On the use of the SYMMBK algorithm for computing negative curvature directions within Newton–Krylov methods

Giovanni Fasano[1], Christian Piermarini[2], and Massimo Roma[2]

[1] Dipartimento di Management, Università Ca' Foscari, Venezia, Italy
fasano@unive.it,
[2] 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 nonconvex functions, within Newton–Krylov methods for large scale unconstrained optimization. This issue has been widely investigated in the literature, and different approaches have been proposed. We focus on the well known SYMMBK method proposed for solving large scale symmetric possibly indefinite linear systems [3, 5, 7, 20], and show how to exploit it to yield an effective negative curvature direction. The distinguishing feature of our proposal is that the computation of such negative curvature direction is iteratively carried out, without storing no more than a couple of additional vectors. The results of a preliminary numerical experience are reported showing the reliability of the novel approach we propose.

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

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 a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. 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

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

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 a linesearch–based Newton–Krylov method encompasses two nested loops: the *outer iterations*, where α_k is computed by a linesearch procedure and starting from the current iterate x_k, a new iterate is generated according to the scheme

(1); the *inner iterations*, namely the iterations of the iterative Krylov–subspace method used for approximately solving the Newton equation

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

In Newton–Krylov methods (also called Truncated Newton methods) inner iterations are terminated according to a suited stopping criterion, still ensuring superlinear converge rate [10, 11]. As concerns global convergence properties, convergence to first order critical points (stationary points) is ensured, i.e. points that satisfy first order necessary optimality conditions. For a complete overview of these methods, we refer the reader to the survey paper [24].

Many machine learning problems can be tackled through the minimization of non-convex functions on a given feasible set. Such problems can arise in training neural networks, in low rank subspace clustering problems and in many other contexts. For instance, Curtis and Robinson [8] present new frameworks for combining descent and negative curvature directions in order to train deep neural networks. 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 [2, 6, 9]).

The use of negative curvature directions within (modified) Newton methods dates back to the seminal papers [22] and [23]. These led to the development of several methods based on a combination of a Newton–type direction d_k and a negative curvature direction, i.e. a direction s_k such that $s_k^\top \nabla^2 f(x_k) s_k < 0$, where the iterative scheme (1) is replaced by

$$x_{k+1} = x_k + \alpha_k^2 d_k + \alpha_k s_k, \tag{3}$$

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 a theoretical point of view, the use of suited negative curvature directions enables defining methods converging to second order critical points, i.e., points that satisfy second order necessary optimality conditions, namely stationary points where the Hessian matrix is positive semidefinite. Note that Newton–type methods based on trust region approach, naturally possess such convergence property (see [7]). Linesearch–based methods converging to second order critical points have been also proposed in the framework of nonmonotone methods [17] and extended to large scale setting in [21]. Broadly speaking, to ensure this stronger convergence property, the negative curvature direction must be a good 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:

$$s_k^\top \nabla^2 f(x_k) s_k \longrightarrow 0 \quad \text{implies} \quad \min\left[0, \lambda_{\min}\left(\nabla^2 f(x_k)\right)\right] \longrightarrow 0, \tag{4}$$

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 (4) is a computationally expensive task. Moreover, most of the strategies proposed in literature for computing negative curvature directions satisfying (4) usually rely on matrix factorizations, that are impracticable in the large scale setting. 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 [21], where the Lanczos process is used and the storage of a matrix of the Lanczos vectors generated at each outer iteration is theoretically needed to compute adequate negative curvature directions. The strategy adopted in [21] 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 [18]. 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 an additional computational effort is required.

To our knowledge, the first attempt to iteratively compute negative curvature directions satisfying (4) without requiring the storage of any large matrix or rerunning the iterative process, is represented by the use of a planar CG algorithm, as proposed in [16] (we refer the reader to the papers [12, 13] for a complete description of the planar CG schemes). Besides providing a general theoretical framework which guarantees convergence to second order critical points, in [16] 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 how to determine effective negative curvature directions, to be used within truncated Newton methods. In particular, apart from 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.

Another issue worth investigating concerns how to combine a Newton–type direction and a negative curvature direction taking into account their possible different scaling. In [18], at each outer iteration, given a descent pair of directions (d_k, s_k), only one of the two directions (the most promising) is selected and a suited linesearch is performed along the chosen direction. Following this approach, in [14] a new truncated Newton method is proposed. On the same guideline, in [25] 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 them. 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 the recent paper [8] where a novel framework has been proposed for combining the two directions, alternating two–step and dynamic step approaches. Following the guidelines of the cited literature, we are aware that suitably selecting/combining the Newton–type direction with a negative curvature represents a key point for

improving both efficiency and effectiveness of an iterative method for large scale non-convex optimization. In this regard, we have already considered the idea of a preliminary general framework which possibly encompasses both selection and combination of two search directions [15].

