Pertanika J. Sci. & Technol. 22 (1): 175 - 192 (2014)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Preconditioned Subspace Quasi-Newton Method for Large Scale Optimization

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

[1]Institute for Mathematical Research, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia
[2]Department of Mathematics, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

ABSTRACT

Subspace quasi-Newton (SQN) method has been widely used in large scale unconstrained optimization problem. Its popularity is due to the fact that the method can construct subproblems in low dimensions so that storage requirement as well as the computation cost can be minimized. However, the main drawback of the SQN method is that it can be very slow on certain types of non-linear problem such as ill-conditioned problems. Hence, we proposed a preconditioned SQN method, which is generally more effective than the SQN method. In order to achieve this, we proposed that a diagonal updating matrix that was derived based on the weak secant relation be used instead of the identity matrix to approximate the initial inverse Hessian. Our numerical results show that the proposed preconditioned SQN method performs better than the SQN method which is without preconditioning.

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

INTRODUCTION

Subspace quasi-Newton (SQN) method is generally used to solve large scale non-linear systems of equations and non-linear least square problems. This method is popular because it has the characteristic to force the next iteration in a low dimensional subspace. At each iteration, we searched for a minimum of the objective function over a subspace spanned by the current gradient and by direction of few previous steps.

The main advantage of this method is that it constructs subproblems in low dimensions so that computation cost can be reduced. It also offers a possible way to handle large scale unconstrained optimization problems. Besides, this method can be implemented extremely fast. This happens when the

Article history:
Received: 22 September 2011
Accepted: 11 November 2011

E-mail addresses:
hongseng0505@hotmail.com (Hong Seng Sim),
leongwj@upm.edu.my (Wah June Leong),
malik@science.upm.edu.my (Malik Abu Hassan),
fudziah@upm.edu.my (Fudziah Ismail)
*Corresponding Author

ISSN: 0128-7680

objective function is a combination of expensive linear mappings with computationally cheap nonlinear functions (Yuan, 2007).

One of the famous subspace algorithms for non-linear optimization is the unbalance property shared by most line search algorithms. Any line search method is considered to have the following form:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1}$$

Where, d_k is the search direction and $\alpha_k \geq 0$ is the step-length that is computed by certain line search technique. Generally, the search direction d_k is computed by solving a subproblem which is an approximation to the original non-linear optimization problem. Therefore, there are two parts combined in each iteration of a line search algorithms; the first part is to find d_k in the whole n dimensional space, while the other part is to search for a suitable step-length in a fixed one dimensional space spanned by the computed d_k. As a result, the overall algorithm swings between the n dimensional search and one dimensional search alternately. Some variants of these methods can be found in [3], [4], [5], [9], [10], [11], [12], [17] and [18].

Furthermore, many well-known existing algorithms essentially have certain subspace features. For example, the conjugate gradient method uses a search direction in a two dimensional subspace spanned by the steepest descent direction and the previous step, the dog-leg method computes a step that is a convex combination of the steepest descent direction and the Newton's direction, and the limited memory quasi-Newton algorithms will also produce search directions that are spanned in a lower dimensional space to speed up the convergency and lower the computation cost.

SUBSPACE METHOD APPROACHES

The well-known nonlinear conjugate gradient methods use a linear combination of the steepest descent direction $-g_k$ and the previous search direction d_{k-1} to form the new search direction, as follows:

$$d_k = -g_k + \beta_k d_{k-1}$$

Hence, one of the important tasks is how to determine the suitable β_k based on certain conjugate gradient principles. Instead of the conjugate property, Stoer and Yuan (1995) suggested to look at the conjugate gradient method from the subspace point of view. In the conjugate gradient method, β_k is used to define search direction d_k and the stepsize α_k to set $x_{k+1} = x_k + \alpha_k d_k$; thus, no matter whatever β_k and α_k are used, the increment in the iterative point will be a linear combination of $-g_k$ and d_{k-1}. They consider a model subproblem as follows:

$$\min_{d \in span\{-g_k, s_{k-1}\}} Q_k(d) \approx f(x_k + d)$$

Let d_k be the solution of the above 2-dimensional subproblem and a successive 2-dimensional search algorithm is presented, which is an example of algorithms that using subspace methods (Stoer & Yuan, 1995).

