

# LIMITED MEMORY BFGS UPDATING IN A TRUST-REGION FRAMEWORK

JAMES V. BURKE   ANDREAS WIEGMANN   LIANG XU

ABSTRACT. The limited memory BFGS method pioneered by Jorge Nocedal is usually implemented as a line search method where the search direction is computed from a BFGS approximation to the inverse of the Hessian. The advantage of inverse updating is that the search directions are obtained by a matrix-vector multiplication. In this paper it is observed that limited memory updates to the Hessian approximations can also be applied in the context of a trust-region algorithm with only a modest increase in the linear algebra costs. At each iteration, a limited memory BFGS Step is computed. If it is rejected, then we compute the solutions for trust-region subproblem with the trust-region radius smaller than the length of L-BFGS Step. Numerical results on a few of the MINPACK-2 test problems show that the initial limited memory BFGS Step is accepted in most cases. In terms of the number of function and gradient evaluations, the trust-region approach is comparable to a standard linesearch implementation.

## 1. INTRODUCTION

In 1980 Nocedal [11] introduced limited memory BFGS (L-BFGS) updating for unconstrained optimization. Subsequent numerical studies on large-scale problems have shown that methods based on this updating scheme can be very effective if the updated inverse Hessian approximations are re-scaled at every iteration [6, 8, 10, 16]. The L-BFGS method performs well on many classes of problems and competes with truncated Newton methods on a variety of very large-scale nonlinear problems [10].

In this paper we an implementation that uses the L-BFGS update in a trust-region framework. The trust-region approach can substantially increase the per iteration linear algebra costs. The hope is that by more extensively exploiting the local information, a better step is produced thereby reducing the overall number of function and gradient evaluation required to obtain a comparable level of accuracy. For this reason we envision the trust-region approach to be most suitable for large-scale problems where the evaluation of the objective function dominates the computational cost. This often occurs in applications where the evaluation of the objective function requires the solution of an underlying system of differential equations, the solution of a sequence of auxiliary optimization problems, and/or simulation. Our numerical results show that on our chosen test set, the trust-region algorithm suffers from only a modest increase in the linear algebra costs, and on most iterations these costs are the same as for the line search based algorithm.

The L-BFGS method is a matrix secant method designed for low storage and linear algebra costs. The L-BFGS update is obtained by applying the BFGS update to an initial positive definite diagonal matrix (a scaling matrix) using data from a few of the most recent iterations. The update can then be stored either as a recursion [11, 10] or in a compact form [3] based on the Sherman-Morrison-Formula for matrix

---

*Date:* April 1, 2008.

This research was supported by National Science Foundation Grant DMS-9303772.

inversion. The search direction is computed by a simple matrix vector multiplication. Changes in the initial scaling matrix and the data from past iterations can be introduced into the update at each iteration and at low cost. This is especially true if the initial scaling matrix is a multiple of the identity as is usual in practice. Once the search direction is computed, an appropriate step-length is obtained from one of the standard line search procedures. However, in practice a line search is rarely required if one re-scales at each iteration using the scaling (2.4) suggested in [6, 8, 10, 15, 16].

An important difference between the L-BFGS method described above and a trust-region strategy is that trust-region methods require the maintenance of an approximate Hessian, not its inverse. In a trust-region approach the next iterate is obtained by solving an sequence of equations rather than by matrix multiplication. Consequently, the per iteration the linear algebra costs can be much greater. However, our limited numerical experience indicates that this may not be the case.

## 2. TRUST-REGIONS AND THE L-BFGS UPDATE

Consider the problem of minimizing the smooth function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  over  $\mathbb{R}^n$ . There are two main difference between the approach we take and a standard trust-region algorithm. At each iteration, we compute the full L-BFGS Step, if the trial step accepted, we update the iterates; otherwise In the sequel, let  $x^\nu \in \mathbb{R}^n$  be the current iterate,  $g^\nu = \nabla f(x^\nu)$ ,  $H_\nu \approx \nabla^2 f(x^\nu)$ , and  $0 < \Delta \leq \infty$  is the trust-region radius. The step to the next iterate  $\bar{s}$  is obtained by solving subproblems of the form

