Computational Optimization and Applications
https://doi.org/10.1007/s10589-019-00112-x

A dense initialization for limited-memory quasi-Newton methods

Johannes Brust[1] · Oleg Burdakov[2] · Jennifer B. Erway[3] · Roummel F. Marcia[4]

Received: 24 May 2018

Abstract
We consider a family of dense initializations for limited-memory quasi-Newton methods. The proposed initialization exploits an eigendecomposition-based separation of the full space into two complementary subspaces, assigning a different initialization parameter to each subspace. This family of dense initializations is proposed in the context of a limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) trust-region method that makes use of a shape-changing norm to define each subproblem. As with L-BFGS methods that traditionally use diagonal initialization, the dense initialization and the sequence of generated quasi-Newton matrices are never explicitly formed. Numerical experiments on the CUTEst test set suggest that this initialization together with the shape-changing trust-region method outperforms other L-BFGS methods for solving general nonconvex unconstrained optimization problems. While this dense initialization is proposed in the context of a special trust-region method, it has broad applications for more general quasi-Newton trust-region and line search methods. In fact, this initialization is suitable for use with any quasi-Newton update that admits a compact representation and, in particular, any member of the Broyden class of updates.

Keywords Large-scale nonlinear optimization · Limited-memory quasi-Newton methods · Trust-region methods · Quasi-Newton matrices · Shape-changing norm

Mathematics Subject Classification 90C53 · 90C06 · 90C26 · 65K05 · 65K10 · 65F10 · 65F15

This research is supported by NSF Grants CMMI-1334042, CMMI-1333326, IIS-1741490, and IIS-1741264.

✉ Jennifer B. Erway
erwayjb@wfu.edu

Extended author information available on the last page of the article

Published online: 29 May 2019

1 Introduction

In this paper we propose a new dense initialization for quasi-Newton methods to solve problems of the form

$$\underset{x\in\Re^n}{\text{minimize}}\ f(x),$$

where $f : \Re^n \to \Re$ is at least a continuously differentiable function, which is not necessarily convex. The dense initialization matrix is designed to be updated each time a new quasi-Newton pair is computed (i.e., as often as once an iteration); however, in order to retain the efficiency of limited-memory quasi-Newton methods, the dense initialization matrix and the generated sequence of quasi-Newton matrices are not explicitly formed. This proposed initialization makes use of a partial eigendecomposition of these matrices for separating $\Re^n$ into two orthogonal subspaces – one for which there is approximate curvature information and the other for which there is no reliable curvature information. This initialization has broad applications for general quasi-Newton trust-region and line search methods. In fact, this work can be applied to any quasi-Newton method that uses an update with a compact representation, which includes any member of the Broyden class of updates. For this paper, we explore its use in one specific algorithm; in particular we consider a limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) trust-region method where each subproblem is defined using a shape-changing norm [3]. The reason for this choice is that the dense initialization is naturally well-suited for solving L-BFGS trust-region subproblems defined by this norm. Numerical results on the CUTEst test set suggest that the dense initialization outperforms other L-BFGS methods.

The BFGS update is the most widely-used quasi-Newton update for large-scale optimization; it is defined by the recursion formula

$$B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T B_k + \frac{1}{s_k^T y_k} y_k y_k^T, \tag{1}$$

where

$$s_k \triangleq x_{k+1} - x_k \quad \text{and} \quad y_k \triangleq \nabla f(x_{k+1}) - \nabla f(x_k), \tag{2}$$

and $B_0 \in \Re^{n\times n}$ is a suitably-chosen initial matrix. This rank-two update to B_k preserves positive definiteness when $s_k^T y_k > 0$.

L-BFGS is a limited-memory variant of BFGS that only stores a predetermined number, m, of the most recently-computed pairs $\{s_i, y_i\}$ where $m \ll n$. (Typically, $m \in [3, 7]$ (see, e.g., [5]).) Together with an initial matrix B_0 that depends on k, these pairs are used to compute B_k. For notational simplicity, we drop the dependence of the initial matrix on k and simply denote it as B_0. This limitation on the number of stored pairs allows for a practical implementation of the BFGS method for large-scale optimization.

There are several desirable properties for picking the initial matrix B_0. First, in order for the sequence $\{B_k\}$ generated by (1) to be symmetric and positive definite, it

is necessary that B_0 is symmetric and positive definite. Second, it is desirable for B_0 to be easily invertible so that solving linear systems with any matrix in the sequence is computable using the so-called "two-loop recursion" [5] or other recursive formulas for B_k^{-1} (for an overview of other available methods see [9]). For these reasons, B_0 is often chosen to be a scalar multiple of the identity matrix, i.e.,

$$B_0 = \gamma_k I, \quad \text{with} \quad \gamma_k > 0. \tag{3}$$

For BFGS matrices, the conventional choice for the initialization parameter γ_k is

$$\gamma_k = \frac{y_k^T y_k}{s_k^T y_k}, \tag{4}$$

which can be viewed as a spectral estimate for $\nabla^2 f(x_k)$ [13]. (This choice was originally proposed in [14] using a derivation based on optimal conditioning.) It is worth noting that this choice of γ_k can also be derived as the minimizer of the scalar minimization problem

$$\gamma_k = \underset{\gamma}{\operatorname{argmin}} \left\| B_0^{-1} y_k - s_k \right\|_2^2, \tag{5}$$

where $B_0^{-1} = \gamma^{-1} I$. For numerical studies on this choice of initialization, see, e.g., the references listed within [4].

In this paper, we consider a specific dense initialization in lieu of the usual diagonal initialization. The aforementioned separation of $\Re^n$ into two orthogonal subspaces allows for different initialization parameters to be used to estimate the curvature of the underlying function in these subspaces. In one space (the space spanned by the most recent updates $\{s_i, y_i\}$ with $k - m \leq i \leq k - 1$), estimates of the curvature of the underlying function are available, and thus, one initialization parameter can be set using this information. However, in its orthogonal complement, curvature information is not available. Therefore, if the component of the trial step in the orthogonal subspace is (relatively) too large, the predictive quality of the whole trial step is expected to deteriorate. As a result, the trust-region radius might be reduced, despite the fact that the predictive quality of the component in the aforementioned small subspace may be sufficiently good. Separating the whole space into these two subspaces allows users to treat each subspace differently. An alternative view of this initialization is that it makes use of *two* spectral estimates of $\nabla^2 f(x_k)$. Finally, the proposed initialization also allows for efficiently solving and computing products with the resulting quasi-Newton matrices.

The paper is organized in five sections. In Sect. 2, we review properties of L-BFGS matrices arising from their special recursive structure as well as overview the shape-changing trust-region method to be used in this paper. In Sect. 3, we present the proposed trust-region method that uses a shape-changing norm together with a dense initialization matrix. While this dense initialization is presented in one specific context, it can be used in combination with any quasi-Newton update that admits a so-called *compact representation*. Numerical experiments comparing this method with other combinations of initializations and L-BFGS methods are reported in Sect. 4, and concluding remarks are found in Sect. 5.

2 Background

In this section, we overview the compact formulation for L-BFGS matrices and how to efficiently compute a partial eigendecomposition. Finally, we review the shape-changing trust-region method considered in this paper.

2.1 The compact representation

The special structure of the recursion formula for L-BFGS matrices admits a so-called compact representation [5], which is overviewed in this section.

Using the m most recently computed pairs $\{s_j\}$ and $\{y_j\}$ given in (2), we define the following matrices

$$S_k \triangleq \begin{bmatrix} s_{k-m} & \cdots & s_{k-1} \end{bmatrix} \quad \text{and} \quad Y_k \triangleq \begin{bmatrix} y_{k-m} & \cdots & y_{k-1} \end{bmatrix}.$$

With L_k taken to be the strictly lower triangular part of the matrix of $S_k^T Y_k$, and D_k defined as the diagonal of $S_k^T Y_k$, the compact representation of an L-BFGS matrix is

$$B_k = B_0 + \Psi_k M_k \Psi_k^T, \tag{6}$$

where

$$\Psi_k \triangleq [B_0 S_k \;\; Y_k] \quad \text{and} \quad M_k \triangleq -\begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1} \tag{7}$$

(see [5] for details). Note that $\Psi_k \in \Re^{n\times 2m}$, and $M_k \in \Re^{2m\times 2m}$ is invertible provided $s_i^T y_i > 0$ for all i [5, Theorem 2.3]. An advantage of the compact representation is that if B_0 is chosen to be a multiple of the identity, then computing products with B_k or solving linear systems with B_k can be done efficiently [9,12].

