Accelerated Scaled Memory-less SR1 method for Unconstrained Optimization

Neculai Andrei
Center for Advanced Modeling and Optimization,
Academy of Romanian Scientists,
54, Splaiul Independenţei, Sector 5,
Bucharest, ROMANIA
E-mail: neculaiandrei70@gmail.com

Technical Report 5/2021
April 9, 2021

Abstract. The scaled memory-less SR1 updating method is obtained from the SR1 quasi-Newton method in which the current approximation to the Hessian is replaced by the identity matrix and the update term is scaled by a parameter. The value of the scaling parameter is computed by using the sufficient descent condition or by using the conjugacy condition. Numerical experiments with a set of 800 unconstrained optimization problems proved that the accelerated scaled memory-less SR1 method is more efficient and more robust than the preconditioned BFGS algorithm implemented in CONMIN.

Key words: Nonlinear programming, symmetric rank-one update, scaled SR1 memory-less update, sufficient descent, conjugacy condition, uniform linear independency

1. Introduction

For solving the unconstrained optimization problem

$$\min f(x), \tag{1}$$

where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, bounded from below we are interested in this paper to use the symmetric rank-one SR1 method. There is a prejudice that SR1 method is full of drawbacks and is not to be selected for solving (1). In this paper we present a variant of this method which proves to be more efficient and more robust than the well known and most appreciated BFGS method.

Many algorithms for solving (1) employ a quadratic model of the function f. The Newton's method and the quasi-Newton methods use a second-order Taylor series of the minimizing function with either an explicit or an approximated Hessian matrix. The quasi-Newton methods use an approximation to the Hessian, or an approximation to the inverse Hessian, which are updated at each iteration. These algorithms are efficient and robust for minimizing functions that satisfy certain assumptions and have a super-linear rate of local convergence. Currently, many variants of the updating formula for the approximation to the Hessian (or to the inverse Hessian) are known: symmetric rank-one

(SR1) [Broyden, 1967; Davidon, 1959, 1968; Fiacco and McCormick, 1968; Wolfe, 1969, 1971] and the rank-two such as the Davidon-Fletcher-Powell (DFP) update [Davidon, 1991; Fletcher and Powell, 1963], Powell-symmetric-Broyden (PSB) [Powell, 1970] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970]. A unifying framework for many of these updates, including both rank-one and rank-two updates was given by Huang [1970].
Theoretical studies and intensive numerical experiments proved that among the quasi-Newton methods the BFGS is the most effective. However on certain problems BFGS method may require a large number of iterations and function and gradient evaluations. The sourced of its inefficiency may be caused by a poor initial approximation to the Hessian, or more importantly, by the ill-conditioning of the Hessian approximations along the iterations. To improve the efficiency and the robustness of the BFGS and to overcome the difficulties, some modified versions of it were given. All these modified BFGS methods can be classified into four classes: the scaling of the BFGS update matrix, the memory-less BFGS method, the BFGS update with modified secant equation and the modified BFGS method using different line search conditions for stepsize computation. Intensive numerical experiments on minimizing functions with different number of variables and complexities showed that all the modified BFGS methods are more efficient and more robust than its unmodified version [Contreras and Tapia, 1993; Oren and Luenberger, 1974; Oren and Spedicato, 1976; Yabe, Martinez and Tapia, 2004; Biggs, 1971, 1973; Liao, 1997; Nocedal and Yuan, 1993; Andrei, 2018a, 2018b, 2018c, 2020a; Yuan, 1991; Yuan and Byrd, 1995; Al-Baali, 1998; Arzam, Babaie-Kafaki and Ghanbari, 2017; Yuan, Sheng, Wang, Hu and Li, 2018; Dehmiry, 2019].

Having in view all these developments on the BFGS method it is quite natural to try to extend them to the SR1 method. In this paper we present only the scaled memory-less SR1 method which combines both the scaling and the memory-less techniques. Other modifications involving modified secant equation or different line search for stepsize computation are postponed for future time. By memory-less techniques the approximation to the Hessian at a given iteration is computed by updating the unity matrix. By scaling we mean that the updating term of the SR1 formula is scaled with a parameter selected in such a way to compensate the lost of information by considering the unity matrix in updating formula.
The structure of the paper is as follows. Section 2 presents the quasi-Newton SR1 method insisting on the updating formula and its weakness. Notice that the SR1 updating formula is very simple, easy to be implemented in computing programs, but it is corrupted by some drawbacks which dramatically limit its applicability. Section 3 is dedicated to the scaled memory-less quasi-Newton symmetric rank-one SR1 update. As we said, this method consists of updating at every iteration the unity matrix and scaling the updating term with parameter. The scaling parameter is determined in two different ways. In the first one it is computed by involving the sufficient descent condition. The second one is based on the conjugacy condition from the conjugate gradient methods. The accelerated scaled memory-less SR1 algorithm is showed in Section 4. The convergence of the scaled memory-less SR1 algorithm is presented in Section 5. The last section is dedicated to the numerical results obtained with an implementation of the suggested algorithm. It is shown that the accelerated scaled memory-less SR1 algorithm is more efficient and more

robust than the BFGS algorithm implemented in CONMIN developed by Shanno and Phua (1976) on solving a se of 800 unconstrained optimization test problems with the number of variables in the range [100, 1000]. Finally the performances of the algorithm are presented for solving five applications from MINPACK2 collection, each of then having 40.000 variables.

2. The quasi-Newton SR1 (Symmetric Rank One) Update

Mainly, the structure of a quasi-Newton method with Wolfe line search, in the variant in which an approximation to the Hessian is used, can be presented as follows:

General quasi-Newton Algorithm

1) Set $k = 0$. Consider an initial point x_0 and an initial Hessian approximation B_0, Select some values for the Wolfe line search conditions σ and ρ with $0 < \rho < \sigma < 1$. Select a sufficiently small parameter $\varepsilon > 0$ used in the criterion for stopping the iterations.
2) **While** $\|\nabla f(x_k)\| > \varepsilon$ do
   a. Evaluate the gradient $g_k = \nabla f(x_k)$.
   b. Solve the system $B_k d_k = -g_k$.
   c. Find the stepsize $\alpha_k > 0$ which satisfies the standard Wolfe line search conditions:

$$f(x_k + \alpha_k d_k) < f(x_k) + \rho \alpha_k g_k^T d_k, \tag{2.a}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k > \sigma \nabla f(x_k)^T d_k. \tag{2.b}$$

   d. Compute: $x_{k+1} = x_k + \alpha_k d_k$ and set $k = k + 1$.
   e. Update B_{k+1}, according the quasi-Newton formula in use.
3) **End while**.

Another variant of the above general quasi-Newton algorithm, often implemented for solving the problem (1), at every iteration uses an approximation to the inverse Hessian H_k. In this variant in step 1) of the algorithm instead of B_0 an initial approximation to the inverse Hessian H_0 is selected and in step 2.b) instead of solving the linear algebraic system $B_k d_k = -g_k$, the search direction is computed as $d_k = -H_k g_k$. In step 2.e) instead of updating the approximation to the Hessian B_{k+1}, a formula for approximation to the inverse Hessian H_{k+1} is used. The quasi-Newton algorithms differentiate in step 2.e) concerning the updating the approximation to the Hessian B_{k+1}, or to the inverse Hessian H_{k+1}.

