Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 55 (2008) 1201–1214
An International Journal
computers & mathematics with applications
www.elsevier.com/locate/camwa

A new quasi-Newton pattern search method based on symmetric rank-one update for unconstrained optimization

Ting Wu*, Linping Sun

Department of Mathematics, Nanjing University, Nanjing, 210093, PR China

Received 3 July 2006; received in revised form 22 March 2007; accepted 21 June 2007

Abstract

This paper proposes a new robust and quickly convergent pattern search method based on an implementation of OCSSR1 (Optimal Conditioning Based Self-Scaling Symmetric Rank-One) algorithm [M.R. Osborne, L.P. Sun, A new approach to symmetric rank-one updating, IMA Journal of Numerical Analysis 19 (1999) 497–507] for unconstrained optimization. This method utilizes the factorization of approximating Hessian matrices to provide a series of convergent positive bases needed in pattern search process. Numerical experiments on some famous optimization test problems show that the new method performs well and is competitive in comparison with some other derivative-free methods.

Keywords: Pattern search; Positive basis; Quasi-Newton algorithm; SR1 update; Derivative-free optimization

1. Introduction

We consider the unconstrained optimization problem

$$\min f(x) \tag{1.1}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$ and $f(x)$ is assumed to be a continuously differentiable function.

To the best of our knowledge, quasi-Newton methods are one of the most classic and effective methods for solving problems (1.1). If given an approximation $B^{(k)}$ to $\nabla^2 f(x^{(k)})$, $H^{(k)}$ the $[\nabla^2 f(x^{(k)})]^{-1}$ and $g^{(k)}$ the gradient $\nabla f(x^{(k)})$ at current iterate $x^{(k)}$, a search direction $d^{(k)}$ can be obtained:

$$d^{(k)} = -H^{(k)} g^{(k)}.$$

Then we implement inexact line search along $d^{(k)}$ to get a new iterate:

$$x^{(k+1)} = x^{(k)} - \alpha H^{(k)} g^{(k)} \tag{1.2}$$

* Corresponding author.
E-mail address: wutingmath@yahoo.com (T. Wu).

0898-1221/$ - see front matter
doi:10.1016/j.camwa.2007.06.012

where $\alpha > 0$ is chosen such that

$$f(x^{(k)}) - f(x^{(k+1)}) \geq \sigma\alpha {g^{(k)}}^T H^{(k)} g^{(k)} \tag{1.3a}$$

and

$$|{g^{(k+1)}}^T H^{(k)} g^{(k)}| \leq \tau {g^{(k)}}^T H^{(k)} g^{(k)} \tag{1.3b}$$

are satisfied. Here, $\sigma \in (0, \frac{1}{2})$, $\tau \in (\sigma, 1)$.

Finally, update $H^{(k)}$ to $H^{(k+1)}$ by requiring the quasi-Newton equation to hold, which is

$$H^{(k+1)} y^{(k)} = s^{(k)}$$

where $y^{(k)} = g^{(k+1)} - g^{(k)}$, $s^{(k)} = x^{(k+1)} - x^{(k)}$.

There are a lot of update formulae for generating $H^{(k+1)}$. Among them, Symmetric Rank-One (SR1) method employs the symmetric rank-one update

$$H^{(k+1)} = H^{(k)} + \frac{(s^{(k)} - H^{(k)} y^{(k)})(s^{(k)} - H^{(k)} y^{(k)})^T}{(s^{(k)} - H^{(k)} y^{(k)})^T y^{(k)}} \tag{1.4}$$

and possesses an interesting finite termination property without requiring exact line searches. However, SR1 update is unstable in the sense that, even if $H^{(k)}$ is positive definite, $H^{(k+1)}$ may be singular, indefinite or undefined. On the other hand, the rank-two update, including DFP update and BFGS update, is most widely used in quasi-Newton methods. Particularly, BFGS update has merits in paratic use. For example, BFGS update could keep the positive definition of matrix $H^{(k+1)}$ if $H^{(k)}$ is positive definite.

Some recent results on SR1 update show that when SR1 update and some other quasi-Newton updates, such as DFP update and BFGS update, are available simultaneously, SR1 update is more efficient. Meanwhile, the sequence of ${H^{(k)}}^{-1}$, the approximation to Hessian matrix of $f(x)$ at iterate $x^{(k)}$, can converge to the actual Hessian matrix at the solution. This has encouraged a lot of researchers to seek some modified schemes of SR1 update which possess not only good features of original SR1 update but also greater numerical stability. We consider the self-scaling SR1 formula given by Sun [2]:

$$H^{(k+1)} = \theta H^{(k)} + \frac{(s^{(k)} - \theta H^{(k)} y^{(k)})(s^{(k)} - \theta H^{(k)} y^{(k)})^T}{(s^{(k)} - \theta H^{(k)} y^{(k)})^T y^{(k)}}. \tag{1.5}$$

For this update formula, Osborne and Sun [1] proposed a new algorithm(OCSSR1) with Davidon's optimal condition, in which the scaling factor θ is taken either as

$$\theta_1 = \frac{c}{b} - \sqrt{\frac{c^2}{b^2} - \frac{c}{a}} \tag{1.6}$$

or

$$\theta_2 = \frac{c}{b} + \sqrt{\frac{c^2}{b^2} - \frac{c}{a}} \tag{1.7}$$

where $a = {y^{(k)}}^T H^{(k)} y^{(k)}$, $b = {s^{(k)}}^T y^{(k)}$ and $c = {s^{(k)}}^T {H^{(k)}}^{-1} s^{(k)}$. Numerical reports show that this algorithm compares favorably with good implementations of the BFGS method.

It is well-known that when the information of derivatives of objective function is unreliable or unavailable, those derivative-based methods will not perform well including quasi-Newton methods. This problem inspires an implementation [3] of OCSSR1 algorithm which compares favorably with good implementations of the BFGS method. If the derivatives are available, the implementation is equivalent to the OCSSR1 algorithm. Moreover, it is especially useful when gradient information is estimated by finite difference formulae.

However, in some cases, for example, random errors may render finite difference approximations to the gradient unreliable. Hence, we cannot only depend on analytical derivatives and need to consider derivative-free methods, such as direct search methods.

Pattern search methods are a kind of classic direct search methods. The original pattern search methods were proposed by Box [4] in the late 1950s and by Hooke and Jeeves [5] in the early 1960s. Owing to their simplicity and practical use, pattern search methods have been still widely used. Recently, more and more researchers paid attention to pattern search methods for unconstrained optimization and did a lot of work on them, including multidirectional search of Dennis and Torczon [6], generalized pattern search method of Torczon [7], and grid-based framework of Coope and Price [8].