Limited memory quasi-Newton method also has the subspace nature. The Quasi-Newton updates have the following form:

$$B_k = U\left(B_{k-1}, s_{k-1}, y_{k-1}\right)$$

which satisfies

$$B_k s_{k-1} = y_{k-1}$$

where, $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. A famous example is the BFGS method.

$$B_k = B_{k-1} - \frac{B_{k-1}s_{k-1}s_{k-1}^T B_{k-1}}{s_{k-1}^T B_{k-1} s_{k-1}} + \frac{y_{k-1}y_{k-1}^T}{s_{k-1}^T y_{k-1}}$$

The limited memory quasi-Newton updates the approximate Hessian repeatedly:

$$B_k^{(i)} = U\left(B_k^{(i-1)}, s_{k-m-1+i}, y_{k-m-1+i}\right) \qquad i = 1, 2, \ldots, m$$

with $B_k^{(0)} = \sigma_k I$ (Liu & Nocedal, 1989). There are various formulae for σ_k, with one choice being $\frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$. The limited memory quasi-Newton matrix can be written as follows:

$$B_k = B_k^{(m)} = \sigma_k I + \begin{bmatrix} S_k & Y_k \end{bmatrix} T_k \begin{bmatrix} S_k^T \\ Y_k^T \end{bmatrix}$$

where, T_k is a $2m \times 2m$ matrix, and

$$\begin{bmatrix} S_k & Y_k \end{bmatrix} = \left[s_{k-1}, s_{k-2}, \ldots, s_{k-m}, y_{k-1}, y_{k-2}, \ldots, y_{k-m}\right] \in \Re^{n \times 2m}.$$

We have $s_k = \alpha_k d_k = -\alpha_k B_k^{-1} g_k$ in the line search type method, while for a trust region type algorithm, we have $s_k = -\left(B_k + \lambda_k I\right)^{-1} g_k$. As a result,

$$s_k = -\left(\rho_k I + \begin{bmatrix} S_k & Y_k \end{bmatrix} T_k \begin{bmatrix} S_k^T \\ Y_k^T \end{bmatrix}\right)^{-1} g_k$$

$$\in span\left\{g_k, s_{k-1}, \ldots, s_{k-m}, y_{k-1}, \ldots, y_{k-m}\right\}.$$

It has been shown that no matter what, the limited memory quasi-Newton algorithm with line search or trust region will always produce a step in the subspace $span\left\{g_k, s_{k-1}, \ldots, s_{k-m}, y_{k-1}, \ldots, y_{k-m}\right\}$ (Wang *et al*., 2004).

A model subspace algorithm for unconstrained optimization is suggested, which is a slight modification of the standard trust region algorithm for unconstrained optimization.

Algorithm 2.1 (A Model Subspace Algorithm for Unconstrained Optimization)

Step 1: Given x_1, define S_1, $\varepsilon > 0$, $k := 1$.

Step 2: Solve a subspace subproblem:

$$\min_{d \in S_k} Q_k(d) = g_k^T d + \frac{1}{2} d^T B_k d \,. \tag{2}$$

to obtain s_k. If $\| s_k \| \leq \varepsilon$, then stop.

Step 3: Define

$x_{k+1} = x_k + s_k$ if $f(x_k + s_k) < f(x_k)$

$x_{k+1} = x_k$, otherwise.

Step 4: Generate S_{k+1} and $Q_{k+1}(d)$.

Step 5: Set $k := k + 1$, go to step 2.

The main difference between the above algorithm and the standard whole space algorithm is the constraint for the step S_k to be in the subspace S_k. Thus, the key issue here is how to choose the subspace S_k. Stoer and Yuan (1995) suggested that the choice for the subspace S_k is a generalization of the 2-dimensional subspace, namely, $S_k = span\{-g_k, s_{k-1}, \ldots, s_{k-m}\}$, since all the points in S_k can be expressed by:

$$d = -\sigma g_k + \sum_{i=1}^{m} \beta_i s_{k-i} \,, \tag{3}$$

using the following approximations,

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

However, the performance of a CG-like search direction can be very slow on certain types of non-linear problem such as ill-conditioned problems. Hence, the main aim of the study is to propose some preconditioners for the search direction (3), namely:

