MATHEMATICS OF COMPUTATION
Volume 66, Number 220, October 1997, Pages 1509–1520
S 0025-5718(97)00866-1

A SUBSPACE LIMITED MEMORY QUASI-NEWTON ALGORITHM FOR LARGE-SCALE NONLINEAR BOUND CONSTRAINED OPTIMIZATION

Q. NI AND Y. YUAN

ABSTRACT. In this paper we propose a subspace limited memory quasi-Newton method for solving large-scale optimization with simple bounds on the variables. The limited memory quasi-Newton method is used to update the variables with indices outside of the active set, while the projected gradient method is used to update the active variables. The search direction consists of three parts: a subspace quasi-Newton direction, and two subspace gradient and modified gradient directions. Our algorithm can be applied to large-scale problems as there is no need to solve any subproblems. The global convergence of the method is proved and some numerical results are also given.

1. INTRODUCTION

The nonlinear programming problem with simple bounds on variables to be considered is

$$\text{minimize} \quad f(x) \tag{1.1}$$

$$\text{subject to} \quad l \leq x \leq u, \tag{1.2}$$

where $x \in \Re^n$. The objective function $f(x)$ is assumed to be twice continuously differentiable, l and u are given bound vectors in $\Re^n$, and n is the number of variables, which is assumed to be large.

Many algorithms have been proposed for solving small to medium-sized problems of the form (1.1)-(1.2) (for example see [4] and [5]). There are also some algorithms which are available for large-scale problems, such as the Lancelot algorithm of Conn, Gould and Toint [6]. Recently, a truncated bound sequential quadratic programming with limited memory [13] and another limited memory algorithm [10] were proposed for solving large-scale problems. An advantage of using limited memory update techniques is that the storage and the computational costs can be reduced. However, these algorithms still need to solve subproblems at every iteration.

In this paper, we propose an algorithm for solving (1.1)-(1.2) that does not need to solve subproblems. The search direction consists of three parts. The first one is

Received by the editor December 2, 1994 and, in revised form, May 8, 1995 and January 26, 1996.

1991 *Mathematics Subject Classification.* Primary 65K05, 90C06, 90C30.

Key words and phrases. Subspace quasi-Newton method, limited memory, projected search, large-scale problem, bound constrained optimization.

The authors were partially supported by the Stat key project "Scientific and Engineering Computing" and Chinese NNSF grant 19525101.

a quasi-Newton direction in the subspace spanned by inactive variables. The other two are subspace gradient and subspace modified gradient directions in the space spanned by active variables. The projected search, the limited memory techniques and the absence of costly subproblems, make the algorithm suitable for large-scale problems.

This paper is organized as follows. In Section 2 we discuss the construction of the algorithm. The global convergence of the algorithm is proved in Section 3 and numerical tests are given in Section 4.

2. Algorithm

We first discuss the determination of search directions.

2.1. **Determination of search directions.** In order to make our algorithm suitable for large-scale bound constrained problems, we do not solve subproblems to obtain line search directions. The algorithm uses limited memory quasi-Newton methods to update the inactive variables, and a projected gradient method to update the active variables. The inactive and active variables can be defined in terms of the active set; the active set $A(x)$ and its complementary set $B(x)$ are defined by

$$\begin{aligned} A(x) &= \{i: \ l_i \leq x_i \leq l_i+\epsilon_b \quad \text{or } u_i-\epsilon_b \leq x_i \leq u_i\}, \\ B(x) &= \{1,\dots,m\}/A(x) = \{i: \ l_i+\epsilon_b < x_i < u_i-\epsilon_b\}. \end{aligned} \tag{2.1}$$

The variables with indices in $A(x)$ are called active variables, while the variables with indices in $B(x)$ are called inactive variables.

The tolerance ϵ_b should be sufficiently small so that

$$0 < \epsilon_b < \min_i \frac{1}{3}(u_i - l_i). \tag{2.2}$$

It follows that ϵ_b satisfies

$$l_i + \epsilon_b < u_i - \epsilon_b \quad \text{for } \ i = 1, \dots, n, \tag{2.3}$$

and $B(x)$ is well-defined.

A subspace quasi-Newton direction is chosen as the search direction for the inactive variables. Let $P_0^{(k)}$ be the matrix whose columns are $\{e_i| \quad i \in B(x_k)\}$, where e_i is the $i-$th column of the identity matrix in $\Re^{n\times n}$. Let $H_k \in \Re^{m_k\times m_k}$ be an approximation of the reduced inverse Hessian matrix, m_k being the number of elements in $B(x_k)$. The search direction for the inactive variables is chosen as $-P_0^{(k)} H_k P_0^{(k)T} \nabla f(x_k)$.

