Optimization
A Journal of Mathematical Programming and Operations Research
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713646500

A scaled nonlinear conjugate gradient algorithm for unconstrained optimization

Neculai Andrei [a]
[a] Research Institute for Informatics, Center for Advanced Modeling and Optimization, Bucharest 1, Romania

First Published: January 2008

To cite this Article: Andrei, Neculai (2008) 'A scaled nonlinear conjugate gradient algorithm for unconstrained optimization', Optimization, 57:4, 549 — 570

To link to this article: DOI: 10.1080/02331930601127909
URL: http://dx.doi.org/10.1080/02331930601127909

Optimization
Vol. 57, No. 4, August 2008, 549–570

A scaled nonlinear conjugate gradient algorithm for unconstrained optimization

Neculai Andrei*

Research Institute for Informatics, Center for Advanced Modeling and Optimization, 8-10, Averescu Avenue, Bucharest 1, Romania

(*Received 29 May 2005; final version received 19 July 2006*)

The best spectral conjugate gradient algorithm by (Birgin, E. and Martínez, J.M., 2001, A spectral conjugate gradient method for unconstrained optimization. *Applied Mathematics and Optimization*, **43**, 117–128). which is mainly a scaled variant of (Perry, J.M., 1977, A class of Conjugate gradient algorithms with a two step varaiable metric memory, Discussion Paper 269, Center for Mathematical Studies in Economics and Management Science, Northwestern University), is modified in such a way as 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 embedded into the restart philosophy of Beale–Powell. The parameter scaling the gradient is selected as spectral gradient or in an anticipative way 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 set consisting of 700 unconstrained optimization problems show that this new scaled nonlinear conjugate gradient algorithm substantially outperforms known conjugate gradient methods including: the spectral conjugate gradient SCG by Birgin and Martínez, the scaled Fletcher and Reeves, the Polak and Ribière algorithms and the CONMIN by (Shanno, D.F. and Phua, K.H., 1976, Algorithm 500, Minimization of unconstrained multivariate functions. *ACM Transactions on Mathematical Software*, **2**, 87–94).

Keywords: unconstrained optimization; conjugate gradient method; spectral gradient method; BFGS formula; numerical comparisons

2000 Mathematics Subject Classifications: 49M07; 49M10; 90C06; 65K

1. Introduction

In a recent article, Birgin and Martínez [3] introduced a spectral conjugate gradient method for solving large-scale unconstrained optimization problems

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

*Email: nandrei@ici.ro

ISSN 0233–1934 print/ISSN 1029–4945 online

DOI: 10.1080/02331930601127909
http://www.informaworld.com

where $f: R^n \to R$ is continuously differentiable and its gradient is available. This method generates a sequence x_k of approximations to the minimum x^* of f, in which

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k, \tag{3}$$

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 $d_0 = -g_0$.

A geometric interpretation of the quadratic function minimization led Birgin and Martínez [3] to the following expression for parameter β_k:

$$\beta_k = \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k}, \tag{4}$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. Motivated by the spectral gradient method introduced by Barzilai and Borwein [2] and analysed by Raydan [20] and Fletcher [10], Birgin and Martínez consider a spectral gradient choice for the scaling factor θ_{k+1} as:

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k}. \tag{5}$$

Observe that the parameter θ_{k+1} given by (5) is the inverse of the Rayleigh quotient. Numerical experiments with the algorithm (2)–(5), implementing the Wolfe line search in a restart environment and using a special procedure for the initial choice of the step-length on a set of 40 test problems, proved that this computational scheme outperforms the classical Polak–Ribière and Fletcher–Reeves methods and is competitive with the CONMIN of Shanno and Phua [23] and the SGM of Raydan [20].

In this article, we modify the best algorithm by Birgin and Martínez [3] 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, thus obtaining a new descent direction. The basic idea of the proposed algorithm is to combine the scaled memoryless BFGS method and the preconditioning technique. The preconditioned, which is also a scaled memoryless BFGS matrix, is reset when a restart criterion holds. Therefore, we get *a preconditioned and scaled memoryless BFGS algorithm*. The algorithm implements the Wolfe line search conditions.

The article is organized as follows: in Section 2, we present the scaled conjugate gradient method with restart. A complete description of the scaled conjugate gradient algorithm, SCALCG, is given in Section 3. The algorithm performs two types of steps: a standard 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 strongly convex functions is described in Section 4. In Section 5, we present the computational results on a set of 700 unconstrained optimization problems and compare the Dolan and Moré [9] performance profiles of the new scaled conjugate gradient scheme SCALCG to the profiles for the Birgin and Martínez's method SCG, the scaled Polak–Ribière–Polyak, the scaled Fletcher–Reeves and the CONMIN of Shanno and Phua.

2. Scaled conjugate gradient method with restart

For solving (1) we consider the iterative process (2), where for $k = 0, 1, \ldots$, the step size α_k is positive, and the directions d_k are generated by (3), in which θ_{k+1} and β_k are parameters. 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 $\theta_{k+1} = 1$, then we get the steepest descent algorithm (Cauchy [5]). 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 when $\theta_{k+1} \neq 0$ is selected in a quasi-Newton way and $\beta_k \neq 0$, then (3) represents a combination between the quasi-Newton and the conjugate gradient methods.

The direction corresponding to β_k given in (4) is as follows:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k} s_k. \tag{6}$$

The following particularizations can be remarked. If $\theta_{k+1} = 1$, then (6) is the direction considered by Perry [17]. At the same time we see that (6) is the direction given by Dai and Liao [6] for $t = 1$, obtained through an interpretation of the conjugacy condition. Additionally, if $s_j^T g_{j+1} = 0$, $j = 0, 1, \ldots, k$, then from (6) we get:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{\theta_{k+1} y_k^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k} s_k, \tag{7}$$

which is the direction corresponding to a generalization of the Polak and Ribière formula. Obviously, if $\theta_{k+1} = \theta_k = 1$ in (7), we get the classical Polak and Ribière formula [18]. If $s_j^T g_{j+1} = 0$, $j = 0, 1, \ldots, k$, and additionally the successive gradients are orthogonal, then from (6)

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{\theta_{k+1} g_{k+1}^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k} s_k, \tag{8}$$

which is the direction corresponding to a generalization of the Fletcher and Reeves formula. Therefore, (6) is a general formula for the direction computation in a conjugate gradient manner including the classical Fletcher and Reeves [11] and the Polak and Ribière [18] formulas.