It should be noted that L-BFGS matrices are just one member of the Broyden class of matrices (see, e.g., [13]), and in fact every member of the Broyden class of matrices admits a compact representation [6,8,12].

2.2 Partial eigendecomposition of B_k

If B_0 is taken to be a multiple of the identity matrix, then the partial eigendecomposition of B_k can be computed efficiently from the compact representation (6) using either a partial QR decomposition [3] or a partial singular value decomposition (SVD) [11]. Below, we review the approach that uses the QR decomposition, and we assume that Ψ_k has rank $r = 2m$. (For the rank-deficient case, see the techniques found in [3].)

Let

$$\Psi_k = QR,$$

be the so-called "thin" QR factorization of Ψ_k, where $Q \in \Re^{n\times r}$ and $R \in \Re^{r\times r}$. Since the matrix RM_kR^T is a small $(r \times r)$ matrix with $r \ll n$ (recall that $r = 2m$, where m is typically between 3 and 7), it is computationally feasible to calculate its eigendecomposition; thus, suppose $W\hat{\Lambda}W^T$ is the eigendecomposition of RM_kR^T. Then,

$$\Psi_k M_k \Psi_k^T = QRM_kR^TQ^T = QW\hat{\Lambda}W^TQ^T = \Psi_k R^{-1}W\hat{\Lambda}W^TR^{-T}\Psi_k^T.$$

Defining

$$P_{\|} = \Psi_k R^{-1} W, \tag{8}$$

gives that

$$\Psi_k M_k \Psi_k^T = P_{\|}\hat{\Lambda}P_{\|}^T. \tag{9}$$

Thus, for $B_0 = \gamma_k I$, the eigendecomposition of B_k can be written as

$$B_k = \gamma_k I + \Psi_k M_k \Psi_k^T = P\Lambda P^T, \tag{10}$$

where

$$P \triangleq \begin{bmatrix} P_{\|} & P_{\perp} \end{bmatrix}, \quad \Lambda \triangleq \begin{bmatrix} \hat{\Lambda} + \gamma_k I_r & \\ & \gamma_k I_{n-r} \end{bmatrix}, \tag{11}$$

and $P_{\perp} \in \mathbb{R}^{n\times(n-r)}$ is defined as the orthogonal complement of $P_{\|}$, i.e., $P_{\perp}^T P_{\perp} = I_{n-r}$ and $P_{\|}^T P_{\perp} = 0_{r\times(n-r)}$. Hence, B_k has r eigenvalues given by the diagonal elements of $\hat{\Lambda} + \gamma_k I_r$ and the remaining eigenvalues are γ_k with multiplicity $n - r$.

2.2.1 Practical computations

Using the above method yields the eigenvalues of B_k as well as the ability to compute products with $P_{\|}$. Formula (8) indicates that Q is not required to be explicitly formed in order to compute products with $P_{\|}$. For this reason, it is desirable to avoid forming Q by computing only R via the Cholesky factorization of $\Psi_k^T\Psi_k$, i.e., $\Psi_k^T\Psi_k = R^TR$ (see [3]).

At an additional expense, the eigenvectors stored in the columns of $P_{\|}$ may be formed and stored. For the shape-changing trust-region method used in this paper, it is not required to store $P_{\|}$. In contrast, the matrix $P_{\perp}$ is prohibitively expensive to form. It turns out that for this work it is only necessary to be able to compute projections into the subspace $P_{\perp}P_{\perp}^T$, which can be done using the identity

$$P_{\perp}P_{\perp}^T = I - P_{\|}P_{\|}^T. \tag{12}$$

2.3 A shape-changing L-BFGS trust-region method

Generally speaking, at the kth step of a trust-region method, a search direction is computed by approximately solving the trust-region subproblem

$$p^* = \underset{\|p\| \le \Delta_k}{\operatorname{argmin}} \; Q(p) \triangleq g_k^T p + \frac{1}{2} p^T B_k p, \tag{13}$$

where $g_k \triangleq \nabla f(x_k)$, $B_k \approx \nabla^2 f(x_k)$, and $\Delta_k > 0$ is the trust-region radius. When second derivatives are unavailable or computationally too expensive to compute, approximations using gradient information may be preferred. Not only do quasi-Newton matrices use only gradient and function information, but in the large-scale case, these Hessian approximations are never stored; instead, a recursive formula or methods that avoid explicitly forming B_k may be used to compute matrix-vector products with the approximate Hessians or their inverses [5,8,9,12].

Consider the trust-region subproblem defined by the shape-changing infinity norm:

$$\underset{\|p\|_{P,\infty} \le \Delta_k}{\text{minimize}} \; Q(p) = g_k^T p + \frac{1}{2} p^T B_k p, \tag{14}$$

where

$$\|p\|_{P,\infty} \triangleq \max\left(\|P_\parallel^T p\|_\infty, \|P_\perp^T p\|_2 \right) \tag{15}$$

and $P_\parallel$ and $P_\perp$ are given in (11). Note that the ratio $\|p\|_2 / \|p\|_{P,\infty}$ does not depend on n and only moderately depends on r. (In particular, $1 \le \|p\|_2 / \|p\|_{P,\infty} \le \sqrt{r+1}$.) Because this norm depends on the eigenvectors of B_k, the shape of the trust region changes each time the quasi-Newton matrix is updated, which is possibly every iteration of a trust-region method. (See [3] for more details and other properties of this norm.) The motivation for this choice of norm is that the the trust-region subproblem (14) decouples into two separate problems for which closed-form solutions exist.

We now review the closed-form solution to (14), as detailed in [3]. Let

$$v = P^T p = \begin{bmatrix} P_\parallel^T p \\ P_\perp^T p \end{bmatrix} \triangleq \begin{bmatrix} v_\parallel \\ v_\perp \end{bmatrix} \quad \text{and} \quad P^T g_k = \begin{bmatrix} P_\parallel^T g_k \\ P_\perp^T g_k \end{bmatrix} \triangleq \begin{bmatrix} g_\parallel \\ g_\perp \end{bmatrix}. \tag{16}$$

With this change of variables, the objective function of (14) becomes

$$\begin{aligned} Q(Pv) &= g_k^T P v + \frac{1}{2} v^T \left(\hat{\Lambda} + \gamma_k I_n \right) v \\ &= g_\parallel^T v_\parallel + g_\perp^T v_\perp + \frac{1}{2} \left(v_\parallel^T \left(\hat{\Lambda} + \gamma_k I_r \right) v_\parallel + \gamma_k \|v_\perp\|_2^2 \right) \\ &= g_\parallel^T v_\parallel + \frac{1}{2} v_\parallel^T \left(\hat{\Lambda} + \gamma_k I_r \right) v_\parallel + g_\perp^T v_\perp + \frac{1}{2} \gamma_k \|v_\perp\|_2^2 . \end{aligned}$$

The trust-region constraint $\|p\|_{P,\infty} \leq \Delta_k$ implies $\|v_{\|}\|_{\infty} \leq \Delta_k$ and $\|v_{\perp}\|_2 \leq \Delta_k$, which decouples (14) into the following two trust-region subproblems:

$$\underset{\|v_{\|}\|_{\infty} \leq \Delta_k}{\text{minimize}}\; q_{\|}\left(v_{\|}\right) \triangleq g_{\|}^T v_{\|} + \frac{1}{2} v_{\|}^T \left(\hat{\Lambda} + \gamma_k I_r\right) v_{\|} \tag{17}$$

$$\underset{\|v_{\perp}\|_2 \leq \Delta_k}{\text{minimize}}\; q_{\perp}\left(v_{\perp}\right) \triangleq g_{\perp}^T v_{\perp} + \frac{1}{2}\gamma_k \|v_{\perp}\|_2^2 . \tag{18}$$

