MSS: MATLAB SOFTWARE FOR L-BFGS TRUST-REGION SUBPROBLEMS FOR LARGE-SCALE OPTIMIZATION

JENNIFER B. ERWAY AND ROUMMEL F. MARCIA

Abstract. A MATLAB implementation of the Moré-Sorensen sequential (MSS) method is presented. The MSS method computes the minimizer of a quadratic function defined by a limited-memory BFGS matrix subject to a two-norm trust-region constraint. This solver is an adaptation of the Moré-Sorensen direct method into an L-BFGS setting for large-scale optimization. The MSS method makes use of a recently proposed stable fast direct method for solving large shifted BFGS systems of equations [9, 8]. This MATLAB implementation is a matrix-free iterative method for large-scale optimization. Numerical experiments show that the MSS method is able to compute solutions to high accuracy.

1. Introduction

In this paper we describe a MATLAB implementation for minimizing a quadratic function defined by a limited-memory BFGS (L-BFGS) matrix subject to a two-norm constraint, i.e., for a given x_k,

$$\underset{p\in\Re^n}{\text{minimize}}\ \ \mathcal{Q}(p) \triangleq g^Tp + \frac{1}{2}p^TBp \quad \text{subject to} \quad \|p\|_2 \leq \delta, \tag{1}$$

where $g \triangleq \nabla f(x_k)$, B is an L-BFGS approximation to $\nabla^2 f(x_k)$, and δ is a given positive constant. Approximately solving (1) is of interest to the optimization community, as it is the computational bottleneck of trust-region methods for large-scale optimization. Generally speaking, there is a trade-off in computational cost per subproblem and the number of overall trust-region iterations (i.e., function and gradient evaluations): The more accurate the subproblem solver, the fewer overall iterations required. Solvers that reduce the overall number of function and gradient evaluations are of particular interest when function (or gradient) evaluations are time-consuming, e.g., simulation-based optimization. This paper presents an algorithm to solve (1) to any user-defined accuracy.

The earliest quasi-Newton methods to solve (1) in a trust-region context were designed for symmetric positive-definite Hessian approximations. These methods used the trust region only to restrict the length of the step (see, e.g., [18, 17, 7, 11]) and did not seek to solve

Date: December 7, 2012.

Key words and phrases. Large-scale unconstrained optimization, trust-region methods, limited-memory quasi-Newton methods, L-BFGS.

J. B. Erway was supported in part by National Science Foundation grant DMS-0811106. R. F. Marcia was supported in part by National Science Foundation grant DMS-0965711.

(1) to high accuracy. In [18, 17], Powell proposes a quasi-Newton trust-region method that accepts the Cauchy point p_c, the solution of the minimization problem

$$\underset{p,\alpha}{\text{minimize}} \quad \{\mathcal{Q}(p) : p = -\alpha g\},$$

as the approximate solution of (1) whenever $\|p_c\|_2 > \delta$. If the Cauchy point satisfies $\|p_c\|_2 \leq \delta$, Powell computes the *quasi-Newton step* $p_{qN} \triangleq -Bg$, and if the quasi-Newton satisfies $\|p_{qN}\|_2 \leq \delta$, this step is taken as the approximate solution of (1). If this second test is not satisfied, the approximate solution of the minimization problem is chosen to be a convex combination of the Cauchy step and the quasi-Newton step that lies on the boundary $\|p\|_2 = \delta$. This method is often referred to as a "dogleg" strategy. In [7], Dennis and Mei propose two modifications: They exchange the order of Powell's tests of the Cauchy step and the quasi-Newton step, and they modify the dogleg strategy into a double-dogleg strategy that biases the approximate minimizer more in the favor of the quasi-Newton direction. In [11], Kauffman extends the double-dogleg strategy to a limited-memory framework. These methods require products with both B and B^{-1}, which are accomplished by updating a matrix factorization at each step. However, these methods only *approximately* solve (1); when the solution lies on the boundary, none of these methods seek to solve the minimization problem to high accuracy.

