

A dense initialization for limited-memory quasi-Newton methods

Johannes Brust, Oleg Burdakov, Jennifer B. Erway and Roummel F. Marcia

The self-archived postprint version of this journal article is available at Linköping University Institutional Repository (DiVA):

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-143315>

N.B.: When citing this work, cite the original publication.

The original publication is available at www.springerlink.com:

Brust, J., Burdakov, O., Erway, J. B., Marcia, R. F., (2019), A dense initialization for limited-memory quasi-Newton methods, *Computational Optimization and Applications*, 1-22. <https://doi.org/10.1007/s10589-019-00112-x>

Original publication available at:

<https://doi.org/10.1007/s10589-019-00112-x>

Copyright: Springer

<http://www.springer.com/>



DENSE INITIALIZATIONS FOR LIMITED-MEMORY QUASI-NEWTON METHODS

JOHANNES BRUST, OLEG BURDAKOV, JENNIFER B. ERWAY, AND ROUMMEL F. MARCIA

ABSTRACT. We consider a family of dense initializations for limited-memory quasi-Newton methods. The proposed initialization uses two parameters to approximate the curvature of the Hessian in two complementary subspaces. 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.

1. INTRODUCTION

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

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

where $f(\vec{x}) : \Re^n \rightarrow \Re$ is a general nonconvex function that is at least continuously differentiable. 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 an eigendecomposition-based separation of \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. A different scaling parameter may then be used for each subspace. 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 application; 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 L-BFGS update is the most widely-used quasi-Newton update for large-scale optimization; it is defined by the recursion formula

$$(1) \quad \vec{B}_{k+1} = \vec{B}_k - \frac{1}{\vec{s}_k^T \vec{B}_k \vec{s}_k} \vec{B}_k \vec{s}_k \vec{s}_k^T \vec{B}_k + \frac{1}{\vec{s}_k^T \vec{y}_k} \vec{y}_k \vec{y}_k^T,$$

where

$$(2) \quad \vec{s}_k \triangleq \vec{x}_{k+1} - \vec{x}_k \quad \text{and} \quad \vec{y}_k \triangleq \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k),$$

Date: October 12, 2017.

Key words and phrases. Large-scale nonlinear optimization, limited-memory quasi-Newton methods, trust-region methods, quasi-Newton matrices, shape-changing norm.

J. B. Erway is supported in part by National Science Foundation grants CMMI-1334042 and IIS-1741264.

R. F. Marcia is supported in part by National Science Foundation grants CMMI-1333326 and IIS-1741490.

and $\vec{B}_0 \in \mathbb{R}^{n \times n}$ is a suitably-chosen initial matrix. This rank-two update to B_k preserves positive definiteness when $\vec{s}_k^T \vec{y}_k > 0$. For large-scale problems only $m \ll n$ of the most recent updates $\{\vec{s}_i, \vec{y}_i\}$ with $k - m \leq i \leq k - 1$ are stored in L-BFGS methods. (Typically, $m \in [3, 7]$ (see, e.g., [5]).)

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 \vec{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, \vec{B}_0 is often chosen to be a scalar multiple of the identity matrix, i.e.,

$$(3) \quad \vec{B}_0 = \gamma_k \vec{I}, \quad \text{with } \gamma_k > 0.$$

For BFGS matrices, the conventional choice for γ_k is

$$(4) \quad \gamma_k = \frac{\vec{y}_k^T \vec{y}_k}{\vec{s}_k^T \vec{y}_k},$$

which can be viewed as a spectral estimate for of $\nabla^2 f(\vec{x}_k)$ [12]. (This choice was originally proposed in [13] 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

$$(5) \quad \gamma_k = \underset{\gamma}{\operatorname{argmin}} \left\| \vec{B}_0^{-1} \vec{y}_k - \vec{s}_k \right\|_2^2,$$

where $\vec{B}_0^{-1} = \gamma^{-1} \vec{I}$. 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 \mathbb{R}^n into two complementary subspaces allows for different parameters to be used to estimate the curvature of the underlying Hessian in these subspaces. 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 Section 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 Section 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 Section 4, and concluding remarks are found in Section 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

$$\vec{S}_k \triangleq [\vec{s}_{k-m} \cdots \vec{s}_{k-1}] \quad \text{and} \quad \vec{Y}_k \triangleq [\vec{y}_{k-m} \cdots \vec{y}_{k-1}].$$

With \vec{L}_k taken to be the strictly lower triangular part of the matrix of $\vec{S}_k^T \vec{Y}_k$, and \vec{D}_k defined as the diagonal of $\vec{S}_k^T \vec{Y}_k$, the compact representation of an L-BFGS matrix is

$$(6) \quad \vec{B}_k = \vec{B}_0 + \vec{\Psi}_k \vec{M}_k \vec{\Psi}_k^T,$$

where

$$(7) \quad \vec{\Psi}_k \triangleq \begin{bmatrix} \vec{B}_0 \vec{S}_k & \vec{Y}_k \end{bmatrix} \quad \text{and} \quad \vec{M}_k \triangleq - \begin{bmatrix} \vec{S}_k^T \vec{B}_0 \vec{S}_k & \vec{L}_k \\ \vec{L}_k^T & -\vec{D}_k \end{bmatrix}^{-1}$$

(see [5] for details). Note that $\vec{\Psi}_k \in \mathbb{R}^{n \times 2m}$, and $\vec{M}_k \in \mathbb{R}^{2m \times 2m}$ 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 \vec{B}_0 is appropriately chosen, then computing products with B_k or solving linear systems with B_k can be done efficiently [9].

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

2.2. Partial eigendecomposition of \vec{B}_k . If \vec{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 $\vec{\Psi}_k$ has rank $r = 2m$. (For the rank-deficient case, see the techniques found in [3].)