Observe that the resulting minimization problems are considerably simpler than the original problem since in both cases the Hessian of the new objective functions are diagonal matrices. The solutions to these decoupled problems have closed-form analytical solutions [2,3]. Specifically, letting $\lambda_i \triangleq \hat{\lambda}_i + \gamma_k$, the solution to (17) is given coordinate-wise by

$$[v_{||}^*]_i = \begin{cases} -\frac{[g_{||}]_i}{\lambda_i} & \text{if } \left|\frac{[g_{||}]_i}{\lambda_i}\right| \leq \Delta_k \text{ and } \lambda_i > 0, \\ c & \text{if } \left[g_{\|}\right]_i = 0 \text{ and } \lambda_i = 0, \\ -\text{sgn}(\left[g_{\|}\right]_i)\Delta_k & \text{if } \left[g_{\|}\right]_i \neq 0 \text{ and } \lambda_i = 0, \\ \pm\Delta_k & \text{if } \left[g_{\|}\right]_i = 0 \text{ and } \lambda_i < 0, \\ -\frac{\Delta_k}{|[g_{||}]_i|}\left[g_{||}\right]_i & \text{otherwise,} \end{cases} \tag{19}$$

where c is any real number in $[-\Delta_k, \Delta_k]$ and 'sgn' denotes the signum function. Meanwhile, the minimizer of (18) is given by

$$v_{\perp}^* = \beta g_{\perp}, \tag{20}$$

where

$$\beta = \begin{cases} -\frac{1}{\gamma_k} & \text{if } \gamma_k > 0 \text{ and } \|g_{\perp}\|_2 \leq \Delta_k |\gamma_k|, \\ -\frac{\Delta_k}{\|g_{\perp}\|_2} & \text{otherwise.} \end{cases} \tag{21}$$

Note that the solution to (14) is then

$$p^* = P v^* = P_{\|} v_{\|}^* + P_{\perp} v_{\perp}^* = P_{\|} v_{\|}^* + \beta P_{\perp} g_{\perp} = P_{\|} v_{\|}^* + \beta P_{\perp} P_{\perp}^T g_k, \tag{22}$$

where the latter term is computed using (12). Additional simplifications yield the following expression for p^*:

$$p^* = \beta g + P_{\|}(v_{\|}^* - \beta g_{\|}). \tag{23}$$

The overall cost of computing the solution to (14) is comparable to that of using the Euclidean norm (see [3]). The main advantage of using the shape-changing norm (15) is that the solution p^* in (23) has a closed-form expression.

3 The proposed method

In this section, we present a new dense initialization and demonstrate how it is naturally well-suited for trust-region methods defined by the shape-changing infinity norm. Finally, we present a full trust-region algorithm that uses the dense initialization, consider its computational cost, and prove global convergence.

3.1 Dense initial matrix $\widehat{B}_0$

In this section, we propose a new dense initialization for quasi-Newton methods. Importantly, in order to retain the efficiency of quasi-Newton methods the dense initialization matrix and subsequently updated quasi-Newton matrices are never explicitly formed. This initialization can be used with any quasi-Newton update for which there is a compact representation; however, for simplicity, we continue to refer to the BFGS update throughout this section. For notational purposes, we use the initial matrix B_0 to represent the usual initialization and $\widehat{B}_0$ to denote the proposed dense initialization. Similarly, $\{B_k\}$ and $\{\widehat{B}_k\}$ will be used to denote the sequences of matrices obtained using the initializations B_0 and $\widehat{B}_0$, respectively.

Our goal in choosing an alternative initialization is four-fold: (i) to be able to treat subspaces differently depending on whether curvature information is available or not, (ii) to preserve properties of symmetry and positive-definiteness, (iii) to be able to efficiently compute products with the resulting quasi-Newton matrices, and (iv) to be able to efficiently solve linear systems involving the resulting quasi-Newton matrices. The initialization proposed in this paper leans upon two different parameter choices that can be viewed as an estimate of the curvature of $\nabla^2 f(x_k)$ in two subspaces: one spanned by the columns of $P_\parallel$ and another spanned by the columns of $P_\perp$.

The usual initialization for a BFGS matrix B_k is $B_0 = \gamma_k I$, where $\gamma_k > 0$. Note that this initialization is equivalent to

$$B_0 = \gamma_k P P^T = \gamma_k P_\parallel P_\parallel^T + \gamma_k P_\perp P_\perp^T.$$

In contrast, for a given $\gamma_k, \gamma_k^\perp \in \Re$, consider the following symmetric, and in general, dense initialization matrix:

$$\widehat{B}_0 = \gamma_k P_\parallel P_\parallel^T + \gamma_k^\perp P_\perp P_\perp^T, \tag{24}$$

where $P_\parallel$ and $P_\perp$ are the matrices of eigenvectors defined in Sect. 2.2. We now derive the eigendecomposition of $\widehat{B}_k$.

Theorem 3.1 *Let $\widehat{B}_0$ be defined as in (24). Then $\widehat{B}_k$ generated using (1) has the eigendecomposition*

$$\widehat{B}_k = \begin{bmatrix} P_\parallel & P_\perp \end{bmatrix} \begin{bmatrix} \hat{\Lambda} + \gamma_k I_r & \\ & \gamma_k^\perp I_{n-r} \end{bmatrix} \begin{bmatrix} P_\parallel & P_\perp \end{bmatrix}^T, \tag{25}$$

where $P_\parallel$, $P_\perp$, and $\hat{\Lambda}$ are given in (8), (11), and (9), respectively.

Proof First note that the columns of S_k are in Range(Ψ_k), where Ψ_k is defined in (7). From (8), Range(Ψ_k) = Range$(P_\parallel)$; thus, $P_\parallel P_\parallel^T S_k = S_k$ and $P_\perp^T S_k = 0$. This gives that

$$\widehat{B}_0 S_k = \gamma_k P_\parallel P_\parallel^T S_k + \gamma_k^\perp P_\perp P_\perp^T S_k = \gamma_k S_k = B_0 S_k. \tag{26}$$

Combining the compact representation of $\widehat{B}_k$ ((6) and (7)) together with (26) yields

$$\begin{aligned}
\widehat{B}_k &= \widehat{B}_0 - \begin{bmatrix} \widehat{B}_0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T \widehat{B}_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T \widehat{B}_0 \\ Y_k^T \end{bmatrix} \\
&= \widehat{B}_0 - \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T B_0 \\ Y_k^T \end{bmatrix} \\
&= \gamma_k P_\parallel P_\parallel^T + \gamma_k^\perp P_\perp P_\perp^T + P_\parallel \hat{\Lambda} P_\parallel^T \\
&= P_\parallel \left(\hat{\Lambda} + \gamma_k I_r \right) P_\parallel^T + \gamma_k^\perp P_\perp P_\perp^T,
\end{aligned}$$

which is equivalent to (25). □

It can be easily verified that (25) holds also for $P_\parallel$ defined in [3] for possibly rank-deficient Ψ_k. (Note that (8) applies only to the special case when Ψ_k is full-rank.)

Theorem 3.1 shows that the matrix $\widehat{B}_k$ that results from using the initialization (24) shares the same eigenvectors as B_k, generated using $B_0 = \gamma_k I$. Moreover, the eigenvalues corresponding to the eigenvectors stored in the columns of $P_\parallel$ are the same for $\widehat{B}_k$ and B_k. The only difference in the eigendecompositions of $\widehat{B}_k$ and B_k is in the eigenvalues corresponding to the eigenvectors stored in the columns of $P_\perp$. This is summarized in the following corollary.

Corollary 3.1 *Suppose B_k is a BFGS matrix initialized with $B_0 = \gamma_k I$ and $\widehat{B}_k$ is a BFGS matrix initialized with (24). Then B_k and $\widehat{B}_k$ have the same eigenvectors; moreover, these matrices have r eigenvalues in common given by $\lambda_i \triangleq \hat{\lambda}_i + \gamma_k$ where $\hat{\Lambda} =$* diag$(\hat{\lambda}_1, \ldots, \hat{\lambda}_r)$.

