Journal of Computational Mathematics, Vol.25, No.1, 2007, 89–96.

LIMITED MEMORY BFGS METHOD FOR NONLINEAR MONOTONE EQUATIONS *1)

Weijun Zhou
(*College of Mathematics, Changsha University of Science and Technology, Changsha 410077, China*
Email: weijunzhou@126.com)
Donghui Li
(*College of Mathematics and Econometrics, Hunan University, Changsha 410082, China*
Email: dhli@hnu.cn)

Abstract

In this paper, we propose an algorithm for solving nonlinear monotone equations by combining the limited memory BFGS method (L-BFGS) with a projection method. We show that the method is globally convergent if the equation involves a Lipschitz continuous monotone function. We also present some preliminary numerical results.

Mathematics subject classification: 90C30, 65K05.
Key words: Limited memory BFGS method, Monotone function, Projection method.

1. Introduction

In this paper, we consider the problem of finding a solution of the nonlinear equation

$$F(x) = 0, \tag{1.1}$$

where $F : R^n \to R^n$ is continuous and monotone. By monotonicity, we mean

$$\langle F(x) - F(y), x - y \rangle \geq 0, \quad \forall x, y \in R^n.$$

Nonlinear monotone equations have strong practical background, which include the subproblems in the generalized proximal algorithms with Bregman distances [10], the first order necessary condition of the unconstrained convex optimization problem and the KKT system of the convex equality constrained convex optimization problem. Some monotone variational inequality problems can also be converted into nonlinear monotone equations by means of fixed point maps or normal maps [22].

Among numerous algorithms for solving systems of smooth equations, the Newton method, quasi-Newton methods, Levenberg-Marquardt method and their variants are particularly useful because of their fast local convergence property [5, 6, 7, 19]. A general way to enlarge the convergence domain of the algorithm is to introduce some line search strategy such that the generated iterates exhibits descent property for some merit function [11]. Solodov and Svaiter [21] presented a Newton-type algorithm with a hybrid projection method for solving systems of monotone equations. The algorithm is globally convergent. The study on the globally convergent quasi-Newton method for solving nonlinear equations is relatively fewer. The major difficulty is the lack of practical line search strategy. Griewank [8] proposed a globally convergent Broyden's method. By using a nonmonotone line search process, Li and Fukushima [13, 14] proposed a Broyden's method for solving nonlinear equations and a Gauss-Newton-based BFGS method for solving symmetric nonlinear equations. Quite recently, Gu, Li, Qi and Zhou [9]

* Received July 13, 2005; final revised November 7, 2005, accepted March 24, 2006.
1) Support by NSF of China grant 10471036 and a 973 project.

introduced a norm descent line search technique and proposed a norm descent BFGS method for solving symmetric equations with global convergence.

A common drawback of the above mentioned quasi-Newton methods is that they need to compute and store an matrix at each iteration. This is computationally costly for large scale problems. To overcome this drawback, Nocedal [18] proposed a limited memory BFGS method (L-BFGS) for unconstrained optimization problems. Numerical results [4, 16] showed that the L-BFGS method is very competitive due to its low storage. This technique has received much attention in recent years, see, e.g., [1, 2, 3, 12, 17, 23] and references therein. However, as far as we know, there seems no related work for solving nonlinear equations. The purpose of this paper is to develop a L-BFGS method for solving nonlinear equations with monotone functions. The method can be regarded as a combination of the L-BFGS method [18] and the projection method [21]. Under some mild assumptions, we prove the global convergence of the method.

In Section 2, we state the steps of the method. In Section 3, we establish the global convergence of the method. In Section 4, we report some preliminary numerical results.

2. Algorithm

In this section, we describe the details of the method. First, we briefly review the L-BFGS method for solving the unconstrained optimization problem:

$$\min f(x), \quad x \in R^n,$$

where $f : R^n \to R$ is continuously differentiable. We denote by $\nabla f(x)$ the gradient of f at x. The steps of the L-BFGS method [16] for solving the unconstrained optimization problem are stated as follows.

Algorithm 1 (L-BFGS algorithm).

Step 1: Given initial point $x_0 \in R^n$, integer m and a symmetric positive definite matrix B_0. Let $k := 0$.

Step 2: Compute d_k by $B_k d_k = -\nabla f(x_k)$, $x_{k+1} = x_k + \alpha_k d_k$, where α_k satisfies some line search.

