An accelerated conjugate gradient algorithm with guaranteed descent and conjugacy conditions for unconstrained optimization

Neculai Andrei
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 paper we suggest a new conjugate gradient algorithm that for all $k \geq 0$ both the descent and the conjugacy conditions are guaranteed. The search direction is selected as a linear combination of $-g_{k+1}$ and s_k, where $g_{k+1} = \nabla f(x_{k+1})$, $s_k = x_{k+1} - x_k$ and the coefficients in this linear combination are selected in such a way that both the descent and the conjugacy condition are satisfied at every iteration. It is shown that for general nonlinear functions with bounded Hessian the algorithm with strong Wolfe line search generates directions bounded away from infinity. The algorithm uses an acceleration scheme that modify the steplength α_k in such a manner as to improve the reduction of the function values along the iterations. Numerical comparisons with some conjugate gradient algorithms using a set of 750 unconstrained optimization problems, some of them from the CUTE library, show that the computational scheme outperform the known conjugate gradient algorithms like Hestenes and Stiefel, Polak, Ribière and Polyak, Dai and Yuan or hybrid Dai and Yuan as well as CG_DESCENT by Hager and Zhang with Wolfe line search.

Keywords: Conjugate gradient, Wolfe line search, descent condition, conjugacy condition, unconstrained optimization.
AMS subject classifications: 49M20, 65K05, 90C30

1. Introduction

For solving the unconstrained optimization problems

$$\min_{x \in R^n} f(x), \tag{1.1}$$

where $f : R^n \to R$ is a continuously differentiable function, bounded from below, one of the most elegant and probably the simplest methods are the conjugate gradient methods. For solving this problem, starting from an initial guess $x_0 \in R^n$, a nonlinear conjugate gradient method, generates a sequence $\{x_k\}$ as:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where $\alpha_k > 0$ is obtained by line search, and the directions d_k are generated as:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \quad d_0 = -g_0. \tag{1.3}$$

In (1.3) β_k is known as the conjugate gradient parameter, $s_k = x_{k+1} - x_k$ and $g_k = \nabla f(x_k)$. The search direction d_k, assumed to be a descent one, plays the main role in these methods. On the other hand, the stepsize α_k guarantees the global convergence in some cases and is crucial in efficiency. Plenty of conjugate gradient methods are known, and an excellent survey of these methods, with a special attention on their global convergence, is given by Hager and Zhang [18]. Different conjugate gradient algorithms correspond to different

choices for the scalar parameter β_k. Line search in the conjugate gradient algorithms often is based on the standard Wolfe conditions [32, 33]

$$f(x_k+\alpha_k d_k)-f(x_k)\le \rho\alpha_k g_k^T d_k, \tag{1.4}$$

$$g(x_k+\alpha_k d_k)^T d_k \ge \sigma g_k^T d_k, \tag{1.5}$$

where d_k is supposed to be a descent direction and $0<\rho\le\sigma<1$.

A numerical comparison of conjugate gradient algorithms (1.2) and (1.3) with Wolfe line search, for different formulae of parameter β_k computation, including the Dolan and Moré performance profile, is given in [6].

If the initial direction d_0 is selected as $d_0=-g_0$, and the objective function to be minimized is a convex quadratic function

$$f(x)=\frac{1}{2}x^T Ax+b^T x+c \tag{1.6}$$

and the exact line searches are used, that is

$$\alpha_k=\arg\min_{\alpha>0} f(x_k+\alpha d_k), \tag{1.7}$$

then the conjugacy condition

$$d_i^T A d_j=0 \tag{1.8}$$

holds for all $i\ne j$. This relation (1.8) is the original condition used by Hestenes and Stiefel [19] to derive the conjugate gradient algorithms, mainly for solving symmetric positive-definite systems of linear equations. Using (1.3) and (1.6)-(1.8) it can be shown that x_{k+1} is the minimum of the quadratic function (1.6) in the subspace $x_k+span\{g_1,g_2,\ldots,g_k\}$ and the gradients $g_1,g_2,\ldots,g_k$ are mutually orthogonal unless that $g_k=0$ [15]. It follows that for convex quadratic functions the solution will be found after at most n iterations. Powell [27] shown that if the initial search direction is not g_0 then even for quadratic functions (1.6) the conjugate gradient algorithms does not terminate within a finitely number of iterations. It is well known that the conjugate gradient algorithm converges at least linearly [23], and an upper bound for the rate of convergence is obtained by Yuan [30].

