 0 & \frac{s \textcircled{1}}{\alpha_{k+1}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\|r_k\|^2} & 0 \\ 0 & -\|Ap_k\|^2 \end{pmatrix} = C_k \Lambda_k C_k, \tag{25}$$

where Λ_k is defined in Lemma 5.1 and

$$C_k = \begin{pmatrix} \frac{1}{\sqrt{\lambda_k \alpha_k s \textcircled{1}}} & 0 \\ 0 & \sqrt{\frac{s \textcircled{1}}{\lambda_{k+1} \alpha_{k+1}}} \end{pmatrix}. \tag{26}$$

In practice, using CG$_\textcircled{1}$ we would like to rearrange the matrices $\bar{L}$ and $\bar{B}$ in (23), obtained by applying the algorithm in [14], so that the equalities $T = LBL^T = \bar{L}\bar{B}\bar{L}^T$ hold and the block Λ_k in $\bar{B}$ is suitably replaced by the left side of (25). Note that the diagonal matrix on the left side of (25) is scaled with respect to the matrix $diag\{1/\alpha_k, 1/\alpha_{k+1}\}$, by using terms containing $\textcircled{1}$. Moreover, it is worth mentioning that by (8) and (9), both the diagonal entries of the matrix on the left side of (25) are finite and not infinitesimal.

The rationale behind this choice is suggested by (11) and Lemma 4.1, where the easy computation of the vectors z and $\bar{y}$ is sought. Indeed, we shortly show that the scaling in (25) both allows to easily find the final negative curvature direction d_j in Lemma 4.1, and ensures that for any j the norm $\|d_j\|$ is suitably bounded. This finally implies that applying CG$_\textcircled{1}$ and exploiting Table 3, we can fulfill Assumption 2.1 without recurring first to the algorithm in [14].

Now, from (26) and (9) we obtain

$$C_k = \begin{pmatrix} \frac{1}{\|r_k\|\sqrt{\lambda_k}} & 0 \\ 0 & \frac{\|Ap_k\|}{\sqrt{-\lambda_{k+1}}} \end{pmatrix}, \tag{27}$$

showing that, apart from infinitesimals we ignored when writing (9), the diagonal entries of C_k are independent of $\textcircled{1}$. Finally, by (25) and considering the matrix Λ_k in Lemma 5.1, we obtain

$$V_k C_k^{-1} \begin{pmatrix} \frac{1}{\alpha_k s \textcircled{1}} & 0 \\ 0 & \frac{s \textcircled{1}}{\alpha_{k+1}} \end{pmatrix} C_k^{-1} V_k = V_k \Lambda_k V_k^T.$$

This also implies that we can now equivalently modify the nonsingular matrices in (23) as

$$
\hat{L} = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & V_k C_k^{-1} & 0 \\ 0 & \hat{L}_{32} & L_{33} \end{pmatrix}, \qquad \hat{D} = \begin{pmatrix} B_{11} & 0 & 0 \\ 0 & \hat{B}_{22} & 0 \\ 0 & 0 & B_{33} \end{pmatrix}, \qquad (28)
$$

where L_{11} and L_{21} are defined in (13), L_{33} in (14), B_{11}, B_{33} in (15) and

$$
\hat{L}_{32} = \begin{pmatrix} \left(-\sqrt{\beta_k \beta_{k+1}} \quad 0\right) \cdot V_k C_k^{-1} \\ \vdots \\ 0 \end{pmatrix}, \qquad \hat{B}_{22} = \begin{pmatrix} \frac{1}{\alpha_k s \textcircled{1}} & 0 \\ 0 & \frac{s \textcircled{1}}{\alpha_{k+1}} \end{pmatrix},
$$

so that in Lemma 4.1 we have for matrix T_j the expression

$$
T_j = L B L^T = \bar{L} \bar{B} \bar{L}^T = \hat{L} \hat{D} \hat{L}^T.
$$

We strongly remark that using $CG_{\textcircled{1}}$ and relation (25), we have simplified the expression of $\bar{D}$, replacing it with $\hat{D}$. This is obtained at the cost of a slight modification of matrix $\bar{L}$ into $\hat{L}$: We shortly prove that this arrangement can easily allow the computation of a bounded negative curvature direction d_j at x_j. Once more we urge to remark that the computation of $\hat{L}$ and $\hat{D}$ can be completely carried on replacing the algorithm in [14] with $CG_{\textcircled{1}}$, as the equivalence/correspondence in Table 3 reveals. (We highlight indeed that the iterate x_{k+2} in [14] and the iterate y_{k+2} in [12] coincide, when neglecting the infinitesimal terms containing $s\textcircled{1}$.) The next lemma proves that $\hat{L}$ in (28) is nonsingular under the assumptions in Lemma 4.1.

Lemma 5.2 *Let the assumptions in Lemma 4.1 hold, with $T_j = \hat{L} \hat{D} \hat{L}^T$ and $\hat{L}$, $\hat{D}$ defined in (28). Then, we have*

$$
V_k C_k^{-1} = \begin{pmatrix} \frac{\|r_k\| \sqrt{\beta_k \lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} & \frac{\sqrt{-\beta_k \lambda_{k+1}}}{\|A p_k\| \sqrt{\beta_k + \lambda_{k+1}^2}} \\ \frac{\|r_k\| \lambda_k \sqrt{\lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} & \frac{\lambda_{k+1} \sqrt{-\lambda_{k+1}}}{\|A p_k\| \sqrt{\beta_k + \lambda_{k+1}^2}} \end{pmatrix}
$$

$$
\left(-\sqrt{\beta_k \beta_{k+1}} \quad 0\right) \cdot V_k C_k^{-1} = \left(-\frac{\|r_{k+2}\| \|A p_k\|}{\|r_k\|} \sqrt{\frac{\lambda_k}{\beta_k + \lambda_k^2}} \quad -\frac{\|r_{k+2}\|}{\|r_k\|^2} \sqrt{-\frac{\lambda_{k+1}}{\beta_k + \lambda_{k+1}^2}}\right),
$$

along with $|\det(\hat{L})| = 1$.

Proof The first two relations follow immediately from (27), Table 3 and recalling that C_k is nonsingular. Moreover, since $\beta_k = \|Ap_k\|^2/\|r_k\|^2$, note that in (28) we have