Methods to solve (1) to high accuracy are often based on optimality conditions given in the following theorem (see, e.g., Gay [10], Sorensen [19], Moré and Sorensen [14] or Conn, Gould and Toint [5]):

Theorem 1. *Let δ be a positive constant. A vector p^* is a global solution of the trust-region subproblem (1) if and only if $\|p^*\|_2 \leq \delta$ and there exists a unique $\sigma^* \geq 0$ such that $B + \sigma^* I$ is positive semidefinite and*

$$(B + \sigma^* I)p^* = -g \quad \text{and} \quad \sigma^*(\delta - \|p^*\|_2) = 0. \tag{2}$$

Moreover, if $B + \sigma^* I$ is positive definite, then the global minimizer is unique.

The Moré-Sorensen algorithm [14] seeks (p^*, σ^*) that satisfy the optimality conditions (2) by trading off between updating p and σ. That is, each iteration, the method updates p (fixing σ) by solving the linear system $(B + \sigma I)p = -g$ using the Cholesky factorization of the $B + \sigma I$; then, σ is updated using a safeguarded Newton method to find a root of

$$\phi(\sigma) \triangleq \frac{1}{\|p(\sigma)\|_2} - \frac{1}{\delta}. \tag{3}$$

The Moré-Sorensen direct method is arguably the best direct method for solving the trust-region subproblem; in fact, the accuracy of each solve can be specified by the user. While this method is practical for smaller-sized problems, in large-scale optimization it is too computationally expensive to compute and store Cholesky factorizations for unstructured Hessians.

In Burke et al. [3], the authors propose an adaption of Moré-Sorensen method to the limited-memory BFGS setting. They begin by computing the quasi-Newton step $p_{qN} \triangleq -Bg$.

If p_{qN} satisfies the two-norm inequality constraint, the problem is solved; otherwise, Burke et al. find a solution $\|p\|_2$ that lies on the boundary of the trust region using More-Sorensen's method by computing a pair $(p(\sigma), \sigma)$ such that

$$(B + \sigma I)p(\sigma) = -g \quad \text{and} \quad \sigma(\delta - \|p(\sigma)\|_2) = 0. \tag{4}$$

similar to the original direct method. The solution of the second equation in (4) is computed using two Choleksy factorizations of $M \times M$ matrices, where M is the number of limited-memory updates. The method is derived using the Sherman-Morrison-Woodbury formula. While this technique is able to exploit properties of L-BFGS updates, there are potential instability issues related to their proposed use of the Sherman-Morrison-Woodbury that are not addressed.

Lu and Monteiro [13] also explore a Moré-Sorensen method implementation when B has special structure; namely, $B = D + VEV^T$, where D and E are positive diagonal matrices, and V has a small number of columns. Their approach uses the Sherman-Morrison-Woodbury formula to replace solves with $(B + \sigma I)$ with solves with an $M \times M$ system composed of a diagonal plus a low rank matrix. Thus, this method is able to avoid computing Choleksy factorizations. Like with [3], there are potential stability issues that are not addressed regarding inverting the $M \times M$ matrix.

Finally, Apostolopoulou et al. [2, 1] derive a closed-form expression for $(B + \sigma I)^{-1}$ to solve the first equation in (4). The authors are able to explicitly compute the eigenvalues of B, provided $M = 1$ [2, 1] or $M = 2$ [1]. While their formula avoids potential instabilities associated the Sherman-Morrison-Woodbury formula, their formula is restricted to the case when the number of updates is at most two.

1.1. Overview of the proposed methods.

In this paper, we describe a new adaptation of the Moré-Sorensen solver into a large-scale L-BFGS setting. The proposed method, called the *Moré-Sorensen sequential (MSS) method*, is able to exploit the structure of BFGS matrices to solve the shifted L-BFGS system in (2) using a fast direct recursion method that the authors originally proposed in [9]. (This recursion was later proven to be stable in [8].) The MSS method is able to solve (1) to any prescribed accuracy.