Let

$$\vec{\Psi}_k = \vec{Q} \vec{R},$$

be the so-called “thin” QR factorization of Ψ_k , where $\vec{Q} \in \mathbb{R}^{n \times r}$ and $\vec{R} \in \mathbb{R}^{r \times r}$. Since the matrix $\vec{R} \vec{M}_k \vec{R}^T$ 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 $\vec{W} \hat{\Lambda} \vec{W}^T$ is the eigendecomposition of $\vec{R} \vec{M}_k \vec{R}^T$. Then,

$$\vec{\Psi}_k \vec{M}_k \vec{\Psi}_k^T = \vec{Q} \vec{R} \vec{M}_k \vec{R}^T \vec{Q}^T = \vec{Q} \vec{W} \hat{\Lambda} \vec{W}^T \vec{Q}^T = \vec{\Psi}_k \vec{R}^{-1} \vec{W} \hat{\Lambda} \vec{W}^T \vec{R}^{-T} \vec{\Psi}_k^T.$$

Defining

$$(8) \quad \vec{P}_{\parallel} = \vec{\Psi}_k \vec{R}^{-1} \vec{W},$$

gives that

$$(9) \quad \vec{\Psi}_k \vec{M}_k \vec{\Psi}_k^T = \vec{P}_{\parallel} \hat{\Lambda} \vec{P}_{\parallel}^T.$$

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

$$(10) \quad \vec{B}_k = \gamma_k \vec{I} + \vec{\Psi}_k \vec{M}_k \vec{\Psi}_k^T = P \Lambda P^T,$$

where

$$(11) \quad P \triangleq \begin{bmatrix} \vec{P}_{\parallel} & \vec{P}_{\perp} \end{bmatrix}, \quad \Lambda \triangleq \begin{bmatrix} \hat{\Lambda} + \gamma_k I_r & \\ & \gamma_k I_{n-r} \end{bmatrix},$$

and $\vec{P}_{\perp} \in \mathbb{R}^{n \times (n-r)}$ is defined as the orthogonal complement of \vec{P}_{\parallel} , i.e., $\vec{P}_{\perp}^T \vec{P}_{\perp} = \vec{I}_{n-r}$ and $\vec{P}_{\parallel}^T \vec{P}_{\perp} = \vec{0}_{r \times (n-r)}$. Hence, \vec{B}_k has r eigenvalues given by the diagonal elements of $\hat{\Lambda} + \gamma_k \vec{I}_r$ 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_{\parallel} . Formula (8) indicates that Q is not required to be explicitly formed in order to compute products with P_{\parallel} . For this reason, it is desirable to avoid forming Q by computing only R via the Cholesky factorization of $\vec{\Psi}_k^T \vec{\Psi}_k$, i.e., $\vec{\Psi}_k^T \vec{\Psi}_k = \vec{R}^T \vec{R}$ (see [3]).

At an additional expense, the eigenvectors stored in the columns of P_{\parallel} may be formed and stored. For the shape-changing trust-region method used in this paper, it is not required to store P_{\parallel} . 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 $\vec{P}_{\perp} \vec{P}_{\perp}^T$, which can be done using the identity

$$(12) \quad \vec{P}_{\perp} \vec{P}_{\perp}^T = \vec{I} - \vec{P}_{\parallel} \vec{P}_{\parallel}^T.$$

2.3. A shape-changing L-BFGS trust-region method. Generally speaking, at the k th step of a trust-region method, a search direction is computed by approximately solving the trust-region subproblem

$$(13) \quad \bar{p}^* = \underset{\|\bar{p}\| \leq \Delta_k}{\operatorname{argmin}} Q(\bar{p}) \triangleq \bar{g}_k^T \bar{p} + \frac{1}{2} \bar{p}^T \bar{B}_k \bar{p},$$

where $\bar{g}_k \triangleq \nabla f(\bar{x}_k)$, $\bar{B}_k \approx \nabla^2 f(\bar{x}_k)$, 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].

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

$$(14) \quad \underset{\|\bar{p}\|_{\bar{P}, \infty} \leq \Delta_k}{\operatorname{minimize}} Q(\bar{p}) = \bar{g}_k^T \bar{p} + \frac{1}{2} \bar{p}^T \bar{B}_k \bar{p},$$

where

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

and \bar{P}_{\parallel} and \bar{P}_{\perp} are given in (11). Note that the ratio $\|\bar{p}\|_2 / \|\bar{p}\|_{\bar{P}, \infty}$ does not depend on n , but only moderately depends on r . (In particular, $1 \leq \|\bar{p}\|_2 / \|\bar{p}\|_{\bar{P}, \infty} \leq \sqrt{r+1}$.) Because this norm depends on the eigenvectors of \bar{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 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

$$(16) \quad \bar{v} = \bar{P}^T \bar{p} = \begin{bmatrix} \bar{P}_{\parallel}^T \bar{p} \\ \bar{P}_{\perp}^T \bar{p} \end{bmatrix} \triangleq \begin{bmatrix} \bar{v}_{\parallel} \\ \bar{v}_{\perp} \end{bmatrix} \quad \text{and} \quad \bar{P}^T \bar{g}_k = \begin{bmatrix} \bar{P}_{\parallel}^T \bar{g}_k \\ \bar{P}_{\perp}^T \bar{g}_k \end{bmatrix} \triangleq \begin{bmatrix} \bar{g}_{\parallel} \\ \bar{g}_{\perp} \end{bmatrix}.$$

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

$$\begin{aligned} Q(\bar{P}\bar{v}) &= \bar{g}_k^T \bar{P}\bar{v} + \frac{1}{2} \bar{v}^T \left(\hat{\Lambda} + \gamma_k \bar{I}_n \right) \bar{v} \\ &= \bar{g}_{\parallel}^T \bar{v}_{\parallel} + \bar{g}_{\perp}^T \bar{v}_{\perp} + \frac{1}{2} \left(\bar{v}_{\parallel}^T \left(\hat{\Lambda} + \gamma_k \bar{I}_r \right) \bar{v}_{\parallel} + \gamma_k \|\bar{v}_{\perp}\|_2^2 \right) \\ &= \bar{g}_{\parallel}^T \bar{v}_{\parallel} + \frac{1}{2} \bar{v}_{\parallel}^T \left(\hat{\Lambda} + \gamma_k \bar{I}_r \right) \bar{v}_{\parallel} + \bar{g}_{\perp}^T \bar{v}_{\perp} + \frac{1}{2} \gamma_k \|\bar{v}_{\perp}\|_2^2. \end{aligned}$$