$$
\begin{aligned}
\det(V_k C^{-1}) &= \frac{\|r_k\|\lambda_{k+1}\sqrt{-\beta_k\lambda_k\lambda_{k+1}}}{\|Ap_k\|\sqrt{\beta_k+\lambda_k^2}\sqrt{\beta_k+\lambda_{k+1}^2}} - \frac{\|r_k\|\lambda_k\sqrt{-\beta_k\lambda_k\lambda_{k+1}}}{\|Ap_k\|\sqrt{\beta_k+\lambda_k^2}\sqrt{\beta_k+\lambda_{k+1}^2}} \\
&= \frac{1}{\sqrt{\beta_k}}\sqrt{\frac{-\beta_k\lambda_k\lambda_{k+1}}{(\beta_k+\lambda_k^2)(\beta_k+\lambda_{k+1}^2)}}(\lambda_{k+1}-\lambda_k) \\
&= \sqrt{\frac{-\lambda_k\lambda_{k+1}}{(\beta_k+\lambda_k^2)(\beta_k+\lambda_{k+1}^2)}}\left[-\sqrt{e_{k+1}^2+4\beta_k}\right] \\
&= -\frac{\sqrt{\beta_k}\sqrt{e_{k+1}^2+4\beta_k}}{\sqrt{(\beta_k+\lambda_k^2)(\beta_k+\lambda_{k+1}^2)}} \\
&= -\frac{\sqrt{\beta_k}\sqrt{e_{k+1}^2+4\beta_k}}{\sqrt{(-\lambda_k\lambda_{k+1}+\lambda_k^2)(-\lambda_k\lambda_{k+1}+\lambda_{k+1}^2)}} \\
&= -\frac{\sqrt{\beta_k}\sqrt{e_{k+1}^2+4\beta_k}}{\sqrt{-\lambda_k\lambda_{k+1}(\lambda_k-\lambda_{k+1})^2)}} = -\frac{\sqrt{e_{k+1}^2+4\beta_k}}{|\lambda_k-\lambda_{k+1}|} \\
&= -\frac{\sqrt{e_{k+1}^2+4\beta_k}}{\sqrt{e_{k+1}^2+4\beta_k}} = -1.
\end{aligned}
$$

Therefore, $|\det(\hat{L})| = 1$.

Now we are ready to compute at iterate x_j the negative curvature direction d_j which complies with Assumption 2.1, exploiting the decomposition $T_j = \hat{L}\hat{D}\hat{L}^T$ from Lemma 4.1. □

Proposition 5.1 *Suppose n iterations of CG$_①$ algorithm are performed to solve Newton's equation (2), at iterate x_j, so that the decompositions*

$$
R^T \nabla^2 f(x_j) R = T, \qquad T = \hat{L}\hat{D}\hat{L}^T
$$

exist, where R is defined in (17), and $\hat{L}$ along with $\hat{D}$ is defined in (28). In the hypotheses of Lemma 4.1, let z be the unit eigenvector corresponding to the (negative) smallest eigenvalue of $\hat{D}$ and let $\hat{y}$ be the solution of the linear system $\hat{L}^T y = z$. Then, the vector $d_j = R\hat{y}$ is bounded and satisfies Assumption 2.1. In addition, the computation of d_j requires the storage of at most two n-real vectors.

Proof First observe that by [53], also in case at the iterate x_j the Hessian matrix $\nabla^2 f(x_j)$ is indefinite, there exists at most one step k, with $0 \le k \le n$, such that in CG$_①$ we might have $p_k^T \nabla^2 f(x_j) p_k = 0$. Thus, similarly to the rest of the paper, without loss

of generality in this proof we assume that possibly the equality $p_k^T \nabla^2 f(x_j) p_k = 0$ only holds at step k. Moreover, the matrix $\hat{D}$ is diagonal, which implies that the unit vector associated with its ith eigenvalue $\mu_i(\hat{D})$ is given by e_i.

To fulfill Assumption 2.1, we first need to compute the vector $\hat{y}$ in Lemma 4.1, i.e., we have to solve the linear system

$$\hat{L}^T y = z, \tag{29}$$

being $z \in \mathbb{R}^n$ the unit eigenvector associated with the (negative) smallest eigenvalue of $\hat{D}$. To this purpose, by Lemma 5.2 the vector $\hat{y}$ exists and is bounded. Now, we distinguish among the next four subcases, where we use the notation $\hat{k} \in \arg\min_i\{\mu_i(\hat{D})\}$, i.e., $\hat{k}$ is an index corresponding to the smallest eigenvalue $\mu_{\hat{k}}(\hat{D})$ of $\hat{D}$.

(I) In this subcase, we assume $\hat{k} \notin \{k, k+1\}$ along with $\hat{k} < k$. In particular, since $\hat{D}$ is diagonal, then (29) reduces to $\hat{L}^T y = e_{\hat{k}}$, i.e., by Lemma 5.2 and Table 3

$$-\sqrt{\beta_1} \cdot y_2 + y_1 = 0$$
$$\vdots$$
$$-\sqrt{\beta_{\hat{k}-1}} \cdot y_{\hat{k}} + y_{\hat{k}-1} = 0$$
$$-\sqrt{\beta_{\hat{k}}} \cdot y_{\hat{k}+1} + y_{\hat{k}} = 1$$
$$-\sqrt{\beta_{\hat{k}+1}} \cdot y_{\hat{k}+2} + y_{\hat{k}+1} = 0$$
$$\vdots$$
$$-\sqrt{\beta_{k-1}} \cdot y_k + y_{k-1} = 0$$
$$\frac{\|r_k\|\sqrt{\beta_k \lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} \cdot y_k + \frac{\|r_k\|\lambda_k\sqrt{\lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} \cdot y_{k+1}$$
$$- \frac{\|Ap_k\|\|r_{k+2}\|}{\|r_k\|}\sqrt{\frac{\lambda_k}{\beta_k + \lambda_k^2}} \cdot y_{k+2} = 0$$
$$\frac{\sqrt{-\lambda_{k+1}\beta_k}}{\|Ap_k\|\sqrt{\beta_k + \lambda_{k+1}^2}} \cdot y_k + \frac{\lambda_{k+1}\sqrt{-\lambda_{k+1}}}{\|Ap_k\|\sqrt{\beta_k + \lambda_{k+1}^2}} \cdot y_{k+1}$$
$$- \frac{\|r_{k+2}\|}{\|r_k\|^2}\sqrt{-\frac{\lambda_{k+1}}{\beta_k + \lambda_{k+1}^2}} \cdot y_{k+2} = 0$$
$$-\sqrt{\beta_{k+2}} \cdot y_{k+3} + y_{k+2} = 0$$
$$\vdots$$
$$-\sqrt{\beta_{n-1}} \cdot y_n + y_{n-1} = 0$$
$$y_n = 0,$$

whose solution $\hat{y} \in \mathbb{R}^n$ can be explicitly computed recalling that, as in Table 3, $r_{k+1} = Ap_k$, $\beta_i = \|r_{i+1}\|^2/\|r_i\|^2$ and backtracking from the value of $\hat{y}_n$ to $\hat{y}_1$, we have

$$\hat{y}_n = \cdots = \hat{y}_{k+1} = 0; \qquad \hat{y}_{\hat{k}} = 1; \qquad \hat{y}_i = \frac{\|r_{\hat{k}}\|}{\|r_i\|}, \quad i = \hat{k}-1, \ldots, 1.$$

Finally, as in Lemma 4.1 and recalling that for the algorithm CG① we have $p_i = r_i + \beta_{i-1}p_{i-1}$, for any $i \geq 1$, the corresponding negative curvature direction

d_j is given by

$$d_j = R_j \hat{y} = \|r_{\hat{k}}\| \sum_{i=1}^{\hat{k}} \frac{r_i}{\|r_i\|^2} = \frac{p_{\hat{k}}}{\|r_{\hat{k}}\|},$$

which exactly coincides with the proposal in [9], when $\hat{k} \notin \{k, k+1\}$ along with $\hat{k} < k$. Finally, it is easily seen that by the conditions $\|r_i\| \geq \varepsilon$, from algorithm $CG_{\textcircled{1}}$, the quantity $\|d_j\|$ is bounded and the computation of d_j simply requires the *storage of the unique vector* $p_{\hat{k}}/\|r_{\hat{k}}\|$.

