ISSN 2281-4299

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Exploiting SYMMBK method for the full computation of negative curvature directions

Giovanni Fasano
Christian Piermarini
Massimo Roma

Technical Report n. 06, 2023

Exploiting SYMMBK method for the full computation of negative curvature directions

Giovanni Fasano
Dipartimento di Management
Università Ca' Foscari Venezia, Italy
fasano@unive.it

Christian Piermarini, Massimo Roma
Dipartimento di Ingegneria Informatica, Automatica e Gestionale "A. Ruberti"
SAPIENZA – Università di Roma, Italy
piermarini@diag.uniroma1.it, roma@diag.uniroma1.it

Abstract

In this paper we consider the issue of computing negative curvature directions, for non-convex functions, within Newton–Krylov methods for large scale unconstrained optimization. In the last decades this issue has been widely investigated in the literature, and different approaches have been proposed. We focus on the well known SYMMBK method introduced for solving large scale symmetric possibly indefinite linear systems [5, 9, 11, 28], and show how to exploit it to yield an effective negative curvature direction in optimization frameworks. The distinguishing feature of our proposal is that the computation of negative curvature directions is basically carried out as by–product of SYMMBK procedure, without storing no more than one additional vector. Hence, no explicit matrix factorization or matrix storage is required. A numerical experience is reported, showing the reliability of the novel approach we propose.

In addition, we also propose a novel general tool, to profile the quality of the solutions found by different solvers in optimization frameworks. The new proposed tool, namely *Quality Profile*, draws its inspiration from both performance and data profiles [18, 36], sharing with them a number of basic properties but also showing several strong differences on fruitful use cases.

Keywords: Large scale unconstrained optimization, Newton–Krylov methods, Negative curvature directions, Second order critical points, Quality profiles

1 Introduction

We focus on linesearch–based Newton–Krylov methods that are widely used for solving large scale unconstrained optimization problems, namely to determine a local minimizer of the problem:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.1}$$

being $f : \mathbb{R}^n \to \mathbb{R}$, with $f(x)$ twice continuously differentiable in $\mathbb{R}^n$. Given an initial guess $x_0 \in \mathbb{R}^n$, at each iteration of these methods, a new iterate is generated according to the iterative scheme

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where d_k is a search direction and $\alpha_k > 0$ is a suited steplength. Since the search direction is determined by means of an iterative Krylov–subspace method, actually our linesearch–based scheme encompasses two nested loops:

- the *outer iterations*, where starting from the current iterate x_k, a new iterate is generated according to the scheme (1.2), and where α_k is computed by a linesearch procedure;
- the *inner iterations*, namely the iterations of the Krylov–subspace method used for approximately solving the Newton equation

$$\nabla^2 f(x_k) d = -\nabla f(x_k). \tag{1.3}$$

Newton–Krylov methods are also called Truncated Newton methods (or inexact Newton methods) since, on large scale problems, the inner iterations are usually "truncated", i.e. terminated according to a suited stopping criterion, still ensuring superlinear converge rate [16, 17]. As concerns global convergence properties, convergence to first order critical points is ensured, i.e. points which satisfies first order necessary optimality conditions, namely toward stationary points. For a complete overview of these methods we refer the reader to the survey paper by Nash [34].

Among the most commonly used iterative methods adopted in the inner iterations we find the Conjugate Gradient (CG) method. In the convex case (positive definitive Hessian), it performs well, but in the case of indefinite Hessian it may untimely breakdown (pivot breakdown) or become numerically unstable. This occurs when in the CG iterations a direction s such that $s^\top \nabla^2 f(x_k) s < 0$ is encountered before satisfying the termination criterion. As well known, such directions are called *negative curvature directions* for the function f at x_k and, as we will discuss afterwards, they may play an important rule for improving both the converge properties and the efficiency of the method. To overcome the drawback of the CG untimely stopping, some authors proposed the use of the Lanczos process in the inner iterations in place of the CG [30, 33]. The two methods are, to some extent, equivalent in the case of convex functions, but the iterations of the Lanczos process do not prematurely stop.

It is important to point out that, even if these methods use second order information on the objective function, actually they are "matrix–free" (Hessian–free), since Hessian matrix is never stored and information is gained by means of a routine which provides the matrix–vector product of the Hessian times a vector. This feature, along with their scale invariance properties, makes these methods very attractive also in recently raised machine learning applications (including deep neural networks). Hessian times vector products can be computed at a cost which is a small multiple of the cost of a gradient evaluation (see, e.g., [3]). Such problems arise, for instance, in training deep neural networks, in low rank subspace clustering problems [29] and many other contexts. As clearly pointed out in the recent paper by Curtis and Robinson [13], today it is really needful to design new methods able to efficiently solve non-convex problems by exploiting negative curvature directions, both in deterministic and stochastic optimization.

Moreover, the use of negative curvature directions allows to avoid saddle points due to the properties that algorithms inherit in terms of convergence to second-order points (see, e.g., statistical physics, random matrix theory and training multilayer perceptron networks [4, 10, 14]).

Indeed, when seeking for local minima, it may take a very large number of iterations to escape from a neighborhood of a saddle point. In particular, saddle points can represent a much more predominant and frequent obstacle to skip with respect to local minima, when training layered feed-forward neural networks [2]. In this regard, negative curvature directions can be extremely beneficial to deal with the proliferation of saddle points.

The use of negative curvature directions within (modified) Newton methods dates back to the seminal papers [31] and [32]. These led to the development of several methods based on a combination of a Newton–type direction d_k and a negative curvature direction s_k where the iterative scheme (1.2) is replaced by

$$x_{k+1} = x_k + \alpha_k^2 d_k + \alpha_k s_k, \tag{1.4}$$

and α_k is obtained by means of a *curvilinear linesearch*. The use of negative curvature directions has a twofold importance: from the computational point of view, a movement along a descent negative curvature direction allows the algorithm to escape from regions of nonconvexity of the objective function. From the theoretical point point of view, the use of suited negative curvature directions enables defining methods converging to second order critical points, i.e., points which satisfy second order necessary optimality conditions, namely stationary points where the Hessian is positive semidefinite. Note that Newton–type methods based on trust region approach, naturally possess such convergence property (see [11]). Linesearch–based methods converging to second order critical points have been also proposed in the framework of nonmonotone methods [24] and extended to large scale setting in [30]. To ensure this stronger convergence property, the negative curvature direction used must be an approximation of an eigenvector of the Hessian matrix corresponding to the most negative eigenvalue. More precisely, the negative curvature direction s_k is required to be a *bounded descent direction* satisfying the following property:

$$s_k^\top \nabla^2 f(x_k) s_k \longrightarrow 0 \quad \text{implies} \quad \min\left[0, \lambda_{\min}\left(\nabla^2 f(x_k)\right)\right] \longrightarrow 0, \tag{1.5}$$

where $\lambda_{\min}\left(\nabla^2 f(x_k)\right)$ is the smallest eigenvalue of the Hessian matrix $\nabla^2 f(x_k)$. Computing a direction s_k satisfying (1.5) is a very computationally expensive task, since it involves the spectrum of $\nabla^2 f(x_k)$. Moreover, most of the strategies proposed in literature for computing negative curvature directions satisfying (1.5) usually rely on matrix factorizations (see e.g., the Bunch and Parlett decomposition proposed in [32]), so that in the large scale setting they become impracticable. On the other hand, also iterative methods usually adopted typically need to store a large matrix, hence they are unsuited for handling large scale problems; this is the case of the method proposed in [30] where the Lanczos process is used and the storage of a matrix of the Lanczos vectors generated at each iteration is theoretically needed to computed adequate negative curvature directions. The strategy adopted in [30] consisting of imposing an upper bound on the number of the Lanczos vectors stored, actually implies the loss of the theoretical property of converging to second order critical points.

A different approach for computing suited negative curvature directions is proposed in [26]. In this paper, based on the close relation between CG and Lanczos methods, the Lanczos vectors are regenerated by rerunning the recurrence when needed. In this manner, matrix storage is avoided, but a non-negligible additional computational effort is required, due to rerunning operations.

A first attempt, in the case of indefinite Hessian, to iteratively compute negative curvature directions satisfying (1.5), without requiring storage of any large matrix or rerunning the iterative process, is represented by the use of the Planar-CG algorithm as proposed in [23] (we refer the reader to the papers [19, 20] for a complete description of the Planar-CG schemes). Besides providing a general theoretical framework which guarantees convergence to second order critical points, in [23] results of a preliminary numerical experience are reported showing that the proposed approach is reliable and promising. However, we believe that there is still need to further investigate on how to determining effective negative curvature directions to be used within a truncated Newton method. In particular, besides guaranteeing convergence toward second order critical points, the use of such directions should improve the overall efficiency of the method and its capability to detect better local minimizers. In the current paper, similarly to [23], we provide the computation

of a negative curvature direction s_k that fulfills (1.5). Conversely, with respect to [23] we reduce the related additional storage requirement (to obtain s_k) to just one vector, by means of replacing the use of the CG method.

Another issue worth investigating concerns how to combine a Newton–type direction and a negative curvature direction taking into account their possible different scaling. As well known, Newton–type direction is well–scaled (particularly when close to a local minimizer), while a negative curvature direction may be possibly not. Hence, possible inefficiency may arise due to the use of a combination of the two directions. Based on this remark, in [26], at each outer iteration, given a descent pair of directions (d_k, s_k), only one of the two directions is selected and a suited linesearch is performed along the chosen direction. The selection of the most promising direction is performed by estimating the rate of decrease of the quadratic model of the objective function in both directions. Following this approach, in [21], a new Truncated Newton method is proposed; a test based on the quadratic model is used to select the most promising between the two directions and an appropriate linesearch procedure is adopted depending on the search direction selected. On the same guideline of selecting the most effective direction, in [35] the authors propose to consider three alternatives: to select one of the two directions d_k and s_k, or possibly to make use of a combination of both the directions. This latter choice is adopted when both the directions are promising in terms of decrease of the quadratic model. On the other hand, as studied in [1], it would be very beneficial to perform a scaling process before combining the two directions. Finally, we mention that in the recent paper [13], a novel framework has been proposed for combining the two directions, alternating two–step and dynamic step approaches.

Observe that negative curvature directions, obtained as by–product of iterative optimization methods, have also been investigated within the recent literature related to preconditioners for large scale linear systems. In particular, quasi–Newton based updates for the construction of preconditioners were proposed, both within Truncated Newton Methods and Nonlinear CG methods, where the combined use of both positive and negative curvature directions can be fruitfully exploited (see also [7] and [8]).

In this paper, we propose the use of an alternative iterative procedure to be used within Newton–Krylov methods, for computing an effective negative curvature direction. In particular, we refer to SYMMBK method for solving large scale symmetric possibly indefinite linear systems [5, 9, 11, 28]. Such method has been recently successfully applied within Truncated Newton methods to yield a gradient related Newton–type direction [6]. More precisely, in that paper a modified Bunch–Kaufmann factorization has been proposed to be used within SYMMBK algorithm for solving the Newton equation, at each outer iteration. Indeed, the Bunch–Kaufmann factorization is an effective and stable matrix decomposition, but when used for solving the Newton equation might provide a direction which is not gradient–related, if the objective function is nonconvex. The modification proposed in [6] enables obtaining a direction that is gradient–related and effective in practice. Hence, the idea in the current paper to possibly use the same procedure for obtaining also a negative curvature direction satisfying (1.5), with a minimal additional storage. For the sake of completeness we remark that the next two novelties are included in the current paper:

- we define the computation of a novel negative curvature direction based on a modified SYMMBK method: we first prove some theoretical achievements associated with the last vector, and then we propose an extended numerical experience on a wide range of test problems from CUTEst collection [27];

- we address a new ad hoc benchmarking procedure, to evaluate the *quality* of the local minimizers computed adopting our proposal for a novel negative curvature direction. The benchmarking procedure (namely *Quality Profiles*) draws inspiration from [18] and [36], and yields specific guidelines to take into account a ranking method among algorithms, when the objective function values they generate converge to different points. Furthermore, on one hand our benchmarking system inherits almost the same appealing features of the profiling methods in [18] and [36]. On the other hand, it allows a clearer comparison among different algorithms, through zooming opportunities in the resulting plots which are not immediately extendable

to [18] and [36]. Finally, the proposed tool is able to suggest a ranking method among many codes, by using the objective function values at the solution points.

The paper is organized as follows. In Section 2 we briefly recall the SYMMBK procedure and provide some preliminaries on the use of negative curvature directions within a Truncated Newton method. Sections 3 and Section 4 describe how to use SYMMBK for defining a suited negative curvature direction. Section 5 deals with the iterative computation of negative curvature directions. Theoretical results concerning the computed negative curvature directions are included in Section 6. Sections 7 and 8 describe a possible alternative proposal based on conjugate directions. The results of an extensive numerical experimentation are reported in Section 9 where novel profiles (named *Quality Profiles*) are introduced for a better comparison when ranking different algorithms.

As regards the taxonomy adopted in this paper, if not specified elsewhere, subscripts will denote the entries of vectors/matrices, $\{e_j\}$ is the canonical basis of $\mathbb{R}^n$, $\|\cdot\|$ specifies the Euclidean norm. Given a square matrix A, $\lambda_{\min}[A]$ denotes the its smallest eigenvalue and $\kappa(A)$ its condition number. To help reader, we recall that in the Truncated Newton scheme we adopt the subscript k indicates the current outer iteration.

2 Preliminaries

We recall that the SYMMBK procedure, that iteratively solves a symmetric linear system basically relies on a couple of relevant tools:

- the Lanczos iterative process for the reduction of the original Newton's equation to a symmetric tridiagonal system;
- the Bunch–Kaufmann decomposition of tridiagonal matrices, through an appropriate pivoting strategy.

Considering the Newton equation (1.3), where the matrix $\nabla^2 f(x_k)$ is possibly indefinite, the first tool allows to transform the symmetric linear system into the system of equalities

$$\begin{cases} T_k y_k = \|\nabla f(x_k)\| e_1 \\ \\ d_k = Q_k y_k, \end{cases} \tag{2.1}$$

being $T_k \in \mathbb{R}^{m\times m}$ symmetric and *tridiagonal*, and $Q_k \in \mathbb{R}^{n\times m}$, where m is the number of iterations performed. In (2.1) the columns of the matrix Q_k are given by the m Lanczos vectors (see also [12]) $q_1, \ldots, q_m$ (with $q_\ell^T q_i = 0$ and $\|q_\ell\|_2 = 1$, being $1 \le \ell \ne i \le m$),

$$Q_k = \begin{pmatrix} q_1 & \vdots & \vdots & q_m \end{pmatrix}. \tag{2.2}$$

We recall that, unlike the method, on indefinite symmetric linear systems the Lanczos process does not suffer for a possible pivot breakdown (see also the seminal book [12]). Given iterate x_k, a relevant property of matrix Q_k is evidenced by the next relation

$$T_k = Q_k^T \nabla^2 f(x_k) Q_k. \tag{2.3}$$

The Bunch–Kaufmann decomposition in SYMMBK allows for an easy factorization of the tridiagonal matrix T_k as in

$$T_k = S_k B_k S_k^T, \tag{2.4}$$

being $S_k \in \mathbb{R}^{m\times m}$ a *block unit lower triangular* matrix, while the matrix $B_k \in \mathbb{R}^{m\times m}$ is *block diagonal*, with blocks of possible dimensions 1×1 or 2×2. By (2.1), after m iterations of the Lanczos process, the vector d_k is available and

- it represents an approximate solution of (2.1);

- it can be used as a search direction within an optimization framework.

Furthermore, in [6] the authors slightly modified the pivot rule within the Bunch–Kaufmann decomposition, so that the resulting vector d_k was provably *Gradient–Related* for the optimization framework where SYMMBK is used.

Our main task here is represented by exploiting SYMMBK procedure, in order to iteratively build an effective negative curvature direction $s_k \in \mathbb{R}^n$ for $f(x)$ at x_k, so that S_k in (2.4) is available almost as by–product and no more than one n–dimensional vector needs to be stored for its computation. This will provide a general matrix–free technique to construct negative curvature directions in large scale settings, where a popular and well renowned tool, namely SYMMBK procedure, is adopted. The vector s_k will be used within an iterative optimization framework to solve (1.1), in order to steer the convergence towards a stationary point $\bar{x}$, where the Hessian matrix $\nabla^2 f(\bar{x})$ is positive semidefinite. As a more specific task, we hereafter technically address the negative curvature direction s_k such that the next conditions are fulfilled:

$$\begin{array}{ll} \text{(i)} & s_k^T \nabla f(x_k) < 0, \text{ for any } s_k \neq 0; \\ \text{(ii)} & s_k^T \nabla^2 f(x_k) s_k < 0, \text{ for any } s_k \neq 0; \\ \text{(iii)} & s_k^T \nabla^2 f(x_k) s_k \to 0 \quad \Longrightarrow \quad \min\left[0, \lambda_{\min}(\nabla^2 f(x_k))\right] \to 0. \end{array} \tag{2.5}$$

Observe that (i) in (2.5) merely imposes that s_k (if any) is a descent direction at x_k for $f(x)$, while (ii) in (2.5) claims that s_k has a nonzero projection on eigenvectors of $\nabla^2 f(x_k)$ associated with negative eigenvalues. Finally, (iii) in (2.5) imposes that, broadly speaking, when we approach a region of convexity for $f(x)$, then s_k eventually approaches a vector in the null space of $\nabla^2 f(x_k)$; (iii) guarantees convergence to second order critical points. Conditions (i)–(iii) in (2.5) allow, in our curvilinear framework, to compute the next iterate x_{k+1} according with a modified Armijo–type linesearch procedure (see e.g. [31]).

3 Assessing the negative curvature direction s_k

We recall that after m iterations of the Lanczos process, when applied in SYMMBK to approximately solving (1.3), we have the matrices Q_k and T_k as in (2.3), while the Bunch–Kaufmann decomposition in SYMMBK iteratively yields the factorization (2.4) of T_k.

Now, given (1.3) and (2.3), let the vector $w \in \mathbb{R}^m$ be an eigenvector of the matrix B_k, associated with a negative eigenvalue λ. Furthermore, assume that computing the vector $y \in \mathbb{R}^k$ such that $S_k^T y = w$ represents a relatively simple task. Then, by (2.3) and (2.4) we obtain

$$\begin{aligned} (Q_k y)^T \nabla^2 f(x_k)(Q_k y) &= y^T \left[Q_k^T \nabla^2 f(x_k) Q_k\right] y \;=\; y^T T_k y \;=\; y^T S_k B_k S_k^T y \\ &= (S_k^T y)^T B_k (S_k^T y) \;=\; w^T B_k w \;=\; \lambda \|w\|_2^2 < 0. \end{aligned}$$

Thus, the vector $Q_k y$ represents a negative curvature direction for the function $f(x)$ at x_k, and in the sequel we are committed to yield a reliable procedure such that the subsequent results are given:

- the efficient (say iterative) computation of the vector $s_k = Q_k y$, exploiting SYMMBK procedure, without storing any matrix;
- the fulfillment of the conditions (i)–(iii) in (2.5) for s_k.

On this purpose we preliminary consider the next result, whose proof can be easily obtained from Lemma 4.3 in [32] and Theorem 3.2 in [23].

Lemma 3.1. *Let us consider the problem (1.1), along with the sequence $\{x_k\}$. Suppose $m = n$ iterations of the Lanczos process are performed by SYMMBK when solving Newton's equation (1.3) at iterate x_k, for a given $k \geq 1$, so that the decompositions*

$$\begin{cases} T_k = Q_k^T \nabla^2 f(x_k) Q_k \\ \\ T_k = S_k B_k S_k^T \end{cases} \tag{3.1}$$

are available. Then, $Q_k \in \mathbb{R}^{n\times n}$ is orthogonal and $T_k \in \mathbb{R}^{n\times n}$ has the same eigenvalues of $\nabla^2 f(x_k)$; moreover, the matrices $S_k \in \mathbb{R}^{n\times n}$ and $B_k \in \mathbb{R}^{n\times n}$ are nonsingular. In addition, if w is a unit eigenvector corresponding to the smallest (negative) eigenvalue λ of B_k, and $\bar{y}$ is a (bounded) solution of the linear system $S_k^T y = w$, then the vector $s_k = Q_k \bar{y}$ is a bounded direction that satisfies (i)–(iii) in (2.5).

Now, observe that the results in Lemma 3.1 assume that the Lanczos process performs exactly n iterations to solve (1.3): this is definitely unaffordable for large n. Hence, we need to generalize the contents in Lemma 3.1 to the case $m < n$. Moreover, we highlight that to compute the vector $\bar{y}$ in Lemma 3.1 we can recur to Lemma 4.3 in [32]. In this regard, with the next lemma we intend to rephrase Lemma 3.1, though obtaining weaker conclusions, being possibly (iii) not fulfilled.

Lemma 3.2. *Let us consider the problem (1.1), along with the sequence $\{x_k\}$. Suppose $m < n$ iterations of the Lanczos process are performed by SYMMBK when solving Newton's equation (1.3) at iterate x_k, for a given $k \geq 1$, so that the decompositions (3.1) are available. Then, we have $Q_k \in \mathbb{R}^{n\times m}$ and $T_k \in \mathbb{R}^{m\times m}$, along with the fact that the matrices $S_k \in \mathbb{R}^{m\times m}$ and $B_k \in \mathbb{R}^{m\times m}$ are nonsingular. In addition, if w is a unit eigenvector corresponding to the smallest (negative) eigenvalue λ of B_k, and $\bar{y}$ is a (bounded) solution of the linear system $S_k^T y = w$, then the vector $s_k = Q_k \bar{y}$ is bounded direction that satisfies (i)–(ii).*

As a further result, Lemma 4.3 in [32] ensures that the outcomes in Lemma 3.1 can be easily generalized when the linear system $S_k^T y = w$ is replaced by

$$S_k^T y = \sum_{1\leq i\leq m \,:\, \lambda_i<0} w_i, \tag{3.2}$$

being w_i the eigenvector of B_k corresponding to a negative eigenvalue λ_i. In this regard, some additional observations require our attention:

- computing all the unit eigenvectors of the matrix B_k may represent in general an expensive task, so that we may limit our analysis to compute an eigenvector associated to (one of) the smallest eigenvalues (namely $\lambda_{\min}$) of B_k, then exploiting Lemma 3.1;
- fully computing all the eigenvectors of B_k does not ensure that an undoubtedly more effective negative curvature direction s_k will be available;
- the computation of $\lambda_{\min}$ is considerably simplified by exploiting a diagonalization of the matrix B_k.