In the quasi-Newton methods the basic requirement for the updating formula to the Hessian is the so called the secant equation to be satisfied at each iteration, namely

$$B_{k+1} s_k = y_k \text{ or } H_{k+1} y_k = s_k, \tag{3}$$

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

The symmetric rank-one SR1 update formula, in which we are interested in this paper, can be derived as solution of the following simple problem. "*Given a symmetric matrix* B_k *and the vectors* s_k *and* y_k*, finds a new symmetric matrix* B_{k+1} *such that* $B_{k+1} - B_k$ *has rank one and such that the secant equation* $B_{k+1}s_k = y_k$ *is satisfied*." It is easy to see that if $(y_k - B_k s_k)^T s_k \neq 0,$ then the unique solution of the above problem is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}. \tag{4}$$

If $y_k = B_k s_k$ then the solution is $B_{k+1} = B_k$. However, if $(y_k - B_k s_k)^T s_k = 0$ and $y_k \neq B_k s_k$, then there is no solution to the problem. Therefore, to get the SR1 method in step 2.e) of the general quasi-Newton algorithm the update B_{k+1} approximation to the Hessian is computed like in (4).

Let H_k be the inverse approximation to the Hessian at iteration k. By using the Sherman-Morrison-Woodbury formula in (4), the following update to the inverse Hessian for SR1 is

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}. \tag{5}$$

Therefore, in the general quasi-Newton algorithm above, instead of solving the linear algebraic system $B_k d_k = -g_k$ in step 2.b) we can simply set the search direction as $d_k = -H_k g_k$. This variant of the algorithm is only applicable in cases in which the inverse H_k exists.

Now a comparison between the BFGS and SR1 updates is welcomed. As we know the most effective quasi-Newton updating of the approximations to the Hessian is considered the BFGS formula [Nocedal, 1992], [Nocedal & Wright, 2006] where

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} - \frac{y_k y_k^T}{y_k^T s_k} \tag{6}$$

which is a rank-two update that satisfies the secant equation (3). If H_k is the inverse approximation to the Hessian at iteration $k,$ then by applying the Sherman-Morrison-Woodbury formula twice the following update to the inverse Hessian for BFGS is

$$H_{k+1} = H_k - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{y_k^T s_k} + \left(1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right)\frac{s_k s_k^T}{y_k^T s_k}. \tag{7}$$

The most important properties of BFGS are as follows. If H_k is positive definite, then also H_{k+1} given by (7) is positive definite for any $k,$ provided that $y_k^T s_k > 0$ (which always is satisfied when the Wolfe line search (2) are satisfied). Therefore, if H_0 is

chosen to be positive definite, then the rest of all the approximations H_k will also be positive definite. Also BFGS has the self-correcting property, i.e. if H_k incorrectly approximates the curvature of the minimizing function and this estimate slows down the iteration, then the inverse Hessian approximation will tend to correct itself in the next few iterations. The self-correcting property depends on the quality of the implementation of the Wolfe line search. For the Wolfe line search, always the initial value $\alpha = 1$ is tried and this produce superlinear convergence of the method. All these properties of BFGS update make this quasi-Newton method one of the best in this class. However, the things are not what they seem to be.

There are great differences between the SR1 update (4) and the BFGS update (6). Firstly, observe that SR1 is a rank-one update of the Hessian and BFGS is a rank-two update, both of them satisfying the secant equation. Secondly, there are some drawbacks of SR1 which are not encountered in BFGS. The main drawbacks of SR1 update are as follows.

1) The denominator $(y_k - B_k s_k)^T s_k$ of the SR1 update term in (4) may vanish, i.e. $(y_k - B_k s_k)^T s_k \cong 0$, cases in which B_{k+1} is not well-defined.

2) The step directions computed by using the SR1 updating formula given by (4) may no longer be uniform linear independent, thus leading to slow down the convergence or even the stalling.

3) The SR1 Hessian approximation may not be positive definite along the iterations, thus resulting a direction that does not produce descent.

To prevent the method from failing due to the first drawback one simple remedy is to set $B_{k+1} = B_k$. However, this will slow down the convergence of the method. Conn, Gould and Toint (1991) and Khalfan, Byrd and Schnabel (1993) showed that the denominator of (4) vanishes rarely in practice and setting $B_{k+1} = B_k$ does not have a significant impact on the performances of the SR1 method subject to the number of the iterations or runtimes.

The second drawback is more subtle, being in close connection with the uniform linear independence of the search directions generated by the SR1 algorithm. A more precise definition of the uniform linear independence was given by Conn, Gould and Toint (1991). "A sequence $\{s_k\}$ is uniformly linearly independent if there exist $\xi > 0$, k_0 and $m \geq n$ such that, for each $k \geq k_0$, there is n distinct indices $k \leq k_1 \leq k_2 \leq \ldots \leq k_n \leq k + m$ for which the minimum singular value of the matrix $S = \left[\frac{s_{k_1}}{\|s_{k_1}\|}, \cdots, \frac{s_{k_n}}{\|s_{k_n}\|} \right]$ is at least ξ." Conn, Gould and Toint (1991) proved that the sequence of matrices generated by the SR1 formula converges to the exact Hessian, when the sequence of iterates converges to a limit point and the sequence of steps is uniformly linearly independent. Kelley and Sachs [1998] provide similar convergence results removing the first of these assumptions. Fiacco and McCormick [1968] showed that if the search directions are linearly independent and the denominator of (4) is always non-zero, then the SR1 method without line searches minimize a strongly convex quadratic function in at most $n+1$ steps. In this case B_{n+1} is exactly the Hessian of the quadratic function. Observe that this result is

significant since it does not require exact line search, as is the case for the BFGS update. Generally, the above condition given by the definition of the uniform linear independency is not implemented in practice, it serves only as one of the main assumptions of a proof that the SR1 approximations to the Hessian converge to the true Hessian as the iterates converge to the solution of (1).
Subject to the uniform linear independency of the search directions Khalfan, Byrd and Schnabel [1993] showed that many problems do not satisfy this requirement, but they proved the local convergence of the SR1 method using only the positive definiteness and boundedness assumptions for the approximate Hessian. More than this Conn, Gould and Toint [1991] proved that if the minimizing function f is twice continuously differentiable and its Hessian is bounded and Lipschitz continuous and the iterates generated by the SR1 method converge to a point x^* and in addition for all k,

$$\left|(y_k - B_k s_k)^T s_k\right| \geq \xi \|y_k - B_k s_k\| \|s_k\|, \tag{8}$$

for some $\xi \in (0,1)$, and the steps s_k are uniformly linearly independent, then

$$\lim_{k\to\infty} \left\|B_k - \nabla^2 f(x^*)\right\| = 0.$$

Often condition (8) is used in implementations of the SR1 method in order to ensure that this update is well behaved. If this condition is not satisfied, then the update is skipped. Conn, Gould and Toint [1991] and Khalfan, Byrd and Schnabel [1993] provide theoretical and computational results, respectively, that if the uniform linear independence assumption is satisfied, then the approximations to the Hessian generated by the SR1 method are more accurate than those generated by BFGS, and SR1 converge faster to the true Hessian than BFGS. Therefore, if all these above drawbacks are addressed in a reliable and efficient manner, then SR1 can be used for solving (1) instead of the rank-two updates. More details on SR1 method concerning the undefined updates, choosing the initial approximate B_0, uniform linear independence of the steps, are found in [Benson and Shanno, 2018] and in [Chen, Lam and Chan, 2019].