Step 3: Let $\tilde{m} = \min\{k+1, m\}$. Choose a symmetric and positive definite matrix $B_k^{(0)}$ and a set of increasing integers $L_k = \{j_0, \cdots, j_{\tilde{m}-1}\} \subseteq \{0, \cdots, k\}$. Update $B_k^{(0)}$ $\tilde{m}$ times using the pairs $\{y_{jl}, s_{jl}\}_{l=0}^{\tilde{m}-1}$, i.e., for $l = 0, \cdots, \tilde{m}-1$ compute

$$B_k^{(l+1)} = B_k^{(l)} - \frac{B_k^{(l)} s_{jl} s_{jl}^T B_k^{(l)}}{s_{jl}^T B_k^{(l)} s_{jl}} + \frac{y_{jl} y_{jl}^T}{y_{jl}^T s_{jl}},$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Set $B_{k+1} = B_k^{(\tilde{m})}$ and $k := k+1$. Go to Step 2.

There are many possible choice of $B_k^{(0)}$ in Step 3, for example, we can let $B_k^{(0)} = B_0$.

To describe our method, let us recall the projection method [21] for solving the nonlinear monotone equation (1.1). By the monotonicity of F, we have

$$\langle F(z_k), \bar{x} - z_k \rangle \le 0$$

for any $\bar{x}$ satisfying $F(\bar{x}) = 0$. Suppose we have obtained a direction d_k. By performing some kind of line search procedure along the direction d_k, a point $z_k = x_k + \alpha_k d_k$ can be computed such that

$$\langle F(z_k), x_k - z_k \rangle > 0.$$

Thus the hyperplane

$$\mathcal{H}_k = \{x \in R^n | \langle F(z_k), x - z_k \rangle = 0\}$$

strictly separates the current iterate x_k from zeros of the equation (1.1). Once the separating hyperplane is obtained, the next iterate x_{k+1} is computed by projecting x_k onto the hyperplane.

By means of the technique of L-BFGS method and the projection method, we state our algorithm as follows.

Algorithm 2.

Step 0: Given initial point $x_0 \in R^n$, integer $m > 0$ and constants $\beta \in (0,1)$, $\sigma \in (0,1)$ and $\varepsilon > 0$. Choose $B_0 = I$ (the identity matrix). Let $k := 0$.

Step 1: Compute d_k by

$$B_k d_k = -F(x_k). \tag{2.1}$$

Stop if $d_k = 0$.

Step 2: Determine steplength $\alpha_k = \beta^{m_k}$ such that m_k is the smallest nonnegative integer m satisfying

$$-\langle F(x_k + \beta^m d_k), d_k \rangle \geq \sigma \beta^m \|d_k\|^2. \tag{2.2}$$

Let $z_k = x_k + \alpha_k d_k$.

Step 3: Compute

$$x_{k+1} = x_k - \frac{\langle F(z_k), x_k - z_k \rangle}{\|F(z_k)\|^2} F(z_k). \tag{2.3}$$

Step 4: Compute B_{k+1} by the following modified L-BFGS update process. Let $\tilde{m} = \min\{k+1, m\}$. Choose $B_k^{(0)} = B_0 = I$. Choose a set of increasing integers $L_k = \{j_0, \cdots, j_{\tilde{m}-1}\} \subseteq \{0, \cdots, k\}$. Update $B_k^{(0)}$ $\tilde{m}$ times using the pairs $\{y_{jl}, s_{jl}\}_{l=0}^{\tilde{m}-1}$, i.e., for $l = 0, \cdots, \tilde{m}-1$, update

$$B_{k+1}^{(l+1)} = \begin{cases} B_k^{(l)} - \frac{B_k^{(l)} s_{jl} s_{jl}^T B_k^{(l)}}{s_{jl}^T B_k^{(l)} s_{jl}} + \frac{y_{jl} y_{jl}^T}{y_{jl}^T s_{jl}}, & \text{if } \dfrac{y_{jl}^T s_{jl}}{\|s_{jl}\|^2} \geq \varepsilon, \\ B_k^{(l)}, & \text{otherwise}, \end{cases} \tag{2.4}$$

where $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. Set $B_{k+1} = B_k^{(\tilde{m})}$, $k := k+1$. Go to Step 1.