The paper is organized in five sections. In Section 2 we review L-BFGS quasi-Newton methods and introduce notation that will be used for the duration of this paper. Section 4 includes numerical results comparing the Moré-Sorensen method and the MSS method. Finally, Section 5 includes some concluding remarks and observations.

1.2. Notation and Glossary.

Unless explicitly indicated, $\|\cdot\|$ denotes the vector two-norm or its subordinate matrix norm. In this paper, all methods use L-BFGS updates and we assume they are selected to ensure the quasi-Newton matrices remain sufficiently positive definite.

2. Background

In this section, we begin with an overview of the L-BFGS quasi-Newton matrices described by Nocedal [15], defining notation that will be used throughout the paper.

The L-BFGS quasi-Newton method generates a sequence of positive-definite matrices $\{B_j\}$ from a sequence of vectors $\{y_j\}$ and $\{s_j\}$ defined as

$$y_j = \nabla f(x_{j+1}) - f(x_j) \quad \text{and} \quad s_j = x_{x+1} - x_j,$$

where $j = 0, \ldots m-1$ where $m \leq M$, and M is the maximum number of allowed stored pairs (y_j, s_j). This method can be viewed as the BFGS quasi-Newton method where no more than the M most recently computed updates are stored and used to update an initial matrix B_0. The L-BFGS quasi-Newton approximation to the Hessian of f is implicitly updated as follows:

$$B_m = B_0 - \sum_{i=0}^{m-1} a_i a_i{}^T + \sum_{i=0}^{m-1} b_i b_i^T, \tag{5}$$

where

$$a_i = \frac{B_i s_i}{\sqrt{s_i^T B_i s_i}}, \quad b_i = \frac{y_i}{\sqrt{y_i^T s_i}}, \quad B_0 = \gamma_m^{-1} I, \tag{6}$$

and $\gamma_m > 0$ is a constant. In practice, γ_m is often defined to be $\gamma_m \triangleq s_{m-1}^T y_{m-1} / \|y_{m-1}\|^2$ (see, e.g., [12] or [15]). In order to maintain that the sequence $\{B_i\}$ is positive definite for $i = 1, \ldots m$, each of the accepted pairs must satisfy $y_i^T s_i > 0$ for $i = 0, \ldots, m-1$.

Suppose that we have computed m updates ($m \leq M$) and have the following updates stored in S and Y:

$$S = [s_0 \ldots s_{m-1}] \quad \text{and} \quad Y = [y_0 \ldots y_{m-1}].$$

The matrices S and Y are updated with the most recently computed vector pair (s_m, y_m) as follows:

Algorithm 2.1: Update S and Y.

```
if $m < M$,
    $S \leftarrow [S \;\; s_m]$;  $Y \leftarrow [Y \;\; y_m]$;  $m \leftarrow m+1$;
else
    for $i = 0, \ldots m-1$
        $s_i \leftarrow s_{i+1}$;  $y_i \leftarrow y_{i+1}$;
    end
    $S \leftarrow [s_0, \ldots s_{m-1}]$;  $Y \leftarrow [y_0, \ldots y_{m-1}]$;
end
```

Thus, at all times we have exactly m stored vectors with $m \leq M$.