$$d_k = -D_k^{-1} g_k + \sum_{i=1}^{m} \beta_i s_{k-i} \tag{4}$$

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

DERIVATION OF THE DIAGONAL PRECONDITIONER

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

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

$$y_k^T D_{k+1} s_k = y_k^T y_k \tag{5}$$

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 follows:

$$D_{k+1} = D_k + \Delta_k \tag{6}$$

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

Theorem 3.1

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 $s_k \neq 0$, the optimal solution of the following minimization problem will then be:

$$\text{minimize } \frac{1}{2} \|\Delta_k\|_F^2$$

$$\text{subject to } y_k^T D_{k+1} s_k = y_k^T y_k \tag{7}$$

and is given by:

$$D_{k+1} = D_k + \frac{\omega_k - \mu_k}{\gamma_k} G_k \tag{8}$$

where, $\|\Delta_k\|_F = \sqrt{tr(\Delta_k^T \Delta_k)}$ is the Frobenius norm and *tr* is the trace operator, $\omega = y_k^T y_k$, $\mu = y_k^T D_k s_k$, $\gamma = \sum_{i=1}^{n} \left(y_k^{(i)} s_k^{(i)}\right)^2$ and $G_k = diag\left(\left(s_k^{(1)} y_k^{(1)}\right), \ldots, \left(s_k^{(i)} y_k^{(i)}\right)\right)$ with $y_k^{(i)}$ and $s_k^{(i)}$ being the $i = th$ component of the y_k and s_k respectively.

Proof

Let $\Delta_k = \begin{pmatrix} a_k^{(1)} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_k^{(k)} \end{pmatrix}$, $s_k = \begin{pmatrix} s_k^{(1)} \\ \vdots \\ s_k^{(n)} \end{pmatrix}$ and $y_k = \begin{pmatrix} y_k^{(1)} \\ \vdots \\ y_k^{(k)} \end{pmatrix}$ for $i = 1,2,3,\dots,n.$

From equation (7), we have:

$$\|\Delta_k\|^2 = \left(\sqrt{tr\left(\Delta_k\right)^T\left(\Delta_k\right)}\right)^2$$

$$= \left(\left(a_k^{(1)}\right)^2 + \dots + \left(a_k^{(i)}\right)^2 + \dots + \left(a_k^{(n)}\right)^2\right). \tag{9}$$

Thus, the minimization equation will become:

$$\text{minimize } \frac{1}{2}\left(\left(a_k^{(1)}\right)^2 + \dots + \left(a_k^{(i)}\right)^2 + \dots + \left(a_k^{(n)}\right)^2\right). \tag{10}$$

By substituting (6) into (7), we obtain:

$$y_k^T\left(D_k + \Delta_k\right)s_k = y_k^T y_k\,. \tag{11}$$

We expand (11) to get the following expression:

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

Rearranging the equation, we get:

$$\mu - \omega + \sum_{i=1}^{n} y_k^{(i)} s_k^{(i)} a_k^{(i)} = 0\,, \tag{12}$$

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

From (12), we have:

$$\sum_{i=1}^{n} y_k^{(i)} s_k^{(i)} a_k^{(i)} = \omega - \mu\,. \tag{13}$$

Finally, we wish to solve the following:

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

$$\text{subject to } \mu - \omega + \sum_{i=1}^{n} y_k^{(i)} s_k^{(i)} a_k^{(i)} = 0\,. \tag{14}$$

Since the objective function in (14) is convex, there exists a unique solution and its Lagrange function will be:

$$L = \frac{1}{2}\left(\left(a_k^{(1)}\right)^2 + \ldots + \left(a_k^{(n)}\right)^2\right) + \lambda\left(\mu - \omega + \sum_{i=1}^{n} y_k^{(i)} s_k^{(i)} a_k^{(i)}\right), \tag{15}$$

where, λ is the Lagrange multiplier associated with the constant. We differentiate (15) with respect to $a_k^{(i)}$, and setting the result to zero, we obtain,

$$\frac{\partial L}{\partial a_k^{(i)}} = a_k^{(i)} + \lambda y_k^{(i)} s_k^{(i)} = 0. \tag{16}$$

