Scaled Conjugate Gradient Algorithms for Unconstrained Optimization

Neculai Andrei[1]
Research Institute for Informatics,
Center for Advanced Modeling and Optimization,
8-10, Averescu Avenue, Bucharest 1, Romania
E-mail: nandrei@ici.ro

Abstract. In this work we propose and analyse a new scaled conjugate gradient algorithm and its implementation, based on an interpretation of the secant equation and on the inexact Wolfe line search conditions. The best spectral conjugate gradient algorithm by Birgin and Martinez [6], which mainly is a scaled variant of Perry's [25], is modified in such a manner to overcome the lack of positive definiteness of the matrix defining the search direction. This modification is based on the quasi-Newton BFGS updating formula. The computational scheme is imbedded into the restart philosophy of Beale-Powell. The parameter scaling the gradient is selected as spectral gradient or in an anticipative manner by means of a formula using the function values in two successive points. In very mild conditions it is shown that, for strongly convex functions, the algorithm is global convergent. Computational results and performance profiles, for a test set consisting of 500 unconstrained optimization problems, show that this new scaled conjugate gradient algorithm substantially outperforms known conjugate gradient methods including: spectral conjugate gradient by Birgin and Martinez, Fletcher and Reeves, Polak and Ribière, and Hestens and Stiefel.

Keywords: Unconstrained optimization, conjugate gradient method, spectral gradient method
AMS Subject Classification: 49M07, 49M10, 90C06, 65K

1. Introduction

Let us consider the following unconstrained optimization problem:

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

where $f:R^n \to R$ is continuously differentiable and its gradient is available. We are interested to elaborate an algorithm for solving large-scale case for which the Hessian of f is either not available or requires a large amount of storage and computational costs.

The conjugate gradient methods, we consider in this paper, represent an important innovation for solving large-scale unconstrained optimization problems. These methods generate a sequence x_k of approximations to the minimum x^* of f, in which

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$
$$d_{k+1} = -g_{k+1} + \beta_k d_k\ , \tag{3}$$

[1] The author was supported by the Romanian Academy Grant 168/2003.

where $g_k = \nabla f(x_k)$, α_k is selected to minimize $f(x)$ along the search direction d_k, and β_k is a scalar parameter. The iterative process is initialized with an initial point x_0 and $d_0 = -g_0$. We know a lot of versions of conjugate gradient methods which correspond to the selection procedure of parameter β_k. A history of conjugate gradient and Lanczos algorithms from the very beginning until 1976 is given by Golub and O'Leary [15].

When the function f is quadratic and α_k is selected to minimize $f(x)$ along the direction d_k, then all choices of β_k are equivalent, but for general nonlinear functions, different choices of β_k give algorithms with very different convergence performances.

These methods start with the paper presented by Hestens and Stiefel [19][2] for solving symmetric, positive definite systems of linear equations in which β_k is selected as:

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \tag{4}$$

where $y_k = g_{k+1} - g_k$. Fletcher and Reeves [13] extended this method to nonlinear unconstrained optimization where the parameter β_k is computed as:

$$\beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \tag{5}$$

where $\|.\|$ is Euclidian norm. Zoutendijk [37] and Powell [30] proved that with an exact line search the Fletcher-Reeves algorithm is global convergent on general functions. Al-Baali [1] established the global convergence of Fletcher-Reeves method for an approximate line search. A modification of the Fletcher and Reeves algorithm has been presented by Polak and Ribière [26] and Polyak [27]. Their method try to avoid the jamming behaviour of the Fletcher-Reeves algorithm, observed by Powell [29], and select β_k as:

$$\beta_k^{PR} = \frac{g_{k+1}^T y_k}{\|g_k\|^2}. \tag{6}$$

In case of jamming $g_{k+1} \cong g_k$, hence $y_k \cong 0$. Therefore $\beta_k^{PR} \cong 0$, and hence $d_{k+1} \cong -g_{k+1}$, i.e. the search direction is along the negative gradient. Therefore the Polak and Ribière algorithm has a restart feature, this making it more rapid convergent that Fletcher and Reeves's. On the other hand, when α_k in (2) is obtained by an exact line search, then the orthogonality condition $d_k^T g_{k+1} = 0$ gives $\beta_k^{HS} = \beta_k^{PR}$. Powell [30] showed that the Polak and Ribière algorithm, with α_k computed by an exact line search, can cycle or badly can be divergent on some functions. To cope with this possible failure of the Polak and Ribière algorithm, Gilbert and Nocedal [14] suggested the following scheme for β_k :

$$\beta_k^{PR+} = max\{\beta_k^{PB}, 0\}. \tag{7}$$

Therefore, when β_k given by (6) is negative, it is simply replaced by zero. The Gilbert and Nocedal scheme avoids both jamming and the possibility of convergence failure. For strongly convex quadratic functions, with an exact line search, it has the $n-$step convergence property. In this case, for every k, $\beta_k^{PR} > 0$. However, due to rounding errors,

[2] Magnus R. Hestens (1906-1991), Eduard Stiefel (1909-1978)

it is possible that $\beta_k^{PR} < 0,$ which implies $\beta_k^{PR+} = 0.$ Therefore, when β_k is seted to zero, the conjugate gradient algorithm is restarted with the negative gradient, leading to slower convergence.

Some other schemes for computing β_k have been given by Dai and Liao [9], and Hager and Zhang [16,17]. Dai and Liao suggest

$$\beta_k^{DL} = \frac{(y_k - ts_k)^T g_{k+1}}{d_k^T y_k}, \tag{8}$$

where $t > 0$ is a constant parameter and $s_k = x_{k+1} - x_k$. They report some numerical results for $t = 0.1$ and $t = 1.$ However, for different choices of t the performance of the corresponding conjugate gradient algorithm is very different. An adaptive version of Dai and Liao scheme is that given by Hager and Zhang:

$$\beta_k^{HZ} = \frac{1}{d_k^T y_k}\left(y_k - 2d_k \frac{\|y_k\|^2}{d_k^T y_k} \right)^T g_{k+1}, \tag{9}$$

which corresponds to $t = 2\|y_k\|^2 / (s_k^T y_k).$ Hager and Zhang proved that their conjugate gradient scheme satisfy the descent condition $g_k^T d_k \le -(7/8)\|g_k\|^2,$ being independent of the line search procedure, and is globally convergent when the line search satisfy the Wolfe conditions. The performance profile of their algorithm, implemented in CG-DESCENT package [17], shows that it is better than profiles for the L-BFGS quasi-Newton method of Nocedal [21] and Liu and Nocedal [20], the Polak-Ribière PR+ method [26], and schemes, of Dai and Yuan [10].

Another way of thinking the conjugate gradient methods has been considered by Perry [25], Oren and Spedicato [24] and Shanno [32]. They link conjugate gradient methods to quasi-Newton ideas, thus obtaining very powerfull conjugate gradient algorithms. Using the Hestens and Stiefel formula for updating β_k, Perry suggests a formula for computing the search direction d_{k+1} which satisfy a system of linear equations, similar but not identical, to the quasi-Newton equation. Shanno [32] reconsiders the method of Perry and interpret it as a memoryless BFGS updating formula. Using a scaling procedure suggested by Oren [22], Oren and Luenberger [23] and Oren and Spedicato [24], combined with Beale[3] restarts and Powell's restart criterion, Shanno presents an algorithm in which g_{k+1} is modified by a positive definite matrix which best estimates the inverse Hessian, without any additional storage requirement. At every nonrestart step a double update scheme is considered. Under an inexact line search Shanno [33] proved that this computational scheme is global convergent for convex functions. For general functions, Powell [28] proved that it may be not convergent, even that the exact line search is used. However, the Shanno's scheme is convergent if restarts are considered, but in this case the speed of convergence is reduced. Using the Wolfe line search, Han, Liu and Yin [18] proved that the Shanno's memoryless quasi-Newton method for nonconvex problems is convergent to a stationary point when $\lim_{k\to\infty} \|y_k\| = 0$ and the gradient of f is Lipschitz continuous. This computational scheme, using an inexact line search, implemented in CONMIN [34], proved to be one of the most powerfull conjugate gradient subroutine able to solve a large variety of unconstrained optimization problems.

