Malaysian Journal of Mathematical Sciences 7(2): 181-201 (2013)

MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES

Journal homepage: http://einspem.upm.edu.my/journal

Some Diagonal Preconditioners for Limited Memory Quasi-Newton Method for Large Scale Optimization

[1,2*]Hong Seng Sim, [1,3]Wah June Leong, [1,3]Malik Abu Hassan and [1,2]Fudziah Ismail

[1]Institute for Mathematical Research, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

[2] Faculty of Applied Sciences, UCSI University, 56000 Cheras, Kuala Lumpur, Malaysia

[3]Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

E-mail: hongseng0505@hotmail.com.my

*Corresponding author

ABSTRACT

One of the well-known methods in solving large scale unconstrained optimization is limited memory quasi-Newton (LMQN) method. This method is derived from a subproblem in low dimension so that the storage requirement as well as the computation cost can be reduced. In this paper, we propose a preconditioned LMQN method which is generally more effective than the LMQN method dueto the main defect of the LMQN method that it can be very slow on certain type of nonlinear problem such as ill-conditioned problems. In order to do this, we propose to use a diagonal updating matrix that has been derived based on the weak quasi-Newton relation to replace the identity matrix to approximate the initial inverse Hessian. The computational results show that the proposed preconditioned LMQN method performs better than LMQN method that without preconditioning.

Keywords: Preconditioned, limited memory quasi-Newton methods, large scale, unconstrained optimization.

1. INTRODUCTION

Limited memory quasi-Newton (LMQN) methods are used to solve the optimization problems especially large scale problems. These methods make simple approximations of the Hessian matrices and they provide a faster rate of linear convergence and only require minimal storage, hence it is more appropriate to use the LMQN methods instead of the quasi-Newton methods.

LMQN methods are the extensions of the conjugate gradient method that through additional storage is used to speed up the convergence. LMQN methods are suitable for most of the large scale unconstrained optimization due to the ability of the user can control the amount of storage that required by the algorithm. Furthermore, this method are actually the implementations of the quasi-Newton methods but with the storage is already restricted.

A general form of the LMQN methods is given by

$$H_{k+1} = \gamma_k P_k^T H_0 Q_k + \sum_{i=1}^{m_k} W_{ik} Z_{ik}^T, \tag{1}$$

where H_0 is a $n \times n$ symmetric positive definite matrix that remains constant for all k; γ_k is a nonzero scalar that iteratively rescales H_0; P_k is a $n \times n$ matrix that a product of projection matrices of the form

$$I - \frac{uv^T}{u^T v}, \tag{2}$$

by which $u \in$ span $\{y_0, \ldots, y_k\}$ and $v \in$ span $\{s_0, \ldots, s_{k+1}\}$; Q_k a $n \times n$ matrix, the product of the projection matrices of the same form where u is any $n-$ vector $v \in$ span $\{s_0, \ldots, s_k\}$; m_k is a nonnegative integer; $W_{ik}(i = 1,2,\ldots,m_k)$ is any $n-$ vector; $Z_{ik}(i = 1,2,\ldots,m_k)$ is any vector in span$\{s_0, \ldots, s_k\}$.

Equation (1) is a general result that characterizes perfect quasi-Newton methods that terminate in n iterations on an n-dimensional strictly convex quadratic. Some variant of these methods can be found in Farid *et al.* (2010), Farid *et al* (2011), Leong and Hassan (2009, 2011), Leong *et al.* (2010) and Waziri *et al.* (2010).

2. LIMITED MEMORY BFGS METHOD

One of the famous LMQN method is the limited memory BFGS method. The limited memory BFGS method (L-BFGS) is proposed by Nocedal (1980). The implementation of the L-BFGS method is almost identical to the BFGS method but with the difference in matrix update, whereby the BFGS corrections are stored separately, and when the available storage is used up, the oldest correction is deleted to make space for the new one. Thus, all subsequent iterations will insert a new correction whereas an old correction will be deleted. Besides that, the user actually can specify the number m of BFGS corrections that are to be kept, and provides a sparse symmetric and positive definite matrix H_0, which approximates the inverse Hessian of f. This method is identical to the BFGS method during the first m iterations. For $k > m$, H_k is obtained by applying m BFGS updates to H_0 using the information from the m previous iterations. (Liu and Nocedal (1989)).

Some of the notations are introduced to give a description of the L-BFGS method. The iterates will be denoted by x_k, and $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$ are defined. According to Dennis and Schnabel (1983), the method will use the inverse BFGS formula in the form as follow

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \tag{3}$$

where

$$\rho_k = \frac{1}{y_k^T s_k},$$

and

$$V_k = I - \rho_k y_k s_k^T.$$

The algorithm of L-BFGS method is shown as follow:

Step 1 **:** Choose x_0, m, $0 < \beta' < \frac{1}{2}$, $\beta' < \beta < 1$, and a symmetric and positive definite matrix H_0. Set $k = 0$.