$$(2.1) \quad \begin{aligned} \mathcal{P}(g^\nu, H_\nu, \Delta) : \quad & \text{minimize } g^T s + \frac{1}{2} s^T H_\nu s \\ & \text{subject to } \|s\| \leq \Delta, \end{aligned}$$

where the standard Euclidean norm used. In order to simplify the notation, we often suppress the iteration index  $\nu$ . The step  $\bar{s}$  is either accepted or rejected and the trust-region radius is kept, or decreased depending on the value of the ratio

$$(2.2) \quad r_\nu(\bar{s}) = \frac{f(x^\nu + \bar{s}) - f(x^\nu)}{g^T \bar{s} + \frac{1}{2} \bar{s}^T H_\nu \bar{s}}.$$

We now describe how the updates  $H_\nu$  are computed. Let  $\{x^\nu\}$  be a sequence generated by the algorithm given above and define the corresponding sequences  $\{s^\nu\}$  and  $\{y^\nu\}$  by  $s^\nu = x^{\nu+1} - x^\nu$  and  $y^\nu = g^{\nu+1} - g^\nu$  for  $\nu \geq 1$ . Given an integer memory length  $m$  (typically  $m = 5$ ), and select  $m$  of the most recent iterates  $\nu_1 < \nu_2 < \dots < \nu_m$  for which  $(s^{\nu_j})^T y^{\nu_j} > 0$ ,  $j = 1, \dots, m$ . These vectors form the basis for our L-BFGS update. Set

$$S = [s^{\nu_1}, \dots, s^{\nu_m}], \quad Y = [y^{\nu_1}, \dots, y^{\nu_m}], \quad \text{and } S^T Y = L + D + R,$$

where  $L$  is strictly lower triangular,  $D$  is positive definite diagonal, and  $R$  is strictly upper triangular. Let  $\lambda > 0$ . Then the L-BFGS update at  $x^\nu$  with initial Hessian approximation  $H_{\nu_0} := \lambda I$  is given by

$$(2.3) \quad H_\nu = \lambda I - \Psi \Gamma^{-1} \Psi^T$$

where

$$\Psi = \begin{bmatrix} \lambda S & Y \end{bmatrix} \quad \text{and} \quad \Gamma = \begin{bmatrix} -D & L^T \\ L & \lambda S^T S \end{bmatrix}.$$

The invertibility of  $\Gamma$  follows from the positive definiteness of the diagonal matrix  $D$ . This form of the L-BFGS update is derived in Byrd, Nocedal, and Schnabel [3, Theorem 2.3]. The condition  $(s^\nu)^T y^\nu > 0$ , can always be assured by using a line search that terminates on either the weak or strong Wolfe conditions. In this case, one can choose  $\nu_1, \dots, \nu_m$  to be the  $m$  most recent iterates. On the other hand, in a trust-region setting this condition may not always be satisfied.

For  $\lambda$ , we use the scaling

$$(2.4) \quad \lambda = (y^{\nu_m})^T y^{\nu_m} / (s^{\nu_m})^T y^{\nu_m}$$

suggested by Shanno and Phua [15]. This scaling appears as one of a class of optimal scalings first proposed in Oren and Spedicato [12]. In practice, this choice of scaling has a profound impact on the performance of the L-BFGS method [6, 8, 10, 15, 16]. Note that if we take  $m = 0$ , then the scaling (2.4) gives rise to the Barzilai-Borwein scaled gradient step [7, 5].

To apply a trust-region strategy, we must be able to efficiently solve equations of the form

$$(2.5) \quad (\mu I + H)s = -g,$$

for a trial step  $s$  where  $\mu \geq 0$  is a Lagrange multiplier estimate for the trust-region constraint,  $g = g^\nu$ , and  $H = H_\nu$ . Moreover, we may have to solve this equation for several values of  $\mu$  for each value of  $\Delta$ . Note that

$$\mu I + H = \tau I - \Psi \Gamma^{-1} \Psi^T,$$

where  $\tau = \mu + \lambda$ . Using the Sherman–Morrison–Woodbury formula, the inverse of this matrix has the form

$$(2.6) \quad (\mu I + H)^{-1} = \frac{1}{\tau} [I + \Psi(\tau \Gamma - \Psi^T \Psi)^{-1} \Psi^T]$$

whenever the matrices  $(\mu I + H)$  and  $\Gamma$  are invertible (in which case  $(\tau \Gamma - \Psi^T \Psi)$  is also invertible).