Conjugate gradient algorithm (1.2) and (1.3) with exact line search always satisfy the condition $g_{k+1}^T d_{k+1}=-\|g_{k+1}\|^2$ which is in a direct connection with the sufficient descent condition

$$g_{k+1}^T d_{k+1}\le -t\|g_{k+1}\|^2 \tag{1.9}$$

for some positive constant $t>0$. The sufficient descent condition has been used often in the literature to analyze the global convergence of the conjugate gradient algorithms with inexact line search based on the strong Wolfe conditions. The sufficient descent condition is not needed in the convergence analyses of the Newton or quasi-Newton algorithms. However, it is necessary for the global convergence of conjugate gradient algorithms [12].

Let us denote $y_k=g_{k+1}-g_k$. For a general nonlinear twice differential function f, by the mean value theorem, there exists some $\xi\in(0,1)$ such that

$$d_{k+1}^T y_k=\alpha_k d_{k+1}^T \nabla^2 f(x_k+\xi\alpha_k d_k)d_k. \tag{1.10}$$

Therefore, it seems reasonable to replace (1.8) with the following conjugacy condition

$$d_{k+1}^T y_k=0. \tag{1.11}$$

In order to accelerate the conjugate gradient algorithm Perry [22] (see also Shanno [28]) extended the conjugacy condition by incorporating the second order information. He used the secant condition $H_{k+1}y_k=s_k$, where H_k is a symmetric approximation to the inverse

Hessian. Since for quasi-Newton method the search direction d_{k+1} is computed as $d_{k+1} = -H_{k+1} g_{k+1}$, it follows that

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

thus obtaining a new conjugacy condition. Recently, Dai and Liao [11] extended this condition and suggested the following new conjugacy condition

$$d_{k+1}^T y_k = -u g_{k+1}^T s_k, \tag{1.12}$$

where $u \geq 0$ is a scalar. Conjugate gradient algorithms are based on the conjugacy condition. To minimize a convex quadratic function in a subspace spanned by a set of mutually conjugate directions is equivalent to minimize this function along each conjugate direction in turn. This is a very good idea, but the performance of these algorithms is dependent on the accuracy of the line search. However, in conjugate gradient algorithms we always use inexact line search. Hence, when the line search is not exact, the "pure" conjugacy condition (1.11) may have disadvantages. Therefore, it seems more reasonable to consider in conjugate gradient algorithms the conjugacy condition (1.12). When the algorithm is convergent observe that $g_{k+1}^T s_k$ tends to zero along the iterations, and therefore conjugacy condition (1.12) tends to the pure conjugacy condition (1.11).

In this paper we suggest a new conjugate gradient algorithm that for all $k \geq 0$ both the descent and the conjugacy conditions are guaranteed. In section 2 we present the main ingredients of the search direction computation. The search direction is selected as a linear combination of $-g_{k+1}$ and s_k, where the coefficients in this linear combination are selected in such a way that both the descent and the conjugacy condition to be satisfied at every iteration. In section 3 we prove the convergence of the algorithm. It is shown that for general nonlinear functions with bounded Hessian the algorithm with strong Wolfe line search generates directions bounded away from infinity. Section 4 is devoted to present an acceleration scheme of the algorithm. The idea of this computational scheme is to take advantage that the step lengths α_k in conjugate gradient algorithms are very different from 1. Therefore, we suggest we modify α_k in such a manner as to improve the reduction of the function values along the iterations. Section 5 is devoted to present the ACGSYS algorithm. We prove that for uniformly convex functions the convergence of the accelerated algorithm is still linear, but the reduction in function values is significantly improved. In section 6 some numerical experiments and performance profiles of Dolan-Moré [14] corresponding to this new conjugate gradient algorithm are given. The performance profiles correspond to a set of 750 unconstrained optimization problems presented in [1]. It is shown that this new conjugate gradient algorithm outperforms the classical Hestenes and Stiefel [19], Dai and Yuan [13], Polak, Ribière and Polyak [24, 25] or hybrid Dai and Yuan [13] conjugate gradient algorithms and also the CG_DESCENT conjugate gradient algorithm with Wolfe line search by Hager and Zhang [17].