Step 2 : Compute

$$d_k = -H_k g_k,$$
$$x_{k+1} = x_k + \alpha_k d_k,$$

where α_k satisfies the Wolfe conditions below:

$$f(x_k + \alpha_k d_k) \le f(x_k) + \beta' \alpha_k g_k^T d_k,$$
$$g(x_k + \alpha_k d_k)^T d_k \ge \beta g_k^T d_k,$$

but we always try the steplength $\alpha_k = 1$ first.

Step 3 : Let $\hat{m} = \min\{k, m-1\}$. Update H_0, $\hat{m}+1$, times using the pairs $\{y_j, s_j\}_{j=k-\hat{m}}^{k}$, i.e. let

$$\begin{aligned} H_{k+1} = &\left(V_k^T \cdots V_{k-\hat{m}}^T\right) H_0 \left(V_{k-\hat{m}} \cdots V_k\right) \\ &+ \rho_{k-\hat{m}} \left(V_k^T \cdots V_{k-\hat{m}+1}^T\right) s_{k-\hat{m}} s_{k-\hat{m}}^T \left(V_{k-\hat{m}+1} \cdots V_k\right) \\ &+ \rho_{k-\hat{m}+1} \left(V_k^T \cdots V_{k-\hat{m}+2}^T\right) s_{k-\hat{m}+1} s_{k-\hat{m}+1}^T \left(V_{k-\hat{m}+2} \cdots V_k\right) \\ &\;\vdots \\ &+ \rho_k s_k s_k^T. \end{aligned}$$

Step 4 : Set $k := k+1$ and go to Step 2.

From the Algorithm above, the matrices H_k are not formed explicitly, but the $\hat{m}+1$ previous values of y_j and S_j are stored separately. There is a efficient formula, due to Strang, for computing the product of $H_k g_k$ [10]. The implementation of L-BFGS method coincides with the one given in [10], except for one detail: the line search is not forced to perform at least one cubic interpolation, but the unit steplength is always tried first, and if it satisfies the Wolfe conditions, it is accepted. The main aim is that the limited memory methods resemble BFGS as much as possible, and disregard quadratic termination properties, which are not very meaningful, in general, for large dimensional problems.

The key issue here is how to choose the subspace S_k. Stoer and Yuan (1995) suggest the choice for the subspace S_k is a generalization of the 2-dimensional subspace, namely $S_k = \text{span}\{-g_k, s_{k-1}, \cdots, s_{k-m}\}$, since all the points in S_k can be expressed by

$$d = -\sigma g_k + \sum_{i=1}^{m} \beta_i s_{k-i}, \tag{4}$$

using the following approximations

$$s_{k-i}^T \nabla^2 f(x_k) s_{k-i} \approx s_{k-i}^T y_{k-i}, \qquad s_{k-i}^T \nabla^2 f(x_k) g_k \approx y_{k-i}^T g_k.$$

However, the performance of a Conjugate Gradient-like search direction can be very slow on certain type of nonlinear problem such as ill-conditioned problems. Hence, our main aim of the study is to propose some preconditioners for the search direction (4), namely,

$$d = -D_k g_k + \sum_{i=1}^{m} \beta_i s_{k-i}, \tag{5}$$

where D_k is the preconditioner in diagonal matrix form and it suppose to have some properties of the Hessian matrix, or a good approximation to Hessian matrix in some sense.

3. DERIVATION OF THE DIAGONAL PRECONDITIONER

In this section, we develop a preconditioner for LMQN algorithm in order to overcome the deficiency of the standard subspace limited memory algorithm when solving ill-conditioned optimization problems.

We shall choose a diagonal matrix D_k that satisfy the weak-quasi-Newton relation as below:

$$s_k^T D_{k+1} s_k = s_k^T y_k, \tag{6}$$

where $y_k = g_{k+1} - g_k$, and $s_k = x_{k+1} - x_k$.

Suppose that the Hessian matrix A of an objective function $f(x)=\frac{1}{2}x^T Ax-b^T x$ is positive definite. We let D_k be a diagonal matrix to approximate the Hessian matrix. Hence, we form our approximation as follow:

$$D_{k+1} = D_k + \Delta_k. \tag{7}$$

Our purpose is to construct a D_{k+1} such that it is a good approximation to the actual Hessian matrix.

