A Self-scaling Quasi-Newton Method for Large Scale Unconstrained Optimization

Xiaowei Jiang*, Yueting Yang

College of Mathematics, Beihua University, Jilin 132013, China

Abstract

We present a new expression for self-scaling symmetric rank one update, and advance a limited memory method (L-SHSR1) for large scale unconstrained optimization. Because of combining with the scaling strategy, it does not only improve the numerical implementation of the SR1 algorithm by adjusting the eigenvalue distribution of updating matrices, but also make updates remain the positive definite property. Numerical experiments show that the new method is efficient for large scale problems.

Keywords: Large Scale Problem; Limited Memory; Quasi-Newton; L-SHSR1; Self-scaling

1 Introduction

We consider the unconstrained optimization problem

\[ \min \ f(x), \]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a smooth nonlinear function, the number of variables \( n \) is large. Liu and Nocedal [1] generalized several most useful methods for solving the above problem, in which limited memory quasi-Newton methods are interesting. The methods are suitable for large scale problems because the amount of storage required by the algorithms can be controlled by the user. Most of studies on limited memory quasi-Newton methods are concentrate on the limited memory BFGS (L-BFGS) method [1, 2, 3, 4]. However, the limited memory quasi-Newton methods related to SR1 update are less fewer. The major causes are that the SR1 methods are numerical instable due to the facts that the denominators of update matrices may be zero or nearly zero, and updates may not preserve the positive definiteness. While some recent researches show that SR1 updates with some improvement techniques are more efficient in comparison with other updates such as the BFGS and DFP updates etc. [5]. Variants are proposed to overcome the setback of the SR1 update [6, 7, 8, 9, 10, 11]. Among them, the SR1 method, that satisfies the usual quasi-Newton

*Project supported by “11th Five-Year Plan” to scientific research of the Department of Education of Jilin Province (No. 2008129).
*Corresponding author.
Email address: jxwbh@126.com (Xiaowei Jiang).
equation, combining with the scaling strategy is very attractive [8]. It does not only improve the numerical implementation of the SR1 algorithm by adjusting the eigenvalue distribution of updating matrices, but also make updates remain the positive definite property.

Yang, Xu and Gao [12] made a little modification for self-scaling symmetric rank one update with Davidon’s optimal condition [11] (SHSR1) as follow

\[ H_{k+1} = \mu_k H_k + \frac{(s_k - \mu_k H_k \hat{y}_k)(s_k - \mu_k H_k \hat{y}_k)^T}{(s_k - \mu_k H_k \hat{y}_k)^T \hat{y}_k}, \]

where \( \mu_k \) is the scaling factor, \( \hat{y}_k = 1 + \theta_k/s_k^T y_k \), \( \theta_k = 6(f_k - f_{k+1}) + 3(g_k + g_{k+1})^T s_k \) and \( f_k = f(x_k) \). We will present a new expression for symmetric rank-one formula (2), and advance a new limited memory algorithm.

2 The SHSR1 Update

Yang, Xu and Gao [12] conducted a detailed discussion of a scaling strategy, in which the scaling factor \( \mu_k > 0 \) can be chosen either as \( \mu_k = \frac{\alpha_k}{b_k} - \sqrt{(\frac{\alpha_k}{b_k})^2 - \frac{\alpha_k}{a_k}} \) or \( \mu_k = \frac{\alpha_k}{b_k} + \sqrt{(\frac{\alpha_k}{b_k})^2 - \frac{\alpha_k}{a_k}} \), where \( a_k = y_k^T H_k \hat{y}_k, b_k = s_k^T \hat{y}_k \) and \( c_k = s_k^T H_k^{-1} s_k \).

Denote \( \delta_k = (s_k - H_k \mu_k \hat{y}_k)^T \hat{y}_k, Z_k = (S_k - H_0 Y_k)^T \mu_k \hat{y}_k \), where \( S_k = [s_0, \cdots, s_{k-1}] \), \( Y_k = [\eta_0 \hat{y}_0, \cdots, \eta_{k-1} \hat{y}_{k-1}] \), \( \eta_k = \prod_{j=0}^{k-1} \eta_j \). Firstly we prove a preliminary lemma that will be useful in subsequent theorem.