However, every sword has two edges. Although simplicity is its characteristic, the implementation of this algorithm is relatively time consuming in comparison with the conventional derivative-based algorithms. This is because there are so many search directions in every pattern search process. To overcome this limitation, many researchers modified the pattern search methods by adopting some techniques.

Generally speaking, there are two difficulties in improving the pattern search method, i.e., choice of the iterate acceptance criterion and determination of the search direction set. Filter approach and nonmonotone technique are used to establish new acceptance criteria [9,10]. For the latter one, in the old time search direction set Φ usually consisted of a basis of $\mathbb{R}^n$ and its opposite direction. Although it is able to explore the full space of $\mathbb{R}^n$, it will bring too many search directions and consequently reduce the efficiency of algorithms. Recently, positive bases, conjugate property and parallel processing are used to deal with this problem, especially positive bases [11,12]. Positive basis is very suitable for replacing basis in Φ. It could explore the full space of $\mathbb{R}^n$ too, but the number of its cardinality is between $n+1$ and $2n$. For example, if V is a basis of $\mathbb{R}^n$, then

$$V_+ = [V, -Ve] \tag{1.8}$$

is a positive basis of $\mathbb{R}^n$, where $e = [1, 1, \ldots, 1]^T$. Hence, it can reduce at most a half of search directions as before. However, this replacement brings a new problem, that is how to select a series of different positive bases in algorithms, which is never referred to in many references to the best of our knowledge.

Based on long-range research on OCSSR1 algorithm and pattern search methods, a new quasi-Newton pattern search method based on symmetric rank-one update is proposed. In this method, the conjugate factorization of approximating Hessian matrices $B^{(k)}$ is utilized to provide a series of positive bases which are needed in pattern search process. The use of conjugate directions will bring benefit to pattern search [13]. Furthermore, the participation of symmetric rank-one algorithm can accelerate the convergence of pattern search methods and save the function evaluations. Besides, a switch [14] can be set here to control the choice of difference formulae according to the property of positive bases which will be detailed below.

This paper is organized as follows. The next section is concerned with a brief description of generalized pattern search methods. Then an implementation of OCSSR1 algorithm is introduced in Section 3. In Section 4, the framework of our new method is proposed, while the convergence analysis is examined in Section 5. Finally, numerical results are reported to show the feasibility and effectiveness of the proposed new method.

2. Pattern search methods

Generally speaking, pattern search methods generate a sequence of iterates $\{x^{(k)}\}$ without using any information of the derivatives, including gradient and second-order derivative, of the objective function. They only depend on function values. In fact, ordinal information about function values is enough. At each iteration, the objective function is evaluated at a finite number of trial points (defined below) and the purpose is to look for one point which can yield a lower function value than the current iterate. If such a point is found, it is set to be the new iterate, and the iteration is called successful; Otherwise, we declare it unsuccessful and update the trial points.

Torczon [7] generalizes this kind of methods and presents a global convergence theory for them. In Torczon's generalized pattern search method, two components are needed, one is a pattern $P^{(k)}$, and the other one is a real scale factor $\Delta^{(k)} > 0$.

The pattern $P^{(k)}$ consists of a basis matrix A and a generating matrix $C^{(k)}$. Among them, any nonsingular matrix in $\mathbb{R}^{n\times n}$ could be chosen as A and $C^{(k)} \in \mathbb{Z}^{n\times p}(p > 2n)$ could be partitioned into three parts:

$$C^{(k)} = [M^{(k)} - M^{(k)}\, D^{(k)}]$$

where $M^{(k)} \in M \subset \mathbb{Z}^{n\times n}$, M is a finite set of nonsingular matrices, $D^{(k)} \in \mathbb{Z}^{n\times(p-2n)}$ contains at least one column, the column of zeros.

Then $P^{(k)}$ could be written as

$$P^{(k)} = AC^{(k)} = [AM^{(k)} \ -AM^{(k)}\, AD^{(k)}].$$

It is noted that because both the basis matrix A and the generating matrix $C^{(k)}$ have rank n, the columns of $P^{(k)}$ can span $\mathbb{R}^n$.

The real scale factor $\Delta^{(k)}$ works as a step length parameter. Let $C^{(k)} = [c_1^{(k)}, c_2^{(k)}, \ldots, c_p^{(k)}]$, then a trial step $s_i^{(k)}$ is defined as any vector of the form

$$s_i^{(k)} = \Delta^{(k)} A c_i^{(k)}.$$

In fact, $Ac_i^{(k)}$ determines the direction of the step and at iteration k, all the trial points possess the form of

$$x_i^{(k)} = x^{(k)} + s_i^{(k)}$$

where $x^{(k)}$ is the current iterate.

Then a generalized pattern search method could be outlined as follows:

Let initial iterate $x^{(0)} \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $C^{(0)} \in \mathbb{Z}^{n\times p}$, $\Delta^{(0)} > 0$, $k = 0$.

While (Stopping conditions do not hold) do

Step 1. Find a step $s^{(k)}$ using Exploratory Moves($\Delta^{(k)}$, $AC^{(k)}$).
Step 2. If $f(x^{(k)} + s^{(k)}) < f(x^{(k)})$, then $x^{(k+1)} = x^{(k)} + s^{(k)}$. Otherwise, $x^{(k+1)} = x^{(k)}$.
Step 3. Update $C^{(k)}$ and $\Delta^{(k)}$to $C^{(k+1)}$ and $\Delta^{(k+1)}$, $k = k + 1$.

More specific information about generalized pattern search method can be seen in [7].

Torczon's generalized pattern search method (GPSM) could be regarded as a method on successively refined meshes $G^{(k)} = \{x^{(k)} + \Delta^{(k)} A c_i^{(k)}\}$, where $c_i^{(k)}$, $i = 1, 2, \ldots, p$ are the columns of $C^{(k)}$. Later, Coope and Price [8] proposed a class of grid-based methods (GBM) which are similar to Torczon's method, but permit greater freedom in the orientation and scaling of successive grids.

Compared with GPSM, the analogues of the matrix A, $C^{(k)}$ and the scaling factor $\Delta^{(k)}$ in GBM are less restricted. Firstly, in GPSM A is fixed and independent of iteration k and only the integer matrix $C^{(k)}$ could be changed in every iteration. However, in GBM Coope and Price use positive bases to replace A and they could change from time to time. Secondly, at every iteration GBM also considers other trial points on the grid besides the points on the directions of positive bases. This behavior equals the effects of integer matrix $C^{(k)}$ in GPSM. Finally, the mesh size in GBM could be irrational, while the scaling factor $\Delta^{(k)}$ in GPSM only could be rational. Anyway, they are both proved to be convergent by different methods.

