A Limited-Memory Reduced Hessian Method for Bound-Constrained Optimization

Michael W. Ferry* Philip E. Gill† Elizabeth Wong† Minxin Zhang†

UCSD Center for Computational Mathematics
Technical Report CCoM-21-01
May 25, 2021

Abstract

Quasi-Newton methods for unconstrained optimization accumulate approximate curvature in a sequence of expanding subspaces. This allows an approximate Hessian to be represented using a smaller reduced Hessian matrix that increases in dimension at each iteration. When the number of variables is large, this feature may be used to define limited-memory reduced-Hessian (L-RH) methods in which the dimension of the reduced Hessian is limited to save storage. In this paper a limited-memory reduced-Hessian method is proposed for the solution of large-scale optimization problems with upper and lower bounds on the variables. The method uses a projected-search method to identify the variables on their bounds at a solution. Conventional projected-search methods are restricted to use an Armijo-like line search. However, by modifying the line-search conditions, a new projected line search based on the Wolfe conditions is used that retains many of the benefits of a Wolfe line search in the unconstrained case. Numerical results are presented for the software package L-RH-B, which implements a limited-memory reduced-Hessian method based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) approximate Hessian. It is shown that L-RH-B is competitive with the code L-BFGS-B on the unconstrained and bound-constrained problems in the CUTEst test collection.

Key words. bound-constrained optimization, quasi-Newton methods, BFGS method, reduced-Hessian methods, projected-search methods

*NVIDIA Corporation, Hillsboro, Oregon (michael@mwferry.com).

†Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (pgill@ucsd.edu, elwong@ucsd.edu, miz151@ucsd.edu) Research supported in part by National Science Foundation grants DMS-0915220 and DMS-1318480. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

1. Introduction

A bound-constrained optimization problem may be written in the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x) \quad \text{subject to} \quad \ell \leq x \leq u, \tag{BC}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a twice-differentiable function, and ℓ and u are n-vectors of lower and upper bounds on the variables such that $\ell_j \leq u_j$. The first-order optimality conditions for problem (BC) at $x^* \in \Omega$ are

$$x^* \in \Omega, \;\; \text{with} \;\; \nabla_i f(x^*) \;\; \begin{cases} \geq 0 & \text{if } x_i^* = \ell_i, \\ = 0 & \text{if } \ell_i < x_i^* < u_i, \\ \leq 0 & \text{if } x_i^* = u_i. \end{cases}$$

where $\nabla_i f(x)$ denotes the ith component of the gradient of f. These conditions impose sign conditions on the gradient at components of x^* associated with the active set $\mathcal{A}(x^*)$, where the active set is the set of indices of the variables that lie on their bounds, i.e., $\mathcal{A}(x) = \{\, i : x_i = \ell_i \;\text{ or }\; x_i = u_i \,\}$.

Projected-search line-search methods for problem (BC) generate a sequence of feasible iterates $\{\, x_k \,\}_{k=0}^{\infty}$ such that $x_{k+1} = \mathbf{proj}_{\Omega}(x_k + \alpha_k p_k)$, where α_k is a positive step length, p_k is a search direction, and $\mathbf{proj}_{\Omega}(x)$ is the projection of x onto the feasible region, i.e.,

$$\big[\, \mathbf{proj}_{\Omega}(x) \,\big]_i = \begin{cases} \ell_i & \text{if } \; x_i < \ell_i, \\ u_i & \text{if } \; x_i > u_i, \\ x_i & \text{otherwise.} \end{cases}$$

A potential benefit of a projected-search method is that many changes to the active set can be made at the cost of computing a single search direction. The projected-search methods of Goldstein [14], Levitin and Polyak [19], and Bertsekas [1] are based on using the steepest-descent direction $p_k = -\nabla f(x_k)$. Bertsekas [3] and Calamai and Moré [5] propose methods that identify the optimal active set using a projected-search method and then switch to a Newton method. Projected-search methods based computing p_k using a quasi-Newton method are proposed by Ni and Yuan [21], and Kim, Sra and Dhillon [16].

In this paper, we propose a quasi-Newton projected-search method L-RH-B, which is an extension of the limited-memory reduced-Hessian method of Leonard [18] and Gill and Leonard [13]. The method is based on the work of Fenelon [9] and Siegel [22], who independently proposed methods that exploit the fact that quasi-Newton methods accumulate approximate curvature in a sequence of expanding subspaces. In particular, Fenelon considered a method in which the search direction is computed using a reduced matrix that represents the approximate Hessian in the subspace. Though the subspace and this reduced matrix increase in dimension at each iteration, the dimension is limited to some fixed number and only the most recent information is used to define the subspace and matrix (similar to limited-memory BFGS methods). As the objective function is not differentiable along the piecewise-linear path, it is not possible to use a line-search based on satisfying the

Wolfe conditions, which involve the derivatives at two points on the search path. This means that the step must be computed using a simpler backtracking method. Methods for conventional unconstrained minimization that use the Wolfe conditions are more reliable and efficient than methods based on simple backtracking. For example, if the search direction is generated using a quasi-Newton method, the Wolfe conditions impose a restriction on the directional derivative that guarantees the satisfaction of a necessary condition for the quasi-Newton update to give a positive-definite approximate Hessian. The method L-RH-B employs a new *quasi-Wolfe* line search that is appropriate for piecewise differentiable functions (see Ferry et al. [11]). The behavior of the line search is similar to that of a conventional Wolfe line search, except that a step is accepted under a wider range of conditions. These conditions take into consideration steps at which the restriction of the objective function on the search path is not differentiable.

The paper is organized in six sections. In Section 2, we briefly review the methods of Gill and Leonard for unconstrained optimization. In Section 3, the L-RH-B algorithm for problem (BC) is described. The projected line-search method is introduced in Section 4. Section 5 describes the matrix factors and updates required by the method. Numerical results for L-RH-B are presented in Section 7.

Notation. Given vectors x and y, the vector consisting of x augmented by y is denoted by (x, y). The subscript i is appended to vectors to denote the ith component of that vector, whereas the subscript k is appended to a vector to denote its value during the kth iteration of an algorithm, e.g., x_k represents the value for x during the kth iteration, whereas $\left[\, x_k \,\right]_i$ denotes the ith component of the vector x_k. The ith component of the gradient of the scalar-valued function f is denoted by $\nabla_i f(x)$. Given vectors a and b with the same dimension, vectors with ith component $a_i b_i$ and a_i/b_i are denoted by $a \cdot b$ and $a \cdot/ b$ respectively. Similarly, $\min(a, b)$ is a vector with components $\min(a_i, b_i)$. The vector e denotes the column vector of ones, and I denotes the identity matrix. The dimensions of e and I are defined by the context. The vector two-norm or its induced matrix norm are denoted by $\|\cdot\|$. The orthogonal complement of a given $\mathcal{S} \subset \mathbb{R}^n$ is denoted by $\mathcal{S}^\perp$.

2. Background

In this section, we give a brief review of the limited-memory reduced-Hessian method L-RHR for the unconstrained minimization of the twice continuously differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$. For more details, see Gill and Leonard [13]. A conventional quasi-Newton method generates a sequence of iterates $\{\, x_k \,\}$ such that $x_{k+1} = x_k + \alpha_k p_k$, where p_k is a descent direction and α_k is a scalar step chosen to enforce a sufficient reduction in f at each iteration. The search direction satisfies $H_k p_k = -\nabla f(x_k)$, where H_k is a positive-definite approximation to the Hessian matrix of f. Given H_k, the BFGS update gives the next approximate Hessian H_{k+1} as

$$H_{k+1} = H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \frac{1}{w_k^T d_k} w_k w_k^T, \tag{2.1}$$

where $d_k = x_{k+1} - x_k$, $w_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, and $w_k^T d_k$ approximates the curvature of f along p_k. An important property of the BFGS update is that $d_k^T H_{k+1} d_k = w_k^T d_k$, so that the curvature along d_k of a quadratic model with Hessian H_{k+1} is equal to the approximate curvature $w_k^T d_k$. To ensure the approximate Hessian remains positive definite, the BFGS update is applied only when $w_k^T d_k > 0$. If f is bounded below, conditions may be imposed on α_k that not only guarantee a sufficient decrease in f, but also provides a vector d_k for which $w_k^T d_k$ is positive. The most common conditions are the Wolfe conditions

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \eta_A \alpha_k \nabla f(x_k)^{\mathrm{T}} p_k, \tag{2.2}$$

$$|\nabla f(x_k + \alpha_k p_k)^{\mathrm{T}} p_k| \leq \eta_W |\nabla f(x_k)^{\mathrm{T}} p_k|, \tag{2.3}$$

where η_A and η_W are fixed scalars such that $0 \leq \eta_A \leq \eta_W < 1$ and $\eta_A < \frac{1}{2}$.