Theorem
Assume that $D_k > 0$ is a positive definite diagonal matrix and D_{k+1} is the updated version of D_k, which is also diagonal. Suppose that $s_k \neq 0$, then the optimal solution of the following minimization problem:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|\Delta_k\|_F^2 \\ &\text{subject to } s_k^T D_{k+1} s_k = s_k^T y_k, \end{aligned} \tag{8}$$

is given by

$$D_{k+1} = D_k + \frac{\omega_k - \mu_k}{\gamma_k} G_k \tag{9}$$

where $\|\Delta_k\|_F = \sqrt{tr(\Delta_k^T \Delta_k)}$ is the Frobenius norm and tr is the trace operator, $\omega_k = s_k^T y_k$, $\mu_k = s_k^T D_k s_k$, $\gamma_k = \sum_{i=1}^{n}\left(s_k^{(i)}\right)^4$ and $G_k = \operatorname{diag}\left(\left(s_k^{(1)}\right)^2,\ldots,\left(s_k^{(n)}\right)^2\right)$ with $s_k^{(n)}$ being the $n-th$ component of the s_k.

Proof.
Let $\Delta_k = \begin{pmatrix} a_k^{(1)} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & a_k^{(n)} \end{pmatrix}$, $s_k = \begin{pmatrix} s_k^{(1)} \\ \vdots \\ s_k^{(n)} \end{pmatrix}$ and $y_k = \begin{pmatrix} y_k^{(1)} \\ \vdots \\ y_k^{(n)} \end{pmatrix}$.

From equation (8), we have

$$\|\Delta_k\|^2 = \left(\sqrt{tr(\Delta_k)^T(\Delta_k)}\right)^2$$
$$= \left(\left(a_k^{(1)}\right)^2 + \ldots + \left(a_k^{(i)}\right)^2 + \ldots + \left(a_k^{(n)}\right)^2\right). \quad (10)$$

Thus the minimization equation will become

$$\text{minimize } \frac{1}{2}\left(\left(a_k^{(1)}\right)^2 + \ldots + \left(a_k^{(i)}\right)^2 + \ldots + \left(a_k^{(n)}\right)^2\right). \quad (11)$$

By substituting (7) into (8), we obtain

$$s_k^T(D_k + \Delta_k)s_k = s_k^T y_k . \quad (12)$$

We expand (12) as:

$$s_k^T D_k s_k + s_k^T \Delta_k s_k = s_k^T y_k .$$

Rearrange the equation above, we get

$$\mu - \omega + \sum_{i=1}^{n} (s_k^{(i)})^2 a_k^{(i)} = 0 , \quad (13)$$

where $\mu = s_k^T D_k s_k$ and $\omega = s_k^T y_k$.

From (13), we have

$$\sum_{i=1}^{n} (s_k^{(i)})^2 a_k^{(i)} = \omega - \mu . \quad (14)$$

Finally, we wish to solve the following:

$$\text{minimize } \frac{1}{2}\left(\left(a_k^{(1)}\right)^2 + \ldots + \left(a_k^{(i)}\right)^2 + \ldots + \left(a_k^{(n)}\right)^2\right)$$

$$\text{subject to } \mu - \omega + \sum_{i=1}^{n} (s_k^{(i)})^2 a_k^{(i)} = 0 . \tag{15}$$

Since the objective function in (15) is convex, then there exists a unique solution and its Lagrange function will be

$$L = \frac{1}{2}\left(\left(a_k^{(1)}\right)^2 + \ldots + \left(a_k^{(n)}\right)^2\right) + \lambda\left(\mu - \omega + \sum_{i=1}^{n} (s_k^{(i)})^2 a_k^{(i)}\right), \tag{16}$$

where λ is the Lagrange multiplier associated with the constant. We differentiate (16) with respect to $a_k^{(i)}$, and set the result to zero, we obtain,

$$\frac{\partial L}{\partial a_k^{(i)}} = a_k^{(i)} + \lambda (s_k^{(i)})^2 = 0 . \tag{17}$$

From (17), it is clear that

$$\lambda (s_k^{(i)})^2 = -a_k^{(i)}. \tag{18}$$

Multiplying (18) with $(s_k^{(i)})^2$ for $i = 1,2,3,\ldots,n$, respectively, we shall obtain

$$\lambda \left(s_k^{(i)}\right)^4 = -(s_k^{(i)})^2 a_k^{(i)} . \tag{19}$$

Summing all of the equation in (19) yields

$$\lambda \sum_{i=1}^{n} \left(s_k^{(i)}\right)^4 = -\sum_{i=1}^{n} (s_k^{(i)})^2 a_k^{(i)} . \tag{20}$$

By equation (14), (20) becomes

$$\lambda \sum_{i=1}^{n} \left(s_k^{(i)}\right)^4 = \mu - \omega , \tag{21}$$

Finally, we get

$$\lambda = \frac{\mu - \omega}{\gamma}, \tag{22}$$

where $\gamma = \sum_{i=1}^{n} \left(s_k^{(i)}\right)^4$.