This paper is motivated by a variant of the conjugate gradient algorithm, called spectral conjugate gradient, given by Birgin and Martinez [6]. Preserving the nice geometrical properties of Perry's direction, Birgin and Martinez present a conjugate gradient algorithm in which the parameter scaling the gradient defining the search direction is selected

[3] E.M.L. Beale (1928-1985)

by means of a spectral formula, firstly suggested by Barzilai and Borwein [4]. Numerical experiments with this algorithm proved that this computational scheme outperforms Polak-Ribière and Fletcher-Reeves, and is competitive with CONMIN of Shanno [34] and SGM of Raydan [31]. In this paper we modify the best algorithm of Birgin and Martinez in order to overcome the lack of positive definiteness of the matrix defining the search direction. This is done using the quasi-Newton BFGS updating philosophy. Using the restart technology of Beale-Powell, we get a new scaled conjugate gradient algorithm in which the scaling parameter is selected as spectral gradient or in an anticipative manner using the function values in two successive points. The algorithm implements both Wolfe conditions.

The paper is organized as follows: In section 2, following the developments of Birgin and Martrinez [6], we present the scaled conjugate gradient method. Section 3 is dedicated to the scaled conjugate gradient with memoryless BFGS updating formula. The scaled conjugate gradient with restart is presented in section 4. Section 5 presents two procedures for selection the scaling parameter: spectral and anticipative. A complete description of the scaled conjugate gradient algorithm is given in section 6. The algorithm performs two types of steps: a normal step in which a double quasi-Newton updating scheme is used, and a restart one where the current information is used to define the search direction. The convergence analysis of the algorithm, for strong convex functions, is described in section 7. In section 8 we present the computational results on a set of 500 unconstrained optimization problems and compare the Dolan and Moré [11] performance profiles of the new scaled conjugate gradient scheme to the profiles for the Birgin and Martinez's method.

2. Scaled conjugate gradient method

For solving (1) we consider the iterative process (2), where for $k = 0,1,\ldots$ the stepsize α_k is positive, and the directions d_k are generated by:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k, \tag{10}$$

in which θ_{k+1} and β_k are parameters which follow to be determined.

Observe that if $\theta_{k+1} = 1,$ then we get the classical conjugate gradient algorithms according to the value of the scalar parameter β_k. On the other hand, if $\beta_k = 0,$ then we get another class of algorithms according to the selection of the parameter θ_{k+1}. There are two possibilities for θ_{k+1} : a positive scalar or a positive definite matrix. If $\theta_{k+1} = 1$ we have the steepest descent (Cauchy [8]) algorithm. If $\theta_{k+1} = \nabla^2 f(x_{k+1})^{-1},$ or an approximation of it, then we get the Newton or the quasi-Newton algorithms, respectively. Therefore, we see that in general case, when $\theta_{k+1} \neq 0$ is selected in a quasi-Newton manner, and $\beta_k \neq 0,$ (10) represents a combination between the quasi-Newton and conjugate gradient methods.

Now, like in [6], assuming that f is a quadratic function and $B = \nabla^2 f(x_k),$ its Hessian, is positive definite, then this implies $y_k \neq 0$ for every $k = 0,1,\ldots$. Therefore, x^* satisfies:

$$x^* = x_{k+1} + d^*, \tag{11}$$

where

$$Bd^* = -g_{k+1}. \tag{12}$$

Now, premultiplying (12) by s_k^T, we get:

$$y_k^T d^* = -s_k^T g_{k+1}. \tag{13}$$

The relation (13) has the following geometric interpretation. The optimum increment d^* belongs to the following hyperplane:

$$P_k = \left\{ d \in R^n : y_k^T d = -s_k^T g_{k+1} \right\}. \tag{14}$$

Observe that if $s_k^T g_{k+1} = 0$, then the null direction $d = 0$ belongs to P_k.

Having in view this property of the quadratic functions with positive definite Hessian, it is natural to consider that

$$d_{k+1} \in P_k, \tag{15}$$

i.e. the search direction for the general problem (1) belongs to the hyperplane P_k for every k. Then by (10) we get:

$$y_k^T \left[-\theta_{k+1} g_{k+1} + \beta_k s_k \right] = -s_k^T g_{k+1},$$

i.e.

$$\beta_k = \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k}. \tag{16}$$

With this, the corresponding direction is as follows:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k} s_k. \tag{17}$$

The following particularizations can be remarked. If $\theta_{k+1} = 1,$ then (17) is the direction considered by Perry [25]. If $s_j^T g_{j+1} = 0, \ \ j = 0,1,\ldots,k,$ then from (17) we get:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{\theta_{k+1} y_k^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k} s_k, \tag{18}$$

which is the direction corresponding to a generalization of th Polak and Ribière formula. Of course, if $\theta_{k+1} = \theta_k = 1$ in (18), we get the classical Polak and Ribière formula. If $s_j^T g_{j+1} = 0, \ \ j = 0,1,\ldots,k,$ and additionally the successive gradients are orthogonal, then from (17)

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{\theta_{k+1} g_{k+1}^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k} s_k, \tag{19}$$

which is the direction corresponding to a generalization of the Fletcher and Reeves formula. We see that (17) is a general formula for direction computation in a conjugate gradient manner including the classical Fletcher and Reeves [13], and Polak and Ribière [26] formulas.

Computational experiments given by Birgin and Martinez, with a spectral gradient selection choice of parameter θ_{k+1}, show that the algorithm (17) of Perry outperforms the variants (18) of Polak and Ribière and (19) of Fletcher and Reeves, and compare favourable with CONMIN computational scheme of Shanno and Phua [34]. It is worth saying that the efficiency of (17) also is comming from a procedure for initial choice of the step-length in line searching algorithm implementing the Wolfe conditions, procedure we also consider in our developents.

3. Scaled conjugate gradient with memoryless BFGS updating

Shanno [32,33] proved that the conjugate gradient methods are exactly the BFGS quasi-Newton method where at every step the approximation to the inverse Hessian is restarted as the identity matrix. In this section we extend this result for the scaled conjugate gradient.

Before considering the developments of the scaled conjugate gradient algorithms with memoryless BFGS updating formula, let us present the following technical lemmas, which can be proved by direct computation.

Lemma 1. *If* $x, y, z \in R^n$ *then:* $(x^T y)z = (zy^T)x$. ■

Lemma 2. *If* $x, y, z \in R^n$ *then:* $x^T(yz^T) = (x^T y)z^T$. ■

From (17) we have:

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

Using lemma 1 we can write:

$$(y_k^T g_{k+1})s_k = (g_{k+1}^T y_k)s_k = (s_k y_k^T)g_{k+1},$$
$$(s_k^T g_{k+1})s_k = (g_{k+1}^T s_k)s_k = (s_k s_k^T)g_{k+1}.$$

Therefore, the direction given by (17) can be written as:

$$d_{k+1} = -\left[\theta_{k+1} I - \theta_{k+1}\frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}\right] g_{k+1} \equiv -Q_{k+1} g_{k+1}, \tag{20}$$

where

$$Q_{k+1} = \theta_{k+1} I - \theta_{k+1}\frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}. \tag{21}$$

If $\theta_{k+1} = 1,$ we have:

$$d_{k+1} = -\left[I - \frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}\right] g_{k+1}, \tag{22}$$

which is exactly the Perry formula. Using lemma 2, by direct computation, we can prove

Proposition 1.

$$y_k^T Q_{k+1} = s_k^T. \blacksquare \tag{23}$$

Observe that (23) is similar, but not identical, to the quasi-Newton equation, which requires that an update to the inverse Hessian H_{k+1} is in such a way as to satisfy:

$$H_{k+1} y_k = s_k. \tag{24}$$

A major difficulty with (20) is that the matrix Q_{k+1}, defined by (21), is not symmetric and hence not positive definite. Thus, the directions d_{k+1} from (20) are not necessarily descent directions and therefore numerical instability can result. Besides, another difficulty arising from this lack of symmetry is that the true quasi-Newton equation (24) is not satisfied.

In order to overcome this difficulty and to get a true quasi-Newton updating we first make the matrix Q_{k+1} from (21) symmetric as follows:

$$Q_{k+1} = \theta_{k+1} I - \theta_{k+1} \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}. \tag{25}$$

Now, we force Q_{k+1} to satisfy the quasi-Newton equation (24) yielding the following symmetric update:

$$Q_{k+1}^* = \theta_{k+1} I - \theta_{k+1} \frac{y_k s_k^T + s_k y_k^T}{y_k^T s_k} + \left[1 + \theta_{k+1} \frac{y_k^T y_k}{y_k^T s_k}\right] \frac{s_k s_k^T}{y_k^T s_k}. \tag{26}$$

By direct computation it is very easy to proof that Q_{k+1}^* satisfy the quasi-Newton equation, i.e.