4 Diagonalizing the matrix B_k

As by Section 3 we can observe that a suitable diagonalization of the block diagonal matrix B_k in (3.1) is mandatory in order to simplify the computation of its eigenpairs. In particular, let

- $D_k \in \mathbb{R}^{m\times m}$ be a diagonal matrix, with $D_k = diag\{\lambda_1, \ldots, \lambda_m\}$ and where $\lambda_1, \ldots, \lambda_m$ are all the eigenvalues (possibly not all distinct) of B_k;
- $X_k \in \mathbb{R}^{m\times m}$ be an orthogonal matrix, such that its columns correspond to the eigenvectors of B_k associated to the eigenvalues $\lambda_1, \ldots, \lambda_m$;

then we have

$$D_k = X_k^T B_k X_k \qquad \Longleftrightarrow \qquad B_k = X_k D_k X_k^T. \tag{4.1}$$

Hence, since B_k is a block diagonal matrix (with 1×1 and 2×2 blocks), then also X_k will be a block diagonal matrix with blocks of dimension at most 2×2. In particular, for any 1×1 diagonal block $B^{(i\ i)}$ (2×2 diagonal block $B^{(i\ i+1)}$) of matrix B_k we will have the corresponding 1×1 block $\chi^{(i\ i)}$ (2×2 block $\chi^{(i\ i+1)}$) of matrix X_k, corresponding to the number 1 (the two eigenvectors of

the sub–matrix $B^{(i\ i+1)}$). Thus, if B_k is given by all 1×1 diagonal blocks apart from the block $B^{(i\ i+1)}$, i.e.:

$$B_k = \begin{pmatrix} * & & & & & \\ & * & & & & \\ & & B^{(i\ i+1)} & & & \\ & & & & * & \\ & & & & & * \end{pmatrix},$$

then the matrix X_k will correspondingly be given by

$$X_k = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \chi^{(i\ i+1)} & & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix}, \tag{4.2}$$

where $\chi^{(i\ i+1)} = (v^i \vdots v^{i+1}) \in \mathbb{R}^{2\times 2}$ and v^i, v^{i+1} are the unit eigenvectors of $B^{(i\ i+1)}$. This last example shows that the computation of the eigenpairs of B_k is relatively easy when it is a 2×2 block diagonal matrix. Hereafter we will indicate with $\lambda_\ell = \lambda_{\min}[B_k]$ the smallest (negative) eigenvalue of B_k, and z_ℓ will be its corresponding unit eigenvector. Thus, ℓ will be used to denote the row (column) index corresponding to the smallest (negative) diagonal entry of matrix D_k. By (2.4) and (4.1) we immediately have

$$T_k = S_k X_k D_k X_k^T S_k^T = W_k D_k W_k^T, \tag{4.3}$$

where $W_k \stackrel{\text{def}}{=} S_k X_k \in \mathbb{R}^{m\times m}$. Now, in order to exploit the theory indicated in Lemma 3.1 and Lemma 3.2, we need to compute the eigenvector corresponding to the smallest eigenvalue of T_k, so that a negative curvature direction for $f(x)$ at x_k can be computed, fulfilling (i)–(ii) in (2.5) and possibly (iii). On this purpose, we have to do nothing else but replacing in (3.2) the matrix S_k with the matrix W_k in (4.3). Hence, on the overall the computation of the negative curvature direction s_k requires solving the linear system

$$W_k^T y = z_\ell. \tag{4.4}$$

4.1 The solution of (4.4) by backtracking

By simple inspection of (4.4), after exploiting the block structure of the matrix W_k we can realize that, following the guidelines in [15] and [23], a straightforward backtracking algorithm allows the computation of the solution $\bar{y} \in \mathbb{R}^m$ for (4.4). Indeed, recalling that X_k is orthogonal, the linear system (4.4) can be re-written as $S_k^T y = w_\ell$, where $w_\ell = X_k z_\ell$. Hence, setting

$$\begin{cases} [S_k]_{v,j} = s_{v,j}, & v = 1,\dots,m\ ,\ j = 1,\dots,m \\ [X_k]_{v,j} = x_{v,j}, & v = 1,\dots,m\ ,\ j = 1,\dots,m \\ z_\ell = \begin{pmatrix} z_\ell^1 \\ \vdots \\ z_\ell^k \end{pmatrix} \end{cases}$$

to compute the vector $\bar{y}$ the next system of m equations must be solved:

$$\sum_{j=1}^{m} y_j s_{v,j} = \sum_{j=1}^{m} z_\ell^j x_{v,j} \quad v = 1,\dots,m. \tag{4.5}$$

Since S_k is unit lower triangular, then $s_{j,j} = 1$, for $j = 1,\dots,m$, and $s_{v,j} = 0$, for any $j < v$. Furthermore, recalling that the matrix B_k is (at most) 2×2 block diagonal, then the vector z_ℓ has

all zero entries but at most two (consecutive) non–zeroes. Hence, exploiting the structure of the S_k^T matrix, (4.5) can be solved recurring to a backward substitution algorithm, which yields to the following solution:

$$\begin{cases} \bar{y}_m = z_\ell^m x_{m,m} \\ \bar{y}_{m-1} = z_\ell^{m-1} x_{m-1,m-1} + z_\ell^m x_{m-1,m} - \bar{y}_m s_{m-1,m} \\ \bar{y}_j = z_\ell^{j-1} x_{j,j-1} + z_\ell^j x_{j,j} + z_\ell^{j+1} x_{j,j+1} - \bar{y}_{j+1} s_{j,j+1} - \bar{y}_{j+2} s_{j,j+2}, \quad j = 1, \ldots, m-2 \end{cases} \tag{4.6}$$

Now, suppose that $\bar{y}$ is the vector resulting from (4.6); by Lemmas 3.1 and 3.2 it is possible to compute the negative curvature direction for the function $f(x)$ at x_k as $s_k = Q_k \bar{y}$. Recalling that $Q_k = \begin{pmatrix} q_1 & \vdots & \vdots & q_m \end{pmatrix}$, since $\bar{y}_{\ell+2} = \bar{y}_{\ell+3} = \cdots = \bar{y}_m = 0$, we have

$$s_k = \sum_{j=1}^{m} q_j \bar{y}_j = \sum_{j=1}^{\ell+1} q_j \bar{y}_j.$$

The last computation immediately reveals that the storage of the vectors $q_1, \ldots, q_m$ is mandatory, inasmuch as the index ℓ will be known only at the end of the m-th Lanczos iteration. This *makes the iterative backtracking procedure to solve (4.4) impracticable for large scale problems*, justifying an alternative method proposed in the next sections.

5 A better exploitation of SYMMBK to compute negative curvature directions for $f(x)$

In this section, following the guidelines in [11, Section 5.2] for finding conjugate directions from an orthonormal Krylov basis, we propose to better exploit SYMMBK to generate $\nabla^2 f(x_k)$–conjugate vectors to be used for computing negative curvature directions.

According with Lemma 3.1 and Lemma 3.2, the matrix T_k in (3.1) can be decomposed as $T_k = S_k B_k S_k^T$, being B_k a block diagonal matrix, with 1×1 or 2×2 blocks. After diagonalizing B_k as in $B_k = X_k D_k X_k^T$, with X_k orthogonal, we obtain (4.3) where

$$W_k \stackrel{\text{def}}{=} S_k X_k = \begin{pmatrix} W^{(1\ 1)} & & & & & \\ W^{(2\ 1)} & W^{(2\ 2)} & & & & \\ & \cdot & \cdot & & & \\ & & \cdot & W^{(j-1\ j-1)} & & \\ & & & W^{(j\ j-1)} & W^{(j\ j)} & \end{pmatrix}, \qquad j \geq 1. \tag{5.1}$$

Here the sizes (both rows and columns) of the sub–diagonal blocks $W^{(i+1\ i)}$, $i = 1, \ldots, j-1$, depend on the sizes of the diagonal blocks[1]. Moreover, recalling that S_k is block unit lower triangular, then the diagonal blocks $W^{(i\ i)} \equiv \chi^{(i\ i+1)}$ are orthogonal (see also (4.2)).
Now, by combining (2.4), (3.1) and (4.3) we can compute a set of conjugate directions for the Hessian matrix $\nabla^2 f(x_k)$, being indeed

$$T_k = W_k D_k W_k^T = Q_k^T \nabla^2 f(x_k) Q_k, \tag{5.2}$$

so that

$$D_k = W_k^{-1} Q_k^T \nabla^2 f(x_k) Q_k W_k^{-T} = G_k^T \nabla^2 f(x_k) G_k, \tag{5.3}$$

where

$$G_k \stackrel{\text{def}}{=} Q_k W_k^{-T} \in \mathbb{R}^{n \times m}. \tag{5.4}$$

Since D_k is a diagonal matrix, by (5.3)–(5.4) the columns of G_k yield a set of m linearly independent (see also Proposition 2.1 of [19] and [20]) $\nabla^2 f(x_k)$–conjugate directions which span the Krylov

[1] E.g., in case $W^{(2\ 2)} \in \mathbb{R}^{1 \times 1}$ and $W^{(3\ 3)} \in \mathbb{R}^{2 \times 2}$, then we will have $W^{(3\ 2)} \in \mathbb{R}^{2 \times 1}$.

subspace $\mathbb{K}(\nabla^2 f(x_k), q_1, m)$. To efficiently compute the matrix G_k, let us define (for the sake of simplicity we drop the dependency on the index k)

$$G_k = (G^1 \; G^2 \; \ldots \; G^{j-1} \; G^j), \tag{5.5}$$

being G^i an $n \times 1$ or an $n \times 2$ sub–matrix, for any $1 \leq i \leq j$. Thus, we can now re–write equation (5.4) as

$$G_k W_k^T = Q_k \stackrel{\text{def}}{=} (Q^1 \; Q^2 \; \ldots \; Q^{j-1} \; Q^j), \tag{5.6}$$

where each Q^i, $1 \leq i \leq j$, represents an $n \times 1$ or an $n \times 2$ matrix whose columns are given by Lanczos vectors, in accordance with the structure of G_k in (5.5). Hence, using the expression (5.1) for matrix W_k, as well as the orthogonality of the blocks $W^{(i\;i)}$, $1 \leq i \leq j$, we obtain from (5.5) and (5.6)

$$G^i = \left[Q^i - G^{i-1} \left(W^{(i\;i-1)} \right)^T \right] W^{(i\;i)}, \qquad 1 < i \leq j, \tag{5.7}$$

with $G^1 = Q^1 W^{(1\;1)}$. Hence, we can efficiently and iteratively compute the blocks $\{G^i\}$, whose columns represent mutually $\nabla^2 f(x_k)$–conjugate directions, as long as the quantities $\{Q^j\}$, $\left\{W^{(i\;i-1)}\right\}$ and $\left\{W^{(i\;i)}\right\}$ are available.

6 Theoretical results for negative curvature directions computation

Relations (5.7) indicate how to fully iteratively compute the matrix G_k in (5.4); moreover, (5.3) indicates that the columns of G_k represent indeed a set of $\nabla^2 f(x_k)$–conjugate vectors. Now, let us define the vector

$$z = \sum_{1 \leq j \leq m \,:\, \mu_j < 0} a_j G_{(j)}, \tag{6.1}$$

where μ_j represents the j-th eigenvalue of the diagonal matrix D_k, while $G_{(j)}$ is[2] the j-th column of G_k and $a_j \in \mathbb{R}$, $j \in \{1, \ldots, m\}$, is such that

$$\sum_{1 \leq j \leq m \,:\, \mu_j < 0} a_j^2 \mu_j \;\leq\; \lambda_{\min}\left[D_k\right] \left[\min_{1 \leq j \leq m \,:\, \mu_j < 0} a_j^2 \right]. \tag{6.2}$$

Then, in the theoretical results which follow, in order to prove that our final negative curvature direction fulfills (i)–(iii) in (2.5), we set

$$s_k = \frac{z}{\|z\|} \tag{6.3}$$

and prove the next results. Note that the use of the normalized direction s_k, in place of z, is only for theoretical purposes (see also [32], where a similar approach is used assuming that both the search directions in (1.4) are bounded).

Proposition 6.1. *Given the function $f : \mathbb{R}^n \to \mathbb{R}$, with $f \in C^2(\mathbb{R}^n)$, let us consider the sequence $\{x_k\}$ of approximate solutions to problem (1.1). Assume at iterate x_k the Hessian matrix $\nabla^2 f(x_k)$ has at least one negative eigenvalue. Let $G_k \in \mathbb{R}^{n \times n}$ be the matrix in (5.4) after n inner iterations of the Lanczos process, and let $\kappa(G_k)$ denote the condition number of G_k. Assume $\{a_j\}$ (with $1 \leq j \leq n$) is a set of real values satisfying (6.2), and $\{\mu_j\}$ is the set of eigenvalues of the diagonal matrix D_k. Then*

$$s_k^T \nabla^2 f(x_k) s_k \;\leq\; \frac{1}{N \cdot [\kappa(G_k)]^2} \left[\frac{\min\limits_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2}{\max\limits_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2} \right] \lambda_{\min}\left[\nabla^2 f(x_k)\right], \tag{6.4}$$

where $N \geq 1$ is the number of negative eigenvalues of $\nabla^2 f(x_k)$, and $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$ is its smallest eigenvalue.

[2] Again, for simplicity we drop the dependency of $G_{(j)}$ on the index k.

Proof. Let us consider the unit eigenvector $u_{\min}$ of $\nabla^2 f(x_k)$ corresponding to the smallest eigenvalue $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$. Hence, $u_{\min}^T \nabla^2 f(x_k) u_{\min} = \lambda_{\min}\left[\nabla^2 f(x_k)\right]$ and therefore by (5.3)

$$\lambda_{\min}\left[\nabla^2 f(x_k)\right] = u_{\min}^T G_k^{-T} D_k G_k^{-1} u_{\min} = w^T D_k w \geq \lambda_{\min}\left[D_k\right] \|w\|^2,$$

where $w = G_k^{-1} u_{\min}$. Now, since $\|w\| \leq \left\|G_k^{-1}\right\| \|u_{\min}\| = \left\|G_k^{-1}\right\|$ we have

$$\lambda_{\min}\left[\nabla^2 f(x_k)\right] \geq \lambda_{\min}\left[D_k\right] \|w\|^2 \geq \lambda_{\min}\left[D_k\right] \left\|G_k^{-1}\right\|^2. \tag{6.5}$$

From (6.1) and recalling that e_j is the j-th unit vector, we have

$$z \;=\; G_k \left[\sum_{1 \leq j \leq n \,:\, \mu_j < 0} a_j e_j \right] \tag{6.6}$$

so that we also obtain from (6.2)

$$\begin{aligned} z^T \nabla^2 f(x_k) z \;&=\; z^T G_k^{-T} D_k G_k^{-1} z \;=\; \left(\sum_{1 \leq j \leq n \,:\, \mu_j < 0} a_j e_j \right)^T D_k \left(\sum_{1 \leq j \leq n \,:\, \mu_j < 0} a_j e_j \right) \\ &=\; \sum_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2 \mu_j \;\leq\; \lambda_{\min}\left[D_k\right] \left[\min_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2 \right]. \end{aligned}$$

Hence, by (6.5)

$$\begin{aligned} z^T \nabla^2 f(x_k) z \left\|G_k^{-1}\right\|^2 \;&\leq\; \lambda_{\min}\left[D_k\right] \left[\min_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2 \right] \left\|G_k^{-1}\right\|^2 \\ &\leq\; \left[\min_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2 \right] \lambda_{\min}\left[\nabla^2 f(x_k)\right] \;<\; 0. \end{aligned}$$

Moreover, since

$$\|z\|^2 \leq \|G_k\|^2 \left\| \sum_{1 \leq j \leq n \,:\, \mu_j < 0} a_j e_j \right\|^2 \leq N \|G_k\|^2 \left[\max_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2 \right]$$

then by (6.6)

$$\begin{aligned} 0 \;&>\; \lambda_{\min}\left[\nabla^2 f(x_k)\right] \;\geq\; \frac{z^T \nabla^2 f(x_k) z \|G_k^{-1}\|^2}{\left[\min_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2 \right]} \\ &\geq\; N \|G_k^{-1}\|^2 \|G_k\|^2 \left[\frac{\max_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2}{\min_{1 \leq j \leq n \,:\, \mu_j < 0} a_j^2} \right] \frac{z^T \nabla^2 f(x_k) z}{\|z\|^2}, \end{aligned} \tag{6.7}$$

so that, recalling (6.3) condition (6.4) holds. □

The last proposition evidences that in case the Lanczos process is able to perform exactly n inner iterations within SYMMBK procedure, then a unit negative curvature direction s_k satisfying (i)–(iii) in (2.5) can be easily available as by–product from (5.4), (6.1) and (6.3).

We also remark that the computation of the vector z in (6.1) does not require the storage of more than one additional vector, i.e. the current sum of the contributions $\{a_j G_{(j)}\}$ up to the m-th inner iteration. The latter result, to these authors' knowledge, represents the first example in the literature of a so cheap computation of the vector s_k fulfilling (i)–(iii) in (2.5).

Observe that the fulfillment of condition (6.2) is a preliminary requirement for the construction of the negative curvature direction s_k to be used within Proposition 6.1. Hence, in the next result we show how to iteratively properly select the coefficients $\{a_j\}$ in (6.1) so that (6.2) holds.

Lemma 6.2. *Let us consider the sequence of the real coefficients* $\{a_j\}$ *in* (6.1) *and* (6.2). *Let us define the real quantities (for any* $1 \le h \le m$*):*

$$\begin{cases} \lambda_{\min}^{(h-1)} = \min\limits_{1\le j\le h-1\,:\,\mu_j<0} \{\mu_j\}, \\ C^{(h-1)} = \min\limits_{1\le j\le h-1\,:\,\mu_j<0} \{a_j^2\}, \\ A^{(h-1)} = \sum\limits_{1\le j\le h-1\,:\,\mu_j<0} a_j^2\mu_j. \end{cases} \tag{6.8}$$

Condition (6.2) *is fulfilled provided that for any* $j \ge 2$, *when* $\mu_j > \lambda_{\min}^{(j-1)}$ *then we set*

$$a_j^2 \le \min\left\{C^{(j-1)}, \frac{-A^{(j-1)}}{\mu_j - \lambda_{\min}^{(j-1)}}\right\}, \qquad j \in \{1,\dots,m\}. \tag{6.9}$$

Proof. The proof proceeds by induction. Relation (6.2) clearly holds when $j = 1$, with no specific assumption on a_1. Then, we assume that by (6.8)–(6.9) it holds for $j-1$, and we prove the result for the index j. Observe that for any j in (6.1)-(6.2) we have $\mu_j < 0$; moreover, the condition

$$A^{(j-1)} \le \lambda_{\min}^{(j-1)} C^{(j-1)} \tag{6.10}$$

is satisfied by inductive hypothesis. Therefore, for the index j we analyze the conditions which guarantee that the inequality (6.2), i.e.

$$A^{(j-1)} + a_j^2\mu_j \le \min\left\{\lambda_{\min}^{(j-1)}\ ,\ \mu_j\right\} \min\left\{C^{(j-1)}, a_j^2\right\} \tag{6.11}$$

is satisfied. This yields the next two cases:

- if $\mu_j < \lambda_{\min}^{(j-1)}$, then (6.11) yields $A^{(j-1)} + a_j^2\mu_j \le \mu_j \min\left\{C^{(j-1)}, a_j^2\right\}$, so that in case $C^{(j-1)} < a_j^2$, since $C^{(j-1)} \ge 0$, we obtain

  $$A^{(j-1)} + a_j^2\mu_j \le \mu_j C^{(j-1)} \le \lambda_{\min}^{(j-1)} C^{(j-1)}$$

  which is always fulfilled recalling that $A^{(j-1)} + a_j^2\mu_j \le A^{(j-1)}$ and considering relation (6.10). Conversely, in case $C^{(j-1)} \ge a_j^2$ we obtain

  $$A^{(j-1)} + a_j^2\mu_j \le \mu_j a_j^2$$

  which is again always fulfilled inasmuch as $A^{(j-1)} < 0$;

- if $\mu_j \ge \lambda_{\min}^{(j-1)}$, then (6.11) yields $A^{(j-1)} + a_j^2\mu_j \le \lambda_{\min}^{(j-1)} \min\left\{C^{(j-1)}, a_j^2\right\}$, where again we distinguish the case $C^{(j-1)} < a_j^2$, for which by (6.10)

  $$A^{(j-1)} + a_j^2\mu_j \le A^{(j-1)} \le \lambda_{\min}^{(j-1)} C^{(j-1)},$$

  that is always fulfilled, and the case $C^{(j-1)} \ge a_j^2$ which yields

  $$\left(\mu_j - \lambda_{\min}^{(j-1)}\right) a_j^2 \le -A^{(j-1)},$$

  that holds by the condition (6.9) on the coefficient a_j.

□

Observation 6.3. We remark that the procedure to update the coefficients $\{a_j\}$ in Lemma 6.2 does not require the storage of any vector/matrix, so that the computation of the negative curvature direction s_k can be iteratively carried on in large scale settings.