To solve the trust-region subproblems, we apply the formula (2.6) to compute the step  $s$  in (2.5). This requires the solution of one or more  $2m \times 2m$  matrix equations involving the matrix  $(\tau \Gamma - \Psi^T \Psi)$  for various values of  $\tau$ . The predominant cost in this approach is the formation of the vector  $\Psi^T g^\nu$ , the matrix  $(\tau \Gamma - \Psi^T \Psi)$ , and the final recovery of the solution by multiplication with the matrix  $\Psi$ . Observe that

$$(2.7) \quad (\tau \Gamma - \Psi^T \Psi) = \begin{bmatrix} -(Y^T Y + (\lambda + \mu)D) & \mu L^T - \lambda(D + R)^T \\ \mu L - \lambda(D + R) & \mu \lambda S^T S \end{bmatrix},$$

where  $(\mu + \lambda)D + Y^T Y$  is positive definite. Let  $M$  be a lower triangular matrix (Cholesky factor) such that

$$(\mu + \lambda)D + Y^T Y = M M^T.$$

Set

$$\hat{L} = \mu L - \lambda(R + D), \quad \hat{D} = (\mu + \lambda)D + Y^T Y, \quad \text{and} \quad \hat{W} = \lambda \mu S^T S.$$

**Proposition 2.1.** *The  $m \times m$  matrix*

$$(2.8) \quad \hat{W} + \hat{L} \hat{D}^{-1} \hat{L}^T$$

*is positive definite.*

*Proof.* Clearly  $\hat{W}$  and  $\hat{L}\hat{D}^{-1}\hat{L}^T$  are positive semi-definite. If there exists a  $w$ , such that  $\hat{W}w = 0$  and  $\hat{L}\hat{D}^{-1}\hat{L}^T w = 0$ , then  $\hat{W}w = Sw = 0$  and  $\hat{L}^T w = 0$ . Therefore

$$\mu L^T w - \lambda(R^T + D^T)w = 0 \Rightarrow \mu(L^T + R^T + D^T)w = (\lambda + \mu)(D^T + R^T)w.$$

Since  $Sw = 0$ , we have  $Y^T S = 0$ , thus  $(\lambda + \mu)(D^T + R^T)w = 0$ , which gives  $w = 0$ .  $\square$

If we now let  $J$  be a lower triangular matrix (Cholesky factor) such that

$$JJ^T = \hat{W} + \hat{L}\hat{D}^{-1}\hat{L}^T,$$

then

$$\tau\Gamma - \Psi^T\Psi = \begin{bmatrix} M & 0 \\ -\hat{L}M^{-T} & J \end{bmatrix} \begin{bmatrix} -M^T & M^{-1}\hat{L}^T \\ 0 & J^T \end{bmatrix}.$$

That is, we can obtain a triangular factorization for  $\tau\Gamma - \Psi^T\Psi$  by computing two  $m \times m$  Cholesky factorizations.

This representation shows the increased computational load for the trust–region approach when  $\mu > 0$ . The parameter  $\mu$  is always zero in the Liu–Nocedal algorithm with line search. Hence the matrices  $S^T S$  and  $L$  are not computed and stored. This has important consequences for how we implement the L–BFGS update in a trust–region framework.

Our numerical experience indicates that the L–BFGS update implemented with the scaling (2.4) provides a step of such quality that a line search is rarely required. We exploit this observation by always testing the step  $\hat{s}$  computed with  $\mu = 0$ . If this step provides a sufficient decrease in the objective function, then we accept it and proceed to the next iteration. If the step is unacceptable, then we initiate a standard trust–region reduction strategy using the magnitude of  $\hat{s}$  as the initial trust–region radius. This deviates from standard trust–region implementations since we do not pass forward a trust–region parameter from one iteration to the next. Instead, we generate this parameter anew at the beginning of each iteration, thereby bringing the linear algebra costs more into line with those associated with a line search implementation.