Shanno [21,22] 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. Now we extend this result for the scaled conjugate gradient. We see that the direction given by (6) can be written as:

$$d_{k+1} = -\left[\theta_{k+1} I - \theta_{k+1} \frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}\right] g_{k+1} \equiv -Q_{k+1} g_{k+1}, \tag{9}$$

where

$$Q_{k+1} = \theta_{k+1} I - \theta_{k+1} \frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}. \tag{10}$$

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

$$d_{k+1} = -\left[I - \frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}\right] g_{k+1}, \tag{11}$$

which is exactly the Perry formula. By direct computation we can prove

PROPOSITION 1

$$y_k^T Q_{k+1} = s_k^T. \tag{12}$$

Observe that (12) is similar but not identical to the quasi-Newton equation, which requires an update to the inverse Hessian H_{k+1} as to satisfy:

$$H_{k+1} y_k = s_k. \tag{13}$$

A major difficulty with (9) is that the matrix Q_{k+1}, defined by (10) is not symmetric and therefore not positive definite. Thus, the direction d_{k+1} from (9) is not necessarily a descent one and so numerical instability can appear. Besides, another difficulty arising from this lack of symmetry is that the true quasi-Newton Equation (13) 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 (10) symmetric as follows:

$$Q_{k+1} = \theta_{k+1} I - \theta_{k+1} \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k}. \tag{14}$$

Now, we force Q_{k+1} to satisfy the quasi-Newton Equation (13) yielding the following symmetric update:

$$Q_{k+1}^* = \theta_{k+1} I - \theta_{k+1} \frac{y_k s_k^T + s_k y_k^T}{y_k^T s_k} + \left[1 + \theta_{k+1} \frac{y_k^T y_k}{y_k^T s_k}\right] \frac{s_k s_k^T}{y_k^T s_k}. \tag{15}$$

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

PROPOSITION 2

$$Q_{k+1}^* y_k = s_k. \tag{16}$$

Notice that

$$d_{k+1} = -Q_{k+1}^* g_{k+1} \tag{17}$$

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

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1} \left(\frac{g_{k+1}^T s_k}{y_k^T s_k}\right) y_k - \left[\left(1 + \theta_{k+1} \frac{y_k^T y_k}{y_k^T s_k}\right) \frac{g_{k+1}^T s_k}{y_k^T s_k} - \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k}\right] s_k. \tag{18}$$

Again observe that if $g_{k+1}^T s_k = 0$, then (18) is reduced to:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k. \tag{19}$$

Thus, the effect is simply one of multiplying the Hestenes and Stiefel [13] 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:

$$H_{k+1} = H_k - \frac{H_k y_k s_k^T + s_k y_k^T H_k}{y_k^T s_k} + \left[1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right] \frac{s_k s_k^T}{y_k^T s_k}. \tag{20}$$

We can see that the conjugate gradient method (17), where Q_{k+1}^* is given by (15), 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 (18), we need to constrain the choice of α_k. We consider the line searches that satisfy the Wolfe conditions [24,25]:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \sigma_1 \alpha_k g_k^T d_k, \tag{21}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \ge \sigma_2 g_k^T d_k, \tag{22}$$

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

THEOREM 1 *Suppose that α_k in* (2) *satisfies the Wolfe conditions* (21) *and* (22), *then the direction d_{k+1} given by* (18) *is a descent direction.*

Proof Since $d_0 = -g_0$, we have $g_0^T d_0 = -\|g_0\|^2 \le 0$. Multiplying (18) 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 $u^T v \le (1/2)(\|u\|^2 + \|v\|^2)$ 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:

$$g_{k+1}^T d_{k+1} \le -\frac{(g_{k+1}^T s_k)^2}{y_k^T s_k}. \tag{23}$$

But, by Wolfe condition (22), $y_k^T s_k > 0$. Therefore, $g_{k+1}^T d_{k+1} < 0$ for every $k = 0, 1, \ldots$ ■

Observe that the second Wolfe condition (22) is crucial for the descent character of direction (18). Moreover, we see that the estimation (23) is independent of the parameter θ_{k+1}.

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 as well. The Powell restarting procedure [19] is to test if there is very little orthogonality left between the current gradient and the previous one. At step r when:

$$|g_{r+1}^T g_r| \ge 0.2\|g_{r+1}\|^2, \tag{24}$$

we restart the algorithm using the direction given by (18). The convergence analysis with this restart criterion can be found in [7]. Another restarting procedure considered by Birgin

and Martínez [3], consists of testing if the angle between the current direction and $-g_{k+1}$ is not very acute. Therefore, at step r when:

$$d_r^T g_{r+1} > -10^{-3}\|d_r\|_2\|g_{r+1}\|_2 \tag{25}$$

the algorithm is restarted using the direction given by (18).

At step r, when one of the two criteria (24) or (25) is satisfied, the direction is computed as in (18). For $k \geq r+1$, we follow the same philosophy used to get (15), 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:

$$d_{k+1} = -H_{k+1}g_{k+1}, \tag{26}$$

where

$$H_{k+1} = H_{r+1} - \frac{H_{r+1}y_k s_k^T + s_k y_k^T H_{r+1}}{y_k^T s_k} + \left[1 + \frac{y_k^T H_{r+1} y_k}{y_k^T s_k}\right]\frac{s_k s_k^T}{y_k^T s_k}. \tag{27}$$

and

$$H_{r+1} = \theta_{r+1}I - \theta_{r+1}\frac{y_r s_r^T + s_r y_r^T}{y_r^T s_r} + \left(1 + \theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{s_r s_r^T}{y_r^T s_r}. \tag{28}$$

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

$$\begin{aligned} v \equiv H_{r+1}g_{k+1} &= \theta_{r+1}g_{k+1} - \theta_{r+1}\left(\frac{g_{k+1}^T s_r}{y_r^T s_r}\right)y_r \\ &+ \left[\left(1 + \theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{g_{k+1}^T s_r}{y_r^T s_r} - \theta_{r+1}\frac{g_{k+1}^T y_r}{y_r^T s_r}\right]s_r, \end{aligned} \tag{29}$$

and

$$\begin{aligned} w \equiv H_{r+1}y_k &= \theta_{r+1}y_k - \theta_{r+1}\left(\frac{y_k^T s_r}{y_r^T s_r}\right)y_r \\ &+ \left[\left(1 + \theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{y_k^T s_r}{y_r^T s_r} - \theta_{r+1}\frac{y_k^T y_r}{y_r^T s_r}\right]s_r. \end{aligned} \tag{30}$$

With these the direction (26), at any non-restart step, can be computed as:

$$d_{k+1} = -v + \frac{(g_{k+1}^T s_k)w + (g_{k+1}^T w)s_k}{y_k^T s_k} - \left(1 + \frac{y_k^T w}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k}s_k \tag{31}$$

We can see that d_{k+1} from (31) is defined as a double quasi-Newton update scheme. It is useful to note that $y_k^T s_k > 0$ is sufficient to ensure that the direction d_{k+1} given by (31) is well defined and it is always a descent direction.