One of the advantages of using an L-BFGS quasi-Newton is that there is an efficient recursion relation to compute products with B_m^{-1}. Given a vector z, the following algorithm [15, 16] terminates with $r \triangleq B_m^{-1} z$:

Algorithm 2.2: Two-loop recursion to compute $r = B_m^{-1}z$.

```
$q \leftarrow z$;
for $k = m-1, \ldots, 0$
    $\rho_k \leftarrow 1/(y_k^T s_k)$;
    $\alpha_k \leftarrow \rho_k s_k^T q$;
    $q \leftarrow q - \alpha_k y_k$;
end
$r \leftarrow B_0^{-1} q$;
for $k = 0, \ldots, m-1$
    $\beta \leftarrow \rho_k y_k^T r$;
    $r \leftarrow r + (\alpha_k - \beta) s_k$:
end
```

The two-term recursion formula requires at most $\mathcal{O}(Mn)$ multiplications and additions. To compute products with the L-BFGS quasi-Newton matrix, one may use the so-called "unrolling" formula, which requires $\mathcal{O}(M^2n)$ multiplications, or one may use a compact matrix representation of the L-BFGS that can be used to compute products with the L-BFGS quasi-Newton matrix, which requires $\mathcal{O}(Mn)$ multiplications (see, e.g., [16]). Further details on L-BFGS updates can found in [16]; further background on the BFGS updates can be found in [6].

Without loss of generality, for the duration of the paper we assume that B is a symmetric positive-definite quasi-Newton matrix formed using m ($m \leq M$) L-BFGS updates.

3. The MSS method

In this section, we present the MSS method to solve the constrained optimization problem (1). We begin by considering the Moré-Sorensen direct method proposed in [14].

The Moré-Sorensen direct method seeks a pair (p, σ) that satisfy the optimality conditions (2) by alternating between updating p and σ. In the case that the B is positive definite (as in L-BFGS matrices), the method simplifies to Algorithm 3.1 [14].

Algorithm 3.1: Moré-Sorensen Method.

```
$\sigma \leftarrow 0$;  $p \leftarrow -B^{-1}g$;
if $\|p\| \leq \delta$
    return;
else
    while not converged do
        Factor $B + \sigma I = R^T R$;
        Solve $R^T R p = -g$;
        Solve $R^T q = p$;
        $\sigma \leftarrow \sigma + \frac{\|p\|^2}{\|q\|^2} \frac{\|p\| - \delta}{\delta}$;
    end do
end
```

The update to σ in Algorithm 3.1 can be shown to be Newton's method applied (3). (Since B is positive definite, a safeguarded Newton's method unnecessary.) Convergence is predicated on solving the optimality conditions (2) to a prescribed accuracy. The only difficulty in implementing the Moré-Sorensen method in a large-scale setting is the shifted solve $(B+\sigma I)p = -g$.

One method to directly solve systems of the form $(B+\sigma I)x = y$ is to view the system matrix as the sum of σI and rank-one L-BFGS updates to an initial diagonal matrix B_0. It is important distinguish between *applying* rank-one L-BFGS updates to $B_0+\sigma I$ and *viewing* the system matrix as the sum of rank-one updates to $B_0+\sigma I$. To compute $(B+\sigma I)^{-1}y$, one cannot simply substitute $B_0+\sigma I$ in for B_0 in Algorithm (2.2). (For a discussion on this, see [9]). In [9], Erway and Marcia present a stable fast direct method for solving L-BFGS systems that are shifted by a constant diagonal matrix (stability is shown in [8]). Specifically, it is shown that products with $(B+\sigma I)^{-1}$ can be computed provided $\gamma\sigma$ is bounded away from zero, where $B_0 \triangleq \gamma^{-1}I$. The following theorem is found in [9]:

Theorem 2. *Let $\gamma > 0$ and $\sigma \geq 0$. Suppose $G = (\gamma^{-1}+\sigma)I$, and let $H = \sum_{i=0}^{2m-1} E_i$, where*

$$E_0 = -a_0a_0^T, \;\; E_1 = b_0b_0^T, \;\; \ldots, \;\; E_{2m-2} = -a_{m-1}a_{m-1}^T, \;\; E_{2m-1} = b_{m-1}b_{m-1}^T,$$

with $\{a_i\}$ and $\{b_i\}$ are defined as in (6). Further, define $C_{m+1} = G + E_0 + \cdots + E_m$. If there exists some $\epsilon > 0$ such that $\gamma\sigma > \epsilon$, then $B+\sigma I = G + H$, and $(B+\sigma I)^{-1}$ is given by