3. The scaled memory-less quasi-Newton SR1 Update

Like for scaled memory-less quasi-Newton BFGS update developed by Andrei [2007, 2018a, 2018b, 2018c, 2020b], in this section let us introduce the *scaled memory-less SR1 update* by considering in (5) $H_k = I$, i.e.

$$H_{k+1} = I + t_k \frac{(s_k - y_k)(s_k - y_k)^T}{(s_k - y_k)^T y_k}, \tag{9}$$

where t_k is the scaling parameter. After some simple algebraic manipulations the scaled memory-less SR1 search direction $d_{k+1} = -H_{k+1} g_{k+1}$ is obtained as

$$d_{k+1} = -g_{k+1} - t_k \frac{(s_k - y_k)^T g_{k+1}}{(s_k - y_k)^T y_k}(s_k - y_k). \tag{10}$$

The main advantage of the scaled memory-less SR1 update (9) is that for its implementation in computer programs only two scalar products $(s_k - y_k)^T g_{k+1}$ and $(s_k - y_k)^T y_k$ must be computed. This is very advantageous for solving large-scale problems. Observe that in the scaled memory-less SR1 update the information on the Hessian approximation from the previous iteration is not accumulated to the current iteration. The scaling parameter t_k in (9) is introduced to compensate this loss of information.

Now, it can be noticed that the scaled memory-less SR1 search direction (10) has three terms. The first one is the negative gradient $-g_{k+1}$, the last two terms involve s_k and y_k, both of them being multiplied by the same scalar. To determine a value for the scaling parameter t_k two procedures are developed in this paper. The first one is based on the sufficient descent condition the second considers the conjugacy condition from the conjugate gradient algorithms.

3.1. Determine t_k from the sufficient descent condition

In the convergence analysis, a key requirement for a line search algorithm is that the search direction d_k is a direction of sufficient descent, which is defined as

$$\frac{g_k^T d_k}{\|g_k\|\|d_k\|} \le -\varepsilon, \tag{11}$$

where $\varepsilon > 0$. This condition bounds the elements of the sequence $\{d_k\}$ of the search directions from being arbitrarily close to the orthogonality to the gradient. Often, the line search methods are so that d_k is defined in a way that satisfies the sufficient descent condition (11), even though an explicit value for $\varepsilon > 0$ is not known.

As we said, a significant difference between SR1 and BFGS updates is that BFGS guarantees to produce a positive definite B_{k+1} if B_k is positive definite and $y_k^T s_k > 0$, while SR1 does not have this important property. Therefore for BFGS the search direction $d_{k+1} = -B_{k+1}^{-1} g_{k+1}$ always is a descent direction. On the other hand, in order to get an efficient scaled memory-less SR1 algorithm we have to impose the sufficient descent condition. Therefore, by imposing the sufficient descent condition

$$g_{k+1}^T d_{k+1} = -c\|g_{k+1}\|^2, \tag{12}$$

where d_{k+1} is given by (10) the following value for the scaling parameter is obtained

$$t_k = \frac{(s_k - y_k)^T y_k}{[(s_k - y_k)^T g_{k+1}]^2}(c-1)\|g_{k+1}\|^2. \tag{13}$$

Observe that in (12) the classical sufficient descent condition (11) is modified with equality. In (13) c is selected close to 1, but not too close. In our numerical experiments we selected $c = 7/8$. Therefore, introducing the value for t_k given by (13) in (10) the following scaled memory-less SR1 search direction is obtained

$$d_{k+1} = -g_{k+1} - \frac{(c-1)\|g_{k+1}\|^2}{(s_k - y_k)^T g_{k+1}}(s_k - y_k). \tag{14}$$

Observe that the iterations are affected if the denominator $(s_k - y_k)^T g_{k+1}$ of the update term becomes too small. In this case the update term in (14) may start to dominate the negative gradient and therefore the influence of the negative gradient in the search direction is lost. In order to accommodate these situations, the rule we apply is that the updates are skipped whenever the denominator $(s_k - y_k)^T g_{k+1}$ is too small in the sense

$$\left|(s_k - y_k)^T g_{k+1}\right| < \eta \|s_k - y_k\| \|g_{k+1}\|, \tag{15}$$

that is, if (15) is satisfied, then $d_{k+1} = -g_{k+1}$. A typical value for η is 10^{-8}.

3.2. Determine t_k from the conjugacy condition

This procedure for determination of the scaling parameter t_k is motivated by the fact that the scaled memory-less SR1 search direction (10) resembles a three-term conjugate gradient algorithm. As we know, for the quadratic functions, it is well known that the linear conjugate gradient methods generate a sequence of search direction d_k, $k = 1, 2, \ldots$ so that the following conjugacy condition holds: $d_i^T B d_j = 0$ for all $i \neq j$, where B is the Hessian of the objective function. For general nonlinear functions, by the mean value theorem there exists $\xi \in (0,1)$ so that

$$d_{k+1}^T g_{k+1} = d_{k+1}^T g_k + \alpha_k d_{k+1}^T \nabla^2 f(x_k + \xi \alpha_k d_k) d_k. \tag{16}$$

Since $y_k = g_{k+1} - g_k$, the following can be written

$$d_{k+1}^T y_k = \alpha_k d_{k+1}^T \nabla^2 f(x_k + \xi \alpha_k d_k) d_k. \tag{17}$$

Therefore, for nonlinear optimization, it is reasonable to replace the conjugacy condition from the linear case with the following one

$$d_{k+1}^T y_k = 0. \tag{18}$$

But, for unconstrained optimization methods, the search direction d_{k+1} can be written as $d_{k+1} = -H_{k+1} g_{k+1}$, where H_{k+1} is an approximation to the inverse of the Hessian $\nabla^2 f(x_{k+1})$, symmetric and positive definite, which satisfies the secant equation $H_{k+1} y_k = s_k$.

Therefore,

$$d_{k+1}^T y_k = -(H_{k+1} g_{k+1})^T y_k = -g_{k+1}^T (H_{k+1} y_k) = -g_{k+1}^T s_k.$$

Hence, the conjugacy condition $d_{k+1}^T y_k = 0$ is satisfied if the line search is exact, since in this case $g_{k+1}^T s_k = 0.$ However, in practical situations, the exact line search is not used. Therefore, it is quite natural to replace the conjugacy condition $d_{k+1}^T y_k = 0$ with this one

$$d_{k+1}^T y_k = -h g_{k+1}^T s_k, \tag{19}$$

where $h \geq 0$ is a scalar. In our numerical experiments we selected $h = 0.5.$
Therefore, from conjugacy condition (19) where d_{k+1} is given by (10) the following value for the scaling parameter t_k is obtained

$$t_k = \frac{(hs_k - y_k)^T g_{k+1}}{(s_k - y_k)^T g_{k+1}}. \tag{20}$$

Now, introducing the value for t_k given by (19) in (10) the following scaled memory-less SR1 search direction is obtained

$$d_{k+1} = -g_{k+1} - \frac{(hs_k - y_k)^T g_{k+1}}{(s_k - y_k)^T y_k}(s_k - y_k). \tag{21}$$

Again observe that the iterations are affected if the denominator $(s_k - y_k)^T y_k$ of the update term is too small. In these cases if

$$\left|(s_k - y_k)^T y_k\right| < \eta \|s_k - y_k\| \|y_k\|, \tag{22}$$

with $\eta = 10^{-8},$ then $d_{k+1} = -g_{k+1}.$

Proposition 1 *The scaled memory-less SR1 search direction (21) computed from the conjugacy conditions is a descent direction*.