Once again, from (18), we get

$$a_k^{(i)} = -\lambda (s_k^{(i)})^2. \tag{23}$$

We substitute (22) into (23), the equation becomes

$$a_k^{(i)} = \frac{\omega_k - \mu_k}{\gamma_k} (s_k^{(i)})^2. \tag{24}$$

Expression (24) is in the form of each component of i. By substituting (24) into the formula of Δ_k, we will get the approximation of D_{k+1} as follow:

$$D_{k+1} = D_k + \frac{\omega_k - \mu_k}{\gamma_k} G_k, \tag{25}$$

where $\omega = s_k^T y_k$, $\mu = s_k^T D_k s_k$, $\gamma = \sum_{i=1}^{n} \left(s_k^{(i)}\right)^4$ and $G_k = \mathrm{diag}\left(\left(s_k^{(1)}\right)^2, \ldots, s_k^{(n)2}\right)$ with $s_k^{(n)}$ being the $n-th$ component of the s_k, and the proof is completed.

Now, we give our algorithm for solving large scale unconstrained optimization, which is called the preconditioned limited memory quasi-Newton algorithm.

LMQN Algorithm

Step 1 : Set $k = 0$; select the initial point x_0 and ε as a stopping condition. We also set $D_0 = I$, where I is $n \times n$ identity matrix.

Step 2 : For $k \geq 0$, compute $g_k = Ax_k - b$. If $\|g_k\| \leq \varepsilon$, stop, else compute D_k where D is a specific diagonal preconditioner.

Step 3 : Compute $d_{k+1} = -D_{k+1} g_{k+1} + \sum_{i=1}^{m} \beta_i s_{k+1-i}$, where $\beta_i = \frac{g_{i+1}^T A d_i}{d_i^T A d_i}$,

$i \le \min\{k, m\}$.

Step 4 : Compute $\alpha_k = -\frac{g_k^T d_k}{d_k^T A d_k}$.

Step 5 : Hence, $x_{k+1} = x_k + \alpha_k d_k$.

Step 6 : Set $k := k+1$; go to step 2.

The LMQN method is tested where in step 2, D is chosen from theorem above.

4. CONVERGENCE ANALYSIS

In this section, we shall look at the convergence properties of the LMQN method. Note that all the Hessian approximations are obtained by updating a bounded matrix using our proposed preconditioned LMQN method. We will prove the convergence properties of our proposed methods based upon the convergence assumptions given by Liu and Nocedal (1989) since it is valid for our preconditioning formulae whose matrices are diagonal and positive definite.

Assumption

(1) The objective function f is twice continuously differentiable.

(2) The level set $D = \{x \in \Re^n : f(x) \le f(x_0)\}$ is convex.

(3) There exist positive constants M_1 and M_2 such that

$$M_1 \|z\|^2 \le z^T G(x) z \le M_2 \|z\|^2 \tag{26}$$

for $\forall z \in \Re^n$ and $\forall z \in D$. This implies that the objective function f has a unique minimize x^* in D.

From (25), we can have another similar inequality as below

$$N_1 \|z\|^2 \le z^T G(x)^{-1} z \le N_2 \|z\|^2, \tag{27}$$

where $N_1 = \dfrac{1}{M_2}$ and $N_2 = \dfrac{1}{M_1}$ are the constants.

Lemma

Let x_0 be a starting point for which f satisfies Assumptions above, and we takes $D_0 = I$, where I is the $n \times n$ identity matrix. Assume that the matrices D_k^0 are chosen so that $\left\{\left\| D_k^{(0)} \right\|\right\}$ and $\left\{\left\| D_k^{(0)-1} \right\|\right\}$ are bounded. Then, $\{D_{k+1}\}$ and $\left\{\left\| D_{k+1}^{-1} \right\|\right\}$ are also bounded, where

$$D_{k+1} = D_k + \frac{\omega_k - \mu_k}{\gamma_k} G_k \tag{28}$$

where $\omega_k = s_k^T y_k$, $\mu_k = s_k^T D_k s_k$, $\gamma_k = \sum_{i=1}^{n} \left(s_k^{(i)}\right)^4$ and $G_k = \text{diag}\left(\left(s_k^{(1)}\right)^2, \cdots, \left(s_k^{(n)}\right)^2\right)$ with $s_k^{(n)}$ being the $n-th$ component of the s_k respectively.

Proof.

Without the loss of generality, we shall assume that $D_0 = I$, where I is the $n \times n$ identity matrix. It is clear that D_0 is bounded as follow:

$$\mu_0 \leq \| D_0 \|_F \leq \omega_0 \tag{29}$$

We shall prove this Lemma by using mathematical induction. Now, we shall prove that $\| D_1 \|_F$ is bounded. If $s_0^T y_0 - s_0^T D_0 s_0 \leq 0$, then by LMQN algorithm, we have $D_1 = D_0$ which implies that $\mu_0 \leq \| D_0 \|_F = \| D_1 \|_F \leq \omega_0$. Hence, we shall prove for the case, $s_0^T y_0 - s_0^T D_0 s_0 > 0$ and $s_k^T y_k - s_k^T D_k s_k > 0$.