The *reduced-Hessian method* of Gill and Leonard takes advantage of the implicit structure of the quasi-Newton Hessian to compute search directions from a smaller search space. The method is implemented in a limited-memory framework by limiting the number of basis vectors for the search space. The *gradient subspace* defined as span $\{ \nabla f(x_0), \nabla f(x_1), \ldots, \nabla f(x_k) \}$ and denoted by $\mathcal{G}_k$, with $\mathcal{G}_k^{\perp}$ denoting the orthogonal complement of $\mathcal{G}_k$ in $\mathbb{R}^n$. Reduced-Hessian methods are based on the following result (see, e.g., Fletcher and Powell [12], Fenelon [9], and Siegel [22]).

Lemma 2.1. *Consider the BFGS method applied to a general nonlinear function. If $H_0 = \sigma I$ ($\sigma > 0$) and $H_k p_k = -\nabla f(x_k)$, then $p_k \in \mathcal{G}_k$ for all k. Moreover, if $z \in \mathcal{G}_k$ and $y \in \mathcal{G}_k^{\perp}$, then $H_k z \in \mathcal{G}_k$ and $H_k y = \sigma y$.* ∎

If r_k denotes $\dim(\mathcal{G}_k)$, let Z_k be an $n \times r_k$ matrix whose columns form an orthonormal basis for $\mathcal{G}_k$. Given an $(n-r_k)\times n$ orthonormal basis Y_k for $\mathcal{G}_k^{\perp}$ the matrix $Q_k = \begin{pmatrix} Z_k & Y_k \end{pmatrix}$ defines an orthogonal transformation $x = Q_k x_Q$. The transformed gradient and approximate Hessian are then given by $Q_k^T \nabla f(x_k)$ and $Q_k^{\mathrm{T}} H_k Q_k$, respectively. If $H_0 = \sigma I$ ($\sigma > 0$), it follows from Lemma 2.1 that the transformation induces a block-diagonal structure, with

$$Q_k^{\mathrm{T}} H_k Q_k = \begin{pmatrix} Z_k^{\mathrm{T}} H_k Z_k & 0 \\ 0 & \sigma I_{n-r_k} \end{pmatrix} \quad \text{and} \quad Q_k^T \nabla f(x_k) = \begin{pmatrix} Z_k^T \nabla f(x_k) \\ 0 \end{pmatrix}. \tag{2.4}$$

The matrix $Z_k^{\mathrm{T}} H_k Z_k$ is positive-definite and is known as the approximate reduced Hessian (or just reduced Hessian). The vector $Z_k^{\mathrm{T}} \nabla f(x_k)$ is known as the reduced gradient. If the equation $H_k p_k = -\nabla f(x_k)$ for the search direction is written as $(Q_k^{\mathrm{T}} H_k Q_k) Q_k^T p_k = -Q_k^T \nabla f(x_k)$, then it follows from (2.4) that

$$p_k = Z_k q_k, \quad \text{where } q_k \text{ satisfies } \quad Z_k^T H_k Z_k q_k = -Z_k^T \nabla f(x_k). \tag{2.5}$$

The matrices Z_k and $Z_k^T H_k Z_k$ may be used to reconstruct H_k, which need not be stored explicitly. In particular, we have

$$\begin{aligned} H_k &= Q_k Q_k^T H_k Q_k Q_k^T \\ &= \begin{pmatrix} Z_k & Y_k \end{pmatrix} \begin{pmatrix} Z_k^{\mathrm{T}} H_k Z_k & 0 \\ 0 & \sigma I_{n-r_k} \end{pmatrix} \begin{pmatrix} Z_k^T \\ Y_k^T \end{pmatrix} \\ &= Z_k (Z_k^T H_k Z_k) Z_k^T + \sigma (I - Z_k Z_k^T). \end{aligned} \tag{2.6}$$

This expression implies that any vector y such that $Z_k^{\mathrm{T}} y = 0$ is an eigenvector of H_k with $H_k y = \sigma y$. If B_k is an $n \times r_k$ matrix with columns that form a basis for $\mathcal{G}_k$, an orthonormal basis Z_k can be defined in terms of the economy-size QR decomposition $B_k = Z_k T_k$, where T_k is a nonsingular $r_k \times r_k$ upper-triangular matrix. In practice, Z_k can be stored explicitly along with T_k, or implicitly by storing only B_k and T_k, with computations involving Z_k utilizing $Z_k = B_k T_k^{-1}$. If the Cholesky factorization $Z_k^T H_k Z_k = R_k^T R_k$ is known, q_k can be computed from the forward substitution $R_k^T d_k = -Z_k^T \nabla f(x_k)$ and back-substitution $R_k q_k = d_k$.

The dimension of $Z_k^T H_k Z_k$ is limited by discarding the oldest basis vector when the number of basis vectors exceeds some predefined limit m. Assume for the moment that the gradients in the sequence $\{ \nabla f(x_k) \}$ are linearly independent. Lemma 2.1 implies that the search direction p_k lies in $\mathcal{G}_k$ for all k. Siegel [22] proposed that a subset of $\{ p_k \}$ be used to form a basis for $\mathcal{G}_k$ instead of $\{ \nabla f(x_k) \}$ and showed that this modification endows the method with finite termination on a strictly convex quadratic function. Consider any iteration k such that $1 \leq k \leq m-1$. At the start of the iteration, the directions $p_0, \ldots, p_{k-1}$ are known, but p_k has yet to be computed from equations (2.5) that use Z_k. This implies that it is not possible to use p_k as part of B_k. Nevertheless, $\mathcal{G}_k$ is spanned by both the gradients and the search directions, which means that the latest gradient $\nabla f(x_k)$ can be used as a temporary basis vector until p_k has been computed, at which point it can be swapped with $\nabla f(x_k)$. The swap does not change Z_k, but the last column of T_k is replaced by the vector $q_k = Z_k^{\mathrm{T}} p_k$ found as part of the computation of p_k in (2.5). If $\nabla f(x_{k+1})$ is accepted after the line search, it is added to the basis and the QR factors are updated as in (2.7). This update expands the reduced Hessian by a row and column (see (2.8)), and the last diagonal is reinitialized with $\sigma_k = w_k^T w_k / w_k^T d_k$.

If $k \geq m-1$, the addition of $\nabla f(x_{k+1})$ gives a basis with $m+1$ columns and the oldest column p_{k-m+1} must be removed before starting iteration $k+1$. The factors Z_{k+1} and T_{k+1} associated with the next basis $B_{k+1} = \begin{pmatrix} p_{k-m+2} & \cdots & p_k & \nabla f(x_{k+1}) \end{pmatrix}$ are updated using two sets of plane rotations applied on the right of the orthogonal factor and left of the triangular factor of $\begin{pmatrix} p_{k-m+1} & \cdots & p_k & \nabla f(x_{k+1}) \end{pmatrix}$. Further details of the methods for updating the QR and Cholesky factors when a column is removed from the basis are given by Gill and Leonard [13].

During the k-th iteration of L-RHR, the number of columns in B_k (and Z_k) can either remain unchanged or increase by one, depending on whether or not the new gradient $\nabla f(x_{k+1})$ lies in $\mathcal{G}_k$. This is determined from the value of the scalar ρ_{k+1} such that $\rho_{k+1} = \|(I - Z_k Z_k^T) \nabla f(x_{k+1})\|$. If $\rho_{k+1} = 0$, then $\nabla f(x_{k+1}) \in \mathcal{G}_k$ and $\nabla f(x_{k+1})$ is said to be *rejected*. The matrix factors for the next iteration remain unchanged. Otherwise, $r_{k+1} = r_k + 1$ and $\nabla f(x_{k+1})$ is said to be *accepted*. In this case, B_k is augmented by a new column $\nabla f(x_{k+1})$, and the matrix factors of B_{k+1} are given by

$$B_{k+1} = \begin{pmatrix} B_k & \nabla f(x_{k+1}) \end{pmatrix} = \begin{pmatrix} Z_k & z_{k+1} \end{pmatrix} \begin{pmatrix} T_k & Z_k^T \nabla f(x_{k+1}) \\ 0 & \rho_{k+1} \end{pmatrix} = Z_{k+1} T_{k+1}, \quad (2.7)$$

where z_{k+1} is defined by the identity $\rho_{k+1} z_{k+1} = (I - Z_k Z_k^T) \nabla f(x_{k+1})$. Note that T_{k+1} is nonsingular as $\rho_{k+1} \neq 0$. The Cholesky factor R_k is updated by adding a row

and column to account for the new last column of Z_{k+1}. It follows from Lemmas 2.1 and (2.4) that

$$Z_{k+1}^T H_k Z_{k+1} = \begin{pmatrix} Z_k^T H_k Z_k & Z_k^T H_k z_{k+1} \\ z_{k+1}^T H_k Z_k & z_{k+1}^T H_k z_{k+1} \end{pmatrix} = \begin{pmatrix} Z_k^T H_k Z_k & 0 \\ 0 & \sigma \end{pmatrix}, \tag{2.8}$$

giving the expanded block-diagonal factor

$$R_k^{(1)} = \begin{pmatrix} R_k & 0 \\ 0 & \sigma^{1/2} \end{pmatrix}.$$

If $\nabla f(x_{k+1})$ is rejected, then $r_{k+1} = r_k$ and $R_k^{(1)} = R_k$.