Remarks.

(i) In updating $B_k^{(l)}$, we used the cautious update rule proposed by Li and Fukushima [15]. An advantage of this update is that the generated matrices B_k are symmetric and positive definite for all k.

(ii) Let $H_k^{(l+1)} = (B_k^{(l+1)})^{-1}$, $\theta_k^{(l)} = 1/y_{jl}^T s_{jl}$ and $V_k^{(l)} = I - \theta_k^{(l)} y_{jl} s_{jl}^T$. It is not difficult to derive the inverse update formula of (2.4):

$$H_k^{(l+1)} = \begin{cases} V_k^{(l)} H_k^{(l)} V_k^{(l)} + \theta_k^{(l)} s_{jl} s_{jl}^T, & \text{if } \dfrac{y_{jl}^T s_{jl}}{\|s_{jl}\|^2} \geq \varepsilon, \\ H_k^{(l)}, & \text{otherwise}. \end{cases}$$

If we assume that F is Lipschitz continuous, i.e., there exists a constant $L > 0$ such that

$$\|F(x) - F(y)\| \le L\|x - y\|, \quad \forall x, y \in R^n, \tag{2.5}$$

then it is not difficult to show that the sequences $\{\|B_k\|\}$ and $\{\|B_k^{-1}\|\}$ are bounded. That is, there exists a constant $\mu > 0$, such that

$$\|B_k\| \le \mu, \qquad \|B_k^{-1}\| \le \mu. \tag{2.6}$$

(iii) In Algorithm 2, we can get d_k by computing $H_k F(x_k)$ which can be obtained by a recursive formula described in [18]. So if $B_k^{(0)} = I$, we need not store B_k or compute H_k directly.

(iv) When $m = 1$, the L-BFGS method reduces to the memoryless BFGS method [20] with the cautious update rule.

(v) The line search (2.2) is a little different from that of [21]. It is not difficult to see from (ii) that it is well-defined.

3. Convergence Property

In order to obtain global convergence, we need the following lemma [21].

Lemma 3.1. *Let F be monotone and assume $x, y \in R^n$ satisfy $\langle F(y), x - y\rangle > 0$. Let*

$$x^+ = x - \frac{\langle F(y), x - y\rangle}{\|F(y)\|^2} F(y).$$

Then for any $\bar{x} \in R^n$ satisfying $F(\bar{x}) = 0$, it holds that

$$\|x^+ - \bar{x}\|^2 \le \|x - \bar{x}\|^2 - \|x^+ - x\|^2.$$

Now we establish the convergence theorem for Algorithm 2.

Theorem 3.2. *Suppose that F is monotone and Lipschitz continuous. Let $\{x_k\}$ be generated by Algorithm 2. Suppose further that the solution set of (1.1) is not empty. Then for any $\bar{x}$ satisfying $F(\bar{x}) = 0$, it holds that*

$$\|x_{k+1} - \bar{x}\|^2 \le \|x_k - \bar{x}\|^2 - \|x_{k+1} - x_k\|^2.$$

In particular, $\{x_k\}$ is bounded. Furthermore, it holds that either $\{x_k\}$ is finite and the last iterate is a solution, or the sequence is infinite and $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$. Moreover, $\{x_k\}$ converges to some $\bar{x}$ satisfying $F(\bar{x}) = 0$.

Proof. We first note that if the algorithm terminates at some iteration k, then $d_k = 0$. By the positive definiteness of B_k, we have $F(x_k) = 0$. This means that x_k is a solution of (1.1).

Suppose that $d_k \neq 0$ for all k. Then an infinite $\{x_k\}$ is generated. It follows from (2.2) that

$$\langle F(z_k), x_k - z_k\rangle = -\alpha_k \langle F(z_k), d_k\rangle \ge \sigma\alpha_k^2 \|d_k\|^2 > 0. \tag{3.1}$$

Let $\bar{x}$ be any point such that $F(\bar{x}) = 0$. By (2.3), (3.1) and Lemma 3.1, we obtain

$$\|x_{k+1} - \bar{x}\|^2 \le \|x_k - \bar{x}\|^2 - \|x_{k+1} - x_k\|^2. \tag{3.2}$$

Hence the sequence $\{\|x_k - \bar{x}\|\}$ is decreasing and convergent. In particular, the sequence $\{x_k\}$ is bounded, and

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0. \tag{3.3}$$

By (2.1) and (2.6), it holds that $\{d_k\}$ is bounded and so is $\{z_k\}$. Since F is continuous, there exists a constant $C > 0$ such that $\|F(z_k)\| \le C$.

