# On Efficiently Combining Limited Memory and Trust-Region Techniques

Oleg Burdakov[a,1]
Lujin Gong[b]
Ya-xiang Yuan[c]
Spartak Zikrin[a]

[a]Department of Mathematics, Linköping University, SE-581 83 Linköping, Sweden

[b]Tencent, Beijing, China

[c]State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, CAS, Beijing, China

[1]Corresponding author. Tel.: +46 13 281473. E-mail address: oleg.burdakov@liu.se

1

# On Efficiently Combining Limited Memory and Trust-Region Techniques

Oleg Burdakov[*]    Lujin Gong[†]

Ya-xiang Yuan[‡]    Spartak Zikrin[*]

**Abstract**  Limited memory quasi-Newton methods and trust-region methods represent two efficient approaches used for solving unconstrained optimization problems. A straightforward combination of them deteriorates the efficiency of the former approach, especially in the case of large-scale problems. For this reason, the limited memory methods are usually combined with a line search. We show how to efficiently combine limited memory and trust-region techniques. One of our approaches is based on the eigenvalue decomposition of the limited memory quasi-Newton approximation of the Hessian matrix. The decomposition allows for finding a nearly-exact solution to the trust-region subproblem defined by the Euclidean norm with an insignificant computational overhead as compared with the cost of computing the quasi-Newton direction in line-search limited memory methods. The other approach is based on two new eigenvalue-based norms. The advantage of the new norms is that the trust-region subproblem is separable and each of the smaller subproblems is easy to solve. We show that our eigenvalue-based limited-memory trust-region methods are globally convergent. Moreover, we propose improved versions of the existing limited-memory trust-region algorithms. The presented results of numerical experiments demonstrate the efficiency of our approach which is competitive with line-search versions of the L-BFGS method.

**Keywords:**  Unconstrained Optimization; Large-scale Problems; Limited Memory Methods; Trust Region Methods; Shape-Changing Norm; Eigenvalue Decomposition.

**Mathematics Subject Classification (2000):** 90C06, 90C30, 90C53.

## 1   Introduction

We consider the following general unconstrained optimization problem

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

---

[*]Department of Mathematics, Linköping University, Linköping 58183, Sweden. (oleg.burdakov, spartak.zikrin@liu.se)

[†]Tencent, Beijing, China. (evansgong@tencent.com)

[‡]State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, CAS, Beijing 100190, China. (yyx@lsec.cc.ac.cn)

where $f$ is assumed to be at least continuously differentiable. Line-search and trust-region methods [14, 34] represent two competing approaches to solving (1). The cases when one of them is more successful than the other are problem dependent.

At each iteration of the trust-region methods, a trial step is generated by minimizing a quadratic model of $f(x)$ within a trust region. The trust-region subproblem is formulated, for the $k$-th iteration, as follows:

$$\min_{s \in \Omega_k} g_k^T s + \frac{1}{2} s^T B_k s \equiv q_k(s), \tag{2}$$

where $g_k = \nabla f(x_k)$, and $B_k$ is either the true Hessian in $x_k$ or its approximation. The trust region is a ball of radius $\Delta_k$

$$\Omega_k = \{s \in R^n : \|s\| \le \Delta_k\}.$$

It is usually defined by a fixed vector norm, typically, scaled $l_2$ or $l_\infty$ norm. If the trial step provides a sufficient decrease of $f(x)$, it is accepted, otherwise the trust-region radius is decreased while keeping the same model function.

Worthy of mention are the following attractive features of the trust-region methods. Firstly, they are able to take advantage of exploiting a negative curvature information contained in $B_k$. They exhibit another important feature when the full quasi-Newton step $-B_k^{-1} g_k$ does not produce a sufficient decrease in $f$. This happens when the quadratic model provides a relatively good prediction of $f$ within a trust region of a radius $\Delta_k < \| - B_k^{-1} g_k\|$. Clearly, the accepted trial step provides a better predicted decrease of $f$ than the one provided within $\Omega_k$ by the quasi-Newton direction. This gives the trust-region methods a higher chance to produce a better actual reduction in $f$. Furthermore, if $B_k \succ 0$ and it is ill-conditioned, the quasi-Newton search direction may be almost orthogonal to $g_k$. This is an unfavorable case for the line-search methods. In contrast, the direction of the vector $s(\Delta_k)$ that solves (2) approaches the direction of $-g_k$ when $\Delta_k$ decreases.

There exists a variety of approaches [14, 34, 39, 43] to approximately solving the trust-region subproblem defined by the Euclidean norm. Depending on how accurately the trust-region subproblem is solved, the methods are categorized as nearly-exact or inexact.

The class of inexact trust-region methods includes, e.g., the dogleg method [36, 37], double-dogleg method [15], truncated conjugate gradient (CG) method [38, 40], Newton-Lanczos method [26], subspace CG method [42] and two-dimensional subspace minimization method [12].

Faster convergence, in terms of the number of iterations, is generally expected when the trust-region subproblem is solved more accurately. Nearly-exact methods are usually based on the optimality conditions, presented by Moré and Sorensen [31] for the Euclidean norm used in (2). These conditions state that there exists a pair $(s^*, \sigma^*)$ such that $\sigma^* \ge 0$ and

$$\begin{aligned} (B_k + \sigma^* I)s^* &= -g_k, \\ \sigma^*(\|s^*\|_2 - \Delta) &= 0, \\ B_k + \sigma^* I &\succeq 0. \end{aligned} \tag{3}$$

In these methods, a nearly-exact solution is obtained by iteratively improving $\sigma$ and solving in $s$ the linear system

$$(B_k + \sigma I)s = -g_k. \tag{4}$$

The class of limited-memory quasi-Newton methods [3, 23, 24, 30, 33] is one of the most effective tools used for solving large-scale problems, especially when the maintaining and operating with dense Hessian approximation is costly. In these methods, a few pairs of vectors

$$s_i = x_{i+1} - x_i \quad \text{and} \quad y_i = \nabla f(x_{i+1}) - \nabla f(x_i) \tag{5}$$

are stored for implicitly building an approximation of the Hessian, or its inverse, by using a low rank update of a diagonal matrix. The number of such pairs is limited by $m \ll n$. This allows for arranging efficient matrix-vector multiplications involving $B_k$ and $B_k^{-1}$.

For most of the quasi-Newton updates, the Hessian approximation admits a compact representation

$$B_k = \delta_k I + V_k^T W_k V_k, \tag{6}$$

where $\delta_k$ is a scalar, $W_k \in R^{\bar{m} \times \bar{m}}$ is a symmetric matrix and $V_k \in R^{n \times \bar{m}}$. This is the main property that will be exploited in this paper. The value of $\bar{m}$ depends on the number of stored pairs (5) and it may vary from iteration to iteration. Its maximal value depends on the updating formula and equals, typically, $m$ or $2m$. To simplify the presentation and our analysis, especially when specific updating formulas are discussed, we shall assume that the number of stored pairs and $\bar{m}$ equal to their maximal values.

So far, the most successful implementations of limited memory methods were associated with line search. Nowadays, the most popular limited memory line-search methods are based on the BFGS-update [34], named after Broyden, Fletcher, Goldfarb and Shanno. The complexity of computing a search direction in the best implementations of these methods is $4mn$.

Line-search methods often employ the strong Wolfe conditions [41] that require additional function and gradient evaluations. These methods have a strong requirement of positive definiteness of the Hessian matrix approximation, while trust-region methods, as mentioned above, can even gain from exploiting information about possible negative curvature. Moreover, the latter methods do not require gradient computation in unacceptable trial points. Unfortunately, any straightforward embedding of limited memory quasi-Newton techniques in the trust-region framework deteriorates the efficiency of the former approach.

The existing refined limited memory trust-region methods [10, 19, 20, 29] typically use the limited-memory BFGS updates (L-BFGS) for approximating the Hessian and the Euclidean norm for defining the trust region. In the double-dogleg approach by Kaufman [29], the Hessian and its inverse are simultaneously approximated using the L-BFGS in a compact representation [11]. The cost of one iteration for this inexact approach varies from $4mn + O(m^2)$ to $O(n)$ operations depending on whether the trial step was accepted at the previous iteration or not. Using the same compact representation, Burke et al. [10] proposed two versions of implementing the Moré-Sorensen approach [31] for finding a nearly-exact solution to the trust-region subproblem. The cost of one iteration varies from $2mn + O(m^3)$ to either $2m^2n + 2mn + O(m^3)$ or $6mn + O(m^2)$ operations, depending on

how the updating of $B_k$ is implemented. Recently Erway and Marcia [18, 19, 20] proposed a new technique for solving (4), based on the unrolling formula of L-BFGS [11]. In this case, the cost of one iteration of their implementation [20] is $O(m^2 n)$ operations. In the next sections, we describe the aforementioned limited memory trust-region methods in more detail and compare them with those we propose here.

The aim of this paper is to develop new approaches that would allow for effectively combining the limited memory and trust region techniques. They should break a wide-spread belief that such combinations are less efficient than the line-search-based methods.

We focus here on the quasi-Newton updates that admit a compact representation (6). It should be underlined that a compact representation is available for the most of quasi-Newton updates, such as BFGS [11], symmetric rank-one (SR1) [11] and multipoint symmetric secant approximations [8], which contain the Powell-symmetric-Broyden (PSB) update [37] as a special case. Most recently, Erway and Marcia [21] provided a compact representation for the entire Broyden convex class of updates.

We begin in Section 2 with showing how to efficiently compute, at a cost of $O(\bar{m}^3)$ operations, the eigenvalues of $B_k$ with implicitly defined eigenvectors. For the case when the trust region is defined by the Euclidean norm, and the implicit eigenvalue decomposition is available, we show in Section 3 how to find a nearly-exact solution to the trust-region subproblem at a cost of $2\bar{m}n + O(\bar{m})$ operations. The idea of using the eigenvalue decomposition for limited memory methods was originally introduced in [7, 9], and then successfully exploited in [2, 21, 22]. In Section 4, we introduce two new norms which leans upon the eigenvalue decomposition of $B_k$. The shape of the trust region defined by these norms changes from iteration to iteration. The new norms allow for decomposing the corresponding trust-region subproblem into a set of easy-to-solve quadratic programming problems. For one of the new norms, the exact solution to the trust-region subproblem is obtained in closed form. For the other one, the solution is reduced to a small $\bar{m}$-dimensional trust-region subproblem in the Euclidean norm. In Section 5, a generic trust-region algorithm is presented, which is used in the implementation of our algorithms. In Section 6, global convergence is proved for eigenvalue-based limited memory methods. In Sections 2-6, $B_k$ is not required to be positive definite. The rest of the paper is focused on specific positive definite quasi-Newton updates, namely, L-BFGS. For this case, we develop in Section 7 an algorithm, in which the computational cost of one iteration varies from $4mn + O(m^3)$ to $2mn + O(m^2)$ operations, depending on whether the trial step was accepted at the previous iteration or not. This means that the highest order term in the computational cost is the same as for computing the search direction in the line-search L-BFGS algorithms. In Section 8, we propose improved versions of the limited memory trust-region algorithms [10, 29]. The results of numerical experiments are presented in Section 9. They demonstrate the efficiency of our limited memory trust-region algorithms. We conclude our work and discuss future direction in Section 10.

# 2 Spectrum of limited-memory Hessian approximation

Consider the trust-region subproblem (2), in which we simplify notations by dropping the subscripts, i.e., we consider

$$\min_{s \in \Omega} g^T s + \frac{1}{2} s^T B s \equiv q(s). \tag{7}$$

It is assumed in this paper that the Hessian approximation admits the compact representation (6), that is,

$$B = \delta I + V W V^T, \tag{8}$$

where $W \in R^{\bar{m} \times \bar{m}}$ is a symmetric matrix and $V \in R^{n \times \bar{m}}$. The main assumption here is that $\bar{m} \ll n$. It is natural to assume that the scalar $\delta$ is positive because there are, in general, no special reasons to base the model $q(s)$ on the hypothesis that the curvature of $f(x)$ is zero or negative along all directions orthogonal to the columns of $V$.

Below, we demonstrate how to exploit compact representation (8) and efficiently compute eigenvalues of $B$. For trust regions of a certain type, this will permit us to easily solve the trust-region subproblem (7).

Suppose that the Cholesky factorization $V^T V = R^T R$ is available, where $R \in R^{\bar{m} \times \bar{m}}$ is upper triangular. The rank of $V$, denoted here by $r$, is equal to the number of nonzero diagonal elements of $R$. Let $R_\dagger \in R^{r \times \bar{m}}$ be obtained from $R$ by deleting the rows that contain zero diagonal elements, and let $R_\ddagger \in R^{r \times r}$ be obtained by additionally deleting the columns of $R$ of the same property. Similarly, we obtain $V_\dagger \in R^{n \times r}$ by deleting the corresponding columns of $V$. Consider the $n \times r$ matrix

$$Q = V_\dagger R_\ddagger^{-1}. \tag{9}$$

Its columns form an orthonormal basis for the column space of both $V_\dagger$ and $V$. The equality

$$V = Q R_\dagger \tag{10}$$

can be viewed as the rank revealing QR (RRQR) decomposition of $V$ [1, Theorem 1.3.4].