In addition, the factor R_{k+1} is computed by modifying $R_k^{(1)}$ to reflect the rank-two BFGS update to $Z_{k+1}^T H_k Z_{k+1}$ resulting from the rank-two update to H_k defined in (2.1). Let $s = Z_{k+1}^T d_k$ and $y = Z_{k+1}^T w_k$. If $u = R_k^{(1)} s / \|R_k^{(1)} s\|$ and $v = y/\sqrt{y^{\mathrm{T}} s} - R_k^{(1)\mathrm{T}} u$, then it may be verified by direct multiplication that

$$Z_{k+1}^T H_{k+1} Z_{k+1} = (R_k^{(1)} + uv^T)^{\mathrm{T}} (R_k^{(1)} + uv^T).$$

Two sets of plane rotations can be applied to restore $R_k^{(1)} + uv^T$ to upper-triangular form. The first, S_1, is the product of plane rotations $P_{1,2} P_{1,3} \cdots P_{1,r_k}$ that zero out components 2 through r_k of u, i.e., $S_1 u = \gamma e_1$, with $\gamma = \pm\|u\|$. The application of S_1 to $R_k^{(1)} + uv^T$ gives

$$S_1 (R_k^{(1)} + uv^T) = S_1 R_k^{(1)} + \gamma e_1 v^T. \tag{2.9}$$

By construction, S_1 applied to $R_k^{(1)}$ results in an upper-Hessenberg matrix. As $\gamma e_1 v^T$ is a matrix with only nonzeros in its first row, the right-hand side of (2.9) is also upper-Hessenberg. A second set of plane rotations S_2 is then defined such that $R_k^{(2)} = S_2 S_1 R_2$, where $S_2 = P'_{1,2} P'_{2,3} \cdots P'_{r_k - 1, r_k}$. The resulting matrix $R_k^{(2)}$ is the upper-triangular factor of $Z_{k+1}^T H_{k+1} Z_{k+1}$. For more details, see Dennis and Schnabel [7], and Gill and Leonard [13]).

In finite-precision arithmetic, the use of the economy QR factorization instead of the full QR may cause a loss of orthogonality in Z_k as columns are added to the basis. When a gradient is accepted, the new column is computed as $z_{k+1} = v_{k+1}/\rho_{k+1}$, where $v_{k+1} = (I - Z_k Z_k^T)\nabla f(x_{k+1})$ and $\rho_{k+1} = \|v_{k+1}\|$. This choice of z_{k+1} is designed to force $Z_k^T v_{k+1}$ to be small relative to $\|\nabla f(x_{k+1})\|$. However, if ρ_{k+1} is small and $\|Z_k^T v_{k+1}\| = \epsilon \|\nabla f(x_{k+1})\|$ for some small ϵ, then the normalized vector $z_{k+1} = v_{k+1}/\rho_{k+1}$ would satisfy only $\|Z_k^T z_{k+1}\| = \epsilon \|\nabla f(x_{k+1})\|/\rho_{k+1}$. In this situation, the error relative to $\|\nabla f(x_{k+1})\|$ may be very large, resulting in a significant loss of orthogonality in the computed z_{k+1}. To rectify this loss of orthogonality, Daniel et al. [6] propose a *reorthogonalization* scheme. If $\|v_{k+1}\|/\|\nabla f(x_{k+1})\|$ is small, then v_{k+1} is refined using the scheme

$$v'_{k+1} = (I - Z_k Z_k^T) v_{k+1}.$$

If $\|v'_{k+1}\|/\|v_{k+1}\|$ is not too small, then v'_{k+1} can be scaled to provide a satisfactory update to Z_{k+1}. Otherwise, the process is repeated.

The initial approximate Hessian can greatly influence the practical performance of quasi-Newton methods. A choice of $H_0 = \sigma I$, with some arbitrary positive σ can result in poor performance, especially when $\nabla^2 f(x^*)$ is ill-conditioned. Moreover, equation (2.4) reveals that σ represents the approximate curvature along all directions in $\mathcal{G}_k^{\perp}$. To enhance the performance of L-RHR, *Hessian reinitialization* is applied to "reset" the approximate Hessian matrix with current curvature information. When a new gradient is accepted, the reduced Hessian is expanded with σ_k rather than σ in equation (2.8). Gill and Leonard [13] use $\sigma_k = w_k^T w_k / w_k^T d_k$ in L-RHR.

3. An L-RHR Method for Bound Constraints

In this section, we introduce the algorithm L-RH-B, which is an extension of the algorithm L-RHR for solving problem (BC). Given an initial $x_0 \in \Omega$, the sequence of iterates $\{ x_k \}$ satisfies $x_{k+1} = x_k(\alpha_k) = \mathbf{proj}_{\Omega}(x_k + \alpha_k p_k)$, where p_k is computed in terms of a direction d_k such that $\nabla f(x_k)^{\mathrm{T}} d_k < 0$. The vector d_k is the unique solution of the subproblem

$$\underset{d}{\text{minimize}}\ \nabla f(x_k)^{\mathrm{T}} d + \tfrac{1}{2} d^{\mathrm{T}} H_k d \quad \text{subject to } d_i = 0 \text{ for all } i \in \mathcal{W}_k(x_k), \tag{3.1}$$

where H_k is a positive-definite limited memory BFGS approximation of $\nabla^2 f(x_k)$, and $\mathcal{W}_k(x)$ is a *working set* of indices of x. The working set is defined as

$$\mathcal{W}_k(x) = \{\, i : x_i \le \ell_i + \epsilon_k \text{ and } \nabla_i f(x) > 0 \text{ or } x_i \ge u_i - \epsilon_k \text{ and } \nabla_i f(x) < 0 \,\},$$

where ϵ_k is a small positive scalar such that $\epsilon_k \to 0$ as $x_k \to x^*$. The matrix H_k is maintained in reduced-Hessian form and is not stored explicitly. Once the subproblem (3.1) has been solved, the components of d_k are modified if necessary to give a line-search direction p_k such that $\big[\, p_k \,\big]_i \ge 0$ if $\big[\, x_k \,\big]_i \le \ell_i + \epsilon_k$, and $\big[\, p_k \,\big]_i \le 0$ if $\big[\, x_k \,\big]_i \ge u_i - \epsilon_k$. This additional step guarantees convergence in the situation where iterates approach a boundary point from the interior of the feasible region—a phenomenon known as zigzagging or jamming (see Bertsekas [2]). The vector p_k retains the descent property of d_k. For example, if the solution of (3.1) has $\big[\, d_k \,\big]_i \ne 0$ and $\big[\, x_k \,\big]_i \le \ell_i + \epsilon_k$, then the definition of $\mathcal{W}_k(x_k)$ implies that $\nabla_i f(x_k) \le 0$. If $\big[\, d_k \,\big]_i > 0$ then $\big[\, p_k \,\big]_i = \big[\, d_k \,\big]_i$. If $\big[\, d_k \,\big]_i < 0$ then $\nabla_i f(x_k) \big[\, d_k \,\big]_i \ge 0$, and setting $\big[\, p_k \,\big]_i = 0$ makes the directional derivative more negative.

To simplify the notation, unless otherwise stated, it is assumed that the working set, vectors, and matrices in this section are associated with the k-th iteration of the algorithm.

The complement of $\mathcal{W}(x)$ in $\{ 1, 2, \ldots, n \}$ is denoted by $\mathcal{F}(x)$, which may be regarded as the set of indices of the variables that are free to move at x. At a given x, the components x_i with $i \in \mathcal{F}(x)$ may be interpreted as the set of *free variables*.

The *projected direction of g with respect to the index set $\mathcal{W}(x)$* is defined as

$$\big[\, P_{\mathcal{W}(x)}(g) \,\big]_i = \begin{cases} 0 & \text{if } \ i \in \mathcal{W}(x), \\ g_i & \text{if } \ i \notin \mathcal{W}(x). \end{cases} \tag{3.2}$$

Let Π denote a matrix with orthonormal columns that spans the set of projected directions with respect to the working set $\mathcal{W}(x)$. The columns of Π can be taken as the columns of the identity matrix of order n associated with the indices in $\mathcal{F}(x)$.

In the following discussion, vectors and matrices associated with the algorithm for unconstrained optimization described in Section 2 are given a suffix "n". The projected-search direction is then computed in the *projected* gradient subspace $\mathcal{G} = \big\{\, \Pi\Pi^T g : g \in \mathcal{G}_n \,\big\}$. Let the columns of the matrix B_n be a basis for $\mathcal{G}_n$.

The projected-search direction is computed as $p = Zq$, where q is the solution of the symmetric positive-definite equations

$$Z^{\mathrm{T}} H Z q = -Z^T \nabla f(x).$$

The matrix Z is the orthogonal factor of the *projected* basis matrix $B = \Pi\Pi^{\mathrm{T}} B_n$. Analogous to L-RHR, the columns of Z span the projected gradient subspace and T is computed as a nonsingular upper-triangular matrix with $B = ZT$. Note that B is the matrix B_n with zeros in the rows corresponding to indices in the current working set. The search direction may be computed efficiently by using a Cholesky factor of the "projected" reduced Hessian matrix $R^{\mathrm{T}} R = Z^{\mathrm{T}} H Z$.