Proof It is easy to see that $s_k^T y_k - y_k^T y_k < 0.$ Therefore, from (21) we have

$$\begin{aligned}
g_{k+1}^T d_{k+1} &= -\|g_{k+1}\|^2 - \frac{(hs_k - y_k)^T g_{k+1}}{(s_k - y_k)^T y_k}(g_{k+1}^T s_k - g_{k+1}^T y_k) \\
&\leq -\|g_{k+1}\|^2 + \frac{\|hs_k - y_k\| \|g_{k+1}\|}{(s_k - y_k)^T y_k}(\|g_{k+1}\| \|s_k\| + \|g_{k+1}\| \|y_k\|) \\
&= -\|g_{k+1}\|^2 + \|g_{k+1}\|^2 \frac{\|hs_k - y_k\|}{(s_k - y_k)^T y_k}(\|s_k\| + \|y_k\|) \\
&= -\left[1 - \frac{\|hs_k - y_k\|}{(s_k - y_k)^T y_k}(\|s_k\| + \|y_k\|)\right] \|g_{k+1}\|^2
\end{aligned}$$

$$= -\left[1 + \frac{\|hs_k - y_k\|}{|(s_k - y_k)^T y_k|}(\|s_k\| + \|y_k\|)\right]\|g_{k+1}\|^2 < 0. \quad \blacklozenge$$

4. Accelerated scaled memory-less SR1 algorithm

With these developments, for solving the unconstrained optimization problem (1), the following accelerated scaled memory-less SR1 algorithm can be presented. The algorithm is equipped with an acceleration scheme presented in [Andrei, 2006, 2020a]. Basically, the acceleration scheme modifies the stepsize α_k determined by the Wolfe line search conditions (2) in a multiplicative way to improve the reduction of the function values along the iterations. In the accelerated algorithm, instead of computing $x_{k+1} = x_k + \alpha_k d_k$, the new estimation of the minimum point is computed as

$$x_{k+1} = x_k + \xi_k \alpha_k d_k, \tag{23}$$

where

$$\xi_k = -\frac{\bar{a}_k}{\bar{b}_k}, \tag{24}$$

$\bar{a}_k = \alpha_k g_k^T d_k$, $\bar{b}_k = -\alpha_k (g_k - g_z)^T d_k$, $g_z = \nabla f(z)$ and $z = x_k + \alpha_k d_k$. Hence, if $\bar{b}_k > 0$, then the new estimation of the solution is computed as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, otherwise $x_{k+1} = x_k + \alpha_k d_k$. The acceleration scheme is motivated by the fact that since ρ in the first Wolfe condition (2.a) is small enough (usually $\rho = 0.0001$), the Wolfe line search leads to very small reductions in the function values along the iterations. In this algorithm the parameter *nmeth* is used for selecting the procedure for computation of the scaling parameter t_k. If $nmeth = 1$ then t_k is computed as in (13) by using the sufficient descent condition, while if $nmeth = 2$ then t_k is computed as in (20) by using the conjugacy condition.

Algorithm ASM-SR1

1.	Initialization. Consider an initial point x_0. Set $k = 0$. Select a value for parameter *nmeth*. Choose some values for the parameters c and h. Select some values for the Wolfe line search conditions σ and ρ with $0 < \rho < \sigma < 1$. Compute $g_0 = \nabla f(x_0)$ and set $d_0 = -g_0$. Select a sufficiently small parameters $\varepsilon > 0$ used in the criterion for stopping the iterations and $\varepsilon_A > 0$ used in acceleration scheme
2.	Test a criterion for stopping the iterations: if $\|g_k\|_\infty \le \varepsilon$ then stop the iterations, otherwise go to step 3
3.	Compute the stepsize α_k using the standard Wolfe line search conditions
4.	Update the variables $x_{k+1} = x_k + \alpha_k d_k$ and compute f_{k+1} and g_{k+1}. Compute $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$
5.	Acceleration scheme: a) Compute: $z = x_k + \alpha_k d_k$, $g_z = \nabla f(z)$ and $y_k = g_k - g_z$

	b) Compute: $\bar{a}_k = \alpha_k g_k^T d_k$, and $\bar{b}_k = -\alpha_k y_k^T d_k$ c) If $\left\|\bar{b}_k\right\| \geq \varepsilon_A$, then compute $\xi_k = -\bar{a}_k / \bar{b}_k$ and update the variables as $x_{k+1} = x_k + \xi_k \alpha_k d_k$. Compute f_{k+1} and g_{k+1}. Compute $y_k = g_{k+1} - g_k$ and $s_k = x_{k+1} - x_k$
6.	If $nmeth = 1$ compute the search direction d_{k+1} as in (14) with (15), otherwise if $nmeth = 2$ compute the search direction as in (21) with (22)
7.	Consider $k = k+1$ and go to step 2 ♦

If f is bounded along the direction d_k, then there exists a stepsize α_k satisfying the modified Wolfe line search conditions (2). The first trial of the stepsize crucially affects the practical behavior of the algorithm. At every iteration $k \geq 1$, the starting guess for the step α_k in the line search is computed as $\alpha_{k-1}\|d_{k-1}\| / \|d_k\|$.

5. Convergence of the algorithm

In this section the global convergence of the algorithm is established under the following assumptions:

(1) The level set $\Omega = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded, i.e. there exists a constant $B > 0$ such that for any $x \in \Omega$, $\|x\| \leq B$.

(2) The function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and its gradient is Lipschitz continuous in a neighborhood N of Ω, i.e. there exists a constant $L > 0$ such that $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, for any $x, y \in N$.

It is easy to see that under these assumptions, there exists a constant $\Gamma > 0$ such that $\|\nabla f(x)\| \leq \Gamma$, for any $x \in \Omega$.

Although the search directions d_{k+1} generated by the algorithm ASM-SR1 are always descent directions, in order to get the convergence of the algorithm we need to derive a lower bound for the stepsize α_k. The following propositions are extracted from [Andrei, 2020a].

Proposition 2 *Suppose that the assumption (2) holds and the sequence of the search direction* $\{d_k\}$ *is generated by the algorithm ASM-SR1. Then the stepsize* α_k *satisfies:*

$$\alpha_k \geq \frac{(1-\sigma)\left|g_k^T d_k\right|}{L\|d_k\|^2}. \tag{25}$$

Under the assumptions (1) and (2), the following result, due to Zoutendijk (1970) and Wolfe (1971), is essential in proving the global convergence of the unconstrained optimization algorithms.