We obtain from (2.3) and (3.1) that

$$\|x_{k+1} - x_k\| = \frac{\langle F(z_k), x_k - z_k \rangle}{\|F(z_k)\|} \ge \frac{\sigma}{C}\alpha_k^2 \|d_k\|^2.$$

From the last inequality and (3.3), we get

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{3.4}$$

If $\liminf_{k\to\infty} \|d_k\| = 0$, it follows from (2.1) and (2.6) that $\liminf_{k\to\infty} \|F(x_k)\| = 0$. By the continuity of F and the boundedness of $\{x_k\}$, it is clear that the sequence $\{x_k\}$ has some accumulation point $\hat{x}$ such that $F(\hat{x}) = 0$. We also have from (3.2) that the sequence $\{\|x_k - \hat{x}\|\}$ converges. Therefore, $\{x_k\}$ converges to $\hat{x}$.

If $\liminf_{k\to\infty} \|d_k\| > 0$, it follows from (2.1) and (2.6) that $\liminf_{k\to\infty} \|F(x_k)\| > 0$. By (3.4), it must hold that

$$\lim_{k\to\infty} \alpha_k = 0.$$

We have from (2.2)

$$-\langle F(x_k + \beta^{m_k - 1} d_k), d_k \rangle < \sigma \beta^{m_k - 1} \|d_k\|^2. \tag{3.5}$$

Since $\{x_k\}$ and $\{d_k\}$ are bounded, we can choose a subsequences $\{x_k\}_K$ and $\{d_k\}_K$ having limits $\hat{x}$ and $\hat{d}$. Taking limit in (3.5) as $k \to \infty$ with $k \in K$, we obtain

$$-\langle F(\hat{x}), \hat{d} \rangle \le 0.$$

However, it is not difficult to deduce from (2.1) and (2.6) (by further taking subsequence if necessary) that

$$-\langle F(\hat{x}), \hat{d} \rangle > 0.$$

This yields a contradiction. Consequently, $\liminf_{k\to\infty} \|F(x_k)\| > 0$ is not possible. The proof is complete.

4. Numerical Results

In this section, we report some numerical results with the proposed method. We test the performance of Algorithm 2 on the following three problems with various sizes.

Problem 1. The elements of function F are given by

$$F_i(x) = 2x_i - \sin|x_i|, \quad i = 1, 2, \cdots, n.$$

Problem 2. The elements of function F are given by

$$F_i(x) = 2x_i - \sin(x_i), \quad i = 1, 2, \cdots, n.$$

Problem 3. The elements of function F are given by $F_1(x) = 2x_1 + \sin(x_1) - 1$,

$$F_i(x) = -2x_{i-1} + 2x_i + \sin(x_i) - 1, \quad i = 2, \cdots, n-1,$$

and $F_n(x) = 2x_n + \sin(x_n) - 1$.

Problems 1 and 2 are similar. The difference is that Problem 1 is not differentiable at $x = 0$ while Problem 2 is smooth everywhere.

We first test the performance of Algorithm 2 on Problem 3 with different dimensions. The results are listed in Table 1 where the numbers stand for the total number of iterations. The parameters are same as that of Algorithm 2 on Problems 1 and 2 below. The results in the table show that Algorithm 2 terminates successfully for all initial points. Moreover, the initial points do not affect the number of iterations very much.

Table 1: Test results for Problem 3 using Algorithm 2

x_0^T	$(0.1,\cdots,0.1)$	$(1,\cdots,1)$	$(1,1/2,\cdots,1/n)$	$(0,\cdots,0)$	$(-0.1,-0.1,\cdots,-0.1)$	$(-1,-1,\cdots,-1)$
n=10	26	23	27	28	28	31
n=100	222	233	222	221	228	218
n=500	1077	1092	1084	1073	1074	1077
n=1000	2003	2017	2011	2000	2001	2007
n=2000	3180	3191	3186	3177	3177	3184
n=3000	3881	3889	3885	3877	3877	3884