Let $\nabla^2 f(\bar{x})$ be defined as

$$\nabla^2 f(\bar{x}) = \int_0^1 \nabla^2 f(x_k + \tau s_k) d\tau .$$

Then, we shall obtain

$$y_k = \nabla^2 f(\bar{x}) s_k . \tag{30}$$

From (26) and (30), we get

$$M_1 \| s_k \|^2 \le s_k^T y_k \le M_2 \| s_k \|^2 , \tag{31}$$

where M_1 and M_2 are constants.

From (31), we have

$$s_0^T y_0 \le M_2 \| s_k \|^2 . \tag{32}$$

From (29), it leads to

$$\mu_0 \| s_0 \|^2 \le s_0^T D_0 s_0 \le \omega_0 \| s_0 \|^2 . \tag{33}$$

From (32) and (33), we yield

$$s_0^T y_0 - s_0^T D_0 s_0 \le | M_2 - \mu_0 | \| s_0 \|^2 . \tag{34}$$

We let

$$\begin{aligned} \| s_0 \|^2 &= s_0^{(1)2} + s_0^{(2)2} + \ldots + s_0^{(n)2} \\ &\le n s_{0m}^2 , \end{aligned} \tag{35}$$

where $s_{0m}^2 = \max \left\{ s_0^{(1)2}, s_0^{(2)2}, \ldots, s_0^{(n)2} \right\}$.

From (28), we obtain

$$\begin{aligned} \| G_0 \|_F^2 &= \mathrm{tr}\left(G_0^T G_0 \right), \\ &= s_0^{(1)4} + s_0^{(2)4} + \ldots + s_0^{(n)4} . \end{aligned}$$

Finally, we should have

$$\| G_0 \|_F \le \sqrt{n} s_{0m}^2 . \tag{36}$$

Hence,

$$\begin{aligned}
\| D_1 \|_F &= \| D_0 + \frac{s_0^T y_0 - s_0^T D_0 s_0}{\sum_{i=1}^{n} \left(s_0^{(i)}\right)^4} G_0 \|_F, \\
&\leq \| D_0 \|_F + \frac{| s_0^T y_0 - s_0^T D_0 s_0 |}{\sum_{i=1}^{n} \left(s_0^{(i)}\right)^4} \| G_0 \|_F, \\
&\leq \| D_0 \|_F + \frac{| M_2 - \mu_0 | n s_{0m}^2}{\sum_{i=1}^{n} \left(s_0^{(i)}\right)^4} \sqrt{n} s_{0m}^2, \\
&\leq \| D_0 \|_F + \frac{k n^{\frac{3}{2}} s_{0m}^4}{\sum_{i=1}^{n} \left(s_0^{(i)}\right)^4}, \\
&\leq \| D_0 \|_F + M_4, \qquad (37)
\end{aligned}$$

where $k = \max\{(M_2 - \mu_0), (M_2 + \mu_0)\}$, $M_4 = kn^{\frac{3}{2}}$ and since

$$\frac{s_{0m}^4}{\sum_{i=1}^{n} \left(s_0^{(i)}\right)^4} \leq 1.$$

From (37), we can conclude that $\| D_1 \|_F$ is bounded since $\| D_0 \|_F$ is bounded. Now, we assume D_k is bounded, then we need to prove that D_{k+1} is also bounded.

From above, we shall get the similar inequalities as follow:

$$\| G_0 \|_F \leq \sqrt{n} s_m^2, \qquad (38)$$

$$\| s_k \|^2 \leq n s_m^2, \qquad (39)$$

$$s_k^T y_k - s_k^T D_k s_k \leq | M_2 - \mu_k | \| s_k \|^2 . \qquad (40)$$

From (28) and (38)-(40), we obtain

$$\| D_{k+1} \|_F \leq \| D_k \|_F + M_4, \tag{41}$$

where $M_4 = kn^{\frac{3}{2}}$, and $k = \max\{(M_2 - \mu_k), (M_2 + \mu_k)\}$.

From the fact that $\| D_k \|_F$ is bounded, i.e. $\| D_k \|_F \leq M_5$. Thus, from (41),

$$\begin{aligned} \| D_{k+1} \|_F &\leq M_5 + M_4, \\ &\leq M_6, \end{aligned}$$

where $M_6 = M_5 + M_4$ and it is a constant. Finally, we have shown that $\| D_{k+1} \|_F$ is bounded and the proof is completed.

In this section, we have shown that the proposed preconditioned LMQN methods are to be convergent on uniformly convex problems and the rate is $R-$ linear. This $R-$ linear convergence results obtained are based upon the assumption by Liu and Nocedal (1989).