In order to obtain the search direction for the active variables, we partition the active set $A(x)$ into three parts,

$$\begin{aligned} &A_1(x) = \{i: \ \ x_i = l_i \ \text{ or } \ x_i = u_i, \text{ and } (l_i+u_i-2x_i)g_i(x) \geq 0\}, \\ &A_2(x) = \{i: \ \ l_i \leq x_i \leq l_i+\epsilon_b \ \text{ or } \ u_i-\epsilon_b \leq x_i \leq u_i, \ (l_i+u_i-2x_i)g_i(x) < 0\}, \\ &A_3(x) = \{i: \ \ l_i < x_i \leq l_i+\epsilon_b \ \text{ or } \ u_i-\epsilon_b \leq x_i < u_i, \ (l_i+u_i-2x_i)g_i(x) \geq 0\}, \end{aligned} \tag{2.4}$$

where $(g_1(x), \dots, g_n(x))^T = g(x) = \nabla f(x)$. $A_1(x)$ is the index set of variables where the corresponding steepest descent directions head towards the outside of the feasible region. Therefore it is reasonable that we fix the variables with indices in $A_1(x_k)$ in the $k-$th iteration. $A_2(x)$ is the index set of the variables where the steepest descent directions move into the interior of the feasible region, and therefore we can use the steepest direction as a search direction in the corresponding subspace. $A_3(x)$ is the set of the active variables where the steepest directions move towards the boundary. Thus the steepest descent directions in this subspace should be truncated to ensure feasibility.

Define $P_j^{(k)}$ as the matrix whose columns are $\{e_i | i \in A_j(x_k)\}$, for $j = 1, 2, 3$, the search direction at the $k-$th iteration is defined by

$$d_k = -(P_0^{(k)} H_k P_0^{(k)T} + P_2^{(k)} P_2^{(k)T} + P_3^{(k)} P_3^{(k)T} \Lambda_k) g_k. \tag{2.5}$$

Here $g_k = g(x_k) = \nabla f(x_k)$, and $\Lambda_k = diag(\lambda_1^{(k)}, \dots, \lambda_n^{(k)})$ which is given by

$$\lambda_i^{(k)} = \begin{cases} 0, & \text{if } i \notin A_3(x_k), \\ (x_i^{(k)} - l_i)/g_i^{(k)}, & \text{if } \ l_i < x_i \le l_i + \epsilon_b \ \text{ and } \ x_i^{(k)} - g_i^{(k)} \le l_i, \\ (x_i^{(k)} - u_i)/g_i^{(k)}, & \text{if } \ u_i - \epsilon_b \le x_i < u_i, \ \text{ and } \ x_i^{(k)} - g_i^{(k)} \ge u_i, \\ 1, & \text{otherwise.} \end{cases} \tag{2.6}$$

The definition of search direction (2.5) and that of Λ_k in (2.6) ensures that

$$l_i \le x_i^{(k)} + d_i^{(k)} \le u_i, \tag{2.7}$$

holds for all $i \in A_3(x_k)$. d_k is a valid search direction because it is always a descent direction unless it is zero.

Lemma 2.1. *If H_k is positive definite, then d_k defined by* (2.5) *satisfies*

$$d_k^T g_k \le 0 \tag{2.8}$$

and the equality holds only if $d_k = 0$.

Proof. Define

$$\hat{H}_k = P_0^{(k)} H_k P_0^{(k)T} + P_2^{(k)} P_2^{(k)T} + P_3 P_3^{(k)T} \Lambda_k + P_1^{(k)} P_1^{(k)T}. \tag{2.9}$$

It is easy to see that $\hat{H}_k$ is positive definite. Because $P_1^{(k)T} d_k = 0$, (2.5) and (2.9) give

$$d_k^T g_k = -d_k^T \hat{H}_k^{-1} d_k \le 0, \tag{2.10}$$

The above relation and the positive definiteness of $\hat{H}_k$ indicate that (2.8) is true and that $d_k^T g_k = 0$ only if $d_k = 0$. □

We now describe the projected search.