$$(B+\sigma I)^{-1} = C_{2m-1}^{-1} - v_{2m-1}C_{2m-1}^{-1}E_{2m-1}C_{2m-1}^{-1}$$

where

$$C_{i+1}^{-1} = C_i^{-1} - v_iC_i^{-1}E_iC_i^{-1}, \;\; i = 0, \ldots, 2m-1,$$

and

$$v_i = \frac{1}{1+\operatorname{trace}\left(C_i^{-1}E_i\right)}.$$

Proof. See [9]. $\square$

This theorem is the basis for the following algorithm derived in [9] to compute products with $(B+\sigma I)^{-1}$. The algorithm was shown to be stable in [8].

Algorithm 3.2: Recursion to compute $x = (B+\sigma I)^{-1}y$.

```
$x \leftarrow (\gamma^{-1}+\sigma)^{-1}y$;
for $k = 0, \ldots, 2m-1$
    if $k$ even
        $c \leftarrow a_{k/2}$;
    else
        $c \leftarrow b_{(k-1)/2}$;
    **end**
    $r_k \leftarrow (\gamma^{-1} + \sigma)^{-1} c$;
    **for** $i = 0, \ldots, k-1$
        $r_k \leftarrow r_k + (-1)^i v_i (r_i^T c) r_i$;
    **end**
    $v_k \leftarrow 1/(1 + (-1)^{k+1} r_k^T c)$;
    $x \leftarrow x + (-1)^k v_k (r_k^T y) r_k$;
**end**
```

It is important to note that $\{a_i\}$ and $\{b_i\}$ need to be precomputed. Algorithm 3.2 requires at most $\mathcal{O}(M^2 n)$. Operations with $C_0 = B_0 + \sigma I$ and C_1 can be easily computed with minimal extra expense since C_0^{-1} is a diagonal matrix. It is generally known that M may be kept small (for example, Byrd et al. [4] suggest $M \in [3, 7]$). When $M^2 \ll n$, the extra storage requirements and computations are affordable.

3.1. **Handling small σ.** In this section we discuss the case of solving $(B + \sigma I)p = -g$ for small σ. For stability, it is important to maintain $\gamma\sigma > \epsilon_\sigma$ for a small positive ϵ_σ. Thus, in order to use Algorithm 3.2, we require that both $\gamma > \sqrt{\epsilon_\sigma}$ and $\sigma > \sqrt{\epsilon_\sigma}$. The first requirement is easily met by thresholding γ, i.e., $\gamma \leftarrow \min\{\sqrt{\epsilon_\sigma}, \gamma\}$. (Recall that $y_i^T s_i > 0$ for each $i = 0, \ldots m-1$, and thus, $\gamma > 0$.) When $\sigma \leq \sqrt{\epsilon_\sigma}$, we set $\sigma = 0$ and use the two-loop recursion (Algorithm 2.2) to solve an unshifted L-BFGS system.

3.2. **The algorithm.** The MSS method adapts the Moré-Sorensen method into an L-BFGS setting by solving $(B + \sigma I)p = -g$ using a recursion method instead of a Cholesky factorization. When σ is sufficiently large, the recently proposed recursion algorithm (Algorithm 3.2) is used to update s; otherwise, $\sigma \approx 0$ and the two-loop recursion (Algorithm 2.2) is used. Also, note that the Moré-Sorensen method updates σ using Newton's method applied to (3), i.e.,

$$\sigma \leftarrow \sigma - \phi(p)/\phi'(p). \tag{7}$$

In Algorithm 3.1 this update is written in terms of the Cholesky factors. In the MSS algorithm, Cholesky factors are unavailable, and thus, the update to σ is performed by explicitly computing the Newton step (7). For details on this update see [5].

The MSS method is summarized in Algorithm 3.3.