Proposition 2.

$$Q_{k+1}^* y_k = s_k. \ \blacksquare \tag{27}$$

Notice that

$$d_{k+1} = -Q_{k+1}^* g_{k+1} \tag{28}$$

does not actually require the matrix Q_{k+1}^*, i.e. the direction d_{k+1} can be computed as:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1} \left(\frac{g_{k+1}^T s_k}{y_k^T s_k}\right) y_k - \left[\left(1 + \theta_{k+1} \frac{y_k^T y_k}{y_k^T s_k}\right) \frac{g_{k+1}^T s_k}{y_k^T s_k} - \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k}\right] s_k, \tag{29}$$

involving only 4 scalar products. Again observe that if $g_{k+1}^T s_k = 0,$ then (29) reduces to:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k. \tag{30}$$

Thus, in this case, the effect is simply one of multiplying the Hestens and Stiefel search direction by a positive scalar.

As we know, the BFGS update to the inverse Hessian, which currently is the best update of the Broyden class, is defined by:

$$H_{k+1} = H_k - \frac{H_k y_k s_k^T + s_k y_k^T H_k}{y_k^T s_k} + \left[1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right] \frac{s_k s_k^T}{y_k^T s_k}. \tag{31}$$

Therefore, we can immediately see that the conjugate gradient method (28), where Q_{k+1}^* is given by (26), is exactly the BFGS quasi-Newton method, where at every step the approximation of the inverse Hessian is restarted as the identity matrix multiplied by the scalar θ_{k+1}.

In order to ensure the convergence of the algorithm (2), with d_{k+1} given by (29), we need to constrain the choice of α_k. We consider line searches that satisfy the Wolfe conditions [35,36]:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \sigma_1 \alpha_k g_k^T d_k, \tag{32}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \ge \sigma_2 g_k^T d_k, \tag{33}$$

where $0 < \sigma_1 \le \sigma_2 < 1.$

Theorem 1. *Suppose that* α_k *in (2) satisfies the Wolfe conditions (32) and (33), then the direction* d_{k+1} *given by (29) is a descent direction.*

Proof: Since $d_0 = -g_0,$ we have $g_0^T d_0 = -\|g_0\|^2 \leq 0.$ Multiplying (29) by $g_{k+1}^T,$ we have

$$g_{k+1}^T d_{k+1} = \frac{1}{(y_k^T s_k)^2}\Big[-\theta_{k+1}\|g_{k+1}\|^2 (y_k^T s_k)^2 + 2\theta_{k+1}(g_{k+1}^T y_k)(g_{k+1}^T s_k)(y_k^T s_k) - (g_{k+1}^T s_k)^2 (y_k^T s_k) - \theta_{k+1}(y_k^T y_k)(g_{k+1}^T s_k)^2\Big].$$

Applying the inequality $u^T v \leq \frac{1}{2}(\|u\|^2 + \|v\|^2)$ to the second term of the right hand side of the above equality, with $u = (s_k^T y_k) g_{k+1}$ and $v = (g_{k+1}^T s_k) y_k$ we get:

$$g_{k+1}^T d_{k+1} \leq -\frac{(g_{k+1}^T s_k)^2}{y_k^T s_k}. \tag{34}$$

But, by Wolfe condition (33), $y_k^T s_k > 0.$ Therefore, $g_{k+1}^T d_{k+1} < 0$ for every $k = 0,1,\ldots$ ■

Observe that the second Wolfe condition (33) is crucial for the descent character of direction (29). More than this, we see that the estimation (34) is independent by the parameter θ_{k+1}.

4. Scaled conjugate gradient with restart

Usually, all conjugate gradient algorithms are periodically restarted. The standard restarting point occurs when the number of iterations is equal to the number of variables, but some other restarting methods can be considered. One such restarting method was proposed by Powell [29], based on an earlier version suggested by Beale [5]. The Powell restarting procedure is to test if there is very little orthogonality left between the curent gradient and the previous one. At step r when:

$$\left|g_{r+1}^T g_r\right| \geq 0.2\|g_{r+1}\|^2, \tag{35}$$

we restart the algorithm using the direction given by (29).

Another restarting procedure, considered by Birgin and Martinez [6], consists of testing if the angle between the current direction and $-g_{k+1}$ is not acute enough. Therefore, at step r when:

$$d_r^T g_{r+1} > -10^{-3}\|d_r\|_2 \|g_{r+1}\|_2, \tag{36}$$

the algorithm is restarted using the direction given by (29).

At step r when one of the two criteria (35) or (36) is satisfied, the direction is computed as in (29). For $k \geq r+1,$ we consider the same philosophy used to get (26), i.e. that of modifying the gradient g_{k+1} with a positive definite matrix which best estimates the inverse Hessian without any additional storage requirements. Therefore, the direction $d_{k+1},$ for $k \geq r+1,$ is computed using a double update scheme as:

$$d_{k+1} = -H_{k+1} g_{k+1}, \tag{37}$$

where

$$H_{k+1} = H_{r+1} - \frac{H_{r+1} y_k s_k^T + s_k y_k^T H_{r+1}}{y_k^T s_k} + \left[1 + \frac{y_k^T H_{r+1} y_k}{y_k^T s_k}\right] \frac{s_k s_k^T}{y_k^T s_k}. \tag{38}$$

and

$$H_{r+1} = \theta_{r+1} I - \theta_{r+1} \frac{y_r s_r^T + s_r y_r^T}{y_r^T s_r} + \left(1 + \theta_{r+1} \frac{y_r^T y_r}{y_r^T s_r}\right) \frac{s_r s_r^T}{y_r^T s_r}. \tag{39}$$

As above, observe that this computational scheme does not involve any matrix. Indeed, using lemma 1, $H_{r+1} g_{k+1}$ and $H_{r+1} y_k$ can be computed as:

$$\begin{aligned} v \equiv H_{r+1} g_{k+1} &= \theta_{r+1} g_{k+1} - \theta_{r+1} \left(\frac{g_{k+1}^T s_r}{y_r^T s_r}\right) y_r \\ &+ \left[\left(1 + \theta_{r+1} \frac{y_r^T y_r}{y_r^T s_r}\right) \frac{g_{k+1}^T s_r}{y_r^T s_r} - \theta_{r+1} \frac{g_{k+1}^T y_r}{y_r^T s_r}\right] s_r, \end{aligned} \tag{40}$$

and

$$\begin{aligned} w \equiv H_{r+1} y_k &= \theta_{r+1} y_k - \theta_{r+1} \left(\frac{y_k^T s_r}{y_r^T s_r}\right) y_r \\ &+ \left[\left(1 + \theta_{r+1} \frac{y_r^T y_r}{y_r^T s_r}\right) \frac{y_k^T s_r}{y_r^T s_r} - \theta_{r+1} \frac{y_k^T y_r}{y_r^T s_r}\right] s_r, \end{aligned} \tag{41}$$

involving 6 scalar products. With these the direction (37), at any nonrestart step can be computed as:

$$d_{k+1} = -v + \frac{(g_{k+1}^T s_k) w + (g_{k+1}^T w) s_k}{y_k^T s_k} - \left(1 + \frac{y_k^T w}{y_k^T s_k}\right) \frac{g_{k+1}^T s_k}{y_k^T s_k} s_k, \tag{42}$$

involving only 4 scalar products. We see that d_{k+1} from (42) is defined as a double quasi-Newton update scheme.

Before considering some other aspects of the algorithm concerning for example the selection of θ_{k+1} parameter and its convergence, it is useful to note that $y_k^T s_k > 0$ is sufficient to ensure that the direction d_{k+1} given by (42) is well defined and it is always a descent direction.

5. Selection of θ_{k+1}

In this section we shall consider some formulas for computation of θ_{k+1}. As we have already seen, in our algorithm θ_{k+1} is defined as a scalar approximation of the inverse Hessian. According to the procedures for a scalar estimation of the inverse Hessian we get a family of scaled conjugate gradient algorithms. The following procedures can be considered.

θ_{k+1} **spectral.** Motivated by the spectral gradient method introduced by Barzilai and Borwein [4] and analyzed by Raydan [31] and Fletcher [12], we consider a spectral gradient choice for θ_{k+1} as:

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k}. \tag{43}$$

The parameter θ_{k+1} given by (43) is the inverse of the Rayleigh quotient:

$$s_k^T \left[\int_0^1 \nabla^2 f(x_k + t s_k) dt \right] s_k \; / \; s_k^T s_k,$$

which lies between the largest and the smallest eigenvalue of the Hessian average

$$\int_0^1 \nabla^2 f(x_k + t s_k) dt.$$

Again we notice that $y_k^T s_k > 0$ is sufficient to ensure that θ_{k+1} in (43) is well defined.