In the following we shall refer to the computation of θ_{k+1}. As we have already seen, in our algorithm θ_{k+1} is defined as a scalar approximation to 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 used.

θ_{k+1} *spectral*. This is given by (5) as the inverse of the Rayleigh quotient. Notice that $y_k^T s_k > 0$ is sufficient to ensure that θ_{k+1} in (5) is well defined.

θ_{k+1} *anticipative*. Recently, Andrei [1], using the information in two successive points of the iterative process, proposed another scalar approximation to the Hessian of function f, thus obtaining a new algorithm which favorably compares with the Barzilai and Borwein's. This is only a half step of the spectral procedure. Indeed, in point $x_{k+1} = x_k + \alpha_k d_k$ we can write

$$f(x_{k+1}) = f(x_k) + \alpha_k g_k^T d_k + \frac{1}{2}\alpha_k^2 d_k^T \nabla^2 f(z) d_k, \tag{32}$$

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 small enough, we can choose $z = x_{k+1}$ and consider $\gamma_{k+1} \in R$ as a scalar approximation of $\nabla^2 f(x_{k+1})$. This is an anticipative viewpoint, 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:

$$\gamma_{k+1} = \frac{2}{d_k^T d_k}\frac{1}{\alpha_k^2}\left[f(x_{k+1}) - f(x_k) - \alpha_k g_k^T d_k\right]. \tag{33}$$

This formula can also be found in Dai *et al.* [8]. Observe that $\gamma_{k+1} > 0$ for convex functions. 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 reduce a little the step size α_k as $\alpha_k - \eta_k$, maintaining the other quantities at their values in such a way so that γ_{k+1} is positive. To get a value for η_k let us select a real $\delta > 0$, "small enough" but comparable with the value of the function, and have

$$\eta_k = \frac{1}{g_k^T d_k}\left[f(x_k) - f(x_{k+1}) + \alpha_k g_k^T d_k + \delta\right], \tag{34}$$

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

$$\gamma_{k+1} = \frac{2}{d_k^T d_k}\frac{1}{(\alpha_k - \eta_k)^2}\left[f(x_{k+1}) - f(x_k) - (\alpha_k - \eta_k) g_k^T d_k\right]. \tag{35}$$

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

$$\theta_{k+1} = \frac{1}{\gamma_{k+1}}, \tag{36}$$

where γ_{k+1} is given by (33) or (35).

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},*

$$\gamma_{k+1} \leq 2L. \tag{37}$$

Proof From (33) 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,

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

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

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

Therefore, from (36) we get a lower bound for θ_{k+1} as:

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

i.e. it is bounded away from zero.

3. 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}, the following family of scaled conjugate gradient algorithms can be presented.

Algorithm SCALCG

Step 1 *Initialization*. Select $x_0 \in R^n$, and the parameters $0 < \sigma_1 \leq \sigma_2 < 1$. 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 (21) and (22). 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 the continuation of iterations*. If this test is satisfied, then the iterations are stopped, else set $k = k + 1$.

Step 4 *Scaling factor computation*. Compute θ_k using a spectral (5) or an anticipative (36) approach.

Step 5 *Restart direction*. Compute the (restart) direction d_k as in (18).

Step 6 *Line search*. Compute the initial guess of the step-length as:

$$\alpha_k = \frac{\alpha_{k-1}\|d_{k-1}\|_2}{\|d_k\|_2}.$$

With this initialization compute α_k satisfying the Wolfe conditions (21) and (22). 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 the continuation of iterations*. If this test is satisfied, then the iterations are stopped, else set $k = k + 1$.

Step 9 *Restart*. If the Powell restart criterion (24) or the angle restart criterion (25) is satisfied, then go to step 4 (a restart step); otherwise continue with step 10 (a standard step).

Step 10 *Standard direction*. 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:

$$\alpha_k = \frac{\alpha_{k-1}\|d_{k-1}\|_2}{\|d_k\|_2}.$$

With this initialization compute α_k as to satisfy the Wolfe conditions (21) and (22). 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 the continuation of iterations*. If this test is satisfied, then 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 is a step-length 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 step α_k in the line search is computed as $\alpha_{k-1}\|d_{k-1}\|_2/\|d_k\|_2$. This initialization, considered for the first time by Shanno and Phua in CONMIN, proves to be one of the best. Concerning the stopping criterion used in steps 3, 8 and 12 we take the following tests:

$$\|g_k\|_\infty \leq \varepsilon_g \quad \text{or} \quad \alpha_k |g_k^T d_k| \leq \varepsilon_f |f(x_{k+1})|, \tag{38}$$

where ε_f and ε_g are tolerances specified by the user. The second criterion in (38) says that the estimated change in the function value is insignificant compared to the function value itself.

4. Convergence analysis for strongly convex functions

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

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

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

$$(\nabla f(x) - \nabla f(y))^T(x - y) \geq \mu\|x - y\|^2 \tag{39}$$

and

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \tag{40}$$

for all x and y from L_0. For the convenience of the reader we include the following Lemma [12].

LEMMA 1 *Assume that d_k is a descent direction and ∇f satisfies the Lipschitz condition*

$$\|\nabla f(x) - \nabla f(x_k)\| \leq L\|x - x_k\| \tag{41}$$

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 (22), *then*

$$\alpha_k \geq \frac{1 - \sigma_2}{L}\frac{|g_k^T d_k|}{\|d_k\|^2}. \tag{42}$$

Proof Subtracting $g_k^T d_k$ from both sides of (22) and using the Lipschitz condition we have

$$(\sigma_2 - 1)g_k^T d_k \leq (g_{k+1} - g_k)^T d_k \leq L\alpha_k\|d_k\|^2. \tag{43}$$

Since d_k is a descent direction and $\sigma_2 < 1$, (42) follows immediately from (43). ■

LEMMA 2 *Assume that ∇f is strongly convex and Lipschitz continuous on L_0. If θ_{k+1} is selected by the spectral gradient, then the direction d_{k+1} given by* (18) *satisfies*:

$$\|d_{k+1}\| \leq \left(\frac{2}{\mu} + \frac{2L}{\mu^2} + \frac{L^2}{\mu^3}\right)\|g_{k+1}\|. \tag{44}$$

Proof By Lipschitz continuity (40) we have

$$\|y_k\| = \|g_{k+1} - g_k\| = \|\nabla f(x_k + \alpha_k d_k) - \nabla f(x_k)\| \leq L\alpha_k\|d_k\| = L\|s_k\|. \tag{45}$$

On the other hand, by the strong convexity (39)

$$y_k^T s_k \geq \mu\|s_k\|^2. \tag{46}$$

Selecting θ_{k+1} as in (5), it follows that

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k} \leq \frac{\|s_k\|^2}{\mu\|s_k\|^2} = \frac{1}{\mu}. \tag{47}$$