Once p has been computed, the next iterate $\widehat{x}$ of the form $x(\alpha) = \mathbf{proj}_{\,\Omega}(x + \alpha p)$ is found using the projected line-search described in Section 4. The point x and the associated working set $\mathcal{W}(x)$ are then updated and the projected matrix factors B, Z, and T are modified to reflect the changes in $\mathcal{W}(x)$. If the projected gradient at $\widehat{x}$ contains components outside of range(Z), then it can be added to the basis. If the value of the scalar $\rho = \|(I - ZZ^T)\nabla f(\widehat{x})\|$ is zero, then the new gradient lies in $\mathcal{G}$ and $\nabla f(\widehat{x})$ is *rejected* for inclusion in $\mathcal{G}_n$. In this case, no further updates to the factors of B are needed. Otherwise, the dimension of $\mathcal{G}$ increases by one and the gradient $\nabla f(\widehat{x})$ is *accepted*. In this case, B_n gains a new column and the change must be incorporated in the QR factors of B analogous to (2.7). The matrix updates associated with changes in the working set and the basis are described in Section 5. In what follows, $\widehat{Z}$ denotes the projected gradient basis at the end of an iteration.

The remaining task is to update R to reflect the curvature information determined in the step from x to $\widehat{x}$. In the unconstrained case, the approximate Hessian is updated using the BFGS formula (2.1) with $d = \widehat{x} - x$ and $w = \nabla f(\widehat{x}) - \nabla f(x)$. In the bound-constrained case, the situation is more complicated because $\mathcal{W}(\widehat{x})$ may differ from $\mathcal{W}(x)$, in which case $\widehat{x} - x$ may not be the same as αp or $\widehat{x} - x$ may not lie in range$(\widehat{Z})$. One approach is to replace the vectors $\widehat{x}$ and $\nabla f(\widehat{x})$ by $\bar{x}$ and $\nabla f(\bar{x})$ in the definitions of d, where $\bar{x} = x + \widehat{Z}\widehat{Z}^{\mathrm{T}}(\alpha p)$. However, this strategy would require an extra gradient evaluation at $\bar{x}$. Instead, $\nabla f(\bar{x})$ is approximated by $\nabla q(\bar{x})$, where $q(z)$ is the quadratic model $f(\widehat{x}) + \nabla f(\widehat{x})^{\mathrm{T}}(z - \widehat{x}) + \frac{1}{2}(z - \widehat{x})^{\mathrm{T}} H (z - \widehat{x})$. This gives

$$\begin{aligned} w = \nabla q(\bar{x}) - \nabla f(x) &= \nabla f(\widehat{x}) + H(\bar{x} - \widehat{x}) - \nabla f(x) \\ &= \nabla f(\widehat{x}) + H(x - \widehat{x}) + H\widehat{Z}\widehat{Z}^T(\alpha p) - \nabla f(x). \end{aligned}$$

Then $s = \widehat{Z}^T(\bar{x} - x) = \widehat{Z}^T(x + \widehat{Z}\widehat{Z}^{\mathrm{T}}(\alpha p) - x) = \widehat{Z}^T\widehat{Z}\widehat{Z}^T(\alpha p) = \alpha\widehat{Z}^T p$, and the vector $y = Z^T w$ may be written as

$$y = \widehat{y} + Z^{\mathrm{T}} H(x - \widehat{x}) + Z^{\mathrm{T}} H\widehat{Z}s, \tag{3.3}$$

with $\widehat{y} = Z^T\big(\nabla f(\widehat{x}) - \nabla f(x)\big)$. The definition of H from (2.6) implies that

$$Z^{\mathrm{T}} H(x - \widehat{x}) = Z^T\big(ZZ^{\mathrm{T}} H\widehat{Z}Z^T + \sigma(I - \widehat{Z}Z^T)\big)(x - \widehat{x}) = -R^{\mathrm{T}} R\widehat{s}, \tag{3.4}$$

with $\widehat{s} = Z^T(\widehat{x} - x)$. Combining (3.3) and (3.4) yields

$$y = \widehat{y} - R^{\mathrm{T}} R\widehat{s} + R^{\mathrm{T}} Rs = \widehat{y} + R^{\mathrm{T}} R(s - \widehat{s}).$$

Note that if $\widehat{x} - x \in \mathrm{range}(\widehat{Z})$, then $\bar{x} = \widehat{x}$, $s = Z^T(\widehat{x} - x)$, and $y = Z^T\big(\nabla f(\widehat{x}) - \nabla f(x)\big)$.

Algorithm LRHB: Limited-memory reduced-Hessian method.

Choose m ($m > 0$); σ ($\sigma > 0$); $x \in \mathbb{R}^n$;
$x \leftarrow \mathbf{proj}_{\Omega}(x)$; $g \leftarrow P_{\mathcal{W}(x)}(\nabla f(x))$;
$B_n \leftarrow \big(\nabla f(x)\big)$; $B \leftarrow \big(g\big)$; $Z \leftarrow \big(g/\|g\|\big)$; $T \leftarrow \big(\|g\|\big)$;
$R \leftarrow \big(\sqrt{\sigma}\big)$; $v \leftarrow \big(\|g\|\big)$;
$\rho \leftarrow 0$;
while not converged **do**
    Compute the search direction $p = Zq$, where $R^{\mathrm{T}}Rq = -v$;
    if $\rho > 0$ **then**
        Replace last column of B_n and B with p; Update T;
    end if
    Compute the step length α;
    $\widehat{x} \leftarrow \mathbf{proj}_{\Omega}(x + \alpha p)$; $\widehat{g} \leftarrow P_{\mathcal{W}(\widehat{x})}(\nabla f(\widehat{x}))$;
    if $\mathcal{W}(\widehat{x}) \neq \mathcal{W}(x)$ **then**
        Update B_n, B, T, R, w, v, q;
    end if
    $w \leftarrow Z^T\widehat{g}$; $\rho \leftarrow \|(I - ZZ^T)\widehat{g}\|$;
    if $\rho > 0$ **then**
        Update B_n, B, T, R, w, v, q;
    end if
    $s \leftarrow \alpha q$; $y \leftarrow w - v$;
    if $\widehat{x} - x \notin \text{range}(B)$ **then**
        $y \leftarrow y + R^{\mathrm{T}}R\big(Z^T(x - \widehat{x}) + s\big)$;
    end if
    If $y^T s > 0$, apply the BFGS update to R;
    Compute new curvature $\sigma > 0$; If $n > \min(m_{\max}, m)$, reinitialize R;
    if $\text{rank}(B) > m$ **then**
        Drop the oldest basis vector in B_n; Update B, T, R, and w;
    end if
    $x \leftarrow \widehat{x}$; $v \leftarrow w$;
end while

4. The Line Search

In L-RH-B each iterate has the form $x_{k+1} = x_k(\alpha_k)$, where $x_k(\alpha) = \mathbf{proj}_{\Omega}(x_k + \alpha p_k)$. The function $x_k(\alpha)$ defines a piecewise linear continuous path, and the line-search function $f\big(x_k(\alpha)\big)$ is not necessarily differentiable along $x_k(\alpha)$. In particular, $f\big(x_k(\alpha)\big)$ has "kinks" where $\big[\,x_k+\alpha p_k\,\big]_i = \ell_i$ or $\big[\,x_k+\alpha p_k\,\big]_i = u_i$. This implies that it is not possible to use a line search based on the Wolfe conditions (2.3) and (2.3). An alternative is to use a *quasi-Armijo* line search based on satisfying the Armijo condition along the path $x_k(\alpha)$. A quasi-Armijo step has the form $\alpha_k = \gamma\sigma^{j_k}$, where j_k is the smallest nonnegative integer such that

$$f\big(x_k(\alpha_k)\big) \leq f(x_k) + \alpha_k \eta_A \nabla f(x_k)^{\mathrm{T}} p_k, \tag{4.1}$$

with γ, σ, and η_A fixed parameters such that $\gamma > 0$, $\sigma \in (0,1)$, and $\eta_A \in (0,1)$ (see Bertsekas [1, 2]). However, as there is no restriction on the magnitude of the directional derivative, the benefit of guaranteeing a positive-definite quasi-Newton update is lost.