3. An approach to symmetric rank-one update without derivatives

As the BFGS update formula and the original SR1 update formula [15], formula (1.5) too could be written in product form and consequently generates an implementation of OCSSR1 algorithm.

At first, we rewrite update formula (1.5) in product form:

$$H^{(k+1)} = (I + u^{(k)} v^{(k)T}) \theta H (I + u^{(k)} v^{(k)T})^T \tag{3.1}$$

where

$$u^{(k)} = \mu(s^{(k)} - \theta H^{(k)} y^{(k)}) \tag{3.2}$$

$$v^{(k)} = \frac{(B^{(k)} s^{(k)} - \theta y^{(k)})}{\theta} \tag{3.3}$$

$$\mu = \frac{-\theta \pm \sqrt{(c\theta - b\theta^2)/(b - a\theta)}}{c - 2b\theta + a\theta^2}. \tag{3.4}$$

Let

$$B^{(k)-1} = L^{(k)} L^{(k)T},$$

where $L^{(k)} = [l_1^{(k)}, l_2^{(k)}, \ldots, l_n^{(k)}]$ is a nonsingular $n \times n$ matrix.

Then (1.5) equals to

$$L^{(k+1)} = \sqrt{\theta}[I + u^{(k)} v^{(k)^T}] L^{(k)}. \tag{3.5}$$

Clearly, two points should be noted:

Firstly, Osborne and Sun [1] pointed out that if $\theta > 0$ and $s^{(k)^T} y^{(k)} > 0$, then $H^{(k+1)}$ is positive definite if and only if

$$\theta \notin \left[\frac{s^{(k)^T} y^{(k)}}{y^{(k)^T} H^{(k)} y^{(k)}}, \frac{s^{(k)^T} H^{(k)^{-1}} s^{(k)}}{s^{(k)^T} y^{(k)}} \right]. \tag{3.6}$$

Thus, in (3.4), it always holds that $\frac{(c\theta - b\theta^2)}{(b - a\theta)} > 0$. In addition, $c - 2b\theta + a\theta^2 > 0$ because $ac \geq b^2$.

Secondly, the columns of $L^{(k)}$ satisfy the conjugacy conditions

$$\begin{cases} l_i^{(k)^T} B^{(k)} l_j^{(k)} = 0, & i \neq j, \\ l_i^{(k)^T} B^{(k)} l_i^{(k)} = 1, & \text{otherwise} \end{cases}$$

and $L^{(k)}$ is not unique here. In fact, if $L^{(k)}$ is post-multiplied by a $n \times n$ orthogonal matrix N, then we have

$$L^{(k)} N \cdot (L^{(k)} N)^T = L^{(k)} N N^T L^{(k)^T} = L^{(k)} L^{(k)^T} = B^{(k)^{-1}}.$$

$L^{(k)} N$ has the same property as $L^{(k)}$.

Now let $\widehat{g}^{(k)} = L^{(k)^T} g^{(k)}$, and its components are:

$$\widehat{g}_i^{(k)} = l_i^{(k)^T} g^{(k)} \quad i = 1, 2, \ldots, n. \tag{3.7}$$

They are just the directional derivatives of $f(x)$ in the directions $l_i^{(k)}$, $i = 1, 2, \ldots, n$, which are the columns of $L^{(k)}$. At the same time, we obtain the quasi-Newton direction

$$d^{(k)} = -L^{(k)} \widehat{g}^{(k)}.$$

According to (3.7), the components of $\widehat{g}^{(k)}$ from $\widehat{g}_1^{(k)}$ to $\widehat{g}_n^{(k)}$ could be calculated directly by finite difference method, and there are two kinds of difference formulae to be chosen, i.e., forward difference formula and the more accurate central difference formula.

Forward difference formula:

$$\widehat{g}_i^{(k)} = \frac{f(x^{(k)} + \delta_i l_i^{(k)}) - f(x^{(k)})}{\delta_i} + O(\delta_i). \tag{3.8a}$$

Central difference formula:

$$\widehat{g}_i^{(k)} = \frac{f(x^{(k)} + \delta_i l_i^{(k)}) - f(x^{(k)} - \delta_i l_i^{(k)})}{2\delta_i} + O(\delta_i^2). \tag{3.8b}$$

Now let

$$\omega^{(k)} = L^{(k)^T} v^{(k)} \tag{3.9}$$

then

$$u^{(k)} = \theta \mu L^{(k)} \omega^{(k)} \tag{3.10}$$

and

$$L^{(k+1)} = \sqrt{\theta} L^{(k)} [I + \theta \mu \omega^{(k)} \omega^{(k)^T}]. \tag{3.11}$$

Moreover, we have

$$\widehat{y}^{(k)} = L^{(k)^T} y^{(k)} = \breve{g}^{(k+1)} - \widehat{g}^{(k)}, \tag{3.12}$$

where $\breve{g}^{(k+1)} = L^{(k)^T} g^{(k+1)}$ and (3.9) can be written as

$$\omega^{(k)} = -\left[\widehat{y}^{(k)} + \left(\frac{\alpha}{\theta}\widehat{g}^{(k)}\right)\right]. \tag{3.13}$$

In order to calculate θ, the following equations are obtained by using (3.12) and $\widehat{g}^{(k)} = L^{(k)^T} g^{(k)}$:

$$a = \widehat{y}^{(k)^T}\widehat{y}^{(k)} \tag{3.14a}$$

$$b = -\alpha\widehat{g}^{(k)^T}\widehat{y}^{(k)} \tag{3.14b}$$

and

$$c = \alpha^2\widehat{g}^{(k)^T}\widehat{g}^{(k)}. \tag{3.14c}$$

Finally, we will consider the key point of the implementation, that is how to choose θ. θ is chosen by a heuristic strategy which aims at making $\kappa(L^{(k+1)}L^{(k+1)^T})$ smaller, where $\kappa(\cdot)$ denotes the condition number of a matrix. We choose $\theta = \theta_1$ except when the following condition is satisfied:

$$\mathrm{Tr}(L^{(k+1)}L^{(k+1)^T})_{\theta=\theta_1} \geq \mathrm{Tr}(L^{(k+1)}L^{(k+1)^T})_{\theta=\theta_2} \tag{3.15}$$

where $\mathrm{Tr}(\cdot)$ is the trace of a matrix. In fact, from Osborne and Sun [1] we know that

$$\det(L^{(k+1)}L^{(k+1)^T})_{\theta=\theta_1} \leq \det(L^{(k+1)}L^{(k+1)^T})_{\theta=\theta_2} \tag{3.16}$$

which implies that

$$\prod_{i=1}^{n}\lambda_i^{\theta_1} < \prod_{i=1}^{n}\lambda_i^{\theta_2}$$

where λ_i^{θ} are the eigenvalues of the matrix $(L^{(k+1)}L^{(k+1)^T})_{\theta}$ at $\theta = \theta_j$ $(j = 1, 2)$ with $\lambda_1^{\theta} \leq \lambda_2^{\theta} \leq \cdots \leq \lambda_n^{\theta}$. On the other hand,

$$\mathrm{Tr}(L^{(k+1)}L^{(k+1)^T})_{\theta} = \sum_{i=1}^{n}\lambda_i^{\theta}.$$