Nevertheless, in this short paper we prefer to uniquely focus on the iterative computation of a novel theoretically sound negative curvature direction, to be used within a curvilinear linesearch–based Newton–Krylov method. In particular, we refer to SYMMBK method for solving large scale symmetric possibly indefinite linear systems [3, 5, 7, 20]. Such method has been recently successfully adopted within truncated Newton methods for computing a gradient related Newton–type direction [4]. More precisely, in this last paper a modified Bunch–Kaufman factorization was adopted within SYMMBK algorithm for solving the Newton equation, at each outer iteration. Indeed, the Bunch–Kaufman factorization is an effective and stable matrix decomposition, but when used for solving the Newton equation might provide a direction not gradient–related, if the objective function is nonconvex. The modification proposed in [4] enables obtaining a direction that is also effective in practice. Hence the idea is to possibly use the same procedure for obtaining also a negative curvature direction satisfying (4), with minimal additional storage.

2 Computing negative curvature directions via SYMMBK procedure

We briefly recall that the SYMMBK procedure basically relies on a couple of relevant tools: the Lanczos iterative process and the Bunch–Kaufman decomposition [3]. The first tool allows to transform the symmetric linear system (2), where the matrix $\nabla^2 f(x_k)$ is possibly indefinite, into the system of equalities

$$\begin{cases} T_k y_k = \|\nabla f(x_k)\| e_1 \\ d_k = Q_k y_k, \end{cases} \tag{5}$$

being $T_k \in \mathbb{R}^{m \times m}$ *tridiagonal* and $Q_k \in \mathbb{R}^{n \times m}$. In (5) the columns of the matrix Q_k are given by the m Lanczos vectors $q_1, \ldots, q_m$ (with $q_\ell^T q_i = 0$ and $\|q_\ell\|_2 = 1$, being $1 \leq \ell \neq i \leq m$), after m inner iterations. Hence, the m Lanczos vectors computed by the SYMMBK method yield

$$Q_k = \begin{pmatrix} q_1 & \vdots & \vdots & q_m \end{pmatrix}. \tag{6}$$

We remark that at a given iterate x_k, a relevant property of matrix Q_k is evidenced by the next relation

$$T_k = Q_k^T \nabla^2 f(x_k) Q_k. \tag{7}$$

Conversely, the Bunch–Kaufman decomposition allows for an easy factorization of the tridiagonal matrix T_k as in

$$T_k = S_k B_k S_k^T, \tag{8}$$

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 (5), after m iterations of the Lanczos process, the vector d_k is available and can be used as a search direction within an optimization framework. Furthermore, in [4] the authors slightly modified the pivot test within the Bunch–Kaufman decomposition, so that the resulting vector d_k was provably gradient–related.

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, such that no more than a couple of n–dimensional vectors need to be stored for its computation. This will provide a general matrix–free technique to construct negative curvature directions in large scale settings. The vector s_k will be used to steer the convergence to a stationary point where the Hessian matrix is positive semidefinite. As a more specific task, we technically address a negative curvature direction s_k such that the next properties are fulfilled (see also (4)):

$$s_k^T \nabla f(x_k) < 0 \tag{9}$$

$$s_k^T \nabla^2 f(x_k) s_k < 0 \tag{10}$$

$$s_k^T \nabla^2 f(x_k) s_k \to 0 \quad \Longrightarrow \quad \min\left[0, \lambda_{\min}(\,\nabla^2 f(x_k)\,)\right] \to 0. \tag{11}$$

Given (2) and (8), 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 $y \in \mathbb{R}^m$ represents a relatively simple task, such that $S_k^T y = w$. Then, by (7) and (8) we obtain

$$\begin{aligned}(Q_k y)^T \nabla^2 f(x_k)(Q_k y) = y^T \left[Q_k^T \nabla^2 f(x_k) Q_k\right] y \;&=\; y^T T_k y \;=\; y^T S_k B_k S_k^T y \\ = (S_k^T y)^T B_k (S_k^T y) \;&=\; w^T B_k w \;=\; \lambda \|w\|_2^2 < 0. \end{aligned} \tag{12}$$

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 show the subsequent results:

- the vector $s_k = Q_k y$ is a direction satisfying (9)–(11);
- the vector $s_k = Q_k y$ can be efficiently (say iteratively) computed by exploiting the (modified) SYMMBK procedure, without storing any matrix.