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

$$(17) \quad \underset{\|\bar{v}_{\parallel}\|_{\infty} \leq \Delta_k}{\operatorname{minimize}} q_{\parallel}(\bar{v}_{\parallel}) \triangleq \bar{g}_{\parallel}^T \bar{v}_{\parallel} + \frac{1}{2} \bar{v}_{\parallel}^T \left(\hat{\Lambda} + \gamma_k \bar{I}_r \right) \bar{v}_{\parallel}$$

$$(18) \quad \underset{\|\bar{v}_{\perp}\|_2 \leq \Delta_k}{\operatorname{minimize}} q_{\perp}(\bar{v}_{\perp}) \triangleq \bar{g}_{\perp}^T \bar{v}_{\perp} + \frac{1}{2} \gamma_k \|\bar{v}_{\perp}\|_2^2.$$

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 [3, 2]. Specifically, letting $\lambda_i \triangleq \hat{\lambda}_i + \gamma_k$, the solution to (17) is given coordinate-wise by

$$(19) \quad [\bar{v}_{\parallel}^*]_i = \begin{cases} -\frac{[\bar{g}_{\parallel}]_i}{\lambda_i} & \text{if } \left| \frac{[\bar{g}_{\parallel}]_i}{\lambda_i} \right| \leq \Delta_k \text{ and } \lambda_i > 0, \\ c & \text{if } [\bar{g}_{\parallel}]_i = 0 \text{ and } \lambda_i = 0, \\ -\operatorname{sgn}([\bar{g}_{\parallel}]_i) \Delta_k & \text{if } [\bar{g}_{\parallel}]_i \neq 0 \text{ and } \lambda_i = 0, \\ \pm \Delta_k & \text{if } [\bar{g}_{\parallel}]_i = 0 \text{ and } \lambda_i < 0, \\ -\frac{\Delta_k}{|[\bar{g}_{\parallel}]_i|} [\bar{g}_{\parallel}]_i & \text{otherwise,} \end{cases},$$

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

$$(20) \quad \vec{v}_\perp^* = \beta \vec{g}_\perp,$$

where

$$(21) \quad \beta = \begin{cases} -\frac{1}{\gamma_k} & \text{if } \gamma_k > 0 \text{ and } \|\vec{g}_\perp\|_2 \leq \Delta_k |\gamma_k|, \\ -\frac{\Delta_k}{\|\vec{g}_\perp\|_2} & \text{otherwise.} \end{cases}$$

Note that the solution to (14) is then

$$(22) \quad \vec{p}^* = \vec{P} \vec{v}^* = \vec{P}_\parallel \vec{v}_\parallel^* + \vec{P}_\perp \vec{v}_\perp^* = \vec{P}_\parallel \vec{v}_\parallel^* + \beta \vec{P}_\perp \vec{g}_\perp = \vec{P}_\parallel \vec{v}_\parallel^* + \beta \vec{P}_\perp \vec{P}_\perp^T \vec{g}_k,$$

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

$$(23) \quad p^* = \beta g + P_\parallel (v_\parallel^* - \beta g_\parallel).$$

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 \hat{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 and \hat{B}_0 to denote the proposed dense initialization. Similarly, $\{B_k\}$ and $\{\hat{B}_k\}$ will be used to denote the sequences of matrices obtained using the initializations B_0 and \hat{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

$$\vec{B}_0 = \gamma_k P P^T = \gamma_k \vec{P}_\parallel \vec{P}_\parallel^T + \gamma_k \vec{P}_\perp \vec{P}_\perp^T.$$

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

$$(24) \quad \hat{B}_0 = \gamma_k \vec{P}_\parallel \vec{P}_\parallel^T + \gamma_k^\perp \vec{P}_\perp \vec{P}_\perp^T,$$

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

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

$$(25) \quad \hat{B}_k = \begin{bmatrix} \vec{P}_\parallel & \vec{P}_\perp \end{bmatrix} \begin{bmatrix} \hat{\Lambda} + \gamma_k \vec{I}_r & \\ & \gamma_k^\perp \vec{I}_{n-r} \end{bmatrix} \begin{bmatrix} \vec{P}_\parallel & \vec{P}_\perp \end{bmatrix}^T,$$

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 $\text{Range}(\Psi_k)$, where Ψ_k is defined in (7). From (8), $\text{Range}(\bar{\Psi}_k) = \text{Range}(\bar{P}_\parallel)$; thus, $P_\parallel P_\parallel^T S_k = S_k$ and $P_\perp^T S_k = 0$. This gives that

$$(26) \quad \hat{B}_0 \bar{S}_k = \gamma_k \bar{P}_\parallel \bar{P}_\parallel^T \bar{S}_k + \gamma_k^\perp \bar{P}_\perp \bar{P}_\perp^T \bar{S}_k = \gamma_k \bar{S}_k = \bar{B}_0 \bar{S}_k.$$

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

$$\begin{aligned} \hat{B}_k &= \hat{B}_0 - \begin{bmatrix} \hat{B}_0 \bar{S}_k & \bar{Y}_k \end{bmatrix} \begin{bmatrix} \bar{S}_k^T \hat{B}_0 \bar{S}_k & \bar{L}_k \\ \bar{L}_k^T & -\bar{D}_k \end{bmatrix}^{-1} \begin{bmatrix} \bar{S}_k^T \hat{B}_0 \\ \bar{Y}_k^T \end{bmatrix} \\ &= \hat{B}_0 - \begin{bmatrix} \bar{B}_0 \bar{S}_k & \bar{Y}_k \end{bmatrix} \begin{bmatrix} \bar{S}_k^T \bar{B}_0 \bar{S}_k & \bar{L}_k \\ \bar{L}_k^T & -\bar{D}_k \end{bmatrix}^{-1} \begin{bmatrix} \bar{S}_k^T \bar{B}_0 \\ \bar{Y}_k^T \end{bmatrix} \\ &= \gamma_k \bar{P}_\parallel \bar{P}_\parallel^T + \gamma_k^\perp \bar{P}_\perp \bar{P}_\perp^T + \bar{P}_\parallel \hat{\Lambda} \bar{P}_\parallel^T \\ &= \bar{P}_\parallel \left(\hat{\Lambda} + \gamma_k \bar{I}_r \right) \bar{P}_\parallel^T + \gamma_k^\perp \bar{P}_\perp \bar{P}_\perp^T, \end{aligned}$$

which is equivalent to (25). \square

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 \hat{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 \hat{B}_k and B_k . The only difference in the eigendecompositions of \hat{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 1. *Suppose B_k is a BFGS matrix initialized with $B_0 = \gamma_k I$ and \hat{B}_k is a BFGS matrix initialized with (24). Then B_k and \hat{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 $\hat{\Lambda} = \text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_r)$.*

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

The results of Theorem 3.1 and Corollary 3.2 may seem surprising at first since every term in the compact representation ((6) and (7)) depends on the initialization; moreover, \hat{B}_0 is, generally speaking, a dense matrix while B_0 is a diagonal matrix. However, viewed from the perspective of (24), the parameter γ_k^\perp only plays a role in scaling the subspace spanned by the columns of P_\perp .