(II) In this subcase, we assume $\hat{k} \notin \{k, k+1\}$ along with $\hat{k} > k+1$. Since again $\hat{D}$ is diagonal, then (29) reduces to

$$-\sqrt{\beta_1} \cdot y_2 + y_1 = 0$$
$$\vdots$$
$$-\sqrt{\beta_{k-1}} \cdot y_k + y_{k-1} = 0$$
$$\frac{\|r_k\|\sqrt{\beta_k \lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} \cdot y_k + \frac{\|r_k\| \lambda_k \sqrt{\lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} \cdot y_{k+1}$$
$$-\frac{\|Ap_k\| \|r_{k+2}\|}{\|r_k\|} \sqrt{\frac{\lambda_k}{\beta_k + \lambda_k^2}} \cdot y_{k+2} = 0$$
$$\frac{\sqrt{-\lambda_{k+1} \beta_k}}{\|Ap_k\| \sqrt{\beta_k + \lambda_{k+1}^2}} \cdot y_k + \frac{\lambda_{k+1} \sqrt{-\lambda_{k+1}}}{\|Ap_k\| \sqrt{\beta_k + \lambda_{k+1}^2}} \cdot y_{k+1}$$
$$-\frac{\|r_{k+2}\|}{\|r_k\|^2} \sqrt{-\frac{\lambda_{k+1}}{\beta_k + \lambda_{k+1}^2}} \cdot y_{k+2} = 0$$
$$-\sqrt{\beta_{k+2}} \cdot y_{k+3} + y_{k+2} = 0$$
$$\vdots$$
$$-\sqrt{\beta_{\hat{k}-1}} \cdot y_{\hat{k}} + y_{\hat{k}-1} = 0$$
$$-\sqrt{\beta_{\hat{k}}} \cdot y_{\hat{k}+1} + y_{\hat{k}} = 1$$
$$-\sqrt{\beta_{\hat{k}+1}} \cdot y_{\hat{k}+2} + y_{\hat{k}+1} = 0$$
$$\vdots$$
$$-\sqrt{\beta_{n-1}} \cdot y_n + y_{n-1} = 0$$
$$y_n = 0.$$

Thus, again backtracking from the value of $\hat{y}_n$ to $\hat{y}_{\hat{k}+1}$ we first obtain

$$\hat{y}_n = \cdots = \hat{y}_{\hat{k}+1} = 0.$$

Then, we have also

$$\hat{y}_{\hat{k}} = 1; \qquad y_i = \frac{\|r_{\hat{k}}\|}{\|r_i\|}, \quad i = \hat{k} - 1, \ldots, k+2,$$

while for $\hat{y}_i$, $i \in \{k+1, k\}$, we have from above the relations

$$\|r_k\| \sqrt{\beta_k} \cdot \hat{y}_k + \|r_k\| \lambda_k \cdot \hat{y}_{k+1} = \frac{\|Ap_k\| \|r_{k+2}\|}{\|r_k\|} \frac{\|r_{\hat{k}}\|}{\|r_{k+2}\|}$$
$$\frac{\sqrt{\beta_k}}{\|Ap_k\|} \cdot \hat{y}_k + \frac{\lambda_{k+1}}{\|Ap_k\|} \cdot \hat{y}_{k+1} = \frac{\|r_{k+2}\|}{\|r_k\|^2} \frac{\|r_{\hat{k}}\|}{\|r_{k+2}\|}.$$

Observing that by Lemma 5.1 $\lambda_k \neq \lambda_{k+1}$, and recalling that in Table 3 $\sqrt{\beta_k} = \|Ap_k\|/\|r_k\|$, we obtain

$$\hat{y}_{k+1} = 0, \qquad \hat{y}_k = \frac{\|r_{\hat{k}}\|}{\|r_k\|},$$

which allow to backtrack and compute also the remaining entries $\hat{y}_{k-1}, \ldots, \hat{y}_1$ of vector $\hat{y}$, being

$$\hat{y}_i = \frac{\|r_{\hat{k}}\|}{\|r_i\|}, \qquad i = k-1, \ldots, 1.$$

On the overall, the final computation of the negative curvature direction d_j yields for this subcase

$$d_j = R_j \hat{y} = \|r_{\hat{k}}\| \sum_{i=1, i \neq k+1}^{\hat{k}} \frac{r_i}{\|r_i\|^2}.$$

Finally, following the guidelines in Table 2 of [9], the conditions $\|r_i\| \geq \varepsilon$ from algorithm CG① yield that $\|d_j\|$ is bounded. Moreover, with a similar analysis in [9] the computation of d_j requires the *storage of just two vectors*.

(III) In this subcase, we assume $\hat{k} = k$. However, note that this subcase can never occur, since by (25)

$$\mu_{\hat{k}}(\hat{D}) = \frac{1}{\alpha_k s①} = \frac{1}{\|r_k\|^2} > 0$$

and therefore no negative curvature direction can be provided from the current step $\hat{k}$.

(IV) As a final subcase, we assume $\hat{k} = k+1$, i.e., by (25) $\mu_{\hat{k}}(\hat{D}) = s①/\alpha_{k+1} = -\|Ap_k\|^2 < 0$. Again, since $\hat{D}$ is diagonal, then the linear system (29) reduces to $\hat{L}^T y = e_{\hat{k}}$ (or equivalently $\hat{L}^T y = e_{k+1}$), with

$$-\sqrt{\beta_1} \cdot y_2 + y_1 = 0$$

$$\vdots$$

$$-\sqrt{\beta_{k-1}} \cdot y_k + y_{k-1} = 0$$

$$\frac{\|r_k\|\sqrt{\beta_k \lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} \cdot y_k + \frac{\|r_k\|\lambda_k\sqrt{\lambda_k}}{\sqrt{\beta_k+\lambda_k^2}} \cdot y_{k+1}$$
$$-\frac{\|Ap_k\|\|r_{k+2}\|}{\|r_k\|}\sqrt{\frac{\lambda_k}{\beta_k+\lambda_k^2}} \cdot y_{k+2} = 0$$

$$\frac{\sqrt{-\lambda_{k+1}\beta_k}}{\|Ap_k\|\sqrt{\beta_k+\lambda_{k+1}^2}} \cdot y_k + \frac{\lambda_{k+1}\sqrt{-\lambda_{k+1}}}{\|Ap_k\|\sqrt{\beta_k+\lambda_{k+1}^2}} \cdot y_{k+1}$$
$$-\frac{\|r_{k+2}\|}{\|r_k\|^2}\sqrt{-\frac{\lambda_{k+1}}{\beta_k+\lambda_{k+1}^2}} \cdot y_{k+2} = 1$$

$$-\sqrt{\beta_{k+2}} \cdot y_{k+3} + y_{k+2} = 0$$

$$\vdots$$

$$-\sqrt{\beta_{n-1}} \cdot y_n + y_{n-1} = 0$$
$$y_n = 0.$$

Now, we have for the last $n - \hat{k}$ entries of vector $\hat{y}$ the expression

$$\hat{y}_n = \cdots = \hat{y}_{\hat{k}+1} = 0.$$

On the other hand, the condition $\hat{y}_{\hat{k}+1} = \hat{y}_{\hat{k}+2} = 0$ and the above relation

$$\frac{\|r_k\|\sqrt{\beta_k \lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} \cdot y_k + \frac{\|r_k\|\lambda_k\sqrt{\lambda_k}}{\sqrt{\beta_k + \lambda_k^2}} \cdot y_{k+1} - \frac{\|Ap_k\|\|r_{k+2}\|}{\|r_k\|}\sqrt{\frac{\lambda_k}{\beta_k + \lambda_k^2}} \cdot y_{k+2} = 0$$

yield

$$\hat{y}_k = -\frac{\lambda_k}{\sqrt{\beta_k}}\hat{y}_{k+1}.$$

Recalling that now $\hat{k} = k + 1$ and in Table 3 $\|Ap_k\| = \|r_{k+1}\| = \|r_{\hat{k}}\|$, then

$$\hat{y}_{k+1} = \frac{\|r_{\hat{k}}\|}{\lambda_{k+1} - \lambda_k}\sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}}.$$