Thus, $\kappa(L^{(k+1)}L^{(k+1)^T})_{\theta=\theta_2}$ is probably smaller than $\kappa(L^{(k+1)}L^{(k+1)^T})_{\theta=\theta_1}$ when (3.15) holds. It is not difficult to calculate the values $\mathrm{Tr}(L^{(k+1)}L^{(k+1)^T})$. For the initial $L^{(1)} = I$, we have

$$\begin{aligned}\mathrm{Tr}(L^{(k+1)}L^{(k+1)^T}) &= \theta\left[\mathrm{Tr}(L^{(k)}L^{(k)^T}) + \frac{r^{(k)^T}r^{(k)}}{\omega^{(k)^T}\widehat{y}^{(k)}}\right] \\ &= \theta\left[\sqrt{\theta}\mathrm{Tr}(L^{(k)}L^{(k)^T}) + \theta^{\frac{3}{2}}\mu\|r^{(k)}\|_2^2 + \frac{r^{(k)^T}r^{(k)}}{\omega^{(k)^T}\widehat{y}^{(k)}}\right]\end{aligned} \tag{3.17}$$

where $r^{(k)} = L^{(k)}\omega^{(k)}$.

4. Framework

It is obvious that the updating of L at every iteration could provide a series of positive bases as discussed in Section 3 and this is just what we need. Consequently, we can certainly combine this process with grid-based pattern search methods. The positive basis generated from unit matrix I by (1.8) could be regarded as the initial search

direction set Φ. When a round of searchings finishes and not any successful step is taken, we will record the current iterate and call it pattern search point in this paper. Directional derivatives of $f(x)$ at current iterate in the search direction set Φ can be calculated by finite difference method, and hence a quasi-Newton direction is obtained. Then inexact line search along this quasi-Newton direction can be done to seek a new iterate and update $L^{(k)}$ to $L^{(k+1)}$ at the same time. Last, we generate a new positive basis from $L^{(k+1)}$ by (1.8) again, which consists of the new search direction set, and start a new pattern search process along directions in the new Φ. As the process repeats, a sequence of pattern search points are obtained. These can converge to one or more stationary points of the objective function $f(x)$.

For convenience, we drop the subscript + to denote positive basis in this paper.

Algorithm SR1P (Symmetric Rank-One Update Pattern Search Method)

Initialize $k = 1$, $m = 1$, $L^{(1)} = I$. Let $x^{(1)}$ be the initial point and produce positive basis $\widetilde{V}_+^{(1)}$ from $L^{(1)}$ by (1.8). Set $V_+^{(1)} = \widetilde{V}_+^{(1)}$ and $F^{(m)} = \xi$, where ξ is a positive constant.

While (Stopping condition does not hold [see Remark 4.2]) do

Step 1. Choose grid mesh size $h^{(k)}$. Set $i = 1, num = 0$. Let $\eta^{(k)} = |V_+^{(k)}|$, where $|V_+^{(k)}|$ denotes the number of columns in matrix $V_+^{(k)}$.

Let $\Sigma^{(m)}$ denote the grid generated by current iterate $x^{(k)}$, grid mesh size $h^{(k)}$ and positive basis $V_+^{(k)}$.

Step 2. While ($num < \eta^{(k)}$) do

(a) Calculate the values of $f(x)$ on a finite number of points on grid $\Sigma^{(m)}$ including $\{x^{(k)} + h^{(k)}\upsilon_{+i}^{(k)}\}$, where $\upsilon_{+i}^{(k)}$ denotes the ith column vector in $V_+^{(k)}$.

If there is any point x_{tempt} which satisfies

$$f(x_{tempt}) < f(x^{(k)}) - (h^{(k)})^2,$$

let it be $x^{(k+1)}$. $h^{(k+1)} = \phi h^{(k)}$, where ϕ is a positive integer. If $h^{(k+1)} > F^{(m)}$, let $h^{(k+1)} = h^{(k)}$. Set $V_+^{(k+1)} = V_+^{(k)}$.

$k = k + 1$.

$num = 0$.

else,

$num = num + 1$.

(b) Set $i = i + 1$.

If $i > \eta^{(k)}$, set $i = 1$.

end

Step 3. Let $\widetilde{x}^{(m)} = x^{(k)}$ and $\widetilde{h}^{(m)} = h^{(k)}$.

Step 4. Construct directional derivatives $\widehat{g}^{(m)}$ at $\widetilde{x}^{(m)}$ in every directions in $L^{(k)}$ from (3.8) according to the property of positive bases.

Step 5. $d^{(m)} = -L^{(m)}\widehat{g}^{(m)}$.

Step 6. Implement inexact line search to look for a descent point $x^{(k+1)}$ along direction $d^{(m)}$ such that criterion (1.3a) is satisfied, which is

$$f(x^{(k+1)}) \le f(\widetilde{x}^{(m)}) - \sigma\alpha\widehat{g}^{(m)^T}\widehat{g}^{(m)},$$

where $\sigma = 0.0001$ is a typical value.

Step 7. Estimate the vector g_{tempt} of directional derivatives at $x^{(k+1)}$ in every direction in $L^{(m)}$ from difference formulae (3.8). Let $\widehat{y}^{(m)} = g_{tempt} - \widehat{g}^{(m)}$. If $\|\widehat{y}^{(m)}\|_2 \ge \epsilon_1$, continue to step 8; Otherwise, terminate the algorithm.

Step 8. If $\widehat{g}^{(m)^T}\widehat{y}^{(m)} > -\epsilon_2\|\widehat{g}^{(m)}\|_2\|\widehat{y}^{(m)}\|_2$,

set $L^{(m+1)} = L^{(m)}$, $V_+^{(m+1)} = V_+^{(m)}$ and go to step 13.

Step 9. If $-(\alpha\widehat{g}^{(m)} + \widehat{y}^{(m)})^T\widehat{y}^{(m)} > \epsilon_2\|\alpha\widehat{g}^{(m)} + \widehat{y}^{(m)}\|_3\|\widehat{y}^{(m)}\|_2$,

set $\theta = 1$ and go to step 12. Otherwise, continue to step 10.