By decomposition (10), we have

$$B = \delta I + Q R_\dagger W R_\dagger^T Q^T,$$

where the matrix $R_\dagger W R_\dagger^T \in R^{r \times r}$ is symmetric. Consider its eigenvalue decomposition $R_\dagger W R_\dagger^T = U D U^T$, where $U \in R^{r \times r}$ is orthogonal and $D \in R^{r \times r}$ is a diagonal matrix composed of the eigenvalues $(d_1, d_2, \ldots, d_r)$. Denote $P_\parallel = QU \in R^{n \times r}$. The columns of $P_\parallel$ form an orthonormal basis for the column space of $V$. This yields the following representation of the quasi-Newton matrix:

$$B = \delta I + P_\parallel D P_\parallel^T.$$

Let $P_\perp \in R^{n \times (n-r)}$ define the orthogonal complement to $P_\parallel$. Then $P = [P_\parallel \; P_\perp] \in R^{n \times n}$ is an orthogonal matrix. This leads to the eigenvalue decomposition:

$$B = P \begin{pmatrix} \Lambda & 0 \\ 0 & \delta I_{n-r} \end{pmatrix} P^T, \tag{11}$$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$ and $\lambda_i = \delta + d_i$, $i = 1, \ldots, r$. From (11), we conclude that the spectrum of $B$ consists of:

- $r$ eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$ with the eigenspace defined by $P_\parallel$;

- $(n - r)$ identical eigenvalues $\delta$ with the eigenspace defined by $P_\perp$.

Thus, $B$ has at most $r + 1$ distinct eigenvalues that can be computed at a cost of $O(r^3)$ operations for the available $W$ and RRQR decomposition of $V$.

In our implementation, we do not explicitly construct the matrix $Q$, but only the triangular matrix $R$, which is obtained from the aforementioned Cholesky factorization of the $\bar{m} \times \bar{m}$ Gram matrix $V^T V$ at a cost of $O(\bar{m}^3)$ operations [25]. The complexity can be decreased to $O(\bar{m}^2)$ if the Cholesky factorization is updated after each iteration by taking into account that the current matrix $V$ differs from the previous one by at most two columns. We show in Section 7.3 how to update $V^T V$ at $\bar{m}n + O(\bar{m}^2)$ operations. Although this will be shown for the L-BFGS update, the same technique works for the other limited memory quasi-Newton updates that admit the compact representation (8). Any matrix-vector multiplications involving $Q$ are implemented using the representation (9). For similar purposes, we make use of the representation

$$P_\parallel = V_\dagger R_\ddagger^{-1} U. \tag{12}$$

In contrast to the eigenvalues that are explicitly computed, the eigenvectors are not computed explicitly. Therefore, we can say that the eigenvalue decomposition of $B$ (11) is defined implicitly. The matrices $P$, $P_\parallel$ and $P_\perp$ will be involved in presenting our approach, but they are not used in any of our algorithms.

In the next section, we describe how to solve the trust-region subproblem (7) in the Euclidean norm by exploiting the implicit eigenvalue decomposition of $B$.

# 3    Trust-region subproblem in the Euclidean norm

It is assumed here that $\Omega = \{s \in R^n : \|s\|_2 \leq \Delta\}$. To simplify notation, $\|\cdot\|$ denotes further the Euclidean vector norm and the induced matrix norm.

The Moré-Sorenson approach [31] seeks for an optimal pair $(s^*, \sigma^*)$ that satisfies conditions (3). If $B \succ 0$ and the quasi-Newton step $s_N = -B^{-1}g \in \Omega$, then $s_N$ solves the trust-region subproblem. Otherwise, its solution is related to solving the equation

$$\phi(\sigma) = 0, \tag{13}$$

where $\phi(\sigma) = 1/\Delta - 1/\|s\|$ and $s = s(\sigma)$ is the solution to the linear system

$$(B + \sigma I)s = -g. \tag{14}$$

In the standard Moré-Sorenson approach, the Cholesky factorization of $B + \sigma I$ is typically used for solving (14). To avoid this computationally demanding factorization, we take advantage of the implicitly available eigenvalue decomposition of $B$ (11), which yields:

$$B + \sigma I = P \begin{pmatrix} \Lambda + \sigma I_r & 0 \\ 0 & (\delta + \sigma)I_{n-r} \end{pmatrix} P^T.$$

Consider a new $n$-dimensional variable defined by the orthogonal matrix $P$ as

$$v = P^T s = \begin{pmatrix} v_\| \\ v_\perp \end{pmatrix} \in R^n, \tag{15}$$

where $v_\| = P_\|^T s \in R^r$ and $v_\perp = P_\perp^T s \in R^{n-r}$. Then equation (14) is reduced to

$$\begin{cases} (\Lambda + \sigma I_r)v_\| = -g_\| \\ (\delta + \sigma)v_\perp = -g_\perp \end{cases}, \tag{16}$$

where $g_\| = P_\|^T g \in R^r$ and $g_\perp = P_\perp^T g \in R^{n-r}$. For the values of $\sigma$ that make this system nonsingular, we denote its solution by $v(\sigma)$. Since $\|s\| = \|v\|$, the function $\phi(\sigma)$ in (13) can now be defined as $\phi(\sigma) = 1/\Delta - 1/\|v(\sigma)\|$.

Let $\lambda_{\min}$ stand for the smallest eigenvalue of $B$. Let $P_{\min}$ be the set of columns of $P$ that span the subspace corresponding to $\lambda_{\min}$. We denote $v^* = P^T s^*$ and seek now for a pair $(v^*, \sigma^*)$ that solves (16). Conditions (3) require also that

$$\sigma^*(\|v^*\| - \Delta) = 0 \quad \text{and} \quad \sigma^* \geq \max\left(0, -\lambda_{\min}\right). \tag{17}$$

We shall show how to find a pair with the required properties separately for each of the following two cases.

*Case I:* $\lambda_{\min} > 0$ or $\|P_{\min}^T g\| \neq 0$.
Here if $\lambda_{\min} > 0$ and $\|v(0)\| \leq \Delta$, we have $v^* = v(0)$ and $\sigma^* = 0$. Otherwise,

$$\sigma^* > \max\left(0, -\lambda_{\min}\right). \tag{18}$$

Then equation (13) is solved by Newton's root-finding algorithm [31], where each iteration takes the form

$$\sigma \leftarrow \sigma - \frac{\phi(\sigma)}{\phi'(\sigma)} = \sigma - \frac{(\|v(\sigma)\| - \Delta) \cdot \|v(\sigma)\|^2}{\Delta \cdot v^T(\sigma)v'(\sigma)}. \tag{19}$$

For this formula, equation (16) yields

$$\|v(\sigma)\|^2 = g_\|^T(\Lambda + \sigma I_r)^{-2} g_\| + (\delta + \sigma)^{-2}\|g_\perp\|^2 \tag{20}$$

and

$$\begin{aligned} v^T(\sigma)v'(\sigma) &= -v^T(\sigma) \begin{pmatrix} \Lambda + \sigma I_r & 0 \\ 0 & (\delta + \sigma)I_{n-r} \end{pmatrix}^{-1} v(\sigma) \\ &= -g_\|^T(\Lambda + \sigma I_r)^{-3} g_\| - (\delta + \sigma)^{-3}\|g_\perp\|^2. \end{aligned} \tag{21}$$

It is easy to control the iterates from below by making use of the property (18), which guarantees that the diagonal matrices $\Lambda + \sigma I_r$ and $(\delta + \sigma)I_{n-r}$ are nonsingular. In practice,

just a pair of iterations (19) are often sufficient for solving (13) to an appropriate accuracy [34]. For the obtained approximate value $\sigma^*$, the two blocks that compose $v^*$ are defined by the formulas

$$v_\parallel^* = -(\Lambda + \sigma^* I_r)^{-1} g_\parallel, \tag{22}$$

$$v_\perp^* = -(\delta + \sigma^*)^{-1} g_\perp. \tag{23}$$

*Case II:* $\lambda_{\min} \leq 0$ and $\|P_{\min}^T g\| = 0$.

Here $\lambda_{\min} < 0$ corresponds to the so-called *hard case* [31]. Since it was assumed that $\delta > 0$, we have $\lambda_{\min} \neq \delta$. Let $\bar{r}$ be the algebraic multiplicity of $\lambda_{\min}$. Suppose that the eigenvalues are sorted in the way that $\lambda_{\min} = \lambda_1 = \ldots = \lambda_{\bar{r}} < \lambda_i$, for all $i > \bar{r}$. Denote $\bar{v} = (v_{\bar{r}+1}, v_{\bar{r}+2}, \ldots, v_n)^T$. The process of finding an optimal pair $(\sigma^*, v^*)$ is based on a simple analysis of the alternatives in (16), which require that, for all $1 \leq i \leq \bar{r}$, either $\lambda_i + \sigma = 0$ or $(v_\parallel)_i = 0$. It is associated with finding the unique solution of the following auxiliary trust-region subproblem:

$$\min_{\bar{v} \in \bar{\Omega}} \sum_{i=\bar{r}+1}^{r} \left( (g_\parallel)_i (v_\parallel)_i + \frac{\lambda_i - \lambda_{\min}}{2} (v_\parallel)_i^2 \right) + g_\perp^T v_\perp + \frac{\delta - \lambda_{\min}}{2} \|v_\perp\|^2,$$

where $\bar{\Omega} = \{\bar{v} \in R^{n-\bar{r}} : \|\bar{v}\| \leq \Delta\}$. This subprolem corresponds to the already considered Case I because its objective function is strictly convex. Let $\bar{\sigma}^*$ and $\bar{v}^* = (v_{\bar{r}+1}^*, v_{\bar{r}+2}^*, \ldots, v_n^*)^T$ be the optimal pair for the auxiliary subproblem. Denote

$$\mu = \begin{cases} 0, & \text{if } \lambda_{\min} = 0, \\ \sqrt{\Delta^2 - \|\bar{v}^*\|^2}, & \text{if } \lambda_{\min} < 0. \end{cases}$$

It can be easily verified that the pair

$$\sigma^* = -\lambda_{\min} + \bar{\sigma}^*, \quad v^* = (\underbrace{\mu, 0, \ldots, 0}_{\bar{r}}, v_{\bar{r}+1}^*, \ldots, v_n^*)^T$$

satisfies the optimality conditions (16) and (17). The vector $v_\perp^*$ is defined by formula (23), but as one can see below, it is not necessary to compute this vector. The same refers to $v_\perp^*$ in Case I.

In each of the cases, we compute, first,

$$g_\parallel = U^T R_\ddagger^{-T} V_\dagger^T g. \tag{24}$$

It is then used for finding $\|g_\perp\|$ from the relation

$$\|g_\perp\|^2 = \|g\|^2 - \|g_\parallel\|^2, \tag{25}$$

which follows from the orthogonality of $P$ represented as

$$P_\perp P_\perp^T = I - P_\parallel P_\parallel^T. \tag{26}$$

The described procedure of finding $\sigma^*$ and $v_\parallel^*$ produces an exact or nearly-exact solution to the trust-region subproblem (7). This solution is computed using (15), (23) and (26) as

$$\begin{aligned} s^* &= P_\parallel v_\parallel^* + P_\perp v_\perp^* = P_\parallel v_\parallel^* - (\delta + \sigma^*)^{-1} P_\perp P_\perp^T g \\ &= -(\delta + \sigma^*)^{-1} g + P_\parallel \left( v_\parallel^* + (\delta + \sigma^*)^{-1} g_\parallel \right). \end{aligned} \tag{27}$$

The presented eigenvalue-based approach to solving the trust-region subproblem has the following attractive feature. Once the eigenvalues of $B$ are computed, in Case I, formula (24) requires $\bar{m}n + O(\bar{m}^2)$ operations[1], and formulas (20) and (21) require $O(\bar{m})$ operations per iteration to approximately solve (13). The computation of $v_\parallel^*$ by formula (22) requires $O(\bar{m})$ operations. In Case II, the computation of $\sigma^*$ and $v_\parallel^*$ has the same order of complexity. The computation of $s^*$ by formula (27) requires a few additional matrix-vector multiplications for $P_\parallel$ defined by (12). The associated cost is $\bar{m}n + O(\bar{m}^2)$.

In the next section, we introduce an alternative eigenvalue-based approach to solving the trust-region subproblem.

# 4 Trust-region subproblem in eigenvalue-based norms

We consider here the trust-region subproblem (7) defined by the norms introduced below. All constituent parts of the compact representation (8) are assumed to be available.

## 4.1 Eigenvalue-based decomposition of the model function

Observe that the new variable defined by (15) allows us to decompose the objective function in (7) as

$$q_P(v) \equiv q(Pv) = q(P_\parallel v_\parallel + P_\perp v_\perp) = q_\parallel(v_\parallel) + q_\perp(v_\perp), \tag{28}$$

where

$$q_\parallel(v_\parallel) \;=\; g_\parallel^T v_\parallel + \frac{1}{2}v_\parallel^T \Lambda v_\parallel = \sum_{i=1}^{r}\left((g_\parallel)_i(v_\parallel)_i + \frac{\lambda_i}{2}(v_\parallel)_i^2\right), \tag{29}$$

$$q_\perp(v_\perp) \;=\; g_\perp^T v_\perp + \frac{\delta}{2}\|v_\perp\|^2. \tag{30}$$

It should be noted that when the trust region is defined by the standard norms like $l_2$ or $l_\infty$, this decomposition does not give any advantage, in contrast to the case of the new norms proposed below.