2. Conjugate gradient algorithm with guaranteed descent and conjugacy conditions

For solving the minimization problem (1.1) consider the following conjugate gradient algorithm

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.1}$$

where $\alpha_k > 0$ is obtained by Wolfe line search, and the directions d_k are generated as:

$$d_{k+1} = -\theta_k g_{k+1} + \beta_k s_k, \tag{2.2}$$

$d_0 = -g_0$, where θ_k and β_k are scalar parameters which follows to be determined. Algorithms of this form, or variations of it, have been studied by many authors. For example, Birgin and Martínez [7] suggested a spectral conjugate gradient method, where

$\theta_k = s_k^T s_k / s_k^T y_k$. On the other hand Andrei [3,4,5] considers a preconditioned conjugate gradient algorithm where the preconditioner is a scaled memoryless BFGS matrix and the parameter scaling the gradient is selected as the spectral gradient. Yuan and Stoer [31] studied the conjugate gradient algorithm on a subspace, where the search direction d_{k+1} at the $k-$th iteration ($k \geq 1$) is taken from the subspace $span\{g_{k+1}, d_k\}$.

In our algorithm for all $k \geq 0$ the θ_k and β_k scalar parameters in (2.2) are determined from the *descent condition*

$$g_{k+1}^T d_{k+1} = -\theta_k g_{k+1}^T g_{k+1} + \beta_k g_{k+1}^T s_k = -t\|g_{k+1}\|^2 \tag{2.3}$$

and the *conjugacy condition* (1.12)

$$y_k^T d_{k+1} = -\theta_k y_k^T g_{k+1} + \beta_k y_k^T s_k = -u(s_k^T g_{k+1}), \tag{2.4}$$

where $t > 0$ and $u > 0$ are scalar parameters. Observe that in (2.3) we modified the classical sufficient descent condition (1.9) with equality. It is worth saying that the main condition in any conjugate gradient algorithm is the descent condition $g_k^T d_k < 0$ or the sufficient descent condition (1.9). The conjugacy condition (1.11) or its modification (1.12) is not so stringent. In fact very few conjugate gradient algorithms satisfy this condition. For example, the Hestenes – Stiefel algorithm has this property that the pure conjugacy condition always holds, independent of the line search.

If $u = 0$, then (2.4) is the "pure" conjugacy condition. However, in our algorithm in order to accelerate the algorithm and incorporate the second order information we take $u > 0$. Now, let us define

$$\Delta_k = (y_k^T g_{k+1})(s_k^T g_{k+1}) - \|g_{k+1}\|^2 (y_k^T s_k). \tag{2.5}$$

Supposing that $\Delta_k \neq 0$ then from the linear algebraic system given by (2.3) and (2.4) we get

$$\theta_k = \frac{-(y_k^T s_k)\|g_{k+1}\|^2 t + (s_k^T g_{k+1})^2 u}{\Delta_k}, \tag{2.6}$$

$$\beta_k = \frac{-(y_k^T g_{k+1})\|g_{k+1}\|^2 t + (s_k^T g_{k+1})\|g_{k+1}\|^2 u}{\Delta_k}. \tag{2.7}$$

If the line search is exact, that is $s_k^T g_{k+1} = 0$, then $\Delta_k = -\|g_{k+1}\|^2 (y_k^T s_k) < 0$, if the line search satisfies the Wolfe condition (1.5) and if $g_{k+1} \neq 0$. Therefore from (2.6) and (2.7) we get $\theta_k = t$ and $\beta_k = (y_k^T g_{k+1})t/(y_k^T s_k)$, i.e.

$$d_{k+1} = t\left(-g_{k+1} + \frac{y_k^T g_{k+1}}{y_k^T s_k} s_k\right) = t d_{k+1}^{HS}, \tag{2.8}$$

where d_{k+1}^{HS} is the Hestenes-Stiefel direction.

Proposition 2.1. *If*

$$\sigma \leq \frac{\|g_{k+1}\|^2}{|y_k^T g_{k+1}| + \|g_{k+1}\|^2}, \tag{2.9}$$

then for all $k \geq 1$, $\Delta_k < 0$.