Lemma 1 Let

\[
N_{k+1} = \frac{1}{\delta_k} \begin{bmatrix}
\mu_k \delta_k N_k + N_k Z_k Z_k^T N_k^T & -N_k Z_k \\
-Z_k^T N_k^T & 1
\end{bmatrix} (k \geq 1),
\]

and \( N_1 = \frac{1}{(s_0 - \mu_0 H_0 y_0)^T y_0} \) is given, then we have

\[
N_{k+1}^{-1} = \eta_k \text{sym}((S_{k+1} - H_0 Y_{k+1})^T Y_{k+1}),
\]

where for any matrix \( A \), the symmetrization operation is defined as

\[
sym(A) = \begin{cases}
A_{ij}, & i \leq j \\
A_{ji}, & \text{otherwise}.
\end{cases}
\]

Proof We first prove that

\[
N_{k+1}^{-1} = \text{sym}((S_{k+1} - H_0 Y_{k+1})^T Y_{k+1} D_{k+1} P_{k+1}),
\]

where \( D_{k+1} = \text{diag}(\eta_0^{-1}, \eta_1^{-1}, \cdots, \eta_{k-1}^{-1}) \), \( P_{k+1} = \prod_{j=1}^{k} \begin{bmatrix}
\mu_j^{-1} I_j & 0 \\
0 & I_{k+1-j}
\end{bmatrix} \), \( I_j \) is the identity matrix with order \( j \) and \( P_1 = 1 \) is given.
Proceeding by induction. It is easy to know that (5) holds for \( k = 0 \). We assume that it holds for some \( k \), then consider \( k + 1 \). By direct multiplication, we can verify

\[
N_{k+1}^{-1} = \begin{bmatrix}
\mu_k^{-1}N_k^{-1} & \mu_k^{-1}Z_k \\
\mu_k^{-1}Z_k^T & (s_k - \eta_kH_0\hat{y}_k)^T \hat{y}_k
\end{bmatrix}.
\]

Note that

\[
(S_{k+1} - H_0Y_{k+1})^TY_{k+1}D_{k+1}P_{k+1}
\]

\[
= (S_k - H_0Y_k \ s_k - H_0\eta_k\hat{y}_k)(Y_k \ \eta_k\hat{y}_k) \begin{bmatrix}
D_k \\
\eta_k^{-1}
\end{bmatrix} \begin{bmatrix}
P_k \\
1
\end{bmatrix} \begin{bmatrix}
\mu_k^{-1}I_k \\
1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\mu_k^{-1}(S_k - H_0Y_k)^TY_kD_kP_k & (S_k - H_0Y_k)^T\hat{y}_k \\
\mu_k^{-1}(s_k - H_0\eta_k\hat{y}_k)^TY_kD_kP_k & (s_k - \eta_kH_0\hat{y}_k)^T\hat{y}_k
\end{bmatrix},
\]

so

\[
sym((S_{k+1} - H_0Y_{k+1})^TY_{k+1}D_{k+1}P_{k+1}) = N_{k+1}^{-1}.
\]

It is easy to derive that \( D_{k+1}P_{k+1} = \eta_kI_{k+1} \) from the definitions of \( D_{k+1} \) and \( P_{k+1} \), so (3) is true.

**Theorem 1** Let \( H_0 \) be symmetric and positive definite and assume that \( b_k > 0, b_k - \mu_k\alpha_k > 0 \), then (2) can be written as

\[
H_{k+1} = \eta_kH_0 + (S_{k+1} - H_0Y_{k+1})N_{k+1}(S_{k+1} - H_0Y_{k+1})^T.
\]

**Proof** Proceeding by induction. It is true for \( k = 0 \). In fact

\[
H_1 = \mu_0H_0 + \frac{(s_0 - \mu_0H_0\hat{y}_0)(s_0 - \mu_0H_0\hat{y}_0)^T}{(s_0 - \mu_0H_0\hat{y}_0)^T\hat{y}_0} = \eta_0H_0 + (S_1 - H_0Y_1)N_1(S_1 - H_0Y_1)^T.
\]