From (16), it is clear that,

$$\lambda y_k^{(i)} s_k^{(i)} = -a_k^{(i)}. \tag{17}$$

Multiplying (17) with $s_k^{(i)} y_k^{(i)}$ for $i = 1,2,3,\ldots,n$, respectively, we shall obtain

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

Summing all of the equations in (18) yields:

$$\lambda\sum_{i=1}^{n}\left(y_k^{(i)} s_k^{(i)}\right)^2 = -\sum_{i=1}^{n} y_k^{(i)} s_k^{(i)} a_k^{(i)}. \tag{19}$$

By equation (13), (19) becomes

$$\lambda\sum_{i=1}^{n}\left(y_k^{(i)} s_k^{(i)}\right)^2 = \mu - \omega, \tag{20}$$

Finally, we get

$$\lambda = \frac{\mu - \omega}{\gamma}, \tag{21}$$

where $\gamma = \sum_{i=1}^{n}\left(y_k^{(i)} s_k^{(i)}\right)^2$.

Once again, from (17), we get

$$a_k^{(i)} = -\lambda y_k^{(i)} s_k^{(i)}. \tag{22}$$

We substitute (21) into (22), the equation becomes,

$$a_k^{(i)} = \frac{\omega_k - \mu_k}{\gamma_k} y_k^{(i)} s_k^{(i)}. \tag{23}$$

Expression (23) is in the form of each component of Δ. By substituting (23) into the formula of Δ_k, we will get the approximation of D_{k+1} as follows:

$$D_{k+1} = D_k + \frac{\omega_k - \mu_k}{\gamma_k} G_k, \tag{24}$$

where $\omega = y_k^T y_k$, $\mu = y_k^T D_k s_k$, $\gamma = \sum_{i=1}^{n} \left(y_k^{(i)} s_k^{(i)}\right)^2$ and $G_k = diag\left(\left(s_k^{(1)} y_k^{(1)}\right), \ldots, \left(s_k^{(i)} y_k^{(i)}\right)\right)$

with $y_k^{(i)}$ and $s_k^{(i)}$ being the $i-th$ component of the y_k and s_k respectively, and the proof is completed.

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

Algorithm 3.1 SQN 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 $\beta_i = \frac{g_{i+1}^T A d_i}{d_i^T A d_i}$, $i \leq \min\{k, m\}$.

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 SQN method is tested where in Step 2, D is chosen from theorem 3.1.

CONVERGENCE ANALYSIS

In this section, we shall look at the convergence properties of the subspace quasi-Newton method. Note that all the Hessian approximations are obtained by updating a bounded matrix using the proposed preconditioned subspace quasi-Newton 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 4.1:

(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

$$M_1\|z\|^2 \le z^T G(x) z \le M_2\|z\|^2 \tag{25}$$

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:

$$N_1\|z\|^2 \le z^T G(x)^{-1} z \le N_2\|z\|^2, \tag{26}$$

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

Lemma 4.1

Let x_0 be a starting point for which f satisfies Assumptions 4.1, and we take $D_0 = I$, where I is the $n \times n$ identity matrix. Assume that the matrices D_k^0 are chosen so that $\{\|D_k^{(0)}\|\}$ and $\{\|D_k^{(0)-1}\|\}$ are bounded. Then, $\{D_{k+1}\}$ and $\{\|D_{k+1}^{-1}\|\}$ are also bounded, where,

$$D_{k+1} = D_k + \frac{\omega_k - \mu_k}{\gamma_k} G_k \tag{27}$$

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

$$\mu_0 \le \| D_0 |_F \le \omega_0$$

Now, we need to prove D_1 is bounded.

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

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

Then, we have,

$$y_k = \nabla^2 f\left(\bar{x}\right) s_k . \tag{28}$$

From (26) and (28), we get

$$N_1 \left\| y_k \right\|^2 \le y_k^T s_k \le N_2 \left\| y_k \right\|^2, \tag{29}$$

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

From (27), we have

$$\left\| D_1 \right\|_F = \left\| D_0 + \frac{\omega_0 - \mu_0}{\gamma_0} G_0 \right\|_F$$

$$\left\| D_1 \right\|_F \le \left\| D_0 \right\|_F + \left\| \frac{\omega_0 - \mu_0}{\gamma_0} G_0 \right\|_F$$

$$\left\| D_1 \right\|_F \le \left\| D_0 \right\|_F + \frac{\left(\omega_0 - \mu_0\right)}{\gamma_0} \left\| G_0 \right\|_F , \tag{30}$$

where $\left\| \cdot \right\|_F^2$ is the square of Frobenius norm and let *tr* be the trace operator.