Step 10. If $\|L^{(m)}(\widehat{y}^{(m)} + \alpha\gamma\widehat{g}^{(m)})\|_2 \le \epsilon_3$,
where $a = \widehat{y}^{(m)^T}\widehat{y}^{(m)}$, $b = -\alpha\widehat{g}^{(m)^T}\widehat{y}^{(m)}$ and $\gamma = \frac{a}{b}$, set

$$L^{(m+1)} = \frac{1}{\sqrt{\gamma}}L^{(m)}$$

and go to step 13. Otherwise, calculate θ_1 and θ_2 by (1.6) and (1.7).
Step 11. Determine θ.
Step 12. Update $L^{(m)}$ to $L^{(m+1)}$ by (3.9)–(3.11).
Step 13. Produce $\widetilde{V}_+^{(m+1)} = [L^{(m+1)}, -L^{(m+1)}e]$, let $V_+^{(k+1)} = \widetilde{V}_+^{(m+1)}$ and set
$F^{(m+1)} = \zeta F^{(m)}$, where $\zeta \in (0, 1)$.
$k = k + 1$, $m = m + 1$.

End

This algorithm consists of two loops, i.e., inner loop and outer loop. The inner loop (step 2) searches the new iterate along all the members of positive basis $V_+^{(k)}$ in turn. In step 2(a) the finite number of points on grid could be only $x^{(k)} + h^{(k)}\upsilon_{+i}^{(k)}$ or include some other points which are random. num is the number recording the times of failed search. When $num = \eta^{(k)}$, it means that no improved point can be found around the current iterate. Hence, a pattern search point, just the current iterate, is obtained. Then the inner loop finishes and the outer loop begins. It is noted that because we use not simple decrease but sufficient decrease here, the current iterate may not be a pattern search point. However it does not influence the convergence of the Algorithm SR1P and the use of sufficient decrease can reduce the number of iteration. Consequently, such a point is still acceptable to us and we just call it acceptable pattern search point. Compared with the inner loop, the outer loop is used to choose grid mesh size, generate quasi-Newton direction, employ inexact line search, update L and produce positive basis. Besides, because Algorithm SR1P is a grid-based algorithm, we just use the grid mesh size $h^{(k)}$ as the difference interval. In a word, these two loops work together to look for acceptable pattern search points and positive bases.

Remark 4.1. It is well-known that when an acceptable pattern search point $\widetilde{x}^{(m)}$ is found, the objective function has been evaluated at a finite number of points including $\{\widetilde{x}^{(m)} + \widetilde{h}^{(m)}\widetilde{\upsilon}_{+i}^{(m)}, i = 1, 2, \ldots, \widetilde{\eta}^{(m)}\}$ and $\widetilde{x}^{(m)}$, where $\widetilde{\eta}^{(m)} = |\widetilde{V}_+^{(m)}|$. Thus, if forward difference formula (3.8a) is used in Algorithm SR1P, no more function value is needed. Even if central difference formula (3.8b) is chosen, only n function values must be calculated. Although more function values are needed in central difference formula, this formula is more accurate than forward difference formula. Consequently, the choice of these two alternative formulae is very important for us.

Consider the simplest positive basis

$$V_+ = [V, -Ve],$$

where $V = [v_1, v_2, \ldots, v_n]$ is an arbitrary basis of $\mathbb{R}^n$ and $e = [1, 1, \ldots, 1]^T$. Let

$$f_0 = f(x),$$
$$f_i = f(x + hv_i), \quad i = 1, 2, \ldots, n$$
$$f_{n+1} = f(x - hVe),$$

where h is set by the grid mesh size.

Then, we want to obtain $\widehat{g}$ by solving $(n + 1)$ equations:

$$\widehat{g}_i = \frac{(f_i - f_0)}{h}, \quad i = 1, 2, \ldots, n \tag{4.1}$$

$$-(\widehat{g}_1 + \widehat{g}_2 + \cdots + \widehat{g}_n) = \frac{f_{n+1} - f_0}{h}. \tag{4.2}$$

Their solution is

$$\widehat{g}_i = \frac{f_i - \overline{f}}{h}, \tag{4.3}$$

where $\overline{f} = \frac{f_1 + f_2 + \cdots + f_{n+1}}{n+1}$.

Consequently, when we ignore the $O(h)$ terms, the values obtained by forward difference formula is the same as that of (4.3). However, we find that if x is sufficiently close to a strict local minimizer, these two values will be different, the reason is that in this case

$$f_0 < f_j$$

then we have

$$(n+1) f_0 < \sum_{j=1}^{n+1} f_j,$$

$$f_0 < \overline{f}.$$

So the value of (4.1) and (4.3) will not be the same.

According to this result, a switch can be set in the process of difference. If the value of both (4.1) and (4.3) are within a relative tolerance, such as 5% or 10%, either of them could be accepted. Otherwise, we will give up forward difference formula and utilize central difference formula. This switch can help us to judge when to use forward difference formula and when to use central difference formula by only requiring an extra function value.

Remark 4.2. In general pattern search methods, the process of adjusting the grid mesh size is a little complex. In Algorithm SR1P, we use a simple self-adjusting method to adjust h. When a new iterate is found, we will augment h by scale and when an acceptable pattern point is obtained, h will be reduced by scale too. This behavior can avoid making h too large or too small, which will decrease the efficiency of algorithms.

Furthermore, h is also used to set stopping condition in Algorithm SR1P as in a lot of pattern search methods. When the mesh size $h < \epsilon$, the algorithm will terminate. This criterion is simple and widely used in the literature.

Moreover, there is another stopping condition in step 7 in Algorithm SR1P which is concerned with $\|\widehat{y}^{(m)}\|_2$. If $\|\widehat{y}^{(m)}\|_2 \geq \epsilon_1$, continue to step 8. Otherwise, terminate the algorithm.

Consequently, it is noted that the participation of the implementation of OCSSR1 algorithm can accelerate the convergence of pattern search methods. For example, even if the initial iterate $x^{(1)}$ is the global minimizer of the objective function, the pattern search algorithm cannot terminate, it will go on running until the grid mesh size is small enough. However, when such a problem occurs in Algorithm SR1P, the stopping condition in step 7 will be satisfied and the algorithm will terminate after a turn of one pattern search.