Table 2: Test results for Problems 1 and 2 using Algorithm 2 and INM method

		Algorithm 2 for P1		INM for P1		Algorithm 2 for P2		INM for P2	
init	n	iter	time	iter	time	iter	time	iter	time
x_1	100	28	0.2	108	2.204	28	0.2	108	1.422
x_2	100	13	0.1	16	1.522	13	0.12	16	0.762
x_3	100	11	0.12	9	0.741	11	0.12	9	1.492
x_4	100	14	0.07	108	0.852	28	0.21	120	0.782
x_5	100	10	0.26	7	0.11	11	0.14	16	0.11
x_6	100	13	0.251	17	0.18	13	0.12	29	0.19
average	100	14.8333	0.16683	44.1667	0.93483	17.3333	0.15167	49.6667	0.793
x_1	500	30	0.721	232	22.483	30	0.721	232	22.011
x_2	500	14	0.34	30	3.976	14	0.29	30	3.415
x_3	500	11	0.24	8	1.543	11	0.23	8	1.863
x_4	500	15	0.44	232	21.201	30	0.671	245	23.013
x_5	500	11	0.44	9	0.952	11	0.21	19	1.633
x_6	500	13	0.47	30	2.633	14	0.42	43	4.016
average	500	15.6667	0.44183	90.1667	8.798	18.3333	0.42367	96.1667	9.3252
x_1	1000	30	1.602	325	104.931	30	1.642	325	105.371
x_2	1000	14	0.811	37	13.47	14	0.761	37	12.859
x_3	1000	11	0.611	8	4.176	11	0.501	8	3.374
x_4	1000	16	1.021	325	104.01	30	1.543	338	106.633
x_5	1000	11	0.761	11	3.675	12	0.651	20	6.399
x_6	1000	14	0.882	40	12.968	14	0.751	53	16.664
average	1000	16	0.948	124.3333	40.5383	18.5	0.97483	130.1667	41.8833
x_1	2000	31	6.119	456	573.104	31	5.929	456	576.649
x_2	2000	14	2.634	51	66.425	14	2.844	51	66.175
x_3	2000	11	2.174	7	11.416	11	2.224	7	11.977
x_4	2000	16	3.145	457	571.832	31	5.938	470	589.818
x_5	2000	12	2.383	11	14.131	12	2.443	22	27.8
x_6	2000	14	2.894	52	65.034	14	2.824	67	84.312
average	2000	16.3333	3.2248	172.3333	216.9903	18.8333	3.7003	178.8333	226.1218

We then compare performance of Algorithm 2 with the Inexact Newton Method (INM) in [21] on Problems 1 and 2 with different initial points. The results are listed in Table 2 where $x_1 = (10, 10, \cdots, 10)^T$, $x_2 = (1, 1, \cdots, 1)^T$, $x_3 = (1, 1/2, \cdots, 1/n)^T$, $x_4 = (-10, -10, \cdots, -10)^T$, $x_5 = (-0.1, -0.1, \cdots, -0.1)^T$, $x_6 = (-1, -1, \cdots, -1)^T$. The parameters in Algorithm 2 are specified as follows. We set $\beta = 0.6$, $\sigma = 0.1$, $\varepsilon = 0.1$, $m = 1$. For INM method in [21], we set $\mu_k = \|F(x_k)\|$, $\rho_k = 0$, $\beta = 0.6$, $\lambda = 0.01$. We use $\|F(x_k)\| < 10^{-4}$ as the stopping criterion. The algorithms were coded in MATLAB and run on Personal Computer with 400GHZ CPU processor. The meaning of the columns in Table 2 is stated as follows. "n" denotes the dimension of the problem, "init" stands for the initial point, "iter" stands for the total number of iterations, "time" stands for CPU time in seconds, "average" is the average iteration and CPU time respectively.

The results in Table 2 show that in most cases Algorithm 2 performs better than the INM method. In particular, for initial points x_1 and x_4, which are far away from the solution of Problems 1 and 2, the performance of Algorithm 2 is much better than that of the INM method.