5. COMPUTATIONAL RESULTS AND DISCUSSION

In this section, the computational results and discussion on the performance of preconditioner limited memory quasi-Newton (LMQN) method will be proposed. All algorithms are written in MATLAB 7.0. The total number of tested problems is 4. All the runs were terminated when

$$\|g_k\| \leq 10^{-4},$$

where $\|\cdot\|$ denotes the Euclidean norm. Furthermore, we also consider the number of function evaluation and gradient calls. We set our upper bound for the number of function evaluation and gradient call is 1000.

The computational results are compared through number of iterations, gradient evaluations as well as function evaluations. In order to test the efficiency of the proposed preconditioned methods, the number of subspaces that we will consider is $m = 2$ and $m = 3$.

The LMQN method was tested using the following preconditioners:

1. LMQN(0)-SQN method without preconditioning.
2. LMQN(D1)-SQN method with diagonal preconditioner D where D is given by theorem above.

In order to compare the efficiency of our proposed preconditioned LMQN methods with the standard LMQN method, we have considered the following quadratic test problem

$$f(x) = \frac{1}{2} x^T A x - b^T x, \tag{42}$$

where A is positive definite diagonal matrix and $b = [1,1,1,1,1,\ldots,1]$.

For all methods, the initial points is $x_0 = [0,0,0,0,\ldots,0]$. A set of unconstrained minimization quadratic problems, consisting of 4 test problems, were used. We now describe the 4 different quadratic test problems (42) with n-dimensional cases.

1. QF1, where $A = \operatorname{diag}[a_{ii}]$, $a_{ii} = i^2 (\operatorname{mod} 5)$, $b = [1,\ldots,1]$.
2. QF2, where $A = \operatorname{diag}[a_{ii}]$, $a_{ii} = i^3 (\operatorname{mod} 5)$, $b = [1,\ldots,1]$.
3. QF3, where $A = \operatorname{diag}[a_{ii}]$, $a_{ii} = i^3 + i (\operatorname{mod} 5)$, $b = [1,\ldots,1]$.
4. QF4, where $A = \operatorname{diag}[a_{ii}]$, $a_{ii} = a_{i-2,i-2} + a_{i-1,i-1}$, $i \geq 3$ and $a_{11} = 1$, $a_{22} = 1$, $b = [1,\ldots,1]$.

We tested the above problems by using $m = 2$ and $m = 3$. In each table, the symbol Ite, $\|g_k\|$, and Fva mean the number of iterations, norm of the gradient and function evaluation respectively.

TABLE 1: Comparison of the Methods of $m=2$ in solving QF1

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	106	9.2e-5	-1.4636	31	4.2e-5	-1.4636
20	109	9.8e-5	-2.9272	31	6.0e-5	-2.9272
40	113	9.6e-5	-5.8544	31	8.4e-5	-5.8544
80	117	9.3e-5	-1.1709e+1	32	1.5e-5	-1.1709e+1
100	118	9.5e-5	-1.4636e+1	32	1.6e-5	-1.4636e+1
200	122	9.3e-5	-2.9272e+1	32	2.3e-5	-2.9272e+1
500	127	9.2e-5	-7.3181e+1	32	3.6e-5	-7.3181e+1
1000	130	9.9e-5	-1.4636e+2	32	5.2e-5	-1.4636e+2
1500	133	9.1e-5	-2.1954e+2	32	6.3e-5	-2.1954e+2
2000	134	9.6e-5	-2.9272e+2	32	7.3e-5	-2.9272e+2

TABLE 2: Comparison of the Methods of $m=2$ in solving QF2

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	598	9.9e-5	-1.1857	34	9.7e-5	-1.1857
20	619	9.9e-5	-2.3713	36	8.2e-5	-2.3713
40	640	9.9e-5	-4.7426	38	8.3e-5	-4.7426
80	661	9.9e-5	-9.4853	39	5.8e-5	-9.4853
100	668	9.9e-5	-1.1857e+1	39	6.5e-5	-1.1857e+1
200	689	9.9e-5	-2.3713e+1	39	9.2e-5	-2.3713e+1
500	716	1.0e-4	-5.9283e+1	40	6.6e-5	-5.9283e+1
1000	737	1.0e-4	-1.1857e+2	40	9.4e-5	-1.1857e+2
1500	750	9.9e-5	-1.7785e+2	42	6.7e-5	-1.7785e+2
2000	758	1.0e-4	-2.3713e+2	42	7.7e-5	-2.3713e+2