5. Convergence analysis

It is noted that Algorithm SR1P generates not a sequence of pattern search points but a sequence of acceptable pattern search points. However, it does not influence the convergence of the algorithm, because the grid mesh size $h^{(k)}$ will tend to zero when k goes to $+\infty$. As such, the convergence of the Algorithm SR1P can be guaranteed according to Theorem 4.1 in [8]. For completeness, the whole proof is given as follows.

Theorem 5.1 (*Convergence of Algorithm SR1P*). *Consider problem (1.1), assume that:*

(1°) $f(x)$ *is continuously differentiable;*
(2°) *The sequence of iterates* $\{x^{(k)}\}$ *generated from Algorithm SR1P is bounded;*
(3°) $\|H^{(m)}\|_2 \leq K$, *where* K *is positive constants independent of* m*;*
(4°) $\lim_{m\to\infty} h^{(m)} = 0$,

then each cluster point of the sequence of acceptable pattern search points is a stationary point of $f(x)$.

Proof. Firstly, we will prove that the sequence of acceptable pattern search points generated in Algorithm SR1P is infinite.

In Algorithm SR1P we use $\{\widetilde{x}^{(m)}\}$ to denote the sequence of acceptable pattern search points, which is the subsequence of $\{x^{(k)}\}$.

We can know from (2°) that there exists a compact set Γ which satisfies $\{x^{(k)}\} \subset \Gamma$. So the set $\Gamma \cap \Sigma^{(m)}$ is finite, where $\Sigma^{(m)}$ denotes the grid on iteration m. Furthermore, because each iterate selected by sufficient decrease is distinctly different from all others, only a finite number of iterates is generated using $\Sigma^{(m)}$. Then it follows from

the finiteness of step 2(a) in the inner loop of Algorithm SR1P that this algorithm generates only a finite number of iterates using each grid. The only way the algorithm can change from the grid $\Sigma^{(m)}$ is, if the last iterate generated using the grid $\Sigma^{(m)}$ is an acceptable pattern search point.

Consequently, the sequence of acceptable pattern search points is infinite.

We can know from condition (3°) that for all m,

$$\|H^{(m)}\|_2 = \|B^{(m)^{-1}}\|_2 = \|L^{(m)}L^{(m)^T}\|_2 = \|L^{(m)^T}\|_2^2 \le K.$$

Then

$$\|L^{(m)^T}\|_2 \le \sqrt{K}.$$

From the compatibility of matrix norm, there exists a positive constant β which satisfies

$$\beta\|L^{(m)^T}\|_\infty \le \|L^{(m)^T}\|_2,$$

and it follows that

$$\|L^{(m)}\|_1 = \|L^{(m)^T}\|_\infty \le \frac{1}{\beta}\|L^{(m)^T}\|_2 \le \frac{\sqrt{K}}{\beta}.$$

$$\|l_i^{(m)}\|_1 \le \frac{\sqrt{K}}{\beta}, \quad i = 1, 2, \ldots, n.$$

Because positive bases $\{V_+^{(m)}\}$ are generated from $\{L^{(m)}\}$ by (1.8), there exists a positive constant K' such that for all m

$$\|\upsilon_{+i}^{(m)}\|_1 \le K', \quad i = 1, 2, \ldots, \eta$$

where $\eta = \eta^{(m)}$, $\eta^{(m)} = |V_+^{(m)}|$.

Then we choose a specific cluster point $\widetilde{x}^{(\infty)}$ of $\{\widetilde{x}^{(m)}\}$. Because $\|\upsilon_{+i}^{(m)}\|_1 \le K'$ for all m and $i = 1, 2, \ldots, n$, there is a subsequence $\{\overline{x}^{(m)}\}$ of $\{\widetilde{x}^{(m)}\}$ which has a unique limit $\widetilde{x}^{(\infty)}$, and corresponding subsequences $\{\overline{\upsilon}_{+i}^{(m)}\}$ of $\{\upsilon_{+i}^{(m)}\}$ which has a unique limit $\{\overline{\upsilon}_{+i}^{(\infty)}\}$ for $i = 1, 2, \ldots, n$.

All the positive bases $\{V_+^{(m)}\}$ in Algorithm SR1P are generated from $\{L^{(m)}\}$ by (1.8). Accordingly, they have the same structure and

$$\lim_{m\to\infty} \overline{\upsilon}_{+i}^{(m)} = \overline{\upsilon}_{+i}^{(\infty)}, \quad \forall i = 1, 2, \ldots, \eta,$$

$$\|\overline{\upsilon}_{+i}^{(\infty)}\|_1 \le K', \quad \forall i = 1, 2, \ldots, \eta.$$

Furthermore, the fact that the sequence of $B^{(k)}$ could converge to the actual Hessian matrix at the solution indicates that $\{\overline{\upsilon}_{+1}^{(\infty)}, \overline{\upsilon}_{+2}^{(\infty)}, \ldots, \overline{\upsilon}_{+n}^{(\infty)}\}$ is a basis of $\mathbb{R}^n$. Hence, $\{\overline{\upsilon}_{+1}^{(\infty)}, \overline{\upsilon}_{+2}^{(\infty)}, \ldots, \overline{\upsilon}_{+\eta}^{(\infty)}\}$ is also a positive basis possessing the same structure with all $\{\widetilde{V}_+^{(m)}\}$.

Now, for all $i \in 1, 2, \ldots, \eta$,

$$f(\overline{x}^{(m)} + \overline{h}^{(m)}\overline{\upsilon}_{+i}^{(m)}) \ge f(\overline{x}^{(m)}) - (\overline{h}^{(m)})^2. \tag{5.1}$$

Furthermore,

$$\begin{aligned} f(\overline{x}^{(m)} + \overline{h}^{(m)}\overline{\upsilon}_{+i}^{(m)}) &= f(\overline{x}^{(m)}) + \int_{t=0}^{\overline{h}^{(m)}} [g(\overline{x}^{(m)} + t\overline{\upsilon}_{+i}^{(m)}) - \overline{g}^{(m)} + \overline{g}^{(m)}]^T \overline{\upsilon}_{+i}^{(m)} \mathrm{d}t \\ &= f(\overline{x}^{(m)}) + \overline{h}^{(m)}(\overline{g}^{(m)})^T \overline{\upsilon}_{+i}^{(m)} + E_i^{(m)} \end{aligned} \tag{5.2}$$

where $g(x) \equiv \nabla f(x)$, $\overline{g}^{(m)} \equiv g(\overline{x}^{(m)})$ and

$$E_i^{(m)} = \int_{t=0}^{\overline{h}^{(m)}} [g(\overline{x}^{(m)} + t\overline{\upsilon}_{+i}^{(m)}) - \overline{g}^{(m)}]^T \overline{\upsilon}_{+i}^{(m)} \mathrm{d}t.$$

Then

$$|E_i^{(m)}| \le \int_{t=0}^{\overline{h}^{(m)}} K' M_i^{(m)} \mathrm{d}t = \overline{h}^{(m)} K' M_i^{(m)},$$

where $M_i^{(m)} = \max\{\|g(\overline{x}^{(m)} + t\overline{v}_{+i}^{(m)}) - \overline{g}^{(m)}\|_1 : \ t \in [0, \overline{h}^{(m)}]\}$.