Algorithm L-RH-B uses a *quasi-Wolfe* line search, which is designed for piecewise differentiable functions. Performing a line search on the univariate function

$$\psi_k(\alpha) = f\big(x_k(\alpha)\big) = f\big(\mathbf{proj}_{\,\Omega}(x_k + \alpha p_k)\big),$$

is complicated by the fact that ψ_k is only piecewise differentiable, with a finite number of jump discontinuities in the derivative. In the following discussion, the suffix k is omitted if the iteration index is not relevant to the discussion. The behavior of a quasi-Wolfe line search is similar to that of a conventional Wolfe line search, except that a step is accepted under a wider range of conditions. These conditions take into consideration steps at which the restriction of the objective function on the search path is not differentiable. The left and right derivatives $\psi'_-(\alpha)$ and $\psi'_+(\alpha)$ of ψ at α are defined as

$$\psi'_-(\alpha) = \lim_{\beta \to \alpha_-} \psi'(\beta) \quad \text{and} \quad \psi'_+(\alpha) = \lim_{\beta \to \alpha_+} \psi'(\beta).$$

A step α is called a *quasi-Wolfe step* if it satisfies the Armijo condition

$$\psi(\alpha) \leq \psi(0) + \alpha \eta_A \psi'_+(0),$$

and at least one of the following conditions:

($\mathbf{C}_1$) $|\psi'_-(\alpha)| \leq \eta_W |\psi'_+(0)|$;

($\mathbf{C}_2$) $|\psi'_+(\alpha)| \leq \eta_W |\psi'_+(0)|$;

($\mathbf{C}_3$) ψ is not differentiable at α and $\psi'_-(\alpha) \leq 0 \leq \psi'_+(\alpha)$.

Conditions for the existence of a quasi-Wolfe step are established in the following result.

Proposition 4.1. *Let f be a scalar-valued continuously differentiable function defined on $\Omega = \{\, x \in \mathbb{R}^n : \ell \leq x \leq u \,\}$. Assume that $x_0 \in \Omega$ is chosen such that the level set $\mathcal{L}\big(f(x_0)\big)$ is closed and bounded, and assume that $\{\, p_k \,\}$ is a sequence of feasible descent directions. If $0 < \eta_A < \eta_W < 1$, then at every iteration k either there exists an $\alpha_L > 0$ and an interval (α_L, α_U) such that every $\alpha \in (\alpha_L, \alpha_U)$ is a quasi-Wolfe step, or there exists a quasi-Wolfe step that satisfies the condition* **($\mathbf{C}_3$)**.

Proof. See Ferry et al. [11]. ∎

The quasi-Wolfe line search makes extensive use of the auxiliary function

$$\omega(\alpha) = \psi(\alpha) - \big(\psi(0) + \alpha \eta_A \psi'_+(0)\big), \quad \text{with} \quad \omega'_\pm(\alpha) = \psi'_\pm(\alpha) - \eta_A \psi'_+(0). \tag{4.2}$$

A quasi-Wolfe line search consists of two stages. The first stage begins with an initial step length α_0 and continues with steps of increasing magnitude until one of three

things occurs: (i) a step satisfying one of the conditions ($\mathbf{C}_1$)–($\mathbf{C}_3$) is found; (ii) an interval that contains a quasi-Wolfe step is found; or (iii) the step is considered to be unbounded. If the first stage terminates with a bounded step, the second stage generates a sequence of nested intervals $\{\mathcal{I}(\alpha_{\text{low}}^{(j)}, \alpha_{\text{high}}^{(j)})\}$, such that

(a) the interval bounded by $\alpha_{\text{low}}^{(j)}$ and $\alpha_{\text{high}}^{(j)}$ contains a quasi-Wolfe step;

(b) among all the step lengths generated so far, $\alpha_{\text{low}}^{(j)}$ gives the least value of ω;

(c) $\alpha_{\text{high}}^{(j)}$ is chosen so that $\omega'_+(\alpha_{\text{low}}^{(j)}) < 0$ if $\alpha_{\text{low}}^{(j)} < \alpha_{\text{high}}^{(j)}$, or $\omega'_-(\alpha_{\text{low}}^{(j)}) > 0$ if $\alpha_{\text{low}}^{(j)} > \alpha_{\text{high}}^{(j)}$.

An interval with end points $\alpha_{\text{low}}^{(j)}$ and $\alpha_{\text{high}}^{(j)}$ is known as an *interval of uncertainty*. Similarly, $\alpha_{\text{low}}^{(j)}$ and $\alpha_{\text{high}}^{(j)}$ are said to *bracket* a quasi-Wolfe step. In practice, an upper bound $\alpha_{\max}$ is imposed on the value of α_j and the search is terminated if this bound is exceeded during the first stage. If the line search terminates at $\alpha_{\max}$ without finding an interval containing a quasi-Wolfe step, then all of the steps computed up to that point satisfy $\omega(\alpha_j)$.

A major difference between a Wolfe and quasi-Wolfe line search concerns how interpolation is used to find new steps in the second stage. For each interval of uncertainty, $(\alpha_{\text{low}}, \alpha_{\text{high}})$ a new trial step α_{new} is generated. In the differentiable case, α_{new} is usually obtained by polynomial interpolation using the objective and its derivatives at α_{low} and α_{high}. If the line-search function is only piecewise differentiable, there may be kink points between α_{low} and α_{high} in which case a conventional interpolation approach may not provide a good estimate of a quasi-Wolfe step. One strategy to speed convergence in this situation is to search for the kink step (if it exists) between α_{low} and α_{high} that is closest to α_{low}.

The search for the kink points proceeds as follows. Once the first stage terminates with an interval $(\alpha_{\text{low}}, \alpha_{\text{high}})$, the kink steps are computed in $O(n)$ flops from

$$\kappa_i = \begin{cases} (u_i - x_i)/p_i & \text{if } p_i > 0, \\ (\ell_i - x_i)/p_i & \text{if } p_i < 0, \\ \infty & \text{if } p_i = 0. \end{cases}$$

As the interval bounded by α_{low} and α_{high} brackets a quasi-Wolfe step, only the kink steps within that interval need be stored. These steps are then sorted in decreasing order within $O(n \log n)$ operations using a heapsort algorithm (see, e.g., Williams [23], Knuth [17, Section 5.2.3]). The kink step closest to α_{low}, say κ_1^*, is either the smallest or the largest kink step within the interval of uncertainty, depending on whether α_{low} is smaller or greater than α_{high}. Once κ_1^* has been found, the search for κ_l^* $(l > 1)$ is made towards α_{low} starting at the kink step κ_{l-1}^* from the preceding iteration. To prevent the iterations from lingering at **Case (4)** for too long, an upper limit is imposed on the number of consecutive kink steps as trial steps. If this limit is reached, a new trial step is generated by bisection. Once all the kinks in the interval of uncertainty have been eliminated, conventional polynomial interpolation may be used to generate a new step length. If there is just

one kink step in the interval of uncertainty, α_{new} is set to be that kink step. For further details, see Ferry et al. [11].

An important benefit of the conventional Wolfe conditions in the unconstrained case is that the restriction on the directional derivative guarantees the satisfaction of a necessary condition for the quasi-Newton update to give a positive-definite approximate Hessian. Unfortunately it is not possible to completely guarantee this property in the bound-constrained case, although the likelihood of a skipped update is significantly less than that for a method using an Armijo step. If the next iterate is given by $x_{k+1} = \mathbf{proj}_{\Omega}(x_k + \alpha_k p)$, where α_k is a quasi-Wolfe step, then the approximate curvature $\big(\nabla f(x_{k+1}) - \nabla f(x_k)\big)^{\mathrm{T}}(x_{k+1} - x_k)$ need not be greater than zero. This situation can occur only if the path $\mathbf{proj}_{\Omega}(x_k + \alpha_k p_k)$ changes direction at some point $\alpha \in (0, \alpha_k)$.

5. Matrix Modifications

The modifications of the matrix factors induced by changes to the working set are described in this section. The strategies defined are based on the work of Daniel et al. [6]. A majority of the computational effort involves the application of a sequence of plane rotation matrices to one or more matrices. We describe the updates required to maintain the matrix factors Z and T for the basis matrix B and the Cholesky factor R of the reduced Hessian matrix. Under certain circumstances, it is more cost-effective to compute "indirect" updates to R via updates to the Cholesky factor of $B^T H B$. We omit these results in this paper, but refer readers to Gill and Leonard [13] and Ferry [10] for further details. In this section, values at the current iteration will be undecorated and values at the next iteration are denoted with a "hat".

5.1. Removing an index from the working set

If variable i moves off its lower or upper bound, then $i \notin \mathcal{W}(\widehat{x})$ and the projected basis matrix B and its factors must be modified to reflect the change in the working set. Because the i-th variable is now free to move, the i-th row of B_n must be restored to the zeroed-out i-th row of B. The change in basis may be represented by the rank-one modification

$$\widehat{B} = B + e_i b^T = \begin{pmatrix} Z & e_i \end{pmatrix} \begin{pmatrix} T \\ b^T \end{pmatrix}, \quad \text{where} \quad b^T = e_i^T B_n.$$

Because $i \in \mathcal{W}(x)$, the columns of $\begin{pmatrix} Z & e_i \end{pmatrix}$ are orthonormal as the i-th row of Z is zero and we must have $Z^T e_i = 0$. The addition of the row b^T to T however creates a row spike that must be removed by the application of plane rotation matrices.