TABLE 3: Comparison of the Methods of $m=2$ in solving QF3

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	311	9.7e-5	-6.5573e-1	65	2.4e-5	-6.5573e-1
20	322	9.7e-5	-1.3115	65	3.4e-5	-1.3115
40	332	1.0e-4	-2.6229	65	4.9e-5	-2.6229
80	343	1.0e-4	-5.2459	65	6.9e-5	-5.2459
100	347	9.8e-5	-6.5573	65	7.7e-5	-6.5573
200	358	9.8e-5	-1.3115e+1	67	9.4e-5	-1.3115e+1
500	372	9.9e-5	-3.2787e+1	71	9.6e-5	-3.2787e+1
1000	383	9.9e-5	-6.5573e+1	75	9.9e-5	-6.5573e+1
1500	390	9.7e-5	-9.8360e+1	84	7.6e-5	-9.8360e+1
2000	394	9.9e-5	-1.3115e+2	84	9.1e-5	-1.3115e+2

TABLE 4: Comparison of the Methods of $m = 2$ in solving QF4

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	252	9.7e-5	-1.6652	66	6.2e-5	-1.6652
20	261	9.7e-5	-3.3305	66	8.8e-5	-3.3305
40	270	9.6e-5	-6.6609	71	9.9e-5	-6.6609
80	278	1.0e-4	-1.3322e+1	78	7.8e-5	-1.3322e+1
100	281	9.9e-5	-1.6652e+1	78	8.7e-5	-1.6652e+1
200	290	9.9e-5	-3.3305e+1	87	3.4e-5	-3.3305e+1
500	301	9.8e-5	-8.3262e+1	87	5.4e-5	-8.3262e+1
1000	311	9.7e-5	-1.6652e+1	87	7.7e-5	-1.6652e+1
1500	316	9.8e-5	-2.4979e+1	87	9.4e-5	-2.4979e+1
2000	320	9.7e-5	-3.3305e+2	88	7.1e-5	-3.3305e+2

TABLE 5: Comparison of the Methods of $m = 3$ in solving QF1

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	81	9.8e-5	-1.4636	53	8.0e-5	-1.4636
20	84	9.6e-5	-2.9272	54	3.0e-5	-2.9272
40	87	9.4e-5	-5.8544	54	4.2e-5	-5.8544
80	90	9.3e-5	-1.1709e+1	54	6.0e-5	-1.1709e+1
100	91	9.2e-5	-1.4636e+1	54	6.7e-5	-1.4636e+1
200	94	9.0e-5	-2.9272e+1	54	9.5e-5	-2.9272e+1
500	97	9.9e-5	-7.3181e+1	55	6.2e-5	-7.3181e+1
1000	100	9.7e-5	-1.4636e+2	55	8.8e-5	-1.4636e+2
1500	102	9.4e-5	-2.1954e+2	61	7.0e-5	-2.1954e+2
2000	103	9.6e-5	-2.9272e+2	61	8.1e-5	-2.9272e+2

TABLE 6: Comparison of the Methods of $m = 3$ in solving QF2

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	577	1.0e-4	-1.1857	157	8.9e-5	-1.1857
20	598	9.8e-5	-2.3713	143	3.2e-5	-2.3713
40	618	9.9e-5	-4.7426	134	9.7e-5	-4.7426
80	638	9.9e-5	-9.4853	154	9.9e-5	-9.4853
100	645	9.9e-5	-1.1857e+1	159	7.7e-5	-1.1857e+1
200	665	9.9e-5	-2.3713e+1	140	9.7e-5	-2.3713e+1
500	692	9.9e-5	-5.9283e+1	192	2.7e-5	-5.9283e+1
1000	712	9.9e-5	-1.1857e+2	205	8.6e-5	-1.1857e+2
1500	724	9.9e-5	-1.7785e+2	218	7.8e-5	-1.7785e+2
2000	732	9.9e-5	-2.3713e+2	158	9.2e-5	-2.3713e+2

TABLE 7: Comparison of the Methods of $m = 3$ in solving QF3

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	300	9.8e-5	-6.5573e-1	75	8.4e-5	-6.5573e-1
20	310	1.0e-4	-1.3115	77	8.9e-5	-1.3115
40	321	9.8e-5	-2.6229	80	5.4e-5	-2.6229
80	332	9.7e-5	-5.2459	82	7.3e-5	-5.2459
100	335	9.8e-5	-6.5573	82	8.2e-5	-6.5573
200	346	9.7e-5	-1.3115e+1	85	8.5e-5	-1.3115e+1
500	359	1.0e-4	-3.2787e+1	86	8.2e-5	-3.2787e+1
1000	370	9.9e-5	-6.5573e+1	99	5.5e-5	-6.5573e+1
1500	376	9.9e-5	-9.8360e+1	88	7.5e-5	-9.8360e+1
2000	381	9.7e-5	-1.3115e+2	90	9.1e-5	-1.3115e+2