## 4.2 New norms and related subproblem properties

In this subsection, we introduce two nonstandard norms to define the trust region. The new norms enable us to decompose the original trust-region subproblem into a set of smaller subproblems, which can be easily solved. For one of the new norms, the solution can be written in closed form.

---

[1] Here and in other estimates of computational complexity, it is assumed that $r = \bar{m}$. This corresponds to the maximal number of arithmetic operations.

### 4.2.1 Shape changing norms

To exploit separability of the objective function, we introduce the following norms:

$$\|s\|_{P,\infty} \equiv \max\left(\|P_\|^T s\|_\infty, \|P_\perp^T s\|\right), \tag{31}$$

$$\|s\|_{P,2} \equiv \max\left(\|P_\|^T s\|, \|P_\perp^T s\|\right). \tag{32}$$

Recall that $\|\cdot\|$ stands for the Euclidean norm. It can be easily verified that (31) and (32) do satisfy the vector norm axioms. Since $P$ changes from iteration to iteration, we refer to them as shape-changing norms. The following result establishes a norm equivalence between the new norms and the Euclidean norm with the equivalence factors not depending on $P$.

**Lemma 1** *For any vector $x \in R^n$ and orthogonal matrix $P = [P_\| \ P_\perp] \in R^{n \times n}$, where $P_\| \in R^{n \times r}$ and $P_\perp \in R^{n \times (n-r)}$, the following inequalities hold:*

$$\frac{\|x\|}{\sqrt{r+1}} \leq \|x\|_{P,\infty} \leq \|x\| \tag{33}$$

*and*

$$\frac{1}{\sqrt{2}}\|x\| \leq \|x\|_{P,2} \leq \|x\|. \tag{34}$$

*Here, the lower and upper bounds are attainable.*

*Proof.* We start by justifying the lower bound in (34). The definition (32) gives $\|P_\|^T x\|^2 \leq \|x\|_{P,2}^2$ and $\|P_\perp^T x\|^2 \leq \|x\|_{P,2}^2$. Then we have

$$\|x\|^2 = \|P^T x\|^2 = \|P_\|^T x\|^2 + \|P_\perp^T x\|^2 \leq 2\|x\|_{P,2}^2, \tag{35}$$

which establishes the first of the bounds (34). Further, the inequality above becomes an equality for every $x$ that satisfies $\|P_\|^T x\| = \|P_\perp x\|$, which shows that this bound is attainable.

Due to (35), the second inequality in (34) obviously holds. Notice that it holds with equality for any $x$ that satisfies $P_\|^T x = 0$.

Consider now the norm (31). Since $\|P_\|^T x\|_\infty \leq \|P_\|^T x\|$, we have $\|x\|_{P,\infty} \leq \|x\|_{P,2}$. Then the upper bound in (33) follows from (34). This bound is attainable for the same choice of $x$ as above.

It remains to justify the lower bound in (33). Using the norm definition (31) and the relations between $l_2$ and $l_\infty$ norms, we get

$$\|x\|_{P,\infty}^2 \geq \|P_\|^T x\|_\infty^2 \geq \frac{1}{r}\|P_\|^T x\|^2,$$
$$\|x\|_{P,\infty}^2 \geq \|P_\perp^T x\|^2.$$

Due to (35), these inequalities imply

$$(r+1)\|x\|_{P,\infty}^2 \geq \|P_\|^T x\|^2 + \|P_\perp^T x\|^2 = \|x\|^2.$$

This proves the first inequality in (33). It holds with equality for every $x$ that satisfies $\|P_{\|}^T x\|_\infty = \|P_\perp x\|$. This accomplishes the proof of the lemma. $\qquad\square$

It should be emphasized that the bounds in (33) and (34) do not depend on $n$. Moreover, according to Lemma 1, the norm (32), in contrast to the $l_\infty$ norm, does not differ too much from the $l_2$ norm in the sense of their ratio. The same refers to the other shape-changing norm when $r$ is sufficiently small. For $r = 10$, which is a typical value in our numerical experiments, the norm (31) is not less than approximately one third of the $l_2$ norm.

### 4.2.2 Subproblem separability for the new norms

For the norm (31), the trust region $\Omega$ is defined by the inequalities

$$\|s\|_{P,\infty} \le \Delta \iff \begin{cases} |(v_{\|})_i| \le \Delta, i = 1, \ldots, r, \\ \|v_\perp\| \le \Delta. \end{cases}$$

By combining this with the separability of the model function (28), (29), (30), we get the following separability of the trust-region subproblem:

$$\min_{\|s\|_{P,\infty} \le \Delta} q(s) = \sum_{i=1}^r \min_{|(v_{\|})_i| \le \Delta} \left( (g_{\|})_i (v_{\|})_i + \frac{\lambda_i}{2} (v_{\|})_i^2 \right) \qquad (36)$$

$$+ \min_{\|v_\perp\| \le \Delta} \left( g_\perp^T v_\perp + \frac{\delta}{2} \|v_\perp\|^2 \right).$$

We can write the solution to each of these subproblems in closed form as

$$(v_{\|}^*)_i = \begin{cases} -\frac{1}{\lambda_i} (g_{\|})_i, & \text{if } |(g_{\|})_i| \le \lambda_i \Delta, \ \lambda_i > 0, \\ \zeta, & \text{if } (g_{\|})_i = 0, \ \lambda_i \le 0, \\ -\frac{\Delta}{|(g_{\|})_i|} (g_{\|})_i, & \text{otherwise,} \end{cases} \quad i = 1, \ldots, r; \qquad (37)$$

$$v_\perp^* = -t g_\perp, \qquad (38)$$

where $\zeta = \pm\Delta$ for $\lambda_i < 0$, $\zeta \in [-\Delta, \Delta]$ for $\lambda_i = 0$ and

$$t = \begin{cases} \frac{1}{\delta}, & \text{if } \|g_\perp\| \le \delta\Delta, \\ \frac{\Delta}{\|g_\perp\|}, & \text{otherwise.} \end{cases} \qquad (39)$$

In the original space, the corresponding optimal solution $s^*$ is calculated as

$$s^* = Pv^* = P_{\|} v_{\|}^* + P_\perp v_\perp^*,$$

where $P_\perp v_\perp^* = -t P_\perp P_\perp^T g$. Recalling (26), we finally obtain

$$s^* = -tg + P_{\|}(v_{\|}^* + t g_{\|}). \qquad (40)$$

Here the cost of computing $v_{\|}^*$ by (37) is $O(\bar{m})$. The formulas for $P_{\|}$ (12) and $g_{\|}$ (24) suggest that the dominant cost in (40) is determined by two matrix-vector multiplications involving $V_{\dagger}$. This requires $2\bar{m}n$ operations. Hence, the overall cost of solving the

trust-region subproblem defined by norm (31) is essentially the same as for the Euclidean norm (see Section 3). The advantage of the new norm (31) over the Euclidean norm is a decomposition of the trust-region subproblem that yields the closed-form solution (40) without invoking any iterative procedure.

Consider now the trust region defined by the norm (32). In this case, the trust-region subproblem is decomposed into the two subproblems:

$$\min_{\|s\|_{P,2} \le \Delta} q(s) = \min_{\|v_\|\| \le \Delta} \left( g_\|^T v_\| + \frac{1}{2} v_\|^T \Lambda v_\| \right) + \min_{\|v_\perp\| \le \Delta} \left( g_\perp^T v_\perp + \frac{\delta}{2} \|v_\perp\|^2 \right). \qquad (41)$$

Here, the first subproblem is a low-dimensional case of problem (7). It can be easily solved by any standard trust-region method [14], especially because $\Lambda$ is diagonal. In case of truncated conjugate gradient method, it requires only a few simple operations with $r$-dimensional vectors per one CG iteration. For the dogleg method, it is required to compute the quasi-Newton step $-\Lambda^{-1} g_\|$ and the steepest descent step $-\mu_\| g_\|$, where $\mu_\| = g_\|^T \Lambda g_\| / g_\|^T g_\|$. These operations require $O(\bar{m})$ multiplications. Moreover, the procedure described in Section 3 can be easily adapted for the purpose of finding a nearly-exact solution $v_\|^*$ to the first subproblem.

The second subproblem in (41) is the same as in (36) with the optimal solution $v_\perp^*$ defined by formulas (38) and (39). Then one can show, as above, that the solution to (41) is of the form (40). The same formula is applied to finding an approximate solution to the trust-region subproblem (41) when $v_\|^*$ represents an approximate solution to the first subproblem.

# 5 Algorithm

In Algorithm 1, we present a generic trust-region framework [14] in the form close to our implementation (see Section 9 for details). In this algorithm, the trust-region subproblem (2) is assumed to be defined by a vector norm $\|\cdot\|_k$. This norm may differ from the Euclidean norm, and moreover, it may change from iteration to iteration, like the norms (31) and (32).

We say that the trust-region subproblem is solved with *sufficient accuracy*, if there exists a scalar $0 < c < 1$ such that the inequality

$$q_k(s_k) \le -c \|g_k\|^2 \min \left( \frac{1}{\|B_k\|}, \frac{\Delta}{\|g_k\|_k} \right)$$

holds for all $k \ge 0$. In other words, the model decrease is at least a fixed fraction of that attained by the Cauchy point [14]. The sufficient accuracy property plays an important role in proving global convergence of inexact trust-region methods.

# 6 Convergence Analysis

In Algorithm 1, we assume that if the norm is defined by (31), then the exact solution is found as described in Section 4.2.2. In case of norm (32), we assume that the first

**Algorithm 1** Trust-Region Method

---

**Require:** $x_0 \in R^n$, $\Delta_0 > 0$, $\epsilon > 0$, $\delta_0 > 0$, $0 \le \tau_1 < \tau_2 < 0.5 < \tau_3 < 1$,
  $0 < c_1 < c_2 \le 0.5 < c_3 < 1 < c_4$
  Compute $g_0$ and $B_0 = \delta_0 I$
  **for** $k = 0, 1, 2, \ldots$ **do**
    **if** $\|g_k\| \le \epsilon$ **then**
      **return**
    **end if**
    Find $s_k$ that solves (2) with sufficient accuracy
    Compute the ratio $\rho_k = \frac{f(x_k + s_k) - f(x_k)}{q_k(s_k)}$
    **if** $\rho_k \ge \tau_1$ **then**
      $x_{k+1} = x_k + s_k$
      Compute $g_{k+1}$ and update $B_{k+1}$
    **else**
      $x_{k+1} = x_k$
    **end if**
    **if** $\rho_k < \tau_2$ **then**
      $\Delta_{k+1} = \min\left(c_1 \Delta_k, c_2 \|s_k\|_k\right)$
    **else**
      **if** $\rho_k \ge \tau_3$ **and** $\|s_k\|_k \ge c_3 \Delta_k$ **then**
        $\Delta_{k+1} = c_4 \Delta_k$
      **else**
        $\Delta_{k+1} = \Delta_k$
      **end if**
    **end if**
  **end for**

---

subproblem in (41) is solved with sufficient accuracy and the second subproblem is solved exactly. This, according to the following result, guarantees that the whole trust-region subproblem is solved with sufficient accuracy.

**Lemma 2** *Let* $v = (v_\parallel, v_\perp)^T$ *be a solution to the trust-region subproblem* (41), *such that*

$$q_\parallel(v_\parallel) \leq -c_0 \|g_\parallel\| \min\left(\frac{\|g_\parallel\|}{\|\Lambda\|}, \Delta\right) \tag{42}$$

*for some* $0 < c_0 < 1$ *and* $v_\perp$ *is the exact solution to the second subproblem defined by* (38) *and* (39). *Suppose that* $g \neq 0$, *then*

$$q_P(v) \leq -c\|g\|^2 \min\left(\frac{1}{\|B\|}, \frac{\Delta}{\|g\|_{P,2}}\right), \tag{43}$$

*where* $c = \min(c_0, \frac{1}{2})$.