Suppose that the basis matrix B_n has r columns at the start of the current iteration. Consider the plane rotation $P_{r+1,j}$ that operates on rows j and $r+1$, zeroing out the j-th element of row $r+1$. A sequence of plane rotations can be applied on the left of the matrix to eliminate each element of the row spike. Thus,

if $S_d = P_{r+1,r}P_{r+1,r-1}\cdots P_{r+1,2}P_{r+1,1}$, then

$$S_d \begin{pmatrix} T \\ b^T \end{pmatrix} = P_{r+1,r}P_{r+1,r-1}\cdots P_{r+1,2}P_{r+1,1} \begin{pmatrix} T \\ b^T \end{pmatrix} = \begin{pmatrix} \bar{T} \\ 0 \end{pmatrix},$$

with $\bar{T}$ nonsingular and upper triangular. Each each plane rotation is orthogonal, and it must hold that

$$\widehat{B} = \begin{pmatrix} Z & e_i \end{pmatrix} S_d^T S_d \begin{pmatrix} T \\ b^T \end{pmatrix} = \begin{pmatrix} \bar{Z} & \bar{z} \end{pmatrix} \begin{pmatrix} \bar{T} \\ 0 \end{pmatrix},$$

where $\bar{Z}$ denotes the first r columns of $\begin{pmatrix} Z & e_i \end{pmatrix} S_d^T$. Thus, $\widehat{Z} = \bar{Z}$ with $\widehat{Z}^T\widehat{Z} = I$, and $\widehat{T} = \bar{T}$.

5.2. Adding an index to the working set

When variable x_i moves onto its bound, i is added to the working set and the i-th component of the search direction at the next iteration must be restricted to zero. As the search direction must lie in the column space of Z, this can be done by zeroing out the i-th row of Z or B with the rank-one modification

$$\widehat{B} = B - e_i b^T, \quad \text{where} \quad b^T = e_i^T B_n \quad \text{is the } i\text{th row of } B_n.$$

If $e_i \in \text{range}(B)$, then $\begin{pmatrix} B & e_i \end{pmatrix}$ is rank deficient and the resulting updated matrix is also rank deficient. To prevent rank deficiency, a column from B (and B_n) is removed. The details of this procedure are discussed in Section 5.4.2.

Unlike the previous case of index removal, there is no guarantee that $Z^T e_i = 0$. If $e_i \notin \text{range}(B)$ (or columns of B were removed so that this holds), define w as the normalized component of e_i orthogonal to Z, i.e.,

$$\rho w = (I - ZZ^T)e_i, \quad \text{where } \rho \text{ is the normalizing scalar.} \tag{5.1}$$

Daniel et al. [6, p. 779] show that the norm of the i-th row of the matrix $\begin{pmatrix} Z & w \end{pmatrix}$ is one, which implies that for any $(r+1)\times(r+1)$ orthogonal matrix S_a it must hold that $\|e_i^T \begin{pmatrix} Z & w \end{pmatrix} S_a^T\| = 1$. In particular, if S_a is a product of plane rotations $S_a = P_{r+1,1}P_{r+1,2}\cdots P_{r+1,r-1}P_{r+1,r}$, then

$$S_a \begin{pmatrix} Z^T \\ w^T \end{pmatrix} e_i = \begin{pmatrix} 0 \\ \tau \end{pmatrix}, \quad \text{with} \quad \tau = \pm 1,$$

or, equivalently,

$$\begin{pmatrix} Z & w \end{pmatrix} S_a^T = \begin{pmatrix} \widehat{Z} & \tau e_i \end{pmatrix},$$

where $\widehat{Z}$ is a matrix with orthonormal columns. The projected basis B may be rewritten in the form

$$\begin{aligned} B = ZT = \begin{pmatrix} Z & w \end{pmatrix} S_a^T S_a \begin{pmatrix} T \\ 0 \end{pmatrix} &= \begin{pmatrix} \widehat{Z} & \tau e_i \end{pmatrix} S_a \begin{pmatrix} T \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \widehat{Z} & \tau e_i \end{pmatrix} \begin{pmatrix} \widehat{T} \\ t^T \end{pmatrix} \\ &= \widehat{Z}\widehat{T} + \tau e_i t^T. \end{aligned}$$

As S_a is constructed to transform the i-th row of $\widehat{Z}_F$ to zero, it must hold that $\tau t = b$. Therefore, $\widehat{B} = B - e_i b^T = B - \tau e_i t^T = \widehat{Z}\widehat{T}$, as required. Note that because T is upper-triangular, the application of S_a introduces a row spike t^T but does not affect the triangular structure of $\widehat{T}$.

The procedure described here depends on defining the vector w orthogonal to the columns of Z. As in the case of adding a vector to the basis, numerical issues may cause a loss of orthogonality in practice. In this case, the reorthogonalization scheme described in Section 2 can be applied to w to ensure orthogonality in the updated matrices.

5.3. Updates to the Cholesky factor

When updates are performed on Z as a result of working set changes, the Cholesky factor R of the reduced Hessian $Z^{\mathrm{T}}HZ$ must also be updated. A similar set of updates is performed on Z regardless of whether a variable is removed or added to the working set. In the first step, Z is expanded by a column vector y that is orthogonal to Z. When adding a variable to $\mathcal{W}$, $y = w$ defined by (5.1); when removing a variable, $y = e_i$. In both cases, it holds that $Z^T y = 0$ and with the definition of H (2.6), the expansion of Z with the vector y leads to

$$\begin{pmatrix} Z^T \\ y^T \end{pmatrix} H \begin{pmatrix} Z & y \end{pmatrix} = \begin{pmatrix} Z^{\mathrm{T}}HZ & Z^{\mathrm{T}}Hy \\ y^{\mathrm{T}}HZ & y^{\mathrm{T}}Hy \end{pmatrix} = \begin{pmatrix} Z^{\mathrm{T}}HZ & 0 \\ 0 & y^{\mathrm{T}}Hy \end{pmatrix}$$

with R appropriately expanded to

$$\begin{pmatrix} R & 0 \\ 0 & \sqrt{\sigma} \end{pmatrix}, \quad \text{where} \quad \sigma = y^{\mathrm{T}}Hy.$$

Next, a product of plane rotations, say S_1, is applied on the right of Z. Applying these rotations directly to R leads to a matrix that is an unsuitable Cholesky factor as the matrix is not upper triangular. A second set of plane rotations S_2 must be applied on the left to obtain a suitable Cholesky factor

$$\bar{R} = S_2 \begin{pmatrix} R & 0 \\ 0 & \sqrt{\sigma} \end{pmatrix} S_1^T.$$

The updated factor $\widehat{R}$ is the $r \times r$ leading submatrix of $\bar{R}$. A detailed explanation of the plane rotations is given in Ferry [10].

5.4. Basis updates

5.4.1. Defining a basis with search directions

Suppose that a limit of m columns is imposed on the dimension of the gradient subspace. The obvious choice is to discard the oldest gradient from the basis. However, previous work has shown that when the basis matrix B is defined by the gradient, this choice is inefficient in practice. An alternate strategy proposed by Siegel [22] (and utilized by Gill and Leonard [13] in L-RHR) is to take the columns of B to be

search directions instead of gradients. This approach preserves the finite termination property of the algorithm when the oldest basis vector is discarded.

Suppose that $\widehat{g}$ is accepted and added to the basis at the end of an iteration. Because the next search direction $\widehat{p}$ is not available until the next iteration, $\widehat{g}$ is added to B and the associated matrix factors are updated until the new search direction $\widehat{p}$ can be swapped in. Once $\widehat{p}$ is known, $\widehat{g}$ is replaced by $\widehat{p}$ in the basis matrix, and the orthogonal factors of the basis must be updated. If the updated basis matrix is $\bar{B} = \begin{pmatrix} B & \widehat{p} \end{pmatrix}$, then from (2.7), $\bar{T}$ is defined as

$$\bar{T} = \begin{pmatrix} \begin{pmatrix} T \\ 0 \end{pmatrix} & Z^T\widehat{p} \end{pmatrix},$$

so that $\bar{B} = Z\bar{T}$.

5.4.2. Removing columns from the basis

During one iteration of L-RH-B, a column may be removed from B_n. The removal occurs (i) when after accepting a new gradient, the number of columns in the resulting basis exceeds the predefined limit of m, or (ii) to prevent rank deficiency when adding variable i to the working set. We describe the associated updates to B, Z and T. The results may be applied to the projected matrices in a similar manner.

Suppose column j is removed from the $n \times r$ basis matrix B_n. If b and t are the j-th columns of B and T, respectively, then B and T may be partitioned such that

$$B = \begin{pmatrix} B_1 & b & B_2 \end{pmatrix} = Z \begin{pmatrix} T_1 & t & T_2 \end{pmatrix}.$$

Plane rotations are applied on the left of the submatrix $\begin{pmatrix} T_1 & T_2 \end{pmatrix}$ to eliminate elements $(i, i+1)$ for $i = j, \ldots, r-1$, giving

$$P \begin{pmatrix} T_1 & T_2 \end{pmatrix} = \begin{pmatrix} \widehat{T} \\ 0 \end{pmatrix}.$$

Then,

$$\widehat{B} = \begin{pmatrix} B_1 & B_2 \end{pmatrix} = ZP^{\mathrm{T}}P \begin{pmatrix} T_1 & T_2 \end{pmatrix} = ZP^T \begin{pmatrix} \widehat{T} \\ 0 \end{pmatrix} = \begin{pmatrix} \widehat{Z} & z \end{pmatrix} \begin{pmatrix} \widehat{T} \\ 0 \end{pmatrix} = \widehat{Z}\widehat{T}.$$

Updates similar to those described in Section 5.3 are applied to R to reflect the changes to Z.