From the continuity of g and the compactness of Γ we can know that g is uniformly continuous on Γ. The bound on $\|\overline{v}_{+i}^{(m)}\|_1$ and condition (4°) ensure that

$$\lim_{m\to\infty} M_i^{(m)} = 0.$$

We can know from (5.1) and (5.2) that for all $i = 1, 2, \ldots, \eta$,

$$\overline{h}^{(m)} (\overline{g}^{(m)})^T \overline{v}_{+i}^{(m)} + \overline{h}^{(m)} K' M_i^{(m)} \ge -(\overline{h}^{(m)})^2. \tag{5.3}$$

Hence, condition (4°) implies that when $m \to +\infty$, we have

$$(\widetilde{g}^{(\infty)})^T \overline{v}_{+i}^{(\infty)} \ge 0 \quad \forall\, i = 1, 2, \ldots, \eta,$$

where $\widetilde{g}^{(\infty)} \equiv g(\widetilde{x}^{(\infty)})$.

It follows from Theorem 2.1 in [8] that $\widetilde{g}^{(\infty)} = \nabla f(\widetilde{x}^{(\infty)}) = 0$.

Because the choice of the cluster point is arbitrary, every cluster point of the sequence of acceptable pattern search points is a stationary point of the objective function.

The proof is completed. □

Remark 5.1. Theorem 5.1 gives the convergence of Algorithm SR1P without the stopping condition in step 7. Even when this stopping condition is considered, the convergence of Algorithm SR1P can still be guaranteed by the property of line search which accelerates the convergence of Algorithm SR1P at the same time.

Remark 5.2. In Algorithm SR1P, the reasons that all the positive bases are generated from (1.8) are that it is the simplest form of positive basis possessing the minimal cardinality and for convenience of proving. However, this is not the only choice and the use of (1.8) may not always be more efficient. More positive bases with other generating methods can also be used and the convergence theorem will still hold with a little modification.

6. Numerical results

Numerical experiments on some famous optimization test problems are implemented to test the efficiency of the proposed algorithm. All tests are implemented on a PC with 2.66 GHz Pentium 4 and 256 MB SDRAM using Matlab 6.5.

In our algorithm, we let $\epsilon_1 = 10^{-8}$, $\epsilon_2 = \epsilon_3 = \epsilon = 10^{-5}$ and restrict the number of iterations within 10 000.

The test functions are outlined as follows:

TF. 1 Beale $x^{(1)} = (1, 1)$,
TF. 2 Rosenbrock $x^{(1)} = (-1.2, 1)$,
TF. 3 Extended Powell $x^{(1)} = (3, -1, 0, 1, \ldots, 3, -1, 0, 1)$,
TF. 4 Dixon $x^{(1)} = (-2, -2, -2, -2, -2, -2, -2, -2, -2, -2)$,
TF. 5 Hilbert $x^{(1)} = (-4, -2, -1.333, -1)$,
TF. 6 Power $x^{(1)} = (1, 1, \ldots, 1)$,
TF. 7 Trigonometric $x^{(1)} = (1/n, 1/n, \ldots, 1/n)$,
TF. 8 Brown Badly Scaled $x^{(1)} = (1, 1)$,
TF. 9 Modified Cragg $x^{(1)} = (1, 2, 2, 2)$,
TF. 10 Broyden Tridiagonal $x^{(1)} = (-1, -1, \ldots, -1)$,
TF. 11 Brown and Dennis $x^{(1)} = (25, 5, -5, -1)$,
TF. 12 Wood $x^{(1)} = (-3, -1, -3, -1)$,
TF. 13 Penalty $x^{(1)} = (1, 2, \ldots, n)$,
TF. 14 Tridia $x^{(1)} = (1, 1, \ldots, 1)$,

Table 6.1
Numerical results for Algorithm SR1P

Function	Dimension	N_t	N_f	f_t	f^*
TF. 1	$N = 2$	29	85	0	0
TF. 2	$N = 2$	29	89	0.368×10^{-20}	0
TF. 3	$N = 4$	59	260	0.678×10^{-10}	0
TF. 4	$N = 10$	20	180	0.740×10^{-31}	0
TF. 5	$N = 4$	23	106	0.196×10^{-37}	0
TF. 6	$N = 20$	2	51	0	0
	$N = 50$	2	111	0	0
TF. 7	$N = 5$	59	315	0.585×10^{-16}	0
TF. 8	$N = 2$	149	427	0	0
TF. 9	$N = 4$	37	161	0.175×10^{-8}	0
TF. 10	$N = 10$	14	149	0.136×10^{-8}	0
TF. 11	$N = 4$	257	1144	0.858222×10^{5}	85822.2
TF. 12	$N = 4$	123	516	0.239×10^{-19}	0
TF. 13	$N = 4$	70	411	0.225×10^{-4}	0.225×10^{-4}
TF. 14	$N = 10$	21	196	0.829×10^{-33}	0
	$N = 50$	75	3034	0.703×10^{-15}	0

where TF. 1, TF. 2, TF. 3, TF. 4, TF. 7, TF. 8, TF. 10, TF. 11 and TF. 12 appear in More et al. [16]; TF. 4 appears in Wolfe and Viazminsky [17]; TF. 5 appears in Schittkowski [18]; TF. 6 appears in Spedicato [19]; TF. 9 appears in Gill and Murray [20]; TF. 13 and TF. 14 appear in Shanno [21].

For each test function, Table 6.1 reports the numerical results of Algorithm SR1P, including the dimension of the objective function argument (N), the number of iterations (N_t), the number of function evaluations (N_f), the value of the objective function at the termination point x_t (f_t) and the value of the objective function at the minimum x^* (f^*).

Table 6.2 presents the experiment results on the examples adopted in Algorithm NGOCSSR1 [3]. For the purpose of comparison, the number of function evaluations (N_f) and the value of the objective function at the termination point x_t (f_t) are given.