Observation 6.4. Relation (6.4) reveals that the effectiveness of the negative curvature direction s_k requires the boundedness of $\kappa(G_k)$. Nevertheless, in case the quantity $\|G_k^{-1}\|$ is itself bounded, by (6.7) of Proposition 6.1 we can alternatively conclude that also the vector z (and not only the vector s_k) satisfies (iii) in (2.5). Thus, z could be used as an alternative negative curvature direction, too, in place of s_k. In this regard, by relation (2.18) in [6], where the matrix W_k plays the role of the matrix G_k in the current paper, the proper choice of the parameter ω in [6] can ensure that when applying the Bunch–Kaufmann decomposition within SYMMBK the quantity $\|G_k^{-1}\|$ is indeed bounded. This partially fills the gap between the current paper and [6], where a proper choice of the parameter ω was needed in order to compute a gradient–related direction by SYMMBK. In this regard, those values of ω in [6] are also worth in the current paper for assessing a negative curvature direction which has possibly not a unit norm (see also Section 6.1 for additional considerations). In the numerical experimentation we adopt the negative curvature direction defined in (6.1), to cope with the well known drawbacks related to careless combining the Newton type direction d_k and a negative curvature direction, when they show a possible mismatch of their norms. Indeed, generally d_k is not expected to have unit norm (in this regard see the detailed discussion reported in Section 9).

Unfortunately, the assumption in Proposition 6.1 that n Lanczos iterations are performed is far from being realistic when n is large, so that a possible extension of the results in Proposition 6.1 would be welcome for practical applications. In this regard, let us define

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \stackrel{\text{def}}{=} \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{[G_k\nu]^T\,\nabla^2 f(x_k)\,[G_k\nu]}{\|G_k\nu\|^2}. \tag{6.12}$$

Observe that $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ represents the smallest value of the Rayleigh quotient for $\nabla^2 f(x_k)$, where the vector $G_k\nu$ spans the Krylov subspace $\mathbb{K}(\nabla^2 f(x_k), q_1, m)$. Hence, $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ can be regarded to some extent as an *approximation from above* (on the Krylov subspace $\mathbb{K}(\nabla^2 f(x_k), q_1, m)$) of $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$, as stated in the next lemma.

Lemma 6.5. *Let us consider the matrix G_k in (5.4), the quantity $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ in (6.12) and the eigenvalue $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$ of $\nabla^2 f(x_k)$. We have for $1 \le h \le n$*

$$\lambda_{\min}\left[\nabla^2 f(x_k)\right] = \lambda_{\min}^{(n)}\left[\nabla^2 f(x_k)\right] \le \cdots \le \lambda_{\min}^{(h)}\left[\nabla^2 f(x_k)\right] \le \cdots \le \lambda_{\min}^{(1)}\left[\nabla^2 f(x_k)\right].$$

Moreover, if the columns of the matrix $G_k = \left(G_{(1)}\cdots G_{(m)}\right) \in \mathbb{R}^{n\times m}$ in (6.12) are $\nabla^2 f(x_k)$–conjugate vectors, then

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] = \min_{1\le j\le m} \frac{G_{(j)}^T\nabla^2 f(x_k)G_{(j)}}{\left\|G_{(j)}\right\|^2}.$$

Proof. The result follows immediately from relation (6.12) and the inequality

$$\min_{\nu\in\mathbb{R}^h,\ \|\nu\|=1} \frac{[G_k\nu]^T\,\nabla^2 f(x_k)\,[G_k\nu]}{\|G_k\nu\|^2} \ge \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{[G_k\nu]^T\,\nabla^2 f(x_k)\,[G_k\nu]}{\|G_k\nu\|^2},$$

for any $1 \le h \le m$. Moreover, by the conjugacy of the columns of G_k we have

$$\begin{aligned}\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] &= \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{[G_k\nu]^T\,\nabla^2 f(x_k)\,[G_k\nu]}{\|G_k\nu\|^2} \\ &= \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{\nu^T G_k^T\nabla^2 f(x_k)G_k\nu}{\|G_k\nu\|^2} = \min_{1\le j\le m} \frac{G_{(j)}^T\nabla^2 f(x_k)G_{(j)}}{\left\|G_{(j)}\right\|^2}.\end{aligned}$$

and this completes the proof. □

Using the above considerations and Lemma 6.5, the next generalization of Proposition 6.1 can be given, when $m < n$.

Proposition 6.6. *Given the function $f : \mathbb{R}^n \to \mathbb{R}$, with $f \in C^2(\mathbb{R}^n)$, let us consider the sequence $\{x_k\}$ of approximate solutions to problem (1.1). Assume at iterate x_k the Hessian matrix $\nabla^2 f(x_k)$ has at least one negative eigenvalue. Let $G_k \in \mathbb{R}^{n\times m}$ be the matrix in (5.4) after $m < n$ inner iterations, and let σ_{min} [σ_{max}] be its smallest [largest] singular value. Assume $\{a_j\}$, with $1 \le j \le n$, is a set of real values satisfying (6.2). Then*

$$s_k^T \nabla^2 f(x_k) s_k \;\le\; \frac{1}{m}\left(\frac{\sigma_{min}}{\sigma_{max}}\right)^2 \left[\frac{\min\limits_{1\le j\le m \,:\, \mu_j<0} a_j^2}{\max\limits_{1\le j\le m \,:\, \mu_j<0} a_j^2}\right] \lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right], \tag{6.13}$$

where $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ is defined in (6.12).

Proof. The proof follows the guidelines of Proposition 6.1, so that we will focus only on their differences. In particular we have from (5.3)

$$\nu^T G_k^T \nabla^2 f(x_k) G_k \nu \;=\; \nu^T D_k \nu \;\ge\; \lambda_{\min}[D_k], \qquad \forall \nu \in \mathbb{R}^m,\ \|\nu\| = 1, \tag{6.14}$$

so that choosing $\nu = \bar\nu$, with $\bar\nu \in \mathbb{R}^m$, in (6.14) we obtain (see also (6.12))

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \;=\; \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{\left[G_k\nu\right]^T \nabla^2 f(x_k)\left[G_k\nu\right]}{\left\|G_k\nu\right\|^2} \;=\; \frac{\bar\nu^T G_k^T \nabla^2 f(x_k) G_k \bar\nu}{\|G_k\bar\nu\|^2}. \tag{6.15}$$

Now, we recall that by the *singular value decomposition* of $G_k = U\Sigma V^T$, with $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{m\times m}$ and $\Sigma \in \mathbb{R}^{n\times m}$, there exist m singular values[3] $\sigma_1, \ldots, \sigma_m$ such that

$$0 < \sigma_{\min} = \sigma_1 \le \cdots \le \sigma_j \le \cdots \le \sigma_m = \sigma_{\max}$$

and

$$G_k V = U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \\ \hline & \emptyset_{n-m} & \end{bmatrix}, \qquad \Longleftrightarrow \qquad G_k v_j = \sigma_j u_j, \quad j = 1, \ldots, m,$$

being $m \le n$, $U = (u_1 \cdots u_m u_{m+1} \cdots u_n)$ and $V = (v_1 \cdots v_m)$. Hence, $\|G_k v_j\| = \sigma_j \|u_j\|$, for $j = 1, \ldots, m$. Moreover, we have from this last relations, along with (5.3) and (6.15)

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \;=\; \frac{\bar\nu^T D_k \bar\nu}{\|G_k\bar\nu\|^2} \;\ge\; \frac{\lambda_{\min}[D_k]}{\|G_k\bar\nu\|^2} \;\ge\; \frac{\lambda_{\min}[D_k]}{\min\limits_{\|\nu\|=1}\|G_k\nu\|^2} = \frac{\lambda_{\min}[D_k]}{\sigma_{\min}^2}, \tag{6.16}$$

where the last inequality holds recalling that $\lambda_{\min}[D_k] < 0$. Furthermore, being $e_j \in \mathbb{R}^n$ the j-th unit vector we have

$$z \;=\; \sum_{1\le j\le m \,:\, \mu_j<0} a_j G_{(j)} \;=\; G_k \sum_{1\le j\le m \,:\, \mu_j<0} a_j e_j,$$

and from (5.3) and (6.2) we have

$$\begin{aligned} z^T \nabla^2 f(x_k) z \;&=\; \left(\sum_{1\le j\le m \,:\, \mu_j<0} a_j e_j\right)^T G_k^T \nabla^2 f(x_k) G_k \left(\sum_{1\le j\le m \,:\, \mu_j<0} a_j e_j\right) \\ &=\; \left(\sum_{1\le j\le m \,:\, \mu_j<0} a_j e_j\right)^T D_k \left(\sum_{1\le j\le m \,:\, \mu_j<0} a_j e_j\right) \\ &=\; \sum_{1\le j\le m \,:\, \mu_j<0} a_j^2 \mu_j \;\le\; \lambda_{\min}\left[D_k\right]\left[\min_{1\le j\le m \,:\, \mu_j<0} a_j^2\right]. \end{aligned}$$

[3]Observe that since G_k has rank m then its m singular values are positive.

Hence, from (6.16) and the last inequality it is

$$\begin{aligned}\frac{z^T\nabla^2 f(x_k)z}{\sigma_{min}^2} &\leq \frac{\lambda_{\min}\left[D_k\right]\left[\min\limits_{1\leq j\leq m\,:\,\mu<0} a_j^2\right]}{\sigma_{min}^2}\\ &\leq \lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]\left[\min_{1\leq j\leq m\,:\,\mu<0} a_j^2\right] \;<\; 0.\end{aligned}$$

Moreover, by (6.1) we have

$$\begin{aligned}\|z\|^2 &\leq \|G_k\|^2\left\|\sum_{1\leq j\leq m\,:\,\mu_j<0} a_j e_j\right\|^2\\ &\leq \max_{\|\nu\|=1}\|G_k\nu\|^2\left[\sum_{1\leq j\leq m\,:\,\mu_j<0} a_j^2\right] \leq \sigma_{\max}^2\left[\sum_{1\leq j\leq m\,:\,\mu_j<0} a_j^2\right]\\ &\leq m\cdot\sigma_{\max}^2\left[\max_{1\leq j\leq m\,:\,\mu_j<0} a_j^2\right].\end{aligned}$$

Thus, the last couple of relations yield

$$\begin{aligned}\frac{z^T\nabla^2 f(x_k)z}{\|z\|^2} &\leq \frac{\sigma_{\min}^2}{\|z\|^2}\cdot\left[\min_{1\leq j\leq m\,:\,\mu_j<0} a_j^2\right]\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]\\ &\leq \frac{\sigma_{min}^2}{m\cdot\sigma_{max}^2}\cdot\frac{\left[\min\limits_{1\leq j\leq m\,:\,\mu_j<0} a_j^2\right]}{\left[\max\limits_{1\leq j\leq m\,:\,\mu_j<0} a_j^2\right]}\cdot\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \;<\; 0,\end{aligned}$$

so that (6.13) holds. □

Observation 6.7. There is not difficulty to conclude that the contents in Lemma 6.2 and Observations 6.3–6.4 could be immediately extended to the results of Proposition 6.6, so that the iterative computation of a negative curvature direction can be fully exploited also when less than n inner iterations are performed at the k-th outer iteration of the Truncated Newton method.

Observation 6.8. Note that the presence of the sequence $\{a_j\}$ in Propositions 6.1 and 6.6 aims at giving generality in (6.1), when computing the negative curvature direction (the use of $\{a_j\}$ will be clarified in the next section). In particular, on structured Hessian problems this may give an indication about those vectors $G_{(j)}$ to privilege for the computation of the negative curvature direction s_k. Conversely, the choice $a_j = 1$, for any $j \geq 1$, fulfills (6.2) and represents the most obvious one, since it also contributes to tighten the bounds in (6.4) and (6.13). Nevertheless, the choice $a_{\hat{\jmath}} = 1$, where $\hat{\jmath} = \arg\min_j\left\{G_{(j)}^T\nabla^2 f(x_k)G_{(j)}/\|G_{(j)}\|^2\right\}$, with $a_j = 0$ for any $j \neq \hat{\jmath}$, again satisfies (6.2) and allows to minimize the gap between $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$ and $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ (see Lemma 6.5). Moreover, it basically encompasses also the choice for the negative curvature direction adopted in [21].

6.1 Issues on the choice of the sequence $\{a_j\}$

Observe that the choice of the sequence $\{a_j\}$ in (6.1) is only claimed to fulfill (6.2), in order to provide a negative curvature direction $s_k = z/\|z\|_2$ satisfying either Proposition 6.1 or Proposition 6.6. The Observations 6.4 and 6.8 give some additional hints on the choice of the sequence $\{a_j\}$, however further considerations suggest some restrictions to the last choice. Indeed, let us consider a Truncated Newton method for solving (1.1); in case s_k were either used in a curvilinear

framework (i.e. combined with a Newton–type direction) or as a stand alone search direction, then it is expected to be at least a *descent direction* and possibly a *gradient–related* one.

Then, recalling [6, Section 3.1], we have sufficient conditions for the choice of the test within the Bunch–Kaufmann decomposition, so that the approximate solution $\bar{d}_k$ to (2.1) is gradient–related. Furthermore, by Propositions 3.1 and 3.2 of [6] it is possible to show that also the vector $a_j G_{(j)}$ in (6.1) is gradient–related (see also Observation 6.4), provided that it is chosen exactly following the guidelines for computing the vectors uu and vv in the Reverse–Scheme of [6] (see also Observation 6.3). Indeed, broadly speaking, the vector $a_j G_{(j)}$ in (6.1) is equivalently obtained by reducing the Lanczos process / Bunch–Kaufmann procedure used in SYMMBK to a CG method. Thus, the proper choice of the coefficient a_j makes the vector $a_j G_{(j)}$ of descent for $f(x)$. In this regard, note that the parameter ω which affects the test within the Bunch–Kaufmann decomposition in [6], yet affects also the computation of the negative curvature direction s_k, through the coefficients $\{a_j\}$. Therefore, on the overall the vector $a_j G_{(j)}$ is gradient–related, provided that (recalling the Reverse–Scheme in [6]) the coefficient a_j is chosen so that $a_j G_{(j)} \equiv uu$ or $a_j G_{(j)} \equiv vv$.

On summary, the next final remarks on the choice of the sequence $\{a_j\}$ in (6.1) hold:

- as long as the search direction $a_j G_{(j)}$ is selected as $a_j G_{(j)} \equiv uu$ or $a_j G_{(j)} \equiv vv$, where uu and vv are computed by the Reverse–Scheme in [6], then the vector s_k will be gradient–related, too;
- as long as a_j is selected so that (6.2) holds, then the negative curvature direction s_k will also fulfill (6.4) and (6.13).

The last two items ensure that under mild assumptions on the sequence $\{a_j\}$ in (6.1), the vector s_k can be safely and fully used within Truncated Newton methods, to guarantee convergence towards stationary points satisfying also second order necessary optimality conditions.

Finally, as suggested in Observation 6.4, in case $\|G_k^{-1}\|$ is bounded, then also the vector z in (6.1) (with the positions (6.2)) may be considered an alternative negative curvature direction fulfilling (i)–(iii) in (2.5). This further viable choice will be better detailed in Section 9.

7 An alternative proposal

Here we propose additional theoretical results which may yield guidelines for the computation of negative curvature directions when solving (1.1). In particular, we rephrase the outcomes in Proposition 6.1 and Proposition 6.6, so that we exploit (5.2) in place of (5.3). In this regard Proposition 6.1 can be reformulated in the following way.

Proposition 7.1. *Given the function $f : \mathbb{R}^n \to \mathbb{R}$, with $f \in C^2(\mathbb{R}^n)$, let us consider the sequence $\{x_k\}$ of approximate solutions to problem (1.1). Assume at iterate x_k the Hessian matrix $\nabla^2 f(x_k)$ has at least one negative eigenvalue. Let $Q_k, T_k \in \mathbb{R}^{n\times n}$ be the matrices in (5.2), after n inner iterations the Lanczos process. Assume a_j, with $1 \le j \le n$, is a set of real values satisfying (6.2). Then*

$$s_k^T \nabla^2 f(x_k) s_k \le \frac{1}{N} \left[\frac{\min\limits_{1\le j\le n \,:\, \mu_j<0} a_j^2}{\max\limits_{1\le j\le n \,:\, \mu_j<0} a_j^2} \right] \lambda_{\min}\left[\nabla^2 f(x_k)\right], \tag{7.1}$$

where $N \ge 1$ is the number of negative eigenvalues of $\nabla^2 f(x_k)$, and $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$ is its smallest eigenvalue.

Proof. The proof basically follows the same guidelines of the one in Proposition 6.1, after recalling that Q_k is an orthogonal matrix, and replacing G_k by Q_k, G_k^{-1} by Q_k^T, D_k by T_k and $G_k^{-T} D_k G_k^{-1}$ by $Q_k T_k Q_k^T$. □

Similarly, as regards Proposition 6.6 it can be reformulated in the following way.

Proposition 7.2. *Given the function $f : \mathbb{R}^n \to \mathbb{R}$, with $f \in C^2(\mathbb{R}^n)$, let us consider the sequence $\{x_k\}$ of approximate solutions to problem (1.1). Assume at iterate x_k the Hessian matrix $\nabla^2 f(x_k)$ has at least one negative eigenvalue. Let $Q_k \in \mathbb{R}^{n\times m}$ and $T_k \in \mathbb{R}^{m\times m}$ be the matrices in (5.2), after $m < n$ inner iterations. Assume $\{a_j\}$, with $1 \le j \le n$, is a set of real values satisfying (6.2). Then*

$$s_k^T \nabla^2 f(x_k) s_k \;\le\; \frac{1}{m} \left[\frac{\min\limits_{1\le j\le m \,:\, \mu_j<0} a_j^2}{\max\limits_{1\le j\le m \,:\, \mu_j<0} a_j^2} \right] \lambda_{\min}^{(k)} \left[\nabla^2 f(x_k)\right], \tag{7.2}$$

where $\lambda_{\min}^{(k)} \left[\nabla^2 f(x_k)\right]$ is defined in (6.12).

Proof. As for Proposition 7.1 the proof basically follows the same guidelines of the one in Proposition 6.6, after recalling that the columns of Q_k are orthogonal vectors, replacing G_k by Q_k, D_k by T_k, and observing that in the singular value decomposition of Q_k the m nonzero singular values satisfy

$$1 = \sigma_{\min} = \sigma_1 = \cdots = \sigma_i = \cdots = \sigma_m = \sigma_{\max} = 1.$$

□

Observation 7.3. Observe that in principle the formulae (7.1) and (7.2) look simpler than (6.4) and (6.13), respectively. However, they reveal that the computation of the vector s_k inherently also requires the computation of the eigenvalues $\{\mu_j\}$ of the matrix T_k, in formulae (6.1) and (6.2). This is of course more computationally expensive than computing the spectrum of the diagonal matrix D_k. Nevertheless, calculating the spectrum of the tridiagonal matrix T_k only requires $O(k)$ operations. Thus, on the overall the user may decide if tightening the bounds (6.4) and (6.13) using (7.1) and (7.2), respectively may be worth the additional computation of the spectrum of T_k. As a general rule, for small values of k the computation of the eigenvalues of T_k may be negligibly small, with respect to the cost of the overall algorithm used to solve (1.1).

8 An alternative viewpoint using the CG method

We observe that the factorizations (5.2)-(5.4) are essential in order to prove the statement of Proposition 6.1. Furthermore, by Observation 6.4, we also know that the condition number $\kappa(G_k)$ is basically bounded, as long as the pivoting strategy in the Bunch–Kaufmann factorization is pursued as in [6]. This raises the next additional question: are there alternatives to the Bunch–Kaufmann factorization, that may fit within the framework of Proposition 6.1? To this purpose, in [23] the authors proved (see [23, Section 3.2]) the next couple of results:

- for the CG method (and for its extensions to the nonconvex quadratics - namely Planar-CG methods) stable factorizations similar to (5.2)-(5.4) are available too, with the additional property that the matrix W_k in (5.2) is structured (being *unit lower bidiagonal* for the CG method and *lower triangular* for Planar-CG methods);
- exploiting the structure of the matrix W_k allows to easily solve the linear system (4.4) by simply backtracking. By Section 4.1 of the current paper we already know that this second task is hardly replicable, using the Bunch–Kaufmann factorization, too.

Hence, we wondered about exploiting the CG-based factorizations from [6] within Proposition 6.1. This would allow for the construction of alternative negative curvature directions, all complying with our theoretical framework. The only one additional concern, using the CG method in place of the Bunch–Kaufmann factorization, is represented, for the former method, by the chance that

$$\limsup_{k\to\infty} \; \kappa(G_k) = +\infty,$$

so that the bound (6.4) may become ineffective.

Conversely, as long as applying the CG method we have that all the conjugate directions are provably uniformly linearly independent, then $\kappa(G_k)$ is *tout court* bounded. Once more, we remark that the last result allows to use the outcomes of Proposition 6.1 and to possibly propose an alternative negative curvature direction, using the directions generated by the CG method (as columns for G_k) in place of the Lanczos process / Bunch–Kaufmann factorization, within formula (6.1). Finally, in order to infer further considerations on the boundedness of $\kappa(G_k)$ when the CG method is used, in place of the Lanczos process / Bunch–Kaufmann factorization, the reader may refer to [19, 20] and [23].

9 Numerical experiments

To assess the performances of a Truncated Newton method which uses negative curvature directions computed according to the approach we described in the previous sections, we carried out an extensive numerical testing. We considered the same optimization framework adopted in [6], namely a Truncated Newton method based on the SYMMBK procedure (implemented in the routine HSL_MI02 of the HSL Mathematical Software Library [28]) to solve the Newton equation. In [6], SYMMBK pivoting rule has been slightly modified in order to provide, at each outer iteration k, a gradient–related Newton–type direction d_k. The reader can refer to [6] for any detail.

Now, as described in the previous sections, we exploit the SYMMBK routine also for iteratively computing, at each outer iteration k, a negative curvature direction s_k (if any). Then, we implemented the iterative scheme (1.4) where the steplength α_k is computed through the standard curvilinear linesearch procedure in [31]. We are aware (see, e.g., [26]) of the well-known problem which arises when combining a Newton-type direction and a negative curvature direction in the scheme (1.4), namely the possible different scaling of the two directions that could lead to inefficiency of the algorithm (as we also observed in preliminary testing). Here, we do not introduce any strategy for possibly selecting the best promising direction between the two, as also proposed in literature (see, e.g., [13, 21, 26, 35] and the discussion regarding this issue included in the Introduction).