By carefully organizing the computations and storing the appropriate information from one iteration to the next, one can show that the number of multiplications required to compute a step in the standard L–BFGS implementation is approximately  $(4m + 2)n$  (see [3]). In a trust–region implementation initialized with  $\mu = 0$  at each iteration, the first trial step is identical to the standard L–BFGS step so the number of multiplications required to compute this step is also  $(4m + 2)n$ . However, if this step is rejected, and a trial step with  $\mu > 0$  needs to be computed, then greater costs are incurred. There are two ways to handle this extra cost. One approach is to update the matrices  $S^T S$  and  $L$  at each iteration. This incurs an additional  $2mn$  multiplications per iteration with an additional  $2mn$  multiplications for each value of  $\mu > 0$  for which equation (2.5) is solved. In this approach, the total number of multiplications in iteration  $\nu$  is  $(6m + 2(t_\nu - 1)m + 1)n$  where  $t_\nu$  is the number of times equation (2.5) is solved on iteration  $\nu$ . However, we do not advocate this approach due to the infrequency with which the trial step with  $\mu = 0$  is rejected. Instead, we suggest that the matrices  $S^T S$  and  $L$  be updated only when they are required for the computation of a trial step with  $\mu > 0$ . This approach adds at most an additional  $(m^2 + 2(t_\nu - 1)m)n$  multiplications to the basic  $(4m + 2)n$  multiplications whenever  $t_\nu > 1$ . If the initial trial step with

$\mu = 0$  is rejected on a sequence of iterations, then the  $m^2n$  term disappears from all but the first iteration in this sequence with subsequent iterations in this sequence incurring a cost of  $(6m + 2(t_\nu - 1)m + 1)n$  multiplications where  $m$  is assumed to be small (usually  $m = 5$  in practise).

**Algorithm 1: Trust–Regions with L-BFGS Updating**

Initialization: Let  $x^0 \in \mathbb{R}^n$ ,  $g^0 = \nabla f(x^0)$ ,  $\lambda_0 > 0$ ,  $m = 0$ ,  $\bar{m} \in \{1, 2, \dots, n\}$ ,  $0 < \kappa < 1$ ,  $0 < \beta_\Delta < 1$ ,  $H_0$  symmetric and positive definite, and  $\nu = 0$ .

Iteration:

- (1) Let  $\bar{s}$  solve the trust–region subproblem  $\hat{\mathcal{P}}(g^\nu, H, 0)$ . If  $\bar{s} = 0$ , stop; otherwise, go to (2).
- (2) If the ratio  $r_\nu(\bar{s})$  given in (2.2) exceeds  $\kappa$ , set  $x^{\nu+1} = x^\nu + \bar{s}$  and go to (4); otherwise, set  $\Delta = \beta_\Delta \|\bar{s}\|$  and go to (3).
- (3) Let  $\bar{s}$  solve the trust–region subproblem  $\mathcal{P}(g^\nu, H, \Delta)$ , and go to (2).
- (4) Set  $m = \min\{m + 1, \bar{m}\}$ , obtain  $H_{\nu+1}$  by (2.3), set  $\nu = \nu + 1$ , and go to (1).

### 3. SOLVING THE TRUST-REGION SUBPROBLEM

Care must be taken in the execution of Step 3 of this algorithm. Our implementation adapts the Newton iteration described in Moré and Sorensen [9] to the limited memory setting. A straightforward implementation of this procedure may require solving equation (2.5) for many values of  $\mu$ . Such an implementation performs  $2mn$  multiplications each time the solution  $s_\mu$  to (2.5) is formed. This is very costly. Fortunately, as we now show, much of this cost can be avoided since it is possible to organize the computations so that the value of  $\mu$  corresponding to  $\bar{s}$  can be found by computations performed entirely in the small dimension  $2m$ .

In Step (1), we compute the unconstrained minimizer of the quadratic model  $g^T s + \frac{1}{2} s^T H s$ . If this step is rejected, then the solution to the trust–region subproblem in Step (3) necessarily lies on the boundary of the trust–region. That is, the solution  $\bar{s}$  to  $\mathcal{P}(g, H, \Delta)$  satisfies  $\|\bar{s}\| = \Delta$  with  $\mu > 0$  (in particular, the so-called *hard case* never occurs). Moré–Sorensen [9] locate the solution  $\bar{s}$  by using Newton’s method to solve the equation  $\phi(\mu) = 0$  where

$$\phi(\mu) = \frac{1}{\Delta} - \frac{1}{\|s(\mu)\|} \quad \text{and} \quad s(\mu) = -(\mu I + H)^{-1} g .$$