θ_{k+1} **anticipative.** Recently, Andrei [2], using the information in two successive points of the iterative process, considered another scalar approximation of Hessian of function f obtaining a new algorithm which compares favourable with Barzilai-Borwein's. Indeed, in point $x_{k+1} = x_k + \alpha_k d_k$ we can write

$$f(x_{k+1}) = f(x_k) + \alpha_k g_k^T d_k + \frac{1}{2} \alpha_k^2 d_k^T \nabla^2 f(z) d_k,$$

where z is on the line segment connecting x_k and x_{k+1}. Having in view the local character of the searching procedure and that the distance between x_k and x_{k+1} is enough small we can choose $z = x_{k+1}$ and consider γ_{k+1} as a scalar approximation of the $\nabla^2 f(x_{k+1})$, where $\gamma_{k+1} \in R$. This is an anticipative view point, in which a scalar approximation of the Hessian at point x_{k+1} is computed using only the local information from two successive points: x_k and x_{k+1}. Therefore, we can write:

$$\gamma_{k+1} = \frac{2}{d_k^T d_k} \frac{1}{\alpha_k^2} \left[f(x_{k+1}) - f(x_k) - \alpha_k g_k^T d_k \right]. \tag{44}$$

Observe that for convex functions $\gamma_{k+1} > 0$. If $f(x_{k+1}) - f(x_k) - \alpha_k g_k^T d_k < 0$, then the reduction $f(x_{k+1}) - f(x_k)$ in function value is smaller than $\alpha_k g_k^T d_k$. In these cases the idea is to change a little the stepsize α_k as $\alpha_k - \eta_k$, maintaining the other quantities at their values, in such a manner that γ_{k+1} to be positive. To get a value for η_k let us select a real $\delta > 0$, "enough small", but comparable with the value of the function, and consider

$$\eta_k = \frac{1}{g_k^T d_k} \left[f(x_k) - f(x_{k+1}) + \alpha_k g_k^T d_k + \delta \right], \tag{45}$$

with which a new value for γ_{k+1} can be computed as:

$$\gamma_{k+1} = \frac{2}{d_k^T d_k} \frac{1}{(\alpha_k - \eta_k)^2} \left[f(x_{k+1}) - f(x_k) - (\alpha_k - \eta_k) g_k^T d_k \right]. \tag{46}$$

With these, the value for parameter θ_{k+1} is selected as:

$$\theta_{k+1} = \frac{1}{\gamma_{k+1}}, \tag{47}$$

where γ_{k+1} is given by (44) or (46).

Proposition 3. *Assume that* $f(x)$ *is continuously differentiable and* $\nabla f(x)$ *is Lipschitz continuous, with a positive constant* L. *Then at point* x_{k+1},

$$\gamma_{k+1} \le 2L. \tag{48}$$

Proof: From (44) we have:

$$\gamma_{k+1} = \frac{2\left[f(x_k) + \alpha_k \nabla f(\xi_k)^T d_k - f(x_k) - \alpha_k \nabla f(x_k)^T d_k\right]}{\|d_k\|^2 \alpha_k^2},$$

where ξ_k is on the line segment connecting x_k and x_{k+1}. Therefore

$$\gamma_{k+1} = \frac{2\left[\nabla f(\xi_k) - \nabla f(x_k)\right]^T d_k}{\|d_k\|^2 \alpha_k}.$$

Using the inequality of Cauchy and the Lipschitz continuity it follows that

$$\gamma_{k+1} \le \frac{2\|\nabla f(\xi_k) - \nabla f(x_k)\|}{\|d_k\|\alpha_k} \le \frac{2L\|\xi_k - x_k\|}{\|d_k\|\alpha_k} \le \frac{2L\|x_{k+1} - x_k\|}{\|d_k\|\alpha_k} = 2L. \ \blacksquare$$

Therefore, from (47) we get a lower bound for θ_{k+1} as:

$$\theta_{k+1} \ge \frac{1}{2L},$$

i.e. it is bounded away from zero.

6. The algorithm

Having in view the above developments and the definitions of g_k, s_k and y_k, as well as the selection procedures for θ_{k+1} computation, the following family of scaled conjugate gradient algorithms can be presented.

Algorithm SCALCG

Step 1. Select $x_0 \in R^n$, and the parameters $0 < \sigma_1 \le \sigma_2 < 1$. Compute $f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $\alpha_0 = 1/\|g_0\|$. Set $k = 0$.

Step 2. Line search. Compute α_k satisfying the Wolfe conditions (32) and (33). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1}), g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 3. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 4. Compute θ_k using a spectral (43) or an anticipative (47) approach.

Step 5. Compute the (restart) direction d_k as in (29).

Step 6. Line search. Compute the initial guess of the step-length as $\alpha_k = \alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe

conditions (32) and (33). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1}), g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 7. Store $\theta = \theta_k$, $s = s_k$ and $y = y_k$.

Step 8. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 9. If the Powell restart criterion (35), or the angle restart criterion (36), is satisfied, then go to step 4 (a restart step); otherwise continue with step 10 (a normal step).

Step 10. Compute

$$v = \theta g_k - \theta\left(\frac{g_k^T s}{y^T s}\right) y + \left[\left(1 + \theta \frac{y^T y}{y^T s}\right)\frac{g_k^T s}{y^T s} - \theta \frac{g_k^T y}{y^T s}\right] s,$$

$$w = \theta y_k - \theta\left(\frac{y_{k-1}^T s}{y^T s}\right) y + \left[\left(1 + \theta \frac{y^T y}{y^T s}\right)\frac{y_{k-1}^T s}{y^T s} - \theta \frac{y_{k-1}^T y}{y^T s}\right] s,$$

and

$$d_k = -v + \frac{(g_k^T s_{k-1})w + (g_k^T w)s_{k-1}}{y_{k-1}^T s_{k-1}} - \left(1 + \frac{y_{k-1}^T w}{y_{k-1}^T s_{k-1}}\right)\frac{g_k^T s_{k-1}}{y_{k-1}^T s_{k-1}} s_{k-1}.$$

Step 11. Line search. Compute the initial guess of the step-length as $\alpha_k = \alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions (32) and (33). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1}), g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 12. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$ and go to step 9. ■

It is well known that if f is bounded below along the direction d_k, then there exists a steplength α_k satisfying the Wolfe conditions. The initial selection of the step-length crucially affects the practical behavior of the algorithm. At every iteration $k \geq 1$ the starting guess for the step α_k in line search is computed as $\alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. This selection prove to be one of the best.

Concerning the stop criterion used in steps 3, 8 and 12 we consider the following tests:

$$\|g_k\|_\infty \leq \varepsilon_g \quad \text{or} \quad \frac{|f(x_{k+1}) - f(x_k)|}{1 + |f(x_k)|} \leq \varepsilon_f. \tag{49}$$

If (49) is satisfied, then the iterations are stoped.

7. Convergence analysis for strong convex functions

Throughout this section we assume that f is strongly convex and Lipschitz continuous on the level set

$$L_0 = \{x \in R^n : f(x) \leq f(x_0)\}. \tag{50}$$

That is, there exists constants $\mu > 0$ and L such that

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2 \tag{51}$$

and

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \tag{52}$$

for all x and y from L_0. For the convenience of the reader we include here the following lemma (see [16]).