Now, using the triangle inequality and the above estimates (45)–(47), after some algebra on $\|d_{k+1}\|$, where d_{k+1} is given by (18), we get (44). ■

LEMMA 3 *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* (18) *satisfies*:

$$\|d_{k+1}\| \leq \left(\frac{1}{m}+\frac{2L}{m\mu}+\frac{1}{\mu}+\frac{L^2}{m\mu^2}\right)\|g_{k+1}\|. \tag{48}$$

Proof By strong convexity on L_0, there exists the constant $m > 0$, so that $\nabla^2 f(x) \geq mI$, for all $x \in L_0$. Therefore, $\gamma_{k+1} \geq m$ for every k. Now, from (36) we see that, for all k,

$$\theta_{k+1} \leq \frac{1}{m}. \tag{49}$$

With this, like in Lemma 2, we get (48). ■

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* (18) *and the step-length α_k selected to satisfy the Wolfe conditions* (21) *and* (22), *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

$$y_k^T d_k = (g_{k+1} - g_k)^T d_k \geq \mu\alpha_k \|d_k\|^2. \tag{50}$$

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 (50) 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 (21) we have

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

Considering the lower bound for α_k given by (42) in Lemma 1 and having in view that d_k is a descent direction, it follows that

$$\sum_{k=1}^{\infty} \frac{\left|g_k^T d_k\right|^2}{\|d_k\|^2} < \infty. \tag{51}$$

Now, from (23), using the inequality of Cauchy and (46) we get

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

Therefore, from (51) it follows that

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty. \tag{52}$$

Inserting in (52) the upperbound of d_k given by (44) or (48) we get

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

which completes the proof. ■

For general functions the convergence of the algorithm is coming from Theorem 1 and the restart procedure. Our algorithm is very close to the Perry/Shanno computational scheme. Therefore, for convex functions and under inexact line search it is global convergent. If restarts are employed, then the algorithm is convergent, but the speed of convergence can decrease. To a great extent, however, SCALCG algorithm is very close to Perry/Shanno computational scheme [21,22]. In fact SCALCG is a scaled memoryless BFGS preconditioned algorithm where the scaling factor is the inverse of a scalar approximation of the Hessian. For general functions that are bounded from below with bounded level sets and bounded second partial derivatives, the convergence of the proposed algorithm can be established using exactly the same analysis given by Shanno in [22]. Although a global convergence result has not been established for SCALCG, recall that for the Perry/Shanno scheme, the iterates either converge to a stationary point or the iterates cycle.

5. Computational results and comparisons

This section presents the performance of a Fortran implementation of the *SCALCG – scaled conjugate gradient algorithms* on a set of 700 unconstrained optimization problems. We compare the performance of SCALCG *versus SCG – the best spectral conjugate gradient algorithm* by Birgin and Martínez [3] (Perry-M1) given by (6), *versus sPRP – scaled Polak–Ribière–Polyak* given by (7), *versus sFR – scaled Fletcher–Reeves* given by (8), as well as *versus CONMIN – conjugate gradient package* by Shanno and Phua [23].

The SCALCG code is authored by Andrei, while the SCG, sPRP and sFR are co-authored by Birgin and Martínez. The CONMIN is co-authored by Shanno and Phua. All codes are written in Fortran using the same style of programming and compiled with f77 (default compiler settings) on a workstation 1.8 GHz. The SCALCG code implements the scaled conjugate gradient both with the spectral choice of scaling parameter θ_{k+1} and with the anticipative choice of this parameter. In order to compare SCALCG with SCG, we manufactured a new SCG code of Birgin and Martínez by introducing a sequence of code to compute θ_{k+1} in an anticipative manner, according to (36), and a sequence of code implementing the Powell restart criterion as in (24). At the same time, in CONMIN we introduced a sequence of code implementing the stopping criteria (38). The SCALCG code needs $11 \times n$ vectors, the SCG, sPRP and sFR needs only $6 \times n$ vectors, while CONMIN $7 \times n$ vectors of memory.

The test problems are the unconstrained problems in CUTE [4] library, along with other large-scale optimization test problems. We selected 70 large-scale unconstrained optimization test problems (29 from CUTE library) in extended or generalized form. For each test function we considered ten numerical experiments with the number of variables $n = 1000, 2000, \ldots, 10{,}000$.

Concerning the restart criterion, we implemented both the Powell and the angle criterion. Both these criteria have a crucial role in the practical efficiency of the algorithms. We compare the SCALCG algorithm with Powell restart criterion to the SCG algorithm with Powell and angle restart criterion, each of them in two variants: θ_{k+1} spectral and θ_{k+1} anticipative, respectively. Finally, we compare all these algorithms subject to Powell restart criterion.

Table 1. Global characteristics of SCALCG with θ^s *vs.* SCALCG with θ^a. Powell restart. 700 problems.

Global characteristics	θ^s	θ^a
Total number of iterations	688,274	689,774
Total number of function evaluations	1,061,853	922,317
Total CPU time (s)	7257.47	6352.70

In all algorithms the Wolfe line search conditions are implemented with $\sigma_1 = 0.0001$ and $\sigma_2 = 0.9$ SCALCG, SCG, sPRP and sFR use exactly the same implementation of Wolfe conditions. In CONMIN we consider the implementation given by Shanno and Phua which is very close to the one used in the algorithms of this study. The initial guess of the step-length at the first iteration is $\alpha_0 = 1/\|g_0\|$. At the following iterations, in all algorithms, the starting guess for step α_k is computed as $\alpha_{k-1}\|d_{k-1}\|_2/\|d_k\|_2$. This strategy proved to be one of the best selection of the initial guess of the step-length.

In all experiments we stopped the iterations whenever (38) was satisfied, where $\|\cdot\|_\infty$ denotes the maximum absolute component of a vector and $\varepsilon_g = 10^{-6}$ and $\varepsilon_f = 10^{-20}$. It is worth saying that (38) is a gradient-based criterion, the second criterion in (38) is based on the function values, testing if the change in the function value is insignificant subject to the function value itself. In our numerical experiments, we noticed that the second criterion in (38) was not very much involved in stopping the iterations.

In the following we present the numerical performances of all these codes, including the performance profiles of Dolan and Moré [9] subject to: the number of iterations, the number of function evaluations and the CPU time metrics. Finally, we show a comparison of all these codes.

In the first set of numerical experiments we compare SCALCG with θ_{k+1} spectral (θ^s) and θ_{k+1} anticipative (θ^a) using the Powell restart criterion. Table 1 shows the global characteristics referring to the total number of iterations, the total number of function evaluations and the total CPU time for these algorithms.