Proposition 3 *Suppose that* f *is bounded below in* $\mathbb{R}^n$ *and that* f *is continuously differentiable in a neighborhood* N *of the level set* Ω. *Assume also that the gradient is Lipschitz continuous with constant* $L>0$. *Consider* d_k *is a descent direction and* α_k *satisfies the Wolfe line search conditions (2). Then,*

$$\sum_{k=1}^{\infty}\frac{(g_k^T d_k)^2}{\|d_k\|^2}<\infty. \tag{26}$$

The following result shows that the sequence of gradient norms $\{\|g_k\|\}$ is bounded away from zero only if $\sum_{k\geq 0}1/\|d_k\|<+\infty$.

Proposition 4. *Suppose that the assumptions (1) and (2) hold. Consider the algorithm ASM-SR1 where* d_k *is a descent direction and* α_k *is obtained by the Wolfe line search (2). If*

$$\sum_{k\geq 0}\frac{1}{\|d_k\|^2}=+\infty, \tag{27}$$

then $\liminf_{k\to\infty}\|g_k\|=0.$

For uniformly convex functions the following global convergence result can be proved.

Theorem 1 *Suppose that the assumptions (1) and (2) hold. Let* $\{x_k\}$ *and* $\{d_k\}$ *be generated by the algorithm ASM-SR1. If the minimizing function* f *is uniformly convex on* Ω, *i.e. there exists a constant* $\mu>0$ *such that* $(\nabla f(x)-\nabla f(y))^T(x-y)\geq\mu\|x-y\|^2$, *for any* $x,y\in N$, *then* $\lim_{k\to\infty}\|g_k\|=0.$

Proof From the assumption (2) it follows that $\|y_k\|\leq L\|s_k\|$. From uniform convexity we have $y_k^T s_k\geq\mu\|s_k\|^2$.
It easy to see that

$$\left|(s_k-y_k)^T g_{k+1}\right|\geq\|s_k\|\|g_{k+1}\|$$

and

$$\begin{aligned}\left|(s_k-y_k)^T y_k\right|&=\left|s_k^T y_k-y_k^T y_k\right|\geq\left|\mu\|s_k\|^2-\|y_k\|^2\right|\\&\geq\left|\mu\|s_k\|^2-L^2\|s_k\|^2\right|=\left|\mu-L^2\right|\|s_k\|^2.\end{aligned}$$

From (14), it follows that

$$\left|\frac{(c-1)\|g_{k+1}\|^2}{(s_k-y_k)^T g_{k+1}}\right|\leq\frac{|c-1|\|g_{k+1}\|^2}{\|s_k\|\|g_{k+1}\|}\leq\frac{\Gamma}{\|s_k\|}.$$

Therefore, for the scaled memory-less SR1 search direction (14) obtained from the sufficient descent condition, we have

$$\|d_{k+1}\| \le \|g_{k+1}\| + \left|\frac{(c-1)\|g_{k+1}\|^2}{(s_k - y_k)^T g_{k+1}}\right| (\|s_k\| + \|y_k\|)$$
$$\le \Gamma + \frac{\Gamma}{\|s_k\|}(\|s_k\| + L\|s_k\|) = \Gamma + \Gamma(1+L) \triangleq M_1. \quad (28)$$

On the other hand, from (21) it follows that,

$$\left|\frac{(hs_k - y_k)^T g_{k+1}}{(s_k - y_k)^T y_k}\right| \le \frac{\|s_k - y_k\|\|g_{k+1}\|}{\left|\mu - L^2\right|\|s_k\|^2} \le \frac{(\|s_k\| + \|y_k\|)\Gamma}{\left|\mu - L^2\right|\|s_k\|^2}$$
$$\le \frac{(\|s_k\| + L\|s_k\|)\Gamma}{\left|\mu - L^2\right|\|s_k\|^2} = \frac{(1+L)\Gamma}{\left|\mu - L^2\right|\|s_k\|}.$$

Therefore, for the scaled memory-less SR1 search direction (21) obtained from the conjugacy condition, we have

$$\|d_{k+1}\| \le \|g_{k+1}\| + \frac{(1+L)\Gamma}{\left|\mu - L^2\right|\|s_k\|}(\|s_k\| + \|y_k\|)$$
$$\le \Gamma + \frac{(1+L)^2\Gamma}{\left|\mu - L^2\right|} \triangleq M_2. \quad (29)$$

From Proposition 4, it follows that (27) holds. Therefore, in both situations in which the search direction is computed from the sufficient descent condition and from conjugacy condition we have $\liminf_{k\to\infty} \|g_k\| = 0,$ which for uniformly convex functions is equivalent to $\lim_{k\to\infty} \|g_k\| = 0.$ ♦

6. Numerical experiments

In this section we report some numerical results obtained with an implementation of the ASM-SR1 algorithm. The code is written in Fortran and compiled with f77 (default compiler settings) on a Workstation Intel Pentium 4 with 1.8 GHz. We selected a number of 80 large-scale unconstrained optimization test functions in generalized or extended form we presented in [Andrei, 2020a]. The vast majority of these problems are taken from CUTE collection [Conn, Gould and Toint, 2018]. For each test function we have taken ten numerical experiments with the number of variables increasing as $n = 100, 200, ..., 1000.$ The algorithm implements the Wolfe line search conditions (2) with $\rho = 0.0001,$ $\sigma = 0.8$ and the same stopping criterion $\|g_k\|_\infty \le 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector. In all the algorithms we considered in this numerical study the maximum number of iterations is limited to 10000, while the

maximum number of function and its gradient evaluations is limited to 10000. In criteria for skipping the updates given by (15) and (22) $\eta = 10^{-8}$ is considered.

The comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal value found by ALG1 and ALG2, for problem $i = 1,\ldots,800,$ respectively. We say that, in the particular problem $i,$ the performance of ALG1 was better than the performance of ALG2 if:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{30}$$

and the number of iterations (#iter), or the number of function-gradient evaluations (#fg), or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

In the first set of numerical experiments, we compare the performances of the proposed ASM-SR1 algorithm in which the search direction is given by (14), determined from the sufficient descent condition we call ASM-S, versus BFGS implemented in CONMIN [Shanno and Phua, 1980], we call BFGS. Figure 1 presents the Dolan and Moré [2002] performance profiles, subject to the CPU computing time, of ASM-S, versus BFGS from CONMIN.

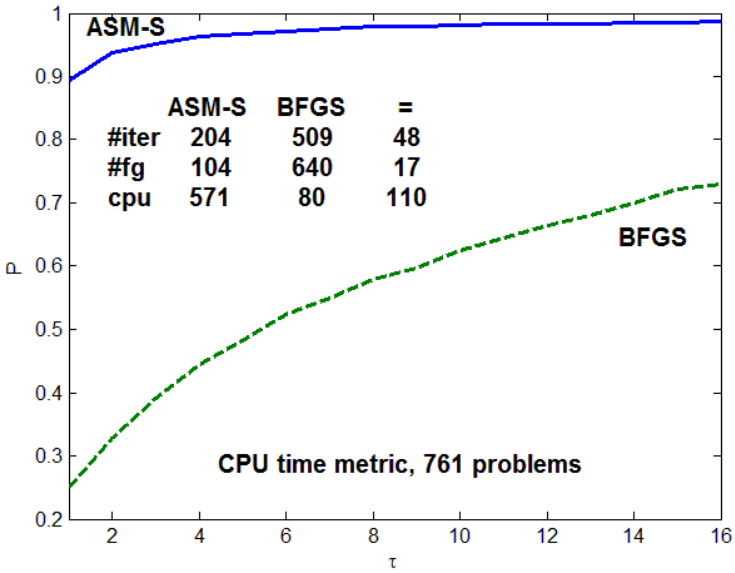

Fig. 1. Performance profile of ASM-S versus BFGS from CONMIN

In a performance profile plot, the top curve corresponds to the method that solved the most problems in a time that was within a given factor of the best time. The percentage of the test problems for which a method is the fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by these algorithms, respectively. Mainly the left side of the plot is a measure of