The initialization \hat{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 \hat{B}_0 provides the same subproblem separability as B_0 does in the case of the shape-changing infinity norm.

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

$$\begin{aligned} Q(\bar{P}\bar{v}) &= g_k^T P v + \frac{1}{2} v^T P^T \hat{B}_k P v \\ &= \bar{g}_\parallel^T \bar{v}_\parallel + \frac{1}{2} \bar{v}_\parallel^T \left(\hat{\Lambda} + \gamma_k \bar{I}_r \right) \bar{v}_\parallel + \bar{g}_\perp^T \bar{v}_\perp + \frac{1}{2} \gamma_k^\perp \|\bar{v}_\perp\|_2^2. \end{aligned}$$

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

$$(27) \quad \underset{\|\bar{v}_\perp\|_2 \leq \Delta_k}{\text{minimize}} \quad q_\perp(\bar{v}_\perp) \triangleq \bar{g}_\perp^T \bar{v}_\perp + \frac{1}{2} \gamma_k^\perp \|\bar{v}_\perp\|_2^2.$$

The solution to (27) is now given by

$$(28) \quad \bar{v}_\perp^* = \hat{\beta} \bar{g}_\perp,$$

where

$$(29) \quad \hat{\beta} = \begin{cases} -\frac{1}{\gamma_k^\perp} & \text{if } \gamma_k^\perp > 0 \text{ and } \|\vec{g}_\perp\|_2 \leq \Delta_k |\gamma_k^\perp|, \\ -\frac{\Delta_k}{\|\vec{g}_\perp\|_2} & \text{otherwise.} \end{cases}$$

Thus, the solution p^* can be expressed as

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

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

3.3. Determining the parameter γ_k^\perp . The values γ_k and γ_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 choices for γ_k^\perp . Note that in the case that $\gamma_k^\perp > 0$ and $\|\vec{g}_\perp\|_2 \leq \Delta_k |\gamma_k^\perp|$, the parameter γ_k^\perp scales the solution v_\perp^* (see 29); large values of γ_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 γ_k^\perp to be large. However, the larger that γ_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 γ_k^\perp becomes irrelevant. Moreover, for values $\gamma_k^\perp < 0$, \hat{B}_k is not guaranteed to be positive definite. For these reasons, we suggest sufficiently large and positive values for γ_k^\perp related to the curvature information gathered in $\gamma_1, \dots, \gamma_k$. Specific choices for γ_k^\perp are presented in the numerical results section.

3.4. 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., $B_0 = \gamma_0^\perp I$.

The only difference between Algorithm 1 and the proposed 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 2. *Let $f : R^n \rightarrow R$ be twice-continuously differentiable and bounded below on R^n . Suppose that there exists a scalar $c_1 > 0$ such that*

$$(31) \quad \|\nabla^2 f(x)\| \leq c_1, \quad \forall x \in R^n.$$

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

$$(32) \quad \gamma_k, \gamma_k^\perp \in (0, c_2], \quad \forall k \geq 0,$$

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

$$(33) \quad s_j^T y_j \geq c_3 \|s_j\| \|y_j\|$$

holds for each quasi-Newton pair $\{s_j, y_j\}$. \hat{B}_k by formula Algorithm 1(L213'). Then, if the stopping criteria is suppressed, the infinite sequence $\{x_k\}$ generated by Algorithm 1 satisfies

$$(34) \quad \lim_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0.$$

Proof. From (32), we have $\|\hat{B}_0\| \leq c_2$, which holds for each $k \geq 0$. Then, by [4, Lemma 3], there exists $c_4 > 0$ such that

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

Then, (34) follows from [4, Theorem 1]. \square

Require: $x_0 \in R^n$, $\Delta_0 > 0$, $\epsilon > 0$, $\gamma_0^\perp > 0$, $0 \leq \tau_1 < \tau_2 < 0.5 < \tau_3 < 1$,
 $0 < \eta_1 < \eta_2 \leq 0.5 < \eta_3 < 1 < \eta_4$
 Compute g_0
for $k = 0, 1, 2, \dots$ **do**
 if $\|g_k\| \leq \epsilon$ **then**
 return
 end if
 Choose at most m pairs $\{s_j, y_j\}$
 Compute $\Psi_k, R^{-1}, M_k, W, \hat{\Lambda}$ and Λ as described in Section 2
 Compute p^* for \hat{B}_k using (22), where β is computed using (29)
 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 $g_{k+1}, s_k, y_k, \gamma_k$ and γ_k^\perp
 else
 $x_{k+1} = x_k$
 end if
 if $\rho_k < \tau_2$ **then**
 $\Delta_{k+1} = \min(\eta_1 \Delta_k, \eta_2 \|s_k\|_{P,\infty})$
 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

ALGORITHM 1: An L-BFGS trust-region method with dense initialization

In the following section, we consider γ_k^\perp parameterized by two scalars, c and λ :

$$(35) \quad \gamma_k^\perp(c, \lambda) = \lambda c \gamma_k^{\max} + (1 - \lambda) \gamma_k,$$

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 γ_k^\perp will be further discussed in Section 4.) We now show that Algorithm 1 converges for these choices of γ_k^\perp . Assuming that (31) and (33) hold, it remains to show that (32) holds for these choices of γ_k^\perp . To see that (32) 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 (32) holds. Thus, Algorithm 1 converges for these choices for γ_k^\perp .