Proof. Observe that

$$s_k^T g_{k+1} = s_k^T y_k + s_k^T g_k < s_k^T y_k. \tag{2.10}$$

The Wolfe condition (1.5) gives

$$g_{k+1}^T s_k \geq \sigma g_k^T s_k = -\sigma y_k^T s_k + \sigma g_{k+1}^T s_k. \tag{2.11}$$

Since $\sigma < 1$, we can rearrange (2.11) to obtain

$$g_{k+1}^T s_k \geq \frac{-\sigma}{1-\sigma} y_k^T s_k. \tag{2.12}$$

Now, combining this lower bound for $g_{k+1}^T s_k$ with the upper bound (2.10) we get

$$\left| g_{k+1}^T s_k \right| \leq \left| y_k^T s_k \right| \max\left\{1, \frac{\sigma}{1-\sigma}\right\}. \tag{2.13}$$

Again, observe that the Wolfe condition gives $y_k^T s_k > 0$ (if $g_k \neq 0$). Therefore, if σ is bounded as in (2.9), then

$$\left| g_{k+1}^T s_k \right| \left| g_{k+1}^T y_k \right| \leq \left| y_k^T s_k \right| \left| g_{k+1}^T y_k \right| \max\left\{1, \frac{\sigma}{1-\sigma}\right\} \leq \left| y_k^T s_k \right| \left\| g_{k+1} \right\|^2.$$

i.e. $\Delta_k < 0$ for all $k \geq 1$. ■

From (2.9) observe that $\sigma < 1$. Since $g_k^T s_k = -t \left\| g_k \right\|^2 < 0$, i.e. d_k is a descent direction, it follows that $\left| g_{k+1}^T y_k \right| \to \left\| g_{k+1} \right\|^2$. Therefore $\sigma \to 1/2$, i.e. $0 < \rho < \sigma < 1$, since usually ρ is selected enough small to ensure the reduction of function values along the iterations.

3. Convergence analysis

In this section we analyze the convergence of the algorithm (2.1) and (2.2), where θ_k and β_k are given by (2.6) and (2.7) respectively, and $d_0 = -g_0$. In the following we consider that $g_k \neq 0$ for all $k \geq 1$, otherwise a stationary point is obtained. Assume that:

(i) *The level set* $S = \left\{ x \in R^n : f(x) \leq f(x_0) \right\}$ *is bounded.*

(ii) *In a neighborhood* N *of* S*, the function* f *is continuously differentiable and its gradient is Lipschitz continuous, i.e. there exists a constant* $L > 0$ *such that* $\left\| \nabla f(x) - \nabla f(y) \right\| \leq L \left\| x - y \right\|$*, for all* $x, y \in N$.

Under these assumptions on f there exists a constant $\Gamma \geq 0$ such that $\left\| \nabla f(x) \right\| \leq \Gamma$ for all $x \in S$. In order to prove the global convergence, we assume that the step size α_k in (2.1) is obtained by the strong Wolfe line search, that is,

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \rho \alpha_k g_k^T d_k, \tag{3.1}$$

$$\left| g(x_k + \alpha_k d_k)^T d_k \right| \leq \sigma g_k^T d_k. \tag{3.2}$$

where ρ and σ are positive constants such that $0 < \rho \leq \sigma < 1$.

Dai *et al.* [12] proved that for any conjugate gradient method with strong Wolfe line search the following general result holds:

Lemma 3.1. *Suppose that the assumptions (i) and (ii) hold and consider any conjugate gradient method (1.2) and (1.3), where* d_k *is a descent direction and* α_k *is obtained by the strong Wolfe line search (3.1) and (3.2). If*

$$\sum_{k \geq 1} \frac{1}{\left\| d_k \right\|^2} = \infty, \tag{3.3}$$

then

$$\liminf_{k \to \infty} \left\| g_k \right\| = 0. \ ■ \tag{3.4}$$

Therefore, the iteration can fail, in the sense that $\|g_k\| \ge \gamma > 0$, for all k, only if $\|d_k\| \to \infty$ sufficiently rapidly. The following theorem can be proved.