As a consequence,

$$\hat{y}_k = -\frac{\lambda_k}{\sqrt{\beta_k}} \cdot \frac{\|r_{\hat{k}}\|}{\lambda_{k+1} - \lambda_k}\sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} = -\frac{\lambda_k\|r_{\hat{k}-1}\|}{\lambda_{k+1} - \lambda_k}\sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}}$$

and for $\hat{y}_{\hat{k}-2}, \ldots, \hat{y}_1$, we have

$$\hat{y}_i = -\frac{\|r_{\hat{k}-1}\|^2}{\|r_i\|} \cdot \frac{\lambda_k}{\lambda_{k+1} - \lambda_k} \sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}}, \qquad i = \hat{k} - 2, \ldots, 1.$$

Finally, the overall negative curvature direction d_j becomes now

$$
\begin{aligned}
d_j &= R_j \hat{y} \\
&= \|r_{\hat{k}-1}\|^2 \left[\sum_{i=1}^{\hat{k}-2} -\frac{r_i}{\|r_i\|^2} \cdot \frac{\lambda_k}{\lambda_{k+1} - \lambda_k} \sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} \right] \\
&\quad -\frac{1}{\lambda_{k+1} - \lambda_k} \sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} \left(\lambda_{\hat{k}-1} r_{\hat{k}-1} - r_{\hat{k}} \right) \\
&= -\frac{\lambda_k}{\lambda_{k+1} - \lambda_k} \sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} \beta_{k-1} p_{k-1} \\
&\quad -\frac{1}{\lambda_{k+1} - \lambda_k} \sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} (\lambda_k r_k - r_{k+1}) \\
&= -\frac{\lambda_k}{\lambda_{k+1} - \lambda_k} \sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} \left[\beta_{k-1} p_{k-1} + r_k - \frac{r_{k+1}}{\lambda_k} \right] \\
&= -\frac{\lambda_k}{\lambda_{k+1} - \lambda_k} \sqrt{\frac{\beta_k + \lambda_{k+1}^2}{-\lambda_{k+1}}} \left[p_k - \frac{A p_k}{\lambda_k} \right],
\end{aligned}
$$

whose computation is well posed, since $\lambda_{k+1} < 0$. Again, by (20)–(21), the fact that $\bar{\lambda} > 0$ in Lemma 4.1 and the other hypotheses, the quantity $\|d_j\|$ is bounded. In addition, the computation of d_j evidently needs the *storage of just two n-real vectors*. □

Observation 5.1 We remark that the computation of the negative curvature direction d_j requires at most the additional storage of a couple of vectors, which confirms the competitiveness of the storage proposed in [9]. Thus, the approach in this paper does not only prove to be applicable to large-scale problems, but it also simplifies the theory in [9], which is currently in our knowledge the only proposal of iterative computation of negative curvatures for large-scale problems, which does not need any recomputing (as in [3]), and which requires neither a full matrix factorization nor any matrix storage.

6 Numerical Experience

In this section, we report the results of a numerical experience concerning the adoption of our approach, within the framework of truncated Newton methods for large-scale unconstrained optimization. We considered the truncated Newton method proposed

in [9], where we replaced the Krylov-based iterative procedure therein by the CG①
procedure. The codes were written in Fortran compiled with Gfortran 6 under Linux
Ubuntu 18.04, and the runs were performed on a PC with Intel Core i7-4790K quad-
core 4.00 GHz Processor and 32 GB RAM.

Now we strongly remark the guidelines and the limits of the numerical experience
reported in this section:

- we show how to detect and assess negative curvatures for the Hessian matrix
  $\nabla^2 f(x_j)$, at the current iterate x_j;
- we compute negative curvatures which could be able to guarantee the overall
  convergence of the optimization method toward second-order critical points;
- we do not claim that our proposal shows better numerical results with respect to
  [9], being the main focus of this paper on theoretical issues. Thus, our numerical
  experience only tests the reliability and the effectiveness of our method, rather
  than proposing a numerical comparison with the current literature;
- we also intend to check for the quality of the stationary points detected by our
  approach.

In particular, we considered all the 112 large-scale unconstrained test problems in
CUTEst [54] suite. The algorithm performs a classic nested loop of outer–inner itera-
tions. Thus, at the current jth outer iteration the algorithm iteratively solves Newton's
equation $\nabla^2 f(x_j)s = -\nabla f(x_j)$, performing a certain number of inner iterations. To
build an approximate solution s_j of Newton's equation, and possibly a negative cur-
vature direction, inner iterations are stopped whenever the following truncation rule
is satisfied

$$\|\nabla^2 f(x_j)s_j + \nabla f(x_j)\| \leq \eta_j \|\nabla f(x_j)\|,$$

being $\{\eta_j\}$ a forcing sequence, with $\eta_j \to 0$. The condition $\eta_j \to 0$ guarantees super-
linear convergence of the overall method, when close enough to the final stationary
point. As regards settings and parameters of the linesearch procedure we adopted,
as well as the overall stopping criterion, the reader can refer to [9]. (We also recall
that, unlike in [9], here we preferred not to include any nonmonotonicity in the used
algorithm, in order to clearly distinguish the contribution from our idea.) At each inner
iteration $k \geq 1$, the algorithm in Table 2 detects a curvature of the objective function,
by computing the term $p_k^T \nabla^2 f(x_j)p_k$, a negative value of the last quantity indicating
a *negative* curvature direction.

We compared two truncated Newton methods: the first (*i*) not including the use
of negative curvature directions (namely *NoNegCurv*), so that convergence to simple
stationary points could be guaranteed; the second (*ii*) including negative curvatures
(namely *NegCurv*) which satisfy Assumption 2.1, implying convergence to stationary
points where second-order necessary optimality conditions are fulfilled. Thus, by a
comparison between them, we might have expected:

(a) (*ii*) to be more efficient than (*i*) in terms of the computational effort (in our large-
    scale setting we measured the computational effort through the number of inner
    iterations, which are representative of the overall computational burden, including
    CPU time);

(b) the quality of the solutions detected by (*ii*), i.e., the value of the objective function at the solution, is expected to be on average not worse than in the case of (*i*), since for (*ii*) the solution points satisfy additional theoretical properties;

(c) the stationarity (measured by $\|\nabla f(x^*)\|$) of the final solution detected using (*ii*) is possibly expected to be competitive with respect to (*i*). This is because in a neighborhood of the solution point, our proposal is expected to collect more information on the objective function.