6. Convergence results

The following result is established in Ferry et al. [11]. The result gives the properties of a quasi-Wolfe search for an arbitrary sequence of search directions $\{ p_k \}$.

Theorem 6.1. (Convergence of quasi-Wolfe line search) *Let f be a scalar-valued continuously differentiable function defined on $\Omega = \{ x \in \mathbb{R}^n : \ell \leq x \leq u \}$. Assume that $x_0 \in \Omega$ is chosen such that the level set $\mathcal{L}(f(x_0))$ is bounded, and*

$\{ x_k \}$ is given by $x_{k+1} = x_k(\alpha_k)$, where α_k is a quasi-Wolfe step. Also assume that $\{ p_k \}$ is a sequence of feasible descent directions with $\|p_k\| \leq \theta$ for some constant θ independent of k. For an arbitrarily fixed $\epsilon > 0$, define $\epsilon_0 = \epsilon$, and

$$\epsilon_k = \min \{\epsilon, \|\Pi_{k-1}^{\mathrm{T}} \nabla f(x_{k-1})\|\}.$$

for $k \geq 1$, where each Π_k is a matrix with orthonormal columns that spans the set of projected directions with respect to the working set $\mathcal{W}_k(x_k)$. If $\Pi_k \Pi_k^T p_k = p_k$, and the components of p_k satisfy $[p_k]_i \geq 0$ if $[x_k]_i \leq \ell_i + \epsilon_k$, and $[p_k]_i \leq 0$ if $[x_k]_i \geq u_i - \epsilon_k$, then

$$\lim_{k \to \infty} |\nabla f(x_k)^{\mathrm{T}} p_k| = 0.$$

If the eigenvalues of the projected approximate Hessian are uniformly bounded, then the projected gradient converges to zero as shown in the theorem below.

Theorem 6.2. *Let $\{ x_k \}$ be a sequence of iterates generated by* Algorithm L-RH-B. *In addition to assumptions of* Theorem 6.1, *if there exist a constants γ such that every eigenvalue of the projected approximate Hessian satisfies*

$$0 < \lambda(\Pi_k^T H_k \Pi_k) \leq \gamma < \infty$$

for all k, where Π_k is a matrix with orthonormal columns that spans the set of projected directions with respect to the working set $\mathcal{W}_k(x_k)$, then

$$\lim_{k \to \infty} \|\Pi_k^T \nabla f(x_k)\| = 0.$$

Proof. Let d_k denote the approximate solution to the subproblem (3.1) within the subspace spanned by columns of $\Pi_k \Pi_k^T B_k$, and let Z_k be the orthogonal factor of the thin QR decomposition of $\Pi_k \Pi_k^T B_k$. Then

$$|\nabla f(x_k)^{\mathrm{T}} d_k| = |\nabla f(x_k)^{\mathrm{T}} Z_k (Z_k^{\mathrm{T}} H_k Z_k)^{-1} Z_k^{\mathrm{T}} \nabla f(x_k)| \geq \|Z_k^{\mathrm{T}} \nabla f(x_k)\|^2 / \lambda_{\max}(\Pi_k^T H_k \Pi_k),$$

for all k, where $\lambda_{\max}(\Pi_k^T H_k \Pi_k)$ represents the largest eigenvalue of the projected approximate Hessian. As $\Pi_k \Pi_k^T \nabla f(x_k)$ lies in the column space of Z_k,

$$\|Z_k^{\mathrm{T}} \nabla f(x_k)\| = \|Z_k^{\mathrm{T}} \Pi_k \Pi_k^T \nabla f(x_k)\| = \|\Pi_k \Pi_k^T \nabla f(x_k)\| = \|\Pi_k^T \nabla f(x_k)\|.$$

It follows that

$$|\nabla f(x_k)^{\mathrm{T}} d_k| \geq \|\Pi_k^T \nabla f(x_k)\|^2 / \lambda_{\max}(\Pi_k^T H_k \Pi_k) \geq \|\Pi_k^T \nabla f(x_k)\|^2 / \gamma.$$

Then

$$0 = \lim_{k \to \infty} |\nabla f(x_k)^{\mathrm{T}} p_k| \geq \lim_{k \to \infty} |\nabla f(x_k)^{\mathrm{T}} d_k| \geq \lim_{k \to \infty} \|\Pi_k^T \nabla f(x_k)\|^2 / \gamma.$$

Therefore,

$$\lim_{k \to \infty} \|\Pi_k^T \nabla f(x_k)\| = 0.$$

∎

A stationary point $x^* \in \Omega$ of (BC) is nondegenerate if

$$\nabla f(x^*) = \sum_{i \in \mathcal{A}(x^*)} \lambda_i e_i,$$

where e_i is the unit vector with i-th component equal to 1 for each i, and $\lambda_i \in \mathbb{R}$ satisfies that $\lambda_i > 0$ if $[x^*]_i = \ell_i$ and $\ell_i < u_i$, while $\lambda_i < 0$ if $[x^*]_i = u_i$ and $\ell_i < u_i$. It is stated in the following theorem that, if the sequence of iterates $\{ x_k \}$ converges to a nondegenrate stationary point, then the optimal active set can be identified with a finite number of iterations. The theorem is established in [11].

Theorem 6.3. *In addition to assumptions of* Theorem 6.1, *assume that* $\{ x_k \}$ *converges to a nondegenerate stationary point* x^*. *Define*

$$\mathcal{A}_{\epsilon_k}(x_k) = \{ i : [x_k]_i \leq \ell_i + \epsilon_k \ \ or \ \ [x_k]_i \geq u_i - \epsilon_k \}.$$

If $\| \Pi_k^T \nabla f(x_k) \| \to 0$, *then* $\mathcal{A}_{\epsilon_k}(x_k) = \mathcal{A}(x_k) = \mathcal{A}(x^*)$ *for all* k *sufficiently large.* ∎

Theorem 6.3 implies that, after a finite number of iterations, L-RH-B will eventually reduce to the L-RHR for the unconstrained minimization with respect to the inactive variables. Therefore, Algorithm L-RH-B has the same convergence properties as L-RHR.

7. Numerical Results

We present numerical results obtained using L-RH-B, the Fortran implementation of the limited-memory reduced-Hessian method discussed in this paper. L-RH-B is designed to solve large-scale unconstrained or bound-constrained problems of the form (BC).

The algorithm described maintains several dense matrices: the basis B_n and the projected counterpart B with factors Z and T and the Cholesky factor R for $Z^{\mathrm{T}} H Z$. In practice, it is not necessary to maintain and store all of these factors. L-RH-B can operate in two modes: "explicit" mode, where Z and T are stored, but not B, or "implicit" mode, where B and T are stored, but not Z.

The algorithm is applied to 417 problems from CUTEst problem collection. Of the 417 problems, 152 are bound-constrained and 265 are unconstrained.

Results are presented for L-RH-B with its default settings on a PC with 3.20GHz Intel Core i7-8700 CPU and 64GB of memory. Version 7.5.0 of the `GCC` compilers was used. The optimized `BLAS` library were used for all solvers.

The results are summarized using performance profiles (in $\log_2$ scale) proposed by Dolan and Moré [8]. If $\mathcal{P}$ denotes the set of problems used for a given numerical experiment. For each method s we define the function $\pi_s : [0, r_M] \mapsto \mathbb{R}^+$ such that

$$\pi_s(\tau) = \frac{1}{|\mathcal{P}|} \left| \{ p \in \mathcal{P} : \log_2(r_{p,s}) \leq \tau \} \right|,$$

where $r_{p,s}$ denotes the ratio of the number of function evaluations needed to solve problem p with method s and the least number of function evaluations needed to solve problem p. The number r_M is the maximum value of $\log_2(r_{p,s})$.

Figures 1–2 provide a relative comparison of the number of function evaluations for L-RH-B using the implicit or explicit storage mode with one of the three line-search implementations for the set of bound-constrained and unconstrained problems.

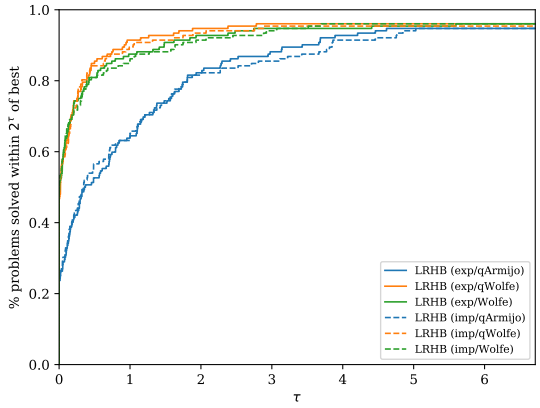

Figure 1: Performance profile of function evaluations for L-RH-B on the bound-constrained CUTEst test problems.