On this purpose we preliminary consider the next result, whose proof can be easily obtained from Lemma 4.3 in [23] and Theorem 3.2 in [16].

Lemma 1. *Suppose $m = n$ iterations of the Lanczos process are performed by SYMMBK when solving Newton's equation (2) at iterate x_k, for a given $k \geq 0$, so that the decompositions*

$$\begin{cases} T_k = Q_k^T \nabla^2 f(x_k) Q_k \\ T_k = S_k B_k S_k^T \end{cases} \tag{13}$$

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 (9)–(11).

Now, observe that the results in Lemma 1 assume that the Lanczos process performs exactly n iterations to solve (2): this is definitely unaffordable for large n. Hence, we need to generalize the contents in Lemma 1 to the case $m < n$. Moreover, we highlight that to compute the vector $\bar{y}$ in Lemma 1 we can recur to Lemma 4.3 in [23]. In this regard, with the next lemma we intend to rephrase Lemma 1 in order to weaken its outcomes.

Lemma 2. *Suppose $m < n$ iterations of the Lanczos process are performed by SYMMBK when solving Newton's equation (2) at iterate x_k, for a given $k \geq 1$, so that the decompositions (13) 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 a bounded direction that satisfies* (9)–(10).

As a further result, Lemma 4.3 in [23] ensures that the outcomes in Lemma 1 can be easily generalized when the linear system $S_k^T y = w$ is replaced by

$$S_k^T y = \sum_{1 \leq i \leq m \,:\, \lambda_i < 0} w_i, \tag{14}$$

being w_i an eigenvector of B_k corresponding to its negative eigenvalue λ_i. In this regard, a couple of 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 1;
- the computation of $\lambda_{\min}$ can be considerably simplified by exploiting a diagonalization of the matrix B_k.

3 Iterative computation of negative curvature directions

According with Lemma 1 and Lemma 2, the matrix T_k can be decomposed as $T_k = S_k B_k S_k^T$, being B_k a 1×1 or 2×2 block diagonal matrix. After diagonalyzing B_k as in $B_k = X_k D_k X_k^T$, with X_k orthogonal, we obtain

$$T_k = S_k X_k D_k X_k^T S_k^T = W_k D_k W_k^T, \tag{15}$$

where

$$W_k \stackrel{\text{def}}{=} S_k X_k = \begin{pmatrix} W^{(1\ 1)} & & & & \\ W^{(2\ 1)} & W^{(2\ 2)} & & & \\ & \cdot & \cdot & & \\ & & \cdot & W^{(j-1\ j-1)} & \\ & & & W^{(j\ j-1)} & W^{(j\ j)} \end{pmatrix}, \qquad j \geq 1. \tag{16}$$

Here the sizes (both rows and columns) of the sub–diagonal blocks $W^{(i+1\ i)}$, $i \in \{1, \ldots, j-1\}$, depend on the sizes of the diagonal blocks $W^{(1\ 1)}, \ldots, W^{(j\ j)}$. Moreover, the diagonal blocks $W^{(i+1\ i+1)}$ are orthogonal. Now, by combining (13) and (15) we can compute a set of conjugate directions for the Hessian matrix $\nabla^2 f(x_k)$, being indeed

$$T_k = W_k D_k W_k^T = Q_k^T \nabla^2 f(x_k) Q_k, \tag{17}$$

so that

$$D_k = W_k^{-1} Q_k^T \nabla^2 f(x_k) Q_k W_k^{-T} = G_k^T \nabla^2 f(x_k) G_k, \tag{18}$$

where

$$G_k \stackrel{\text{def}}{=} Q_k W_k^{-T} \in \mathbb{R}^{n \times m}. \tag{19}$$

Since D_k is a diagonal matrix, by (19) the columns of G_k yield a set of m linearly independent (see also Proposition 2.1 of [12], for completeness) $\nabla^2 f(x_k)$-conjugate directions which span the Krylov subspace $\mathbb{K}(\nabla^2 f(x_k), q_1, m)$. To efficiently compute the matrix G_k, let us define

$$G_k = (G^1\ G^2\ \ldots\ G^{j-1}\ G^j), \tag{20}$$

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 (19) as

$$G_k W_k^T = Q_k \stackrel{\text{def}}{=} (Q^1\ Q^2\ \ldots\ Q^{j-1}\ Q^j), \tag{21}$$

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 the Lanczos vectors. Hence, using the expression (16) for matrix W_k, as well as the orthogonality of the blocks $W^{(i\ i)}$, $1 \leq i \leq j$, we obtain from (20) and (21)

$$G^i = \left[Q^i - G^{i-1} \left(W^{(i\ i-1)} \right)^T \right] W^{(i\ i)}, \qquad 2 \leq i \leq j, \tag{22}$$

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 (see also Section 5.2.3 of [7]).