the efficiency of an algorithm, while the right side is a measure of the robustness of an algorithm.
Comparing ASM-S versus BFGS from CONMIN, (see Fig. 1) subject to the number of iterations, we see that ASM-S was better in 204 problems (i.e. it achieved the minimum number of iterations in 204 problems). BFGS was better in 509 problems and they achieved the same number of iterations in solving 48 problems, etc. Out of 800 problems considered in this numerical experiment, only for 761 problems does the criterion (30) hold. Observe that the differences are significant. Subject to the CPU time metric, ASM-S was better in 571 problems, while BFGS from CONMIN was better only in 80. Both ASM-S and BFGS use the same implementation of the standard Wolfe line search (2) based on cubic interpolation. Therefore, in comparison with BFGS, ASM-S appears to generate the best search direction, on average.

In the second set of numerical experiments we compare the performances of the proposed algorithm ASM-SR1 algorithm in which the search direction is given by (21), determined from the conjugacy condition, we call ASM-C, versus BFGS implemented in CONMIN. Figure 2 presents the Dolan and Moré performance profiles, subject to the CPU computing time, of ASM-C, versus BFGS from CONMIN.

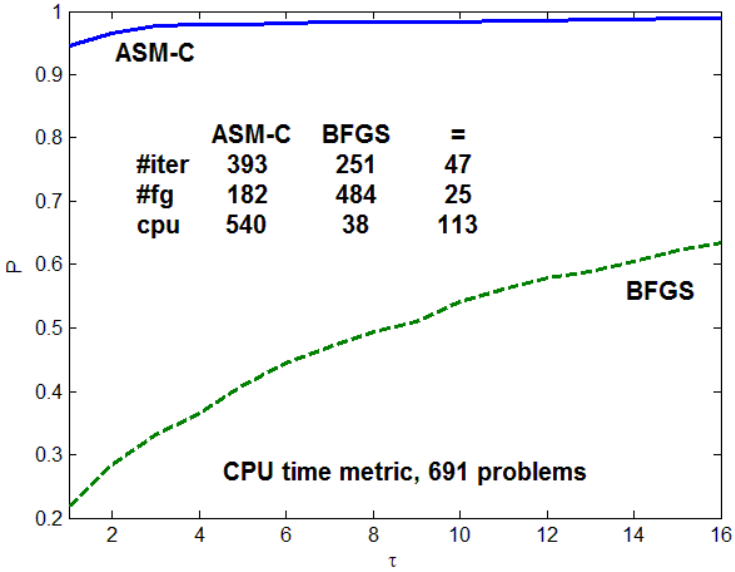

Fig. 2. Performance profile of ASM-C versus BFGS from CONMIN

From Figure 2 we see that subject to the CPU computing time the ASM-C algorithm based on conjugacy condition is way more efficient and more robust than the BFGS in implementation of CONMIN. From Figure 2 we see that, subject CPU time metric, ASM-C was better in solving 540 problems, while BFGS was better only in 38 problems. Observe the difference between the performance profiles presented in Figure 1 and in Figure 2, respectively. The performances of ASM-C versus BFGS are significantly better than those of ASM-S versus BFGS.

Table 1 presents the number of iterations (#iter) and the CPU computing time (cpu) in seconds for solving 9 problems from our collection, each of them having $n = 1000$ variables.

Table 1. Performances of ASM-C, ASM-S and BFGS for solving 9 problems, each of them with $n = 1000$ variables

	ASM-C		**ASM-S**		**BFGS**	
Problem's name	#iter	cpu	#iter	cpu	#iter	cpu
Generalized Rosenbrock	4749	1.98	3951	2.17	3492	128.39
Extended trigonometric	3659	3.42	4682	3.40	1925	49.94
DIXMAANL	4960	8.93	4947	10.55	4465	147.91
NONDQAR	4000	0.71	4624	2.66	5577	154.22
DIXON3DQ	4993	0.56	3839	0.50	1329	36.20
CUBE	4029	1.71	3942	3.44	3780	119.78
Perturbed quadratic PQ2	3350	0.46	3724	1.73	3074	83.88
BIGGSB1	4993	0.54	4035	1.76	685	18.66
QUARTICM	30	0.0	54	0.01	620	22.59
Total	**34763**	**18.31**	**33798**	**26.22**	**24947**	**761.37**

From Table 1 we see that subject to the number of iterations BFGS is more efficient versus both ASM-C and ASM-S. However, subject to the CPU computing time, both ASM-C and ASM-S are way more efficient. For example, ASM-C is approximately 41 times faster than BFGS, while ASM-S is 29 times faster than BFGS.

In the third set of numerical experiments let us compare ASM-S versus ASM-C. Figure 3 presents the performances of these algorithms for solving 800 unconstrained optimization problems from our collection with the number of variable in the range [100, 1000].

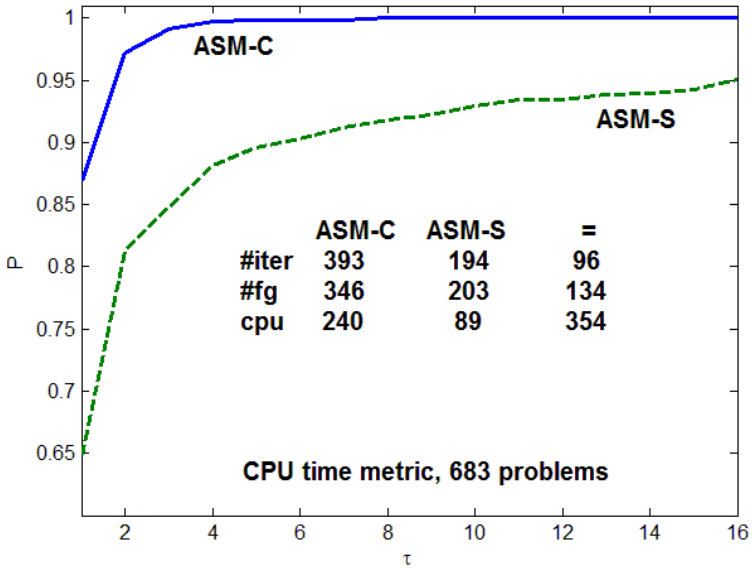

Fig. 3. Performance profiles of ASM-C versus ASM-S. $n \in [100,1000]$

Figure 3 shows that ASM-C is more efficient and more robust than ASM-S. For example, subject to the CPU computing time, ASM-C was faster in 240 problems, while ASM-S was faster only in 89 problems. In Proposition 1 we proved that the search direction generated by the ASM-C algorithm is a descent direction. Additionally, it satisfies the conjugacy condition which proves to be an important property subject to its efficiency and robusteness. It is worth mentioning that in our numerical experiments we noticed that the condition (22) is very rare fulfilled, i.e. the search direction given by the negative gradient very rare is used. In contrast, in ASM-S the negative gradient is more often used.