Theorem 3.1. *Suppose that the assumptions (i) and (ii) hold and consider the conjugate gradient algorithm (2.1), where the direction d_{k+1} is given by (2.2), (2.5)-(2.7) and the step length α_k is obtained by the strong Wolfe line search (3.1) and (3.2). Assume that $\nabla^2 f(x)$ is bounded, i.e. $\nabla^2 f(x) \le MI$, for any $x \in S$, where M is a positive constant, then $\liminf_{k\to\infty} \|g_k\| = 0$.*

Proof. Since $\nabla^2 f(x)$ is bounded we have

$$y_k^T s_k = (g_{k+1} - g_k)^T s_k = s_k^T \nabla^2 f(\bar{x}_k) s_k \le M \|s_k\|^2 = O(\|s_k\|^2), \tag{3.5}$$

where $\bar{x}_k$ is a point on the line segment connecting x_k and x_{k+1}. Observe that

$$s_k^T g_{k+1} \le \|s_k\| \|g_{k+1}\| \le \Gamma \|s_k\| = O(\|s_k\|),$$
$$y_k^T g_{k+1} \le \|y_k\| \|g_{k+1}\| \le L\Gamma \|s_k\| = O(\|s_k\|).$$

Hence,

$$(s_k^T g_{k+1})(y_k^T g_{k+1}) = O(\|s_k\|^2). \tag{3.6}$$

Therefore for all sufficiently large k,

$$\Delta_k = O(\|s_k\|^2). \tag{3.7}$$

On the other hand, observe that

$$-(y_k^T s_k)\|g_{k+1}\|^2 t + (s_k^T g_{k+1})^2 u = \max\left\{O(\|s_k\|^2), O(\|s_k\|^2)\right\} = O(\|s_k\|^2),$$
$$-(y_k^T g_{k+1})\|g_{k+1}\|^2 t + (s_k^T g_{k+1})\|g_{k+1}\|^2 u = \max\left\{O(\|s_k\|), O(\|s_k\|)\right\} = O(\|s_k\|).$$

Therefore for all sufficiently large k,

$$\theta_k = \frac{O(\|s_k\|^2)}{O(\|s_k\|^2)} = O(1) \quad \text{and} \quad \beta_k = \frac{O(\|s_k\|)}{O(\|s_k\|^2)} = \frac{1}{O(\|s_k\|)}. \tag{3.8}$$

From (2.2) we have

$$\|d_{k+1}\| \le |\theta_k| \|g_{k+1}\| + |\beta_k| \|s_k\| \le \Gamma O(1) + \frac{1}{O(\|s_k)\|} \|s_k\| = O(1). \tag{3.9}$$

Therefore, there is an index k_0 and a positive constant B, such that for all $k \ge k_0$, $\|d_k\| \le B$, i.e. $\sum_{k\ge 1} \frac{1}{\|d_k\|^2} = \infty$. By Lemma 1 we have $\liminf_{k\to\infty} \|g_k\| = 0$. ■

From (2.6) and (2.7) we see that

$$d_{k+1} = \left[\frac{(y_k^T s_k)\|g_{k+1}\|^2}{\Delta_k} g_{k+1} - \frac{(y_k^T g_{k+1})\|g_{k+1}\|^2}{\Delta_k} s_k \right] t$$
$$+ \left[\frac{(s_k^T g_{k+1})\|g_{k+1}\|^2}{\Delta_k} s_k - \frac{(s_k^T g_{k+1})^2}{\Delta_k} g_{k+1} \right] u. \tag{3.10}$$

Since the algorithm is convergent, i.e. $\{x_k\} \to x^*$, where x^* is the local optimal point of (1.1), it follows that $\lim_{k\to\infty} \|s_k\| = 0$. On the other hand, $s_k^T g_{k+1} \to 0$ for $k \to \infty$. Therefore the coefficient of u in (3.10) tends to zero, i.e. the algorithm is not sensitive to the values of parameter u.
However, since $s_k^T g_{k+1} \to 0$ for $k \to \infty$, it follows that

$$\frac{t(y_k^T s_k)\|g_{k+1}\|^2}{\Delta_k} \to -t,$$

showing that the descent condition (2.3) is more important than the conjugacy condition (2.4). However, the conjugacy condition is important in the economy of our algorithm because it includes the information of the second order.