Moreover, an additional safeguard must be considered in dealing with negative curvature directions within a Truncated Newton method. Indeed, when the iterates approach a local minimizer (hence the Newton-type direction entails a superlinear convergence rate), the use of negative curvature directions might partially "spoil" such good convergence rate, imposing a tight exploitation of local geometries associated with the function topology.

To overcome these two drawbacks, in the light of the conclusions proposed in Section 6.1 and Observation 6.4, we adopted the next two *zeroing rules* for the negative curvature direction: at the outer iteration k, a negative curvature direction z (if any) is ignored when

– its norm consistently differs from the norm of d_k, namely

$$\|z\| > \eta_1 \|d_k\| \quad \text{or} \quad \|z\| < \eta_2 \|d_k\|, \qquad \eta_1 > 0, \eta_2 > 0, \tag{9.1}$$

– it results

$$\|\nabla f(x_k)\| < \gamma_1 \qquad \text{and} \qquad \frac{z^T \nabla^2 f(x_k) z}{\|z\|^2} > -\gamma_2, \qquad \gamma_1 > 0, \gamma_2 > 0. \tag{9.2}$$

The rationale behind the first rule (based on (9.1)) relies on the fact that, computing a negative curvature direction z as described in Section 6 implies that both d_k and z are built using the same conjugate directions (the columns of the matrix G_k in (5.4)). Hence, one would expect that both the vectors z and d_k have a similar scaling. If this does not occur, then the scaling problem possibly arises: we believe that the computed negative curvature direction could introduce a detrimental effect on the efficiency of the algorithm, hence we do not consider it.

The second zeroing rule (based on (9.2)), concerns the situation that may occur whenever the algorithm generates iterates sufficiently close to a second order critical point, i.e. a stationary

point where the Hessian matrix is nearly positive semidefinite, and thus the Rayleigh–Ritz quotient (along z) is negative and close to zero. Also in this case, we do not consider the contribution of the negative curvature direction.

As concerns the values of the parameters adopted in (9.1) and (9.2), in our experimentation we use the following: $\eta_1 = 10^2, \eta_2 = 10^{-2}$ and $\gamma_1 = 10^{-3}, \gamma_2 = 10^{-2}$. More sophisticated strategies could be certainly adopted and will be the subject of future work (see also [22]).

Regarding the truncation criterion of the inner iterations, we adopt the standard *residual based criterion* [16], namely $\|\nabla f(x_k) + \nabla^2 f(x_k) d_k\| \leq \eta_k \|\nabla f(x_k)\|$, where $\eta_k = \min\left(\|\nabla f(x_k)\|, \frac{\sqrt{n}}{k}\right)$ is the forcing function we use. For the stopping criterion of the algorithm (outer iterations) we use the standard one $\|\nabla f(x_k)\| \leq 10^{-5} \max(1, \|x_k\|)$. We state that an algorithm fails to solve a problem if 3600 seconds of CPU time limit is exceeded.

To perform an extensive numerical testing, in our experimentation we considered all the large scale unconstrained test problems from CUTEst collection [27], amounting to 166 test problems, with sizes in the range 1,000–10,000. All the runs were performed on a PC with Intel Core i7-4790K CPU @ 4.00GHz with 32 Gb RAM.

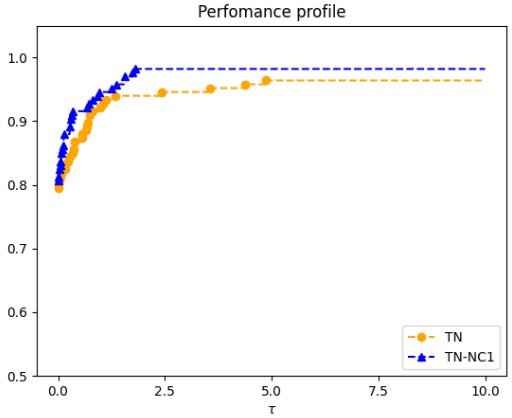
The results are reported in terms of number of outer iterations (*it*), number of function evaluations (*f-eval*), number of inner iterations (*inner-it*), optimal function value (*function value*) and CPU time (*time*) in seconds. In the case an algorithm incorporates negative curvature directions, the number of negative curvature directions (*neg. curv.*) actually used by the algorithm is reported, too. Tables A.1, A.2 and A.3 in the Appendix A include the results obtained on the whole test set in the case negative curvature directions are not considered in the Truncated Newton algorithm. We name this algorithm TN.

9.1 Use of the negative curvature direction (6.1)

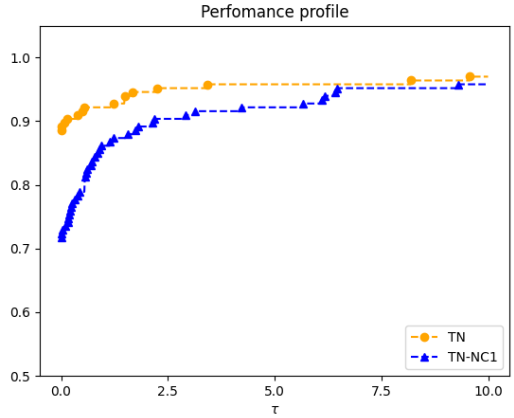
Tables A.4, A.5 and A.6 in the Appendix A report the results obtained by the algorithm which uses the negative curvature direction defined in (6.1), adopting the zeroing rule in (9.1) and (9.2). We name this algorithm TN-NC1.

Now, we compare the performance of TN-NC1 and TN algorithms by using the widely adopted *performance profiles* [18]. Figures 9.1a, 9.1b, 9.1c and 9.1d report such performance profiles in terms of number of (outer) iterations, number of function evaluations, number of inner iterations and CPU time, respectively. These plots refer to the whole set of test problems. They clearly highlight the efficiency and the robustness in terms of iterations and inner iterations of the algorithm TN-NC1, with respect to the algorithm TN which does not use negative curvature directions. On the other hand, TN-NC1 algorithm on the overall requires a larger number of function evaluations. Actually, this behaviour was expected, since it is due to the standard curvilinear linesearch procedure used which is based on a rough combination the two search directions. We are convinced that a more sophisticated linesearch technique might be adopted, when second order information related to negative curvature directions is available. In this regard, a further investigation seems mandatory in the light of the outcomes of the present numerical experiences.

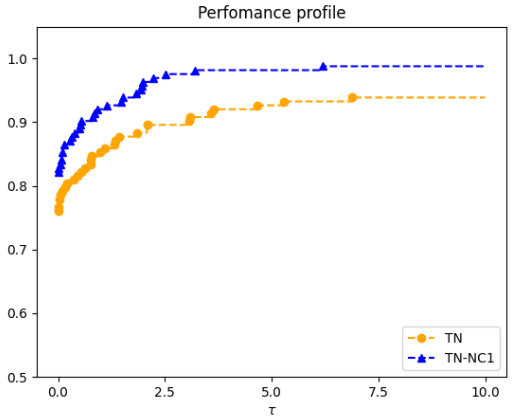
The detailed results in Tables A.1, A.2, A.3 and in Tables A.4, A.5 and A.6 provide us with some further interesting evidences. First, it can be observed that on 9 difficult test problems both the algorithms fail to converge within 3600 seconds of CPU time. On 1 test problem (CURLY30 with $n = 10,000$) the use of even a few negative curvature directions (say 5) enables the algorithm TN-NC1 to converge in only 129.58 seconds, whereas TN fails to converge. On 71 test problems negative curvature directions are actually encountered and used by TN-NC1 algorithm; on 30 of these test problems, the two algorithms converge to different local minimizers. Table 9.1 reports the optimal function values obtained by TN and TN-NC1 algorithms on the latter test problems. This table clearly highlights the expected capability of the algorithm TN-NC1 which uses negative curvature directions to converge towards better local minimizers. The cases in which a better value is obtained by TN algorithm are comparatively very few. From the detailed results reported in the Appendix A it can be also observed that, in many cases the additional effort due to the computation of negative curvature directions, is balanced by a greater overall efficiency of the



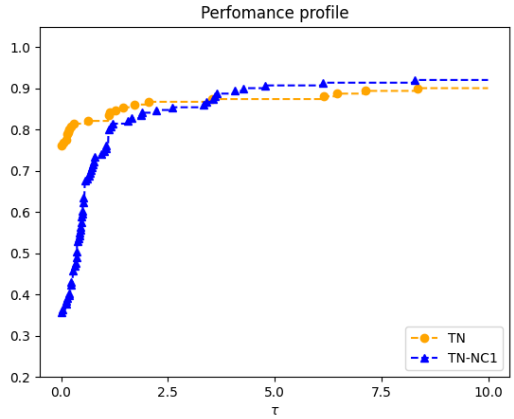
(a) Performance profiles based on *outer iterations*.



(b) Performance profiles based on *funct. evaluations*.



(c) Performance profiles based on *inner iterations*.



(d) Performance profiles based on *CPU time*.

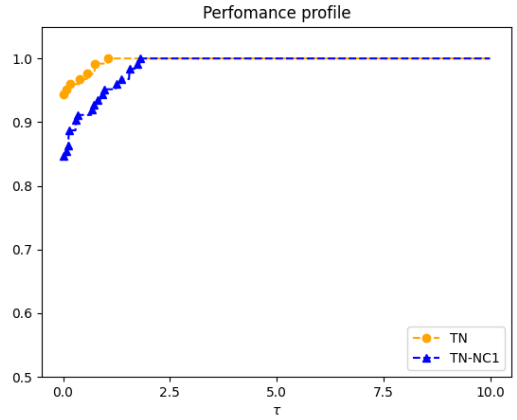
Figure 9.1: Performance profiles for the *whole set* of 166 test problems.

Problem	n	TN algorithm *function value*	TN-NC1 algorithm *function value*
BROYDN7D	1000	5.598036D+02	**3.047159D+02**
BROYDN7D	5000	3.125637D+03	**1.659976D+03**
BROYDN7D	10000	**1.233247D+00**	6.883895D+03
CHAINWOO	1000	4.324322D+02	**2.514980D+02**
CHAINWOO	4000	2.294394D+03	**1.582515D+03**
CHAINWOO	10000	6.305963D+03	**2.890864D+03**
COSINE	1000	-9.985153D+02	**-9.990000D+02**
CURLY10	1000	-9.765683D+04	**-1.003125D+05**
CURLY10	5000	-4.665618D+05	**-5.015815D+05**
CURLY10	10000	-1.002761D+06	**-1.003163D+06**
CURLY20	1000	-9.814893D+04	**-1.003093D+05**
CURLY20	5000	-4.844972D+05	**-5.015758D+05**
CURLY20	10000	-1.002861D+06	**-1.003162D+06**
CURLY30	1000	-9.854450D+04	**-1.000507D+05**
CURLY30	5000	-4.929491D+05	**-5.015808D+05**
FLETCBV3	1000	**-1.083664D+05**	-3.189503D+03
FLETCBV3	5000	-2.221834D+07	**-6.880147D+07**
FLETCBV3	10000	**-1.321473D+09**	-1.147714D+09
GENHUMPS	1000	3.252751D+02	**2.359907D-10**
NCB20	1010	9.663298D+02	**9.208174D+02**
NCB20	5010	-1.394735D+03	**-1.447533D+03**
NCB20	10010	-2.642929D+03	**-5.313355D+03**
NONCVXUN	1000	2.405825D+03	**2.327616D+03**
NONCVXU2	1000	2.381367D+03	**2.317103D+03**
SINQUAD	10000	-2.642227D+07	**-2.642315D+07**
SPARSINE	1000	6.341091D+05	**1.618616D+05**
SPARSINE	5000	**1.567241D+07**	1.697258D+07
SPARSINE	10000	**6.448147D+07**	7.005234D+07
SPMSRTLS	1000	6.321999D+01	**5.608497D-02**
SPMSRTLS	10000	2.937016D+00	**3.409363D-11**

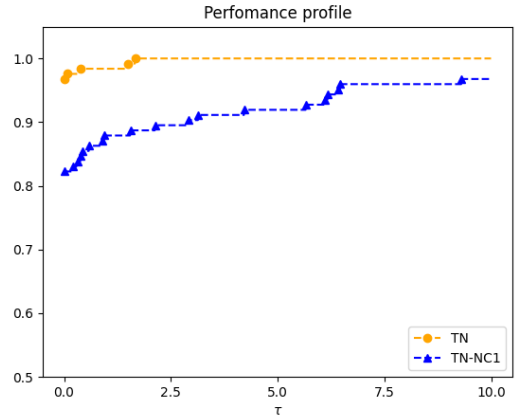
Table 9.1: Optimal function values obtained by TN and TN-NC1 algorithms, on those test problems where they converge to different local minimizers. For each problem the best function value obtained is highlighted in bold.

algorithm, so that CPU time required by TN-NC1 algorithm on average shows a modest increase.

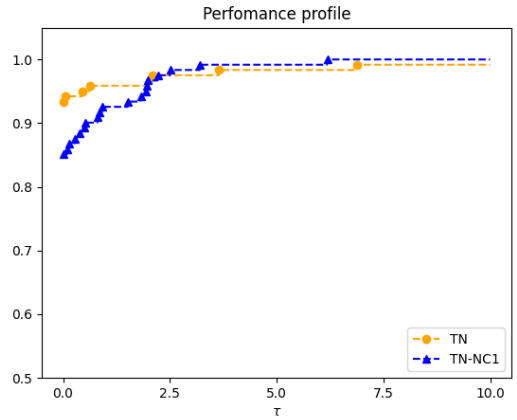
Since on a number of test problems the two algorithms converge towards different minimizers, in a second experiment we again plot performance profiles for comparing TN and TN-NC1 algorithms but considering only those test problems where they converge to the same local minimizer. Figures 9.2a, 9.2b, 9.2c and 9.2d report these plots in terms of number of (outer) iterations, number of function evaluations, number of inner iterations and CPU time, respectively.



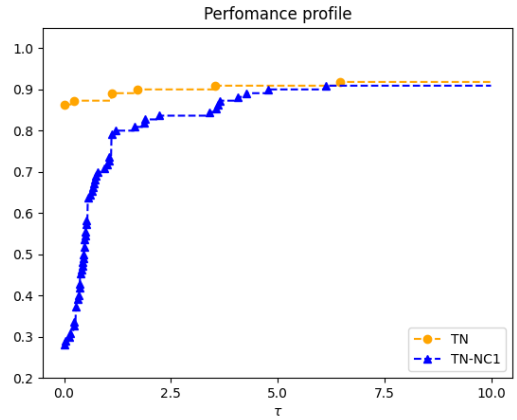
(a) Performance profiles based on *outer iterations*.



(b) Performance profilse based on *funct. evaluations*.



(c) Performance profiles based on *inner iterations*.



(d) Performance profiles based on *CPU time*.

Figure 9.2: Performance profiles for test problems where both the algorithms TN and TN-NC1 converge to the *same local minimizer* (136 test problems).

A comparison of these new plots with the ones in Figure 9.1 leads to an important consideration regarding the use of the performance profiles when ranking different algorithms. Indeed, in terms of number of function evaluations and CPU time a better performance of TN algorithm is observed both in Figures 9.1b, 9.2b, and in Figures 9.1d, 9.2d. Conversely, in terms of number of outer iterations (see Figure 9.1a and Figure 9.2a) and in terms of number of inner iterations (see Figure 9.1c and Figure 9.2c) we carry out a different conclusion: the profiles referred to the whole test set do not seem to agree with those related to test functions where both the algorithms converge to the same local minimizer. On the other hand, one could reasonably claim that including in the performance comparison test problems where the algorithms converge towards different local minimizers could be unfair. Actually, the key point is that, in comparing the performance among different algorithms, performance profiles do not take into account the "quality" of the solution found by algorithms, i.e., their capability to determine better local minimizers. This drawback

could be overcome neither by adopting *performance profiles* nor by using the *data profiles* proposed in [36]. This motivates the introduction, in the current paper, of novel profiles based on the "quality" of the solution provided by the algorithms (see the next Section 9.3). Of course, this new tool appears of fundamental importance for comparing algorithms which use negative curvature directions versus algorithms that do not use them.

9.2 Use of alternative negative curvature directions

The numerical experimentation reported in the previous Section 9.1 relies on the use of the negative curvature direction given by (6.1). However, to enhance our investigation on negative curvature directions within Truncated Newton methods it might be worth considering two alternatives for computing such directions. In place of considering the sum in (6.1), we might select only one column of the matrix $G_{(j)}$ (along with the associated coefficient a_j). In particular, it is possible to consider in place of the choice (6.1)

$$z = a_{\bar{h}} G_{(\bar{h})}, \quad \text{where} \quad \bar{h} = \operatorname*{argmin}_h \{\mu_h \mid \mu_h < 0\}, \tag{9.3}$$

and

$$z = a_{\bar{l}} G_{(\bar{l})}, \quad \text{where} \quad \bar{l} = \min_h \{h \mid \mu_h < 0\}. \tag{9.4}$$

The rationale behind the choice in (9.3) is to consider only the smallest negative eigenvalue of the matrix D_k in (5.3). The selection (9.4) consists of considering only the first negative eigenvalue of D_k. We denote by TN-NC2 and TN-NC3 the algorithms which use the negative curvature directions in (9.3) and (9.4), respectively. In both the cases, we adopt again the zeroing rules in (9.1) and (9.2) for negative curvature directions, too.

Tables A.7, A.8 and A.9 in the Appendix A report the results obtained by algorithm TN-NC2. Tables A.10, A.11 and A.12 report the results obtained by algorithm TN-NC3. To compare the capability of the algorithms which incorporate negative curvature directions to determine better local minimizers, in the next Section 9.3 we introduce novel profiles to assess the effectiveness of the negative curvature directions adopted in TN-NC1, TN-NC2 and TN-NC3 versus TN.

9.3 Quality Profiles for single–objective smooth optimization

Benchmarking different algorithms for local constrained and unconstrained optimization over a set of test problems has always been a challenging issue, because any proposed procedure needs to be unbiased with respect to the features of the compared codes, as well as independent of the test set used for benchmarking. This last consideration, was successfully addressed by the seminal papers [18] (where *Performance Profiles* have been introduced) and [36] (where *Data Profiles* have been defined). However, neither of them proposed a *specific tool* to benchmarking the "quality" of the optimal solution determined by different algorithms, when they are experienced on a set of test problems. As we pointed out at the end of Section 9.1, when ranking different algorithms, it might be very fruitful to assess their capability of determining better local minimizers, i.e. solution points with lower value of the objective function.

Drawing inspiration from both [18] and [36], the main purpose of the present section is to possibly fill the last gap, by adopting a perspective that does not upset the basics in [18, 36]. To this aim, we now introduce a novel class of profiles that we name *Quality Profiles*.

Assume the set $\mathcal{S}$ of iterative solvers and the set $\mathcal{P}$ of test problems are considered, being $|\mathcal{S}| \geq 2$ and $|\mathcal{P}| \geq 1$ their cardinalities. Let $x_0^{(p)} \in \mathbb{R}^n$ be the starting point for all the solvers on the test problem $p \in \mathcal{P}$, and let $f^{(p)}(x)$ be the objective function of the problem $p \in \mathcal{P}$. Let $f_L^{(p)}$ be a reference value (for the objective function) on the problem p. If x^* indicates the best iterate found by the solver $s \in \mathcal{S}$ on the test problem p (observe that for the sake of simplicity—and with the idea of reducing redundancy—we drop in x^* the dependency on both s and p), as remarked also in [18] and [36] possible choices for $f_L^{(p)}$ can be, for instance:

1. $f_L^{(p)} = \min_{s \in \mathcal{S}} \left\{ f_s^{(p)}(x^*) \right\}$,

2. a *reference value* generated by an algorithm (typically an efficient one) non included in the list of the compared solvers,

3. $f_L^{(p)} = \dfrac{1}{|\mathcal{S}|} \displaystyle\sum_{s \in \mathcal{S}} \left\{ f_s^{(p)}(x^*) \right\}$,

where $f_s^{(p)}(x^*)$ denotes the optimal function value determined by the solver s on the problem p. In this regard, in accordance with [18, 36], we adopt the first one among the above three choices and define for each solver $s \in \mathcal{S}$ the ratio

$$Q_s(\tau) = \frac{1}{|\mathcal{P}|} size \left\{ p \in \mathcal{P} \ : \ f_s^{(p)}(x^*) - f_L^{(p)} \leq \tau \left[f^{(p)}(x_0^{(p)}) - f_L^{(p)} \right] \right\}, \tag{9.5}$$

being $\tau \geq 0$. Since evidently for any solver s we have

$$f_s^{(p)}(x^*) \leq f^{(p)}(x_0^{(p)}), \tag{9.6}$$

then for any $s \in \mathcal{S}$ we obtain $Q_s(\tau) = 1$ for all $\tau \geq 1$. This immediately suggests that it suffices to consider hereafter for the parameter τ the range

$$\tau \in [0, 1].$$

We can now introduce the following formal definition of *Quality Profiles*.

Definition 9.1. *Given the benchmark set of test problems $\mathcal{P}$ and the set of solvers $\mathcal{S}$, the corresponding* Quality Profile *is defined as the collection of plots of all the functions $\{Q_s(\tau)\}$ in* (9.5) *when $\tau \in [0, 1]$.*

As preliminary facts, the next considerations hold for the quality profiles:

- by definition we have $0 \leq Q_s(0) \leq 1$, for any solver $s \in \mathcal{S}$;
- for $\tau = 0$ the value $Q_s(0)$ indicates the percentage of problems where the solver s provides the best value of the objective function, with respect to all the other solvers (note that an almost identical property holds for the performance profiles when the abscissa value is equal to 1);
- quality profiles suitably take into account possible failures of the solver s on some benchmark problems: indeed in the last case the corresponding problem will be discarded and will not contribute to increase the quantity $size\{\cdot\}$ in (9.5). Hence, we will have $Q_s(1) < 1$ for the corresponding solver s (again, here we can immediately realize that a similar property holds for performance profiles);
- performance profiles can hardly be seen as a possible alternative to quality profiles, for at least a couple of reasons:
  - since for a given solver $s \in \mathcal{S}$ and a test problem $p \in \mathcal{P}$ we might have $f_s^{(p)}(x^*) \gtreqless 0$, then the definition of the performance profile, when applied for benchmarking the *objective function value*, might not be well posed (because some of the positive ratios $r_{p,s}$ in [18], needed to compute performance profiles, might be undefined or inconsistent);
  - if performance profiles were used to benchmark the final value of the objective function, then they would completely disregard the starting point $x_0^{(p)}$, while the definition (9.5) explicitly aims at monitoring the progress of each solver with respect to the function value at the initial iterate $x_0^{(p)}$.