Lemma 3. *Assume that d_k is a descent direction and ∇f satisfies the Lipschitz condition*

$$\|\nabla f(x)-\nabla f(x_k)\| \le L\|x-x_k\|, \tag{53}$$

for every x on the line segment connecting x_k and x_{k+1}, where L is a constant. If the line search satisfies the second Wolfe condition (33), then

$$\alpha_k \ge \frac{1-\sigma_2}{L}\frac{|g_k^T d_k|}{\|d_k\|^2}. \tag{54}$$

Proof: Subtracting $g_k^T d_k$ from both sides of (33) and using the Lipschitz condition we have

$$(\sigma_2-1)g_k^T d_k \le (g_{k+1}-g_k)^T d_k \le L\alpha_k\|d_k\|^2. \tag{55}$$

Since d_k is a descent direction and $\sigma_2<1$, (54) follows immediately from (55). ■

Lemma 4. *Assume that ∇f is strongly convex and Lipschitz continuous on L_0. If θ_{k+1} is selected by spectral gradient, then the direction d_{k+1} given by (29) satisfies:*

$$\|d_{k+1}\| \le \left(\frac{2}{\mu}+\frac{2L}{\mu^2}+\frac{L^2}{\mu^3}\right)\|g_{k+1}\|. \tag{56}$$

Proof: By Lipschitz continuity (52) we have

$$\|y_k\| = \|g_{k+1}-g_k\| = \|\nabla f(x_k+\alpha_k d_k)-\nabla f(x_k)\| \le L\alpha_k\|d_k\| = L\|s_k\|. \tag{57}$$

On the other hand, by strong convexity (51)

$$y_k^T s_k \ge \mu\|s_k\|^2. \tag{58}$$

Selecting θ_{k+1} as in (43), it follows that

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k} \le \frac{\|s_k\|^2}{\mu\|s_k\|^2} = \frac{1}{\mu}. \tag{59}$$

Now, using the triangle inequality and the above estimates (57)-(59), after some algebra on $\|d_{k+1}\|$, where d_{k+1} is given by (29), we get (56). ■

Lemma 5. *Assume that ∇f is strongly convex and Lipschitz continuous on L_0. If θ_{k+1} is selected by the anticipative procedure, then the direction d_{k+1} given by (29) satisfies:*

$$\left\|d_{k+1}\right\| \leq \left(\frac{1}{m}+\frac{2L}{m\mu}+\frac{1}{\mu}+\frac{L^2}{m\mu^2}\right)\left\|g_{k+1}\right\|. \tag{60}$$

Proof: By strong convexity on L_0, there exists the constant $m>0$, such that $\nabla^2 f(x) \geq mI,$ for all $x \in L_0.$ Therefore, for every k, $\gamma_{k+1} \geq m.$ Now, from (47) we see that, for all k,

$$\theta_{k+1} \leq \frac{1}{m}. \tag{61}$$

With this, like in lemma 4, we get (60). ■

The convergence of the scaled conjugate gradient algorithm (SCALCG) when f is strongly convex is given by

Theorem 2. *Assume that f is strongly convex and Lipschitz continuous on the level set L_0. If at every step of the conjugate gradient (2) with d_{k+1} given by (29) and the step length α_k selected to satisfy the Wolfe conditions (32) and (33), then either $g_k = 0$ for some k, or $\lim_{k\to\infty} g_k = 0.$*

Proof: Suppose $g_k \neq 0$ for all $k.$ By strong convexity we have

$$y_k^T d_k = (g_{k+1}-g_k)^T d_k \geq \mu\alpha_k \left\|d_k\right\|^2. \tag{62}$$

By theorem 1, $g_k^T d_k < 0.$ Therefore, the assumption $g_k \neq 0$ implies $d_k \neq 0.$ Since $\alpha_k > 0,$ from (62) it follows that $y_k^T d_k > 0.$ But f is strongly convex over $L_0,$ therefore f is bounded from below. Now, summing over k the first Wolfe condition (32) we have

$$\sum_{k=0}^{\infty} \alpha_k g_k^T d_k > -\infty.$$

Considering the lower bound for α_k given by (54) in lemma 3, and having in view that d_k is a descent direction, it follows that

$$\sum_{k=1}^{\infty} \frac{\left|g_k^T d_k\right|^2}{\left\|d_k\right\|^2} < \infty. \tag{63}$$

Now, from (34), using the inequality of Cauchy and (58) we get

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

Therefore, from (63) it follows that

$$\sum_{k=0}^{\infty} \frac{\left\|g_k\right\|^4}{\left\|d_k\right\|^2} < \infty. \tag{64}$$

Now, inserting the upperbound (56), or (60), for d_k in (64) yields

$$\sum_{k=0}^{\infty} \|g_k\|^2 < \infty,$$

which completes the proof. ■

For general functions the convergence of the algorithm is comming from theorem 1 and the restart procedure. However, our algorithm is very close to Perry/Shanno computational scheme. Therefore, for convex functions and under inexact line search it is global convergent, but in general, even when the line search is exact, it may be divergent. If restarts are employed, the algorithm is convergent, but the speed of converegence can decrease. On the other hand, the convergence of the proposed algorithm can be established using the analysis given by Shanno in [33].

8. Computational results and comparisons

In this section we present the performance of a Fortran implementation of the SCALCG algorithm on a number of test unconstrained optimization problems. At the same time, we compare the performance of SCALCG to the best spectral conjugate gradient algorithm, SCG (Perry-M1), of Birgin and Martinez. The test problems are the unconstrained problems in the CUTE [7] library, or MINPACK-2 [3] library, along with other large-scale optimization test problems.

In order to compare SCALCG with SCG we manufactured a new SCG code of Birgin and Martinez by introducing a sequence of code to compute the θ_{k+1} anticipative according to (47). Concerning the restart criterion, we implemented both the Powell and angle criterion. Both these criteria have a crucial role on the practical efficiency of the algorithms. However, for SCALCG the Powell criterion is more performant than the angle criterion. As the numerical experiments prove a large percentage of the SCALCG's iterations are restart iterations. Our numerical experiments show that the Powell restart criterion in SCG package is not profitable. Therefore we compare SCALCG algorithm with Powell restart criterion to SCG algorithm with angle restart criterion, each of them in two variants: θ_{k+1} spectral and θ_{k+1} anticipative, respectively.

The Wolfe conditions are implemented as in SCG with $\sigma_1 = 0.0001$ and $\sigma_2 = 0.9$. The initial guess of the step-length at the first iteration is $\alpha_0 = 1/\|g_0\|$. At the following iteration, both for SCALCG and SCG, the starting guess for the step α_k is computed as $\alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. This proved to be one of the best selection of the initial guess of the step length.

In all experiments we stopped the iterations whenever (49) is satisfied, where $\|.\|_\infty$ denotes the maximum absolute component of a vector and $\varepsilon_g = 10^{-6}$ and $\varepsilon_f = 10^{-12}$.

The numerical results concerning the number of iterations, number of restart iterations, number of function and gradient evaluations, cpu time in seconds, for each of the methods are posted at the following web site:

http://www.ici.ro/camo/neculai/ansoft.htm/scalcg

Table 1 shows the number of iterations and cpu time corresponding to SCALCG and SCG algorithms for θ_{k+1} spectral (θ^s) and anticipative (θ^a), for 50 test functions. For each test function we present the total number of iterations and total cpu time (seconds) for solving 10 problems with $n = 1000, 2000, \ldots, 10000$. For example, for Extended Freud. & Roth function we considered 10 numerical experiments in which $n = 1000, 2000, \ldots, 10000$. The total number of iterations and the total cpu time for solving these 10 problems, with SCALCG and SCG respectively, are give in the table.

Table 1. Cumulative results for test functions.

Function	θ	SCALCG		SCG	
		# iter	Time (s)	# iter	Time (s)
1) Ex. Freud. & Roth	θ^s	87	2.09	111	3.35
	θ^a	90	2.14	115	3.46
2) Ex. Trigonometric	θ^s	369	18.29	343	17.41
	θ^a	537	24.83	479	25.87
3) Ex. Rosenbrock	θ^s	248	7.85	280	10.55
	θ^a	265	8.24	272	10.55
4) Ex. White & Holst	θ^s	232	6.79	261	9.44
	θ^a	265	7.09	282	9.88
5) Ex. Beale	θ^s	100	1.87	134	3.19
	θ^a	117	2.08	138	3.30
6) Ex. Penalty	θ^s	118	4.51	111	5.00
	θ^a	110	4.95	115	5.05
7) Perturbed Quadratic	θ^s	5827	88.32	9249	225.25
	θ^a	5798	87.38	9928	246.95
8) Raydan 1	θ^s	2256	38.06	585	12.25
	θ^a	2234	37.68	619	13.40
9) Raydan 2	θ^s	30	0.88	40	1.15
	θ^a	30	0.88	40	1.21
10) Diagonal 1	θ^s	2043	35.70	663	15.99
	θ^a	2058	35.59	653	16.09
11) Diagonal 2	θ^s	3816	73.93	3496	98.97
	θ^a	3727	70.14	3593	102.71
12) Diagonal 3	θ^s	1896	46.47	459	13.24
	θ^a	1942	48.12	462	13.29
13) Hager	θ^s	412	10.93	183	6.04
	θ^a	409	10.54	177	5.88
14) Gen. Tridiagonal 1	θ^s	150	7.41	118	6.65
	θ^a	150	7.36	118	6.64
15) Ex. Tridiagonal 1	θ^s	87	2.42	78	3.07
	θ^a	83	2.31	131	4.34
16) Ex. Three Expo Term	θ^s	70	2.42	75	2.86
	θ^a	70	2.42	72	2.86

Table 1 (continued)