4. Acceleration of the algorithm

It is common to see that in conjugate gradient algorithms the search directions tend to be poorly scaled and as a consequence the line search must perform more function evaluations in order to obtain a suitable steplength α_k. Therefore, the research efforts was directed to design procedures for direction computation which takes the second order information. The algorithms implemented in CONMIN by Shanno and Phua [29] or SCALCG by Andrei [3-5] use the BFGS preconditioning with remarkable results. In this section we focus on the step length modification. In the context of gradient descent algorithm with backtracking this idea of step length modification has been considered for the first time in [2].
Jorge Nocedal [21] pointed out that in conjugate gradient methods the step lengths may differ from 1 in a very unpredictable manner. They can be larger or smaller than 1 depending on how the problem is scaled. This is in very sharp contrast to the Newton and quasi-Newton methods, including the limited memory quasi-Newton methods, which accept the unit steplength most of the time along the iterations, and therefore usually they require only few function evaluations per search direction. Numerical comparisons between conjugate gradient methods and the limited memory quasi Newton method, by Liu and Nocedal [20], show that the latter is more successful [6]. One explanation of efficiency of this limited memory quasi-Newton method is given by its ability to accept unity step lengths along the iterations. In this section we take advantage of this behavior of conjugate gradient algorithms and present an acceleration scheme. Basically it modifies the step length in a multiplicative manner to improve the reduction of the function values along the iterations. First, for completeness, we prove that the step length α_k given by the Wolfe line search conditions is bounded away from zero (see also [17]). Secondly, we present the acceleration scheme.

Line search. For implementing the algorithm (1.2) one of the crucial elements is the stepsize computation. In the following we consider the line searches that satisfy the Wolfe conditions (1.4) and (1.5).

Proposition 4.1. *Assume that d_k is a descent direction and ∇f satisfies the Lipschitz condition $\|\nabla f(x) - \nabla f(x_k)\| \le L\|x - x_k\|$ for all x on the line segment connecting x_k and x_{k+1}, where L is a positive constant. If the line search satisfies the Wolfe conditions (1.4) and (1.5), then*

$$\alpha_k \ge \frac{(1-\sigma)}{L}\frac{|g_k^T d_k|}{\|d_k\|^2}. \tag{4.1}$$

Proof. To prove (4.1) subtract $g_k^T d_k$ from both sides of (1.5) and using the Lipschitz condition we get:

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

But, d_k is a descent direction and since $\sigma < 1$, we immediately get (4.1). ■

Therefore, satisfying the Wolfe line search conditions α_k is bounded away from zero, i.e. for all $k \ge 0$ there exists a positive constant ω, such that $\alpha_k \ge \omega$.

Acceleration scheme. Suppose that the function f is twice continuously differentiable. At the iteration $k = 1, 2, \ldots$ we know x_k, f_k, g_k and d_k. Now, by the Wolfe line search (1.4) and (1.5) we can compute α_k with which the following point $z = x_k + \alpha_k d_k$ is determined. The first Wolfe condition (1.4) shows that the steplength $\alpha_k > 0$, satisfies:

$$f(z) = f(x_k + \alpha_k d_k) \le f(x_k) + \rho \alpha_k g_k^T d_k .$$

With these, let us introduce the accelerated conjugate gradient algorithm by means of the following iterative scheme:

$$x_{k+1} = x_k + \gamma_k \alpha_k d_k , \tag{4.2}$$

where $\gamma_k > 0$ is a parameter which follows to be determined in such a manner as to improve the behavior of the algorithm. Now, we have:

$$f(x_k + \alpha_k d_k) = f(x_k) + \alpha_k g_k^T d_k + \frac{1}{2} \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + o\left(\|\alpha_k d_k\|^2 \right). \tag{4.3}$$

On the other hand, for $\gamma > 0$ we have:

$$f(x_k + \gamma \alpha_k d_k) = f(x_k) + \gamma \alpha_k g_k^T d_k + \frac{1}{2} \gamma^2 \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + o\left(\|\gamma \alpha_k d_k\|^2 \right). \tag{4.4}$$

With these we can write:

$$f(x_k + \gamma \alpha_k d_k) = f(x_k + \alpha_k d_k) + \Psi_k(\gamma), \tag{4.5}$$

where

$$\begin{aligned} \Psi_k(\gamma) = & \frac{1}{2} (\gamma^2 - 1) \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + (\gamma - 1) \alpha_k g_k^T d_k \\ & + \gamma^2 \alpha_k o\left(\alpha_k \|d_k\|^2 \right) - \alpha_k o\left(\alpha_k \|d_k\|^2 \right). \end{aligned} \tag{4.6}$$

Let us denote:

$$\begin{aligned} a_k &= \alpha_k g_k^T d_k \le 0, \\ b_k &= \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k , \\ \varepsilon_k &= o\left(\alpha_k \|d_k\|^2 \right). \end{aligned}$$