Now we assume that (6) holds for some \( k - 1 \), and consider \( k \). First note that

\[
s_k - \mu_kH_k\hat{y}_k = s_k - \mu_k(\eta_{k-1}H_0 + (S_k - H_0Y_k)N_k(S_k - H_0Y_k)^T)\hat{y}_k
\]

\[
= [S_k - H_0Y_k \ s_k - \eta_kH_0\hat{y}_k] \begin{bmatrix}
-\mu_kN_k(S_k - H_0Y_k)^T\hat{y}_k \\
1
\end{bmatrix}
\]

\[
= (S_{k+1} - H_0Y_{k+1}) \begin{bmatrix}
-N_kZ_k \\
1
\end{bmatrix},
\]

and \( \delta_k \) can be written as

\[
\delta_k = \begin{bmatrix}
-Z_k^TN_k^T & 1
\end{bmatrix} (S_{k+1} - H_0Y_{k+1})^T\hat{y}_k = \frac{-Z_k^TN_k^TZ_k}{\mu_k} + (s_k - H_0\eta_k\hat{y}_k)^T\hat{y}_k.
\]

So

\[
H_{k+1} = \mu_kH_k + \frac{(s_k - H_k\mu_k\hat{y}_k)(s_k - H_k\mu_k\hat{y}_k)^T}{(s_k - H_k\mu_k\hat{y}_k)^T\hat{y}_k} = \eta_kH_0 + (S_{k+1} - H_0Y_{k+1})N_{k+1}(S_{k+1} - H_0Y_{k+1})^T.
\]
3 L-SHSR1 Algorithm

Now we describe a limited memory performance based on the SHSR1 update. Let \( m(\geq 0) \) be a memory index number, and assume that a positive definite symmetric matrix \( H_0 \) is given. The calculation of the search direction \( d_{k+1} = -H_{k+1}g_{k+1} \) is a key issue that must be addressed. For incorporating more up-to-date information, we replace the matrix \( H_0 \) in (6) by \( H^{(0)}_{k+1} \), where

\[
H^{(0)}_{k+1} = \gamma_k I + \eta_k^{-1}(S_{k+1} - \gamma_k Y_{k+1})sym((S_{k+1} - \gamma_k Y_{k+1})^T Y_{k+1})^{-1}(S_{k+1} - \gamma_k Y_{k+1})^T. \tag{7}
\]

During the first \( m \) iterative steps, we set \( S_{k+1} = [s_0, \cdots, s_k] \) and \( Y_{k+1} = [\eta_0 \tilde{y}_0, \cdots, \eta_k \tilde{y}_k] \). But for \( k \geq m \), we only use the latest \( m \) steps information, which are \( S_{k+1} = [s_{k-m+1}, \cdots, s_k] \) and \( Y_{k+1} = [\eta_{k-m+1} \tilde{y}_{k-m+1}, \cdots, \eta_k \tilde{y}_k] \).

**Algorithm 3.1 (L-SHSR1 method)**

Step 1 Choose \( x_0 \in \mathbb{R}^n, m, 0 < \tau_1 < 1/2, \tau_1 < \tau_2 < 1, \epsilon \) and a symmetric and positive definite starting matrix \( H_0 \). Set \( d_0 = -H_0g_0, k = 0 \).

Step 2 Select a step factor \( \alpha_k \) such that \( f(x_k + \alpha_k d_k) \leq f(x_k) + \tau_1 \alpha_k g_k^T d_k \) and \( g(x_k + \alpha_k d_k)^T d_k \geq \tau_2 g_k^T d_k \). (We always try the step length \( \alpha_k = 1 \) first).

Step 3 Set \( x_{k+1} = x_k + \alpha_k d_k \).

Step 4 If \( \|g_{k+1}\| < \epsilon \) or \( f_k - f_{k+1} \leq \epsilon \max\{1.0, |f_k|\} \), then stop.

Step 5 Update \( S_k \) and \( Y_k \) for \( S_{k+1} \) and \( Y_{k+1} \), and calculate \( \gamma_k \) and \( \eta_k \).

Step 6 Compute a search direction \( d_{k+1} = -H_{k+1}g_{k+1} \). Set \( k = k + 1 \) and goto Step 2.