The choice of the function  $\phi$  was proposed by Reinsch [14] in the context of smoothing by spline functions where similar kinds of equations arise again within a Lagrangian framework. Reinsch shows that  $\phi$  both convex and nearly linear making it particularly amenable to Newton’s method. The Newton iteration takes the form

$$(3.1) \quad \mu_+ = \mu - \frac{\phi(\mu)}{\phi'(\mu)}, \quad \text{where} \quad \phi'(\mu) = -\frac{g^T(\mu I + H)^{-3} g}{\|s(\mu)\|^3} .$$

To compute these iterates we make use of the formulas

$$(\mu I + H)^{-1} = \frac{1}{\tau} [I + \Psi(\tau \Gamma - \Psi^T \Psi)^{-1} \Psi^T]$$

and

$$(\mu I + H)^{-2} = \frac{1}{\tau^2} [I + \Psi(\tau\Gamma - \Psi^T\Psi)^{-1}\Psi^T + \tau\Psi(\tau\Gamma - \Psi^T\Psi)^{-1}\Gamma(\tau\Gamma - \Psi^T\Psi)^{-1}\Psi^T],$$

where  $\tau = \mu + \lambda$  (see [2] for a general expression for  $(\mu I + H)^{-k}$ ). Setting

$$v_0 = -\Psi^T g, \quad v_1 = (\tau\Gamma - \Psi^T\Psi)^{-1}v_0, \quad \text{and} \quad v_2 = (\tau\Gamma - \Psi^T\Psi)^{-1}\Gamma v_1,$$

iteration (3.1) can be written as

$$\mu_+ = \mu + \frac{\sigma}{\delta} \left[ \frac{\sqrt{\sigma}}{\Delta} - \tau \right],$$

where

$$\sigma = \tau^2 \|s(\mu)\|^2 = v_1^T \Psi^T \Psi v_1 + 2v_0^T v_1 + \|g\|^2 \quad \text{and} \quad \delta = \tau^3 g^T (\mu I + H)^{-3} g = \sigma + \tau [v_1^T \Psi^T \Psi v_2 + v_0^T v_2].$$

These observations yield the following implementation of Newton's method applied to the function  $\phi$ .

**Algorithm 2: The Newton Iteration for the Trust-Region Subproblem**

Initialization Let  $H$  be as given in (2.3) and set  $\Phi = \Psi^T \Psi$ ,  $\gamma = \|g^\nu\|$ , and  $v_0 = \Psi^T g^\nu$ . Let  $\epsilon$  be the stopping tolerance.

Iteration

- (1) Set  $\tau = \mu + \lambda$ .
- (2) Set  $v_1 = (\tau\Gamma - \Phi)^{-1}v_0$ .
- (3) Set  $v_2 = (\tau\Gamma - \Phi)^{-1}\Gamma v_1$ .
- (4) Set  $\sigma = v_1^T \Phi v_1 + 2v_0^T v_1 + \gamma^2$ .
- (5) If  $|\sqrt{\sigma} - \tau\Delta| < \tau\epsilon$ , go to (9).
- (6) Set  $\delta = \sigma + \tau [v_1^T \Phi v_2 + v_0^T v_2]$ .
- (7) Set  $\mu_+ = \mu + \frac{\sigma}{\delta} [\frac{\sqrt{\sigma}}{\Delta} - \tau]$ .
- (8) If  $\mu_+ \geq 0$ , set  $\mu = \mu_+$ ; otherwise, set  $\mu = 0.2\mu$ . Go to (1).
- (9) Set  $\bar{s} = \frac{-1}{\tau} (g^\nu + \Psi v_1)$ .

We reiterate that once the matrix  $\Phi$  and the vector  $v_0$  are formed, then the only operation involving the large dimension  $n$  occurs when the final solution  $\bar{s}$  needs to be recovered from the lower dimensional variables in Step 9. The value  $\gamma$  is obtained as part of the formation of the matrix  $\Phi$ . Moreover, as  $\mu$  is varied, expression (2.7) shows that only operations in the small dimension are required to update the matrix  $(\tau\Gamma - \Phi)$ .