Proof The corollary follows immediately by comparing (10) with (25). □

The results of Theorem 3.1 and Corollary 3.1 may seem surprising at first since every term in the compact representation ((6) and (7)) depends on the initialization; moreover, $\widehat{B}_0$ is, generally speaking, a dense matrix while B_0 is a diagonal matrix. However, viewed from the perspective of (24), the parameter $\gamma_k^\perp$ only plays a role in scaling the subspace spanned by the columns of $P_\perp$.

The initialization $\widehat{B}_0$ allows for two separate curvature approximations for the BFGS matrix: one in the space spanned by columns of $P_\parallel$ and another in the space spanned by the columns of $P_\perp$. In the next subsection, we show that this initialization is naturally well-suited for solving trust-region subproblems defined by the shape-changing infinity norm.

3.2 The trust-region subproblem

Here we will show that the use of $\widehat{B}_0$ provides the same subproblem separability as B_0 does in the case of the shape-changing infinity norm.

For $\widehat{B}_0$ given by (24), consider the objective function of the trust-region subproblem (14) resulting from the change of variables (16):

$$\begin{aligned} Q(Pv) &= g_k^T Pv + \frac{1}{2} v^T P^T \widehat{B}_k Pv \\ &= g_\parallel^T v_\parallel + \frac{1}{2} v_\parallel^T \left(\hat{\Lambda} + \gamma_k I_r \right) v_\parallel + g_\perp^T v_\perp + \frac{1}{2} \gamma_k^\perp \|v_\perp\|_2^2 . \end{aligned}$$

Thus, (14) decouples into two subproblems: The corresponding subproblem for $q_\parallel(v_\parallel)$ remains (17) and the subproblem for $q_\perp(v_\perp)$ becomes

$$\underset{\|v_\perp\|_2 \leq \Delta_k}{\text{minimize}} \; q_\perp(v_\perp) \triangleq g_\perp^T v_\perp + \frac{1}{2} \gamma_k^\perp \|v_\perp\|_2^2 . \tag{27}$$

The solution to (27) is now given by

$$v_\perp^* = \widehat{\beta} g_\perp, \tag{28}$$

where

$$\widehat{\beta} = \begin{cases} -\frac{1}{\gamma_k^\perp} & \text{if } \gamma_k^\perp > 0 \text{ and } \|g_\perp\|_2 \leq \Delta_k |\gamma_k^\perp|, \\ -\frac{\Delta_k}{\|g_\perp\|_2} & \text{otherwise.} \end{cases} \tag{29}$$

Thus, the solution p^* can be expressed as

$$p^* = \widehat{\beta} g + P_\parallel (v_\parallel^* - \widehat{\beta} g_\parallel), \tag{30}$$

which can computed as efficiently as the solution in (23) for conventional initial matrices since they differ only by the scalar ($\widehat{\beta}$ in (30) versus β in (23)).

3.3 Determining the parameter $\gamma_k^\perp$

The values γ_k and $\gamma_k^\perp$ can be updated at each iteration. Since we have little information about the underlying function f in the subspace spanned by the columns of $P_\perp$, it is reasonable to make conservative (i.e., large) choices for $\gamma_k^\perp$. Note that in the case that $\gamma_k^\perp > 0$ and $\|g_\perp\|_2 \leq \Delta_k |\gamma_k^\perp|$, the parameter $\gamma_k^\perp$ scales the solution $v_\perp^*$ (see 29); thus, large values of $\gamma_k^\perp$ will reduce these step lengths in the space spanned by $P_\perp$. Since the space $P_\perp$ does not explicitly use information produced by past iterations, it seems desirable to choose $\gamma_k^\perp$ to be large. However, the larger that $\gamma_k^\perp$ is chosen, the closer $v_\perp^*$ will be to the zero vector. Also note that if $\gamma_k^\perp < 0$ then the solution to the subproblem (27) will always lie on the boundary, and thus, the actual value of $\gamma_k^\perp$

becomes irrelevant. Moreover, for values $\gamma_k^{\perp} < 0$, $\widehat{B}_k$ is not guaranteed to be positive definite. For these reasons, we suggest sufficiently large and positive values for $\gamma_k^{\perp}$ related to the curvature information gathered in $\gamma_1, \ldots, \gamma_k$. Specific choices for $\gamma_k^{\perp}$ are presented in the numerical results section.

3.4 Implementation details

In this section, we describe how we incorporate the dense initialization within the existing LMTR algorithm [3]. At the beginning of each iteration, the LMTR algorithm with dense initialization checks if the unconstrained minimizer (also known as the *full quasi-Newton trial step*),

$$p_u^* = -\hat{B}_k^{-1} g_k \tag{31}$$

lies inside the trust region defined by the two-norm. Because our proposed method uses a dense initialization, the so-called "two-loop recursion" [6] is not applicable for computing the unconstrained minimizer p_u^* in (31). However, products with $\hat{B}_k^{-1}$ can be performed using the compact representation without involving a partial eigendecomposition. Specifically, if $V_k = [S_k \; Y_k]$ with Cholesky factorization $V_k^T V_k = R_k^T R_k$, then

$$\hat{B}_k^{-1} = \frac{1}{\gamma_k^{\perp}} I + V_k \hat{M}_k V_k^T, \tag{32}$$

where

$$\hat{M}_k = \begin{bmatrix} T_k^{-T}(D_k + \gamma_k^{-1} Y_k^T Y_k) T_k^{-1} & -\gamma_k^{-1} T_k^{-T} \\ -\gamma_k^{-1} T_k^{-1} & 0_m \end{bmatrix} + \alpha_k R_k^{-1} R_k^{-T},$$

$\alpha_k = \left(\dfrac{1}{\gamma_k} - \dfrac{1}{\gamma_k^{\perp}} \right)$, T_k is the upper triangular part of the matrix $S_k^T Y_k$, and D_k is its diagonal. Thus, the inequality

$$\|p_u^*\|_2 \leq \Delta_k \tag{33}$$

is easily verified without explicitly forming p_u^* using the identity

$$\|p_u^*\|_2^2 = g_k^T \hat{B}_k^{-2} g_k = \gamma_k^{-2} \|g_k\|^2 + 2\gamma_k^{-1} u_k^T \hat{M}_k u_k + u_k^T \hat{M}_k (R_k^T R_k) \hat{M}_k u_k. \tag{34}$$

Here, as in the LMTR algorithm, the vector $u_k = V_k^T g_k$ and $\|g_k\|^2$ can be computed efficiently at each iteration (see [3] for details). Thus, the computational cost of $\|p_u^*\|_2$ is low because (34) involves linear algebra operations in a small $2m$-dimensional space, the most expensive of which are related to solving triangular systems with T_k and R_k. These operations grow in proportion to m^2 while the number of operations in (31)–(32) grows in proportion to mn. Thus, the computational complexity ratio between using

(34) and (31)–(32) is $m^2/(nm) = m/n \ll 1$ since we assume that $m \ll n$. The norm equivalence for the shape-changing infinity norm studied in [3] guarantees that (33) implies that the inequality $\|p_u^*\|_{P,\infty} \leq \Delta_k$ is satisfied; in this case, p_u^* is the exact solution of the trust-region subproblem defined by the shape-changing infinity norm.

If (33) holds, the algorithm computes p_u^* for generating the trial point $x_k + p_u^*$. It can be easily seen that the cost of computing p_u^* is $4mn$ operations, i.e. it is the same as for computing search direction in the line search L-BFGS algorithm [6].

On the other hand, if (33) does not hold, then for producing a trial point, the partial eigendecomposition is computed, and the trust-region subproblem is decoupled and solved exactly as described in Sect. 3.2.

3.5 The algorithm and its properties

In Algorithm 1, we present a basic trust-region method that uses the proposed dense initialization. In this setting, we consider the computational cost of the proposed method, and we prove global convergence of the overall trust-region method. Since P may change every iteration, the corresponding norm $\|\cdot\|_{P,\infty}$ may change each iteration. Note that initially there are no stored quasi-Newton pairs $\{s_j, y_j\}$. In this case, we assume $P_\perp = I_n$ and $P_\parallel$ does not exist, i.e., $\hat{B}_0 = \gamma_0^\perp I$.