3.5. 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*),

$$(36) \quad p_u^* = -\hat{B}_k^{-1} g_k$$

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 (36). However, products with \hat{B}_k^{-1} can be performed using the compact

representation without involving a partial eigendecomposition, i.e.,

$$(37) \quad \hat{B}_k^{-1} = \frac{1}{\gamma_k^\perp} I + V_k \hat{M}_k V_k^T,$$

where $V_k = [S_k \ Y_k]$,

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

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

$$(38) \quad \|p_u^*\|_2 \leq \Delta_k$$

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

$$(39) \quad \|p_u^*\|_2^2 = g_k^T \hat{B}_k^{-2} g_k = \gamma_k^{-2} \|g_k\|^2 + 2\gamma_k^{-1} u_k^T \hat{M}_k u_k + u_k^T \hat{M}_k (V_k^T V_k) \hat{M}_k u_k.$$

Here, as in the LMTR algorithm, the vector $u_k = V_k^T g_k$ is computed at each iteration when updating the matrix $V_k^T V_k$. Thus, the computational cost of $\|p_u^*\|_2$ is low because the matrices $V_k^T V_k$ and \hat{M}_k are small in size. The norm equivalence for the shape-changing infinity norm studied in [3] guarantees that (38) 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 (38) 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 (38) 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 Section 3.2.

4. NUMERICAL EXPERIMENTS

We perform numerical experiments on 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

$$\|\vec{g}_k\|_2 \leq \epsilon \max(1, \|\vec{x}_k\|_2),$$

with $\epsilon = 10^{-10}$. The initial step, p_0 , was determined by a backtracking line-search along the normalized steepest descent direction. The rank of $\vec{\Psi}_k$ was estimated by the number of positive diagonal elements in the diagonal matrix of the LDL^T decomposition (or eigendecomposition of $\vec{\Psi}_k^T \vec{\Psi}_k$) that are larger than the threshold $\epsilon_r = (10^{-7})^2$. (Note that the columns of Ψ_k are normalized.)

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

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

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:

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

TABLE 1. Values for γ_k^\perp used in Experiment 1.

- (1) A comparison of results for different values of $\gamma_k^\perp(c, \lambda)$.
- (2) Two versions of computing full quasi-Newton trial step (see Section 3.5) are compared. One version uses the dense initialization to compute p_u^* as described in Section 3.5; the other uses the conventional initialization, i.e., p_u^* is computed as $p_u^* = B_k^{-1}g_k$. In the both cases, the dense initialization is used for computing trial steps obtained from explicitly solving the trust-region subproblem (Section 3.2) when the full quasi-Newton trial step is not accepted.
- (3) A comparison of alternative ways of computing the partial eigendecomposition (Section 2.2), namely, those based on QR and SVD factorizations.
- (4) A comparison of LMTR together with a dense initialization and the line search L-BFGS method with the conventional initialization.
- (5) 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.
- (6) A comparison of the dense and conventional initializations.

In the experiments below, the dense initial matrix \hat{B}_0 corresponding to $\gamma_k^\perp(c, \lambda)$ given in (35) will be denoted by

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

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

Experiment 1. In this experiment, we consider various scalings of a proposed γ_k^\perp using LMTR. As argued in Section 3.3, it is reasonable to choose γ_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 (35). These choices correspond to conservative strategies for computing steps in the space spanned by \vec{P}_\perp (see the discussion in Section 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 \hat{B}_0 using (35). 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 γ_k^\perp (see (19)); in contrast, in the orthogonal subspace, $\|v_\perp^*\|$ inversely depends on γ_k^\perp (see (28) and (29)).

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, \hat{B}_k with the dense initialization was used for the shape-changing component of the algorithm with γ_k^\perp defined as in (35). The values of c and λ chosen for Experiment 1 are found in Table 4. (See Section 3.5 for details on the LMTR algorithm.)

Figure 1 displays the performance profiles using the chosen values of c and λ to define γ_k^\perp in the case when the dense initialization was used for both the computation of the unconstrained minimizer p_u^* as well as for the shape-changing component of the algorithm, which is denoted in the legend of plots in Figure 1 by the use of an asterisk (*). The results of Figure 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 $\lambda = \frac{1}{2}$.

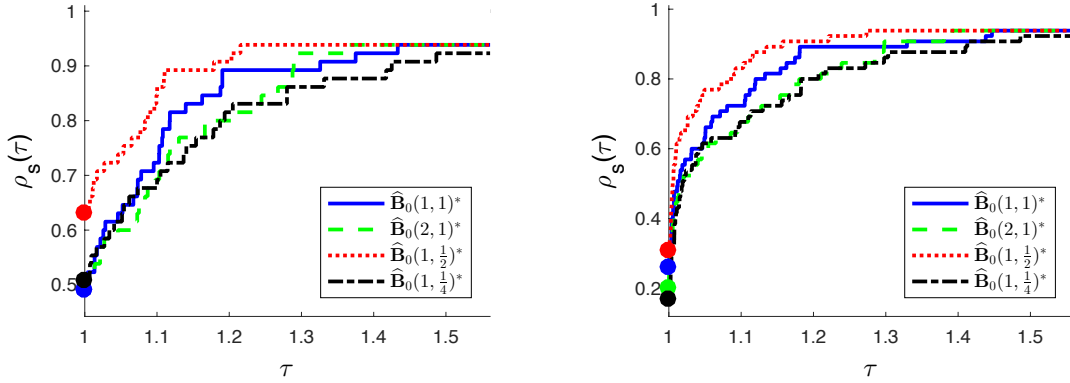


FIGURE 1. Performance profiles comparing *iter* (left) and *time* (right) for the different values of γ_k^\perp given in Table 4. In the legend, $\hat{B}_0(c, \lambda)$ denotes the results from using the dense initialization with the given values for c and λ to define γ_k^\perp . In this experiment, the dense initialization was used for all aspects of the algorithm.

Figure 2 reports the performance profiles for using the chosen values of c and λ to define γ_k^\perp in the case when the dense initialization was only used for the shape-changing component of the LMTR algorithm—denoted in the legend of plots in Figure 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 γ_k^\perp in terms of both then number of iterations and the total computational time. Based on the results in Figure 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 γ_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 γ_k^\perp . For this reason, it is not surprising that the performance profiles in Figure 2 for the different values of γ_k^\perp are more indistinguishable than those in Figure 1.