*Proof.* Since $v_\perp$ is the Cauchy point for the second subproblem, the following inequality holds (see, e.g., [34, Lemma 4.3]):

$$q_\perp(v_\perp) \leq -\frac{1}{2}\|g_\perp\| \min\left(\frac{\|g_\perp\|}{|\delta|}, \Delta\right). \tag{44}$$

Since $P$ is orthogonal, the eigenvalue decomposition of $B$ (11) implies

$$\|B\| = \left\|\begin{array}{cc} \Lambda & 0 \\ 0 & \delta I_{n-r} \end{array}\right\| = \max\left(\|\Lambda\|, |\delta|\right). \tag{45}$$

By the norm definition (32), we have $\|g\|_{P,2} = \max(\|g_\parallel\|, \|g_\perp\|)$. This formula along with (42), (44) and (45) yield

$$q_\parallel(v_\parallel) \leq -c\|g_\parallel\| \min\left(\frac{\|g_\parallel\|}{\|B\|}, \Delta\frac{\|g_\parallel\|}{\|g\|_{P,2}}\right) = -c\|g_\parallel\|^2 \min\left(\frac{1}{\|B\|}, \frac{\Delta}{\|g\|_{P,2}}\right),$$
$$q_\perp(v_\perp) \leq -c\|g_\perp\| \min\left(\frac{\|g_\perp\|}{\|B\|}, \Delta\frac{\|g_\perp\|}{\|g\|_{P,2}}\right) = -c\|g_\perp\|^2 \min\left(\frac{1}{\|B\|}, \frac{\Delta}{\|g\|_{P,2}}\right).$$

Combining these inequalities with (25), we finally obtain the inequality

$$q_\parallel(v_\parallel) + q_\perp(v_\perp) \leq -c\|g\|^2 \min\left(\frac{1}{\|B\|}, \frac{\Delta}{\|g\|_{P,2}}\right).$$

Then the trust-region decomposition (28) implies (43). This accomplishes the proof. □

**Corollary 3** *If inequality* (42) *holds for all* $k \geq 0$, *where* $c_0$ *does not depend on* $k$, *then the trust-region subproblem* (41) *is solved with sufficient accuracy.*

Although the shape of the trust region defined by the new norms changes from iteration to iteration, it turns out that Algorithm 1, where the trust region subproblem is solved as proposed in Section 4.2.2, converges to a stationary point. This fact is justified by the following result.

**Theorem 4** *Let* $f : R^n \to R$ *be twice-continuously differentiable and bounded from below on* $R^n$. *Suppose that there exists a scalar* $c_1 > 0$ *such that*

$$\|\nabla^2 f(x)\| \leq c_1$$

*for all* $x \in R^n$. *Consider the infinite sequence* $\{x_k\}$ *generated by Algorithm 1, in which the norm* $\|\cdot\|_k$ *is defined by any of the two formulas* (31) *or* (32), *and the stopping criterion is suppressed. Suppose also that there exists a scalar* $c_2 > 0$ *such that*

$$\|B_k\| \leq c_2, \quad \forall k \geq 0. \tag{46}$$

*Then*

$$\lim_{k \to \infty} \|\nabla f(x_k)\| = 0. \tag{47}$$

*Proof.* By Lemma 1, there holds the equivalence between the norms $\|\cdot\|_k$ and the Euclidean norm where the coefficients in the lower and upper bounds do not depend on $k$. Moreover, Algorithm 1 explicitly requires that the trust-region subproblem is solved with sufficient accuracy. All this and the assumptions of the theorem allow us to apply here Theorem 6.4.6 in [14] which proves the convergence (47). □

In the case of convex $f(x)$, Theorem 4 holds for the L-BFGS updates due to the boundedness of $B_k$ established, i.e., in [33].

Suppose now that $f(x)$ is not necessarily convex. Consider the boundedness of $B_k$ for the limited memory versions of BFGS, SR1 and stable multipoint symmetric secant updates [4, 5, 6, 8]. Let $K_k$ denote the sequence of the iteration indexes of those pairs $\{s_i, y_i\}$ that are involved in generating $B_k$ starting from an initial Hessian approximation $B_k^0$. The number of such pairs is assumed to be limited by $m$, i.e.

$$|K_k| \leq m. \tag{48}$$

In L-BFGS, the positive definiteness of $B_k$ can be enforced by composing $K_k$ of only those indexes of the recently generated pairs $\{s_i, y_i\}$ that satisfy the inequality

$$s_i^T y_i > c_3 \|s_i\| \|y_i\| \tag{49}$$

for a positive constant $c_3$ (see [34]). This requirement permits us to show in the following lemma that the boundedness of $B_k$, and hence Theorem 4, hold in the nonconvex case.

**Lemma 5** *Suppose that the assumptions of Theorem 4 concerning* $f(x)$ *are satisfied. Let all* $B_k$ *be generated by the L-BFGS updating formula. Let the updating start at each*

*iteration from $B_k^0$ and involve the pairs $\{s_i, y_i\}_{i \in K_k}$, whose number is limited in accordance with (48). Suppose that there exists a constant $c_3 \in (0, 1)$ such that, for all $k$, (49) is satisfied. Suppose also that there exists a constant $c_4 > 0$ such that the inequality*

$$\|B_k^0\| \le c_4, \quad \forall k \ge 0, \tag{50}$$

*holds with the additional assumption that $B_k^0$ is positive semi-definite. Then there exists a constant $c_2 > 0$ such that (46) holds.*

*Proof.* For each $i \in K_k$, the process of updating by the L-BFGS formula the current Hessian approximation $B$ (initiated with $B_k^0$) can be presented as follows

$$B^{new} = B^{\frac{1}{2}} \left( I - \frac{B^{\frac{1}{2}} s_i s_i^T B^{\frac{1}{2}}}{\|B^{\frac{1}{2}} s_i\|^2} \right) B^{\frac{1}{2}} + \frac{y_i y_i^T}{s_i^T y_i}.$$

This equation along with inequality (49) give

$$\|B^{new}\| \le \|B\| + \frac{\|y_i\|}{c_3 \|s_i\|},$$

where in accordance with the boundedness of $\nabla_{xx} f(x)$ we have $\|y_i\| \le c_1 \|s_i\|$. After summing these inequalities over all $i \in K_k$, we obtain the inequality

$$\|B_k\| \le \|B_k^0\| + mc_1/c_3,$$

which, due to (50), finally proves inequality (46) for $c_2 = c_4 + mc_1/c_3$. $\qquad \square$

The boundedness of $B_k$ generated by SR1 and stable multipoint symmetric secant updates will be proved in a separate paper, which will be focused on the case of nonconvex $f(x)$.

To guarantee the boundedness of $B_k$ generated by the limited memory version of SR1, we require that the pairs $\{s_i, y_i\}_{i \in K_k}$ satisfy, for a positive constant $c_3$, the inequality

$$|s_i^T (y_i - Bs_i)| > c_3 \|s_i\| \|y_i - Bs_i\|,$$

where $B$ is the intermediate matrix to be updated based on the pair $\{s_i, y_i\}$ in the process of generating $B_k$. This makes the SR1 updates well defined (see [13, 34]).

The stable multipoint symmetric secant updating process is organized in the way that a uniform linear independence of the vectors $\{s_i\}_{i \in K_k}$ is maintained. The Hessian approximations $B_k$ are uniquely defined by the equations

$$s_i^T B_k s_j = s_i^T y_j, \quad p^T B_k s_l = p^T y_l, \quad p^T B_k p = p^T B_k^0 p, \tag{51}$$

which hold for all $i, j, l \in K_k$, $i < j$, and also for all $p \in R^n$, such that $p^T s_t = 0$ for all $t \in K_k$. The boundedness of the generated approximations $B_k$ in the case of nonconvex $f(x)$ follows from the mentioned uniform linear independence and equations (51).

# 7 Implementation details for L-BFGS

In this section, we consider the Hessian approximation $B$ in (7) defined by the L-BFGS update [11]. It requires to store at most $m$ pairs of vectors $\{s_i, y_i\}$ obtained at those of the most recent iterations for which (49) holds. As it was mentioned above, the number of stored pairs is assumed, for simplicity, to be equal to $m$. The compact representation (8) of the L-BFGS update has the form

$$B = \delta I - [S\ Y] \begin{bmatrix} S^T S/\delta & L/\delta \\ L^T/\delta & -E \end{bmatrix}^{-1} \begin{bmatrix} S^T \\ Y^T \end{bmatrix}, \tag{52}$$

in which case $\bar{m} = 2m$. In terms of (8), the matrix $V = [S\ Y]$ is composed of the stored pairs (5) in the way that the columns of $S = [\ldots, s_i, \ldots]$ and $Y = [\ldots, y_i, \ldots]$ are sorted in increasing iteration index $i$. The sequence of these indexes may have some gaps that correspond to the cases, in which (49) is violated. The matrix $W$ is the inverse of a $2m \times 2m$-matrix, which contains a strictly lower triangular part of the matrix $S^T Y$, denoted in (52) by $L$, and the main diagonal of $S^T Y$, denoted by $E$.

At iteration $k$ of L-BFGS, the Hessian approximation of $B_k$ is determined by the stored pairs $\{s_i, y_i\}$ and the initial Hessian approximation $\delta_k I$. The most popular choice of the parameter $\delta_k$ is defined, like in [34], by the formula

$$\delta_k = \frac{y_k^T y_k}{s_k^T y_k}, \tag{53}$$

which represents the most recent curvature information about the function.

## 7.1 Uniform representation of eigenvalue-based solutions

Recall that the approaches presented above rely on the implicitly defined RRQR decomposition of $V$ and eigenvalue decomposition of $B$. In this section, we show that each of the eigenvalue-based solutions of the considered trust-region subproblems (7) can be presented as

$$s^* = -\alpha g + V_{\dagger} p, \tag{54}$$

where $\alpha$ is a scalar and

$$p = R_{\ddagger}^{-1} U(v_{\parallel}^* + g_{\parallel}).$$

The specific values of $\alpha$ and $v_{\parallel}^*$ are determined by the norm defining the trust region and the solution to the trust region subproblem.

Let us firstly consider the trust-region subproblem defined by the Euclidean norm. Due to (12), we can rewrite formula (27) for a nearly-exact solution $s^*$ in the form (54), where $\alpha = (\delta + \sigma^*)^{-1}$.

Consider now the trust-region subproblem (36) defined by the norm (31). Its solution can be represented in the form (54), where $\alpha$ stands for $t$ defined by (39). Note that since the Hessian approximations generated by the L-BFGS update are positive definite, the case

of $\lambda_i \leq 0$ in (37) is excluded. Therefore, the optimal solution to the first subproblem in (36) is computed as $v_\parallel^* = -Ag_\parallel$, where $A \in R^{r \times r}$ is a diagonal matrix defined as

$$A_{ii} = \begin{cases} \frac{1}{\lambda_i}, & \text{if } |(g_\parallel)_i| \leq \lambda_i \Delta, \\ \frac{\Delta}{|(g_\parallel)_i|}, & \text{otherwise.} \end{cases}$$

When the trust region subproblem (41) is defined by the norm (32) and $v_\parallel^*$ is an approximate solution to the first subproblem in (41), formula (54) holds for the same $\alpha = t$.

In each of the considered three cases, the most expensive operations, $4mn$, are the two matrix-vector multiplications $V_\dagger^T g$ and $V_\dagger p$. The linear systems involving the triangular matrix $R$ can be solved at a cost of $O(m^2)$ operations.

## 7.2 Model function evaluation

In Algorithm 1, the model function value is used to decide whether to accept the trial step. Let $s^*$ denote a nearly-exact or exact solution to the trust-region subproblem. In this subsection, we show how to reduce the evaluation of $q(s^*)$ to cheap manipulations with the available low-dimensional matrix $V^T V$ and vector $V^T g$. It is assumed that $\|g\|^2$ has also been calculated before the model function evaluation.

Consider, first, the trust-region subproblem defined by the Euclidean norm. Suppose that $s^*$ is of the form (54) and satisfies (14) for $\sigma^* \geq 0$. Then

$$
\begin{aligned}
q(s^*) &= g^T s^* - \frac{1}{2}(g + \sigma^* s^*)^T s^* = \frac{1}{2}\left(g^T s^* - \sigma^* \|s^*\|^2\right) \qquad (55) \\
&= \frac{1}{2}\left(-\alpha\|g\|^2 + p^T(V_\dagger^T g) - \sigma^* \|s^*\|^2\right),
\end{aligned}
$$

where $\|s^*\|^2$ is calculated by the formula

$$\|s^*\|^2 = \alpha^2 \|g\|^2 - 2\alpha p^T(V_\dagger^T g) + p^T(V_\dagger^T V_\dagger)p.$$

Thus, the most expensive operation in calculating $q(s^*)$ is the multiplication of the matrix $V_\dagger^T V_\dagger$ by the vector $p$ at a cost of $O(m^2)$ operations. Note that this does not depend on whether the eigenvalue decomposition is used for computing $s^*$.

Consider now the trust-region subproblem defined by any of our shape-changing norms. Let $v_\parallel^*$ be the available solution to the first of the subproblems in (36) or (41), depending on which norm, (31) or (32), is used. The separability of the model function (28) and formulas (25), (38) give

$$q(s^*) = (g_\parallel)^T v_\parallel^* + \frac{1}{2}(v_\parallel^*)^T \Lambda v_\parallel^* + \left(t^2 \delta/2 - t\right)\left(\|g\|^2 - \|g_\parallel\|^2\right).$$

One can see that only cheap operations with $r$-dimensional vectors are required for computing $q(s^*)$.

In the next subsection, we show how to exploit the uniform representation of the trust-region solution (54) for efficiently implementing the L-BFGS update once the trial step is accepted.