The above considerations are to large extent confirmed by our numerical experience as detailed in the following.

First, note that using (*ii*), we detected negative curvatures on 40 test problems out of 112; of course, this does not imply that the remaining 72 test problems only include *convex* functions. It rather implies that on 72 problems, no regions of concavity for the objective function were encountered. For these 40 test problems, the obtained results in terms of number of (outer) iterations (*it*), number of function evaluations (*nf*), number of inner iterations (*inner-it*), optimal function value ($f(x^*)$), gradient norm at the optimal point ($\|g(x^*)\|$), solution time in seconds (*time*), are reported in Tables 4 and 5. In particular, for each test problem, we report results using both the *NoNegCurv* method (top row) and the *NegCurv* method (bottom row).

By observing these results, we first note that on two test problems both the algorithms fail to converge within the maximum CPU time of 900 s. The comparison on the remaining test problems shows that in most cases algorithm *NegCurv* performs the best in terms of solution time and inner iterations, confirming expectation (a) which is our main goal. The results highlight only one test problem (GENHUPMS 1000) where the use of *NegCurv* yields a significant worsening of the performance. We easily realize that including our procedure to compute negative curvature directions allows to both speed up the overall convergence and decrease the number of inner iterations.

The detailed results only partially validate also (b) and (c). As regards (b), since in a few test problems the algorithms converge to different points, a sound statistical analysis cannot be given, though a better optimal value is sometimes observed by using *NegCurv* algorithm. Similarly, as concerns (c), the values of $\|\nabla f(x^*)\|$ provided by *NoNegCurv* and *NegCurv* seem to a large extent comparable on this test set.

To have an overview of the effectiveness and the robustness of the approach we propose in this paper, we now consider summary results by using *performance profiles* [55]. The performance profiles represent a popular and widely used tool for providing objective information when benchmarking optimization algorithms. Their meaning can be summarized as follows: Suppose you have a set of solvers $\mathcal{S}$ to be compared on a set of test problems $\mathcal{P}$.

For each problem $p \in \mathcal{P}$ and solver $s \in \mathcal{S}$, define t_{ps} the statistic obtained by running solver s on problem p. Namely, $t_{ps} \geq 0$ is a performance measure of interest, e.g., solution time, number of function evaluations, etc. The performance on problem p by solver s is compared with the best performance by any solver on this problem by means of the performance ratio

$$r_{ps} = \frac{t_{ps}}{\{t_{ps} \mid s \in \mathcal{S}\}}.$$

Table 4 Complete results for CUTEst test problems where negative curvature directions are encountered. For each test problem, results for *NoNegCurv* (top row) and *NegCurv* (bottom row) are reported. Part 1

PROBLEM	n	it	nf	inner-it	$f(x^*)$	$\|\nabla f(x^*)\|$	time
BROYDN7D	1000	97	286	4951	0.415613D+03	0.327528D-04	0.65
		89	243	4160	0.313698D+03	0.162507D-03	0.55
BROYDN7D	10000	215	633	33514	0.370260D+04	0.235258D-03	25.03
		224	682	27745	0.424134D+04	0.447003D-03	20.23
CHAINWOO	1000	62	69	280	0.159945D+02	0.102320D-03	0.05
		58	65	226	0.124217D+02	0.197676D-03	0.04
CHAINWOO	10000	120	140	251	0.100000D+01	0.234299D-03	0.53
		116	122	254	0.124217D+02	0.472326D-03	0.48
COSINE	1000	13	14	15	−0.999000D+03	0.264043D-03	0.00
		10	11	11	−0.999000D+03	0.203307D-04	0.00
COSINE	10000	12	13	14	−0.999900D+04	0.716347D-03	0.03
		10	12	11	−0.999900D+04	0.106893D-04	0.03
CURLY10	1000	61	64	6345	−0.100316D+06	0.163323D-03	0.72
		61	63	6715	−0.100316D+06	0.113065D-03	0.78
CURLY10	10000	134	137	58009	−0.100316D+07	0.413774D-03	44.49
		134	139	53351	−0.100316D+07	0.746587D-03	40.84
CURLY20	1000	63	72	7779	−0.100316D+06	0.154139D-04	1.24
		64	67	6474	−0.100316D+06	0.998525D-04	1.01

Table 4 continued

PROBLEM	n	it	nf	inner-it	$f(x^*)$	$\|\nabla f(x^*)\|$	time
CURLY20	10000	138	142	76317	−0.100316D+07	0.145586D-03	90.51
		142	147	67658	−0.100316D+07	0.280150D-03	80.75
CURLY30	1000	66	67	7057	−0.100316D+06	0.107850D-04	1.36
		64	73	6571	−0.100316D+06	0.145846D-03	1.29
DIXMAANH	3000	20	25	381	0.100000D+01	0.102275D-05	0.11
		20	26	382	0.100000D+01	0.144888D-05	0.12
DIXMAANJ	1500	23	41	2495	0.100000D+01	0.494360D-05	0.37
		25	65	2859	0.100000D+01	0.232700D-05	0.42
DIXMAANJ	3000	31	66	8510	0.100000D+01	0.222723D-05	2.09
		31	90	4532	0.100000D+01	0.761067D-05	1.12
DIXMAANK	1500	23	40	2773	0.100000D+01	0.281390D-05	0.41
		23	51	2896	0.100000D+01	0.360457D-05	0.43
DIXMAANK	3000	30	79	6545	0.100000D+01	0.446484D-05	1.63
		29	77	5409	0.100000D+01	0.388748D-06	1.38
DIXMAANL	1500	26	56	3550	0.100000D+01	0.887471D-06	0.51
		30	95	3575	0.100000D+01	0.614269D-05	0.53
DIXMAANL	3000	29	63	2134	0.100000D+01	0.911311D-05	0.57
		30	99	6895	0.100000D+01	0.250139D-05	1.72
FLETCBV3	1000	9	9	15	−0.443759D+05	0.656347D+00	0.00
		116	116	116	−0.507763D+05	0.322581D+00	0.05
FLETCBV3	10000	1449	1449	3912	−0.599843D+11	0.141601D+03	7.25
		107	107	178	−0.247675D+11	0.187512D+03	0.47

Table 5 Complete results for CUTEst test problems where negative curvature directions are encountered. For each test problem, results for *NoNegCurv* (top row) and *NegCurv* (bottom row) are reported. Part 2