The only difference between Algorithm 1 and the LMTR algorithm in [3] is the initialization matrix. Computationally speaking, the use of a dense initialization in lieu of a diagonal initialization plays out only in the computation of p^* by (22). However, there is no computational cost difference: The cost of computing the value for β using (29) in Algorithm 1 instead of (21) in the LMTR algorithm is the same. Thus, the dominant cost per iteration for both Algorithm 1 and the LMTR algorithm is $4mn$ operations (see [3] for details). Note that this is the same cost-per-iteration as the line search L-BFGS algorithm [5].

In the next result, we provide a global convergence result for Algorithm 1. This result is based on the convergence analysis presented in [3].

Theorem 3.2 *Let $f : R^n \to R$ be twice-continuously differentiable and bounded below on R^n. Suppose that there exists a scalar $c_1 > 0$ such that*

$$\|\nabla^2 f(x)\| \leq c_1, \quad \forall\, x \in R^n. \tag{35}$$

Furthermore, suppose for $\hat{B}_0$ defined by (24), that there exists a positive scalar c_2 such that

$$\gamma_k, \gamma_k^\perp \in (0, c_2], \quad \forall\, k \geq 0, \tag{36}$$

and there exists a scalar $c_3 \in (0, 1)$ such that the inequality

$$s_j^T y_j > c_3 \|s_j\| \|y_j\| \tag{37}$$

Algorithm 1 An L-BFGS trust-region method with dense initialization

Require: $x_0 \in R^n$, $\Delta_0 > 0$, $\epsilon > 0$, $\gamma_0^{\perp} > 0$, $0 \leq \tau_1 < \tau_2 < 0.5 < \tau_3 < 1$, $0 < \eta_1 < \eta_2 \leq 0.5 < \eta_3 < 1 < \eta_4, 0 < c_3 < 1$

Compute g_0
for $k = 0, 1, 2, \ldots$ **do**
  if $\|g_k\| \leq \epsilon$ **then**
    return
  end if
  Compute $\|p_u^*\|_2$ using (34)
  if $\|p_u^*\|_2 > \Delta_k$ **then**
    Compute p^* for $\hat{B}_k$ using (30), where $\widehat{\beta}$ is computed using (29) and $v_{\|}^*$ as in (19)
  else
    Compute p_u^* using (31)–(32) and set $p^* \leftarrow p_u^*$
  end if
  Compute the ratio $\rho_k = \frac{f(x_k+p^*)-f(x_k)}{Q(p^*)}$
  if $\rho_k \geq \tau_1$ **then**
    $x_{k+1} = x_k + p^*$
    Compute g_{k+1}, s_k, y_k, γ_{k+1} and $\gamma_{k+1}^{\perp}$
    Choose at most m pairs $\{s_j, y_j\}$ such that $s_j^T y_j > c_3 \|s_j\| \|y_j\|$
    Compute Ψ_{k+1}, R^{-1}, M_{k+1}, W, $\hat{\Lambda}$ and Λ as described in Sect. 2
  else
    $x_{k+1} = x_k$
  end if
  if $\rho_k < \tau_2$ **then**
    $\Delta_{k+1} = \min\left(\eta_1 \Delta_k, \eta_2 \|s_k\|_{P,\infty}\right)$
  else
    if $\rho_k \geq \tau_3$ **and** $\|s_k\|_{P,\infty} \geq \eta_3 \Delta_k$ **then**
      $\Delta_{k+1} = \eta_4 \Delta_k$
    else
      $\Delta_{k+1} = \Delta_k$
    end if
  end if
end for

holds for each quasi-Newton pair $\{s_j, y_j\}$. *Then, if the stopping criteria is suppressed, the infinite sequence* $\{x_k\}$ *generated by Algorithm 1 satisfies*

$$\lim_{k \to \infty} \|\nabla f(x_k)\| = 0. \tag{38}$$

Proof From (36), we have $\|\hat{B}_0\| \leq c_2$, which holds for each $k \geq 0$. Then, by [3, Lemma 3], there exists $c_4 > 0$ such that

$$\|\hat{B}_k\| \leq c_4.$$

Then, (38) follows from [3, Theorem 1]. □

In the following section, we consider $\gamma_k^{\perp}$ parameterized by two scalars, c and λ:

$$\gamma_k^{\perp}(c, \lambda) = \lambda c \gamma_k^{\max} + (1 - \lambda)\gamma_k, \tag{39}$$

where $c \geq 1$, $\lambda \in [0, 1]$, and

$$\gamma_k^{\max} \triangleq \max_{1 \leq i \leq k} \gamma_i,$$

where γ_k is taken to be the conventional initialization given by (4). (This choice for $\gamma_k^{\perp}$ will be further discussed in Sect. 4.) We now show that Algorithm 1 converges for these choices of $\gamma_k^{\perp}$. Assuming that (35) and (37) hold, it remains to show that (36) holds for these choices of $\gamma_k^{\perp}$. To see that (36) holds, notice that in this case,

$$\gamma_k = \frac{y_k^T y_k}{s_k^T y_k} \leq \frac{y_k^T y_k}{c_3 \|s_k\| \|y_k\|} \leq \frac{\|y_k\|}{c_3 \|s_k\|}.$$

Substituting in for the definitions of y_k and s_k yields that

$$\gamma_k \leq \frac{\|\nabla f(x_{k+1}) - \nabla f(x_k)\|}{c_3 \|x_{k+1} - x_k\|},$$

implying that (36) holds. Thus, Algorithm 1 converges for these choices for $\gamma_k^{\perp}$.

4 Numerical experiments

We performed numerical experiments using a Dell Precision T1700 machine with an Intel i5-4590 CPU at 3.30GHz X4 and 8GB RAM using MATLAB 2014a. The test set consisted of 65 large-scale ($1000 \leq n \leq 10000$) CUTEst [10] test problems, made up of all the test problems in [3] plus an additional three (FMINSURF, PENALTY2, and TESTQUAD [10]) since at least one of the methods in the experiments detailed below converged on one of these three problems. The same trust-region method and default parameters as in [3, Algorithm 1] were used for the outer iteration. At most five quasi-Newton pairs $\{s_k, y_k\}$ were stored, i.e., $m = 5$. The relative stopping criterion was

$$\|g_k\|_2 \leq \epsilon \max(1, \|x_k\|_2),$$

with $\epsilon = 10^{-10}$. The initial step, p_0, was determined by a backtracking line-search along the normalized steepest descent direction. To compute the partial eigendecomposition of B_k, we used the QR factorization instead of the SVD because the QR version outperformed the SVD version in numerical experiments not presented here. The rank of Ψ_k was estimated by the number of positive diagonal elements in the diagonal matrix of the LDL$^{\mathrm{T}}$ decomposition (or eigendecomposition of $\Psi_k^T \Psi_k$) that are larger than the threshold $\epsilon_r = (10^{-7})^2$. (Note that the columns of Ψ_k are normalized.). We used the value $c_3 = 10^{-8}$ in (37) for testing whether to accept a new quasi-Newton pair.

We provide performance profiles (see [7]) for the number of iterations (`iter`) where the trust-region step is accepted and the average time (`time`) for each solver

on the test set of problems. The performance metric, ρ, for the 65 problems is defined by

$$\rho_s(\tau) = \frac{\text{card}\left\{p : \pi_{p,s} \leq \tau\right\}}{65} \quad \text{and} \quad \pi_{p,s} = \frac{t_{p,s}}{\min\limits_{1 \leq i \leq S} t_{p,i}},$$

where $t_{p,s}$ is the "output" (i.e., time or iterations) of "solver" s on problem p. Here S denotes the total number of solvers for a given comparison. This metric measures the proportion of how close a given solver is to the best result. We observe as in [3] that the first runs significantly differ in time from the remaining runs, and thus, we ran each algorithm ten times on each problem, reporting the average of the final eight runs.

In this section, we present the following six types of experiments involving LMTR:

1. A comparison of results for different values of $\gamma_k^{\perp}(c, \lambda)$.
2. Two versions of computing the full quasi-Newton trial step are compared. One version uses the dense initialization to compute p_u^* as described in Sect. 3.4 (see (31)); the other uses the conventional initialization, i.e., p_u^* is computed as $p_u^* = B_k^{-1} g_k$. When the full quasi-Newton trial step is not accepted in any of the versions, the dense initialization is used for computing trial step by explicitly solving the trust-region subproblem (Sect. 3.2).
3. A comparison of LMTR together with a dense initialization and the line search L-BFGS method with the conventional initialization.
4. A comparison of LMTR with a dense initialization and L-BFGS-TR [3], which computes a scaled quasi-Newton direction that lies inside a trust region. This method can be viewed as a hybrid line search and trust-region algorithm.
5. A comparison of the dense and conventional initializations.