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

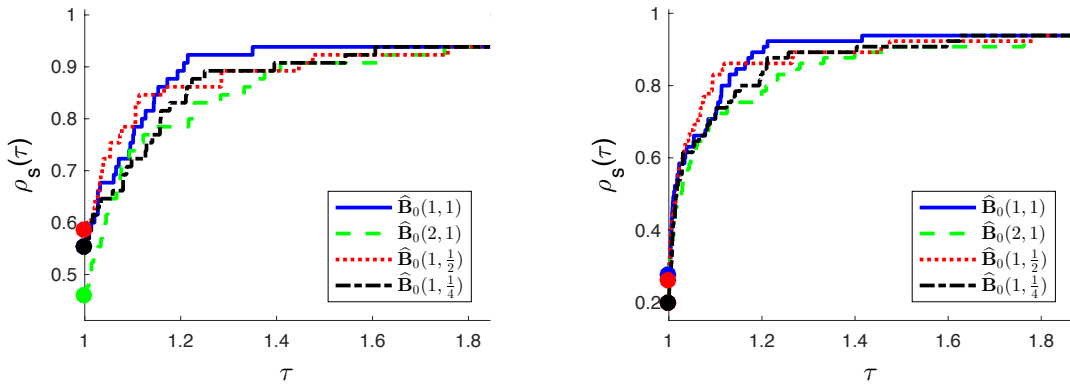


FIGURE 2. Performance profiles comparing *iter* (left) and *time* (right) for the different values of γ_k^\perp given in Table 4. In the legend, $\hat{B}_0(c, \lambda)$ denotes the results from using the dense initialization with the given values for c and λ to define γ_k^\perp . In this experiment, the dense initialization was only used for the shape-changing component of the algorithm.

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 shape-changing component of algorithm. 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 shape-changing component of the algorithm, i.e., $(\gamma_k^\perp(1, 1))$.

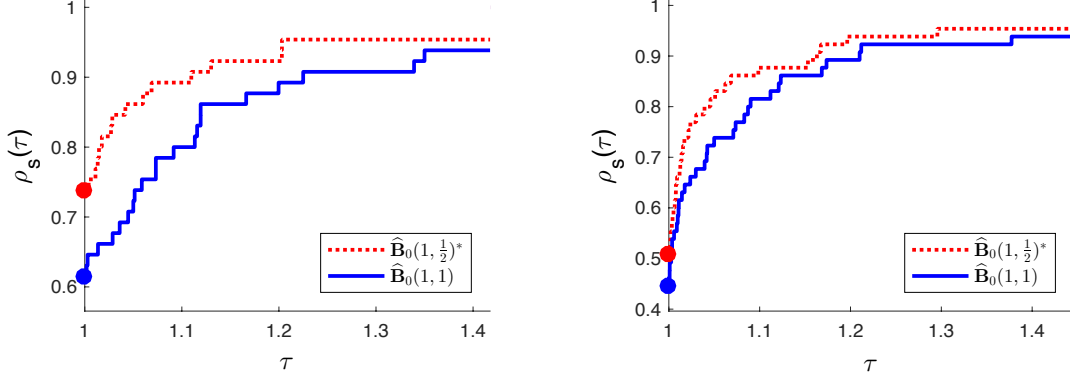


FIGURE 3. Performance profiles of `iter` (left) and `time` (right) for Experiment 2. In the legend, the asterisk after $\hat{B}_0(1, \frac{1}{2})^*$ signifies that the dense initialization was used for all aspects of the LMTR algorithm; without the asterisk, $\hat{B}_0(1, 1)$ signifies the test where the dense initialization is used only for the shape-changing component of the algorithm.

The performance profiles comparing the results of this experiment are presented in Figure 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 shape-changing component of the algorithm. 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. As noted in Section 2.2, a partial SVD may be used in place of a partial QR decomposition to derive alternative formulas for computing products with P_\parallel . Specifically, if the SVD of $\Psi_k^T \Psi_k$ is given by $\tilde{U} \tilde{\Sigma}^2 \tilde{U}^T$ and the SVD of $\tilde{\Sigma} \tilde{U}^T \tilde{M}_k^{-1} \tilde{U} \tilde{\Sigma}$ is given by $\tilde{G} \tilde{\Lambda} \tilde{G}^T$, then P_\parallel can be written as follows:

$$(40) \quad \tilde{P}_\parallel = \tilde{\Psi}_k \tilde{U} \tilde{\Sigma}^{-1} \tilde{G}.$$

Alternatively, in [11], P_\parallel is written as

$$(41) \quad P_\parallel = \tilde{\Psi}_k \tilde{M}_k^{-1} \tilde{U} \tilde{\Sigma} \tilde{G} \tilde{\Lambda}^{-1}.$$

Note that both of the required SVD computations for this approach involve $r \times r$ matrices, where $r \leq 2m \ll n$.

For this experiment, we consider LMTR with the dense initialization $\hat{B}_0(1, \frac{1}{2})^*$ used for all aspects of the algorithm (i.e., the top performer in Experiment 2). We compare an implementation of this method using the QR decomposition to compute products with P_\parallel to the two SVD-based methods. The results of this experiment given in Figure 4 suggest that the QR decomposition outperforms the two SVD decompositions in terms of both the number of iterations and time. (Note that the QR factorization was used for both Experiments 1 and 2.)

Experiment 4. 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. [14]. 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 [14]:

$$\|\tilde{g}_k\|_\infty \leq \epsilon.$$

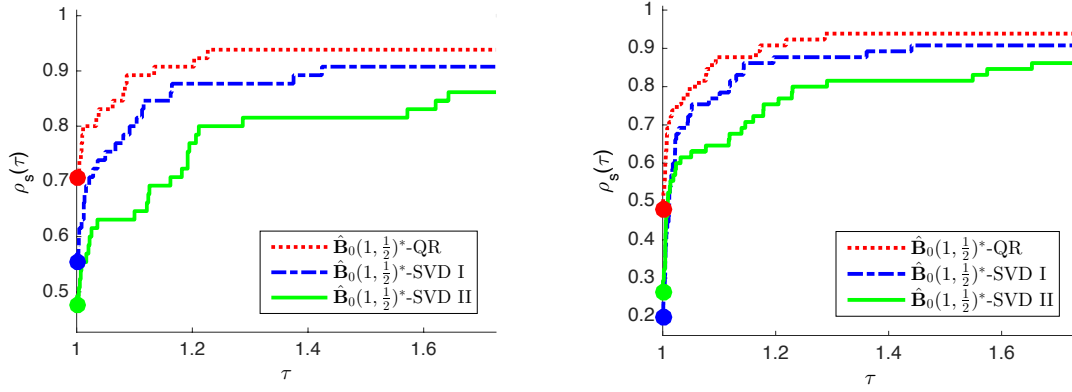


FIGURE 4. Performance profiles of `iter` (left) and `time` (right) for Experiment 3 comparing three formulas for computing products with \vec{P}_{\parallel} . In the legend, "QR" denotes results using (8), "SVD I" denotes results using (40), and "SVD II" denotes results using (41). These results used the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$.