At Step 6, we must update \( H_k \) to get \( H_{k+1} \), and compute the search direction \( -H_{k+1}g_{k+1} \). Note that the matric \( H_k \) are not formed explicitly, but \( S_k \) and \( Y_k \) are stored, respectively. We can update each of them by adding a new column on the right for \( k \leq m - 1 \), otherwise, deleting the first column and adding a new column on the right. In detail, we can compute \( d_{k+1} \) by the following algorithm.

**Algorithm 3.2**

\[
d_{k+1} = -H_{k+1}g_{k+1}
\]

Step 1 Set \( SY = S_{k+1} - \gamma_k Y_{k+1} \).

Step 2 Set \( d = SY^T \cdot g_{k+1} \).

Step 3 Calculate \( SYM = \eta_k^{-1} \cdot sym(SY^T Y_{k+1})^{-1} \).

Step 4 Set \( d = SY \cdot SYM \cdot d \).

Step 5 Calculate \( d_{k+1} = -\gamma_k \eta_k g_{k+1} - d \).
There are three strong points to support L-SHSR1 algorithm. Firstly, L-SHSR1 update is based on the modified quasi-Newton equation $H_{k+1}y_k = s_k$, so it exploits more information than the conventional one. Secondly, the parameter $\eta_k = \prod_{j=0}^{k} \mu_j$ preserves most of the information that is from the beginning until now, which makes the limited memory SR1 method use more useful information so as to promote the efficiency of the algorithm in the condition of deleting the first $k - m$ steps. Thirdly, the use of the self-scaling strategy overcomes the difficulty in calculating the denominator of SR1 update.

4 Numerical Results

We report some numerical results of the L-SHSR1 in this section. The test problems [14, 15] are listed in Table 1. All tests are done on a PC with 1.8 GHz Pentium IV and 512 MB SDRAM using MATLAB6.5. The parameter values are $\tau_1 = 0.01, \tau_2 = 0.9$, and $\epsilon = 10^{-8}$. The results of our numerical experiments are summarized in Table 2, where the algorithms L-SHSR1 and L-SR1 (no scaling) are implemented, respectively. Both of them use the same line search procedure (Wolfe conditions). The Table 2 reports tests with $m = 3$. The basic data reported for each method are the dimension of the objective function ($\text{dim}$), the arithmetic time (CPU), the number of iteration ($\text{NI}$), the number of calls to the function evaluation routine ($\text{Nf}$), the number of update for iterative matrix ($\text{Sr}$) and the final function value ($f^*$).

<table>
<thead>
<tr>
<th>No.</th>
<th>Problem name</th>
<th>No.</th>
<th>Problem name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Variable dimension function</td>
<td>2</td>
<td>Broyden tridiagonal function</td>
</tr>
<tr>
<td>3</td>
<td>Boundary value function</td>
<td>4</td>
<td>Chebyquad function</td>
</tr>
<tr>
<td>5</td>
<td>Schittkowski function</td>
<td>6</td>
<td>the trigonometric function</td>
</tr>
<tr>
<td>7</td>
<td>Gould</td>
<td>8</td>
<td>Yang tridiagonal function</td>
</tr>
<tr>
<td>9</td>
<td>Separable cubic function</td>
<td>10</td>
<td>Conn</td>
</tr>
<tr>
<td>11</td>
<td>Nearly separable function</td>
<td>12</td>
<td>Freudenstein and Roth function</td>
</tr>
<tr>
<td>13</td>
<td>Extended Rosenbrock function</td>
<td>14</td>
<td>Extended Powell singular function</td>
</tr>
</tbody>
</table>

The numerical results listed in Table 2 show the superiority of the algorithm L-SHSR1 over the L-SR1 algorithm. For most of test problems, we can obtain more accurate results for objective function with less time and the number of iteration.

5 Conclusions

This paper propose a new self-scaling SR1 algorithm for large scale optimization. We exploit more information in update matrices than the conventional ones. In the new method, the positive definiteness of the update can be maintained and the difficulty in calculating the denominator of MSR1 disappears automatically.
### Table 2: Numerical results