TABLE 8: Comparison of the Methods of $m = 3$ in solving QF4

	LMQN(0)			LMQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	230	9.8e-5	-1.6652	81	9.8e-5	-1.6652
20	238	9.8e-5	-3.3305	90	9.4e-5	-3.3305
40	246	9.9e-5	-6.6609	93	8.4e-5	-6.6609
80	254	9.9e-5	-1.3322e+1	98	9.3e-5	-1.3322e+1
100	257	9.8e-5	-1.6652e+1	95	7.6e-5	-1.6652e+1
200	265	9.8e-5	-3.3305e+1	103	5.2e-5	-3.3305e+1
500	276	9.7e-5	-8.3262e+1	103	9.6e-5	-8.3262e+1
1000	284	9.8e-5	-1.6652e+1	114	7.6e-5	-1.6652e+1
1500	289	9.7e-5	-2.4979e+1	107	8.1e-5	-2.4979e+1
2000	292	9.8e-5	-3.3305e+2	106	9.3e-5	-3.3305e+2

The number of iterations is the successive in a computational method. In this study, we will compare the number of iterations between the standard LMQN method and the four proposed LMQN methods.

Tables 1-4 show the comparison results between proposed preconditioned SLMQN methods and standard LMQN method for $m = 2$. Generally, the computational results show that the proposed methods are performed better when compare to that standard LMQN method. As in the Tables, the proposed methods required less number of iterations than the standard method.

Although all the methods show the same values of function evaluation, but the norms of gradient for the proposed methods are less than the norms of gradient of the standard method. Once again, this shows that the proposed LMQN methods are promising alternative compared to the standard LMQN method.

Tables 5-8 show the comparison results between proposed preconditioned LMQN methods and standard LMQN method for $m = 3$. Once again, the results show that the proposed methods clearly outperform than the standard method. The number of iterations and the norms of the gradient are the best evidence to show that our proposed methods generally have performed well than the standard LMQN method.

6. CONCLUSION

Our tests indicate that the implementation of the proposed preconditioned LMQN method performs better than the standard LMQN method. The computational results have convinced us that the preconditioned LMQN method is a good alternative for large scale unconstrained optimization. The preconditioned LMQN method is appealing for several reasons: it is easy to implement; it requires only function and gradient values and lastly it works better than the standard LMQN method. In conclusion, our proposed preconditioned LMQN method is inexpensive and required only minimal storage, thus, it is worth to extend the use of this method.

ACKNOWLEDGEMENTS

The arthurs gratefully acknowledged the financial support of Graduate Research Fellowship (GRF) from Universiti Putra Malaysia and Ministry of Higher Education Malaysia.

REFERENCES

Dennis, J. E. and Schnabel, R. B. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* England Cliffs: New Jersey Prentice-Hall.

Farid, M., Hassan, M. A. and Leong, W. J. 2011. Improved Hessian approximation with modified secant equations for symmetric rank-one method. *J. Comput. Appl. Math.* **235**: 2423-2431.

Farid, M., Leong, W. J. and Hassan, M. A. 2010. A new two-step gradient type method for large scale unconstrained optimization. *Computers and Mathematics with Applications*. **59**: 3301-3307.

Farid, M., Leong, W. J. and Hassan, M. A. 2010. An improved multi-step gradient-type method for large scale optimization. *Computers and Mathematics with Applications.* **61**: 3312-3318.

Leong, W. J. and Hassan, M. A. 2009. A restarting approach for the symmetric rank one update for unconstrained optimization. *Computational Optimization and applications.* **43**: 327-334.

Leong, W. J. and Hassan, M. A. 2011. A new gradient method via least change secant update. *International Journal of Computer Mathematics.* **88**: 816-828.

Leong, W. J., Hassan, M. A. and Farid, M. 2010. A monotone gradient method via weak secant equations for unconstrained optimization. *Taiwanese J. Math.* **14**(2): 413-423.

Leong, W. J., Mahboubeh, F., and Malik, A. H. 2010. Improved Hessian approximation with modified quasi-Cauchy relation for a gradient type method. *Advanced Modeling and Optimization.* **12**: 37-44.

Liu, D. C. and Nocedal, J. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming.* **45**: 503-528.

Nocedal, J. 1980. Updating quasi_newton matrices with limited storage. *Mathematics of Computation.* **35**: 773-782.

Stoer, J. and Yuan, Y. 1995. A subspace study on conjugate gradient algorithms. *ZAMMZ. Angew. Math. Mech.* **75**: 69-77.

Waziri, M. Y., Leong, W. J., Hassan, M. A. and Monsi, M. 2010. A new Newton's method with diagonal Jacobian approximation for systems of nonlinear equations. *J. Mathematics and Statistics.* **6**: 246-252.

Waziri, M. Y., Leong, W. J., Hassan, M. A. and Monsi, M. 2010. Jacobian computation-free Newton method for systems of non-linear equations. *Journal of numerical Mathematics and Stochastic.* **2**(1): 54-63.