Out of 700 problems solved in this set of experiments the criterion $\|g_k\|_\infty < \varepsilon_g$ stopped the iterations for 605 problems, i.e. 86.4%, in case of SCALCG with θ^s, and for 577 problems, i.e. 82.4%, in case of SCALCG with θ^a.

Table 2 shows the number of problems, out of 700, for which SCALCG with θ^s and SCALCG with θ^a achieved the minimum number of iterations, the minimum number of function evaluations and the minimum CPU time, respectively.

Observe that the total number in Table 2 exceeds 700 due to ties for some problems.

The performances of these algorithms were evaluated using the profiles of Dolan and Moré [9]. 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. The left side of these figures gives the percentage of the test problems, out of 700, for which an algorithm is more efficient; the right side gives the percentage of the test problems that were successfully solved by each of the algorithms. Mainly, the right side represents a measure of an algorithm's robustness. In Figures 1 and 2, we compare the performance profiles of SCALCG with θ^s and SCALCG with θ^a referring to the number of function evaluations and CPU time metrics, respectively.

The top curve corresponds to the algorithm that solved most problems in a number of function evaluations (Figure 1) or in a CPU time (Figure 2) that was within a given factor

Table 2. Performance of SCALCG algorithms. Powell restart. 700 problems.

Performance criterion	No. of problems
SCALCG with θ^s achieved minimum no. of iterations in	392
SCALCG with θ^a achieved minimum no. of iterations in	475
Both algorithms achieved the same no. of iterations in	167
SCALCG with θ^s achieved minimum no. of function evaluations in	375
SCALCG with θ^a achieved minimum no. of function evaluations in	472
Both algorithms achieved the same no. of function evaluations in	147
SCALCG with θ^s achieved minimum CPU time in	322
SCALCG with θ^a achieved minimum CPU time in	512
SCALCG with θ^s and SCALCG with θ^a achieved the same CPU time in	134

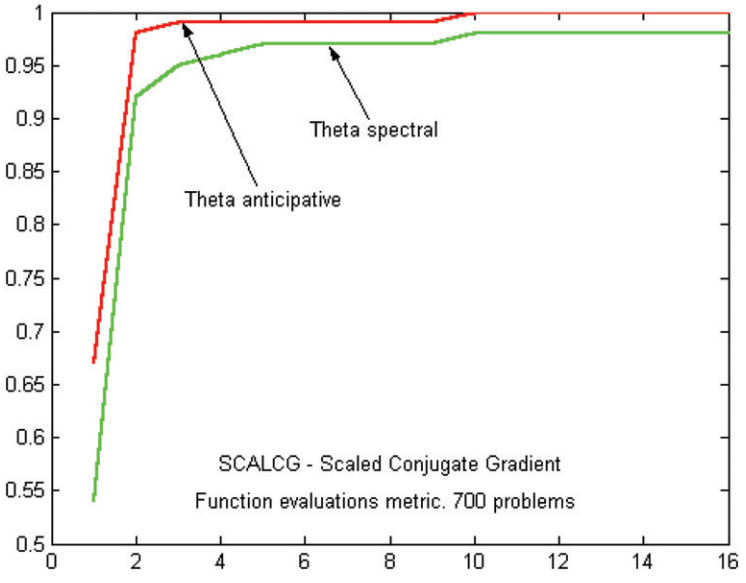

Figure 1. SCALCG with Powell restart. Function evaluations metric.

τ of the best number of function evaluations or CPU time, respectively. Since the top curve in Figures 1 and 2 corresponds to SCALCG θ^{a} with, this algorithm is clearly better than SCALCG with θ^s. However, both codes have similar performances, SCALCG with θ^a being slightly more robust.

The second set of numerical experiments refers to the performances of the SCG algorithm. Tables 3 and 4 show the global characteristics for SCG with θ^s *versus* SCG with θ^a using the Powell or angle restart criterion, respectively.

Observe that both codes have similar performances. Since the codes only differ in the procedure for θ_{k+1} computation, we see that θ_{k+1} computed in an anticipative way is competitive with the spectral formula. It is worth saying that out of 700 problems solved by SCG with θ^a in this numerical experiment, only for 204 (i.e. 29%) γ_{k+1} in (33) was negative in the case of Powell restart, and for only 176 (i.e. 25%) γ_{k+1} in (33) was negative in the case of angle restart. From Tables 3 and 4, we see that SCG with Powell restart

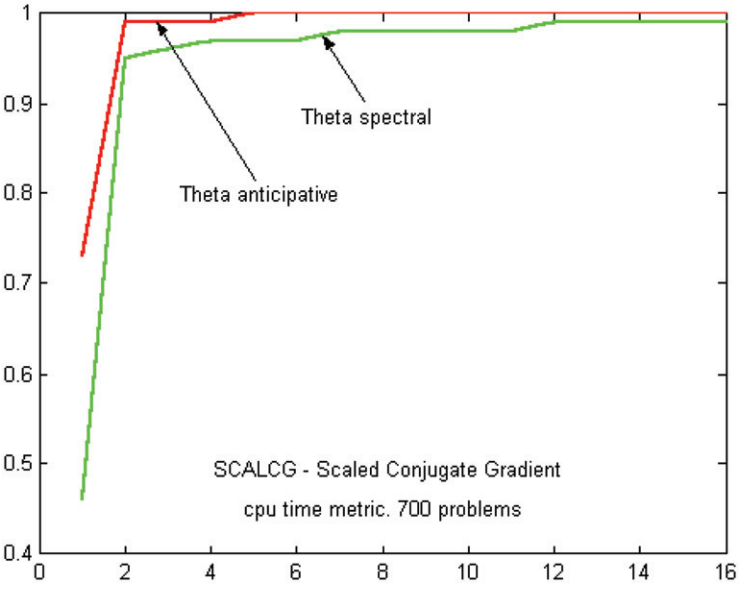

Figure 2. SCALCG with Powell restart. CPU time metric.

Table 3. Global characteristics of SCG with θ^s *vs.* SCG with θ^a. Powell restart. 700 problems.

Global characteristics	θ^s	θ^a
Total number of iterations	627,822	621,177
Total number of function evaluations	1,343,920	1,024,925
Total CPU time (s)	7780.47	7418.26

Table 4. Global characteristics of SCG with θ^s *vs.* SCG with θ^a. Angle restart. 700 problems.

Global characteristics	θ^s	θ^a
Total number of iterations	1,018,435	1,018,389
Total number of function evaluations	1,354,309	1,675,017
Total CPU time (s)	7702.57	7840.04

criterion in both implementations using θ^s and θ^a is slightly better than SCG with angle criterion. For example, for solving this set of 700 problems SCG with Powell restart and θ^a is with 421.78 s faster than SCG with angle restart.