## 7.3 Updating Hessian approximation

The updating of the Hessian approximation $B$ is based on updating the matrices $S^T S$, $L$ and $E$ in (52), which, in turn, is based on updating the matrix $V^T V$. Restoring the omitted subscript $k$, we note that the matrix $V_{k+1}$ is obtained from $V_k$ by adding the new pair $\{s_k, y_k\}$, provided that (49) holds, and possibly removing one of the old pairs. Hence, the updating procedure for $V_k^T V_k$ requires computing $V_k^T s_k$ and $V_k^T y_k$ after computing $s_k$. The straightforward implementation would require $4mn$ operations. It is shown below how to implement these matrix-vector products more efficiently.

Assuming that $V^T g$, $V^T V$ and $p$ have already been computed, we conclude from formula (54) that the major computational burden in

$$V_k^T s_k = V^T s = -\alpha V^T g + V^T V_\dagger p \tag{56}$$

is associated with computing $(V^T V_\dagger) \cdot p$ at a cost of $4m^2$ multiplications. Recalling that $y_k = g_{k+1} - g_k$, we observe that

$$V_k^T y_k = V_k^T g_{k+1} - V_k^T g_k \tag{57}$$

is a difference between two $2m$-dimensional vectors, of which $V_k^T g_k (= V^T g)$ is available and $V_k^T g_{k+1}$ is calculated at a cost of $2mn$ operations. Then at the next iteration, the vector $V_{k+1}^T g_{k+1}$ can be obtained from $V_k^T g_{k+1}$ at a low cost, because these two vectors differ only in two components.

Thus, $V_k^T s_k$ and $V_k^T y_k$ can be computed by formulas (56) and (57) at a cost in which $2mn$ is a dominating term. This cost is associated with computing $V_k^T g_{k+1}$ and allows for saving on the next iteration the same $2mn$ operations on computing $V_{k+1}^T g_{k+1}$.

In the next subsection, we discuss how to make the implementation of our approaches more numerically stable.

## 7.4 Numerical stability

Firstly, in our numerical experiments we observed that the Gram matrix $V^T V$ updated according to (56) and (57) was significantly more accurate if we used normalized vectors $s/\|s\|$ and $y/\|y\|$ instead of $s$ and $y$, respectively. More importantly, the columns of $R$ produced by the Cholesky factorization are, in this case, also of unit length. This is crucial for establishing rank deficiency of $V$ in the way described below. It can be easily seen that the compact representation of $B$ (52) takes the same form for $V$ composed of the normalized vectors. To avoid $2n$ operations, the normalized vectors are actually never formed, but the matrix-vector multiplications involving $V$ are preceded by multiplying the vector by a $2m \times 2m$ diagonal matrix whose diagonal elements are of the form $1/\|s\|$, $1/\|y\|$.

Secondly, at the first $2m$ iterations, the matrix $V$ is rank-deficient, i.e., $r = \text{rank}(V) < 2m$. The same may hold at the subsequent iterations. To detect linear dependence of the columns of $V$, we used the diagonal elements of $R$. In the case of normalized columns, $R_{ii}$ is equal to $\sin \psi_i$, where $\psi_i$ is the angle between the $i$-th column of $V$ and the linear

subspace, generated by the columns $1, 2, \ldots, i - 1$. We introduced a threshold parameter $\nu \in (0, 1)$. It remains fixed for all $k$, and only those parts of $V$ and $R$, namely, $V_\dagger$, $R_\dagger$ and $R_\ddagger$, which correspond to $|R_{ii}| > \nu$, were used for computing $s^*$.

## 7.5 Computational complexity

The cost of one iteration is estimated as follows. The Cholesky factorization of $V^T V \in R^{2m \times 2m}$ requires $O(m^3)$ operations, or only $O(m^2)$ if it is taken into account that the new $V$ differs from the old one in a few columns. Computing $R_\dagger W R_\dagger^T \in R^{2m \times 2m}$, where $W$ is the inverse of a $2m \times 2m$ matrix, takes $O(m^3)$ operations. The eigenvalue decomposition for $R_\dagger W R_\dagger^T$ costs $O(m^3)$. Note that $O(m^3) = \left( \frac{m^2}{n} \right) O(mn)$ is only a small fraction of $mn$ operations when $m \ll n$. Since $V^T g$ is available from the updating of $B$ at the previous iteration, the main cost in (54) for calculating $s^*$ is associated with the matrix-vector product $V_\dagger p$ at a cost of $2mn$ operations. The Gram matrix $V^T V$ is updated by formulas (56) and (57) at a cost of $2mn + O(m^2)$ operations. Thus, the dominating term in the overall cost is $4mn$, which is the same as for the line-search versions of the L-BFGS.

## 7.6 Computing the quasi-Newton step

In our numerical experiments, we observed that, for the majority of the test problems, the quasi-Newton step $s_N = -B^{-1} g$ was accepted at more than 75% of iterations. If the trust region is defined by the Euclidean norm, we can easily check if this step belongs to the trust region without calculating $s_N$ or the eigenvalues of $B$. Indeed, consider the following compact representation of the inverse Hessian approximation [11]:

$$B^{-1} = \gamma I + [S\,Y] M \begin{bmatrix} S^T \\ Y^T \end{bmatrix}. \tag{58}$$

Here $\gamma = \delta^{-1}$ and the symmetric matrix $M \in R^{2m \times 2m}$ is defined as

$$M = \begin{bmatrix} T^{-T}(E + \gamma Y^T Y) T^{-1} & -\gamma T^{-T} \\ -\gamma T^{-T} & 0 \end{bmatrix},$$

where the matrix $T = S^T Y - L$ is upper-triangular. Then, since $Y^T Y$ and $T$ are parts of $V^T V$, the norm of the quasi-Newton step can be computed as

$$\|s_N\|^2 = \gamma^2 \|g\|^2 + 2\gamma (V^T g)^T M V^T g + \|V M V^T g\|^2. \tag{59}$$

The operations that involve the matrix $M$ can be efficiently implemented as described in [11, 29]. Formula (59) requires only $O(m^2)$ operations because $V^T g$ and $\|g\|$ have already been computed. If $\|s_N\| \leq \Delta$, the representation (58) of $B^{-1}$ allows for directly computing $s_N$ without any extra matrix factorizations. The dominating term in the cost of this operation is $4mn$. The factorizations considered above are used only when the quasi-Newton step is rejected and it is then required to solve the trust-region subproblem.

When $s_N$ is used as a trial step, the saving techniques discussed in Sections 7.2 and 7.3 can be applied as follows. The uniform representation (54) holds for $s_N$ with $V_\dagger$ replaced by $V$, and

$$\alpha = \gamma \quad \text{and} \quad p = -MV^T g.$$

This allows for evaluating the model function in a cheap way by formula (55), where $\sigma^* = 0$. Moreover, $V^T s_N$ can be computed by formula (56), which saves $2mn$ operations.

Note that if $\|s_N\| \le \Delta$ in the Euclidean norm, then, by Lemma 1, $s_N$ belongs to the trust region defined by any of our new norms. This permits us to use formula (59) for checking if $s_N$ is guaranteed to belong to the trust region in the corresponding norm.

# 8   Alternative limited memory trust-region approaches, improved versions

In this section, we describe in more detail some of those approaches mentioned in Section 1, which combine limited memory and trust-region techniques, namely, the algorithms proposed in [10, 29]. They both use the L-BFGS approximation and the Euclidean norm. We propose below improved versions of these algorithms. They do not require the eigenvalue decomposition. The purpose of developing the improved versions was to compare them with our eigenvalue-based algorithms. A comparative study is performed in Section 9.

## 8.1   Nearly-exact trust-region algorithm

A nearly-exact trust-region algorithm was proposed by Burke et al. [10]. It does not formally fall into the conventional trust-region scheme, because at each iteration, the full quasi-Newton step is always computed at a cost of $4mn$ operations like in [11] and used as a trial step independently of its length. If it is rejected, the authors exploit the Sherman-Morrison-Woodbury formula and obtain the following representation:

$$(B + \sigma I)^{-1} = (\delta + \sigma)^{-1} \left( I - V \left( (\delta + \sigma)^{-1} W^{-1} + V^T V \right)^{-1} V^T \right).$$

Furthermore, by exploiting a special structure of $W$ for the BFGS update (52), a triangular factorization of a $2m \times 2m$ matrix $(\delta + \sigma)^{-1} W^{-1} + V^T V$ is computed using Cholesky factorizations of two $m \times m$ matrices. This allows for efficiently implementing Newton's iterations in solving (13), which, in turn, requires solving in $u, w \in R^{2m}$ the following system of linear equations:

$$\begin{cases} \left( (\delta + \sigma) W^{-1} + V^T V \right) u = V^T g \\ \left( (\delta + \sigma) W^{-1} + V^T V \right) w = W^{-1} u \end{cases}. \tag{60}$$

Then, a new trial step

$$s^* = -(\delta + \sigma^*)^{-1} \left( g - V \left( (\delta + \sigma^*) W^{-1} + V^T V \right)^{-1} V^T g \right) \tag{61}$$

is computed at an additional cost of $2mn + O(m^3)$ operations. The authors proposed also to either update $S^T S$ and $L$ after each successful iteration at a cost of $2mn$ operations or,

22

alternatively, compute, if necessary, these matrices at $2m^2n$ operations only. Thus, the worst case complexity of one iteration is $6mn + O(m^2)$ or $2m^2n + 2mn + O(m^3)$.

We improve the outlined algorithm as follows. The full quasi-Newton step $s_N$ is not computed first, but only $\|s_N\|$ by formula (59) at a cost of $2mn + O(m^2)$ operations, which allows to check if $s_N \in \Omega$. Then only one matrix-vector multiplication of the additional cost $2mn$ is required to compute the trial step, independently on whether it is the full quasi-Newton step. The matrices $S^TS$ and $L$ are updated after each successful iteration at a cost of $O(m^2)$ operations as it was proposed in Section 7.3, because formula (61) can be presented in the form (54) with $V_{\dagger}$ replaced by $V$, and

$$\alpha = (\delta + \sigma^*)^{-1} \quad \text{and} \quad p = (\delta + \sigma^*)^{-1} \left( (\delta + \sigma^*)W^{-1} + V^TV \right)^{-1} V^Tg.$$

The cost of the model function evaluation is estimated in Section 7.2 as $O(m^2)$. The proposed modifications allow us to reduce the cost of one iteration to $4mn + O(m^3)$ operations.

## 8.2   Inexact trust-region algorithm

The ways of solving trust-region subproblems considered so far in this paper were either exact or nearly-exact. In this subsection, we consider inexact algorithms proposed by Kaufman in [29], where the trust-region subproblem is approximately solved with the help of the double-dogleg approach [15]. At each iteration of the algorithms, the Hessian and its inverse are simultaneously approximated using the L-BFGS updates. Techniques, similar to those presented in Section 7.3 are applied. The main feature of these algorithms is that the parameter $\delta$ is fixed, either for a series of iterations followed by a restart, or for all iterations. Here the restart means removing all stored pairs $\{s_i, y_i\}$. The reason for fixing $\delta$ is related to author's intention to avoid computational complexity above $O(m^2)$ in manipulations with small matrices. As it was mentioned in [29], the performance of the algorithms is very sensitive to the choice of $\delta$. In the line-search L-BFGS algorithms, the parameter $\delta$ is adjusted after each iteration, which is aimed at estimating the size of the true Hessian along the most recent search direction. This explains why a good approximation of the Hessian matrix in [29] requires larger values of $m$ than in the case of the line-search L-BFGS.

We propose here the following algorithm based on the double-dogleg approach with $\delta$ changing at each iteration. It combines our saving techniques with some of those used in [29]. The Gram matrix $V^TV$ is updated like in Section 7.3. In accordance with the double-dogleg approach, an approximate solution to the trust-region subproblem is found by minimizing the model function along a piecewise linear path that begins in $s = 0$, ends in the quasi-Newton step and has two knots. One of them is the Cauchy point

$$s_C = -\min \left( \frac{\|g\|^2}{g^TBg}, \frac{\Delta}{\|g\|} \right) g \equiv -\mu g.$$

The other knot is the point $\tau s_N$ on the quasi-Newton direction, where $\tau \in (0, 1)$ is such that $q(\tau s_N) < q(s_C)$ and $\|\tau s_N\| > \|s_C\|$. Since $q(s)$ and $\|s\|$ are monotonically decreasing

and increasing, respectively, along the double-dogleg path, the minimizer of $q(s)$ on the feasible segment of the path is the endpoint of the segment. Thus,

$$s = \begin{cases} s_N, & \text{if } \|s_N\| \leq \Delta, \\ (\Delta/\|s_N\|)s_N, & \text{if } \|\tau s_N\| \leq \Delta < \|s_N\|, \\ s_C + \theta(\tau s_N - s_C), & \text{otherwise}, \end{cases} \tag{62}$$

where $\theta \in [0, 1)$ is such that $\|s\| = \Delta$. In our implementation, we used

$$\tau = 0.2 + 0.8\|g\|^4 / \left((g^T B^{-1} g)(g^T B g)\right),$$

as suggested in [15].