Note that

$$\left\| y_0 \right\|^2 = y_0^{(1)2} + y_0^{(2)2} + \ldots + y_0^{(n)2}$$

$$\le n\left(y_0^M\right)^2 , \tag{31}$$

where $\left(y_0^M\right)^2 = \max\left\{y_0^{(1)2}, y_0^{(2)2}, \ldots, y_0^{(n)2}\right\}$.

From (29), (30) and (31), we will get

$$\begin{aligned}\left\|G_0\right\|_F^2 &= tr\left(G_0^T G_0\right)\\ &= s_0^{(1)2} y_0^{(1)2} + s_0^{(2)2} y_0^{(2)2} + \ldots + s_0^{(n)2} y_0^{(n)2}\\ &\le \left(y_0^T s_0\right)^2\end{aligned}$$

$$\| G_0 \| \le N_2 n\left(y_0^M\right)^2 . \tag{32}$$

From (27), we have

$$\begin{aligned}\omega_0 &= y_0^T y_0\\ &= \left\|y_0\right\|^2\\ &\le n\left(y_0^M\right)^2\end{aligned} \tag{33}$$

and

$$\begin{aligned}\mu_0 &= y_0^T D_0 s_0\\ \mu_0 &\le M_3 y_0^T s_0\\ \mu_0 &\le M_3 N_2 \left\|y_0\right\|^2\\ \mu_0 &\le M_3 N_2 \left(y_0^M\right)^2\end{aligned} \tag{34}$$

where, $M_3 = 1$.

Hence, from (30), (32), (33) and (34), we shall have

$$\left\|D_1\right\|_F \le \left\|D_0\right\|_F + \frac{\left(n\left(y_0^M\right)^2 - N_2 n\left(y_0^M\right)^2\right)}{\sum_{i=1}^{n}\left(s_0^{(i)} y_0^{(i)}\right)^2} N_2 n\left(y_0^M\right)^2$$

$$\left\|D_1\right\|_F \le \left\|D_0\right\|_F + \frac{\left(n - N_2 n\right)\left(y_0^M\right)^2}{\sum_{i=1}^{n}\left(s_0^{(i)} y_0^{(i)}\right)^2} N_2 n\left(y_0^M\right)^2$$

$$\left\|D_1\right\|_F \le \left\|D_0\right\|_F + \frac{\left(1 - N_2\right) N_2 n^2 \left(y_0^M\right)^4}{\sum_{i=1}^{n}\left(s_0^{(i)} y_0^{(i)}\right)^2}$$

$$\left\|D_1\right\|_F \le \left\|D_0\right\|_F + \frac{kN_2n^2\left(y_0^M\right)^4}{\sum_{i=1}^{n}\left(s_0^{(i)}y_0^{(i)}\right)^2}$$

$$\left\|D_1\right\|_F \le \left\|D_0\right\|_F + M_4$$

$$\left\|D_1\right\|_F \le n + M_4, \tag{35}$$

Where, $M_4 = kN_2n^2$ and $k = \max\{(1-N_2),(1+N_2)\}$, and

$$\frac{\left(y_0^M\right)^4}{\sum_{i=1}^{n}\left(s_0^{(i)}y_0^{(i)}\right)^2} \le 1 \tag{36}$$

From (35), we can conclude that $\left\|D_1\right\|_F$ is bounded since $\left\|D_0\right\|_F$ is also bounded. Now, we assume that D_k is bounded, and then, we need to prove that D_{k+1} is also bounded.