In our limited experience, we have found that the initial trial step taken as the solution to  $\hat{\mathcal{P}}(g^\nu, H, 0)$  is extremely good. Indeed, on the problems we have tested, this step is accepted more than 98% of the time with a value for the ratio  $r_\nu(\bar{s})$  in (2.2) that is nearly 1 or exceeds 1. This is quite surprising given the very limited amount of second-order information that the L-BFGS update possesses. As discussed above, it is the great practical success of the straightforward L-BFGS update that prompted us to use this update as the initial trial step in our trust-region implementations.

## 4. NUMERICAL EXPERIMENTS

We compare the performance of the trust-region updates against a FORTRAN limited memory BFGS (L-BFGS) implementation of the Liu and Nocedal algorithm [8] (with a line-search by Moré and Thuente), as it comes with the MINPACK-2 test problem collection [1] (by Averick, Carter and Moré). The MINPACK-2 implementation does not use the compact form of the BFGS update, but rather the original recursion formula by Nocedal [11]. We consider four unconstrained minimization problems from that collection on which L-BFGS is known to perform well (suggested by Moré), and two more unconstrained versions of constrained problems in that collection, on which L-BFGS performs well also. Our results on these selected problems using the trust region updates are comparable to the results obtained with L-BFGS.

The unconstrained versions of constrained problems are:

- Elastic-Plastic Torsion (EPT)
- Pressure in a Journal Bearing (PJB)

The unconstrained problems are:

- (Enneper's) Minimal Surface Area (MSA)
- Optimal Design with Composite materials (ODC)
- Steady State Combustion (SSC)
- homogeneous superconductors: 2-D Ginzburg-Landau (GL2).

In all cases we used the default parameter values and computed for 2,500, 10,000, 40,000 and 160,000 variables.

As stated in the introduction, our goal is to develop an algorithm that is efficient in its use of function and gradient evaluations at the expense of the more intensive linear algebra required to solve the trust-region subproblems. For this reason we first compare the performance of the two algorithms with respect to the sum of all function and gradient evaluations prior to termination. We do not give a higher weight to gradient evaluations. Relative performance is illustrated using the performance profile technique described in Dolan and Moré [4], i.e., for each algorithm, we plot the proportion  $P$  of the problems for which method is within factor  $\tau$  of the smallest sum of the number of function and gradient evaluations (on a log scale). Since the curve w.r.t trust-region algorithm is on the top of L-BFGS-linesearch, it implies that the MINPACK problems favor the trust-region algorithm over L-BFGS-linesearch with respect to the number of function and gradient evaluations. To see that the profile is not biased by the requested precision, we also include the graph for the performance profile with termination criteria

$$\frac{|f(x^k) - f_{best}|}{\max(|f_{best}|, 1)} \leq 10^{-3} \text{ and } nf \leq 1000.$$

Next we compare the performance of these two algorithms with respect to CPU time. Note that the solution time consists of two part:

1. The time is used to evaluate the functions and gradients.
2. The remaining time is dominated by the cost of the linear algebra.

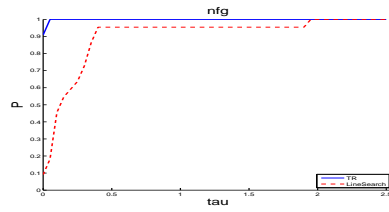


Figure 1a

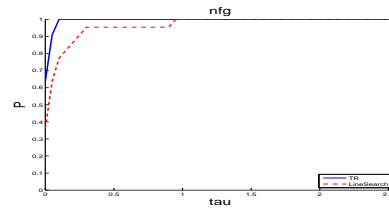


Figure 1b

Figure 1a. Performance profiles, sum of the function and gradient evaluations, relative accuracy  $10^{-5}$ . Figure 1b. Performance profiles, sum of the function and gradient evaluations, relative accuracy  $10^{-3}$ .

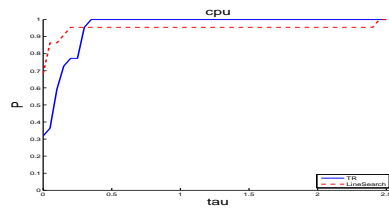


Figure 2a

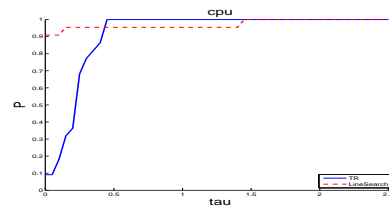


Figure 2b

Figure 2a. Performance profiles, cpu time, relative accuracy  $10^{-5}$ . Figure 2b. Performance profiles, cpu time, relative accuracy  $10^{-3}$ .