4 Theoretical property of negative curvature directions

Relations (22) indicate how to fully iteratively compute the matrix G_k in (19); moreover, (18) 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

$$z = \sum_{1 \leq j \leq m\ :\ \mu_j < 0} a_j G_{(j)}, \tag{23}$$

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

$$\sum_{1\le j\le n \,:\, \mu_j<0} a_j^2\mu_j \;\le\; \lambda_{\min}\left[D_k\right]\left[\min_{1\le j\le n \,:\, \mu_j<0} a_j^2\right]. \tag{24}$$

Then, we set $s_k = z/\|z\|$ and we can prove the next result (which can be suitably generalized, through some algebra, to the case where $m < n$ inner iterations are performed).

Proposition 1. *Assume that at 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 (19) after n inner iterations, and let $\kappa(G_k)$ denote the condition number of G_k. Assume a_j, with $1 \le j \le n$, is a set of real values satisfying (24). Then*

$$s_k^T\nabla^2 f(x_k)s_k \le \frac{1}{N\cdot[\kappa(G_k)]^2}\left[\frac{\min\limits_{1\le j\le n \,:\, \mu_j<0} a_j^2}{\max\limits_{1\le j\le n \,:\, \mu_j<0} a_j^2}\right]\lambda_{\min}\left[\nabla^2 f(x_k)\right], \tag{25}$$

where $N \ge 1$ represents 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.

The last proposition evidences that in the case the Lanczos process performs exactly n inner iterations within SYMMBK procedure, then a negative curvature direction s_k satisfying (9)–(11) can be easily available.

5 Numerical experiments

We performed a preliminary numerical testing in order to assess the reliability of the approach detailed in this paper. We relied on the same optimization procedure proposed in [4]; in particular, to solve Newton's equation we adopted the HSL_MI02 routine available from the HSL Mathematical Software Library [20], in which the pivoting rule is modified as described in Section 3 of the aforementioned paper [4] (we refer to this paper for any implementation details). We embedded in this implementation the iterative computation of the negative curvature direction, as described in the previous sections, requiring a small amount of additional storage. We adopted the iterative scheme (3), where the steplength α_k is computed by the standard curvilinear linesearch procedure in [22], without considering any sophisticated strategy for possibly choosing the most promising between the two directions. This is motivated by the fact that we are mainly interested here in assessing the reliability of the approach we proposed. In order to address the well–known scaling problem, possibly occurring when combining a Newton–type (d_k) and a negative curvature (s_k) direction, we ignored s_k whenever its norm consistently differed from the norm of d_k. More precisely, we did not use s_k when

$$\|s_k\| > \eta_1\|d_k\| \quad \text{or} \quad \|s_k\| < \eta_2\|d_k\|,$$

where $\eta_1 > 0$ and $\eta_2 > 0$ are suited scalars. The rationale behind this choice is suggested by recalling that both d_k and s_k are built using the same conjugate directions, i.e. the columns of the matrix G_k in (19). It is a "brute force" rule that we adopted in the spirit of this preliminary testing. More sophisticated strategies could be certainly adopted and will be the subject of future work (see also [15]). In our tests, we set $\eta_1 = 4 \cdot 10$ and $\eta_2 = 1/4 \cdot 10^{-1}$.

The algorithm was tested on all the large scale unconstrained test problems in the CUTEst collection [19], including both convex and nonconvex functions. The resulting test bed consists of 111 test problems, whose dimension ranges between 1,000 and 10,000. We report the results obtained in terms of the number of outer iterations, number of function evaluations, number of inner iterations and CPU time. Moreover, we monitored the optimal objective function values obtained. A comparison is performed with an implementation ignoring at all negative curvature directions. In particular, in Table 1 we report the number of test problems in which the algorithm which uses negative curvatures performed better, worse or equal to the benchmark algorithm. We defined specific tolerances

Table 1. Performance comparison ($\eta_1 = 4 \cdot 10$ and $\eta_2 = 1/4 \cdot 10^{-1}$). E.g., the value 18 in the first row and fifth column implies that the algorithm which uses our negative curvature direction achieves better (lower) local minima on 18 out of 111 test problems, while reaching worse ones on 8 test problems.