From the above, we shall get a similar equation and inequality, as follows:

$$\left\|G_k\right\|_F \le N_2n\left(y^M\right)^2, \tag{37}$$

$$\omega_k \le \left\|y_k\right\|^2, \tag{38}$$

$$\mu_k \le M_3N_2\left\|y_k\right\|^2, \tag{39}$$

$$\left\|y_k\right\|^2 \le n\left(y_k^M\right)^2. \tag{40}$$

From (27) and (37)-(40), we obtain

$$\left\|D_{k+1}\right\|_F \le \left\|D_k\right\|_F + M_4, \tag{41}$$

Where, $M_4 = kN_2n^2$ and $k = \max\{(1-M_3N_2),(1+M_3N_2)\}$.

From the fact that $\left\|D_k\right\|_F$ is bounded, i.e. $\left\|D_k\right\|_F \le M_5$, and from (41),

$$\left\|D_{k+1}\right\|_F \le M_5 + M_6$$

$$\left\|D_{k+1}\right\|_F \le M_6,$$

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

In this section, we have shown that the proposed preconditioned subspace quasi-Newton methods are 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).

COMPUTATIONAL RESULTS AND DISCUSSION

The computational results and discussion on the performance of preconditioner subspace quasi-Newton (SQN) method are given in this section. All the 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 to be 1000.

The computational results are compared through the 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 is considered is $m = 2$ and $m = 3$.

Meanwhile, the SQN method was tested using the following preconditioners:

1. SQN(0)-SQN method without preconditioning.
2. SQN(D1)-SQN method with diagonal preconditioner D, where D is given by Theorem 3.1.

In order to compare the efficiency of the proposed preconditioned SQN methods with the standard SQN method, the following quadratic test problem is considered:

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

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

For all the methods, the initial point 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 (43) with n-dimensional cases.

1. QF1, where $A = diag[a_{ii}]$, $a_{ii} = i^2 \pmod 5$, $b = [1,\ldots,1]$.
2. QF2, where $A = diag[a_{ii}]$, $a_{ii} = i^3 \pmod 5$, $b = [1,\ldots,1]$.
3. QF3, where $A = diag[a_{ii}]$, $a_{ii} = i^3 + i \pmod 5$, $b = [1,\ldots,1]$.
4. QF4, where $A = diag[a_{ii}]$, $a_{ii} = a_{i-2,i-2} + a_{i-1,i-1}$, $i \geq 3$ 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
A comparison of the Method of $m = 2$ in solving QF1

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	106	9.2e-5	-1.4636	40	6.2e-5	-1.4636
20	109	9.8e-5	-2.9272	40	8.8e-5	-2.9272
40	113	9.6e-5	-5.8544	41	6.6e-5	-5.8544
80	117	9.3e-5	-1.1709e+1	41	9.3e-5	-1.1709e+1
100	118	9.5e-5	-1.4636e+1	45	7.6e-5	-1.4636e+1
200	122	9.3e-5	-2.9272e+1	46	9.2e-5	-2.9272e+1
500	127	9.2e-5	-7.3181e+1	52	8.7e-5	-7.3181e+1
1000	130	9.9e-5	-1.4636e+2	53	8.9e-5	-1.4636e+2
1500	133	9.1e-5	-2.1954e+2	56	4.6e-5	-2.1954e+2
2000	134	9.6e-5	-2.9272e+2	56	5.3e-5	-2.9272e+2

TABLE 2
A comparison of the Method of $m = 2$ in solving QF2

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	598	9.9e-5	-1.1857	66	9.8e-5	-1.1857
20	619	9.9e-5	-2.3713	78	4.7e-5	-2.3713
40	640	9.9e-5	-4.7426	78	6.6e-5	-4.7426
80	661	9.9e-5	-9.4853	79	2.5e-5	-9.4853
100	668	9.9e-5	-1.1857e+1	79	3.5e-5	-1.1857e+1
200	689	9.9e-5	-2.3713e+1	79	5.5e-5	-2.3713e+1
500	716	1.0e-4	-5.9283e+1	79	6.2e-5	-5.9283e+1
1000	737	1.0e-4	-1.1857e+2	91	1.0e-5	-1.1857e+2
1500	750	9.9e-5	-1.7785e+2	102	7.0e-5	-1.7785e+2
2000	758	1.0e-4	-2.3713e+2	116	3.1e-5	-2.3713e+2