Figure 4 presents the performances of ASM-C versus ASM-S for solving our set of 800 problems in which the number of variables is in the range [1000, 10000]. Observe that ASM-C is more efficient and more robust than ASM-S and the differences are significant.
Observe that the search directions corresponding to ASM-S and ASM-C are descent. Besides, the coefficients multiplying $(s_k - y_k)$ in both search directions (14)-(15) of ASM-S and (21)-(22) of ASM-C are positive. However, the search direction of ASM-C satisfies the conjugacy condition. The conjugacy condition is important in the economy of unconstrained optimization algorithms. The main reason for using conjugate directions is that to minimize a convex quadratic function in a subspace spanned by a set of mutually conjugate directions is equivalent to minimize the function along each conjugate direction in turn. This is the reason why the ASM-C is more efficient and more robust versus ASM-S. Of course, the performances of algorithms satisfying the conjugacy condition are strongly dependent on the accuracy of the line search. In our numerical experiments we used the Wolfe line search (2) in implementation of Shanno (1983) (see also Andrei (2020a), Chapter 5).

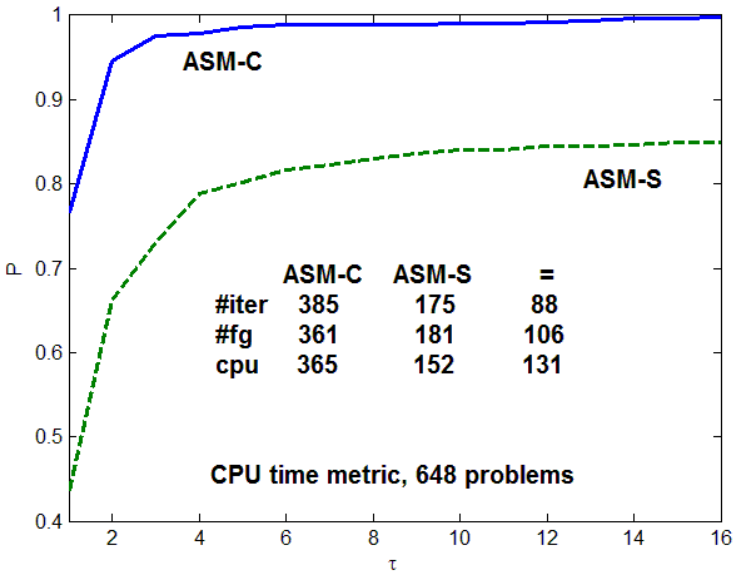

Fig. 4. Performance profiles of ASM-C versus ASM-S $n \in [1000,10000]$

In the last set of numerical experiments let us present comparisons between ASM-C and ASM-S algorithms for solving some applications from the MINPACK-2 test problem collection [Averick, Carter, Moré, & Xue, 1992]. The minimizing function of all these applications is quadratic. In Table 2, we present these applications, as well as the values of their parameters. The infinite-dimensional version of these problems is transformed into a finite element approximation by triangulation. Thus a finite-dimensional minimization problem is obtained whose variables are the values of the piecewise linear function at the vertices of the triangulation. The discretization steps are $nx = 200$ and $ny = 200$, thus obtaining minimization problems with 40.000 variables.

Table 2
Applications from the MINPACK-2 collection.

A1	Elastic–plastic torsion [Glowinski, 1984, pp. 41–55], c = 5
A2	Pressure distribution in a journal bearing [Cimatti, 1977], b = 10, ε = 0.1
A3	Optimal design with composite materials [Goodman, Kohn, & Reyna, 1986], λ = 0.008
A4	Steady-state combustion [Aris, 1975, pp. 292–299], [Bebernes, & Eberly, 1989], λ = 5
A5	Minimal surfaces with Enneper conditions [Nitsche, 1989, pp. 80–85]

The performances of ASM-C versus ASM-S are given in Table 3, where #iter is the number of iterations, #fg is the number of function and its gradient evaluations and cpu is the CPU time computing.

Table 3
Performances of ASM-C versus ASM-S (40.000 variables, CPU seconds)

	ASM-C			**ASM-S**		
	#iter	#fg	cpu	#iter	#fg	cpu
A1	13138	26297	273.64	1197	3061	32.19
A2	66922	133876	1608.61	9691	20279	228.76
A3	89559	179187	3430.25	6665	16587	295.33
A4	49631	99287	3815.94	1715	4394	172.60
A5	11374	22784	300.46	718	1831	20.83
TOTAL	**230624**	**461431**	**9428.90**	**19986**	**46152**	**749.71**

Observe that both these algorithms are able to solve large-scale unconstrained optimization problems, but subject to CPU computing time ASM-S is more efficient than ASM-C. For solving these applications the search direction given by the negative gradient (see criteria (15) and (22)) very rare is used. For example, for solving the application A3, ASM-C needs 89559 iterations, out of which only in 2130 iterations the negative gradient is used.

7. Conclusion

This paper presents a new variant of the symmetric rank-one SR1 updating known as the scaled memory-less SR1 method in which the updating formula is applied to the unity

matrix and the updating term is scaled with a parameter determined by sufficient descent condition or by the conjugacy condition. Convergence results and numerical tests show that the scaled memory-less SR1 method is both globally convergent and faster than a variant of the preconditioned BFGS method implemented in CONMIN.

References

Al-Baali, M., (1998). Numerical experience with a class of self-scaling quasi-Newton algorithms, *Journal of Optimization Theory and Applications, 96*, 533-553.

Andrei, N., (2006). An acceleration of gradient descent algorithm with backtracking for unconstrained optimization. *Numerical Algorithms, 42*(1), 63-73.

Andrei, N., (2007). Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. *Optimization Methods and Software, 22*(4), 561-571.

Andrei, N., (2018a). A double parameter scaled BFGS method for unconstrained optimization. *Journal of Computational and Applied Mathematics, 332*, 26-44.

Andrei, N., (2018b). A double-parameter scaling Broyden–Fletcher–Goldfarb–Shanno method based on minimizing the measure function of Byrd and Nocedal for unconstrained optimization. *Journal of Optimization Theory and Applications 178*, 191-218.

Andrei, N., (2018c). An adaptive scaled BFGS method for unconstrained optimization. *Numerical Algorithms 77*, 413-432.

Andrei, N., (2020a). *Nonlinear conjugate gradient methods for unconstrained optimization.* Springer Optimization and Its Applications 158.

Andrei, N., (2020b). New conjugate gradient algorithms based on self‑scaling memoryless Broyden–Fletcher–Goldfarb–Shanno method. *CALCOLO* Article number 17 (2020), 57-17.

Aris, R., (1975). The Mathematical Theory of Diffusion and Reaction in Permeable Catalysts, Oxford.

Arzam, M.R., Babaie-Kafaki, S., & Ghanbari, R., (2017). An extended Dai-Liao conjugate gradient method with global convergence for nonconvex functions. *Glasnik Matematicki, 52*(72), 361-375.

Averick, B.M., Carter, R.G., Moré, J.J. & Xue, G.L., (1992). *The MINPACK-2 test problem collection*, Mathematics and Computer Science Division, Argonne National Laboratory, Preprint MCS-P153-0692.