2.2. **Projected search.** The projected search has been used by several authors for solving quadratic and nonlinear programming problems with simple bounds on the variables (see e.g. [9] and [10]). The projected search requires that a steplength, α_k, be chosen such that

$$\phi_k(\alpha) \le \phi_k(0) + \mu \phi_k'(0) \alpha \tag{2.11}$$

is satisfied for some constant $\mu \in (0, 1/2)$. Here ϕ_k is the piecewise twice continuously differentiable function

$$\phi_k(\alpha) = f(P_\Omega[x_k + \alpha d_k]), \tag{2.12}$$

where d_k is the search direction described in the previous section,

$$\Omega = \{x \in \Re^n : \ l \le x \le u\}, \tag{2.13}$$

and P_Ω is the projection into Ω defined by

$$(P_\Omega x)_i = \begin{cases} x_i & \text{if } l_i \le x_i \le u_i, \\ l_i & \text{if } x_i < l_i, \\ u_i & \text{if } x_i > u_i. \end{cases} \tag{2.14}$$

An initial trial value of $\alpha_{k,0}$ is chosen as 1. For $j = 1, 2, \dots$, let $\alpha_{k,j}$ be the maximum of $0.1\alpha_{k,j-1}$ and $\alpha^*_{k,j-1}$, where $\alpha^*_{k,j-1}$ is the minimizer of the quadratic function that interpolates $\phi_k(0), \phi'_k(0)$ and $\phi_k(\alpha_{k,j-1})$. Set $\alpha_k = \alpha_{k,j_k}$, where j_k is the first index j such that $\alpha_{k,j}$ satisfies (2.11).

Before the discussion on the termination of the projected search, we prove a lemma, which is similar to that in [10].

Lemma 2.2. *Let d_k be the search direction from* (2.5) *and assume that $d_k \neq 0$, then*

$$\min\{1, \|u - l\|_\infty / \|d_k\|_\infty\} \ge \beta_k \ge \min\{1, \epsilon_b / \|d_k\|_\infty\}, \tag{2.15}$$

where $\beta_k = \sup_{0 \le \gamma \le 1}\{\gamma : \ l \le x_k + \gamma d_k \le u\}$.

Proof. By the definition of β_k, x_k and $x_k + \beta_k d_k$ are feasible points of (1.1)-(1.2), which gives

$$\|\beta_k d_k\|_\infty \le \|u - l\|_\infty. \tag{2.16}$$

Thus the first part of (2.15) is true.

Now we show the second part of (2.15). It is sufficient to prove that

$$x_i^{(k)} + \bar\beta d_i^{(k)} \in [l_i, u_i] \tag{2.17}$$

for all $i = 1, \cdots, n$, where $\bar\beta = \min\{1, \epsilon_b / \|d_k\|_\infty\}$. If $i \in B(x_k)$, (2.17) follows from (2.1) and $|\bar\beta d_i^{(k)}| \le \epsilon_b$. If $i \in A_1(x_k)$, (2.17) is trivial as $d_i^{(k)} = 0$. If $i \in A_3(x_k)$, it follows from definition (2.6) that

$$x_i^{(k)} + d_i^{(k)} \in [l_i, u_i] \tag{2.18}$$

which implies (2.17). Finally we consider the case when $i \in A_2(x_k)$. We have $d_i^{(k)} = -g_i^{(k)} \neq 0$. If $d_i > 0$, then $x_i^{(k)} \in [l_i, l_i + \epsilon_b]$ which shows that

$$l_i \le x_i^{(k)} < x_i^{(k)} + \bar\beta d_i^{(k)} \le x_i^{(k)} + \epsilon_b \le l_i + 2\epsilon_b < u_i. \tag{2.19}$$

Similarly if $d_i < 0$, we have

$$l_i < u_i - 2\epsilon_b \le x_i^{(k)} - \epsilon_b \le x_i^{(k)} + \bar\beta d_i^{(k)} < x_i^{(k)} \le u_i. \tag{2.20}$$

Therefore we have shown that (2.17) holds for all $i = 1, \cdots, n$. □

It follows from Lemma 2.2 that

$$P_\Omega[x_k + \alpha d_k] = x_k + \alpha d_k, \text{ if } 0 \le \alpha \le \beta_k. \tag{2.21}$$

Thus

$$\phi_k(\alpha) = f(x_k + \alpha d_k) \quad \text{for} \quad 0 \le \alpha \le \beta_k \tag{2.22}$$

is a twice continuously differentiable function with respect to α. Hence the termination of the projected search in a finite number of steps is guaranteed if $\phi'_k(0) < 0$. Lemma 2.1 implies that $\phi'_k(0) = -d_k^T g_k < 0$ provided that $d_k \neq 0$.

2.3. **Subspace limited memory quasi-Newton algorithm.** Now, we give our algorithm for solving problem (1.1)-(1.2), which is called the subspace limited memory quasi-Newton (SLMQN) algorithm.