At each iteration, we first compute the full quasi-Newton step using (58) and its norm by (59). If this step belongs to the trust region, it is then used as a trial point. Otherwise, the Cauchy point and $\tau$ are computed, which requires $O(m)$ operations for calculating

$$g^T B^{-1} g = \gamma\|g\|^2 + (V^T g)^T (MV^T g),$$

where the $2m$-dimensional vectors $V^T g$ and $MV^T g$ have already been computed for $s_N$. The additional $O(m^3)$ operations are required for calculating

$$g^T B g = \delta\|g\|^2 + (V^T g)^T W(V^T g),$$

where $W$ is the inverse of a $2m \times 2m$ matrix. Note that in our implementation the multiplication of the matrix $W$ by the vector $V^T g$ is done as in [11]. This cannot be implemented at a cost of $O(m^2)$ operations like in [29], because $\delta$ is updated by formula (53) after each successful iteration. Note that $\theta$ in (62) can be computed at a cost of $O(1)$ operations. To show this, denote $\hat{s} = \tau s_N - s_C$. Then

$$\theta = \frac{\beta}{\psi + \sqrt{\psi^2 + \|\hat{s}\|^2 \beta}},$$

where $\beta = \Delta^2 - \|s_C\|^2$ and $\psi = \hat{s}^T s_C$. Observing that

$$\begin{aligned} \|s_C\| &= \mu\|g\| = \min\left(\frac{\|g\|^3}{g^T B g}, \Delta\right), \\ \psi &= \tau s_N^T s_C - \|s_C\|^2 = \tau\mu g^T B^{-1} g - \|s_C\|^2, \\ \|\hat{s}\|^2 &= \tau^2\|s_N\|^2 - 2\tau s_C^T s_N + \|s_C\|^2 = \tau^2\|s_N\|^2 - 2\psi - \|s_C\|^2, \end{aligned}$$

one can see that the computation of $\theta$ involves just a few scalar operations.

For estimating the cost of computing the double-dogleg solution, consider separately the two cases depending on whether the trial step was accepted at the previous iteration, or not. In the former case, the major computational burden in finding $s$ by formula (62) is related to computing the quasi-Newton step $s_N$ at a cost of $4mn$ operations. Otherwise, the new trial step requires only $O(n)$ operations, because $s_N$ is available and $s_C$ is updated for the new $\Delta$ at this cost.

According to (62), the double-dogleg solution is a linear combination of the gradient and the quasi-Newton step, i.e., $s = \alpha_1 g + \alpha_2 s_N$, where $\alpha_1, \alpha_2$ are scalars. Then the model function in the trial point is computed by the formula

$$
\begin{aligned}
q(\alpha_1 g + \alpha_2 s_N) &= \alpha_1 \|g\|^2 - \alpha_2 g^T B^{-1} g + (\alpha_1 g + \alpha_2 s_N)^T B(\alpha_1 g + \alpha_2 s_N)/2 \\
&= \alpha_1 \|g\|^2 - \alpha_2 g^T B^{-1} g + (\alpha_1^2 g^T B g - 2\alpha_1 \alpha_2 \|g\|^2 + \alpha_2^2 g^T B^{-1} g)/2 \\
&= (\alpha_1 - \alpha_1 \alpha_2)\|g\|^2 - (\alpha_2 - \alpha_2^2/2)g^T B^{-1} g + (\alpha_1^2/2)g^T B^T g
\end{aligned}
$$

at a cost of $O(1)$ operations.

As it was shown in Section 7.6, the representation (54) holds for the quasi-Newton step $s_N$. The same obviously refers to the second alternative in formula (62). In the case of the third alternative in (62), representation (54) holds with $V_\dagger$ replaced by $V$, and

$$
\alpha = (1 - \theta)\mu + \theta\tau\gamma \quad \text{and} \quad p = -\theta\tau M V^T g.
$$

This allows for applying the saving techniques presented in Sections 7.2 and 7.3. Therefore, the worst case complexity of one iteration of our inexact algorithm is $4mn + O(m^3)$. It is the same as for the proposed above exact and nearly-exact algorithms. However, the actual computational burden related to the term $O(m^3)$ and required for implementing the product $W \cdot (V^T g)$ in accordance with [11] is lower for our double-dogleg algorithm because it comes from one Cholesky factorization of a smaller $m \times m$ matrix. Moreover, if in our algorithm the trial step is rejected, the calculation of the next trial step requires, as mentioned earlier, only $O(n)$ operations, whereas the same requires at least $2mn$ operations in the other algorithms proposed above.

# 9    Numerical tests

The developed here limited memory trust-region algorithms were implemented in MATLAB R2011b. The numerical experiments were performed on a Linux machine HP Compaq 8100 Elite with 4 GB RAM and quad-core processor Intel Core i5-650 (3,20 GHz).

All our implementations of the trust-region approach were based on Algorithm 1 whose parameters were chosen as $\delta_0 = 1$, $\tau_1 = 0$, $\tau_2 = 0.25$, $\tau_3 = 0.75$, $c_1 = 0.25$, $c_2 = 0.5$, $c_3 = 0.8$, $c_4 = 2$. The very first trial step was obtained by a backtracking line search along the steepest descent direction, where the trial step-size was increased or decreased by factor two. For a fairer comparison with line-search algorithms, the number of accepted trial steps was counted as the number of iterations. In such a case, each iteration requires at most two gradient evaluations (see below for details). To take into account the numerical errors in computing $\rho_k$, we adopted the techniques discussed in [14, 28] by setting $\rho_k = 1$ whenever

$$
|f(x_k + s_k) - f(x_k)| \leq 10^{-11} \cdot |f(x_k)|. \tag{63}
$$

This precaution may result in a small deterioration of the objective function value, but it helps to prevent from stopping because of too small trust region. The most recent pair $\{s_k, y_k\}$ was not stored if

$$
s_k^T y_k \leq 10^{-8} \cdot \|s_k\| \cdot \|y_k\|.
$$

The stopping criterion in Algorithm 1 was

$$\|g_k\| \leq 10^{-5} \cdot \max\left(1, \|x\|\right).$$

We also terminated algorithm and considered it failed when the trust-region radius was below $10^{-15}$ or the number of iterations exceeded 100000.

Algorithms were evaluated on 62 large-scale ($1000 \leq n \leq 10000$) CUTEr test problems [27] with their default parameters. The average run time of algorithm on each problem was calculated on the base of 10 runs. The following problems were excluded from the mentioned set: PENALTY2, SBRYBND, SCOSINE, SCURLY10, SCURLY20, SCURLY30, because all tested algorithms failed; CHAINWOO, because the algorithms converged to local minima with different objective function values; FLETCBV2, because it satisfied the stopping criterion in the starting point; FMINSURF, because we failed to decode it.

One of the features of CUTEr is that it is computationally faster to make simultaneous evaluation of function and gradient in one call instead of two separate calls. In Algorithm 1, the function is evaluated in all trial points, while the gradient is evaluated in accepted trial points only. We observed that, for the majority of the test problems, the quasi-Newton step was accepted at more than 75% of iterations. Then we decided to simultaneously evaluate $f(x)$ and $\nabla f(x)$ in one call whenever the quasi-Newton step belongs to the trust region, independently on whether the corresponding trial point is subsequently accepted.

We used performance profiles [16] to compare algorithms. This is done as follows. For each problem $p$ and solver $s$, denote

$$t_{p,s} = \text{the result gained in solving problem } p \text{ by solver } s,$$

which can be, e.g., the CPU time, the number of iterations, or the number of function evaluations. The performance ratio is defined as

$$\pi_{p,s} = \frac{t_{p,s}}{\min_{l} t_{p,l}}.$$

For each solver, we plot the distribution function of a performance metric

$$\rho_s(\tau) = \frac{1}{n_p} \text{card}\{p : \pi_{p.s} \leq \tau\},$$

where $n_p$ is the total number of test problems. For given $\tau > 1$, the function $\rho_s(\tau)$ returns the portion of problems that solver $s$ could solve within a factor $\tau$ of the best result.

We shall refer to the trust-region algorithm based on the shape-changing norm (31) as EIG($\infty$, 2). We used it as a reference algorithm for comparing with other algorithms, because EIG($\infty$, 2) was one of the most successful. We studied its performance for the parameter values $m = 5$, 10 and 15, which means storing at most 5, 10 and 15 pairs $\{s_k, y_k\}$, respectively. We performed numerical experiments for various values of the threshold parameter $\nu$ introduced in Section 7.4 for establishing rank-deficiency of $V$. Since the best results were obtained for $\nu = 10^{-7}$, we used this value in our algorithms. In some test problems, we observed that computing $V_k^T s_k$ according to (56) could lead to numerical errors in $V_k^T V_k$. To easily identify such cases we computed a relative error in $s_{k-1}^T s_k$, and if

the error was larger than $10^{-4}$, a restart was applied meaning to store only the latest pair $\{s_k, y_k\}$. This test was implemented in all our eigenvalue-based trust-region algorithms. An alternative could be to recompute $V_k^T V_k$ but it is computationally more expensive, and in our experiments, it did not sufficiently decrease the number of iterations.
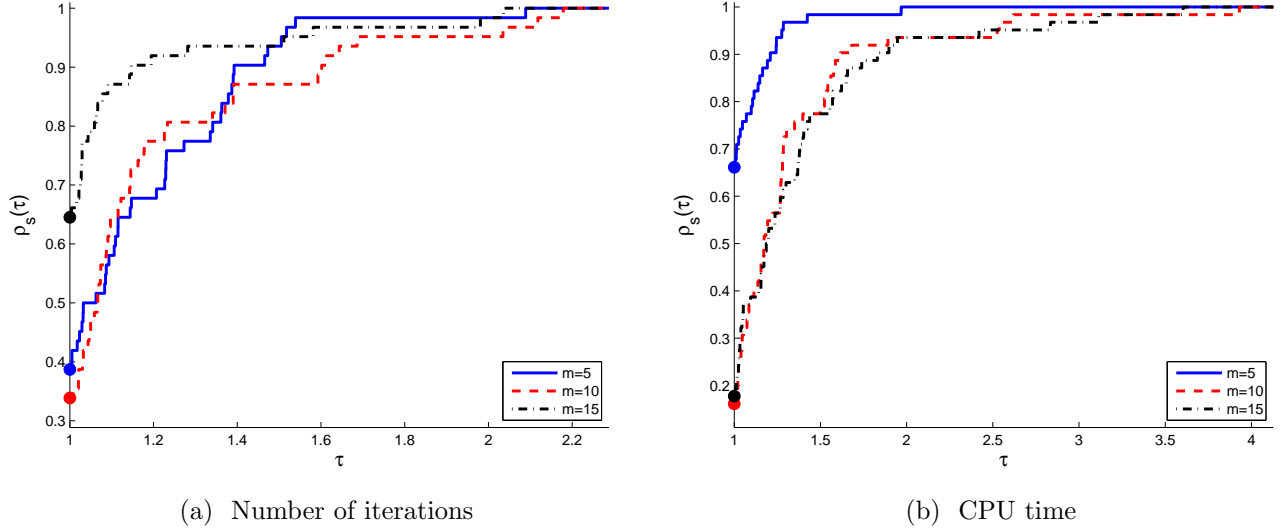


(a) Number of iterations          (b) CPU time

Figure 1: Performance profile for $\mathrm{EIG}(\infty, 2)$ for $m = 5, 10, 15$.

In Figure 1a one can see that the memory size $m = 15$ corresponds to the fewest number of iterations. This is a typical behavior of limited memory algorithms, because larger memory allows for using more complete information about the Hessian matrix, carried by the pairs $\{s_k, y_k\}$, which tends to decrease the number of iterations. On the other hand, each iteration becomes computationally more expensive. For each test problem there exists its own best value of $m$ that reflects a tradeoff. Figure 1b shows that the fastest for the most of the test problems was the case of $m = 5$. A similar behavior was demonstrated by the trust-region algorithms based on the Euclidean norm and shape-changing norm (32). This motivates the use of $m = 5$ in our numerical experiments.

We implemented in MATLAB three versions of the line-search L-BFGS. Two of them use the Moré-Thuente line search [32] implemented in MATLAB by Dianne O'Leary [35] with the same line-search parameter values as in [30]. The difference is in computing the search direction, which is based either on the two-loop recursion [30, 33] or on the compact representation of the inverse Hessian approximation [11] presented by formula (58). These two line-search versions have the same theoretical properties, namely, they generate identical iterates and have the same computational complexity, $2mn$. Nevertheless, owing to the efficient matrix operations in MATLAB, the former version was faster.