Table 5 shows the global characteristics for SCALCG with θ^a *versus* SCG with θ^a using Powell restart criterion.

Table 6 shows the number of problems out of 700, for which SCALCG with θ^a and SCG with θ^a, both using Powell restart criterion, achieved the minimum number of

Table 5. Global characteristics of SCALCG with θ^a *vs.* SCG with θ^a. Powell restart. 700 problems.

Global characteristics	SCALCG	SCG
Total number of iterations	689, 774	621, 177
Total number of function evaluations	922, 317	1,024,925
Total CPU time (s)	6352.70	7418.26

Table 6. Performance of SCALCG (θ^a) *vs.* SCG (θ^a) algorithms. Powell restart. 700 problems.

Performance criterion	No. of problems
SCALCG achieved minimum no. of iterations in	572
SCG achieved minimum no. of iterations in	151
SCALCG and SCG achieved the same no. of iterations in	23
SCALCG achieved minimum no. of function evaluations in	573
SCG achieved minimum no. of function evaluations in	228
SCALCG and SCG achieved the same no. of function evaluations in	101
SCALCG achieved minimum CPU time in	324
SCG achieved minimum CPU time in	415
SCALCG and SCG achieved the same CPU time in	39

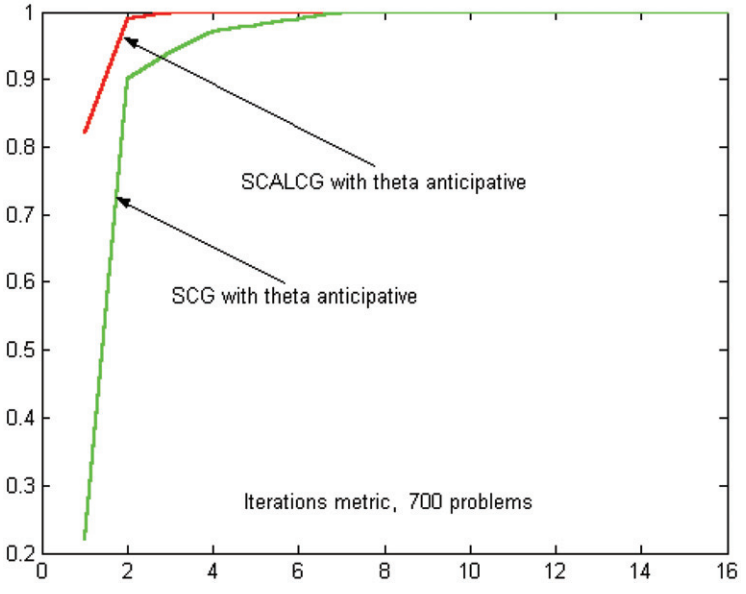

Figure 3. SCALCG *vs.* SCG. Powell restart. θ anticipative. Iterations metric.

iterations, the minimum number of function evaluations and the minimum CPU time, respectively.

In Figures 3–5 we compare the performance profiles of SCALCG and SCG subject to the number of iterations, the number of function evaluations and the CPU time metrics, respectively. From these figures it follows that SCALCG is the top performer for all values

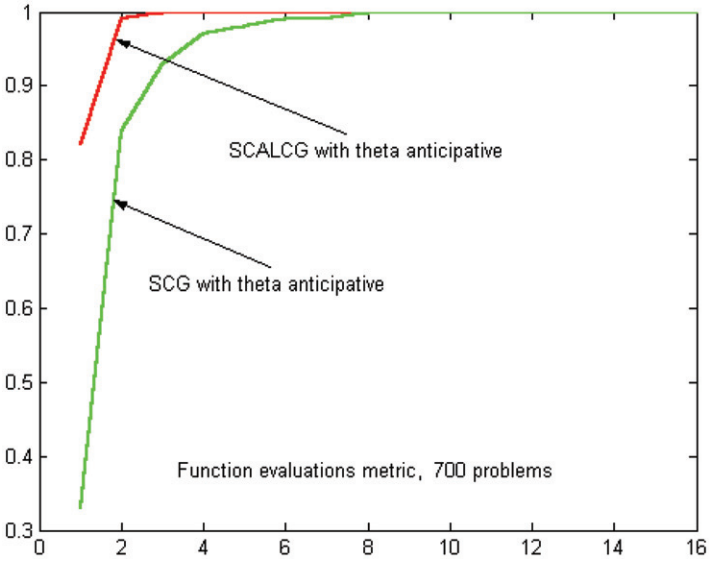

Figure 4. SCALCG *vs.* SCG. Powell restart. θ anticipative. Function evaluations metric.

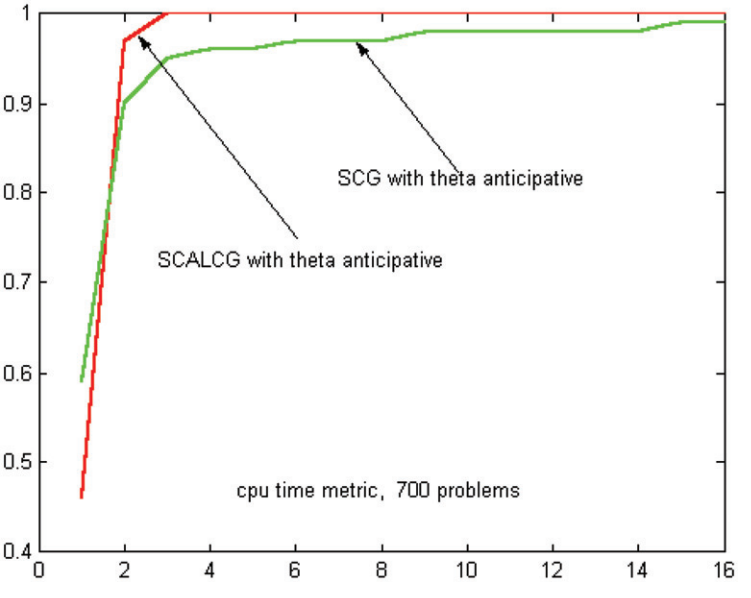

Figure 5. SCALCG *vs.* SCG. Powell restart. θ anticipative. CPU time metric.

of τ. Now, since both SCALCG and SCG use the same Wolfe line search, (with the same values for σ_1 and σ_2 parameters), the same restart criterion (Powell) and the same stopping criteria, these codes mainly differ in their choice of the search direction. SCALCG appears to generate a better search direction. The direction d_{k+1} used in SCALCG is more elaborate, its corresponding matrix is symmetric and positive definite and, more importantly, it satisfies the quasi-Newton equation in a restart environment. Although

Table 7. Global characteristics of SCALCG *vs.* SCG, sPRP, sFR and CONMIN. Powell restart (θ^a). 700 problems.