Algorithm 2.3. (SLMQN Algorithm)

Step 0 Choose a positive number $\mu \in (0, 1/2)$, $x_0 \in \Re^n$ and $H_0 = I$, where x_0 satisfies $l \leq x_0 \leq u$.
Compute $f(x_0), \nabla f(x_0)$ and set $k = 0$.

Step 1 Determine the search direction.
Determine $B(x_k)$, $A_1(x_k)$, $A_2(x_k)$ and $A_3(x_k)$ according to (2.1) and (2.4), and compute d_k from (2.5).

Step 2 Find a steplength α_k using the projected search described in Section 2.2.
Set

$$x_{k+1} = P_\Omega[x_k + \alpha_k d_k]. \tag{2.23}$$

If the termination condition is satisfied, then stop.

Step 3 Determine H_{k+1} by the limited memory BFGS inverse l-update [10]. In order to retain $s_k^T y_k > 0$, replace s_k with s_k' [12], defined by

$$\begin{aligned} s_k' &= \theta s_k + (1-\theta) H_k y_k, \\ \theta &= \begin{cases} 1 & \text{if } a \geq 0.2b, \\ 0.8b/(b-a) & \text{otherwise,} \end{cases} \end{aligned}$$

where $a = s_k^T y_k$, $b = y_k^T H_k y_k$, $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
$k = k + 1$, go to Step 1.

Remark. In Step 3, H_k is the reduced matrix

$$H_k = P_0^{(k)T} \bar{H}_k P_0^{(k)}, \tag{2.24}$$

where $\bar{H}_k$ is an approximation of the full space inverse Hessian matrix. The limited memory BFGS inverse m-update (see [11] and [10]) is as follows

$$\begin{aligned} \bar{H}_{k+1} = \; & Q_k^T \cdots Q_t^T \bar{H}_0 Q_t \cdots Q_k \\ & + Q_k^T \cdots Q_{t+1}^T \rho_t s_t s_t^T Q_{t+1} \cdots Q_k \\ & \cdots\cdots \\ & + Q_k^T \rho_{k-1} s_{k-1} s_{k-1}^T Q_k \\ & + \rho_k s_k s_k^T \end{aligned} \tag{2.25}$$

where $t = \max\{0, k - m + 1\}$, m is a given positive integer, $\bar{H}_0$ is a given positive definite matrix, and

$$\rho_i = \frac{1}{s_i^T y_i}, \quad Q_i = I - \rho_i y_i s_i^T. \tag{2.26}$$

Therefore, in Step 3 of SLMQN, we can update H_{k+1} by (2.24)-(2.26). Another way of updating H_k is to use only reduced gradient and projected steps. Namely we can let

$$\begin{aligned}\bar{H}_{k+1} &= Q_k^{(k+1)T}\cdots Q_t^{(k+1)T}P_0^{(k+1)T}\bar{H}_0P_0^{(k+1)}Q_t^{(k+1)}\cdots Q_k^{(k+1)}\\ &\quad + Q_k^{(k+1)T}\cdots Q_{t+1}^{(k+1)T}\rho_t s_t^{(k+1)}s_t^{(k+1)T}Q_{t+1}^{(k+1)}\cdots Q_k^{(k+1)}\\ &\quad \cdots\cdots\\ &\quad + Q_k^{(k+1)T}\rho_{k-1}s_{k-1}^{(k+1)}s_{k-1}^{(k+1)T}Q_k^{(k+1)}\\ &\quad + \rho_k s_k^{(k+1)}s_k^{(k+1)T}\ , \end{aligned} \tag{2.27}$$

where

$$s_i^{(k+1)} = P_0^{(k+1)}s_i, \quad y_i^{(k+1)} = P_0^{(k+1)}y_i, \tag{2.28}$$

$$Q_i^{(k+1)} = I - \rho_i y_i^{(k+1)}s_i^{(k+1)T}. \tag{2.29}$$

Updating formula (2.27) uses only vectors in $\Re^{m_{k+1}}$.

3. Convergence analysis

It is well known (for example, see [7]) that $x \in \Omega$ is a Kuhn-Tucker point of problem (1.1)-(1.2) if there exist $\lambda_i \geq 0,\ \mu_i \leq 0\ (i = 1, \cdots, n)$ such that

$$g(x) = \sum_{i=1}^{n}\lambda_i e_i + \sum_{i=1}^{n}\mu_i e_i,$$
$$\lambda[x_i - l_i] = 0,$$
$$\mu_i[u_i - x_i] = 0.$$

The above Kuhn-Tucker conditions are equivalent to

$$\begin{aligned} g_i(x)(l_i + u_i - 2x_i) \geq 0, &\quad \text{if}\ \ x_i = l_i\ \ \text{or}\ \ x_i = u_i,\\ g_i(x) = 0, &\quad \text{otherwise}. \end{aligned} \tag{3.1}$$

The following lemma shows that the search direction does not vanish if the iteration point is not a Kuhn-Tucker point.