Apart from the last basic properties of quality profiles a number of theoretical results can be proved for them, included the next two ones.

Lemma 9.2. *Quality Profiles are invariant under any affine transformation of the objective functions.*

Proof. Let us consider, without loss of generality, the benchmark problem $p \in \mathcal{P}$ and the corresponding objective function $f^{(p)}(x)$. For any $a > 0$ and $b \in \mathbb{R}$ we define the affine transformation $af^{(p)}(x) + b$ so that from the definition of $\{Q_s(\tau)\}$, and for any $s \in \mathcal{S}$, it is

$$\left[\left(af_s^{(p)}(x^*) + b\right) - \left(af_L^{(p)} + b\right)\right] = a\left[f_s^{(p)}(x^*) - f_L^{(p)}\right]$$

$$\left[\left(af_s^{(p)}(x_0) + b\right) - \left(af_L^{(p)} + b\right)\right] = a\left[f_s^{(p)}(x_0) - f_L^{(p)}\right].$$

Thus, the next two inequalities are equivalent

$$\left[\left(af_s^{(p)}(x^*) + b\right) - \left(af_L^{(p)} + b\right)\right] \leq \tau\left[\left(af^{(p)}(x_0) + b\right) - \left(af_L^{(p)} + b\right)\right]$$

$$a\left[f_s^{(p)}(x^*) - f_L^{(p)}\right] \leq \tau a\left[f^{(p)}(x_0) - f_L^{(p)}\right],$$

so that the definition of $\{Q_s(\tau)\}$ proves the result. □

Lemma 9.3. *For any $p \in \mathcal{P}$, let $f^{(p)}(x_0) \geq f_L^{(p)}$. Then, for any $s \in \mathcal{S}$ the function $Q_s(\tau) : [0,1] \to [0,1]$ is nondecreasing.*

Proof. Given $\tau_1, \tau_2 \in [0,1]$, with $\tau_1 \leq \tau_2$, let $p \in \mathcal{P}$. Then, by the definition of $Q_s(\tau)$, if

$$f_s^{(p)}(x^*) - f_L^{(p)} \leq \tau_1\left[f^{(p)}(x_0) - f_L^{(p)}\right]$$

then we have also

$$f_s^{(p)}(x^*) - f_L^{(p)} \leq \tau_1\left[f^{(p)}(x_0) - f_L^{(p)}\right] \leq \tau_2\left[f^{(p)}(x_0) - f_L^{(p)}\right].$$

Thus, the benchmark problems where the leftmost inequality in the last relation is fulfilled, is included in the set of benchmark problems where the rightmost inequality is fulfilled. Hence, the definition of $Q_s(\tau)$ yields the result. □

Lemma 9.4. *Let $f^{(p)}(x_0) \neq f_L^{(p)}$ for any $p \in \mathcal{P}$, and let*

$$\tau_M = \max_{s\in\mathcal{S},\ p\in\mathcal{P}} \left\{ \frac{f_s^{(p)}(x^*) - f_L^{(p)}}{f^{(p)}(x_0) - f_L^{(p)}} \right\}.$$

Then $Q_s(\tau) = 1$ for any $\tau \geq \tau_M$ and $s \in \mathcal{S}$.

Proof. The proof straightforwardly follows from (9.6) and the definition of $Q_s(\tau)$. □

Observation 9.5. The functions $\{Q_s(\tau)\}$, for $\tau \in [0,1]$, may be of difficult comparison in case the number of solvers increases, in particular when $\tau \in \{0,1\}$ (i.e. at the extremes of the interval for τ). The last fact may be due to close tracks associated to the functions $\{Q_s(\tau)\}$ when τ belongs to $\{0,1\}$. This suggests that a more sophisticated tool needs to be introduced in order to compress / expand portions of the plot of the functions $\{Q_s(\tau)\}$, both along the abscissa and the ordinate axes of the quality profile. On this purpose, note that any logarithm / exponential scaling would be unadvisable, because of the range of interest $[0,1]$ on both the axes.

The last observation suggests that to compress / expand portions of a quality profile, in order to better sort and rank the different tracks associated with codes, we can modify (9.5) into ($\tau \in [0,1]$)

$$[Q_s(\tau)]^{r_2} = \frac{1}{|\mathcal{P}|} size \left\{ p \in \mathcal{P} \ : \ f_s^{(p)}(x^*) - f_L^{(p)} \leq \tau^{r_1} \left[f^{(p)}(x_0^{(p)}) - f_L^{(p)} \right] \right\}. \tag{9.7}$$

where r_1 and r_2 are positive real numbers, so that

- for $r_1 \in \{1, 2, 3, 4, \ldots\}$ we obviously have $0 \leq \tau^{r_1} \leq 1$. Moreover, for a given solver $s \in S$, when r_1 increases, the portion of the quality profile corresponding to *small* values of the abscissa (say equivalently τ close to zero) will be *expanded*. Conversely, when r_1 increases, the portion of the quality profile corresponding to *large* values of the abscissa (say equivalently τ close to one) will be *compressed*. As an example, note that when $r_1 = 1$ the mid–value of the abscissa axis is $1/2$, while for $r_1 = 5$ the mid–value of the abscissa axis becomes $(1/2)^5 = 1/32$;
- for $r_1 \in \{1, 1/2, 1/3, 1/4, \ldots\}$ we obviously have again $0 \leq \tau^{r_1} \leq 1$. Moreover, now for smaller values of r_1 the portion of the quality profile corresponding to *large* values of the abscissa (say equivalently τ close to one) will be *expanded*, and for larger values of r_1 the portion of the quality profile corresponding to *small* values of the abscissa (say equivalently τ close to zero) will be *compressed*;
- a dynamics similar to the one detailed in the previous items is experienced by increasing $r_2 \in \{1, 2, 3, 4, \ldots\}$ or decreasing $r_2 \in \{1, 1/2, 1/3, 1/4, \ldots\}$. As an example, when $r_2 = 1$ the mid–value of the ordinate axis is $1/2$ while for $r_2 = 6$ the mid–value of the ordinate axis is $(1/2)^6 = 1/64$.

The last considerations, along with relation (9.2), reveal a straightforward result (the proof is omitted) summarized in the next lemma.

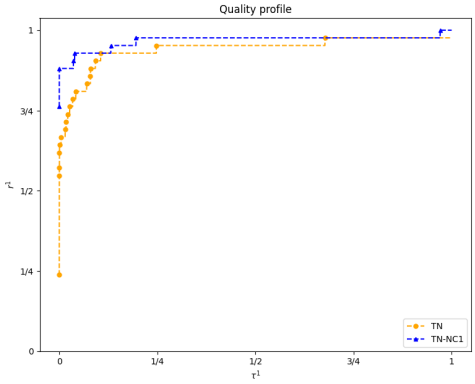
Lemma 9.6. *Given the test set $\mathcal{P}$ and the set of solvers $\mathcal{S} = \{s_1, \ldots, s_r\}$, the relative ranking among the non-negative quantities $[Q_{s_1}(\tau)]^{r_2}, \ldots, [Q_{s_r}(\tau)]^{r_2}$, when $\tau \in [0,1]$, is invariant with respect to the choice of the positive parameters r_1 and r_2.*

An extension of quality profiles to multi-objective smooth optimization and to derivative-free optimization is reported in the Appendix B. In a further study they will be better investigated with specific reference to the context in the corresponding literature.

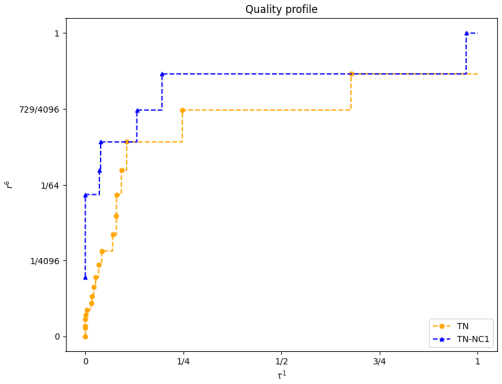
9.4 A numerical experience with quality profiles

In Section 9.1, when comparing TN and TN-NC1 algorithms, we already highlighted how performance profiles may be inadequate for ranking algorithms which show convergence to different solution points, on a set of test problems. Hence, we now complete the comparison between these two algorithms by using the *quality profiles* detailed in Section 9.3, so that the quality of the obtained solutions can be duly considered. An introductory example of application for quality profiles can be found in Figure 9.3, which corresponds to set in (9.7) the values of r_1 and r_2 reported in Table 9.2. We can immediately appreciate that the outcomes in Figure 9.3 definitely comply, as expected, with all the results in Lemmas 9.2, 9.3 and 9.4, along with Observation 9.5. As a further consideration, unlike *performance* and *data profiles*, the chance *to possibly zoom* in any region of the quality profiles (i.e., the area $[0,1] \times [0,1]$) is definitely appealing and allows to precisely distinguish among the relative positions of the tracks associated with the solvers. On the overall, plots in Figure 9.3 clearly show that, as expected, in terms of quality of the solution found, the algorithm TN-NC1 which uses negative curvature directions outperforms the algorithm TN which does not.

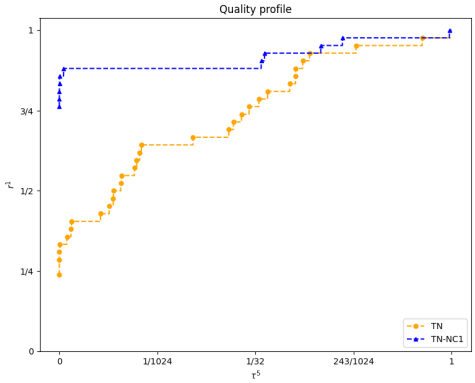
Moreover, in the spirit of carrying out the investigation on the use of negative curvature directions, to enhance the capability of an algorithms to determine better local minimizers, we now include in the comparison also the results obtained by algorithms TN-NC2, TN-NC3 described in



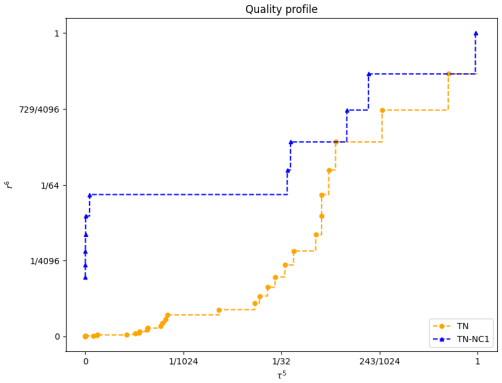
(a) $r_1 = 1$ and $r_2 = 1$ in (9.7).



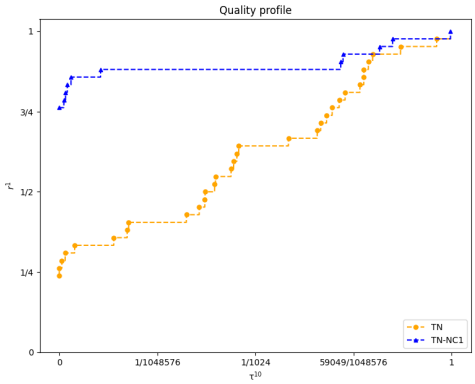
(b) $r_1 = 1$ and $r_2 = 6$ in (9.7).



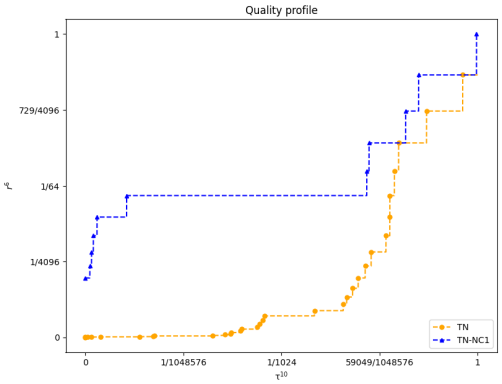
(c) $r_1 = 5$ and $r_2 = 1$ in (9.7).



(d) $r_1 = 5$ and $r_2 = 6$ in (9.7).



(e) $r_1 = 10$ and $r_2 = 1$ in (9.7).



(f) $r_1 = 10$ and $r_2 = 6$ in (9.7).

Figure 9.3: Samples of Quality Profiles with the two solvers TN and TN-NC1.

(a)	$r_1 = 1$	$r_2 = 1$
(b)	$r_1 = 1$	$r_2 = 6$
(c)	$r_1 = 5$	$r_2 = 1$
(d)	$r_1 = 5$	$r_2 = 6$
(e)	$r_1 = 10$	$r_2 = 1$
(f)	$r_1 = 10$	$r_2 = 6.$

Table 9.2: Values of r_1 and r_2 in (9.7) adopted in the quality profiles

Section 9.2 (which use alternative negative curvature directions). The Figure 9.4 reports a comparison, adopting quality profiles, for the 4 different algorithms TN, TN-NC1, TN-NC2 and TN-NC3. As for Figure 9.3, the same values for the parameters r_1 and r_2 reported in Table 9.2 were adopted in Figure 9.4, too.

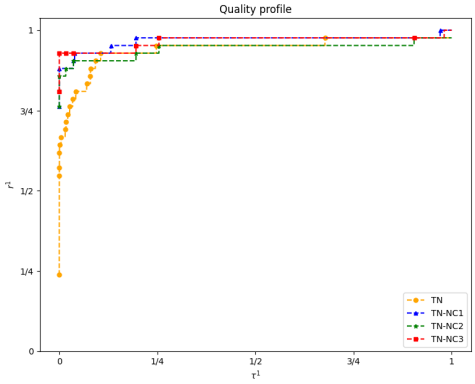
We highlight that not all the four tracks in the last figure converge to the same value (i.e. one) when $\tau = 1$. This can indeed be easily explained (see also Section 9.3) by recalling that the piece of information related to the number of failures associated to each solver is duly taken into account by quality profiles. The last consideration can be regarded as a counterpart of a similar property that holds for performance and data profiles.

Finally, observe that all the quality profiles in Figures 9.3 and 9.4 report five tickers on both the axes. They correspond, on each axis, to the five values $\{0, 1/4, 1/2, 3/4, 1\}$ transformed through the selection of the values r_1 and r_2 in the corresponding quality profile. E.g. when $r_1 = 1$ and $r_2 = 6$ (see the quality profile in the position (b) of the Figures 9.3 and 9.4) the values $\{0, 1/4, 1/2, 3/4, 1\}$ appear unaltered on the abscissa axis (since $r_1 = 1$), while they are mapped to the points $\{0^6\ ,\ (1/4)^6\ ,\ (1/2)^6\ ,\ (3/4)^6\ ,\ 1^6\} \equiv (0\ ,\ 1/4096\ ,\ 1/64\ ,\ 720/4096\ ,\ 1)$ on the ordinate axis (since $r_2 = 6$).

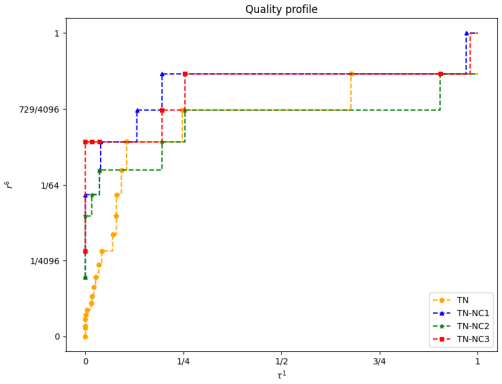
Once more, we strongly highlight the importance of introducing the scaling parameters r_1 and r_2. Indeed, Figure 9.4a shows that the natural (i.e. without any scaling) choice $r_1 = r_2 = 1$ may deteriorate our capability of distinguishing and ranking the quality among solvers. That is an expected effect, associated to solvers that over a given region of the abscissa axis (in the quality profile) have close tracks. Hence, a suitable alternative choice for r_1 and r_2 may represent a winning strategy for precisely benchmarking multiple codes, for any given subset of the abscissa axis. On this guideline, let us recall that a so precise choice for scaling parameters within performance and data profiles is far from being possible. This is due to the fact that unlike quality profiles, where we always find a unique interval for the abscissa values (say $[0, 1]$), performance and data profiles may have to consider different abscissa intervals, depending on the achieved results for the compared solvers.

The plots in Figure 9.4 confirm the fact that algorithms which use negative curvature directions in most cases determine better local minimizers. Moreover, the most effective among the three codes incorporating negative curvature directions seems to be TN-NC3. We recall that in this last algorithm the negative curvature direction is computed only selecting information associated to the first negative eigenvalue of the diagonal matrix D_k in (4.1). The last fact confirms what was also pointed out in [21], and reveals that on selected large scale settings the accurate and expensive computation of a negative curvature direction can be possibly dodged.

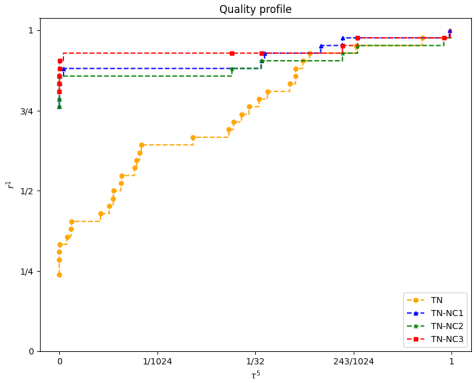
Observation 9.7. As a further comment, observe that given the set of benchmark functions, the area below any plot of a quality profile is uniquely associated with the corresponding algorithm. Hence, recalling the rationale behind performance and quality profiles, the larger that area the better the corresponding algorithm in terms of performance. Furthermore, in quality profiles the range of values of the abscissa is always $[0, 1]$, while for performance profiles it can be much different depending on the performance of any algorithm. Hence, for a given set of benchmark functions, the idea of associating a unique positive number to any track (i.e. the area below the track) of a quality profile cannot immediately be extended also to performance profiles and data profiles. This last comment highlights an additional feature of our proposal, with respect to the current



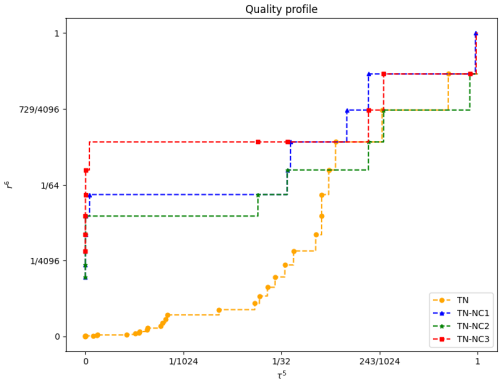
(a) $r_1 = 1$ and $r_2 = 1$ in (9.7).



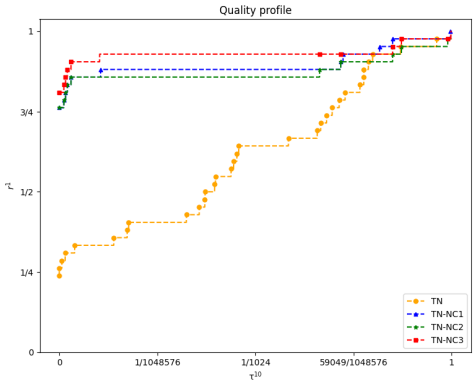
(b) $r_1 = 1$ and $r_2 = 6$ in (9.7).



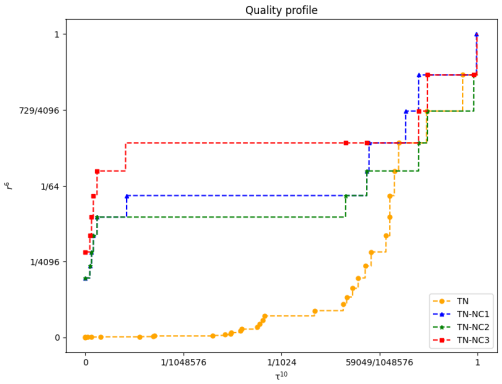
(c) $r_1 = 5$ and $r_2 = 1$ in (9.7).



(d) $r_1 = 5$ and $r_2 = 6$ in (9.7).



(e) $r_1 = 10$ and $r_2 = 1$ in (9.7).



(f) $r_1 = 10$ and $r_2 = 6$ in (9.7).

Figure 9.4: Samples of Quality Profiles with the four solvers TN, TN-NC1, TN-NC2 and TN-NC3.

literature on performance benchmarking, within the Continuous Optimization community.

The use of quality profiles to benchmark multiple codes should possibly not be confined only to algorithms that converge to second order critical points. Indeed, quality profiles can also provide a tool that can confirm the effectiveness of algorithms converging to first order critical points, too. In particular, the use of quality profiles can also be a valuable tool for first order methods, to show the reliability of the solutions they yield.

Finally, the arrangement in (9.7), with respect to (9.5), is subject to further generalizations, since the power τ^{r_1} can be replaced by another continuous function $\phi_1(\tau)$, such that

(a) $0 \leq \phi_1(\tau) \leq 1$;

(b) ϕ_1 is strictly increasing;

(c) $\phi_1(0) = 0$ and $\phi_1(1) = 1$.