Algorithm 3.3: MSS Method.

```
Specify $\gamma > \sqrt{\epsilon_\sigma}$ where $0 < \epsilon_\sigma \ll 1$;
$\sigma \leftarrow 0$;  $p \leftarrow -B^{-1} g$;
**if** $\|p\| \leq \delta$
    **return**;
**else**
    **while** *not* converged **do**
        $\phi(p) \leftarrow 1/\|p\| - 1/\delta$;
```

```
            if $\sigma > \sqrt{\epsilon_\sigma}$
                Compute $\hat{p}$ such that $(B + \sigma)\hat{p} = -p$ using Algorithm 3.2;
            else
                Compute $\hat{p}$ such that $B\hat{p} = -p$ using Algorithm 2.2;
            end
            $\phi'(p) \leftarrow -(p^T\hat{p})/\|p\|^3$;
            $\sigma \leftarrow \sigma - \phi(p)/\phi'(p)$;
            if $\sigma > \sqrt{\epsilon_\sigma}$
                Compute $p$ such that $(B + \sigma)p = -g$ using Algorithm 3.2;
            else
                Compute $p$ such that $Bp = -g$ using Algorithm 2.2;
            end
        end
end
```

3.3. **Stopping criteria.** When the solution to (1) lies on the constraint boundary $\|p\| = \delta$, it is reasonable to accept p when $\|p\|$ is sufficiently close to δ. Thus, for a fixed $\tau \ll 1$, the MSS method tests

$$|\|p\| - \delta| \le \tau\delta, \tag{8}$$

for convergence on the constraint boundary. This test may also be used in addition to the test $\|p\| \le \delta$ for the quasi-Newton step $p_{qN} = -B^{-1}g$, making the condition

$$\|p\| \le \delta \qquad \text{or} \qquad |\|p\| - \delta| \le \tau\delta. \tag{9}$$

4. Numerical Results

In this section, we test the accuracy of the proposed Moré-Sorensen sequential (MSS) method. Numerical results were obtained using MATLAB implementations of the MSS and Moré-Sorensen methods. For all experiments, vector pairs $\{(s_i, y_i)\}$, $i = 0, \ldots m-1$, $(m < M)$, were randomly generated by computing two $m \times 1$ vectors S and Y such that whenever $s_i^T y_i < 0$, the sign of s_i was flipped, i.e., $s_i \leftarrow -s_i$. Thus, the sequence of L-BFGS matrices were positive definite. For each experiment, g and δ were also randomly generated. The number of limited-memory updates m was set to 5. Finally, the lower bound on $\gamma\sigma$ was taken to be $\epsilon_\sigma \triangleq \epsilon$, where ϵ is machine precision.

For the numerical experiments, the stopping criteria (8) and (9) were used with

$$\tau \triangleq \min\{\|g\| * 10^{-5}, \sqrt{\epsilon}\},$$

where ϵ is machine precision. For each problem, no more than 500 iterations were allowed for either MSS method and Moré-Sorensen method.

In the first set of experiments, random problems of size n=100, 500, 1000, 2000, and 5000 were generated. The results of both solvers are presented in Table 1. For each problem,

Table 1. Comparison of results from the Moré-Sorensen (MS) method and the Moré-Sorensen Sequential (MSS) method on small problems.

n	δ	γ	$\|g\|$	τ	MS error	MSS error
100	6.83e-01	1.04e-01	9.79e+00	1.49e-08	6.17e-14	2.21e-14
500	6.98e-02	3.77e-02	2.28e+01	1.49e-08	3.57e-14	1.69e-14
1000	4.12e-01	2.82e-02	3.23e+01	1.49e-08	4.54e-07	1.62e-07
2500	3.86e-01	2.91e-02	4.99e+01	1.49e-08	4.37e-07	1.85e-07
5000	1.93e-01	5.41e-03	7.18e+01	1.49e-08	1.04e-07	3.49e-08

Table 2. Moré-Sorensen Sequential (MSS) method on large problems.