In the third version, the search direction is computed with the use of the compact representation. We adapted here Algorithm 1 to make a fairer comparison with our trust-region algorithms under the same choice of parameter values. The trial step in this version is obtained by minimizing the same model function along the quasi-Newton direction bounded by the trust region. Like in our trust-region algorithms, it is accepted whenever (63) holds, which makes the line search non-monotone. This version of L-BFGS was superior

to the other two line-search versions in every respect. The success can be explained as follows. In comparison to the Wolve conditions, it required less number of function and gradient evaluations for satisfying the acceptance conditions of Algorithm 1. Furthermore, the aforementioned possible non-monotonicity due to (63) made the third version more robust. We shall refer to it as L-BFGS. Only this version is involved in our comparative study of the implemented algorithms.
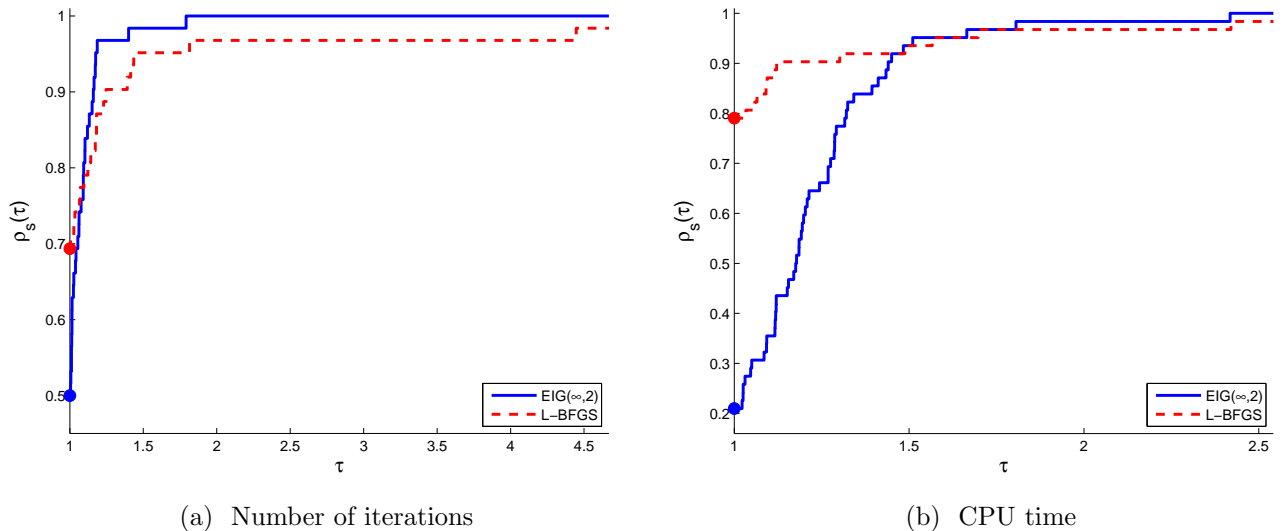


(a) Number of iterations

(b) CPU time

Figure 2: Performance profile for $EIG(\infty, 2)$ and L-BFGS.

As one can see in Figure 2a, algorithm $EIG(\infty, 2)$ demonstrated a good performance in terms of the number of iterations. In contrast to L-BFGS, it was able to solve all the test problems, which says in favor of its robustness and better numerical stability. The performance profiles for the number of gradient evaluations were almost identical to those for the number of iterations. Note that, owing to the automatic differentiation, the gradients are efficiently calculated in the CUTEr. This means that if the calculation of gradients were much more time consuming, the performance profiles for the CPU time would more closely resemble those in Figure 2a.

We observed that, for each CUTEr test problem, the step-size one was used for, at least, 60% of iterations of L-BFGS. To demonstrate the advantage of the trust-region framework over the line search, we selected all those test problems (10 in total), where the step-size one was rejected by L-BFGS in, at least, 30% of iterations. At each of these iterations, the line-search procedure computed function values more than once. The corresponding performance profiles are given in Figure 3 (the only figure where the profiles are presented for the reduced set of problems). Algorithm L-BFGS failed on one of these problems, and it was obviously less effective than $EIG(\infty, 2)$ on the rest of them, both in terms of the number of iterations and the CPU time.

Our numerical experiments indicate that algorithm $EIG(\infty, 2)$ is, at least, competing with L-BFGS. It is natural to expect that $EIG(\infty, 2)$ will be dominating in those of the problems originating from simulation-based applications and industry, where the cost of function and gradient evaluations is much more expensive than computing a trial step.
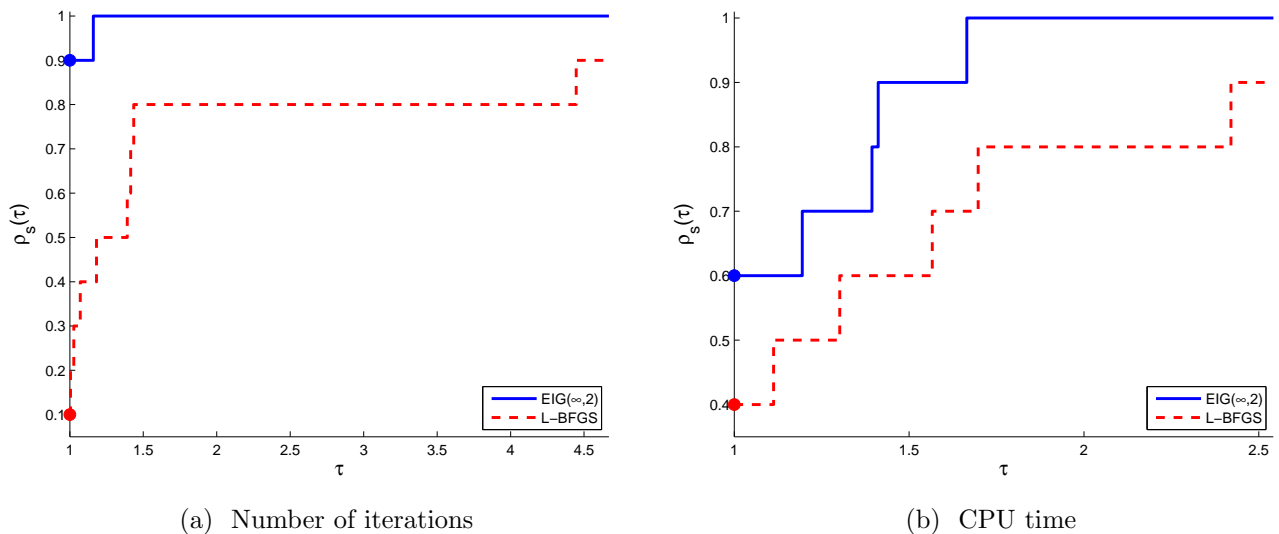
(a) Number of iterations                    (b) CPU time

Figure 3: Performance profile for EIG$(\infty, 2)$ and L-BFGS on problems with the step-size one rejected in, at least, 30% of iterations.

We compared EIG$(\infty, 2)$ also with some other eigenvalue-based limited memory trust-region algorithms. In one of them, the trust region is defined by the Euclidean norm, and the other algorithm uses the shape-changing norm (32). We refer to them as EIG-MS and EIG-MS(2,2), respectively. In EIG-MS, the trust-region subproblem is solved by the Moré-Sorenson approach. We used the same approach in EIG-MS(2,2) for solving the first subproblem in (41) defined by the Euclidean norm in a lower-dimensional space. Notice that since BFGS updates generate positive definite Hessian approximation, the hard case is impossible. In all our experiments, the tolerance of solving (13) was defined by the inequality

$$\left| \|s\| - \Delta \right| \leq \Delta \cdot 10^{-1},$$

which almost always required to perform from one to three Newton iterations (19). We observed also that the higher accuracy was usually increasing the total computational time without any noticeable improvement in the number of iterations.

Figure 4 shows that EIG$(\infty, 2)$ and EIG-MS(2,2) were able to solve all the test problems, whereas EIG-MS(2,2) failed on one of them. Algorithm EIG$(\infty, 2)$ often required the same or even fewer number of iterations than the other two algorithms. The behavior of EIG-MS(2,2) was very similar to EIG-MS, which can be explained as follows.

In our numerical experiments with L-BFGS updates, we observed that

$$\|g_\perp\| \ll \|g_\parallel\| \approx \|g\|. \tag{64}$$

Our intuition about this property is presented in the next paragraph. For $s$ that solves the trust region subproblem, (64) results in $\|P_\perp^T s\| \ll \|P_\parallel^T s\|$, i.e., the component of $s$ that belongs to the subspace defined by $P_\perp$ is often vanishing, and therefore, the shape-changing norm (32) of $s$ is approximately the same as its Euclidean norm. This is expected to result, for the double-dogleg approach, in approximately the same number of iterations
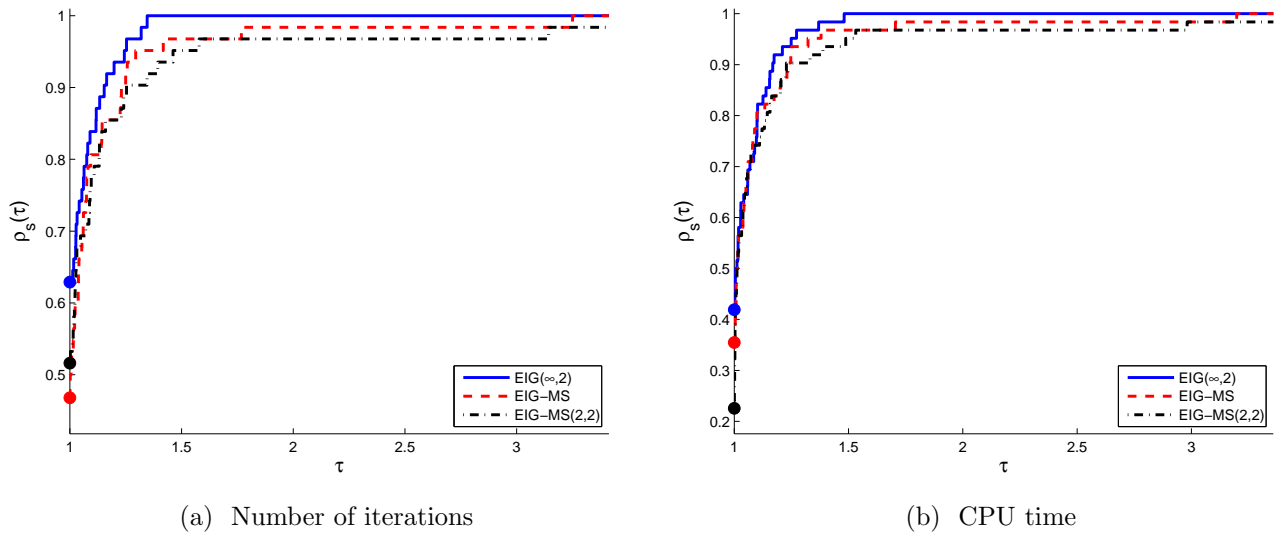
(a)  Number of iterations            (b)  CPU time

Figure 4: Performance profile for EIG($\infty$, 2), EIG-MS and EIG-MS(2,2).

in the case of the two norms, because the Cauchy vectors are approximately the same for these norms. However, it is unclear how to make the computational cost of one iteration for the norm (32) lower than for the Euclidean norm in $R^n$. This is the reason why our combination of the double-dogleg approach with the norm (32) was nor successful.

One of the possible explanations why (64) is typical for L-BFGS originates from its relationship with CG. Recall that, in the case of quadratic $f(x)$, the first $m$ iterates generated by L-BFGS with exact line search are identical to those generated by CG. Furthermore, CG has the property that $g_k$ belongs to the subspace spanned by the columns of $S_k$ and $Y_k$, i.e., $g_\perp = 0$. The numerical experiments show that L-BFGS inherits this property in an approximate form when $f(x)$ is not quadratic.

We implemented also our own version of the limited memory trust-region algorithm by Burke et al. [10]. This version was presented in Section 8.1, and it will be referred to as BWX-MS. It has much better performance than its original version. We compare it with EIG($\infty$, 2). Note that BWX-MS requires two Cholesky factorizations of $m \times m$ matrices for solving (60) at each Newton's iteration (19) (see [10]). Algorithm EIG($\infty$, 2) requires one Cholesky factorization of a $(2m) \times (2m)$ matrix and one eigenvalue decomposition for a matrix of the same size, but in contrast to BWX-MS, this is to be done only once when $x_{k+1} \neq x_k$, and this is not required to be done for a decreased trust-region radius when the trial point is rejected. This explains why the performance profile demonstrated in Figure 5 is obviously better for our eigenvalue-based approach than for our improved version of the one proposed in [10]. In our numerical experiments, the advantage in performance was getting more significant when a higher accuracy of solving the trust region subproblem was set. Algorithm EIG($\infty$, 2) is particularly more efficient than BWX-MS in problems where the trial step is often rejected.