## 5. APPENDIX

Global convergence is established by adapting techniques of Powell [13]. For the remainder of the section  $\{x^k\}$  denotes a sequence generated by our explicit trust-region algorithm, write  $\delta_k$  to be the trust-region radius of the successful updating at  $x^k$ . Set  $\delta_k = \alpha_k \|B_k^{-1}g^k\|$ ,  $\alpha_k = \beta^{j_k}$ . Our goal is to show that either  $f(x^k) \rightarrow -\infty$  or  $g^k \rightarrow 0$ . For this we make the following blanket assumptions.

A1. There are constants  $m$  and  $M$ , such that  $m \leq \|B_k\| \leq M$  for all  $k$ .

A2.  $\nabla f$  is Lipschitz continuous with Lipschitz constant  $L$ .

**Lemma 5.1.** *Given a  $x$  and trust-region radius  $\delta$ , we have*

$$-q \geq \frac{1}{2} \alpha \frac{\|g\|^2}{M}$$

where  $g$  is the gradient at  $x$ ,  $q$  is the optimal value  $P(x, B, \alpha \|B^{-1}g\|)$ .

*Proof.* By [], we have

$$\begin{aligned}
-q &\geq \frac{1}{2} \|g\| \min\left(\frac{\|g\|^2}{M}, \delta\right) \\
&= \frac{1}{2} \|g\| \min\left(\frac{\|g\|}{M}, \alpha \|B^{-1}g\|\right) \\
&\geq \frac{1}{2} \|g\| \min\left(\frac{\|g\|}{M}, \alpha \frac{\|g\|}{M}\right) \\
&= \frac{1}{2} \alpha \frac{\|g\|^2}{M}.
\end{aligned}$$

□

**Lemma 5.2.** *For a given point  $x$ , where  $\nabla f(x) \neq 0$ , then there exist a  $\alpha > 0$ , such that if  $\bar{s}$ ,  $q$  are optimal solution  $P(x, B, \alpha \|B^{-1}g\|)$  respectively, then  $r = \frac{f(x+\bar{s})-f(x)}{-q} > \kappa$ .*

*Proof.* By Taylor's theorem,

$$f(x + \beta) = f(x) + g^T \bar{s} + \int_0^1 [\nabla f(x + t\bar{s}) - \nabla f(x)]^T \bar{s} dt,$$

so

$$\begin{aligned}
|f(x + \bar{s}) - f(x) - q| &= \left| -\frac{1}{2} \bar{s}^T B \bar{s} + \int_0^1 [\nabla f(x + t\bar{s}) - \nabla f(x)]^T \bar{s} dt \right| \\
&\leq c_1 \|s\|^2.
\end{aligned}$$

where  $c_1 = \frac{1}{2}(M + L)$ . Therefore

$$\begin{aligned}
|r - 1| &\leq \frac{c_1 \|\bar{s}\|^2}{\frac{1}{2} \alpha \|g\|^2} \\
&= \frac{2c_1 \alpha^2 \|B^{-1}g\|^2}{\alpha \|g\|^2} \\
&= 2c_1 \alpha \frac{\|B^{-1}g\|^2}{\|g\|^2}.
\end{aligned}$$

Since  $g \neq 0$ , there is an  $\alpha$ , such that  $r > \kappa$ . □

**Lemma 5.3.** *For the sequence of  $\{x^k\}$  generated by our algorithm, there exists an  $\epsilon$ , such that  $\alpha_k \geq \epsilon$  for all  $k$ .*

*Proof.* Let  $q$  be the optimal value with respect to the trust-region subproblem whose trust-region radius is  $\frac{\alpha_k}{\beta} \|B^{-1}g\|$ , set  $r = \frac{f(x+\bar{s})-f(x)}{-q}$ . Then by Lemma 5.2, we have

$$\begin{aligned}
1 - \kappa &\leq 1 - r \\
&\leq 2c_1 \frac{c_1}{\beta} \frac{\|B_k^{-1}g^k\|^2}{\|g^k\|^2}
\end{aligned}$$