Observe that $a_k \le 0$, since d_k is a descent direction, and for convex functions $b_k \ge 0$. Besides, ε_k is independent of γ. Therefore,

$$\Psi_k(\gamma) = \frac{1}{2} (\gamma^2 - 1) b_k + (\gamma - 1) a_k + \gamma^2 \alpha_k \varepsilon_k - \alpha_k \varepsilon_k . \tag{4.7}$$

Now, we see that $\Psi_k'(\gamma) = (b_k + 2\alpha_k \varepsilon_k)\gamma + a_k$ and $\Psi_k'(\gamma_m) = 0$ where

$$\gamma_m = -\frac{a_k}{b_k + 2\alpha_k \varepsilon_k} . \tag{4.8}$$

Observe that $\Psi'_k(0) = a_k < 0$. Therefore, assuming that $b_k + 2\alpha_k\varepsilon_k > 0$, then $\Psi_k(\gamma)$ is a convex quadratic function with minimum value in point γ_m and

$$\Psi_k(\gamma_m) = -\frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} \le 0.$$

Considering $\gamma = \gamma_m$ in (4.5) and since $b_k \ge 0$, we see that for every $k \ge 1$

$$f(x_k + \gamma_m\alpha_k d_k) = f(x_k + \alpha_k d_k) - \frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} \le f(x_k + \alpha_k d_k),$$

which is a possible improvement of the values of function f (when $a_k + (b_k + 2\alpha_k\varepsilon_k) \ne 0$). Therefore, using this simple multiplicative modification of the stepsize α_k as $\gamma_k\alpha_k$ where $\gamma_k = \gamma_m = -a_k/(b_k + 2\alpha_k\varepsilon_k)$ we get:

$$\begin{aligned} f(x_{k+1}) &= f(x_k + \gamma_k\alpha_k d_k) \le f(x_k) + \rho\alpha_k g_k^T d_k - \frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} \\ &= f(x_k) - \left[\frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} - \rho a_k\right] \le f(x_k), \end{aligned} \tag{4.9}$$

since $a_k \le 0$, (d_k is a descent direction).

Now, neglecting the contribution of ε_k in (4.9), we still get an improvement of the function values as

$$f(x_{k+1}) \le f(x_k) - \left[\frac{(a_k + b_k)^2}{2b_k} - \rho a_k\right] \le f(x_k).$$

In order to get the algorithm we have to determine a way for b_k computation. For this, at point $z = x_k + \alpha_k d_k$ we have:

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

where $\tilde{x}_k$ is a point on the line segment connecting x_k and z. On the other hand, at point $x_k = z - \alpha_k d_k$ we have:

$$f(x_k) = f(z - \alpha_k d_k) = f(z) - \alpha_k g_z^T d_k + \frac{1}{2}\alpha_k^2 d_k^T \nabla^2 f(\bar{x}_k) d_k,$$

where $g_z = \nabla f(z)$ and $\bar{x}_k$ is a point on the line segment connecting x_k and z. Having in view the local character of searching and that the distance between x_k and z is small enough, we can consider $\tilde{x}_k = \bar{x}_k = x_k$. So, adding the above equalities we get:

$$b_k = -\alpha_k y_k^T d_k, \tag{4.10}$$

where $y_k = g_k - g_z$. Observe that for strictly convex functions $b_k > 0$. However, if $b_k = 0$, then the acceleration scheme doesn't have any effect by considering $\gamma_k = 1$ in (4.2).

Observe that if $|a_k| > b_k$, then $\gamma_k > 1$. In this case $\gamma_k\alpha_k > \alpha_k$ and it is also possible that $\gamma_k\alpha_k \le 1$ or $\gamma_k\alpha_k > 1$. Hence, the steplength $\gamma_k\alpha_k$ can be greater than 1. On the other hand, if $|a_k| \le b_k$, then $\gamma_k \le 1$. In this case $\gamma_k\alpha_k \le \alpha_k$, so the steplength $\gamma_k\alpha_k$ is reduced. Therefore, if $|a_k| \ne b_k$, then $\gamma_k \ne 1$ and the steplength α_k computed by Wolfe conditions will be modified by its increasing or its reducing through factor γ_k.