As an example, the choice of $\phi_1(\tau)$ may be driven by the functions $\{Q_{s_i}(\tau)\}$, first computed using (9.5), with $s_i \in \mathcal{S}$, so that these functions can be better *scattered* in $[0,1] \times [0,1]$ after introducing $\phi_1(\tau)$ in (9.7), and defining a measure for scattering. A similar alternative arrangement can be conceived for the ordinate axes in the quality profiles, i.e. replacing the power $[Q_s(\tau)]^{r_2}$ by an alternative function $\phi_2[Q_s(\tau)]$ endowed with properties analogous to (a)-(c). Observe, that in principle, the idea of rewriting (9.7) as

$$\phi_2\left[Q_s(\tau)\right] = \frac{1}{|\mathcal{P}|} size\left\{p \in \mathcal{P} \ : \ f_s^{(p)}(x^*) - f_L^{(p)} \leq \phi_1(\tau)\left[f^{(p)}(x_0^{(p)}) - f_L^{(p)}\right]\right\}, \qquad 0 \leq \tau \leq 1,$$

where both ϕ_1 and ϕ_2 satisfy the above assumptions and the tracks $\{\phi_2\left[Q_s(\tau)\right]\}_{s\in\mathcal{S}}$ are better scattered in the area $[0,1] \times [0,1]$, may correspond to find ϕ_1 and ϕ_2 such that they solve the maximization problem

$$\max_{\phi_1,\phi_2} \int_0^1 \sum_{\substack{s,t\in\mathcal{S},\\ s\neq t}} \left\{\phi_2[Q_s(\tau)] - \phi_2[Q_t(\tau)]\right\}^2 \, d\tau. \tag{9.8}$$

This last issue (strictly related to quality profiles) deserves additional attention, since proving the existence and the uniqueness of the solution of (9.8) needs further investigation, with the specific reference to the criticisms raised in [25]. Indeed, the chance to select ϕ_1 and ϕ_2 so that a clear ranking among solvers is also identified, would be definitely appealing.

A Appendix

In this appendix we give the complete results obtained in our experimentation. They are reported in terms of number of outer iterations (*it*), number of functions evaluations (*f-eval*), number of inner iterations (*inner-it*), optimal function value (*function value*) and CPU time (*time*) in seconds. In the case an algorithm incorporates negative curvature directions, the number of negative curvature directions (*neg. curv.*) actually used by the algorithm is reported, too

In particular, Tables A.1, A.2 and A.3 include the results obtained by the algorithm that does not use negative curvature directions, namely TN. Tables A.4, A.5 and A.6 report the results for the algorithm TN-NC1; Tables A.7, A.8 and A.9 refer to results for the algorithm TN-NC2; finally, in Tables A.10, A.11 and A.12 the results for the algorithm TN-NC3 are included.

Observe that when "> 3600" is reported in the column *time* it means that the corresponding algorithm experiences a failure due to exceeded CPU allowed. On the contrary, "-" in the column *time* indicates indeed a linesearch failure (i.e., a too small stepsize was computed).

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*time*
ARWHEAD	1000	6	6	6	1.690288D-10	0.01
ARWHEAD	5000	6	6	6	1.110001D-12	0.01
ARWHEAD	10000	6	13	6	4.440448D-12	0.02
BDQRTIC	1000	13	13	79	3.983818D+03	0.01
BDQRTIC	5000	13	13	75	2.000626D+04	0.05
BDQRTIC	10000	13	13	75	4.003431D+04	0.11
BROYDN7D	1000	166	674	76121	5.598036D+02	3.53
BROYDN7D	5000	380	1899	756275	3.125637D+03	146.81
BROYDN7D	10000	4293	18481	146179	1.233247D+00	95.26
BRYBND	1000	8	8	55	4.530351D-12	0.01
BRYBND	5000	12	17	80	2.788633D-16	0.06
BRYBND	10000	11	12	67	6.060220D-16	0.12
CHAINWOO	1000	183	517	46530	4.324322D+02	2.55
CHAINWOO	4000	263	852	248636	2.294394D+03	48.60
CHAINWOO	10000	472	1785	1080439	6.305963D+03	513.12
COSINE	1000	12	14	46	-9.985153D+02	0.01
COSINE	5000	10	15	23	-4.999000D+03	0.02
COSINE	10000	10	14	23	-9.999000D+03	0.04
CRAGGLVY	1000	15	15	144	3.364231D+02	0.02
CRAGGLVY	5000	15	15	131	1.688215D+03	0.08
CRAGGLVY	10000	17	17	174	3.377956D+03	0.20
CURLY10	1000	163	450	153010	-9.765683D+04	7.83
CURLY10	5000	395	1370	1925009	-4.665618D+05	442.00
CURLY10	10000	20	26	102819	-1.002761D+06	46.46
CURLY20	1000	140	383	129945	-9.814893D+04	9.34
CURLY20	5000	537	1246	2635010	-4.844972D+05	866.32
CURLY20	10000	21	30	105447	-1.002861D+06	69.23
CURLY30	1000	128	284	116050	-9.854450D+04	10.72
CURLY30	5000	322	970	1556214	-4.929491D+05	661.91
CURLY30	10000	–	–	–	–	> 3600
DIXMAANA	1500	7	7	10	1.000000D+00	0.02
DIXMAANA	3000	7	7	10	1.000000D+00	0.01
DIXMAANB	1500	8	8	9	1.000000D+00	0.00
DIXMAANB	3000	8	8	9	1.000000D+00	0.01
DIXMAANC	1500	9	9	11	1.000000D+00	0.00
DIXMAANC	3000	9	9	10	1.000000D+00	0.01
DIXMAAND	1500	9	9	11	1.000000D+00	0.00
DIXMAAND	3000	10	10	12	1.000000D+00	0.01
DIXMAANE	1500	42	164	21636	1.000000D+00	1.56
DIXMAANE	3000	10	10	401	1.000000D+00	0.06
DIXMAANF	1500	15	15	518	1.000000D+00	0.05
DIXMAANF	3000	20	34	1458	1.000000D+00	0.25
DIXMAANG	1500	15	16	516	1.000000D+00	0.05
DIXMAANG	3000	15	15	704	1.000000D+00	0.12
DIXMAANH	1500	18	22	691	1.000000D+00	0.06
DIXMAANH	3000	15	16	689	1.000000D+00	0.13
DIXMAANI	1500	16	24	2757	1.000000D+00	0.20
DIXMAANI	3000	11	11	7062	1.000000D+00	0.92
DIXMAANJ	1500	16	16	4971	1.000000D+00	0.39
DIXMAANJ	3000	18	21	12784	1.000000D+00	1.86
DIXMAANK	1500	25	43	11852	1.000000D+00	0.91
DIXMAANK	3000	17	17	9132	1.000000D+00	1.38
DIXMAANL	1500	21	29	9771	1.000000D+00	0.78
DIXMAANL	3000	17	17	8050	1.000000D+00	1.18
DQDRTIC	1000	6	6	13	4.807773D-32	0.00
DQDRTIC	5000	6	6	13	2.106842D-30	0.01
DQDRTIC	10000	6	6	13	4.441441D-30	0.02

Table A.1: Results of the algorithm TN which does not use negative curvature direction – Part 1

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*time*
DQRTIC	1000	23	23	814	6.455565D-02	0.03
DQRTIC	5000	26	26	3947	1.609143D+00	0.58
DQRTIC	10000	27	27	7866	1.051569D+01	2.08
EDENSCH	1000	14	14	35	6.003285D+03	0.01
EDENSCH	5000	14	15	31	3.000328D+04	0.03
EDENSCH	10000	16	19	37	6.000328D+04	0.08
ENGVAL1	1000	10	10	33	1.108195D+03	0.01
ENGVAL1	5000	10	10	27	5.548668D+03	0.02
ENGVAL1	10000	9	20	26	1.109926D+04	0.06
FLETCBV2	1000	1	1	0	-5.013384D-01	0.00
FLETCBV2	5000	1	1	0	-5.002682D-01	0.00
FLETCBV2	10000	1	1	0	-5.001341D-01	0.00
FLETCBV3	1000	4	4	555	-1.083664D+05	0.02
FLETCBV3	5000	9	9	13937	-2.221834D+07	2.67
FLETCBV3	10000	5	5	14372	-1.321473D+09	5.64
FLETCHCR	1000	1478	1685	27648	3.676966D-17	1.72
FLETCHCR	5000	7334	8325	146111	1.594618D-16	37.98
FLETCHCR	10000	13	15	130	2.591926D-11	0.10
FMINSURF	1024	37	285	9170	1.000000D+00	1.34
FMINSURF	5625	30	257	26526	1.000000D+00	20.54
FMINSURF	10000	36	277	45680	1.000000D+00	62.69
FREUROTH	1000	9	13	33	1.214697D+05	0.01
FREUROTH	5000	10	14	24	6.081592D+05	0.02
FREUROTH	10000	11	16	24	1.216521D+06	0.05
GENHUMPS	1000	17155	119689	16873677	3.252751D+02	937.81
GENHUMPS	5000	–	–	–	–	> 3600
GENHUMPS	10000	–	–	–	–	> 3600
GENROSE	1000	519	1619	81661	1.000000D+00	3.57
GENROSE	5000	2451	8199	1541388	1.000000D+00	295.86
GENROSE	10000	6451	14531	1027183	1.000000D+00	396.17
LIARWHD	1000	14	14	22	1.177518D-14	0.01
LIARWHD	5000	14	14	20	1.556232D-13	0.02
LIARWHD	10000	15	17	21	6.526480D-09	0.04
MOREBV	1000	2	2	374	1.600522D-09	0.01
MOREBV	5000	2	2	603	2.260918D-11	0.11
MOREBV	10000	2	2	738	3.774461D-12	0.24
MSQRTALS	1024	5644	14198	5777101	3.423661D-04	3038.87
MSQRTALS	4900	–	–	–	–	> 3600
MSQRTBLS	1024	2319	11064	2372611	3.364740D-10	1241.25
MSQRTBLS	4900	–	–	–	–	> 3600
NCB20	1010	159	670	79375	9.663298D+02	33.78
NCB20	5010	106	457	41608	-1.394735D+03	102.11
NCB20	10010	13	26	341	-2.642929D+03	1.92
NCB20B	1000	9	11	859	1.676011D+03	0.39
NCB20B	5000	9	13	454	7.351301D+03	1.17
NCB20B	10000	17	25	3540	1.423567D+04	18.32
NONCVXUN	1000	1188	12111	1187000	2.405825D+03	147.97
NONCVXUN	5000	–	–	–	–	> 3600
NONCVXUN	10000	–	–	–	–	> 3600
NONCVXU2	1000	875	8650	874000	2.381367D+03	108.75
NONCVXU2	5000	–	–	–	–	> 3600
NONCVXU2	10000	–	–	–	–	> 3600
NONDIA	1000	7	7	8	5.328548D-12	0.01
NONDIA	5000	5	5	5	9.325523D-09	0.01
NONDIA	10000	5	5	5	5.799705D-10	0.01

Table A.2: Results of the algorithm TN which does not use negative curvature direction – Part 2

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*time*
NONDQUAR	1000	44	124	11367	3.035378D-07	0.33
NONDQUAR	5000	51	135	7591	3.959005D-06	1.01
NONDQUAR	10000	55	146	7631	1.170181D-05	1.99
PENALTY1	1000	41	43	53	9.686175D-03	0.01
PENALTY1	5000	45	46	55	4.929490D-02	0.04
PENALTY1	10000	47	49	57	9.900151D-02	0.08
POWELLSG	1000	20	20	72	1.805168D-08	0.01
POWELLSG	5000	20	20	71	1.155101D-07	0.02
POWELLSG	10000	21	29	72	7.617756D-08	0.05
POWER	1000	31	37	1327	1.532847D-10	0.05
POWER	5000	33	49	3736	3.009006D-10	0.54
POWER	10000	36	64	6058	1.001399D-10	1.69
QUARTC	1000	23	23	814	6.455565D-02	0.03
QUARTC	5000	26	26	3947	1.609143D+00	0.50
QUARTC	10000	27	27	7866	1.051569D+01	1.91
SCHMVETT	1000	6	6	60	-2.994000D+03	0.02
SCHMVETT	5000	6	6	52	-1.499400D+04	0.06
SCHMVETT	10000	5	7	41	-2.999400D+04	0.10
SINQUAD	1000	17	23	33	-2.942505D+05	0.01
SINQUAD	5000	17	20	27	-6.757014D+06	0.04
SINQUAD	10000	21	25	41	-2.642227D+07	0.10
SPARSINE	1000	23	123	16537	6.341091D+05	1.34
SPARSINE	5000	55	352	203006	1.567241D+07	82.12
SPARSINE	10000	175	4207	1570334	6.448147D+07	1362.71
SPARSQUR	1000	20	20	118	6.930295D-09	0.02
SPARSQUR	5000	22	22	129	7.653730D-09	0.09
SPARSQUR	10000	23	23	70	9.235821D-09	0.14
SPMSRTLS	1000	8292	10435	8283328	6.321999D+01	573.30
SPMSRTLS	4999	–	–	–	–	> 3600
SPMSRTLS	10000	41	128	5421	2.937016D+00	3.58
SROSENBR	1000	8	8	9	3.831667D-09	0.00
SROSENBR	5000	8	8	9	1.915834D-08	0.01
SROSENBR	10000	8	18	9	1.844479D-08	0.01
TESTQUAD	1000	9	9	1200	1.638736D-19	0.04
TESTQUAD	5000	10	10	2022	8.895678D-14	0.27
TESTQUAD	10000	11	11	2662	5.122922D-13	0.70
TOINTGSS	1000	4	4	7	1.001002D+01	0.00
TOINTGSS	5000	3	3	3	1.000200D+01	0.00
TOINTGSS	10000	2	3	1	1.000100D+01	0.00
TQUARTIC	1000	2	2	2	7.228532D-24	0.00
TQUARTIC	5000	8	12	9	8.085249D-05	0.01
TQUARTIC	10000	7	11	8	4.812590D-04	0.01
TRIDIA	1000	8	8	592	1.631743D-14	0.02
TRIDIA	5000	10	10	1340	9.481966D-15	0.19
TRIDIA	10000	12	12	2458	2.819989D-20	0.65
VARDIM	1000	18	140	4027	3.072518D-18	0.14
VARDIM	5000	40	141	126509	4.692765D-17	17.93
VARDIM	10000	48	160	322402	2.857023D-16	90.24
VAREIGVL	1000	14	14	1146	7.926475D-09	0.08
VAREIGVL	5000	15	15	921	5.571834D-09	0.30
VAREIGVL	10000	18	28	1795	5.990309D-09	1.11
WOODS	1000	36	42	129	3.976477D-15	0.01
WOODS	4000	52	69	181	1.378622D-11	0.06
WOODS	10000	48	61	157	1.011483D-14	0.13

Table A.3: Results of the algorithm TN which does not use negative curvature direction – Part 3

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
ARWHEAD	1000	6	6	6	1.690288D-10	0	0.00
ARWHEAD	5000	6	6	6	1.110001D-12	0	0.01
ARWHEAD	10000	6	13	6	4.440448D-12	0	0.02
BDQRTIC	1000	13	13	79	3.983818D+03	0	0.02
BDQRTIC	5000	13	13	75	2.000626D+04	0	0.08
BDQRTIC	10000	13	13	75	4.003431D+04	0	0.16
BROYDN7D	1000	88	815	38611	3.047159D+02	59	2.82
BROYDN7D	5000	405	3977	797871	1.659976D+03	230	248.63
BROYDN7D	10000	11	20	65	6.883895D+03	1	0.12
BRYBND	1000	8	8	55	4.530351D-12	0	0.01
BRYBND	5000	15	29	186	1.431842D-13	3	0.20
BRYBND	10000	11	12	67	6.060220D-16	0	0.17
CHAINWOO	1000	114	598	17537	2.514980D+02	30	1.64
CHAINWOO	4000	153	975	65718	1.582515D+03	49	21.11
CHAINWOO	10000	353	2743	577517	2.890864D+03	124	453.78
COSINE	1000	6	6	8	-9.990000D+02	2	0.00
COSINE	5000	6	6	8	-4.999000D+03	2	0.01
COSINE	10000	6	6	8	-9.999000D+03	2	0.02
CRAGGLVY	1000	15	15	144	3.364231D+02	0	0.02
CRAGGLVY	5000	15	15	131	1.688215D+03	0	0.11
CRAGGLVY	10000	17	17	174	3.377956D+03	0	0.31
CURLY10	1000	33	214	11362	-1.003125D+05	19	0.92
CURLY10	5000	22	36	38038	-5.015815D+05	8	14.06
CURLY10	10000	16	42	60984	-1.003163D+06	2	44.82
CURLY20	1000	26	94	10463	-1.003093D+05	12	1.26
CURLY20	5000	26	81	44709	-5.015758D+05	11	25.39
CURLY20	10000	22	89	89161	-1.003162D+06	5	101.19
CURLY30	1000	122	742	113366	-1.000507D+05	45	18.04
CURLY30	5000	24	66	41499	-5.015808D+05	10	31.68
CURLY30	10000	22	50	83642	-1.003162D+06	5	129.58
DIXMAANA	1500	7	7	10	1.000000D+00	0	0.00
DIXMAANA	3000	7	7	10	1.000000D+00	0	0.01
DIXMAANB	1500	8	8	9	1.000000D+00	0	0.00
DIXMAANB	3000	8	8	9	1.000000D+00	0	0.01
DIXMAANC	1500	9	9	11	1.000000D+00	0	0.01
DIXMAANC	3000	9	9	10	1.000000D+00	0	0.01
DIXMAAND	1500	9	9	11	1.000000D+00	0	0.00
DIXMAAND	3000	10	10	12	1.000000D+00	0	0.01
DIXMAANE	1500	55	392	13842	1.000000D+00	37	1.59
DIXMAANE	3000	10	10	401	1.000000D+00	0	0.09
DIXMAANF	1500	19	72	744	1.000000D+00	9	0.10
DIXMAANF	3000	32	223	2519	1.000000D+00	19	0.67
DIXMAANG	1500	36	193	1429	1.000000D+00	22	0.21
DIXMAANG	3000	28	160	1935	1.000000D+00	16	0.51
DIXMAANH	1500	31	143	1258	1.000000D+00	16	0.18
DIXMAANH	3000	19	58	1012	1.000000D+00	6	0.27
DIXMAANI	1500	12	18	4848	1.000000D+00	1	0.54
DIXMAANI	3000	11	11	7062	1.000000D+00	0	1.51
DIXMAANJ	1500	41	220	16204	1.000000D+00	17	2.07
DIXMAANJ	3000	40	128	49606	1.000000D+00	7	12.11
DIXMAANK	1500	60	403	32987	1.000000D+00	35	4.24
DIXMAANK	3000	31	115	27373	1.000000D+00	10	6.69
DIXMAANL	1500	55	395	25739	1.000000D+00	32	3.34
DIXMAANL	3000	28	65	19015	1.000000D+00	6	4.78
DQDRTIC	1000	6	6	13	4.807773D-32	0	0.00
DQDRTIC	5000	6	6	13	2.106842D-30	0	0.01
DQDRTIC	10000	6	6	13	4.441441D-30	0	0.02

Table A.4: Results of the algorithm TN-NC1 which uses negative curvature direction – Part 1

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
DQRTIC	1000	23	23	814	6.455565D-02	0	0.04
DQRTIC	5000	26	26	3947	1.609143D+00	0	0.70
DQRTIC	10000	27	27	7866	1.051569D+01	0	2.68
EDENSCH	1000	14	14	35	6.003285D+03	0	0.01
EDENSCH	5000	14	15	31	3.000328D+04	0	0.03
EDENSCH	10000	17	18	40	6.000328D+04	1	0.10
ENGVAL1	1000	10	10	33	1.108195D+03	0	0.01
ENGVAL1	5000	10	10	27	5.548668D+03	0	0.02
ENGVAL1	10000	9	20	26	1.109926D+04	0	0.05
FLETCBV2	1000	1	1	0	-5.013384D-01	0	0.00
FLETCBV2	5000	1	1	0	-5.002682D-01	0	0.00
FLETCBV2	10000	1	1	0	-5.001341D-01	0	0.00
FLETCBV3	1000	5	6	1121	-3.189503D+03	3	0.08
FLETCBV3	5000	8758	8758	20753	-6.880147D+07	4378	22.56
FLETCBV3	10000	5	5	9680	-1.147714D+09	2	6.22
FLETCHCR	1000	1476	1682	27568	5.746676D-13	1	2.29
FLETCHCR	5000	7338	8351	146138	6.757181D-14	1	55.25
FLETCHCR	10000	13	15	130	2.591926D-11	0	0.11
FMINSURF	1024	37	285	9170	1.000000D+00	0	3.32
FMINSURF	5625	30	257	26526	1.000000D+00	0	40.04
FMINSURF	10000	36	277	45680	1.000000D+00	0	121.36
FREUROTH	1000	10	18	32	1.214697D+05	1	0.01
FREUROTH	5000	11	18	30	6.081592D+05	1	0.03
FREUROTH	10000	11	16	24	1.216521D+06	0	0.06
GENHUMPS	1000	12964	135467	12857731	2.359907D-10	7518	1124.97
GENHUMPS	5000	–	–	–	–	–	> 3600
GENHUMPS	10000	–	–	–	–	–	> 3600
GENROSE	1000	455	2965	19072	1.000000D+00	387	1.41
GENROSE	5000	1263	9720	139446	1.000000D+00	1159	43.45
GENROSE	10000	4287	26124	142729	1.000000D+00	3608	94.96
LIARWHD	1000	14	14	22	1.177518D-14	0	0.01
LIARWHD	5000	14	14	20	1.556232D-13	0	0.02
LIARWHD	10000	15	17	21	6.526480D-09	0	0.04
MOREBV	1000	2	2	374	1.600522D-09	0	0.02
MOREBV	5000	2	2	603	2.260918D-11	0	0.16
MOREBV	10000	2	2	738	3.774461D-12	0	0.38
MSQRTALS	1024	2563	9854	2622210	3.210999D-04	398	2719.62
MSQRTALS	4900	–	–	–	–	–	> 3600
MSQRTBLS	1024	552	3657	563003	1.329878D-06	156	582.05
MSQRTBLS	4900	–	–	–	–	–	> 3600
NCB20	1010	106	1103	36483	9.208174D+02	83	30.93
NCB20	5010	110	1173	18243	-1.447533D+03	92	88.67
NCB20	10010	137	995	4643	-5.313355D+03	124	50.34
NCB20B	1000	19	75	5636	1.676011D+03	4	4.75
NCB20B	5000	9	13	454	7.351301D+03	0	2.23
NCB20B	10000	19	73	2534	1.423567D+04	3	25.90
NONCVXUN	1000	1294	21376	1293000	2.327616D+03	746	302.93
NONCVXUN	5000	–	–	–	–	–	> 3600
NONCVXUN	10000	–	–	–	–	–	> 3600
NONCVXU2	1000	1127	17481	1117982	2.317103D+03	621	260.82
NONCVXU2	5000	–	–	–	–	–	> 3600
NONCVXU2	10000	–	–	–	–	–	> 3600
NONDIA	1000	7	7	8	5.328548D-12	0	0.01
NONDIA	5000	5	5	5	9.325523D-09	0	0.01
NONDIA	10000	5	5	5	5.799705D-10	0	0.01