In the experiments below, the dense initial matrix $\widehat{B}_0$ corresponding to $\gamma_k^{\perp}(c, \lambda)$ given in (39) will be denoted by

$$\widehat{B}_0(c, \lambda) \triangleq \gamma_k P_{\parallel} P_{\parallel}^T + \gamma_k^{\perp}(c, \lambda) P_{\perp} P_{\perp}^T.$$

Using this notation, the conventional initialization $B_0(\gamma_k)$ can be written as $\widehat{B}_0(1, 0)$.

Experiment 1 In this experiment, we consider various scalings of a proposed $\gamma_k^{\perp}$ using LMTR. As argued in Sect. 3.3, it is reasonable to choose $\gamma_k^{\perp}$ to be large and positive; in particular, $\gamma_k^{\perp} \geq \gamma_k$. Thus, we consider the parametrized family of choices $\gamma_k^{\perp} \triangleq \gamma_k^{\perp}(c, \lambda)$ given in (39). These choices correspond to conservative strategies for computing steps in the space spanned by $P_{\perp}$ (see the discussion in Sect. 3.3). Moreover, these can also be viewed as conservative strategies since the trial step computed using B_0 will always be larger in Euclidean norm than the trial step computed using $\widehat{B}_0$ using (39). To see this, note that in the parallel subspace the solutions will be identical using both initializations since the solution $v_{\parallel}^*$ does not depend on $\gamma_k^{\perp}$ (see (19)); in contrast, in the orthogonal subspace, $\|v_{\perp}^*\|$ inversely depends on $\gamma_k^{\perp}$ (see (28) and (29)).

Table 1 Values for $\gamma_k^{\perp}$ used in Experiment 1

Parameters		
c	λ	$\gamma_k^{\perp}$
1	1	$\gamma_k^{\max}$
2	1	$2\gamma_k^{\max}$
1	$\frac{1}{2}$	$\frac{1}{2}\gamma_k^{\max} + \frac{1}{2}\gamma_k$
1	$\frac{1}{4}$	$\frac{1}{4}\gamma_k^{\max} + \frac{3}{4}\gamma_k$

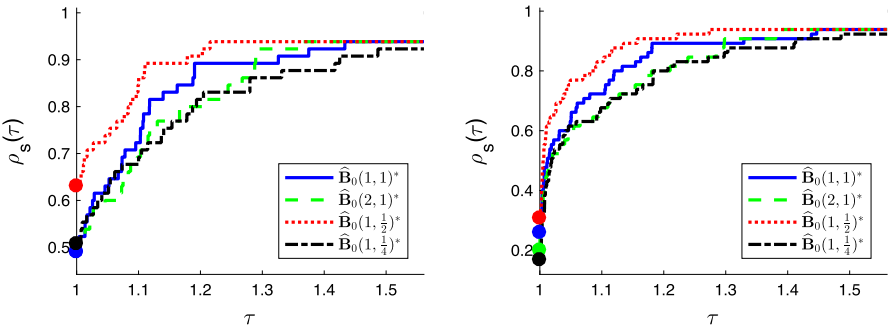

Fig. 1 Performance profiles comparing `iter` (left) and `time` (right) for the different values of $\gamma_k^{\perp}$ given in Table 1. In the legend, $\widehat{B}_0(c,\lambda)$ denotes the results from using the dense initialization with the given values for c and λ to define $\gamma_k^{\perp}$. In this experiment, the dense initialization was used for all aspects of the algorithm

We report results using different values of c and λ for $\gamma_k^{\perp}(c,\lambda)$ on two sets of tests. On the first set of tests, the dense initialization was used for the entire LMTR algoirthm. However, for the second set of tests, the dense initialization was not used for the computation of the unconstrained minimizer p_u^*; that is, LMTR was run using B_k (initialized with $B_0 = \gamma_k I$ where γ_k is given in (4)) for the computation of the unconstrained minimizer $p_u^* = -B_k^{-1} g_k$. However, if the unconstrained minimizer was not taken to be the approximate solution of the subproblem, $\widehat{B}_k$ with the dense initialization was used for computing the constrained minimizer with respect to the shape-changing norm (see line 8 in Algorithm 1) with $\gamma_k^{\perp}$ defined as in (39). The values of c and λ chosen for Experiment 1 are found in Table 1. (See Sect. 3.4 for details on the LMTR algorithm.)

Figure 1 displays the performance profiles using the chosen values of c and λ to define $\gamma_k^{\perp}$ in the case when the dense initialization was used for both the computation of the unconstrained minimizer p_u^* (line 10 of Algorithm 1) as well as for the constrained minimizer with respect to the shape-changing norm (line 8 of Algorithm 1), which is denoted in the legend of plots in Fig. 1 by the use of an asterisk $(*)$. The results of Fig. 1 suggest the choice of $c = 1$ and $\lambda = \frac{1}{2}$ outperform the other chosen combinations for c and λ. In experiments not reported here, larger values of c did not appear to improve performance; for $c < 1$, performance deteriorated. Moreover, other choices for λ, such as $\lambda = \frac{3}{4}$, did not improve results beyond the choice of $\lambda = \frac{1}{2}$.

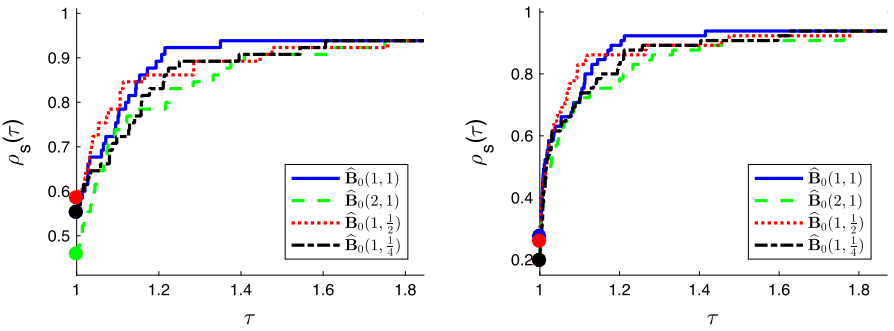

Fig. 2 Performance profiles comparing `iter` (left) and `time` (right) for the different values of $\gamma_k^{\perp}$ given in Table 1. In the legend, $\widehat{B}_0(c, \lambda)$ denotes the results from using the dense initialization with the given values for c and λ to define $\gamma_k^{\perp}$. In this experiment, the dense initialization was only used for the computation of the constrained minimizer (line 8 of Algorithm 1)

Figure 2 reports the performance profiles for using the chosen values of c and λ to define $\gamma_k^{\perp}$ in the case when the dense initialization was only used for the computation of the constrained minimizer (line 8 of Algorithm 1) –denoted in the legend of plots in Fig. 2 by the absence of an asterisk $(*)$. In this test, the combination of $c = 1$ and $\lambda = 1$ as well as $c = 1$ and $\lambda = \frac{1}{2}$ appear to slightly outperform the other two choices for $\gamma_k^{\perp}$ in terms of both then number of iterations and the total computational time. Based on the results in Fig. 2, we do not see a reason to prefer either combination $c = 1$ and $\lambda = 1$ or $c = 1$ and $\lambda = \frac{1}{2}$ over the other.

Note that for the CUTEst problems, the full quasi-Newton trial step is accepted as the solution to the subproblem on the overwhelming majority of problems. Thus, if the scaling $\gamma_k^{\perp}$ is used only when the full trial step is rejected, it has less of an affect on the overall performance of the algorithm; i.e., the algorithm is less sensitive to the choice of $\gamma_k^{\perp}$. For this reason, it is not surprising that the performance profiles in Fig. 2 for the different values of $\gamma_k^{\perp}$ are more indistinguishable than those in Fig. 1.