Bebernes, J., & Eberly, D., (1989). Mahematical Problems from Combustion Theory, in: Applied Mathematical Sciences, vol. 83, Springer-Verlag.

Benson, H.Y., & Shanno, D.F., (2018). Cubic regularization in symmetric rank-1 quasi-Newton methods. *Math. Progr. Comput. 10*, 457-486.

Biggs, M.C., (1971). Minimization algorithms making use of non-quadratic properties of the objective function. *Journal of the Institute of Mathematics and Its Applications, 8*, 315-327.

Biggs, M.C., (1973). A note on minimization algorithms making use of non-quadratic properties of the objective function. *Journal of the Institute of Mathematics and Its Applications, 12*, 337-338.

Broyden, C.G., (1967) Quasi-newton methods and their applications to function minimization. *Mathematics of Computation 21*(99), 368-381.

Broyden, C.G., (1970). The convergence of a class of double-rank minimization algorithms. I. General considerations. *Journal of the Institute of Mathematics and Its Applications, 6,* 76-90.

Chen, H., Lam, W.H., & Chan, S.C., (2019). On the convergence analysis of cubic regularized symmetric rank-1 quasi-Newton method and the incremental version in the application of large-scale problems. *IEEE Access*, volume 7, 114042-114059.

Cimatti, G., (1977). On a problem of the theory of lubrication governed by a variational inequality, *Applied Mathematics and Optimization 3*, 227–242.

Conn, A.R., Gould, N.I.M., & Toint, Ph.L., (1991). Convergence of quasi-newton matrices generated by the symmetric rank one update. *Mathematical Programming 50*(1-3), 177-195.

Conn, A.R., Gould, N.I.M., & Toint, Ph.L., (2018). Constrained and unconstrained testing environment. http://www.cuter.rl.ac.uk/Problems/mastsif.shtml.

Contreras, M., & Tapia, R.A., (1993). Sizing the BFGS and DFP updates: A numerical study. *Journal of Optimization Theory and Applications, 78*, 93-108.

Davidon, W.C., (1959). *Variable metric method for minimization.* (Research and Development Report ANL-5990. Argonne National Laboratories.)

Davidon, W.C., (1968). Variance algorithm for minimization. Computer Journal 10(4), 406-410.

Davidon, W.C., (1991). Variable metric method for minimization. *SIAM J. Optim. 1*(1), 1-17.

Dehmiry, A.H., (2019). The global convergence of the BFGS method under a modified Yuan-Wei-Lu line search technique. *Numerical Algorithms*. DOI.org/10.1007/s11075-019-00779-7

Dolan, E.D., & Moré, J.J., (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*, 201-213.

Fiacco, A.V., & McCormick, G.P., (1968). Nonlinear programming: sequential unconstrained minimization techniques. Research Analysis Corporation, McLean Virginia. Republished in 1990 by SIAM, Philadelphia.

Fletcher, R., & Powell, M.J.D., (1963). A rapidly convergent descent method for minimization, *Computer Journal*, 163–168.

Fletcher, R., (1970). A new approach to variable metric algorithms. *The Computer Journal, 13*, 317-322.

Glowinski, R., (1984). *Numerical Methods for Nonlinear Variational Problems*, Springer-Verlag, Berlin.

Goldfarb, D., (1970). A family of variable metric method derived by variation mean. *Mathematics of Computation, 23*, 23-26.

Goodman, J., Kohn, R., & Reyna, L., (1986). Numerical study of a relaxed variational problem from optimal design, *Computer Methods in Applied Mechanics and Engineering 57*, 107–127.

Huang, H.Y., (1970). Unified approach to quadratically convergent algorithms for functions minimization. *Journal of Optimization Theory and Applications 5*(6), 405-423.

Kelley, C.T., & Sachs, E.W., (1998). Local convergence of the symmetric rank one iteration. *Computational Optimization and Applications, 9*, 43–63.

Khalfan, H.F., Byrd, R.H., & Schnabel, R.B., (1993). A theoretical and experimental study of the symmetric rank-one update. *SIAM J. Optimization 3*(1), 1-24.

Liao, A., (1997). Modifying BFGS method. *Operations Research Letters, 20*, 171-177.

Nitsche, J.C.C., (1989). *Lectures on Minimal Surfaces*, Vol. 1, Cambridge University Press.

Nocedal, J., (1992). Theory of algorithms for unconstrained optimization. *Acta Numerica, 1*, 199-242.

Nocedal, J., & Wright, S.J., (2006). *Numerical optimization.* Springer Series in Operations Research. Springer Science+Business Media, New York, Second edition, 2006.

Nocedal, J., & Yuan, Y.X., (1993). Analysis of self-scaling quasi-Newton method. *Mathematical Programming, 61*, 19-37.

Oren, S.S., & Luenberger, D.G., (1974). Self-scaling variable metric (SSVM) algorithms, part I: criteria and sufficient conditions for scaling a class of algorithms. *Management Science, 20*, 845-862.

Oren, S.S., & Spedicato, E., (1976). Optimal conditioning of self-scaling variable metric algorithm. *Mathematical Programming, 10*, 70-90.

Powell, M.J.D., (1970). A new algorithm for unconstrained optimization. In: J.B. Rosen, O.L. Mangasarian & K. Ritter (Eds.) *Nonlinear Programming*. Academic Press, New York, 31-66.

Shanno, D.F., (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation, 24*, 647-656.

Shanno, D.F., (1978a). Conjugate gradient methods with inexact searches, *Mathematics of Operations Research, 3*, 244-256.

Shanno, D.F., (1978b). On the convergence of a new conjugate gradient algorithm, *SIAM Journal on Numerical Analysis, 15*, 1247-1257.

Shanno, D.F., (1983). *CONMIN – A Fortran subroutine for minimizing an unconstrained nonlinear scalar valued function of a vector variable x either by the BFGS variable metric algorithm or by a Beale restarted conjugate gradient algorithm*. Private communication, October 17, 1983.

Shanno, D.F., & Phua, K.H., (1976). Algorithm 500. Minimization of unconstrained multivariable functions. *ACM Transactions on Mathematical Software, 2*, 87-94.

Wolfe, P., (1969). Convergence conditions for ascent methods. *SIAM Review, 11*, 226-235.

Wolfe, P., (1971). Convergence conditions for ascent methods. II: Some corrections. *SIAM Review, 13*, 185-188.

Yabe, H., Martínez, H.J., & Tapia, R.A., (2004). On sizing and shifting the BFGS update within the sized Broyden family of secant updates. *SIAM Journal on Optimization, 15*(1), 139-160.

Yuan, Y.X., (1991). A modified BFGS algorithm for unconstrained optimization. *IMA Journal of Numerical Analysis, 11*, 325-332.

Yuan, Y.X., & Byrd. R., (1995). Non-quasi-Newton updates for unconstrained optimization. *Journal of Computational Mathematics, 13*(2), 95-107.

Yuan, G., Sheng, Z., Wang, B., Hu, W., & Li, C., (2018). The global convergence of a modified BFGS method for nonconvex functions. *Journal of Computational and Applied Mathematics, 327,* 274-294.

Zoutendijk, G., (1970). Nonlinear programming, computational methods. In J. Abadie (Ed.) *Integer and Nonlinear Programming*. North-Holland, Amsterdam, 38-86.

-----ooooo0ooooo-----