TABLE 3
A comparison of the Method of $m = 2$ in solving QF3

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	311	9.7e-5	-6.5573e-1	48	9.7e-5	-6.5573e-1
20	322	9.7e-5	-1.3115	54	8.1e-5	-1.3115
40	332	1.0e-4	-2.6229	58	7.4e-5	-2.6229
80	343	1.0e-4	-5.2459	64	9.4e-5	-5.2459
100	347	9.8e-5	-6.5573	66	7.2e-5	-6.5573
200	358	9.8e-5	-1.3115e+1	67	5.3e-5	-1.3115e+1
500	372	9.9e-5	-3.2787e+1	67	8.4e-5	-3.2787e+1
1000	383	9.9e-5	-6.5573e+1	68	5.3e-5	-6.5573e+1
1500	390	9.7e-5	-9.8360e+1	68	6.4e-5	-9.8360e+1
2000	394	9.9e-5	-1.3115e+2	68	7.4e-5	-1.3115e+2

TABLE 4
A comparison of the Method of $m = 2$ in solving QF4

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	252	9.7e-5	-1.6652	74	9.6e-5	-1.6652
20	261	9.7e-5	-3.3305	77	9.1e-5	-3.3305
40	270	9.6e-5	-6.6609	85	8.8e-5	-6.6609
80	278	1.0e-4	-1.3322e+1	86	8.1e-5	-1.3322e+1
100	281	9.9e-5	-1.6652e+1	86	9.1e-5	-1.6652e+1
200	290	9.9e-5	-3.3305e+1	92	8.2e-5	-3.3305e+1
500	301	9.8e-5	-8.3262e+1	100	1.0e-5	-8.3262e+1
1000	311	9.7e-5	-1.6652e+1	103	7.4e-5	-1.6652e+1
1500	316	9.8e-5	-2.4979e+1	103	9.5e-5	-2.4979e+1
2000	320	9.7e-5	-3.3305e+2	106	9.4e-5	-3.3305e+2

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

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	81	9.8e-5	-1.4636	36	8.6e-5	-1.4636
20	84	9.6e-5	-2.9272	38	8.1e-5	-2.9272
40	87	9.4e-5	-5.8544	39	7.7e-5	-5.8544
80	90	9.3e-5	-1.1709e+1	40	5.9e-5	-1.1709e+1
100	91	9.2e-5	-1.4636e+1	40	6.6e-5	-1.4636e+1

TABLE 5 (*continue*)

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
200	94	9.0e-5	-2.9272e+1	40	9.4e-5	-2.9272e+1
500	97	9.9e-5	-7.3181e+1	47	5.6e-5	-7.3181e+1
1000	100	9.7e-5	-1.4636e+2	47	8.0e-5	-1.4636e+2
1500	102	9.4e-5	-2.1954e+2	47	9.8e-5	-2.1954e+2
2000	103	9.6e-5	-2.9272e+2	48	9.5e-5	-2.9272e+2

TABLE 6
A comparison of the Method of $m = 3$ in solving QF2

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	577	1.0e-4	-1.1857	74	8.9e-5	-1.1857
20	598	9.8e-5	-2.3713	80	6.1e-5	-2.3713
40	618	9.9e-5	-4.7426	80	8.6e-5	-4.7426
80	638	9.9e-5	-9.4853	87	1.5e-5	-9.4853
100	645	9.9e-5	-1.1857e+1	87	6.2e-5	-1.1857e+1
200	665	9.9e-5	-2.3713e+1	89	5.3e-5	-2.3713e+1
500	692	9.9e-5	-5.9283e+1	89	8.8e-5	-5.9283e+1
1000	712	9.9e-5	-1.1857e+2	90	7.9e-5	-1.1857e+2
1500	724	9.9e-5	-1.7785e+2	92	7.6e-5	-1.7785e+2
2000	732	9.9e-5	-2.3713e+2	96	3.2e-5	-2.3713e+2