Lemma 3.1. *Let x_k, d_k be given iterates of the SLMQN algorithm. Then x_k is a Kuhn-Tucker point of* (1.1)-(1.2) *if and only if $d_k = 0$.*

Proof. First we assume that x_k is a Kuhn-Tucker point of (1.1)-(1.2). From (2.4) and (3.1), it follows $A_2(x_k) = \emptyset$ and $g_i(x_k) = 0$ for $i \in B(x_k) \cup A_3(x_k)$. Hence, we obtain $d_k = 0$.

Now, suppose that $d_k = 0$. According to (2.5), we have

$$P_0^{(k)}H_kP_0^{(k)T}g_k = 0,\ P_2^{(k)}P_2^{(k)T}g_k = 0,\ \ P_3^{(k)}P_3^{(k)T}\Lambda_k g_k = 0. \tag{3.2}$$

Because H_k is positive definite and $\lambda_i^{(k)} \neq 0$ for $i \in A_3(x_k)$, it follows that

$$P_j^{(k)}g_k = 0, \quad j = 0, 2, 3.$$

Therefore $g_i^{(k)} = 0$ if $i \neq A_1(x_k)$, which implies that (3.1) holds for $x = x_k$. □

It follows from Lemmas 2.1 and 3.1 that d_k is a descent direction if x_k is not a Kuhn-Tucker point. Now we prove the global convergence theorem for the SLMQN algorithm.

Theorem 3.2. *Let x_k, d_k and H_k be computed by the SLMQN algorithm for solving the problem* (1.1)-(1.2) *and assume that*

(i) *$f(x)$ is twice continuously differentiable in Ω;*
(ii) *there are two positive constants $\gamma_1,\ \gamma_2$ such that*

$$\gamma_1\|P_0^{(k)T}g_k\|^2 \leq g_k^T P_0^{(k)} H_k P_0^{(k)T} g_k \tag{3.3}$$

$$\|P_0^{(k)T} H_k P_0^{(k)T}\| \leq \gamma_2 \tag{3.4}$$

for all k.

Then every accumulation point of $\{x_k\}$ is a Kuhn-Tucker point of the problem (1.1)-(1.2).

Proof. First we establish an upper bound for $d_k^T g_k$:

$$\begin{aligned} d_k^T g_k &= -g_k^T P_0^{(k)} H_k P_0^{(k)T} g_k - \|P_2^{(k)T} g_k\|_2^2 - \|P_3^{(k)T}\Lambda_k^{1/2} g_k\|_2^2 \\ &\leq -(\gamma_1\|P_0^{(k)T}g_k\|_2^2 + \|P_2^{(k)T}g_k\|_2^2 + \sum_{i\in A_3(x_k)} \tau_i^{(k)}|g_i^{(k)}|) \end{aligned} \tag{3.5}$$

where $\tau_i^{(k)} = \min\{|g_i^{(k)}|,\ |x_i^{(k)} - l_i|,\ |u_i - x_i^{(k)}|\}$. We also have

$$\|d_k\|_2^2 = \|P_0^{(k)} H_k P_0^{(k)T} g_k\|_2^2 + \|P_2^{(k)T} g_k\|_2^2 + \|P_3^{(k)T}\Lambda_k g_k\|_2^2. \tag{3.6}$$

Because $\lambda_i^{(k)} \in [0,1]$, and because H_k satisfies (3.4), it follows from (3.5) and (3.6) that

$$\|d_k\|_2^2 \leq -\max\{1,\gamma_2\} g_k^T d_k. \tag{3.7}$$

Further, from (3.6) and (3.4) yield

$$\|d_k\|_2^2 \leq \gamma_2^2\|g_k^T\|_2^2 + \|g_k\|_2^2 \leq (\gamma_2^2+1)\eta_1 \tag{3.8}$$

where $\eta_1 = \max_{x\in\Omega}\|g(x)\|_2^2$. Thus, from (2.15) and (3.8), there exists a constant $\tilde\beta \in (0,1)$ such that