PROBLEM	n	it	nf	inner-it	$f(x^*)$	$\|\nabla f(x^*)\|$	time
FLETCHCR	1000	1528	1838	18100	0.361098D-05	0.546173D-02	2.22
		1533	1892	18151	0.933135D-08	0.290578D-02	2.06
FLETCHCR	10000	14836	17273	199422	0.149913D-10	0.560105D-04	154.84
		14832	17284	199458	0.103640D-05	0.178020D-02	153.25
FREUROTH	1000	35	39	44	0.121470D+06	0.127313D-03	0.01
		35	40	44	0.121470D+06	0.136981D-03	0.01
GENHUMPS	1000	4391	53691	3241842	0.140582D-11	0.749888D-06	370.50
		5574	67930	4460650	0.115301D-10	0.214771D-05	509.64
GENHUMPS	10000	–	–	–	–	–	> 900
		–	–	–	–	–	> 900
GENROSE	1000	594	2665	10066	0.100000D+01	0.392231D-02	1.18
		600	2854	10307	0.100000D+01	0.313600D-02	1.18
GENROSE	10000	5444	25674	114123	0.100000D+01	0.212735D-02	87.06
		5338	27904	110871	0.100000D+01	0.740876D-03	85.18
MSQRTALS	1024	69	74	3595	0.807903D-07	0.207559D-03	3.62
		70	76	4237	0.290514D-08	0.648068D-04	4.20
MSQRTBLS	1024	65	67	2765	0.598454D-09	0.154215D-04	2.85
		62	64	1398	0.252892D-06	0.797780D-03	1.48
NONCVXUN	1000	115	850	24786	0.232077D+04	0.832905D-02	6.70
		107	719	18243	0.232901D+04	0.932126D-02	4.78

Table 5 continued

PROBLEM	n	it	nf	inner-it	f(x*)	‖∇f(x*)‖	time
NONCVXUN	10000	–	–	–	–	–	> 900
		–	–	–	–	–	> 900
NONCVXU2	1000	109	885	27859	0.231865D+04	0.215093D-02	7.74
		105	842	26149	0.231710D+04	0.154901D-02	7.46
NONCVXU2	10000	–	–	–	–	–	> 900
		–	–	–	–	–	> 900
SINQUAD	1000	38	44	49	−0.294250D+06	0.350896D-02	0.02
		39	44	49	−0.294250D+06	0.523566D-02	0.02
SINQUAD	10000	77	78	81	−0.264231D+08	0.410340D-03	0.34
		100	104	105	−0.264231D+08	0.410251D-03	0.44
SPMSRTLS	1000	33	34	115	0.928595D-08	0.207045D-03	0.03
		34	36	117	0.289313D-08	0.127994D-03	0.03
SPMSRTLS	10000	43	44	177	0.106828D-06	0.362738D-03	0.33
		40	41	182	0.221391D-07	0.274285D-03	0.31
TQUARTIC	1000	13	15	16	0.146710D-06	0.242775D-04	0.00
		12	16	15	0.877004D-05	0.187297D-03	0.00
WOODS	1000	87	120	175	0.490714D-06	0.863474D-03	0.03
		89	139	176	0.473558D-06	0.848294D-03	0.03
WOODS	10000	162	207	250	0.262918D-10	0.411512D-04	0.39
		167	217	253	0.720362D-09	0.215006D-03	0.39

Moreover, an upper bound $\bar{r}$ is chosen such that $r_{ps} \leq \bar{r}$ for all $s \in \mathcal{S}$ and $p \in \mathcal{P}$ and if a solver s fails to solve problem p then r_{ps} is set to $\bar{r}$. The performance profile of the solver s is the function

$$\rho_s(\tau) = \frac{\left|\{p \in \mathcal{P} \mid t_{ps} \leq \tau\}\right|}{|\mathcal{S}|},$$

namely the cumulative distribution function for the performance ratio.

In particular, we report in Figs. 1 (*full profile*) and 2 (*detail profile*) the performance profiles comparing *NoNegCurv* and *NegCurv* algorithms in terms of inner iterations. The test set considered includes the 40 test problems reported in Tables 4 and 5.

The detailed plot reported in Fig. 2 clearly shows the effectiveness of *NegCurv* algorithm with respect to *NoNegCurv*. Indeed, as an example, let us consider the value of abscissa 1.2 in Fig. 2. The plots show that the *NegCurv algorithm* is able to solve about 62% of the test problems within 1.2 times the number of inner iterations of the best algorithm. Conversely, the *NoNegCurv algorithm* is able to solve up to 78% of the test problems within the same number of inner iterations. On the other hand, in terms of robustness the algorithms can be considered comparable with a slight preference for *NegCurv* algorithm as evidenced by Fig. 1. The last consideration follows from the observation that for values of the abscissa parameter larger than 3.5, the two plots basically tend to overlap.

7 Conclusions

We proposed a novel approach for the efficient solution of large-scale unconstrained optimization problems, where the detected solutions likely are endowed with strong theoretical properties. Our proposal exploits the simplicity of an algebra associated with the numeral *grossone*, which was recently introduced in the literature to handle infinite and infinitesimal quantities.

We were able to extend the results in [9] in view of a theoretical simplification, avoiding to make reference to Planar-CG methods which require a more complex analysis. The theory in this paper allows us to guarantee that the iterative computation of negative curvatures does not need any matrix storage, while preserving convergence toward points satisfying second-order necessary optimality conditions.

Then, we also provided numerical results, which show the efficiency of our proposal. We remark that the focus of this paper is not on a numerical comparison among different algorithms which exploit negative curvature directions. Rather, we paired the approach in [9] with a novel paradigm provided by *grossone*, in light of preserving numerical efficiency within a sound theoretical framework, in dealing with nonconvex problems. This is the first stage toward a more complete numerical experience where the iterative algorithm $CG_{\text{①}}$ can be fully tested, including even more challenging problems from real-world applications.