The dense initialization was used for all aspects of LMTR.

The performance profiles for this experiment is given in Figure 5. On this test set, the dense initialization outperforms L-BFGS-B in terms of both the number of iterations and the total computational time.

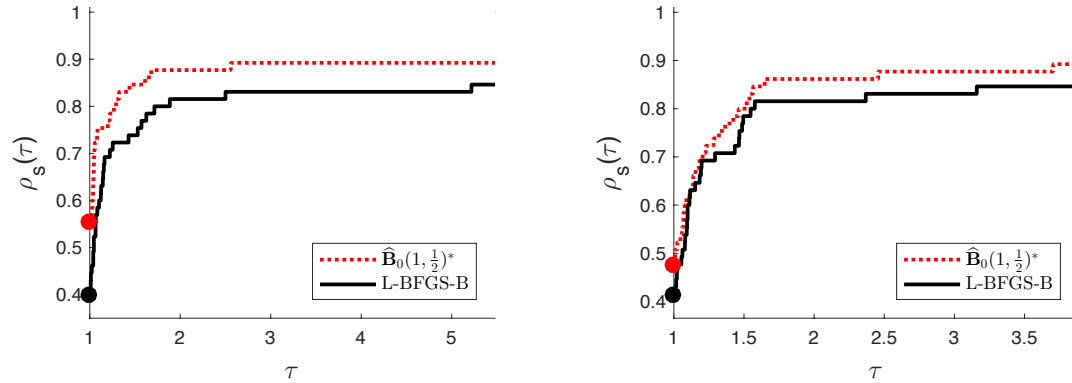


FIGURE 5. Performance profiles of `iter` (left) and `time` (right) for Experiment 4 comparing LMTR with the dense initialization with $\gamma_k^{\perp}(1, \frac{1}{2})$ to L-BFGS-B.

Experiment 5. 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].)

Figure 6 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 6 (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, Figure 6 (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

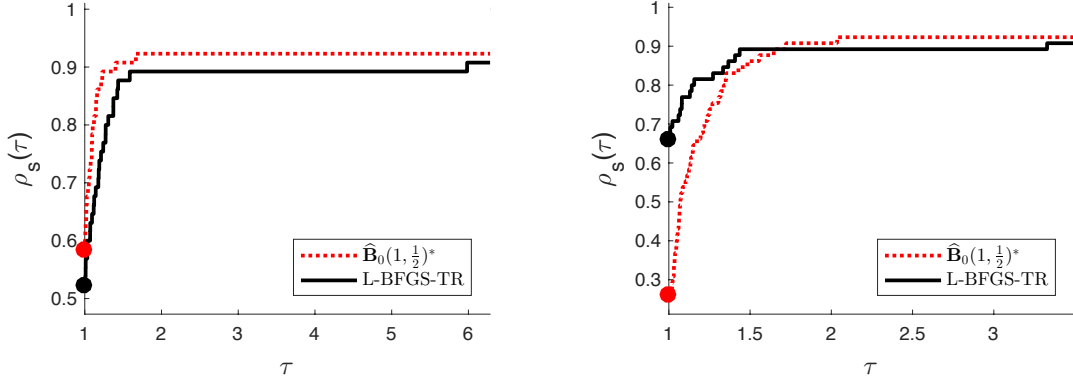


FIGURE 6. Performance profiles of `iter` (left) and `time` (right) for Experiment 5 comparing LMTR with the dense initialization with $\gamma_k^\perp(1, \frac{1}{2})$ to L-BFGS-TR.

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 Figure 7. On this smaller test set, LMTR outperforms L-BFGS-TR both in terms of total number of iterations and computational time.

Finally, Figures 6 and 7 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, Figure 7 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.

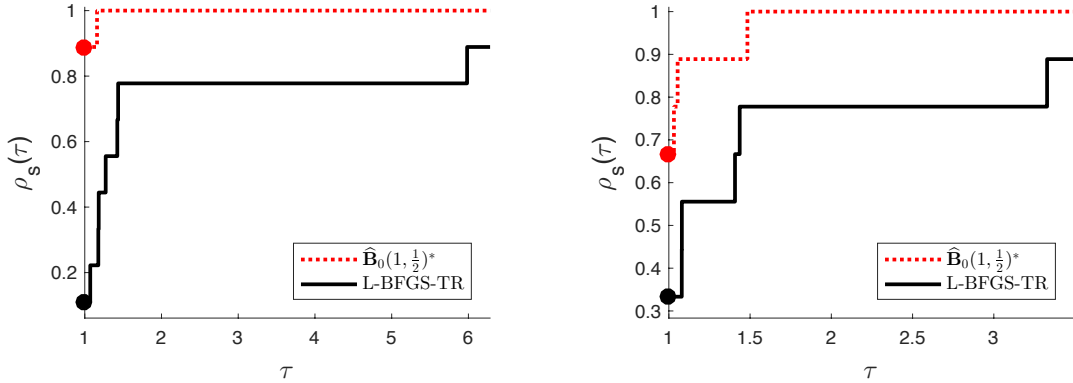


FIGURE 7. Performance profiles of `iter` (left) and `time` (right) for Experiment 5 comparing LMTR with the dense initialization with $\gamma_k^\perp(1, \frac{1}{2})$ to L-BFGS-TR on the subset of 14 problems for which L-BFGS-TR implements a line search more than 30% of the iterations.