n	δ	γ	$\|g\|$	τ	MSS error
10000	9.03e-01	1.96e-02	1.00e+02	1.49e-08	1.30e-09
50000	9.37e-01	2.37e-03	2.22e+02	1.49e-08	1.83e-11
100000	5.34e-01	3.74e-03	3.15e+02	1.49e-08	1.24e-07
500000	5.00e-01	2.01e-03	7.06e+02	1.49e-08	2.57e-12
1000000	6.35e-01	7.08e-04	1.00e+03	1.49e-08	1.39e-12

$\delta, \gamma, \|g\|$, and τ are given. The reported errors for each problem is the sum of the errors in the optimality conditions, i.e.,

$$\text{error} = \|(B + \sigma^* I)p + g\| + |\sigma^*(\delta - \|p^*\|)| \,. \tag{10}$$

In addition to storage limitations in memory and because of the time requirements needed to compute Cholesky factorizations, the Moré-Sorensen method was not tested on problems with $n > 5000$. Both methods were able to solve all constrained optimization problems to the required tolerance. (Note that convergence depends on τ, but the sum of the errors in the optimality conditions do not have to be less than τ.)

Larger values of n were chosen for the second set of experiments. We tested the MSS method on five random problems of large size and computed the error as in (10). Table 2 reports these results. In all cases, the MSS method was able to solve the optimization problem to the required tolerance.

5. Concluding Remarks

In this paper, we have presented a MATLAB implementation of the MSS algorithm that is able to solve problems of the form (1). This solver is stable and numerical results confirm that the method can compute solutions to any prescribed accuracy. Future research includes implementing the MSS method in a trust-region algorithm for large-scale optimization.

References

[1] M. S. Apostolopoulou, D. G. Sotiropoulos, C. A. Botsaris, and P. E. Pintelas. A practical method for solving large-scale TRS. *Optimization Letters*, 5:207–227, 2011.

[2] M. S. Apostolopoulou, D. G. Sotiropoulos, and P. Pintelas. Solving the quadratic trust-region subproblem in a low-memory bfgs framework. *Optimization Methods Software*, 23(5):651–674, Oct. 2008.

[3] J. V. Burke, A. Wiegmann, and L. Xu. Limited memory bfgs updating in a trust-region framework. Technical report, University of Washington, 1996.

[4] 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.

[5] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.

[6] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. Corrected reprint of the 1983 original.

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

[8] J. B. Erway, V. Jain, and R. F. Marcia. Shifted l-bfgs systems. Technical Report 2012-6, Wake Forest University, 2012.

[9] J. B. Erway and R. F. Marcia. Limited-memory bfgs systems with diagonal updates. *Linear Algebra and its Applications*, 437(1):333 – 344, 2012.

[10] D. M. Gay. Computing optimal locally constrained steps. *SIAM J. Sci. Statist. Comput.*, 2(2):186–197, 1981.

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

[12] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45:503–528, 1989.

[13] Z. Lu and R. D. C. Monteiro. A modified nearly exact method for solving low-rank trust region subproblem. *Math. Program.*, 109(2):385–411, Jan. 2007.

[14] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM J. Sci. and Statist. Comput.*, 4:553–572, 1983.

[15] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comput.*, 35:773–782, 1980.

[16] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, second edition, 2006.

[17] M. J. D. Powell. A fortran subroutine for solving systems of nonlinear algebraic equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*. Gordon and Breach, 1970.

[18] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*, pages 87–114. Gordon and Breach, 1970.

[19] D. C. Sorensen. Newton's method with a model trust region modification. *SIAM J. Numer. Anal.*, 19(2):409–426, 1982.

E-mail address: `erwayjb@wfu.edu`

Department of Mathematics, Wake Forest University, Winston-Salem, NC 27109

E-mail address: `rmarcia@ucmerced.edu`

Appied Mathematics, University of California, Merced, Merced, CA 95343