$$\beta_k \geq \tilde\beta \quad \text{for all} \quad k. \tag{3.9}$$

If $\alpha_k < 0.1\tilde\beta$, by the definition of α_k there exists $j \geq 0$ such that $\alpha_{k,j} \leq 10\alpha_k$ and $\alpha_{k,j}$ is an unacceptable steplength, which implies that

$$\begin{aligned} f(x_k) + \mu\alpha_{k,j} g_k^T d_k &\leq f(x_k + \alpha_k d_k) \\ &\leq f(x_k) + \alpha_{k,j} g_k^T d_k + \frac{1}{2}\eta_2\alpha_{k,j}^2\|d_k\|^2, \end{aligned} \tag{3.10}$$

where $\eta_2 = \max_{x\in\Omega}\|\nabla^2 f(x)\|_2$. The above inequality and (3.7) imply that

$$\alpha_{k,j} \geq \frac{-2(1-\mu)g_k^T d_k}{\eta_2\|d_k\|_2^2} \geq \frac{2(1-\mu)}{\eta_2\max\{1,\gamma_2\}}. \tag{3.11}$$

Hence the above inequality and $\alpha_k \geq 0.1\alpha_{k,j}$ yield

$$\alpha_k \geq \min[\frac{-(1-\mu)}{5\eta_2\max\{1,\gamma_2\}},\ 0.1\tilde\beta] > 0 \tag{3.12}$$

for all k. Because Ω is a bounded set,

$$\begin{aligned} \infty &> \sum_{k=1}^{\infty}(f(x_k) - f(x_{k+1})) \\ &\geq \sum_{k=1}^{\infty} -\mu\alpha_k g_k^T d_k. \end{aligned} \tag{3.13}$$

(3.12) and (3.13) show that

$$\sum_{k=1}^{\infty} -g_k^T d_k < \infty \tag{3.14}$$

which implies

$$\lim_{k\to\infty} g_k^T d_k = 0. \tag{3.15}$$

It follows from (3.15) and (3.5) that

$$\lim_{k\to\infty} \|P_0^{(k)} g_k\| = 0, \tag{3.16}$$

$$\lim_{k\to\infty} \|P_2^{(k)} g_k\| = 0, \tag{3.17}$$

$$\lim_{k\to\infty} \sum_{i\in A_3(x_k)} \tau_i^{(k)} |g_i^{(k)}| = 0. \tag{3.18}$$

Let x^* be any accumulation point of $\{x_i\}$, there exists a subsequence $\{x_{k_i}\}$ $(i = 1, 2, \cdots)$ such that

$$\lim_{i\to\infty} x_{k_i} = x^*. \tag{3.19}$$

Define $A^* = \{i : \; x_i^* = l_i \text{ or } \; x_i^* = u_i\}$. If x^* is not a Kuhn-Tucker point, there exists $j \in A^*$ such that

$$g_j(x^*)(l_j + u_j - 2x_j^*) < 0 \tag{3.20}$$

or there exists $j \notin A^*$ such that

$$g_j(x^*) \neq 0. \tag{3.21}$$

If (3.20) holds for some $j \in A^*$, then

$$j \in A_2(x_{k_i}) \tag{3.22}$$

for all sufficiently large i. (3.22) and (3.17) show that

$$g_j(x^*) = 0, \tag{3.23}$$

which contradicts (3.20). If (3.21) holds for some $j \notin A^*$, we have

$$g_j(x^*)(l_j + u_j - 2x_j^*) \neq 0. \tag{3.24}$$

(3.24) and (3.16)-(3.18) imply that for all sufficiently large i,

$$j \notin B(x_{k_i}) \cup A_2(x_{k_i}) \cup A_3(x_{k_i}). \tag{3.25}$$

Therefore $j \in A_1(x_{k_i})$ for all large i, which would imply $x_j^{(k_i)} = l_j$ or $x_j^{(k_i)} = u_j$ for all sufficiently large k_0. This contradicts $x_{k_i} \to x^*$ and $j \notin A^*$. $\square$

Conditions (3.3) and (3.4) are satisfied if the matrix H_k is adjusted by limited memory BFGS inverse m-update (2.24)-(2.26) or (2.27)-(2.29).

4. Numerical tests

In this section some numerical results are reported. We have chosen 14 sets of test problems from [8] to compare our algorithm with the well-known L-BFGS-B algorithm in [13]. The termination condition is the projected gradient of the objective function below 10^{-5}, namely

$$\|P_\Omega(x_k - \nabla f(x_k)) - x_k\|_\infty \leq 10^{-5}, \tag{4.1}$$

where P_Ω is defined by (2.14). Computations are carried out on an SGI Indigo R4000 XS workstation. All codes are written in FORTRAN with double precision.