<table>
<thead>
<tr>
<th>No.</th>
<th>dim</th>
<th>CPU/NI/Nf/Sr/f* (L-SR1)</th>
<th>CPU/NI/Nf/Sr/f* (L-SHSR1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0.01/10/28/10/1.6728 × 10^{-15}</td>
<td>0.01/12/59/11/7.5913 × 10^{-16}</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.06/21/46/20/9.0649 × 10^{-11}</td>
<td>0.03/11/65/10/9.0651 × 10^{-11}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.05/49/116/35/9.2984 × 10^{-12}</td>
<td>0.05/40/68/23/5.0252 × 10^{-17}</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0.02/26/126/26/2.4295 × 10^{-5}</td>
<td>0.03/36/105/35/3.0216 × 10^{-9}</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.02/33/98/29/8.5354 × 10^{-8}</td>
<td>0.03/35/85/30/2.1325 × 10^{-11}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.07/78/210/78/1.0315 × 10^{-5}</td>
<td>0.06/66/201/66/1.3652 × 10^{-7}</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0.02/5/30/5/6.3300 × 10^{-5}</td>
<td>0.01/6/40/5/6.0123 × 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.22/10/69/9/1.3721 × 10^{-5}</td>
<td>0.13/11/29/7/1.3718 × 10^{-7}</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0.01/8/13/7/1.4244 × 10^{-11}</td>
<td>0.01/8/11/8/2.9274 × 10^{-12}</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.01/9/37/8/1.0086 × 10^{-16}</td>
<td>0.02/10/39/9/1.0132 × 10^{-16}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.44/37/95/24/5.0615 × 10^{-18}</td>
<td>0.38/35/89/20/9.0651 × 10^{-19}</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0.02/6/47/2/4.0328 × 10^{-5}</td>
<td>0.01/5/40/4/2.0316 × 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.02/11/21/10/8.2082 × 10^{-4}</td>
<td>0.02/10/20/10/8.2082 × 10^{-4}</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0.01/6/28/2/8.2333</td>
<td>0.01/3/20/2/8.8057</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.02/7/47/7/98.9995</td>
<td>0.02/9/16/10/98.3536</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0.17/225/489/224/6.9167</td>
<td>0.02/65/108/45/6.9167</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>0.02/15/24/14/1.4373 × 10^{-8}</td>
<td>0.01/6/11/6/4.5855 × 10^{-11}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.55/11/21/10/4.8947 × 10^{-10}</td>
<td>0.44/7/11/6/3.1026 × 10^{-12}</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>54.91/12/23/11/2.0991 × 10^{-10}</td>
<td>39.75/7/11/7/1.4471 × 10^{-11}</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>0.02/8/10/8/4.5726 × 10^{-6}</td>
<td>0.02/7/8/6/1.2970 × 10^{-36}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.06/7/9/7/9.3060 × 10^{-4}</td>
<td>0.03/10/15/9/2.9619 × 10^{-23}</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>0.28/7/9/7/9.0089</td>
<td>0.23/21/32/18/3.9334 × 10^{-45}</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0.02/13/21/12/3.6681 × 10^{-5}</td>
<td>0.01/10/15/10/4.1532 × 10^{-6}</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.82/85/108/79/4.2528 × 10^{-5}</td>
<td>0.47/72/93/66/5.9945 × 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>24.10/371/404/328/3.5142 × 10^{-4}</td>
<td>20.14/278/314/210/2.4578 × 10^{-4}</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0.02/19/21/18/48.9412</td>
<td>0.01/18/31/15/48.9843</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0.05/45/65/43/1.7686 × 10^{-8}</td>
<td>0.04/40/52/36/8.7565 × 10^{-9}</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.39/46/48/45/3.5124 × 10^{-8}</td>
<td>0.31/39/69/30/1.3353 × 10^{-8}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>37.56/154/171/142/8.6425 × 10^{-7}</td>
<td>35.94/125/144/96/4.4475 × 10^{-7}</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>0.34/24/43/20/9.2000 × 10^{-5}</td>
<td>0.31/21/39/16/2.0183 × 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>6.72/37/60/31/1.8824 × 10^{-4}</td>
<td>6.41/35/54/27/1.1127 × 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>28.44/92/163/86/4.0376 × 10^{-4}</td>
<td>26.72/84/123/56/2.2254 × 10^{-5}</td>
</tr>
</tbody>
</table>
References


[14] Ph. L. Toint, Test problems for partially separable optimization and results for the routine PSP-MIN, Report Nn8314, Department of Mathematics, Faculte’s Universitaries de Namur (Namur, Belgium, 1983a)