Finally, similar to the results in the case when the dense initialization was used for the entire algorithm (Fig. 1), other values of c and λ did not significantly improve the performance provided by $c = 1$ and $\lambda = \frac{1}{2}$.

Experiment 2 This experiment was designed to test whether it is advantageous to use the dense initialization for all aspects of the LMTR algorithm or just for the computation of the constrained minimizer (line 8 of Algorithm 1). For any given trust-region subproblem, using the dense initialization for computing the unconstrained minimizer is computationally more expensive than using a diagonal initialization; however, it is possible that extra computational time associated with using the dense initialization for all aspects of the LMTR algorithm may yield a more overall efficient solver. For these tests, we compare the top performer in the case when the dense initialization is used for all aspects of LMTR, i.e., $(\gamma_k^{\perp}(1, \frac{1}{2}))$, to one of the top performers in the case when the dense initialization is used only for the computation of the constrained minimizer (line 8 of Algorithm 1), i.e., $(\gamma_k^{\perp}(1, 1))$.

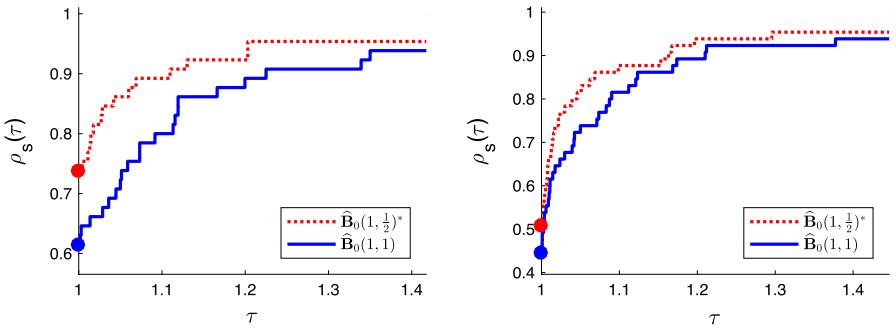

Fig. 3 Performance profiles of `iter` (left) and `time` (right) for Experiment 2. In the legend, the asterisk after $\widehat{B}_0(1, \frac{1}{2})^*$ signifies that the dense initialization was used for all aspects of the LMTR algorithm; without the asterisk, $\widehat{B}_0(1, 1)$ signifies the test where the dense initialization is used only for the computation of the constrained minimizer (line 8 of Algorithm 1)

The performance profiles comparing the results of this experiment are presented in Fig. 3. These results suggest that using the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ for all aspects of the LMTR algorithm is more efficient than using dense initializations only for the computation of the constrained minimizer (line 8 of Algorithm 1). In other words, even though using dense initial matrices for the computation of the unconstrained minimizer imposes an additional computational burden, it generates steps that expedite the convergence of the overall trust-region method.

Experiment 3 In this experiment, we compare the performance of the dense initialization $\gamma_k^{\perp}(1, 0.5)$ to that of the line-search L-BFGS algorithm. For this comparison, we used the publicly-available MATLAB wrapper [1] for the FORTRAN L-BFGS-B code developed by Nocedal et al. [15]. The initialization for L-BFGS-B is $B_0 = \gamma_k I$ where γ_k is given by (4). To make the stopping criterion equivalent to that of L-BFGS-B, we modified the stopping criterion of our solver to [15]:

$$\|g_k\|_\infty \leq \epsilon.$$

The dense initialization was used for all aspects of LMTR.

The performance profiles for this experiment is given in Fig. 4. On this test set, the dense initialization outperforms L-BFGS-B in terms of both the number of iterations and the total computational time.

Experiment 4 In this experiment, we compare LMTR with a dense initialization to L-BFGS-TR [3], which computes an L-BFGS trial step whose length is bounded by a trust-region radius. This method can be viewed as a hybrid L-BFGS line search and trust-region algorithm because it uses a standard trust-region framework (as LMTR) but computes a trial point by minimizing the quadratic model in the trust region along the L-BFGS direction. In [3], it was determined that this algorithm outperforms two other versions of L-BFGS that use a Wolfe line search. (For further details, see [3].)

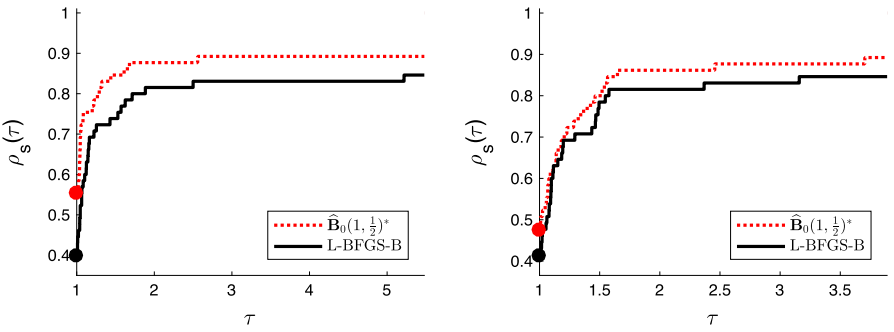

Fig. 4 Performance profiles of `iter` (left) and `time` (right) for Experiment 3 comparing LMTR with the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ to L-BFGS-B

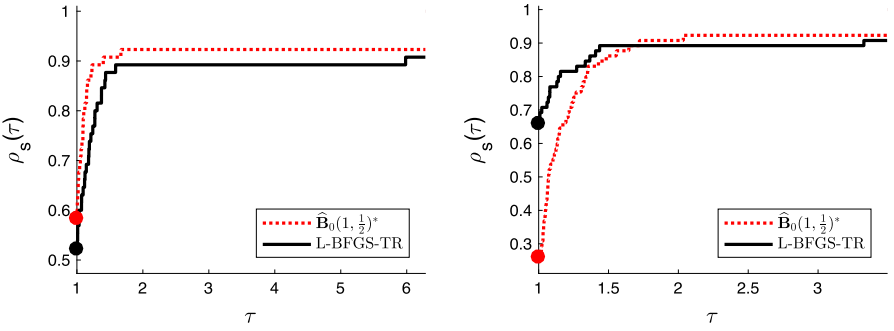

Fig. 5 Performance profiles of `iter` (left) and `time` (right) for Experiment 4 comparing LMTR with the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ to L-BFGS-TR

Figure 5 displays the performance profiles associated with this experiment on the entire set of test problems. For this experiment, the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ was used in all aspects of the LMTR algorithm. In terms of total number of iterations, LMTR with the dense initialization outperformed L-BFGS-TR; however, L-BFGS-TR appears to have outperformed LMTR with the dense initialization in computational time.

Figure 5 (left) indicates that the quality of the trial points produced by solving the trust-region subproblem exactly using LMTR with the dense initialization is generally better than in the case of the line search applied to the L-BFGS direction. However, Fig. 5 (right) shows that LMTR with the dense initialization requires more computational effort than L-BFGS-TR. For the CUTEst set of test problems, L-BFGS-TR does not need to perform a line search for the majority of iterations; that is, the full quasi-Newton trial step is accepted in a majority of the iterations. Therefore, we also compared the two algorithms on a subset of the most difficult test problems–namely, those for which an *active* line search is needed to be performed by L-BFGS-TR. To this end, we select, as in [3], those of the CUTEst problems in which the full L-BFGS (i.e., the step size of one) was rejected in at least 30% of the iterations. The number of problems in this subset is 14. The performance profiles associated with this reduced test set are in Fig. 6. On

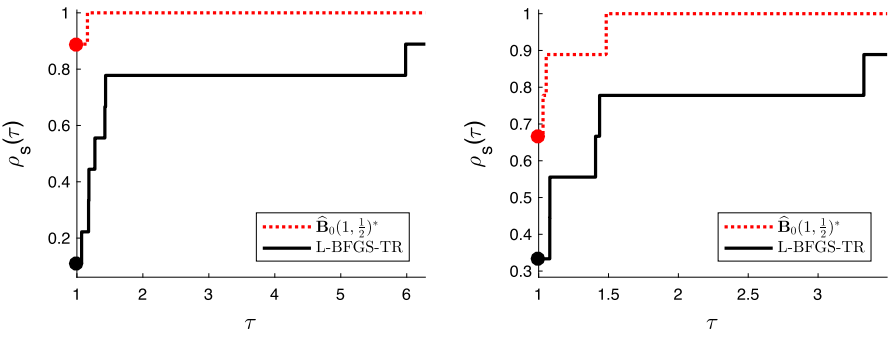

Fig. 6 Performance profiles of `iter` (left) and `time` (right) for Experiment 4 comparing LMTR with the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ to L-BFGS-TR on the subset of 14 problems for which L-BFGS-TR implements a line search more than 30% of the iterations