5. Conclusions

We have proposed a limited memory BFGS method for solving nonlinear monotone equations and proved its global convergence. We have presented some preliminary numerical results to show its efficiency. The proposed method is globally convergent even if the Jacobian matrix of the equation is not symmetric. As demonstrated in Section 4, the method works well for Problems 1 and 2. On the other hand, we found that the performance of the method was not so satisfactory for Problem 3. We note that the Jacobian matrices in Problems 1 and 2 are symmetric and positive semi-definite, while in Problem 3, the Jacobian matrix is positive definite but not symmetric. This may show that the L-BFGS method is more suitable for solving symmetric equations. As pointed out by an anonymous referee, nonsymmtric quasi-Newton methods such as the Broyden's rank one method may work better when used for solving nonsymmetric equations. We refer to some recent papers [8, 13] for the study of Broyden's method.

Acknowledgment. The authors would like to thank the referees for their helpful comments on the paper.

References

[1] M. Al-Baali, Extra updates for the BFGS method, *Optim. Methods Softw.*, **13** (2000), 159-179.

[2] M. Al-Baali, Improved Hessian approximations for the limited memory BFGS methods, *Numer. Algorithms*, **22** (1999), 99-112.

[3] R.H. Byrd, J. Nocedal and R.B. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, *Math. Program.*, **63** (1994), 129-156.

[4] R.H. Byrd, J. Nocedal and C. Zhu, Towards a discrete Newton method with memory for large-scale optimization, Optimization Technology Center Report OTC-95-1, EEC Department, Northwestern University, 1995.

[5] J.E. Dennis and J.J. Moré, A characterization of superlinear convergence and its application to quasi-Newton methods, *Math. Comput.*, **28** (1974), 549-560.

[6] J.E. Dennis and J.J. Moré, Quasi-Newton method, motivation and theory, *SIAM Rev.*, **19** (1977), 46-89.

[7] J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, N.F., 1983.

[8] A. Griewank, The 'global' convergence of Broyden-like methods with a suitable line search, *J. Austral. Math. Soc.*, Ser. B, **28** (1986), 75-92.

[9] G.Z. Gu, D.H. Li, L. Qi and S.Z. Zhou, Descent directions of quasi-Newton methods for symmetric nonlinear equations, *SIAM J. Numer. Anal.*, **40** (2003), 1763-1774.

[10] A.N. Iusem and M.V. Solodov, Newton-type methods with generalized distances for constrained optimization, *Optimization*, **41** (1997), 257-278.

[11] H. Jiang and D. Ralph, Global and local superlinear convergence analysis of Newton-type methods for semismooth equations with smooth least squares. in Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods (Lausanne, 1997), pp. 181-209, Edited by M.Fukushima and L.Qi, Appl. Optim., 22, Kluwer Acad. Publ., Dordrecht, 1999.

[12] T.G. Kolda, D.P. O'Leary and L. Nazareth, BFGS with update skipping and varying memory, *SIAM J. Optim.*, **8** (1998), 1060-1083.

[13] D.H. Li and M. Fukushima, A derivative-free line search and global convergence of Broyden-like method for nonlinear equations, *Optim. Methods and Softw.*, **13** (2000), 181-201.

[14] D.H. Li and M. Fukushima, A globally and superlinear convergent Gauss-Newton-based BFGS method for symmetric nonlinear equations, *SIAM J. Numer. Anal.*, **37** (1999), 152-172.

[15] D.H. Li and M. Fukushima, On the global convergence of the BFGS method for nonconvex unconstrained optimization problems, *SIAM J. Optim.*, **11** (2001), 1054-1064.

[16] D.C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Program.*, **45** (1989), 503-528.

[17] 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. Optim.*, **1** (1991), 358-372.

[18] J. Nocedal, Updating quasi-Newton matrices with limited storage, *Math. Comput.*, **35** (1980), 773-782.

[19] J.M. Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, 1970.

[20] D.F. Shanno, Conjugate gradient methods with inexact searches, *Math. Oper. Res.*, **3** (1978), 244-256.

[21] M.V. Solodov and B.F. Svaiter, A globally convergent inexact Newton method for systems of monotone equations. in Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods (Lausanne, 1997), pp. 355-369, Edited by M.Fukushima and L.Qi, Appl. Optim., 22, Kluwer Acad. Publ., Dordrecht, 1999.

[22] Y.B. Zhao and D. Li, Monotonicity of fixed point and normal mapping associated with variational inequality and its application, *SIAM J. Optim.*, **11** (2001), 962-973.

[23] 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. Optim.*, **3** (1993), 582-608.