Hence

$$\begin{aligned}\alpha_k &\geq \frac{(1-\kappa)\beta}{2c_1} \frac{\|g^k\|^2}{\|B_k^{-1}g^k\|^2} \\ &\geq \frac{(1-\kappa)\beta m^2}{2c_1} \text{ by } \|B_k^{-1}g^k\| \leq \|g^k\|/m\end{aligned}$$

Set  $\epsilon = \frac{(1-\kappa)\beta m^2}{2c_1}$ , then we have  $\alpha_k \geq \epsilon$  for all  $k$ . □

**Theorem 5.4.** *If  $\{f(x^k)\}$  is bounded below, then  $g^k \rightarrow 0$ .*

*Proof.* By our notation and Lemma 5.1, we have

$$f(x^k) - f(x^{k+1}) \geq \beta(-q_k) \geq \frac{1}{2}\alpha_k \frac{\|g^k\|^2}{M} \geq \frac{1}{2}\epsilon \frac{\|g^k\|^2}{M}.$$

Since  $\{f(x^k)\}$  is non-increasing and bounded below,  $f(x^k)$  is convergent. Thus  $\sum \|g^k\|^2$  converges. Hence  $g^k \rightarrow 0$ . □

## REFERENCES

- [1] B.M. Averick, R.G. Carter, and J.J. Moré. The MINPACK-2 Test Problem Collection (preliminary version). Technical Report TM-150, Mathematics and Computer Science Division, Argonne National Laboratory, 1991.
- [2] J.V. Burke. Sherman–morrison–woodbury formula for powers of the inverse. Preprint, 1996.
- [3] R.H. Byrd, J. Nocedal, and R.B. Schnabel. Representations of quasi–Newton matrices and their use in limited memory methods. *Math. Prog.*, 63:129–156, 1994.
- [4] E. D. Dolen and J.J. Moré. Benchmarking optimization software with performance profiles. Technical report, Mathematics and Computer Science, Argonne National Laboratory, Argonne, Illinois, USA, 2001.
- [5] R. Fletcher. On the Barzilai–Borwein method. Technical Report NA/207, Dundee Numerical Analysis Group, University of Dundee, 2001.
- [6] J.C. Gilbert and C. Lemarechal. Some numerical experiments with variable storage quasi–Newton algorithms. *Math. Prog.*, 45:407–435, 1989.
- [7] J.M. Borwein J. Barzilai. Two-point step-size gradient methods. *IMA J. Numer. Anal.*, 8:141–148, 1988.
- [8] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Prog.*, 45:503–528, 1989.
- [9] J.J. Moré and D.C. Sorensen. Computing a trust region step. *SIAM J. Sci. Stat. Comput.*, 4:553–572, 1983.
- [10] S.G. Nash and J. Nocedal. A numerical study of the limited memory BFGS method and the truncated Newton method for large scale optimization. *SIAM J. Optimization*, 1:358–372, 1991.
- [11] J. Nocedal. Updating quasi–Newton matrices with limited storage. *Math. Prog.*, 35:773–782, 1980.
- [12] S. Oren and E. Spedicato. Optimal conditioning of self scaling variable metric algorithms. *Math. Programming*, 10:70–90, 1976.
- [13] M.J.D. Powell. Convergence properties of a class of minimization algorithms. In O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 2*, pages 1–27. Academic Press, 1975.
- [14] C. H. Reinsch. Smoothing by spline functions. II. *Numer. Math.*, 16:451–454, 1971.
- [15] D.F. Shanno and K.H. Phua. Remark on algorithm 500: Minimization of unconstrained multivariate functions. *ACM Trans. Math. Software*, 6:618–622, 1980.
- [16] X. Zou, I.M. Navon, M. Berger, K.H. Phua, T. Schlick, , and F.X. Le Dimet. Numerical experience with limited–memory quasi–Newton and truncated newton methods. *SIAM J. Optimization*, 3:582–608, 1993.

(A.Wiegmann, J.V.Burke, Liang Xu) UNIVERSITY OF WASHINGTON, DEPT. OF MATHEMATICS, BOX 354350, SEATTLE, WA 98195