Observe that the proposed approach is independent under multiplication of the function by a positive scaling constant or adding a shifting constant. This is an important property that is specially exploited in the global optimization framework (see, e.g., [24]), since *strongly homogeneous* algorithms are definitely appealing. Furthermore,
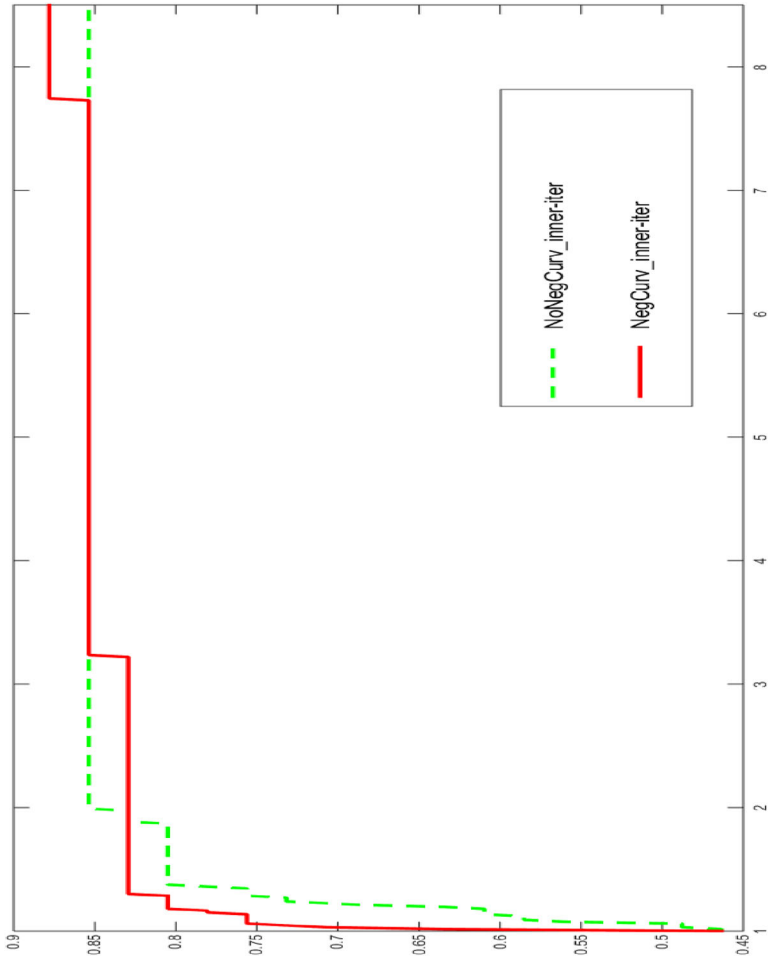
Fig. 1 Performance profile relative to a comparison between (*ii*) and (*i*), with respect to the number of inner iterations, where negative curvatures are (*red line*)/are not (*green dashed line*) used (*full profile*)
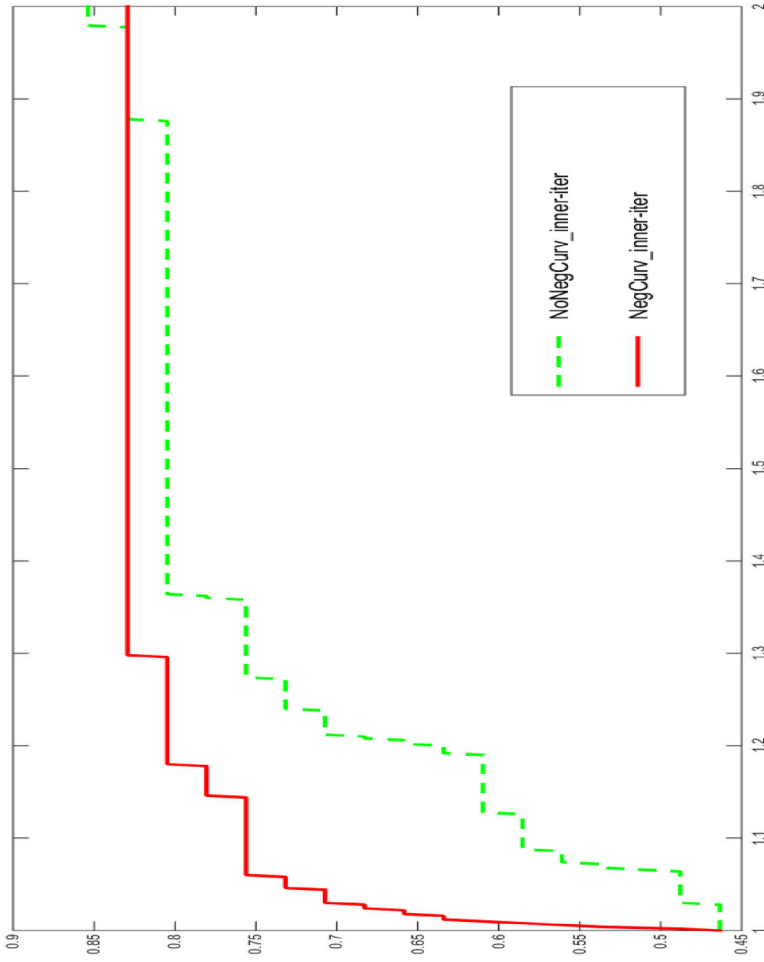
Fig. 2 Performance profile relative to a comparison between (*ii*) and (*i*), with respect to the number of inner iterations, where negative curvatures are (*red line*)/are not (*green dashed line*) used (*detail profile*)

the local solver described in Sect. 6 may be considered to enhance the efficiency of the algorithm in [56], simply replacing the local solver used therein by our proposal.

References

1. Ferris, M., Lucidi, S., Roma, M.: Nonmonotone curvilinear line search methods for unconstrained optimization. Comput. Optim. Appl. **6**, 117–136 (1996)
2. Goldfarb, D.: Curvilinear path steplength algorithms for minimization which use directions of negative curvature. Math. Program. **18**(1), 31–40 (1980)
3. Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Exploiting negative curvature directions in linesearch methods for unconstrained optimization. Optim. Methods Softw. **14**, 75–98 (2000)
4. Goldfarb, D., Mu, C., Wright, J., Zhou, C.: Using negative curvature in solving nonlinear programs. Comput. Optim. Appl. **68**(3), 479–502 (2017)
5. Lucidi, S., Rochetich, F., Roma, M.: Curvilinear stabilization techniques for truncated Newton methods in large-scale unconstrained optimization. SIAM J. Optim. **8**, 916–939 (1998)
6. McCormick, G.P.: A modification of Armijo's step-size rule for negative curvature. Math. Program. **13**(1), 111–115 (1977)
7. Moré, J., Sorensen, D.: On the use of directions of negative curvature in a modified Newton method. Math. Program. **16**, 1–20 (1979)
8. Curtis, F., Robinson, D.: Exploiting negative curvature in deterministic and stochastic optimization. Math. Program. **176**, 69–94 (2019)
9. Fasano, G., Roma, M.: Iterative computation of negative curvature directions in large scale optimization. Comput. Optim. Appl. **38**(1), 81–104 (2007)
10. Fasano, G., Lucidi, S.: A nonmonotone truncated Newton–Krylov method exploiting negative curvature directions, for large scale unconstrained optimization. Optim. Lett. **3**(4), 521–535 (2009)
11. Hestenes, M.R., Stiefel, E.L.: Methods of conjugate gradients for solving linear systems. J. Res. Nat. Bur. Stand. **49**, 409–436 (1952)
12. De Leone, R., Fasano, G., Sergeyev, Y.D.: Planar methods and grossone for the conjugate gradient breakdown in nonlinear programming. Comput. Optim. Appl. **71**, 73–93 (2018)
13. Sergeyev, Y.D.: Numerical infinities and infinitesimals: methodology, applications, and repercussions on two Hilbert problems. EMS Surv. Math. Sci. **4**(2), 219–320 (2017)
14. Fasano, G.: Conjugate gradient (CG)-type method for the solution of Newton's equation within optimization frameworks. Optim. Methods Softw. **19**(3–4), 267–290 (2004)
15. Lolli, G.: Metamathematical investigations on the theory of grossone. Appl. Math. Comput. **255**, 3–14 (2015)
16. Margenstern, M.: Using Grossone to count the number of elements of infinite sets and the connection with bijections. p-Adic Numbers Ultrametric Anal. Appl. **3**(3), 196–204 (2011)
17. Montagna, F., Simi, G., Sorbi, A.: Taking the Pirahã seriously. Commun. Nonlinear Sci. Numer. Simul. **21**(1–3), 52–69 (2015)