Function	θ	SCALCG		SCG	
		# iter	Time (s)	# iter	Time (s)
17) Gen. Tridiagonal 2	θ^s	456	17.08	548	25.38
	θ^a	466	16.97	534	24.71
18) Diagonal 4	θ^s	31	0.66	44	1.15
	θ^a	31	0.67	44	1.15
19) Diagonal 5	θ^s	30	2.20	30	1.93
	θ^a	30	2.25	30	1.92
20) Ex. Himmelblau	θ^s	60	1.09	70	1.76
	θ^a	59	1.11	87	2.14
21) Gen. PSC1	θ^s	1095	90.63	314	26.14
	θ^a	1155	102.82	367	31.20
22) Ex. PSC1	θ^s	60	4.72	60	4.51
	θ^a	50	4.07	58	4.34
23) Ex. Powell	θ^s	719	15.27	1386	34.17
	θ^a	1290	27.84	1339	38.23
24) Ex. BD1	θ^s	181	4.17	310	20.65
	θ^a	209	7.75	411	19.71
25) Ex. Maratos	θ^s	479	8.46	481	11.92
	θ^a	461	8.13	479	11.75
26) Ex. Cliff	θ^s	221	3.90	222	4.72
	θ^a	207	2.97	218	4.89
27) Quadratic Diag. Per.	θ^s	2046	32.95	15472	423.08
	θ^a	1899	31.92	16887	474.33
28) Ex. Wood	θ^s	495	16.64	991	38.84
	θ^a	637	19.99	829	34.00
29) Ex. Hiebert	θ^s	543	9.01	542	13.18
	θ^a	543	9.01	540	13.24
30) Quadratic QF1	θ^s	6322	54.54	9729	168.02
	θ^a	6228	52.34	9756	180.37
31) Ex. QP1	θ^s	60	2.90	55	1.75
	θ^a	59	2.58	61	1.82
32) Ex. QP2	θ^s	300	12.86	324	15.60
	θ^a	282	11.26	313	14.66
33) Quadratic QF2	θ^s	6502	177.35	10660	393.93
	θ^a	6626	178.89	10585	389.75

Table 1 (continued)

Function	θ	SCALCG		SCG	
		# iter	Time (s)	# iter	Time (s)
34) Ex. EP1	θ^s	19	0.44	20	0.38
	θ^a	19	0.49	20	0.50
35) Ex. Tridiagonal 2	θ^s	224	3.35	116	2.58
	θ^a	218	3.19	115	2.46
36) BDQRTIC	θ^s	1440	102.16	5739	554.26
	θ^a	1065	72.39	5429	514.49
37) TRIDIA	θ^s	20349	302.74	58034	1435.70
	θ^a	20270	299.18	60143	1532.26
38) ARWHEAD	θ^s	59	9.08	94	29.93
	θ^a	49	6.32	187	156.15
39) NONDIA	θ^s	69	3.35	86	4.18
	θ^a	68	3.24	86	4.13
40) NONDQUAR	θ^s	20236	785.10	53994	10588.97
	θ^a	22049	898.14	56098	2674.75
41) DQDRTIC	θ^s	72	3.02	101	4.50
	θ^a	72	2.96	101	4.50
42) EG2	θ^s	984	23.79	3170	110.24
	θ^a	1015	26.86	1296	51.41
43) DIXMAANA	θ^s	53	5.93	70	7.25
	θ^a	53	5.93	80	8.24
44) DIXMAANB	θ^s	100	8.95	111	10.16
	θ^a	100	9.01	110	9.94
45) DIXMAANC	θ^s	134	11.86	147	13.40
	θ^a	130	11.31	148	13.40
46) DIXMAANE	θ^s	3840	296.66	5483	482.41
	θ^a	3985	313.63	5240	461.48
47) Partial Pert. Quadratic	θ^s	551	225.91	2759	1878.56
	θ^a	557	233.76	2732	1933.44
48) Broyden Tridiag.	θ^s	608	5.01	678	10.99
	θ^a	544	4.50	400	5.99
49) Almost Pert. Quad.	θ^s	5889	50.60	9719	172.08
	θ^a	5750	48.34	9620	171.97
50) Tridiag. Pert. Quad.	θ^s	5878	88.92	9493	233.21
	θ^a	6139	91.28	9800	243.60

Table 2 shows the global characteristics, corresponding to these 500 test problems, refering to the total number of iterations and total cpu time for SCALCG and SCG with spectral and anticipative variants of parameter θ_{k+1} selection, respectively. Considering the number of iterations we see that, for theta spectral, SCALCG is about 2.11 times more performant than SCG and about 6.28 times fastest. On the other hand, for theta anticipative, SCALCG is about 2.1 times more performant, and 3.32 times fastest than SCG.

Table 2. Global characteristics of SCALCG and SCG

Global characteristics	θ	**SCALCG**	**SCG**
Total number of iterations. 500 problems	θ^s	97846	207238
	θ^a	100200	211337
Total cpu time (seconds). 500 problems	θ^s	2729.24	17159.94
	θ^a	2862.85	9508.40

Table 3 shows the number of problems, out of 500, for which SCALCG and SCG algorithms achieved the minimum number of iterations and the minimum cpu time, respectively. We see that SCALCG is more performant than SCG, both in number of iterations and cpu time. Refering to the number of iterations SCALCG is about 2.14 times more performant than SCG. Considering cpu time, SCALCG is 3.16 fastest than SCG. On the other hand, we see that both spectral and anticipative variants of the algorithms have almost the same performances. In particular, for spectral variant SCALCG achieved the minimum number of iterations for about 75.4% of the test problems, and 73.8% for the anticipative variant. Considering the cpu time, for spectral variant SCALCG achieved the minimum cpu time for about 80.4% of the test problems, and 79.8% for the anticipative variant. Clearly, SCG is dominated by SCALCG, both in number of iterations and time.

Table 3. Performance of SCALCG and SCG

Performance criterion	**# of problems**	
	θ^s	θ^a
SCALCG achieved minimum # of iterations in	377	369
SCG achieved minimum # of iterations in	176	172
SCALCG and SCG achieved the same # of iterations in	53	41
SCALCG achieved minimum cpu time in	402	399
SCG achieved minimum cpu time in	127	130
SCALCG and SCG achieved the same cpu time in	29	29

The performance of these two algorithms, both for spectral and anticipative variants, have been evaluated using the profiles of Dolan and Moré [11]. That is, for each algorithm, we plot the fraction of problems for which the algorithm is within a factor of the best number of iterations and cpu time, respectively. In figures 1-4 we compare the performance of SCALCG and SCG codes refering to the number of iterations and cpu time, both for spectral and anticipative variants. The left side of these figures gives the percentage of the test problems, out of 500, for which an algorithm is more performant; the right side gives the percentage of the test problems that were successfully solved by each of the algorithms. The top curve corresponds to the algorithm that solved the most problems in a number of iterations (fig. 1, fig.3) or in a cpu time (fig. 2, fig. 4) that was within a given factor of the best number of iterations and time, respectively.

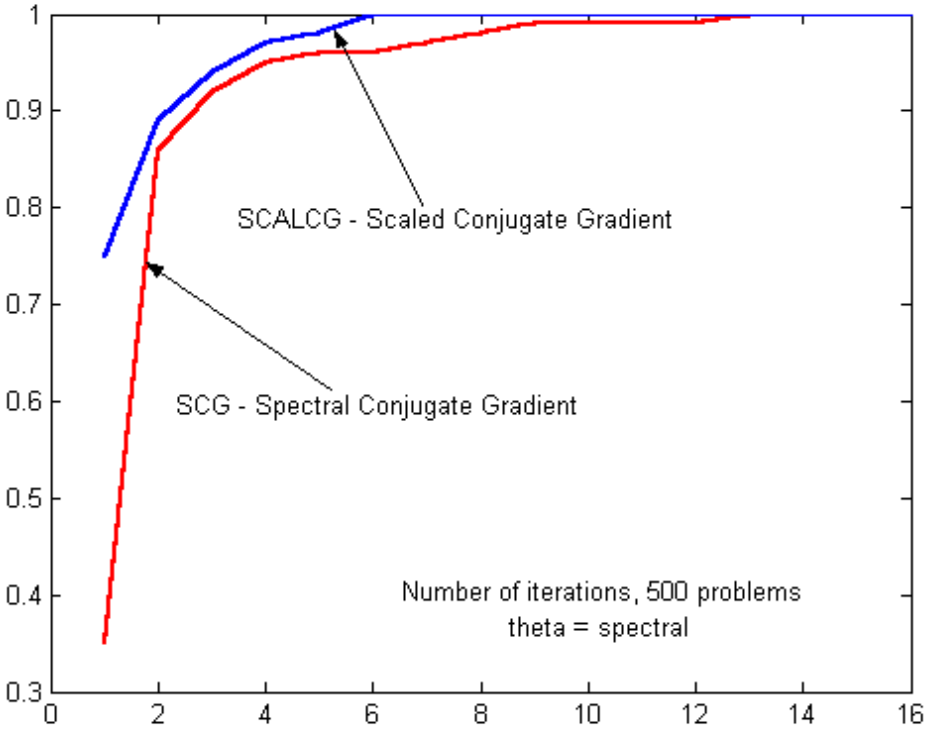

Fig. 1. Performance profiles. Number of iterations. Theta = spectral.