TABLE 7
A comparison of the Method of $m = 3$ in solving QF3

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	300	9.8e-5	-6.5573e-1	44	8.2e-5	-6.5573e-1
20	310	1.0e-4	-1.3115	58	7.0e-5	-1.3115
40	321	9.8e-5	-2.6229	58	1.0e-4	-2.6229
80	332	9.7e-5	-5.2459	60	4.6e-5	-5.2459
100	335	9.8e-5	-6.5573	60	5.2e-5	-6.5573
200	346	9.7e-5	-1.3115e+1	60	7.3e-5	-1.3115e+1
500	359	1.0e-4	-3.2787e+1	62	9.0e-5	-3.2787e+1
1000	370	9.9e-5	-6.5573e+1	63	6.1e-5	-6.5573e+1
1500	376	9.9e-5	-9.8360e+1	63	7.5e-5	-9.8360e+1
2000	381	9.7e-5	-1.3115e+2	63	8.7e-5	-1.3115e+2

TABLE 8
A comparison of the Method of $m = 3$ in solving QF4

	SQN(0)			SQN(D1)		
N	Ite	$\|g_k\|$	Fva	Ite	$\|g_k\|$	Fva
10	230	9.8e-5	-1.6652	62	7.9e-5	-1.6652
20	238	9.8e-5	-3.3305	63	5.4e-5	-3.3305
40	246	9.9e-5	-6.6609	63	7.7e-5	-6.6609
80	254	9.9e-5	-1.3322e+1	64	8.3e-5	-1.3322e+1
100	257	9.8e-5	-1.6652e+1	64	9.3e-5	-1.6652e+1
200	265	9.8e-5	-3.3305e+1	68	3.5e-5	-3.3305e+1
500	276	9.7e-5	-8.3262e+1	68	9.5e-5	-8.3262e+1
1000	284	9.8e-5	-1.6652e+1	73	8.2e-5	-1.6652e+1
1500	289	9.7e-5	-2.4979e+1	73	9.6e-5	-2.4979e+1
2000	292	9.8e-5	-3.3305e+2	74	8.8e-5	-3.3305e+2

The number of iterations is the success index in a computational method. In this study, the number of iterations was compared between the standard SQN method and the proposed SQN method.

Tables 1-4 show the comparison results between the proposed preconditioned SQN methods and the standard SQN method for $m = 2$. Generally, the computational results show that the proposed methods performed better when compared to that of the standard SQN method. As shown in the Tables, the proposed methods require less number of iterations than the standard method. Although all the methods show the same values of function evaluation, the norms of the gradient for the proposed methods are less than the norms of the gradient of the standard method. Once again, this shows that the proposed SQN methods are promising alternatives as compared to the standard SQN method.

Tables 5-8 show the comparison results between the proposed preconditioned SQN methods and the standard SQN method for $m = 3$. Once again, the results reveal that the proposed methods clearly outperform the standard method. The number of iterations and the norms of the gradient are the best evidences to show that the proposed methods generally perform better than the standard SQN method.

CONCLUSION

The preconditioner for SQN method that is based upon variational technique and weak secant relation is proposed in this paper. The numerical results obtained suggest that the preconditioned SQN method is a good alternative for large-scale unconstrained optimization. Moreover, the preconditioned SQN method is preferred for reasons including simple implementation and it requires only function and gradient values.

ACKNOWLEDGEMENTS

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

REFERENCES

Farid, M., Hassan, M. A., & 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., & Hassan, M. A. (2010a). 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., & Hassan, M. A. (2010b). An improved multi-step gradient-type method for large scale optimization. *Computers and Mathematics with Applications, 61*, 3312-3318.

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

Leong, W. J., Farid, M., & Hassan, M. A. (2010b). Improved Hessian approximation with modified quasi-Cauchy relation for gradient-type method. *Advanced Modeling and Optimization, 12*, 37-44.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS Method for large scale optimization. *Mathematical Programming, 45*, 503-528.

Ni, Q., & Yuan, Y. (1997). A subspace limited memory quasi-Newton algorithm for large scale nonlinear bound constrained optimization *Mathematics of Computation, 66*, 1509-1520.

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

Wang, Z. H. Wen, Z. W., & Yuan, Y. (2004). A subspace trust region method for large scale unconstrained ooptimization. *Numerical Linear Algebra and Optimization,* 265-274.

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

Yuan, Y. (2007). Subspace techniques for nonlinear optimization. *Some Topics in Industrial and Applied Mathematics (Series in Contemporary Applied Mathematics CAM), 8*, 206-218.