Table A.5: Results of the algorithm TN-NC1 which uses negative curvature direction – Part 2

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
NONDQUAR	1000	44	124	11367	3.035378D-07	0	0.46
NONDQUAR	5000	51	135	7591	3.959005D-06	0	1.45
NONDQUAR	10000	55	146	7631	1.170181D-05	0	2.92
PENALTY1	1000	41	43	53	9.686175D-03	0	0.01
PENALTY1	5000	45	46	55	4.929490D-02	0	0.05
PENALTY1	10000	47	49	57	9.900151D-02	0	0.09
POWELLSG	1000	20	20	72	1.805168D-08	0	0.01
POWELLSG	5000	20	20	71	1.155101D-07	0	0.03
POWELLSG	10000	21	29	72	7.617756D-08	0	0.05
POWER	1000	31	37	1327	1.532847D-10	0	0.06
POWER	5000	33	49	3736	3.009006D-10	0	0.76
POWER	10000	36	64	6058	1.001399D-10	0	2.40
QUARTC	1000	23	23	814	6.455565D-02	0	0.04
QUARTC	5000	26	26	3947	1.609143D+00	0	0.65
QUARTC	10000	27	27	7866	1.051569D+01	0	2.51
SCHMVETT	1000	6	6	60	-2.994000D+03	0	0.03
SCHMVETT	5000	6	6	52	-1.499400D+04	0	0.11
SCHMVETT	10000	5	7	41	-2.999400D+04	0	0.17
SINQUAD	1000	19	35	44	-2.942505D+05	9	0.02
SINQUAD	5000	16	27	30	-6.757014D+06	6	0.05
SINQUAD	10000	13	17	24	-2.642315D+07	4	0.09
SPARSINE	1000	952	10241	901142	1.618616D+05	566	127.99
SPARSINE	5000	42	358	121002	1.697258D+07	13	90.18
SPARSINE	10000	55	438	302382	7.005234D+07	14	479.08
SPARSQUR	1000	20	20	118	6.930295D-09	0	0.03
SPARSQUR	5000	22	22	129	7.653730D-09	0	0.13
SPARSQUR	10000	23	23	70	9.235821D-09	0	0.19
SPMSRTLS	1000	128	1248	61408	5.608497D-02	84	7.32
SPMSRTLS	4999	–	–	–	–	–	> 3600
SPMSRTLS	10000	26	114	1439	3.409363D-11	9	1.78
SROSENBR	1000	8	8	9	3.831667D-09	0	0.00
SROSENBR	5000	8	8	9	1.915834D-08	0	0.01
SROSENBR	10000	8	18	9	1.844479D-08	0	0.02
TESTQUAD	1000	9	9	1200	1.638736D-19	0	0.05
TESTQUAD	5000	10	10	2022	8.895678D-14	0	0.36
TESTQUAD	10000	11	11	2662	5.122922D-13	0	0.93
TOINTGSS	1000	4	4	7	1.001002D+01	0	0.00
TOINTGSS	5000	3	3	3	1.000200D+01	0	0.01
TOINTGSS	10000	2	3	1	1.000100D+01	0	0.00
TQUARTIC	1000	2	2	2	7.228532D-24	0	0.00
TQUARTIC	5000	8	18	10	3.582716D-08	1	0.01
TQUARTIC	10000	7	11	8	4.812590D-04	0	0.02
TRIDIA	1000	8	8	592	1.631743D-14	0	0.03
TRIDIA	5000	10	10	1340	9.481966D-15	0	0.25
TRIDIA	10000	12	12	2458	2.819989D-20	0	0.90
VARDIM	1000	18	140	4027	3.072518D-18	0	0.18
VARDIM	5000	40	141	126509	4.692765D-17	0	25.49
VARDIM	10000	49	240	300755	5.942719D-17	0	119.93
VAREIGVL	1000	14	14	1146	7.926475D-09	0	0.13
VAREIGVL	5000	15	15	921	5.571834D-09	0	0.50
VAREIGVL	10000	15	34	110	6.162802D-16	3	0.17
WOODS	1000	37	63	133	2.352916D-13	3	0.01
WOODS	4000	46	69	158	2.110406D-12	7	0.07
WOODS	10000	47	66	153	2.271874D-15	7	0.15

Table A.6: Results of the algorithm TN-NC1 which uses negative curvature direction – Part 3

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
ARWHEAD	1000	6	6	6	1.690288D-10	0	0.00
ARWHEAD	5000	6	6	6	1.110001D-12	0	0.01
ARWHEAD	10000	6	13	6	4.440448D-12	0	0.02
BDQRTIC	1000	13	13	79	3.983818D+03	0	0.02
BDQRTIC	5000	13	13	75	2.000626D+04	0	0.08
BDQRTIC	10000	13	13	75	4.003431D+04	0	0.16
BROYDN7D	1000	118	1193	62826	3.918212D+02	89	4.52
BROYDN7D	5000	535	4976	709815	1.949549D+03	485	206.70
BROYDN7D	10000	11	20	65	6.883895D+03	1	0.10
BRYBND	1000	8	8	55	4.530351D-12	0	0.01
BRYBND	5000	15	29	186	1.431842D-13	3	0.18
BRYBND	10000	11	12	67	6.060220D-16	0	0.14
CHAINWOO	1000	130	675	22818	3.333991D+02	35	1.82
CHAINWOO	4000	164	1212	105743	1.249443D+03	64	31.70
CHAINWOO	10000	276	2288	384586	3.465589D+03	119	285.69
COSINE	1000	6	6	8	-9.990000D+02	2	0.00
COSINE	5000	6	6	8	-4.999000D+03	2	0.01
COSINE	10000	6	6	8	-9.999000D+03	2	0.02
CRAGGLVY	1000	15	15	144	3.364231D+02	0	0.02
CRAGGLVY	5000	15	15	131	1.688215D+03	0	0.09
CRAGGLVY	10000	17	17	174	3.377956D+03	0	0.24
CURLY10	1000	32	196	13857	-1.003125D+05	20	1.02
CURLY10	5000	27	69	38270	-5.015777D+05	12	13.33
CURLY10	10000	16	42	60984	-1.003163D+06	2	42.18
CURLY20	1000	33	238	17797	-1.003100D+05	22	1.97
CURLY20	5000	32	146	52112	-5.015599D+05	16	27.98
CURLY20	10000	19	44	76689	-1.003163D+06	2	82.78
CURLY30	1000	54	388	44582	-1.001797D+05	35	6.55
CURLY30	5000	24	48	47539	-5.015815D+05	9	34.25
CURLY30	10000	22	42	86066	-1.003162D+06	4	123.96
DIXMAANA	1500	7	7	10	1.000000D+00	0	0.00
DIXMAANA	3000	7	7	10	1.000000D+00	0	0.01
DIXMAANB	1500	8	8	9	1.000000D+00	0	0.00
DIXMAANB	3000	8	8	9	1.000000D+00	0	0.01
DIXMAANC	1500	9	9	11	1.000000D+00	0	0.00
DIXMAANC	3000	9	9	10	1.000000D+00	0	0.01
DIXMAAND	1500	9	9	11	1.000000D+00	0	0.00
DIXMAAND	3000	10	10	12	1.000000D+00	0	0.01
DIXMAANE	1500	66	485	25372	1.000000D+00	55	2.62
DIXMAANE	3000	10	10	401	1.000000D+00	0	0.09
DIXMAANF	1500	19	73	1030	1.000000D+00	9	0.13
DIXMAANF	3000	27	145	1925	1.000000D+00	15	0.47
DIXMAANG	1500	30	148	1734	1.000000D+00	18	0.21
DIXMAANG	3000	23	107	1832	1.000000D+00	11	0.44
DIXMAANH	1500	24	89	945	1.000000D+00	11	0.12
DIXMAANH	3000	22	79	1277	1.000000D+00	8	0.30
DIXMAANI	1500	12	18	4848	1.000000D+00	1	0.49
DIXMAANI	3000	11	11	7062	1.000000D+00	0	1.44
DIXMAANJ	1500	56	364	34229	1.000000D+00	33	3.93
DIXMAANJ	3000	46	251	45955	1.000000D+00	23	10.36
DIXMAANK	1500	68	509	29825	1.000000D+00	47	3.44
DIXMAANK	3000	38	202	48353	1.000000D+00	16	10.90
DIXMAANL	1500	51	375	28045	1.000000D+00	36	3.25
DIXMAANL	3000	34	125	19781	1.000000D+00	16	4.47
DQDRTIC	1000	6	6	13	4.807773D-32	0	0.00
DQDRTIC	5000	6	6	13	2.106842D-30	0	0.01
DQDRTIC	10000	6	6	13	4.441441D-30	0	0.02

Table A.7: Results of the algorithm TN-NC2 which uses negative curvature direction – Part 1

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
DQRTIC	1000	23	23	814	6.455565D-02	0	0.03
DQRTIC	5000	26	26	3947	1.609143D+00	0	0.64
DQRTIC	10000	27	27	7866	1.051569D+01	0	2.48
EDENSCH	1000	14	14	35	6.003285D+03	0	0.01
EDENSCH	5000	14	15	31	3.000328D+04	0	0.03
EDENSCH	10000	17	18	40	6.000328D+04	1	0.07
ENGVAL1	1000	10	10	33	1.108195D+03	0	0.01
ENGVAL1	5000	10	10	27	5.548668D+03	0	0.02
ENGVAL1	10000	9	20	26	1.109926D+04	0	0.03
FLETCBV2	1000	1	1	0	-5.013384D-01	0	0.00
FLETCBV2	5000	1	1	0	-5.002682D-01	0	0.00
FLETCBV2	10000	1	1	0	-5.001341D-01	0	0.00
FLETCBV3	1000	6	6	1323	-1.038422D+04	3	0.08
FLETCBV3	5000	5567	5567	13195	-5.273278D+07	2840	12.16
FLETCBV3	10000	5	5	9680	-9.857319D+08	1	5.91
FLETCHCR	1000	1476	1682	27568	5.746676D-13	1	2.08
FLETCHCR	5000	7338	8351	146138	6.757181D-14	1	48.44
FLETCHCR	10000	13	15	130	2.591926D-11	0	0.10
FMINSURF	1024	37	285	9170	1.000000D+00	0	2.23
FMINSURF	5625	30	257	26526	1.000000D+00	0	36.25
FMINSURF	10000	36	277	45680	1.000000D+00	0	116.85
FREUROTH	1000	10	18	32	1.214697D+05	1	0.01
FREUROTH	5000	11	18	30	6.081592D+05	1	0.03
FREUROTH	10000	11	16	24	1.216521D+06	0	0.05
GENHUMPS	1000	10472	121609	10353839	5.929871D-11	8651	792.12
GENHUMPS	5000	–	–	–	–	–	> 3600
GENHUMPS	10000	–	–	–	–	–	> 3600
GENROSE	1000	474	3009	33425	1.000000D+00	417	2.32
GENROSE	5000	1526	10358	115887	1.000000D+00	1402	36.25
GENROSE	10000	4484	27144	151983	1.000000D+00	3888	99.17
LIARWHD	1000	14	14	22	1.177518D-14	0	0.01
LIARWHD	5000	14	14	20	1.556232D-13	0	0.02
LIARWHD	10000	15	17	21	6.526480D-09	0	0.04
MOREBV	1000	2	2	374	1.600522D-09	0	0.02
MOREBV	5000	2	2	603	2.260918D-11	0	0.15
MOREBV	10000	2	2	738	3.774461D-12	0	0.37
MSQRTALS	1024	413	3745	420485	4.327283D-10	237	433.83
MSQRTALS	4900	–	–	–	–	–	> 3600
MSQRTBLS	1024	374	3332	380548	6.404756D-12	206	391.64
MSQRTBLS	4900	–	–	–	–	–	> 3600
NCB20	1010	276	1936	85039	9.063045D+02	248	71.15
NCB20	5010	85	688	16166	-1.472158D+03	73	77.45
NCB20	10010	160	916	8840	-5.275134D+03	148	92.16
NCB20B	1000	21	103	6405	1.676011D+03	9	5.36
NCB20B	5000	20	122	5368	7.351301D+03	10	25.79
NCB20B	10000	19	73	2534	1.423567D+04	3	25.89
NONCVXUN	1000	1548	21219	1547000	2.325729D+03	1093	380.37
NONCVXUN	5000	–	–	–	–	–	> 3600
NONCVXUN	10000	–	–	–	–	–	> 3600
NONCVXU2	1000	896	12067	894904	2.333203D+03	629	210.15
NONCVXU2	5000	–	–	–	–	–	> 3600
NONCVXU2	10000	–	–	–	–	–	> 3600
NONDIA	1000	7	7	8	5.328548D-12	0	0.01
NONDIA	5000	5	5	5	9.325523D-09	0	0.01
NONDIA	10000	5	5	5	5.799705D-10	0	0.01

Table A.8: Results of the algorithm TN-NC2 which uses negative curvature direction – Part 2

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
NONDQUAR	1000	44	124	11367	3.035378D-07	0	0.46
NONDQUAR	5000	51	135	7591	3.959005D-06	0	1.43
NONDQUAR	10000	55	146	7631	1.170181D-05	0	2.86
PENALTY1	1000	41	43	53	9.686175D-03	0	0.01
PENALTY1	5000	45	46	55	4.929490D-02	0	0.04
PENALTY1	10000	47	49	57	9.900151D-02	0	0.09
POWELLSG	1000	20	20	72	1.805168D-08	0	0.01
POWELLSG	5000	20	20	71	1.155101D-07	0	0.02
POWELLSG	10000	21	29	72	7.617756D-08	0	0.05
POWER	1000	31	37	1327	1.532847D-10	0	0.06
POWER	5000	33	49	3736	3.009006D-10	0	0.74
POWER	10000	36	64	6058	1.001399D-10	0	2.37
QUARTC	1000	23	23	814	6.455565D-02	0	0.04
QUARTC	5000	26	26	3947	1.609143D+00	0	0.64
QUARTC	10000	27	27	7866	1.051569D+01	0	2.49
SCHMVETT	1000	6	6	60	-2.994000D+03	0	0.03
SCHMVETT	5000	6	6	52	-1.499400D+04	0	0.11
SCHMVETT	10000	5	7	41	-2.999400D+04	0	0.16
SINQUAD	1000	19	35	44	-2.942505D+05	9	0.02
SINQUAD	5000	16	27	30	-6.757014D+06	6	0.05
SINQUAD	10000	–	–	–	–	–	–
SPARSINE	1000	2571	28325	2469982	2.806062D-06	2025	346.66
SPARSINE	5000	369	4084	1002629	1.626756D+07	254	731.20
SPARSINE	10000	165	1881	855419	6.961817D+07	97	1337.03
SPARSQUR	1000	20	20	118	6.930295D-09	0	0.03
SPARSQUR	5000	22	22	129	7.653730D-09	0	0.13
SPARSQUR	10000	23	23	70	9.235821D-09	0	0.19
SPMSRTLS	1000	538	2470	507351	1.136539D+01	155	59.88
SPMSRTLS	4999	–	–	–	–	–	> 3600
SPMSRTLS	10000	26	167	1300	1.561664D-12	15	1.58
SROSENBR	1000	8	8	9	3.831667D-09	0	0.00
SROSENBR	5000	8	8	9	1.915834D-08	0	0.01
SROSENBR	10000	8	18	9	1.844479D-08	0	0.01
TESTQUAD	1000	9	9	1200	1.638736D-19	0	0.05
TESTQUAD	5000	10	10	2022	8.895678D-14	0	0.36
TESTQUAD	10000	11	11	2662	5.122922D-13	0	0.93
TOINTGSS	1000	4	4	7	1.001002D+01	0	0.00
TOINTGSS	5000	3	3	3	1.000200D+01	0	0.01
TOINTGSS	10000	2	3	1	1.000100D+01	0	0.00
TQUARTIC	1000	2	2	2	7.228532D-24	0	0.00
TQUARTIC	5000	8	18	10	3.582716D-08	1	0.01
TQUARTIC	10000	7	11	8	4.812590D-04	0	0.02
TRIDIA	1000	8	8	592	1.631743D-14	0	0.02
TRIDIA	5000	10	10	1340	9.481966D-15	0	0.25
TRIDIA	10000	12	12	2458	2.819989D-20	0	0.89
VARDIM	1000	18	140	4027	3.072518D-18	0	0.18
VARDIM	5000	40	141	126509	4.692765D-17	0	25.96
VARDIM	10000	49	240	300755	5.942719D-17	0	121.87
VAREIGVL	1000	14	14	1146	7.926475D-09	0	0.13
VAREIGVL	5000	15	15	921	5.571834D-09	0	0.50
VAREIGVL	10000	15	34	110	6.162802D-16	3	0.17
WOODS	1000	37	63	133	2.352916D-13	3	0.02
WOODS	4000	46	69	158	2.110406D-12	7	0.06
WOODS	10000	47	66	153	2.271874D-15	7	0.15

Table A.9: Results of the algorithm TN-NC2 which uses negative curvature direction – Part 3

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
ARWHEAD	1000	6	6	6	1.690288D-10	0	0.00
ARWHEAD	5000	6	6	6	1.110001D-12	0	0.01
ARWHEAD	10000	6	13	6	4.440448D-12	0	0.02
BDQRTIC	1000	13	13	79	3.983818D+03	0	0.02
BDQRTIC	5000	13	13	75	2.000626D+04	0	0.08
BDQRTIC	10000	13	13	75	4.003431D+04	0	0.15
BROYDN7D	1000	31	124	11736	2.460676D+02	13	0.84
BROYDN7D	5000	38	160	36259	1.415682D+03	22	10.72
BROYDN7D	10000	11	20	65	6.883895D+03	1	0.09
BRYBND	1000	8	8	55	4.530351D-12	0	0.01
BRYBND	5000	15	29	186	1.431842D-13	3	0.17
BRYBND	10000	11	12	67	6.060220D-16	0	0.13
CHAINWOO	1000	143	744	31837	3.298250D+02	43	2.52
CHAINWOO	4000	161	931	112160	1.159655D+03	50	33.04
CHAINWOO	10000	169	945	125135	8.922417D+02	54	90.96
COSINE	1000	6	6	8	-9.990000D+02	2	0.00
COSINE	5000	6	6	8	-4.999000D+03	2	0.01
COSINE	10000	6	6	8	-9.999000D+03	2	0.02
CRAGGLVY	1000	15	15	144	3.364231D+02	0	0.02
CRAGGLVY	5000	15	15	131	1.688215D+03	0	0.09
CRAGGLVY	10000	17	17	174	3.377956D+03	0	0.24
CURLY10	1000	22	37	8991	-1.003157D+05	9	0.67
CURLY10	5000	22	45	36107	-5.015815D+05	9	12.57
CURLY10	10000	16	42	60984	-1.003163D+06	2	42.10
CURLY20	1000	21	38	8919	-1.003163D+05	8	0.99
CURLY20	5000	26	60	47582	-5.015682D+05	11	25.72
CURLY20	10000	19	35	76231	-1.003163D+06	1	83.02
CURLY30	1000	36	218	18401	-1.003119D+05	20	2.72
CURLY30	5000	24	48	47539	-5.015815D+05	9	34.21
CURLY30	10000	22	43	95292	-1.003163D+06	4	137.47
DIXMAANA	1500	7	7	10	1.000000D+00	0	0.00
DIXMAANA	3000	7	7	10	1.000000D+00	0	0.01
DIXMAANB	1500	8	8	9	1.000000D+00	0	0.00
DIXMAANB	3000	8	8	9	1.000000D+00	0	0.01
DIXMAANC	1500	9	9	11	1.000000D+00	0	0.00
DIXMAANC	3000	9	9	10	1.000000D+00	0	0.01
DIXMAAND	1500	9	9	11	1.000000D+00	0	0.01
DIXMAAND	3000	10	10	12	1.000000D+00	0	0.01
DIXMAANE	1500	11	15	1797	1.000000D+00	3	0.19
DIXMAANE	3000	10	10	401	1.000000D+00	0	0.09
DIXMAANF	1500	21	83	1112	1.000000D+00	10	0.14
DIXMAANF	3000	18	47	1088	1.000000D+00	6	0.26
DIXMAANG	1500	26	115	1511	1.000000D+00	15	0.19
DIXMAANG	3000	21	58	1162	1.000000D+00	7	0.28
DIXMAANH	1500	30	138	1346	1.000000D+00	17	0.17
DIXMAANH	3000	29	117	1612	1.000000D+00	15	0.39
DIXMAANI	1500	12	18	4848	1.000000D+00	1	0.50
DIXMAANI	3000	11	11	7062	1.000000D+00	0	1.40
DIXMAANJ	1500	40	200	21845	1.000000D+00	20	2.52
DIXMAANJ	3000	23	83	10082	1.000000D+00	11	2.29
DIXMAANK	1500	34	153	18163	1.000000D+00	16	2.09
DIXMAANK	3000	39	264	54797	1.000000D+00	25	12.87
DIXMAANL	1500	19	56	4006	1.000000D+00	7	0.47
DIXMAANL	3000	50	287	44881	1.000000D+00	30	10.15
DQDRTIC	1000	6	6	13	4.807773D-32	0	0.00
DQDRTIC	5000	6	6	13	2.106842D-30	0	0.01
DQDRTIC	10000	6	6	13	4.441441D-30	0	0.02

Table A.10: Results of the algorithm TN-NC3 which uses negative curvature direction – Part 1