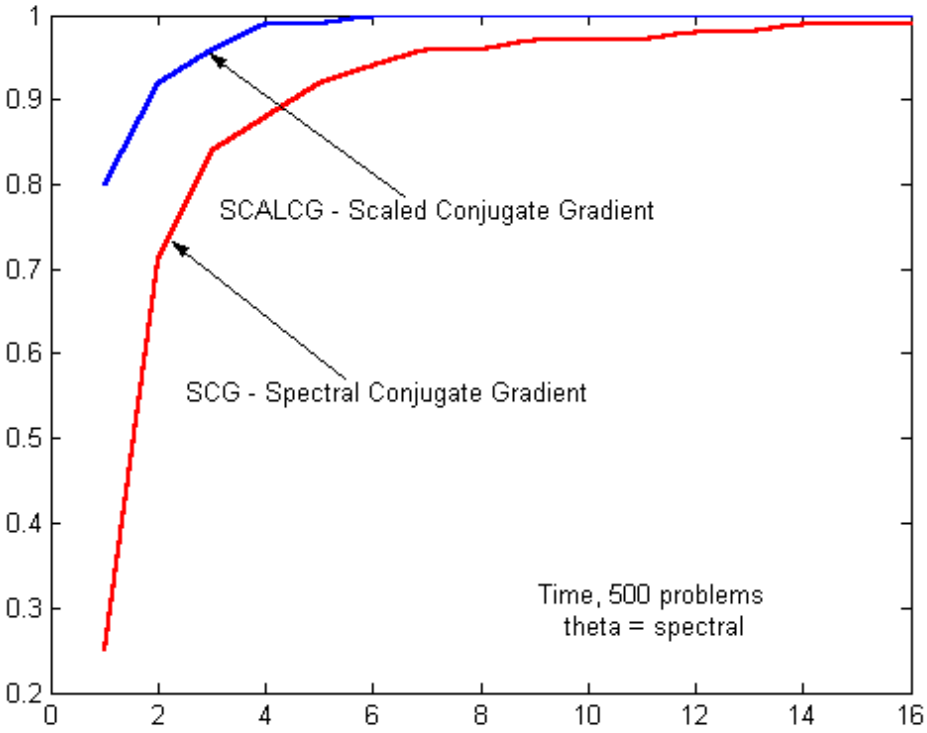

Fig. 2. Performance profiles. Time. Theta = spectral

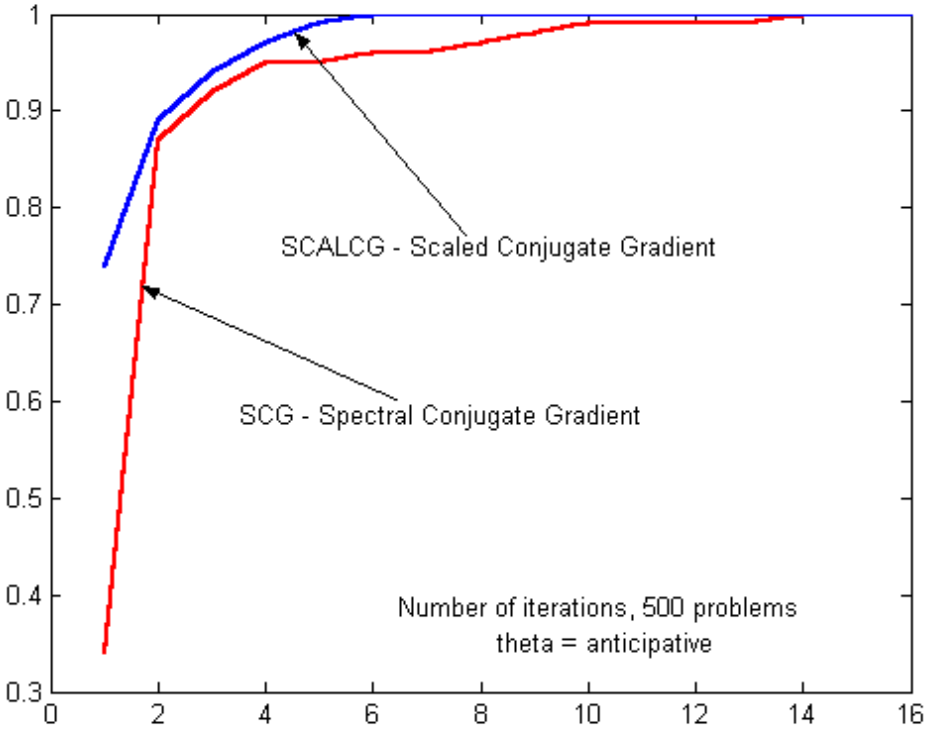

Fig. 3. Performance profiles. Number of iterations. Theta = anticipative.

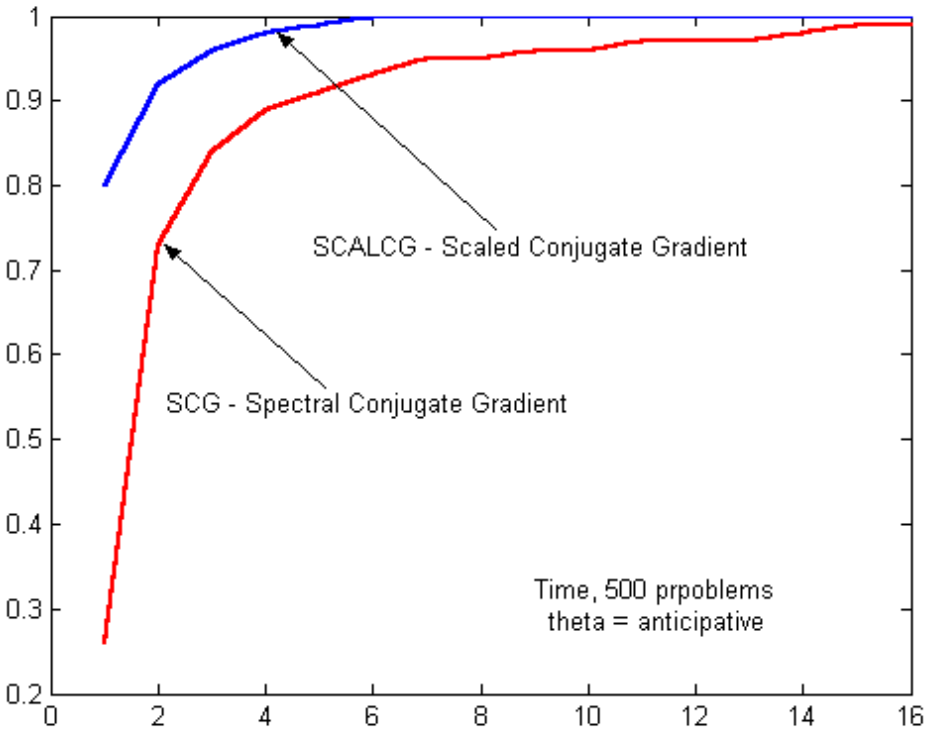

Fig. 4. Performance profiles. Time. Theta = anticipative.

Since the top curves in these figures correspond to SCALCG, this algorithm is clearly more performant both from view points of the number of iterations and cpu time, and both in spectral and anticipative variants, for this set of 500 test problems with dimensions ranging from 1000 to 10,000.

It is worth saying that both SCALCG and SCG conjugate gradient algorithms with θ^s are very close tothose using θ^a. Refering to the number of iterations, from table 2 we see that, for this set of 500 test problems, SCALCG with θ^s is about 1.02 times more performant than SCALCG with θ^a. Concerning the cpu time SCALCG with θ^s is about 1.04 times fastest than SCALCG variant with θ^a. Therefore, θ^a approach compares favourable with θ^s. Besides, the numerical experiments show that the number of problems, out 500, for which $\gamma_{k+1} < 0$ is only 58, representing 11.6%. More than this, out of a total of 100200 iterations, for solving 500 problems, only 154 iterations involved $\gamma_{k+1} < 0$, i.e. 0.153%. The salient difference between SCALCG and SCG algorithms is that while the percentage of restart iterations for SCALCG is about 90%, SCG involves maximum 10%.