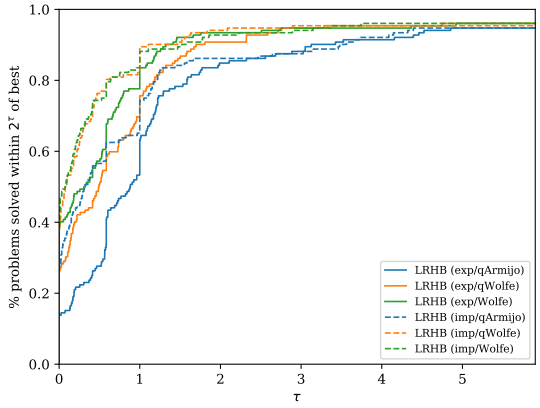

Figure 2: Performance profile of wall time for L-RH-B on the bound-constrained CUTEst test problems.

Based on Figure 1, the choice of line search appears to have the greatest effect on the number of function evaluations performed by L-RH-B, with the quasi-Wolfe and Wolfe line searches performing more efficiently and robustly than the quasi-Armijo line search. The choice of storage mode however affects the time required by the

solver. Figure 2 show that the explicit storage of Z generally required more time than implicit storage.

Comparison with the solver L-BFGS-B (Byrd, Lu, Nocedal and Zhu [4], Morales and Nocedal [20]) and the solver ASA_CG (Hager and Zhang [15]) are presented in Figures 3–6, which depict the performance profiles with respect to the number of function evaluations.

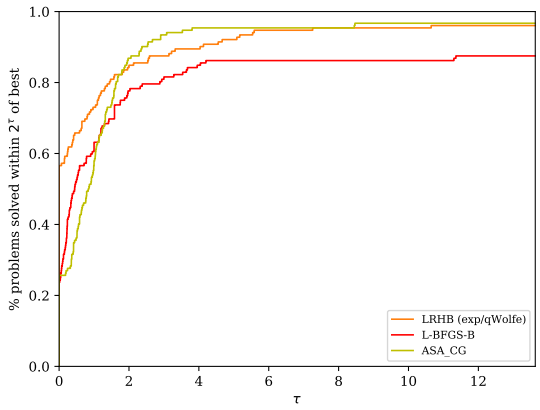

Figure 3: Performance profile of function evaluations for L-RH-B and L-BFGS-B on the bound-constrained CUTEst test problems.

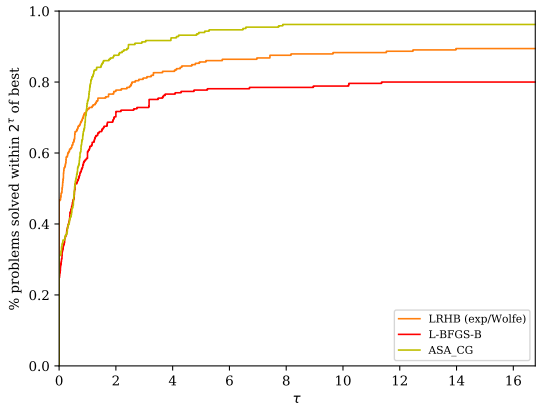

Figure 4: Performance profile of function evaluations for L-RH-B and L-BFGS-B on the unconstrained CUTEst test problems.

These profiles show that L-RH-B is better than lbfgsb but not as good as ASA for bound-constrained problems. For unconstrained, we are again still better than lbfgsb but not very robust. We are getting a lot of line search failures. We probably won't include the time profiles since only about 15 problems take more than 2 seconds. Most are in the 10^{-2} range.

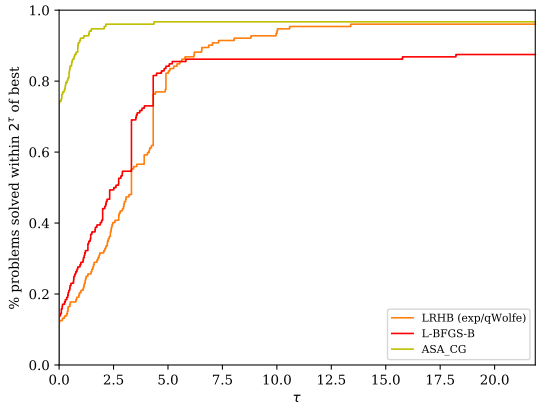

Figure 5: Performance profile of wall time for L-RH-B and L-BFGS-B on the bound-constrained CUTEst test problems.

References

[1] D. P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Trans. Automatic Control*, AC-21(2):174–184, 1976. 2, 11

[2] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Computer Science and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1982. 7, 11

[3] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM J. Control Optim.*, 20(2):221–246, 1982. 2

[4] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16:1190–1208, 1995. 20

[5] P. H. Calamai and J. J. Moré. Projected gradient methods for linearly constrained problems. *Math. Program.*, 39:93–116, 1987. 2

[6] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comput.*, 30:772–795, 1976. 6, 13, 14

[7] J. E. Dennis, Jr. and R. B. Schnabel. A new derivation of symmetric positive definite secant updates. In *Nonlinear Programming, 4 (Proc. Sympos., Special Interest Group on Math. Programming, Univ. Wisconsin, Madison, Wis., 1980)*, pages 167–199. Academic Press, New York, 1981. 6

[8] E. D. Dolan and J. J. Moré. Benchmarking optimization software with COPS. Technical Memorandum ANL/MCS-TM-246, Argonne National Laboratory, Argonne, IL, 2000. 18

[9] M. C. Fenelon. *Preconditioned Conjugate-Gradient-Type Methods for Large-Scale Unconstrained Optimization*. PhD thesis, Department of Operations Research, Stanford University, Stanford, CA, 1981. 2, 4

[10] M. W. Ferry. *Projected-Search Methods for Box-Constrained Optimization*. PhD thesis, Department of Mathematics, University of California, San Diego, May 2011. 13, 15

[11] M. W. Ferry, P. E. Gill, E. Wong, and M. Zhang. Projected-search methods for bound-constrained optimization. Center for Computational Mathematics Report CCoM 20-01, Center for Computational Mathematics, University of California, San Diego, La Jolla, CA, 2020. 3, 11, 13, 16, 18

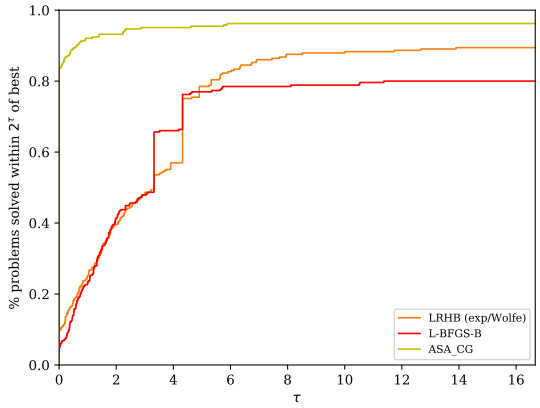

Figure 6: Performance profile of wall time for L-RH-B and L-BFGS-B on the unconstrained CUTEst test problems.

[12] R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168, 1963. 4

[13] P. E. Gill and M. W. Leonard. Limited-memory reduced-Hessian methods for large-scale unconstrained optimization. *SIAM J. Optim.*, 14:380–401, 2003. 2, 3, 5, 6, 7, 13, 15

[14] A. A. Goldstein. Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, 70(5):709–710, 1964. 2

[15] W. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIAM J. Optim.*, 17(2):526–557, 2006. 20

[16] D. Kim, S. Sra, and I. S. Dhillon. Tackling box-constrained optimization via a new projected quasi-Newton approach. *SIAM J. Sci. Comput.*, 32(6):3548–3563, December 2010. 2

[17] D. Knuth. *The Art of Computer Programming, 3.* Addison-Wesley Publishing Company, Redwood City, third edition, 1997. 12

[18] M. W. Leonard. *Reduced Hessian Quasi-Newton Methods for Optimization.* PhD thesis, Department of Mathematics, University of California, San Diego, 1995. 2

[19] E. S. Levitin and B. T. Polyak. Constrained minimization methods. *U.S.S.R. Comput. Math. and Math. Physics*, 6(5):1–50, 1966. 2

[20] J. L. Morales and J. Nocedal. Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Trans. Math. Softw.*, 38(1):7:1–7:4, December 2011. 20

[21] Q. Ni and Y. Yuan. A subspace limited memory quasi-Newton algorithm for large-scale nonlinear bound constrained optimization. *Math. Comput.*, 66:1509–1520, 10 1997. 2

[22] D. Siegel. Modifying the BFGS update by a new column scaling technique. *Math. Program.*, 66:45–78, 1994. Ser. A. 2, 4, 5, 15

[23] J. W. J. Williams. Algorithm 232 - Heapsort. *Communications of the Association for Computing Machinery*, 7:347–348, 1964. 12