Experiment 6. 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., $\gamma_k^\perp(1, \frac{1}{2})$), and the QR factorization was used to compute products with P_\parallel .

From Figure 8, 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 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

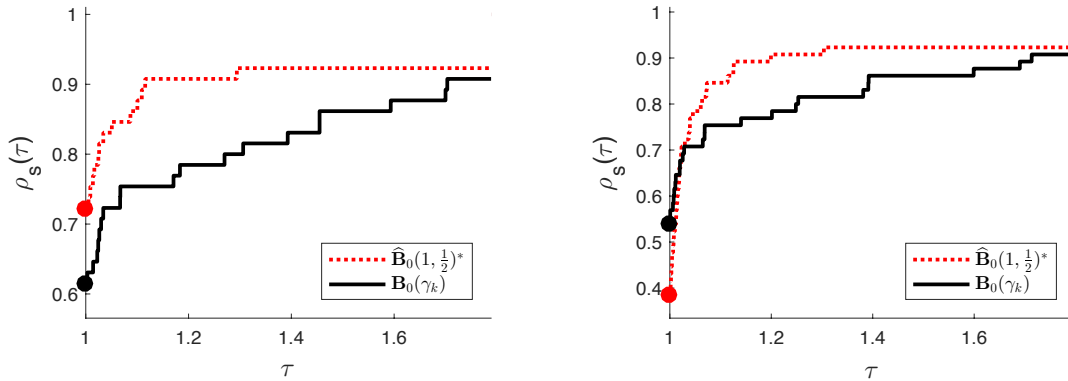


FIGURE 8. Performance profiles of `iter` (left) and `time` (right) for Experiment 6 comparing LMTR with the dense initialization with $\gamma_k^\perp(1, \frac{1}{2})$ to LMTR with the conventional initialization.

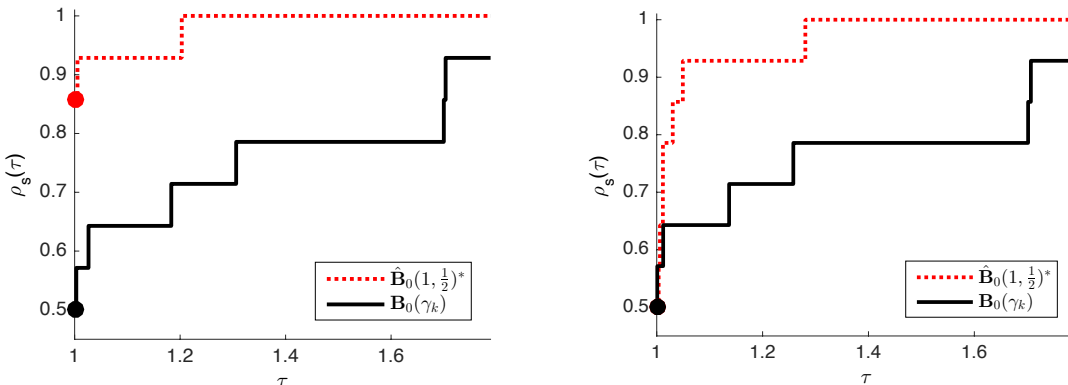


FIGURE 9. Performance profiles of `iter` (left) and `time` (right) for Experiment 6 comparing LMTR with the dense initialization with $\gamma_k^\perp(1, \frac{1}{2})$ to LMTR with the conventional initialization on the subset of 14 problems in which the unconstrained minimizer is rejected at 30% of the iterations.

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 Figure 9. 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. It should 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 eigendecomposition; 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] S. Becker. LBFGSB (L-BFGS-B) mex wrapper. <https://www.mathworks.com/matlabcentral/fileexchange/35104-lbfgsb-l-bfgs-b-mex-wrapper>, 2012–2015.
- [2] J. Brust, O. Burdakov, J. B. Erway, R. F. Marcia, and Y. Ya-xiang. Shape-changing L-SR1 trust-region methods. Technical Report 2016-2, Wake Forest University, 2016.
- [3] O. Burdakov, L. Gong, Y.-X. Yuan, and S. Zikrin. On efficiently combining limited memory and trust-region techniques. *Mathematical Programming Computation*, 9:101–134, 2016.
- [4] J. V. Burke, A. Wiegmann, and L. Xu. Limited memory BFGS updating in a trust-region framework. Technical Report, University of Washington, 1996.
- [5] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited-memory methods. *Math. Program.*, 63:129–156, 1994.
- [6] O. DeGuchy, J. B. Erway, and R. F. Marcia. Compact representation of the full Broyden class of quasi-Newton updates. Technical Report 2016-4, Wake Forest University, 2016.
- [7] E. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.
- [8] J. B. Erway and R. F. Marcia. On efficiently computing the eigenvalues of limited-memory quasi-Newton matrices. *SIAM Journal on Matrix Analysis and Applications*, 36(3):1338–1359, 2015.
- [9] J. B. Erway and R. F. Marcia. On solving large-scale limited-memory quasi-Newton equations. *Linear Algebra and its Applications*, 515:196–225, 2017.
- [10] N. I. M. Gould, D. Orban, and P. L. Toint. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Trans. Math. Software*, 29(4):373–394, 2003.
- [11] X. Lu. *A study of the limited memory SR1 method in practice*. PhD thesis, University of Colorado, 1992.
- [12] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.
- [13] D. F. Shanno and K. H. Phua. Matrix conditioning and nonlinear optimization. *Mathematical Programming*, 14(1):149–160, 1978.
- [14] C. Zhu, R. Byrd, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23:550–560, 1997.

E-mail address: jbrust@ucmerced.edu

APPLIED MATHEMATICS, UNIVERSITY OF CALIFORNIA, MERCED, MERCED, CA 95343

E-mail address: oleg.burdakov@liu.se

DEPARTMENT OF MATHEMATICS, LINKÖPING UNIVERSITY, SE-581 83 LINKÖPING, SWEDEN

E-mail address: erwayjb@wfu.edu

DEPARTMENT OF MATHEMATICS, WAKE FOREST UNIVERSITY, WINSTON-SALEM, NC 27109

E-mail address: rmarcia@ucmerced.edu

APPLIED MATHEMATICS, UNIVERSITY OF CALIFORNIA, MERCED, MERCED, CA 95343