	Number of outer iter.	*Number of funct. eval.*	*Number of inner iter.*	*CPU Time (seconds)*	*Objective funct. value*
Number of wins	23	14	26	10	18
Number of losses	21	30	20	14	8
Number of draws	67	67	65	87	85

when comparing the objective function value and CPU time. If the difference between the performance of the two algorithms is within these tolerances, then the comparison resulted in a draw. This aims at ensuring the significance of the achieved results in terms of quality of the solutions and required time. We use a tolerance of 0.01 between the objective function values and 1 second between the CPU times.

By observing the last column of Table 1 we can appreciate the enhanced capability of the algorithm which uses our negative curvature direction to achieve better minima. This is a result of the fact that negative curvature directions allow the algorithm to escape from flat zones or valleys where any standard truncated–Newton method can get stuck into. As expected, the main downside of our approach is the necessity to perform more function evaluations with respect to the benchmark algorithm that ignores negative curvature directions (see the second column of Table 1). This is mainly due to the use of the curvilinear linesearch. This behaviour may result in a larger CPU time. Of course, the test bed includes several problems where negative curvature directions are not encountered, hence a large number of draws.

Many further comparison runs were carried out, using different reasonable threshold values, obtaining similar conclusions. As an example, we report in Table 2 the results obtained by setting in our numerical experience $\eta_1 = 2 \cdot 10$ and $\eta_2 = 1/2 \cdot 10^{-1}$. As it can be easily observed, these results show conclusions

Table 2. Performance comparison ($\eta_1 = 2 \cdot 10$ and $\eta_2 = 1/2 \cdot 10^{-1}$).

	Number of outer iter.	*Number of funct. eval.*	*Number of inner iter.*	*CPU Time (seconds)*	*Objective funct. value*
Number of wins	21	13	23	10	15
Number of losses	22	31	23	16	5
Number of draws	68	67	65	85	91

similar to those for Table 1.

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. 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)
3. Bunch, J., Kaufman, L.: Some stable methods for calculating inertia and solving symmetric linear equations. Mathematics of Computations **31**, 163–179 (1977)
4. 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)
5. Chandra, R.: Conjugate gradient methods for partial differential equations. Ph.D. thesis, Yale University, New Haven (1978). Research Report 129
6. 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)
7. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust–region methods. MPS–SIAM Series on Optimization, Philadelphia, PA (2000)
8. Curtis, F.E., Robinson, D.P.: Exploiting negative curvature in deterministic and stochastic optimization. Mathematical Programming **176**, 69–94 (2019)
9. 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)
10. Dembo, R., Eisenstat, S., Steihaug, T.: Inexact Newton methods. SIAM Journal on Numerical Analysis **19**, 400–408 (1982)
11. Dembo, R., Steihaug, T.: Truncated-Newton algorithms for large-scale unconstrained optimization. Mathematical Programming **26**, 190–212 (1983)
12. Fasano, G.: Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 1: Theory. Journal of Optimization Theory and Applications **125**, 523–541 (2005)

13. Fasano, G.: Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 2: Application. Journal of Optimization Theory and Applications **125**, 543–558 (2005)
14. 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)
15. 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)
16. Fasano, G., Roma, M.: Iterative computation of negative curvature directions in large scale optimization. Computational Optimization and Applications **38**, 81–104 (2007)
17. Ferris, M., Lucidi, S., Roma, M.: Nonmonotone curvilinear linesearch methods for unconstrained optimization. Computational Optimization and Applications **6**, 117–136 (1996)
18. 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)
19. 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)
20. HSL 2013: A collection of Fortran codes for large scale scientific computation. URL http://www.hsl.rl.ac.uk/
21. 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)
22. McCormick, G.: A modification of Armijo's step-size rule for negative curvature. Mathematical Programming **13**, 111–115 (1977)
23. Moré, J., Sorensen, D.: On the use of directions of negative curvature in a modified Newton method. Mathematical Programming **16**, 1–20 (1979)
24. Nash, S.: A survey of truncated-Newton methods. Journal of Computational and Applied Mathematics **124**, 45–59 (2000)
25. 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)