18. Sergeyev, Y.D.: Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them. USA patent 7,860,914 (2010)
19. Cococcioni, M., Cudazzo, A., Pappalardo, M., Sergeyev, Y.D.: Solving the lexicographic multi-objective mixed-integer linear programming problem using branch-and-bound and Grossone methodology. Commun. Nonlinear Sci. Numer. Simul. **84**, 105177 (2020)
20. Cococcioni, M., Pappalardo, M., Sergeyev, Y.D.: Lexicographic multi-objective linear programming using grossone methodology: theory and algorithm. Appl. Math. Comput. **318**, 298–311 (2018)
21. De Cosmis, S., Leone, R.D.: The use of grossone in mathematical programming and operations research. Appl. Math. Comput. **218**(16), 8029–8038 (2012)
22. De Leone, R.: Nonlinear programming and grossone: quadratic programming and the role of constraint qualifications. Appl. Math. Comput. **318**, 290–297 (2018)
23. Gaudioso, M., Giallombardo, G., Mukhametzhanov, M.S.: Numerical infinitesimals in a variable metric method for convex nonsmooth optimization. Appl. Math. Comput. **318**, 312–320 (2018)
24. Sergeyev, Y.D., Kvasov, D.E., Mukhametzhanov, M.S.: On strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales. Commun. Nonlinear Sci. Numer. Simul. **59**, 319–330 (2018)
25. Caldarola, F.: The Sierpinski curve viewed by numerical computations with infinities and infinitesimals. Appl. Math. Comput. **318**, 321–328 (2018)
26. Sergeyev, Y.D.: Numerical point of view on Calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains. Nonlinear Anal. Ser. A Theory Methods Appl. **71**(12), e1688–e1707 (2009)
27. Sergeyev, Y.D.: Numerical infinities applied for studying Riemann series theorem and Ramanujan summation. In: AIP Conference Proceedings of ICNAAM 2017, vol. 1978, p. 020004. AIP Publishing, New York (2018). https://doi.org/10.1063/1.5043649
28. Zhigljavsky, A.: Computing sums of conditionally convergent and divergent series using the concept of grossone. Appl. Math. Comput. **218**(16), 8064–8076 (2012)
29. Caldarola, F.: The exact measures of the Sierpinski d-dimensional tetrahedron in connection with a diophantine nonlinear system. Commun. Nonlinear Sci. Numer. Simul. **63**, 228–238 (2018)
30. D'Alotto, L.: A classification of two-dimensional cellular automata using infinite computations. Indian J. Math. **55**, 143–158 (2013)
31. Sergeyev, Y.D.: Evaluating the exact infinitesimal values of area of Sierpinski's carpet and volume of Menger's sponge. Chaos Solitons Fractals **42**(5), 3042–3046 (2009)
32. Falcone, A., Garro, A., Mukhametzhanov, M.S., Sergeyev, Y.D.: A simulink-based infinity computer simulator and some applications. Lecture Notes in Computer Science 11974 LNCS, pp. 362–369 (2020). https://doi.org/10.1007/978-3-030-40616-5_31
33. Iudin, D.I., Sergeyev, Y.D., Hayakawa, M.: Infinity computations in cellular automaton forest-fire model. Commun. Nonlinear Sci. Numer. Simul. **20**(3), 861–870 (2015)
34. Margenstern, M.: Fibonacci words, hyperbolic tilings and grossone. Commun. Nonlinear Sci. Numer. Simul. **21**(1–3), 3–11 (2015)
35. Sergeyev, Y.D.: Counting systems and the First Hilbert problem. Nonlinear Anal. Ser. A Theory Methods Appl. **72**(3–4), 1701–1708 (2010)
36. Sergeyev, Y.D., Garro, A.: Single-tape and multi-tape Turing machines through the lens of the Grossone methodology. J. Supercomput. **65**(2), 645–663 (2013)
37. Fiaschi, L., Cococcioni, M.: Numerical asymptotic results in game theory using Sergeyev's Infinity Computing. Int. J. Unconv. Comput. **14**(1), 1–25 (2018)
38. Rizza, D.: A study of mathematical determination through Bertrand's Paradox. Philosophia Mathematica **26**(3), 375–395 (2018)
39. Rizza, D.: Numerical methods for infinite decision-making processes. Int. J. Unconv. Comput. **14**(2), 139–158 (2019)
40. Amodio, P., Iavernaro, F., Mazzia, F., Mukhametzhanov, M., Sergeyev, Y.D.: A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic. Math. Comput. Simul. **141**, 24–39 (2017)
41. Sergeyev, Y.D.: Higher order numerical differentiation on the Infinity Computer. Optim. Lett. **5**(4), 575–585 (2011)
42. Iavernaro, F., Mazzia, F., Mukhametzhanov, M.S., Sergeyev, Y.D.: Conjugate-symplecticity properties of Euler–Maclaurin methods and their implementation on the Infinity Computer. Appl. Numer. Math. **155**, 58–72 (2020). https://doi.org/10.1016/j.apnum.2019.06.011

43. Sergeyev, Y.D., Mukhametzhanov, M.S., Mazzia, F., Iavernaro, F., Amodio, P.: Numerical methods for solving initial value problems on the Infinity Computer. Int. J. Unconv. Comput. **12**(1), 3–23 (2016)
44. Sergeyev, Y.D.: Independence of the grossone-based infinity methodology from non-standard analysis and comments upon logical fallacies in some texts asserting the opposite. Found. Sci. **24**(1), 153–170 (2019)
45. Golub, G.H., Loan, C.F.V.: Matrix Computations, 4th edn. The Johns Hopkins University Press, Baltimore (2013)
46. Paige, C., Saunders, M.: Solution of sparse indefinite systems of linear equations. SIAM J. Numer. Anal. **12**, 617–29 (1975)
47. HSL_MI02 Symmetric possibly-indefinite system: SYMMBK method. Harwell Mathematical Software Library http://www.hsl.rl.ac.uk (2013)
48. Fasano, G.: Planar-conjugate gradient algorithm for large scale unconstrained optimization, part 1: theory. J. Optim. Theory Appl. **125**(3), 523–541 (2005)
49. Fasano, G.: Planar-conjugate gradient algorithm for large scale unconstrained optimization, part 2: application. J. Optim. Theory Appl. **125**(3), 543–558 (2005)
50. Fasano, G.: Lanczos-conjugate gradient method and pseudoinverse computation, on indefinite and singular systems. J. Optim. Theory Appl. **132**(2), 267–285 (2007)
51. Nash, S.G.: A survey of truncated-Newton methods. J. Comput. Appl. Math. **124**, 45–59 (2000)
52. Fasano, G.: Planar-CG methods and matrix tridiagonalization in large scale unconstrained optimization. In: Di Pillo, G., Murli, A. (eds.) High Performance Algorithms and Software for Nonlinear Optimization, pp. 243–263. Kluwer Academic Publishers, Dordrecht (2003)
53. Fasano, G., Pesenti, R.: Conjugate direction methods and polarity for quadratic hypersurfaces. J. Optim. Theory Appl. **175**, 764–794 (2017)
54. 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)
55. Dolan, E.D., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
56. Žilinskas, A., Gillard, J., Scammell, M., Zhigljavsky, A.: Multistart with early termination of descents. J. Global Optim. (2019). https://doi.org/10.1007/s10898-019-00814-w

Affiliations

Renato De Leone[1] · **Giovanni Fasano[2]** · **Massimo Roma[3]** · **Yaroslav D. Sergeyev[4,5]**

Renato De Leone
renato.deleone@unicam.it

Massimo Roma
roma@diag.uniroma1.it

Yaroslav D. Sergeyev
yaro@dimes.unical.it

[1] Università degli Studi di Camerino, Camerino, Italy

[2] Università Ca' Foscari Venezia, Venezia, Italy

[3] SAPIENZA – Università di Roma, Rome, Italy

[4] Università della Calabria, Rende, Italy

[5] Lobachevsky State University, Nizhniy Novgorod, Russia