It should be mentioned here an alternative approach developed by Erway and Marcia [18, 19, 20] for solving the trust-region subproblem for the L-BFGS updates. The available version of their implementation [20], called MSS, was far less efficient in our numerical
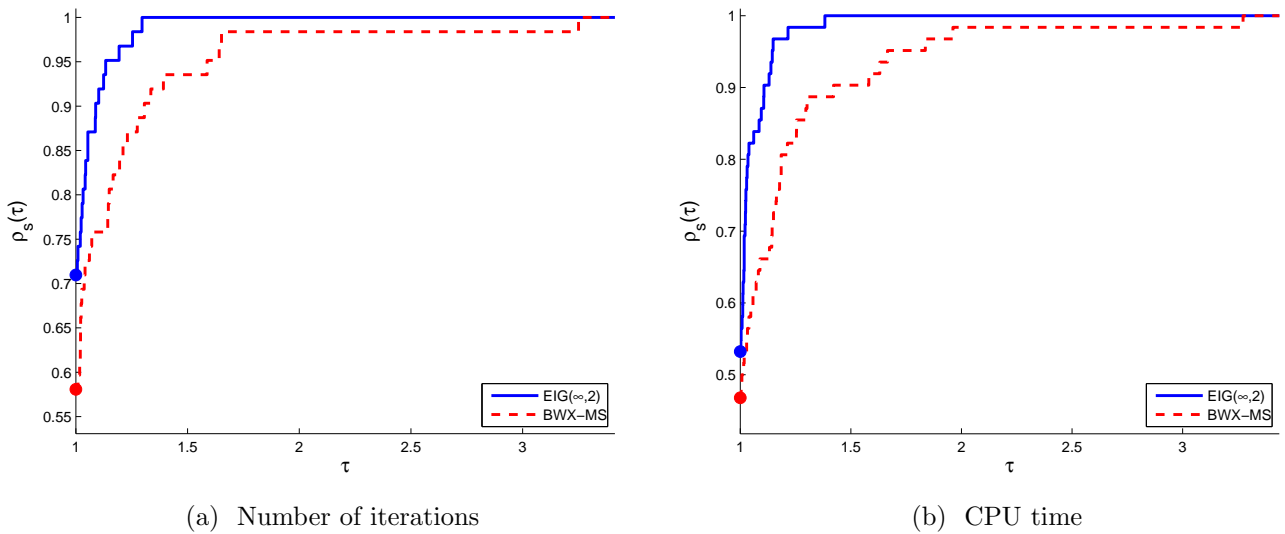
30

(a) Number of iterations                     (b) CPU time

Figure 5: Performance profile for EIG$(\infty, 2)$ and BWX-MS.

experiments than EIG$(\infty, 2)$.

We implemented the original version of the double-dogleg algorithm proposed in [29] and outlined in Section 8.2. This inexact limited-memory trust-region algorithm failed on 12 of the CUTEr problems, and it was far less efficient than EIG$(\infty, 2)$ on the rest of the problems.

We implemented also our own version of the double-dogleg approach presented in Section 8.2. We refer to it as D-DOGL. The performance profiles for D-DOGL and EIG$(\infty, 2)$ are presented in Figure 6. It shows that D-DOGL generally required more iterations than EIG$(\infty, 2)$, because the trust-region subproblem was solved to a low accuracy. However, it was often faster unless it took significantly more iterations to converge. This algorithm does not require eigenvalue decomposition of $B$ and when the trial step is rejected, computes the new one only at a cost of $O(n)$ operations. We should note that, as it was observed earlier, e.g., in [17], the CUTEr collection of large-scale test problems is better suited for applying inexact trust-region algorithms like D-DOGL. But such algorithms are not well suited for problems where a higher accuracy of solving trust-region subproblem is required for a better total computational time and robustness.

An alternative approach to approximately solving trust-region subproblem is related to the truncated conjugate gradients [38]. We applied it to solving the first subproblem in (41). Since this CG-based algorithm produces approximate solutions, the number of external iterations was, in general, larger than in the case of the algorithms producing exact or nearly-exact solutions to the trust-region subproblem. The cost of one iteration was not low enough to compete in computational time with the fastest implementations considered in this section.
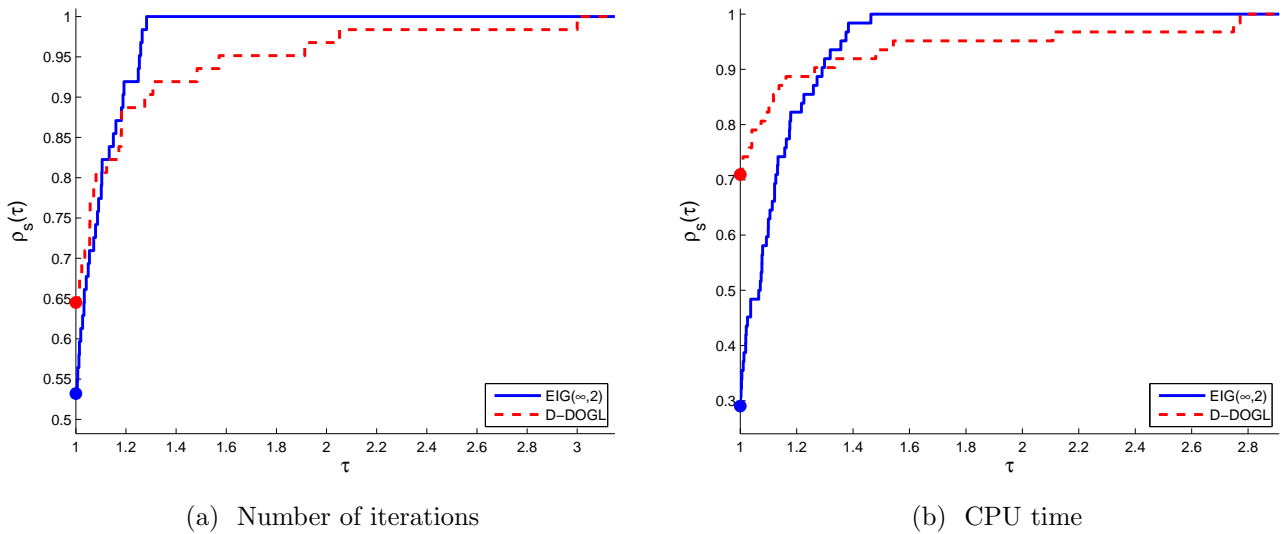
<table>
<tr><td>(a) Number of iterations</td><td>(b) CPU time</td></tr>
</table>

Figure 6: Performance profile for EIG($\infty$, 2) and D-DOGL.

# 10 Conclusions

We have developed efficient combinations of limited memory and trust-region techniques. The numerical experiments indicate that our limited memory trust-region algorithms are competitive with the line-search versions of the L-BFGS method. Our eigenvalue-based approach, originally presented in [9] and further developed in the earlier version of this paper [7], has already been successfully used in [2, 21, 22].

The future aim is to extend our approaches to limited memory SR1 and multipoint symmetric secant approximations. In case of indefinite matrix, we are going to exploit the useful information about negative curvature directions along which the objective function is expected to decrease most rapidly.

Furthermore, the proposed here computationally efficient techniques, including the implicit eigenvalue decomposition, could be considered for improving the performance of limited memory algorithms used, e.g., for solving constrained and bound constrained optimization problems.

# Acknowledgements

# References

[1] Åke Björk: Numerical methods for least squares problems. SIAM, Philadelphia (1996)

[2] Brust, J., Erway, J.B., Marcia, R.F.: On solving L-SR1 trust-region subproblems. http://arxiv.org/abs/1506.07222 (2015)

[3] Buckley, A., LeNir, A.: QN-like variable storage conjugate gradients. Math. Program. **27**(2), 155–175 (1983)

[4] Burdakov, O.P.: Methods of the secant type for systems of equations with symmetric Jacobian matrix. Numer. Funct. Anal. Optim. **6**, 183–195 (1983)

[5] Burdakov, O.P.: Stable versions of the secant method for solving systems of equations. U.S.S.R. Comput. Math. Math. Phys. **23**(5), 1–10 (1983)

[6] Burdakov, O.P.: On superlinear convergence of some stable variants of the secant method. Z. Angew. Math. Mech. **66**(2), 615–622 (1986)

[7] Burdakov, O., Gong, L., Zikrin, S., Yuan, Y.: On efficiently combining limited memory and trust-region techniques. Tech. rep. 2013:13, Linköping University (2013)

[8] Burdakov, O.P., Martínez, J.M., Pilotta, E.A.: A limited-memory multipoint symmetric secant method for bound constrained optimization. Ann. Oper. Res. **117**, 51–70 (2002)

[9] Burdakov, O., Yuan, Y: On limited-memory methods with shape changing trust region. In: Proceedings of the First International Conference on Optimization Methods and Software, Huangzhou, China, p.21 (2002)

[10] Burke, J.V., Wiegmann, A., Xu, L.: Limited memory BFGS updating in a trust-region framework. Tech. rep., University of Washington (2008)

[11] Byrd, R.H., Nocedal, J., Schnabel, R.B.: Representations of quasi-Newton matrices and their use in limited memory methods. Math. Program. **63**, 129–156 (1994)

[12] Byrd, R.H., Schnabel, R.B., Shultz, G.A.: Approximate solution of the trust region problem by minimization over two-dimensional subspaces. Math. Program. **40**(1-3), 247–263 (1988)

[13] Conn, A.R., Gould, N.I.M., Toint, P.L.: Convergence of quasi-Newton matrices generated by the symmetric rank one update. Math. Program. **50**(1-3), 177–195 (1991)

[14] Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-region methods. MPS/SIAM Ser. Optim. 1, SIAM, Philadelphia (2000)

[15] Dennis Jr., J.E., Mei, H.H.W.: Two new unconstrained optimization algorithms which use function and gradient values. J. Optim. Theory Appl. **28**(4), 453–482 (1979)

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

[17] Erway, J.B., Gill, P.E., Griffin, J.: Iterative methods for finding a trust-region step. SIAM J. Optim. **20**(2), 1110–1131 (2009)

[18] Erway, J.B., Jain, V., Marcia, R.F.: Shifted L-BFGS systems. Optim. Methods Softw. **29**(5), 992–1004 (2014)

[19] Erway, J.B., Marcia, R.F.: Limited-memory BFGS systems with diagonal updates. Linear Algebr. Appl. **437**(1), 333–344 (2012)

[20] Erway, J.B., Marcia, R.F.: Algorithm 943: MSS: MATLAB Software for L-BFGS Trust-Region Subproblems for Large-Scale Optimization. ACM Trans. Math. Softw. **40**(4), 28:1–28:12 (2014)

[21] Erway, J.B., Marcia, R.F.: On efficiently computing the eigenvalues of limited-memory quasi-Newton matrices. SIAM J. Matrix Anal. Appl. **36**(3), 1338–1359 (2015)

[22] Erway, J.B., Marcia, R.F.: On solving limited-memory quasi-Newton equations. http://arxiv.org/abs/1510.06378 (2015)

[23] Gilbert, J.C., Lemaréchal, C.: Some numerical experiments with variable-storage quasi-Newton algorithms. Math. Program. **45**(1-3), 407–435 (1989)

[24] Gill, P.E., Leonard, M.W.: Limited-memory reduced-Hessian methods for large-scale unconstrained optimization. SIAM J. Optim. **14**, 380–401 (2003)

[25] Golub, G., Van Loan, C.: Matrix Computations, 4th edn. Johns Hopkins University Press (2013)

[26] Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Solving the trust-region subproblem using the Lanczos method. SIAM J. Optim. **9**(2), 504–525 (1999)

[27] Gould, N.I.M., Orban, D., Toint, P.L.: CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. ACM Trans. Math. Software **29**(4), 373–394 (2003)

[28] Hager, W., Zhang, H.: A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J. Optim. **16**(1), 170–192 (2005)

[29] Kaufman, L.: Reduced storage, quasi-Newton trust region approaches to function optimization. SIAM J. Optim. **10**(1), 56–69 (1999)

[30] Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**(1-3), 503–528 (1989)

[31] Moré, J.J., Sorensen, D.: Computing a trust region step. SIAM J. Sci. Stat. Comput. **4**(3), 553–572 (1983)

[32] Moré, J.J., Thuente, D.J.: Line search algorithms with guaranteed sufficient decrease. ACM Trans. Math. Softw. **20**(3), 286–307 (1994)

[33] Nocedal, J.: Updating quasi-Newton matrices with limited storage. Math. Comp. **35**(151), 773–782 (1980)

[34] Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer Ser. Oper. Res. Springer, New York. (2006)

[35] O'Leary, D.: A Matlab implementation of a MINPACK line search algorithm by Jorge J. Moré and David J. Thuente (1991). https://www.cs.umd.edu/users/oleary/software/. Accessed 1 November 2012

[36] Powell, M.J.D.: A hybrid method for nonlinear equations. In: P. Rabinowitz (ed.) Numerical Methods for Nonlinear Algebraic Equations, pp. 87–114. Gordon and Breach, London (1970)

[37] Powell, M.J.D.: A new algorithm for unconstrained optimization. In: O.L.M. J. B. Rosen, K. Ritter (eds.) Nonlinear Programming. Academic Press, New York (1970)

[38] Steihaug, T.: The conjugate gradient method and trust regions in large scale optimization. SIAM J. Numer. Anal. **20**(3), 626–637 (1983)

[39] Sun, W., Yuan, Y.: Optimization Theory and Methods. Nonlinear Programming, Springer Optimization and Its Applications 1, Springer, New York (2006)

[40] Toint, P.L.: Towards an efficient sparsity exploiting Newton method for minimization. In: I.S. Duff (ed.) Sparse Matrices and Their Uses, pp. 57–88. Academic Press, London (1981)

[41] Wolfe, P.: Convergence conditions for ascent methods. SIAM Rev. **11**(2), 226–235 (1969)

[42] Yuan, Y., Stoer, J.: A subspace study on conjugate gradient algorithms. ZAMM Z. Angew. Math. Mech. **75**(1), 69–77 (1995)

[43] Yuan, Y.: Recent advances in trust region algorithms. Math. Program. **151**(1), 249–281 (2015)