Numerical results are listed on Tables 1–4. In the tables, "Primal", "Dual" and "CG" stands for the L-BFGS-B Method, using primal, dual and CG methods for subspace minimization, respectively. The number of iterations (IT), the number of function evaluations (NF) and the CPU time in seconds (TIME) are given in the tables. The number of gradient evaluations is the same as the number of iterations for the SLMQN method and it equals the number of function evaluations for the L-BFGS-B method. N_a is the number of active variables at the solution. In all runs, we choose $\mu = 0.1$ and $\epsilon_b = 10^{-8}$.

The test results on EDENSCH and PENALTY1 are shown in Tables 1 and 2. The number of updates in the limited memory matrix, m, is chosen as 2 for all runs. The difference between SLMQN and L-BFGS-B is not great. CG takes a few more CPU seconds than other methods.

The test results on TORSION and JOURNAL are shown in Table 3, where m is chosen as 2. SLMQN is a little better than CG and slightly worse than Primal and Dual.

RAYBENDL problem is difficult. If m is chosen below 4, all methods terminate while the gradient stopping test is not met. Table 4 shows the results of all methods with $m = 4$. SLMQN takes a little more iterations than Primal and Dual, but less CPU seconds than them and CG.

Table 1. Test results on EDENSCH

	N_a	SLMQN	Primal	Dual	CG
1	0	21/28/1.20	22/32/1.41	22/32/1.05	24/34/2.92
2	0	15/19/0.84	14/18/0.88	14/18/0.67	14/18/1.07
3	667	14/21/0.71	12/16/0.63	12/16/0.65	12/16/0.67
4	999	13/20/0.66	11/14/0.75	11/14/0.87	10/14/0.50
5	1000	10/15/0.47	8/12/0.39	8/12/0.45	10/51/0.74

additional bounds: IT/NF/CPU sec.

1 $[-10^{20}, 10^{20}]\ \forall\ i$
2 $[0, 1.5]\ \forall$ odd i
3 $[-1, 0.5]\ \forall\ i = 3k+1$ number of variables = 2000
4 $[0, 0.99]\ \forall$ odd i
5 $[0, 0.5]\ \forall$ odd i

TABLE 2. Test results on PENALTY1

	N_a	SLMQN	Primal	Dual	CG
1	0	29/64/0.82	94/139/3.14	90/134/2.29	90/138/3.46
2	0	83/143/2.37	76/109/2.49	76/109/2.01	76/109/2.94
3	334	10/30//0.32	29/44/0.85	29/44/0.82	29/44/0.80
4	500	20/52/0.58	27/42/0.78	27/42/0.79	27/42/0.71

additional bounds: IT/NF/CPU sec.

1 $[-10^{20}, 10^{20}]\ \forall\, i$

2 $[0, 1]\ \forall$ odd i

3 $[0.1, 1]\ \forall\, i = 3k + 1$ number of variables = 1000

4 $[0.1, 1]\ \forall$ odd i

TABLE 3. Test results on TORSION and JOURNAL

	N_a	SLMQN	Primal	Dual	CG
TORSION	320	77/82/2.97	64/70/2.26	64/70/2.28	145/150/5.52
JOURNAL	330	154/185/6.28	148/155/6.06	145/150/5.44	165/176/7.22

additional bounds: IT/NF/CPU sec.

$[-10^{20}, 10^{20}]\ \forall\, i$ number of variables = 1024

TABLE 4. Test results on RAYBENDL

	N_a	SLMQN	Primal	Dual	CG
1	4	1144/1214/2.85	1058/1110/4.01	1103/1184/2.72	1138/1194/10.62
2	6	1202/1295/3.12	1098/1151/4.09	1115/1153/3.58	1279/1342/11.29

additional bounds: IT/NF/CPU sec.

1 $[-10^{20}, 10^{20}]\ \forall\, i$ number of variables = 44

2 $[2, 95]\ \forall\, i$ 4 variables are fixed (i.e. $u_i = l_i$)

In order to investigate the behavior of the SLMQN algorithm for very large problems, we choose 10 test problems from [5], where the number of variables is enlarged to $n = 10000$. The termination condition is that the infinity norm of the projected gradient is reduced below 10^{-4}, and m is chosen as 2. Numerical results are shown in Table 5. For TP6, TP7, TP10, TP11, TP20 and TP21, CG is better than the other three methods. There is little difference among SLMQN, Primal and Dual.