Algorithms	Iterations	Function evaluations	CPU time
SCALCG	689,774	922,317	6352.70
SCG	621,177	1,024,925	7418.26
sPRP	571,002	952,556	7245.89
sFR	625,641	1,034,764	7953.19
CONMIN	338,625	1,668,183	13278.64

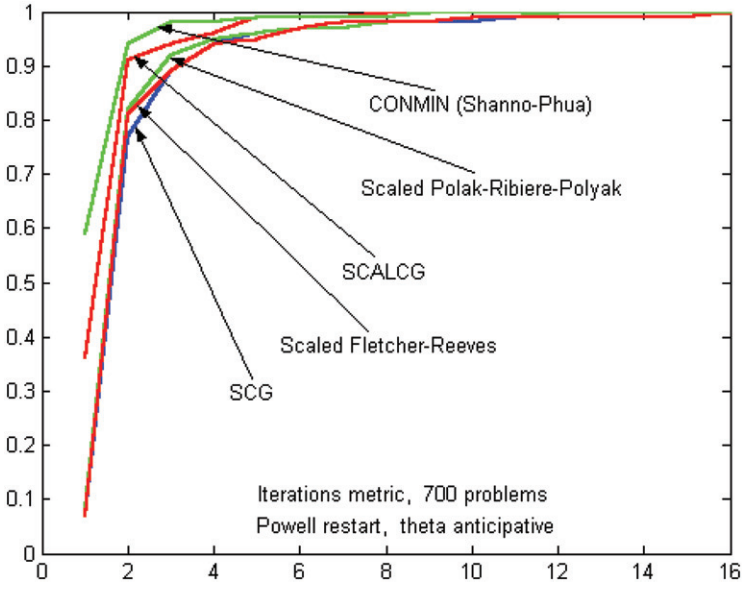

Figure 6. SCALCG *vs.* SCG, sPRP, sFR and CONMIN. θ anticipative, Powell restart, iterations metric.

the update formulas (18) and (29)–(31) are more complicated than the computational scheme (6) used in SCG, the scheme used in SCALCG proved to be more efficient and more robust in numerical experiments.

In the third set of experiments we compare SCALCG with: the SCG given by (6), the scaled Polak–Ribière–Polyak (sPRP) given by (7), the scaled Fletcher–Reeves (sFR) given by (8), all these algorithms with θ^a, and the CONMIN by Shanno and Phua, all of them using Powell restart criterion. Table 7 shows the global characteristics, i.e. the total number of iterations, the total number of function evaluations and the total CPU time (seconds), of SCALCG *versus* SCG, sPRP, sFR and CONMIN, for the 700 unconstrained optimization test problems considered in this numerical study.

The CPU time shows that SCALCG is twice faster than CONMIN. As for the number of function evaluations, SCALCG is about 1.8 times more efficient than CONMIN. The performance of SCG, sPRP and sFR algorithms in this implementation that uses the stopping criterion (38) is close to SCALCG. However, SCALCG is top performer among these algorithms.

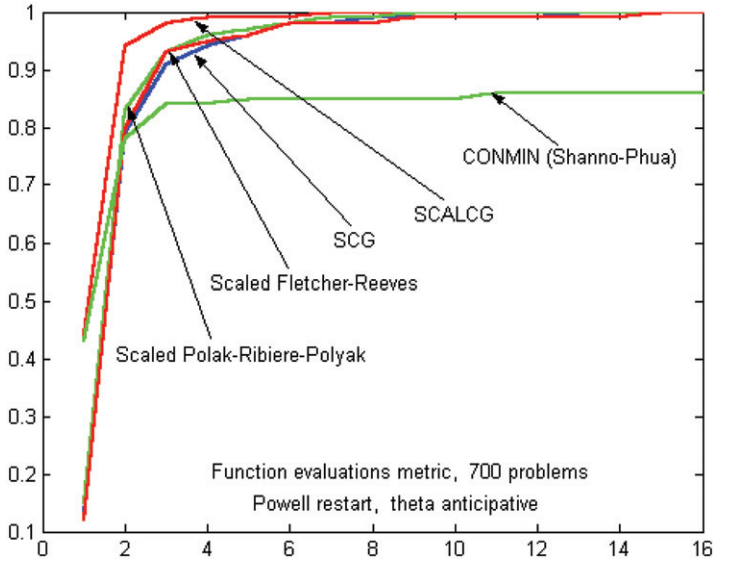

Figure 7. SCALCG *vs.* SCG, sPRP, sFR and CONMIN. θ anticipative, Powell restart, function evaluations metric.

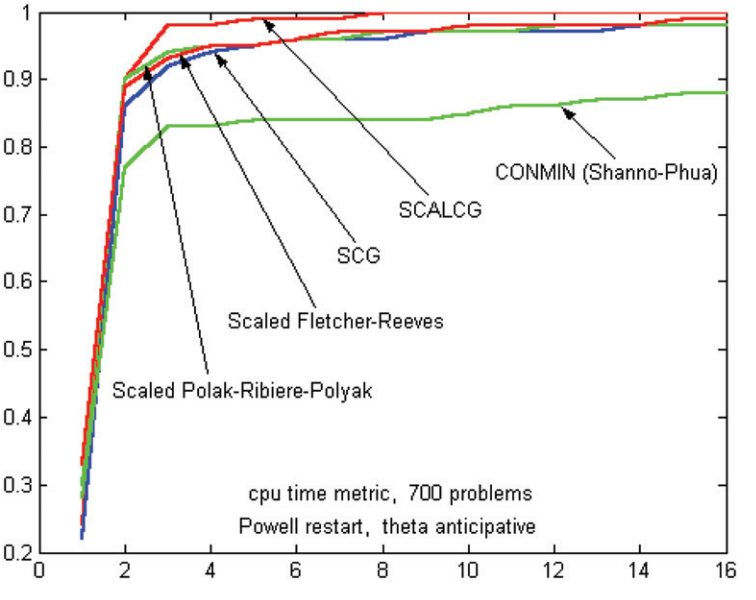

Figure 8. SCALCG *vs.* SCG, sPRP, sFR and CONMIN. θ anticipative, Powell restart, CPU time metric.

In Figures 6–8 we compare the performance profiles of these algorithms subject to the number of iterations, the number of function evaluations and the CPU time, respectively.

Referring to the iteration metric, from Figure 6 we see that CONMIN is better than SCALCG and the rest of algorithms. The algorithms SCG, sPRP and sFR have similar performances, sPRP being slightly better.