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
DQRTIC	1000	23	23	814	6.455565D-02	0	0.03
DQRTIC	5000	26	26	3947	1.609143D+00	0	0.64
DQRTIC	10000	27	27	7866	1.051569D+01	0	2.48
EDENSCH	1000	14	14	35	6.003285D+03	0	0.01
EDENSCH	5000	14	15	31	3.000328D+04	0	0.03
EDENSCH	10000	17	18	40	6.000328D+04	1	0.07
ENGVAL1	1000	10	10	33	1.108195D+03	0	0.01
ENGVAL1	5000	10	10	27	5.548668D+03	0	0.02
ENGVAL1	10000	9	20	26	1.109926D+04	0	0.04
FLETCBV2	1000	1	1	0	-5.013384D-01	0	0.00
FLETCBV2	5000	1	1	0	-5.002682D-01	0	0.00
FLETCBV2	10000	1	1	0	-5.001341D-01	0	0.00
FLETCBV3	1000	5	5	501	-2.079032D+03	2	0.03
FLETCBV3	5000	8698	8698	10530	-6.833002D+07	4348	16.26
FLETCBV3	10000	5	5	9680	-9.857319D+08	1	5.64
FLETCHCR	1000	1476	1682	27568	5.746676D-13	1	2.08
FLETCHCR	5000	7338	8351	146138	6.757181D-14	1	48.32
FLETCHCR	10000	13	15	130	2.591926D-11	0	0.10
FMINSURF	1024	37	285	9170	1.000000D+00	0	2.23
FMINSURF	5625	30	257	26526	1.000000D+00	0	36.86
FMINSURF	10000	36	277	45680	1.000000D+00	0	113.88
FREUROTH	1000	10	18	32	1.214697D+05	1	0.01
FREUROTH	5000	11	18	30	6.081592D+05	1	0.03
FREUROTH	10000	11	16	24	1.216521D+06	0	0.05
GENHUMPS	1000	4315	23860	4200513	2.140010D-10	2126	313.22
GENHUMPS	5000	–	–	–	–	–	> 3600
GENHUMPS	10000	–	–	–	–	–	> 3600
GENROSE	1000	506	2893	16262	1.000000D+00	419	1.22
GENROSE	5000	1517	9729	66684	1.000000D+00	1381	21.59
GENROSE	10000	4664	26843	143374	1.000000D+00	3968	94.67
LIARWHD	1000	14	14	22	1.177518D-14	0	0.01
LIARWHD	5000	14	14	20	1.556232D-13	0	0.02
LIARWHD	10000	15	17	21	6.526480D-09	0	0.04
MOREBV	1000	2	2	374	1.600522D-09	0	0.02
MOREBV	5000	2	2	603	2.260918D-11	0	0.15
MOREBV	10000	2	2	738	3.774461D-12	0	0.37
MSQRTALS	1024	213	2072	214535	2.453652D-10	186	219.79
MSQRTALS	4900	–	–	–	–	–	> 3600
MSQRTBLS	1024	641	3084	653017	1.054637D-06	193	671.55
MSQRTBLS	4900	54	365	162001	1.761111D-09	24	1885.52
NCB20	1010	47	176	11836	9.036323D+02	40	10.07
NCB20	5010	39	154	9836	-1.478669D+03	25	47.08
NCB20	10010	96	300	2436	-5.229099D+03	84	26.56
NCB20B	1000	19	95	6157	1.676011D+03	6	5.15
NCB20B	5000	19	100	5189	7.351301D+03	9	24.81
NCB20B	10000	19	73	2534	1.423567D+04	3	26.20
NONCVXUN	1000	306	2708	304021	2.325970D+03	180	72.16
NONCVXUN	5000	–	–	–	–	–	> 3600
NONCVXUN	10000	–	–	–	–	–	> 3600
NONCVXU2	1000	304	2598	298606	2.320695D+03	200	68.99
NONCVXU2	5000	–	–	–	–	–	> 3600
NONCVXU2	10000	–	–	–	–	–	> 3600
NONDIA	1000	7	7	8	5.328548D-12	0	0.01
NONDIA	5000	5	5	5	9.325523D-09	0	0.01
NONDIA	10000	5	5	5	5.799705D-10	0	0.01

Table A.11: Results of the algorithm TN-NC3 which uses negative curvature direction – Part 2

PROBLEM	*n*	*it*	*f-eval*	*inner-it*	*function value*	*neg. curv.*	*time*
NONDQUAR	1000	44	124	11367	3.035378D-07	0	0.46
NONDQUAR	5000	51	135	7591	3.959005D-06	0	1.43
NONDQUAR	10000	55	146	7631	1.170181D-05	0	2.86
PENALTY1	1000	41	43	53	9.686175D-03	0	0.01
PENALTY1	5000	45	46	55	4.929490D-02	0	0.05
PENALTY1	10000	47	49	57	9.900151D-02	0	0.09
POWELLSG	1000	20	20	72	1.805168D-08	0	0.01
POWELLSG	5000	20	20	71	1.155101D-07	0	0.02
POWELLSG	10000	21	29	72	7.617756D-08	0	0.05
POWER	1000	31	37	1327	1.532847D-10	0	0.06
POWER	5000	33	49	3736	3.009006D-10	0	0.75
POWER	10000	36	64	6058	1.001399D-10	0	2.39
QUARTC	1000	23	23	814	6.455565D-02	0	0.03
QUARTC	5000	26	26	3947	1.609143D+00	0	0.64
QUARTC	10000	27	27	7866	1.051569D+01	0	2.48
SCHMVETT	1000	6	6	60	-2.994000D+03	0	0.03
SCHMVETT	5000	6	6	52	-1.499400D+04	0	0.11
SCHMVETT	10000	5	7	41	-2.999400D+04	0	0.16
SINQUAD	1000	19	35	44	-2.942505D+05	9	0.02
SINQUAD	5000	16	27	30	-6.757014D+06	6	0.05
SINQUAD	10000	–	–	–	–	–	–
SPARSINE	1000	256	2390	251060	1.186251D-07	209	35.34
SPARSINE	5000	419	3725	2073882	6.356014D-07	299	1501.92
SPARSINE	10000	–	–	–	–	–	> 3600
SPARSQUR	1000	20	20	118	6.930295D-09	0	0.04
SPARSQUR	5000	22	22	129	7.653730D-09	0	0.13
SPARSQUR	10000	23	23	70	9.235821D-09	0	0.19
SPMSRTLS	1000	112	837	54826	3.711627D-11	92	6.48
SPMSRTLS	4999	36	186	7818	3.165356D-01	20	4.43
SPMSRTLS	10000	22	133	1483	3.352784D-08	11	1.74
SROSENBR	1000	8	8	9	3.831667D-09	0	0.00
SROSENBR	5000	8	8	9	1.915834D-08	0	0.01
SROSENBR	10000	8	18	9	1.844479D-08	0	0.01
TESTQUAD	1000	9	9	1200	1.638736D-19	0	0.05
TESTQUAD	5000	10	10	2022	8.895678D-14	0	0.36
TESTQUAD	10000	11	11	2662	5.122922D-13	0	0.93
TOINTGSS	1000	4	4	7	1.001002D+01	0	0.00
TOINTGSS	5000	3	3	3	1.000200D+01	0	0.01
TOINTGSS	10000	2	3	1	1.000100D+01	0	0.00
TQUARTIC	1000	2	2	2	7.228532D-24	0	0.00
TQUARTIC	5000	8	18	10	3.582716D-08	1	0.01
TQUARTIC	10000	7	11	8	4.812590D-04	0	0.02
TRIDIA	1000	8	8	592	1.631743D-14	0	0.03
TRIDIA	5000	10	10	1340	9.481966D-15	0	0.25
TRIDIA	10000	12	12	2458	2.819989D-20	0	0.89
VARDIM	1000	18	140	4027	3.072518D-18	0	0.18
VARDIM	5000	40	141	126509	4.692765D-17	0	25.07
VARDIM	10000	49	240	300755	5.942719D-17	0	117.72
VAREIGVL	1000	14	14	1146	7.926475D-09	0	0.13
VAREIGVL	5000	15	15	921	5.571834D-09	0	0.50
VAREIGVL	10000	15	34	110	6.162802D-16	3	0.17
WOODS	1000	37	63	133	2.352916D-13	3	0.02
WOODS	4000	46	69	158	2.110406D-12	7	0.06
WOODS	10000	47	66	153	2.271874D-15	7	0.14

Table A.12: Results of the algorithm TN-NC3 which uses negative curvature direction – Part 3

B Appendix

B.1 Quality Profiles for multi–objective smooth optimization

In this section we want to extend the definition of Quality Profiles also to smooth multi–objective optimization problems, so that we need some cares to generalize what detailed in Section 9.3. On this purpose, let us consider the set of solvers $\mathcal{S}$ and the set of multi–objective benchmark problems $\mathcal{P}$. Then, following a similar taxonomy reported in Section 9.3, we denote by

- $f_i^{(p)}(x)$ the i-th objective function of the test problem $p \in \mathcal{P}$, $i = 1, \ldots, m$;
- $f_{is}^{(p)}(x^*)$ the optimal value of the i-th objective function of the test problem $p \in \mathcal{P}$, $i = 1, \ldots, m$, obtained by the solver s;
- $f_{iL}^{(p)}$ the reference value of the i-th objective function of the test problem $p \in \mathcal{P}$, $i = 1, \ldots, m$.

For any $s \in \mathcal{S}$ and for any $\tau \in [0, 1]$ we define the vector

$$Q_s(\tau) = \begin{bmatrix} Q_s^{(1)}(\tau) \\ \vdots \\ Q_s^{(m)}(\tau) \end{bmatrix} \in \mathbb{R}^m$$

according with the following relations

$$\begin{cases} Q_s^{(1)}(\tau) = \dfrac{1}{|\mathcal{P}|} size\left\{p \in \mathcal{P} \ : \ f_{1s}^{(p)}(x^*) - f_{1L}^{(p)} \leq \tau \left[f_1^{(p)}(x_0^{(p)}) - f_{1L}^{(p)}\right]\right\} \\ \qquad \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\ Q_s^{(m)}(\tau) = \dfrac{1}{|\mathcal{P}|} size\left\{p \in \mathcal{P} \ : \ f_{ms}^{(p)}(x^*) - f_{mL}^{(p)} \leq \tau \left[f_m^{(p)}(x_0^{(p)}) - f_{mL}^{(p)}\right]\right\}. \end{cases}$$

Then, for any solver $s \in \mathcal{S}$, when τ ranges in $[0, 1]$ we obtain the sequence of m dimensional vectors

$$\{Q_s(\tau)\},$$

that can be used to build the two fronts $L_s^{\min}$ and $L_s^{\max}$ as follows:

$$L_s^{\min} \stackrel{\text{def}}{=} \left\{Q \in \{Q_s(\tau)\},\ \tau \in [0,1] \ : \ \nexists\, \bar{Q} \in \{Q_s(\tau)\} \text{ s.t. } \bar{Q}^{(i)} \leq Q^{(i)},\ i = 1, \ldots, m, \right.$$
$$\left. \text{and } \bar{Q}^{(j)} < Q^{(j)} \text{ for at least an index } j \in \{1, \ldots, m\}\right\},$$

$$L_s^{\max} \stackrel{\text{def}}{=} \left\{Q \in \{Q_s(\tau)\},\ \tau \in [0,1] \ : \ \nexists\, \bar{Q} \in \{Q_s(\tau)\} \text{ s.t. } \bar{Q}^{(i)} \geq Q^{(i)},\ i = 1, \ldots, m, \right.$$
$$\left. \text{and } \bar{Q}^{(j)} > Q^{(j)} \text{ for at least an index } j \in \{1, \ldots, m\}\right\},$$

Observe that according with the definitions of $L_s^{\min}$ and $L_s^{\max}$, we can indicate that for any solver $s \in \mathcal{S}$ we have $L_s^{\min} \preceq L_s^{\max}$, meaning that

$$L_s^{\min} \preceq L_s^{\max} \qquad \Longleftrightarrow \qquad \text{for all } \bar{Q} \in L_s^{\min},\ \text{for all } \tilde{Q} \in L_s^{\max},\ \bar{Q}^{(i)} \leq \tilde{Q}^{(i)},\ i = 1, \ldots, m.$$

As a more general statement, given $s_1, s_2 \in \mathcal{S}$, $s_1 \neq s_2$, we say that:

- $L_{s_1}^* \preceq L_{s_2}^{**}$, where "$*$" and "$**$" may be either "min" or "max", if

$$\text{for all } \bar{Q} \in L_{s_1}^*,\ \text{for all } \widetilde{Q} \in L_{s_2}^{**},\ \bar{Q}^{(i)} \leq \widetilde{Q}^{(i)},\ i = 1, \ldots, m;$$

- $L_{s_1}^* \prec L_{s_2}^{**}$, where "$*$" and "$**$" may be either "min" or "max", if

$$\text{for all } \bar{Q} \in L_{s_1}^*,\ \text{for all } \widetilde{Q} \in L_{s_2}^{**},\ \bar{Q}^{(i)} < \widetilde{Q}^{(i)},\ i = 1, \ldots, m.$$

Finally, compare two selected solvers $s_1, s_2 \in \mathcal{S}$, according with the next taxonomy:

- if $L^{\min}_{s_1} \prec L^{\min}_{s_2}$, then we say that s_1 min–*dominates* s_2;
- if $L^{\min}_{s_1} \preceq L^{\min}_{s_2}$, then we say that s_1 min–*weakly dominates* s_2;
- if $L^{\max}_{s_1} \prec L^{\max}_{s_2}$, then we say that s_2 max–*dominates* s_1;
- if $L^{\max}_{s_1} \preceq L^{\max}_{s_2}$, then we say that s_2 max–*weakly dominates* s_1.

Of course, for any pair of solvers s_1, s_2 there is the chance that none of the last four cases possibly holds, because the fronts $L^{\min}_{s_1}, L^{\min}_{s_2}$ and/or $L^{\max}_{s_1}, L^{\max}_{s_2}$ might show several intersection points. This can be interpreted by saying that the above four cases may hold in just some sub–intervals of values of $\tau \in [0, 1]$. Hence, as an example we can conclude that the solver s_1 can be min–*dominated* / min–*weakly dominated* (respectively max–*dominated* / max–*weakly dominated*) by the solver s_2 in the sub–interval of values $\tau \in [\tau_1, \tau_2] \subseteq [0, 1]$.

Note that as regards the comments in Observation 9.5, some additional care should be considered in the multi–objective framework.

B.2 Quality Profiles for derivative–free optimization

As for the derivative–free frameworks, where an n dimensional non–smooth function is minimized, we know that *Data Profiles* [36] are an effective tool to measure performances, depending on different budgets (i.e. number of function evaluations) allowed when running the different solvers. In particular, each data profile basically represents a performance profile among solvers (where in the abscissa axis we consider the number of simplex evaluations) built using a partial information. The last partial information is obtained for each solver $s \in \mathcal{S}$, and represents the computational effort (i.e. the number of function evaluations) needed by s to solve different fractions of the test set, assuming that the accuracy required for the solution is given.

As an example, we can plot several data profiles, being each plot with respect to the number of function evaluations (one simplex evaluation is equivalent to $n + 1$ function evaluations), and the relative accuracy is given by setting a value $\sigma_i \in [0, 1]$. The value σ_i is used to compute the performances (number of function evaluations) of each solver s, on the set of benchmark problems $\mathcal{P}$, so that (borrowing the taxonomy used in Section 9.3)

$$f_s^{(p)}(x^*) - f_L^{(p)} \leq \sigma_i \left[f^{(p)}(x_0^{(p)}) - f_L^{(p)} \right]. \tag{B.1}$$

This yields a sequence of ℓ data profile plots, each of which is associated to a different precision level

$$\sigma_1 < \cdots < \sigma_\ell.$$

The basic idea of adopting quality profiles for derivative–free frameworks is that of *reversing the rationale behind data profiles*. Indeed, each quality profile is explicitly built in accordance with a relation similar to (B.1). Hence, we might conceive a sequence of quality profiles where, from a plot to another one, the budget allowed for the computation (by each solver) changes. Hence, considering e.g. the above problem of comparing the performances in terms of function evaluations, we might set the budget levels $nf_1 < \cdots < nf_\ell$ and build a quality profile for each value in the sequence $\{nf_i\}$. Thus, as an example, the corresponding first plot in the sequence will report a quality profile allowing for each solver a maximum number of function evaluations equal to nf_1.

We strongly remark that, by the respective definitions of data profiles and quality profiles, even if they are both built focusing on the number of function evaluations, on the overall they definitely provide different outcomes for the solvers.

References

[1] Avelino, C.P., Moguerza, J.M., Olivares, A., Prieto, F.J.: Combining and scaling descent and negative curvature directions. Mathematical Programming **128**, 285–319 (2011)

[2] Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. Neural networks **2**(1), 53–58 (1989)

[3] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large–scale machine learning. SIAM Review **60**, 223–311 (2018)

[4] Bray, A.J., Dean, D.S.: Statistics of critical points of Gaussian fields on large-dimensional spaces. Physical review letters **98**(15), 150,201 (2007)

[5] Bunch, J., Kaufman, L.: Some stable methods for calculating inertia and solving symmetric linear equations. Mathematics of Computations **31**, 163–179 (1977)

[6] Caliciotti, A., Fasano, G., Potra, F., Roma, M.: Issues on the use of a modified Bunch and Kaufman decomposition for large scale Newton's equation. Computational Optimization and Applications **77**, 627—651 (2020)

[7] Caliciotti, A., Fasano, G., Roma, M.: Novel preconditioners based on quasi–Newton updates for nonlinear conjugate gradient methods. Optimization Letters **11**, 835–853 (2017)

[8] Caliciotti, A., Fasano, G., Roma, M.: Preconditioned nonlinear conjugate gradient methods based on a modified secant equation. Applied Mathematics and Computation **318**, 196–214 (2018)

[9] Chandra, R.: Conjugate gradient methods for partial differential equations. Ph.D. thesis, Yale University, New Haven (1978). Research Report 129

[10] Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y.: The loss surfaces of multilayer networks. In: Artificial intelligence and statistics, pp. 192–204. PMLR (2015)

[11] Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust–region methods. MPS–SIAM Series on Optimization, Philadelphia, PA (2000)

[12] Cullum, J., Willoughby, R.: Lanczos algorithms for large symmetric eigenvalue computations. Birkhauser, Boston (1985)

[13] Curtis, F.E., Robinson, D.P.: Exploiting negative curvature in deterministic and stochastic optimization. Mathematical Programming **176**, 69–94 (2019)

[14] Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. Advances in neural information processing systems **27** (2014)

[15] De Leone, R., Fasano, G., Roma, M., Sergeyev, Y.D.: Iterative grossone-based computation of negative curvature directions in large-scale optimization. Journal of Optimization Theory and Applications **186**(2), 554–589 (2020)

[16] Dembo, R., Eisenstat, S., Steihaug, T.: Inexact Newton methods. SIAM Journal on Numerical Analysis **19**, 400–408 (1982)

[17] Dembo, R., Steihaug, T.: Truncated-Newton algorithms for large-scale unconstrained optimization. Mathematical Programming **26**, 190–212 (1983)

[18] Dolan, E.D., Moré, J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002)

[19] Fasano, G.: Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 1: Theory. Journal of Optimization Theory and Applications **125**, 523–541 (2005)

[20] Fasano, G.: Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 2: Application. Journal of Optimization Theory and Applications **125**, 543–558 (2005)

[21] Fasano, G., Lucidi, S.: A nonmonotone truncated Newton-Krylov method exploiting negative curvature directions, for large scale unconstrained optimization. Optimization Letters **3**, 521—535 (2009)

[22] Fasano, G., Piermarini, C., Roma, M.: Bridging the gap between trust–region methods (TRMs) and linesearch based methods (LBMs) for nonlinear programming: Quadratic sub–problems. Department of Management, Università Ca' Foscari Venezia, Working Paper (8) (2022)

[23] Fasano, G., Roma, M.: Iterative computation of negative curvature directions in large scale optimization. Computational Optimization and Applications **38**, 81–104 (2007)

[24] Ferris, M., Lucidi, S., Roma, M.: Nonmonotone curvilinear linesearch methods for unconstrained optimization. Computational Optimization and Applications **6**, 117–136 (1996)

[25] Gould, N., Scott, J.: A note on performance profiles for benchmarking software. ACM Transactions on Mathematical Software (TOMS) **43**(2), 1–5 (2016)

[26] Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Exploiting negative curvature directions in linesearch methods for unconstrained optimization. Optimization methods and software **14**, 75–98 (2000)

[27] Gould, N.I.M., Orban, D., Toint, P.L.: CUTEst: a constrained and unconstrained testing environment with safe threads. Computational Optimization and Applications **60**, 545–557 (2015)

[28] HSL 2013: A collection of Fortran codes for large scale scientific computation. URL http://www.hsl.rl.ac.uk/

[29] Jiang, H., Robinson, D.P., Vidal, R., You, C.: A nonconvex formulation for low rank subspace clustering: algorithms and convergence analysis. Computational Optimization and Applications **70**, 395–418 (2018)

[30] Lucidi, S., Rochetich, F., Roma, M.: Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization. SIAM Journal on Optimization **8**, 916–939 (1998)

[31] McCormick, G.: A modification of Armijo's step-size rule for negative curvature. Mathematical Programming **13**, 111–115 (1977)

[32] Moré, J., Sorensen, D.: On the use of directions of negative curvature in a modified Newton method. Mathematical Programming **16**, 1–20 (1979)

[33] Nash, S.: Newton-type minimization via the Lanczos method. SIAM Journal on Numerical Analysis **21**, 770–788 (1984)

[34] Nash, S.: A survey of truncated-Newton methods. Journal of Computational and Applied Mathematics **124**, 45–59 (2000)

[35] Olivares, A., Moguerza, J.M., Prieto, F.J.: Nonconvex optimization using negative curvature within a modified linesearch. European Journal of Operational Research **189**, 706–722 (2008)

[36] Wild, S., Moré, J.: Benchmarking derivative–free optimization algorithms. SIAM J. Optimization **20**, 172–191 (2009)