Table 6.2
Comparison of SR1P with NGOCSSR1

Function	Dimension	Algorithm	N_f	f_t
TF. 1	$N = 2$	SR1P	85	0
		NGOCSSR1	81	0.339×10^{-10}
TF. 2	$N = 2$	SR1P	89	0.368×10^{-20}
		NGOCSSR1	124	0.148×10^{-10}
TF. 3	$N = 4$	SR1P	260	0.678×10^{-10}
		NGOCSSR1	387	0.543×10^{-10}
TF. 4	$N = 10$	SR1P	180	0.740×10^{-31}
		NGOCSSR1	971	0.353×10^{-10}
TF. 5	$N = 4$	SR1P	106	0.196×10^{-37}
		NGOCSSR1	47	0.598×10^{-11}
TF. 6	$N = 20$	SR1P	51	0
		NGOCSSR1	1480	0.699×10^{-10}
	$N = 50$	SR1P	111	0
		NGOCSSR1	819	0.183×10^{-13}
TF. 7	$N = 5$	SR1P	315	0.585×10^{-16}
		NGOCSSR1	355	0.845×10^{-11}

Table 6.2 (*continued*)

Function	Dimension	Algorithm	N_f	f_t
TF. 8	$N = 2$	SR1P	427	0
		NGOCSSR1	58	0.575×10^{-14}
TF. 9	$N = 4$	SR1P	161	0.175×10^{-8}
		NGOCSSR1	455	0.726×10^{-10}
TF. 10	$N = 10$	SR1P	149	0.136×10^{-8}
		NGOCSSR1	845	0.158×10^{-11}
TF. 11	$N = 4$	SR1P	1144	0.858222×10^{5}
		NGOCSSR1	209	0.858222×10^{5}
TF. 12	$N = 4$	SR1P	516	0.239×10^{-19}
		NGOCSSR1	354	0.266×10^{-11}
TF. 13	$N = 4$	SR1P	411	0.225×10^{-4}
		NGOCSSR1	653	0.225×10^{-4}
TF. 14	$N = 10$	SR1P	196	0.829×10^{-33}
		NGOCSSR1	255	0.188×10^{-11}
	$N = 50$	SR1P	3034	0.703×10^{-15}
		NGOCSSR1	5053	0.647×10^{-18}

It can be seen from Table 6.1 that Algorithm SR1P is practical and efficient. Furthermore, Table 6.2 shows that this algorithm is competitive when compared with some other derivative-free algorithms, such as [3]. Even when facing some functions with higher dimension, such as TF. 4, TF. 6, TF. 10 and TF. 14, it can still perform well with a small number of function evaluations.

7. Conclusions

This paper proposes a new pattern search algorithm based on an implementation of OCSSR1 algorithm for unconstrained optimization. Incorporating such symmetric rank-one update algorithm (i.e. OCSSR1) into the framework of the conventional pattern search algorithm, a series of different positive bases can be obtained. Apart from this, the participation of the implementation of OCSSR1 algorithm can help in accelerating the convergence of pattern search methods and save the function evaluation number. In addition, a switch of Coope and Price [14] based on positive bases is utilized here to determine how to choose the difference formula in the process of differences. The results from numerical experiments show that the new algorithm is practical, efficient and performs better than some other derivative-free algorithms.

Anyway, the pattern search methods are a kind of classic, simple and efficient method for unconstrained optimization. More modifications could be made by incorporating some new elements into pattern search methods to develop and perfect them!

References

[1] M.R. Osborne, L.P. Sun, A new approach to symmetric rank-one updating, IMA Journal of Numerical Analysis 19 (1999) 497–507.
[2] L.P. Sun, A self-scaling symmetric rank one updating algorithm for unconstrained optimization, Journal of Chinese Universities (English Series) 3 (1984) 15–25.
[3] L.P. Sun, A quasi-Newton algorithm without calculating derivatives for unconstrained optimization, Journal of Computational Mathematics 12 (1994) 380–386.
[4] G.E.P. Box, Evolutionary operation: A method for increasing industrial productivity, Applied Statistics 6 (1957) 81–101.
[5] R. Hooke, T.A. Jeeves, Direct search solution of numerical and statistical problems, Journal of the Association for Computing Machinery (ACM) 8 (1961) 212–219.
[6] J.E. Dennis, V. Torczon, Direct search methods on parallel machines, SIAM Journal on Optimization 1 (1991) 448–474.
[7] V. Torczon, On the convergence of pattern search algorithms, SIAM Journal on Optimization 7 (1997) 1–25.
[8] I.D. Coope, C.J. Price, On the convergence of grid-based methods for unconstrained minimization, SIAM Journal on Optimization 11 (2001) 859–869.

[9] T. Wu, L.P. Sun, A filter-based pattern search method for unconstrained optimization, Numerical Mathematics. A Journal of Chinese Universities (English Series) 15 (3) (2006) 209–216.
[10] T. Wu, L.P. Sun, Solving unconstrained optimization problem with a filter-based nonmonotone pattern search algorithm, Report 05/02, Division of Optimization, Department of Mathematics, Nanjing University, 2005.
[11] C. Davis, Theory of positive linear dependence, American Journal of Mathematics 76 (1954) 733–746.
[12] R.M. Lewis, V. Torczon, Rank ordering and positive bases in pattern search algorithms, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681–0001, 1996.
[13] D. Byatt, I.D. Coope, C.J. Price, Conjugate grids for unconstrained optimisation, Computational Optimization and Applications 29 (1) (2004) 49–68.
[14] I.D. Coope, C.J. Price, Positive bases in numerical optimization algorithm with automatic scaling, Computational Optimization and Applications 21 (2) (2002) 169–175.
[15] K.W. Brodlie, A.R. Gourlay, J. Greenstadt, Rank-one and rank-two corrections to positive definite matrices expressed in product form, Journal of the Institute of Mathematics and its Applications 11 (1973) 73–82.
[16] J.J. More, B.S. Garbow, K.E. Hillstrom, Testing unconstrained optimization software, ACM Transactions on Mathematical Software 7 (1981) 17–41.
[17] M.A. Wolfe, C. Viazminsky, Supermemory descent methods for unconstrained minimization, Journal of Optimization Theory and Applications 18 (1976) 455–469.
[18] K. Schittkowski, More Test Examples for Nonlinear Programming Codes, in: Lecture Notes in Economics and Mathematical Systems, vol. 282, Springer, Berlin, 1987.
[19] E. Spedicato, Computational experience with quasi-Newton algorithms for minimization problems of moderately large size, Report CISE-N-175, CISE Documentation Series, Segrato, 1975.
[20] P.E. Gill, W. Murray, Nonlinear least squares and nonlinearly constrained optimization, in: G.A. Watsom (Ed.), Numerical Analysis (Dundee, 1975), 1976, pp. 135–147.
[21] D.F. Shanno, Conjugate gradient methods with inexact searches, Mathematics of Computation 34 (1978) 499–514.