Other values of m $(2 < m < 10)$ have also been tried. But, they did not significantly alter the numerical results, but the CPU increased with m. The numerical results indicate that SLMQN is a promising algorithm and that SLMQN is not worse than L-BFGS-B. We have also observed that the sets $A_2(x_k)$ and $A_3(x_k)$ (see (2.4)) are empty for most of the iterations. Hence the search direction is often a subspace quasi-Newton step. Therefore the slow convergence of the projected gradient may not be a serious problem, though theoretically the use of the projected

TABLE 5. Test results on 10 problems ($N = 10000$)

	N_a	SLMQN	Primal	Dual	CG
TP1	4998	43/67/12.20	35/94/9.01	32/73/10.51	67/227/25.03
TP4	2500	30/43/8.29	27/38/7.14	27/38/7.13	25/32/8.00
TP5	5000	29/60/8.55	41/51/10.76	41/51/12.47	40/51/13.03
TP6	5823	83/123/24.22	81/183/22.98	79/123/25.82	65/85/21.34
TP7	5000	23/34/6.23	13/90/4.35	11/50/3.84	11/13/2.91
TP10	5000	17/27/9.34	16/20/8.97	16/20/9.45	15/19/8.89
TP11	5000	12/21/3.58	13/30/4.56	12/23/4.61	9/13/2.95
TP17	5000	71/91/21.98	41/49/13.93	40/48/11.71	43/57/19.73
TP20	5000	12/68/4.40	8/12/2.01	7/11/2.08	7/11/1.66
TP21	5000	6/7/3.66	5/7/3.48	3/5/2.35	3/5/2.26

gradient step cannot ensure superlinear convergence. There is a possibility of improving the SLMQN both theoretically and practically if we can find techniques to avoid slow convergence of the projected gradient steps. We could also consider the use of different computation formulas for the limited memory matrix (see [2]) used in SLMQN.

Acknowledgment

The authors would like to thank Professor J. Nocedal for providing us the L-BFGS-B programs and anonymous referees for their helpful comments on a previous version of this paper.

References

[1] R.H. Byrd, P. Lu, J. Nocedal and C. Zhu, *A limited memory algorithm for bound constrained optimization,* Report NAM-8, EECS Department, Northwestern University, 1994.

[2] R.H. Byrd, J. Nocedal and B. Schnabel, *Representation of quasi-Newton matrices and their use in limited memory methods*, Math. Prog., Vol. 63, (1994), 129-156. MR **95a:**90116

[3] I. Bongartz, A.R. Conn, N. Gould, and Ph.L. Toint, *CUTE: constrained and unconstrained testing environment,* Research Report, IBM T.J. Watson Research Center, Yorktown, USA.

[4] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *Global convergence of a class of trust region algorithm for optimization with simple bounds,* SIAM J. Numer. Anal. **25** (1988), 433-460. MR **89h:**90192

[5] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *Testing a class of methods for solving minimization problems with simple bounds on the variables,* Math. Comp. **50** (1988), 399-430. MR **89e:**65061

[6] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A),* Number 17 in Springer Series in Computational Mathematics, Springer Verlag, Heidelberg, New York, 1992. CMP 93:12

[7] R. Fletcher, *Practical Methods of Optimization,* John Wiley and Sons, Chichester, 1987. MR **89j:**65050

[8] P. Lu, *Bound constrained nonlinear optimization and limited memory methods,* Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, 1992.

[9] J.J. Moré and G. Toraldo, *On the solution of large quadratic programming problems with bound constraints,* SIAM J. Optimization **1** (1991), 93-113. MR **91k:**90137

[10] Q. Ni, *General large-scale nonlinear programming using sequential quadratic programming methods,* Bayreuther Mathematische Schriften, **45** (1993), 133-236. MR **94h:**90052

[11] J. Nocedal, *Updating quasi-Newton matrices with limited storage,* Math.Comp. **35** (1980), 773-782. MR **81g:**65077

[12] M.J.D. Powell, *A fast algorithm for nonlinearly constrained optimization calculations,* Lecture Notes in Mathematics 630. (1978), 144-157. MR **58:**3448

[13] C. Zhu, R.H. Byrd, P. Lu, and J. Nocedal, *L-BFGS-B Fortran subroutines for large-scale bound constrained optimization,* Report NAM-11, EECS Department, Northwestern University, 1994.

LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, P.O.Box 2719, Beijing 100080, China
E-mail address: niq@lsec.cc.ac.cn

LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, P.O.Box 2719, Beijing 100080, China
E-mail address: yyx@lsec.cc.ac.cn