Performances on MINPACK-2 test problem collection. In table 4, we present the performance of SCALCG algorithm for some unconstrained optimization problems from MINPACK-2 collection [3].

Table 4. Performance of SCALCG for MINPACK-2 test problems

Problem / Parameters	n	# iterations		Time (s)		$f(x^*)$
		θ^s	θ^a	θ^s	θ^a	
Elastic-Plastic Torsion, $nx = 100,\ ny = 100,$ $c = 5.$	10000	252	203	15.66	12.30	-0.439163197
Pressure Distribution in a Journal Bearing, $nx = 100,\ ny = 100,$ $ecc = 0.1,\ b = 10.$	10000	542	490	34.05	30.38	-0.28284
Optimal Design with Composite Materials, $nx = 100,\ ny = 100,$ lambda = 0.008	10000	512	581	39.77	44.65	-0.01137724
Inhomogeneous Superconductors (1-dimensional Ginzburg-Landau problem), $t = 7.$	1000	7422	5560	134.02	102.54	.920843E-06
Leonard-Jones Cluster Problem, ndim=3, natoms=1000	3000	1538	1469	358.50	333.07	-6658.1153
Steady State Combustion (Solid fuel ignition), $nx = 100,\ ny = 100,$ lambda = 0.07	10000	330	278	37.90	31.20	-0.07008636

9. Conclusion

The best algorithm of Birgin and Martinez, which mainly is a scaled variant of Perry's, was modified in order to overcome the lack of positive definiteness of the matrix defining the search direction. This modification takes the advantage of the quasi-Newton BFGS updating formula. Using the restart technology of Beale-Powell, we get a scaled conjugate gradient algorithm in which the parameter scaling the gradient is selected as spectral gradient or in an anticipative manner by means of a formula using the function values in two successive points. Although the update formulas (29) and (40)-(42) are more complicated the scheme proved to be efficient and robust in numerical experiments. The algorithm implements the Wolfe conditions, and we prove that the steps are along the descent directions.
The performance profiles for our scaled conjugate gradient algorithm was higher than those of spectral conjugate gradient method of Birgin and Martinez for a test set consisting of 500 unconstrained optimization problems.

References

1. M. Al-Baali, "*Descent property and global convergence of the Fletcher-Reeves method with inexact line search*", IMA J. Numer. Anal., 5, pp.121-124, 1985
2. N. Andrei, "*A new gradient descent method for unconstrained optimization*", ICI Technical Report, March 2004.
3. B.M.Averick, R.G. Carter, J.J. Moré and G.L. Xue, "*The MINPACK-2 test problem collection*", Argonne National Laboratory, Preprint MCS-P153-0692, June 1992.
4. J. Barzilai and J.M. Borwein, "*Two point step size gradient method*", IMA J. Numer. Anal., 8, pp.141-148, 1988.
5. E.M.L. Beale, "*A derivation of conjugate gradients*", in: F.A. Lootsma, ed., *Numerical Methods for Nonlinear Optimization*, Academic Press, London, pp.39-43, 1972.
6. E. Birgin and J.M. Martinez, "*A spectral conjugate gradient method for unconstrained optimization*", Applied Math. and Optimization, 43, pp.117-128, 2001
7. I. Bongartz, A.R. Conn, N.I.M. Gould and P.L. Toint, "*CUTE: constrained and unconstrained testing environments*", ACM Trans. Math. Software, 21, pp.123-160, 1995.
8. A. Cauchy, "*Méthodes générales pour la résolution des systèmes déquations simultanées*", C.R. Acad. Sci. Par., 25, pp.536-538, 1847.
9. Y.H. Dai and L.Z. Liao, "*New conjugate conditions and related nonlinear conjugate gradient methods*", Appl. Math. Optim., vol. 43 pp.87-101, 2001.
10. Y.H. Dai and Y. Yuan, "*A nonlinear conjugate gradient method with a strong global convergence property*", SIAM J. Optim., 10, pp.177-182, 1999.
11. E.D. Dolan and J.J.Moré, "*Benchmarking optimization software with performance profiles*", Math. Programming, 91, pp. 201-213, 2002.
12. R. Fletcher, "*On the Barzilai-Borwein method*", Numerical Analysis Report NA/207, 2001.
13. R. Fletcher and C.M. Reeves, "*Function minimization by conjugate gradients*", Comput. J. 7, pp. 149-154, 1964.
14. J.C. Gilbert and J. Nocedal, "*Global convergence properties of conjugate gradient methods for optimization*", SIAM J. Optim., 2, pp.21-42, 1992.
15. G.H. Golub and D.P. O'Leary, "*Some history of the conjugate gradient and Lanczos algorithms: 1948-1976*", SIAM Rev., 31 pp.50-102, 1989.
16. W.W. Hager and H. Zhang, "*A new conjugate gradient method with guaranteed descent and an efficient line search*", University of Florida, Department of Mathematics, November 17, 2003 (theory and comparisons), revised July 3, 2004.

17. W.W. Hager and H. Zhang, "*CG-DESCENT, A conjugate gradient method with guaranteed descent (algorithm details and comparisons)*", University of Florida, Department of Mathematics, January 15, 2004.
18. J.Y. Han, G.H. Liu and H.X. Yin, "*Convergence of Perry and Shanno's memoryless quasi-Newton method for nonconvex optimization problems*", OR Trans., 1, pp.22-28, 1997.
19. M.R. Hestens and E. Stiefel, "*Methods of conjugate gradients for solving linear systems*", J. Research Nat. Bur. Standards Sec. B. 48, pp. 409-436, 1952.
20. D.C. Liu and J. Nocedal, *"On the limited memory BFGS method for large scale optimization*", Math. Programming, 45, pp.503-528, 1989.
21. J. Nocedal, "*Updating quasi-Newton matrices with limited storage*", Math. Comp., 35, pp.773-782, 1980.
22. S.S. Oren, "*Self-scaling variable metric algorithm. Part II*", Management Sci., 20, pp.863-874, 1974.
23. S.S. Oren and D.G. Luenberger, "*Self-scaling variable metric algorithm. Part I*", Management Sci., 20, pp.845-862, 1976.
24. S.S. Oren and E. Spedicato, "*Optimal conditioning of self-scaling variable metric algorithms*", Math. Programming, 10, pp.70-90, 1976.
25. J.M. Perry, "*A class of conjugate gradient algorithms with a two step variable metric memory*", Discussion paper 269, Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1977.
26. E. Polak and G. Ribière, "*Note sur la convergence de methods de directions conjugres*", Revue Francaise Informat. Reserche Opérationnelle 16, pp. 35-43, 1969.
27. B.T. Polyak, "*The conjugate gradient method in extreme problems*", USSR Comp. Math. Math. Phys., 9, pp.94-112, 1969.
28. M.J.D. Powell, "*Some convergence properties of the conjugate gradient method*". Math. Programming, 11, pp.42-49, 1976.
29. M.J.D. Powell, "*Restart procedures for the conjugate gradient method*", Math. Programming, 12, pp.241-254, 1977.
30. M.J.D. Powell, "*Nonconvex minimization calculations and the conjugate gradient method*", in: Lecture Notes in Mathematics vol. 1066, Berlin: Springer-Verlag, pp.122-141, 1984.
31. M. Raydan, "*The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*", SIAM J. Optim., 7, 26-33, 1997.
32. D.F. Shanno, "*Conjugate gradient methods with inexact searches*", Mathematics of Operations Research, vol. 3, pp.244-256, 1978.
33. D.F. Shanno, "*On the convergence of a new conjugate gradient algorithm*", SIAM J. Numer. Anal. vol. 15, pp.1247-1257, 1978.
34. D.F. Shanno and K.H. Phua, *"Algorithm 500, Minimization of unconstrained multivariate functions [E4]*", ACM Trans. on Math. Soft., 2, pp.87-94, 1976.
35. P. Wolfe, "*Convergence conditions for ascent methods*", SIAM Rev., 11, pp.226-235, 1969.
36. P. Wolfe, "*Convergence conditions for ascent methods II: some corrections*", SIAM Rev. 13, pp.185-188, 1971.
37. G. Zoutendijk, "*Nonlinear programming, computational methods*", in: *Integer and Nonlinear Programming*, J. Abadie, ed., Amsterdam: North-Holland, pp.37-86, 1970.

November 29, 2004