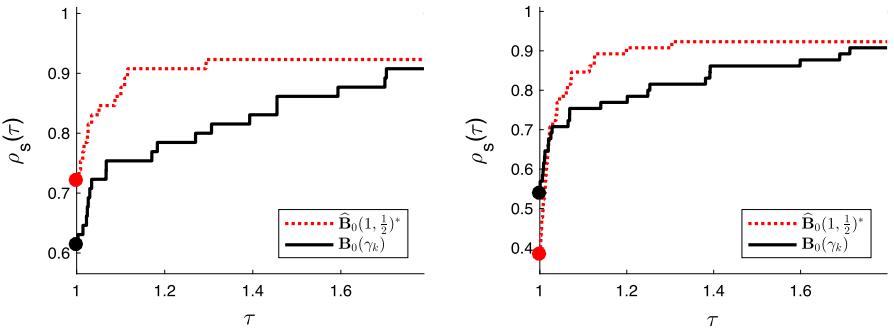

Fig. 7 Performance profiles of `iter` (left) and `time` (right) for Experiment 5 comparing LMTR with the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ to LMTR with the conventional initialization

this smaller test set, LMTR outperforms L-BFGS-TR both in terms of total number of iterations and computational time.

Finally, Figs. 5 and 6 suggest that when function and gradient evaluations are expensive (e.g., simulation-based applications), LMTR together with the dense initialization is expected to be more efficient than L-BFGS-TR since both on both test sets LMTR with the dense initialization requires fewer overall iterations. Moreover, Fig. 6 suggests that on problems where the L-BFGS search direction often does not provide sufficient decrease of the objective function, LMTR with the dense initialization is expected to perform better.

Experiment 5 In this experiment, we compare the results of LMTR using the dense initialization to that of LMTR using the conventional diagonal initialization $B_0 = \gamma_k I$ where γ_k is given by (3). The dense initialization selected was chosen to be the top performer from Experiment 2 (i.e., $\gamma_k^{\perp}(1, \frac{1}{2})$).

From Fig. 7, the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ outperforms the conventional initialization for LMTR in terms of iteration count; however, it is unclear whether the algorithm benefits from the dense initialization in terms of computational time. The reason for this is that the dense initialization is being used for all aspects of the LMTR

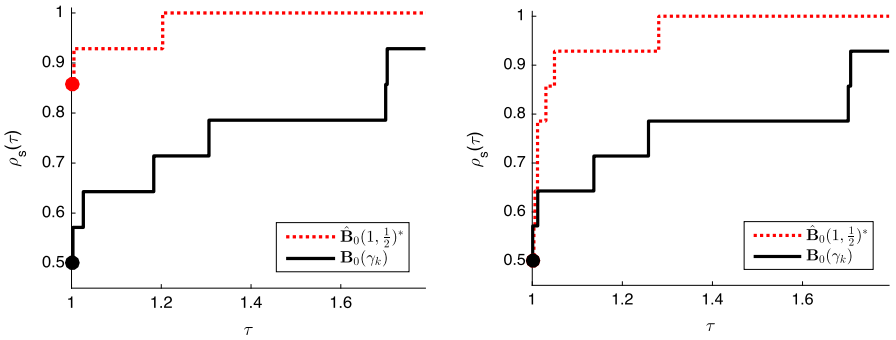

Fig. 8 Performance profiles of `iter` (left) and `time` (right) for Experiment 5 comparing LMTR with the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ to LMTR with the conventional initialization on the subset of 14 problems in which the unconstrained minimizer is rejected at 30% of the iterations

algorithm; in particular, it is being used to compute the full quasi-Newton step p_u^* (see the discussion in Experiment 1), which is typically accepted most iterations on the CUTEst test set. Therefore, as in Experiment 5, we compared LMTR with the dense initialization and the conventional initialization on the subset of 14 problems in which the unconstrained minimizer is rejected at least 30% of the iterations. The performance profiles associated with this reduced set of problems are found in Fig. 8. The results from this experiment clearly indicate that on these more difficult problems the dense initialization outperforms the conventional initialization in both iteration count and computational time.

5 Conclusion

In this paper, we presented a dense initialization for quasi-Newton methods to solve unconstrained optimization problems. This initialization makes use of two curvature estimates for the underlying function in two complementary subspaces. Importantly, this initialization neither introduces additional computational cost nor increases storage requirements; moreover, it maintains theoretical convergence properties of quasi-Newton methods. It should also be noted that this initialization still makes it possible to efficiently compute products and perform solves with the sequence of quasi-Newton matrices.

The dense initialization is especially well-suited for use in the shape-changing infinity-norm L-BFGS trust-region method. Numerical results on the outperforms both the standard L-BFGS line search method as well as the same shape-changing trust-region method with the conventional initialization. Use of this initialization is possible with any quasi-Newton method for which the update has a compact representation. While this initialization has broad applications for quasi-Newton line search and trust-region methods, its use makes most sense from a computational point of view when the quasi-Newton method already computes the compact formulation and partial eigen-decomposition; if this is not the case, using the dense initialization will result in additional computational expense that must be weighed against its benefits.

References

1. Becker, S.: LBFGSB (L-BFGS-B) mex wrapper (2012–2015). https://www.mathworks.com/matlabcentral/fileexchange/35104-lbfgsb-l-bfgs-b-mex-wrapper. Accessed Jan 2017
2. Brust, J., Burdakov, O., Erway, J.B., Marcia, R.F., Yuan, Y.X.: Shape-changing L-SR1 trust-region methods. Technical Report 2016-2, Wake Forest University (2016)
3. Burdakov, O., Gong, L., Yuan, Y.X., Zikrin, S.: On efficiently combining limited memory and trust-region techniques. Math. Program. Comput. **9**, 101–134 (2016)
4. Burke, J.V., Wiegmann, A., Xu, L.: Limited memory BFGS updating in a trust-region framework. Technical Report, University of Washington (1996)
5. 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)
6. DeGuchy, O., Erway, J.B., Marcia, R.F.: Compact representation of the full Broyden class of quasi-Newton updates. Numer. Linear Algebra Appl. **25**(5), e2186 (2018)
7. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
8. 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)
9. Erway, J.B., Marcia, R.F.: On solving large-scale limited-memory quasi-Newton equations. Linear Algebra Appl. **515**, 196–225 (2017)
10. Gould, N.I.M., Orban, D., Toint, P.L.: CUTEr and SifDec: a constrained and unconstrained testing environment, revisited. ACM Trans. Math. Softw. **29**(4), 373–394 (2003)
11. Lu, X.: A study of the limited memory SR1 method in practice. Ph.D. thesis, University of Colorado (1992)
12. Lukšan, L., Vlček, J.: Recursive form of general limited memory variable metric methods. Kybernetika **49**, 224–235 (2013)
13. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
14. Shanno, D.F., Phua, K.H.: Matrix conditioning and nonlinear optimization. Math. Program. **14**(1), 149–160 (1978)
15. Zhu, C., Byrd, R., Nocedal, J.: Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw. **23**, 550–560 (1997)

Affiliations

Johannes Brust[1] · Oleg Burdakov[2] · Jennifer B. Erway[3] · Roummel F. Marcia[4]

Johannes Brust
jbrust@ucmerced.edu

Oleg Burdakov
oleg.burdakov@liu.se

Roummel F. Marcia
rmarcia@ucmerced.edu

[1] Department of Applied Mathematics, University of California Merced, Merced, CA, USA

[2] Department of Mathematics, Linköping University, Linköping, Sweden

[3] Department of Mathematics and Statistics, Wake Forest University, Winston-Salem, NC, USA

[4] Department of Applied Mathematics, University of California Merced, Merced, CA, USA