Referring to the function evaluation metric, from Figure 7 we see that for all values of τ, SCALCG (θ^a) is better than SCG (θ^a), sPRP, sFR and CONMIN, at least for this set of 700 test unconstrained optimization problems, with dimensions ranging from 10^3 to 10^4. The figures show that SCALCG (θ^a) appears to be the best, followed by: the scaled Polak–Ribiere–Polyak, the scaled Fletcher–Reeves, the SCG and finally followed by CONMIN. However, the SCG and the scaled variants of Polak–Ribiere–Polyak and Fletcher–Reeves have very similar performances for all values of τ.

Referring to the CPU time metric, from Figure 8 we see that for $\tau = 1$. All these algorithms have similar performances. However, SCALCG (θ^a) is the top performer for almost all values of τ. As for robustness, we see that SCALCG (θ^a) is more robust, followed by the scaled Polak–Ribiere–Polyak, the scaled Fletcher–Reeves, the SCG (θ^a) and CONMIN. Since all these algorithms use the same scaling parameter (θ^a), the same Wolfe line search (with the same values for σ_1 and σ_2 parameters), the same restart criterion (Powell), and the same stopping criterion, these algorithms mainly differ in their procedures for search direction computation. The Figures 6–8 gives computational evidence that SCALCG generates a better direction.

The comparison between SCALCG and CONMIN subject to function evaluations and CPU time metrics reveals that SCALCG is much more efficient than CONMIN. An explanation of this behavior seems to be as follows. As we know Oren [14], Oren and Luenberger [15] and Oren and Spedicato [16] modified the Broyden class of quasi-Newton methods by introducing a scalar parameter in order to make the sequence of inverse Hessian invariant under multiplication of function f by a scalar constant. For this scaling parameter, Shanno [21] suggests the value $s_k^T y_k / y_k^T y_k$ as the value minimizing the condition number of $H_k^{-1} H_{k+1}$. This scaling factor is used in CONMIN. On the other hand, in SCALCG we use another value for the scaling parameter θ_{k+1}, as a scalar approximation of the inverse Hessian given by (5) or (36) yielding to a more efficient direction. This factor greatly increases both the computational stability and the efficiency as the problem size increases, explaining the numerical behavior of SCALCG in comparison with CONMIN subject to function evaluations and CPU time metrics.

On the other hand, when we compare SCG (θ^a) with CONMIN (both of them using the Powell restart), from Table 7 we see that subject to the CPU time, SCG is about 1.8 times faster. From Tables 4 and 7 we see that SCG (θ^a) with angle restart criterion is about 1.69 times faster than CONMIN. Therefore, SCG (Perry-M1) by Birgin and Martínez with Powell restart procedure compares even better against CONMIN, at least for this set of 700 large-scale test problems.

6. Conclusion

The algorithm of Birgin and Martínez, which is mainly 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 was selected as spectral gradient or in an anticipative way by means of a formula using the function values in two successive points. Although the update formulas (18) and (29)–(31) are more complicated, the scheme proves to be efficient and robust in numerical experiments. The algorithm

implements the Wolfe conditions, and we proved that the steps were along the descent directions.

For the stopping criterion considered, the performance profiles for our scaled conjugate gradient algorithm was higher than those of the spectral conjugate gradient method of Birgin and Martínez, the scaled version of Polak–Ribière–Polyak and Fletcher–Reeves, as well as of CONMIN by Shanno and Phua, for a set consisting of 700 unconstrained optimization problems.

References

[1] N. Andrei, *A new gradient descent method for unconstrained optimization*, ICI Technical Report, March 2004.

[2] J. Barzilai and J.M. Borwein, *Two point step size gradient method*, IMA J. Num. Anal. 8 (1998), pp. 141–148.

[3] E. Birgin and J.M. Martínez, *A spectral conjugate gradient method for unconstrained optimization*, Appl. Math. Optim. 43 (2001), pp. 117–128.

[4] I. Bongartz, A.R. Conn, N.I.M. Gould, and P.L. Toint, *CUTE: constrained and unconstrained testing environments*, ACM Transactions on Mathematical Software 21 (1995), pp. 123–160.

[5] A. Cauchy, *Méthodes générales pour la résolution des systèmes déquations simultanées*, C.R. de l'Académie des Sciences. Paris, 25 (1847), pp. 536–538.

[6] Y.H. Dai and L.Z. Liao, *New conjugate conditions and related nonlinear conjugate gradient methods*, Appl. Math. Optim. 43 (2001), pp. 87–101.

[7] Y.H. Dai and J.Y. Yuan, *Convergence properties of the Beale–Powell restart algorithm*, Sciences in China (Series A) 41(11) (1998), pp. 1142–1150.

[8] Y.H. Dai, J.Y. Yuan, and Y. Yuan, *Modified two-point stepsize gradient methods for unconstrained optimization*, Comput. Optim. Appl. 22 (2002), pp. 103–109.

[9] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program. 91 (2002), pp. 201–213.

[10] R. Fletcher, *On the Barzilai–Borwein method*, in Optimization and Control Applications, L. Qi, K. Teo, and X. Yang, eds., Series: Applied Optimization, Vol. 96, Springer, Berlin, pp. 235–256.

[11] R. Fletcher and C.M. Reeves, *Function minimization by conjugate gradients*, Com. J. 7 (1964), pp. 149–154.

[12] W.W. Hager and H. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM J. Optim. 16 (2005), pp. 170–192.

[13] M.R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bureau Stand. Sec. B 48 (1952), pp. 409–436.

[14] S.S. Oren, *Self–scaling variable metric algorithm. Part II*, Manag. Sci. 20 (1974), pp. 863–874.

[15] S.S. Oren and D.G. Luenberger, *Self-scaling variable metric algorithm. Part I*, Manag. Sci. 20 (1976), pp. 845–862.

[16] S.S. Oren and E. Spedicato, *Optimal conditioning of self-scaling variable metric algorithms*, Math. Prog. 10 (1976), pp. 70–90.

[17] 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.

[18] E. Polak and G. Ribière, *Note sur la convergence de méthodes de directions conjuguées*, Revue Francaise Informat. Reserche Opérationnelle. 16 (1969), pp. 35–43.

[19] M.J.D. Powell, *Restart procedures for the conjugate gradient method*, Math. Prog. 12 (1977), pp. 241–254.

[20] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim. 7 (1997), pp. 26–33.

[21] D.F. Shanno, *Conjugate gradient methods with inexact searches*, Math. Oper. Res. 3 (1978), pp. 244–256.
[22] ———, *On the convergence of a new conjugate gradient algorithm*, SIAM J. Num. Anal. 15 (1978), pp. 1247–1257.
[23] D.F. Shanno and K.H. Phua, *Algorithm 500, minimization of unconstrained multivariate functions*, ACM Transactions on Mathematical Software 2 (1976), pp. 87–94.
[24] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev. 11 (1969), pp. 226–235.
[25] ———, *Convergence conditions for ascent methods II: some corrections*, SIAM Rev. 13 (1971), pp. 185–188.