Neglecting ε_k in (4.7), we see that $\Psi_k(1)=0$ and if $|a_k| \le b_k/2$, then $\Psi_k(0)=-a_k-b_k/2\le 0$ and $\gamma_k<1$. Therefore, for any $\gamma\in[0,1]$, $\Psi_k(\gamma)\le 0$. As a consequence for any $\gamma\in(0,1)$, it follows that $f(x_k+\gamma\alpha_k d_k)<f(x_k)$. In this case, for any $\gamma\in[0,1]$, $\gamma_k\alpha_k\le\alpha_k$. However, in our algorithm we selected $\gamma_k=\gamma_m$ as the point achieving the minimum value of $\Psi_k(\gamma)$.

5. ACGSYS algorithm

Step 1. Select a starting point $x_0\in dom\, f$ and compute: $f_0=f(x_0)$ and $g_0=\nabla f(x_0)$. Select some positive values for t and u. Set $d_0=-g_0$ and $k=0$.

Step 2. Test a criterion for stopping the iterations. If the test is satisfied, then stop; otherwise continue with step 3.

Step 3. Using the Wolfe line search conditions determine the steplength α_k.

Step 4. Compute: $z=x_k+\alpha_k d_k$, $g_z=\nabla f(z)$ and $y_k=g_k-g_z$.

Step 5. Compute: $a_k=\alpha_k g_k^T d_k$, and $b_k=-\alpha_k y_k^T d_k$.

Step 6. If $b_k\ne 0$, then compute $\gamma_k=-a_k/b_k$ and update the variables as $x_{k+1}=x_k+\gamma_k\alpha_k d_k$, otherwise update the variables as $x_{k+1}=x_k+\alpha_k d_k$. Compute f_{k+1} and g_{k+1}. Compute $y_k=g_{k+1}-g_k$ and $s_k=x_{k+1}-x_k$.

Step 7. Determine θ_k and β_k as in (2.6) and (2.7) respectively, where Δ_k is computed as in (2.5).

Step 8. Compute the search direction as: $d_{k+1}=-\theta_k g_{k+1}+\beta_k s_k$.

Step 9. Restart criterion. If $|g_{k+1}^T g_k|>0.2\|g_{k+1}\|^2$ then set $d_{k+1}=-g_{k+1}$.

Step 10. Consider $k=k+1$ and go to step 2. ■

It is well known that if f is bounded along the direction d_k then there exists a stepsize α_k satisfying the Wolfe line search conditions (1.4) and (1.5). In our algorithm when the Powell restart condition is satisfied, then we restart the algorithm with the negative gradient $-g_{k+1}$. Under reasonable assumptions, the Wolfe conditions and the Powell restart criterion are sufficient to prove the global convergence of the algorithm. The first trial of the step length crucially affects the practical behavior of the algorithm. At every iteration $k\ge 1$ the starting guess for the step α_k in the line search is computed as $\alpha_{k-1}\|d_{k-1}\|/\|d_k\|$. This selection was used for the first time by Shanno and Phua in CONMIN [29] and in SCALCG by Andrei [3-5].

In the following, for uniformly convex functions, we prove the linear convergence of the acceleration scheme. Recall that a function f is uniformly convex on the level set $S=\{x: f(x)\le f(x_0)\}$ if there is a positive constant m such that

$$f(y)\ge f(x)+\nabla f(x)^T(y-x)+\frac{1}{2}m\|y-x\|^2$$

for all $x,y\in S$. For uniformly convex functions it is easy to prove that

$$\|\nabla f(x)\|^2\ge 2m\left(f(x)-f(x^*)\right),$$

for all $x\in S$, where x^* is a local solution of (1.1) [9].

Proposition 5.1. *Suppose that* f *is a uniformly convex function on the level set* $S=\{x: f(x)\le f(x_0)\}$. *Assume that* d_k *satisfies the descent condition* $g_k^T d_k=-t\|g_k\|^2$, *where* $t>0$, *there is the constant* $c>0$ *such that* $-c\|g_k\|^2\le g_k^T d_k$ *and* $c_3\|g_k\|^2\le\|d_k\|^2\le c_2\|g_k\|^2$, *where* $c_2,c_3>0$. *Then the sequence generated by ACGSYS converges linearly to* x^*, *solution to the problem (1.1).*

Proof. From (4.9) we have that $f(x_{k+1})\le f(x_k)$ for all k. Since f is bounded from below, it follows that

$$\lim_{k\to\infty}(f(x_k)-f(x_{k+1}))=0.$$

Now, since f is uniformly convex there exist positive constants m and M, such that $mI\le\nabla^2 f(x)\le MI$ on S. Suppose that $x_k+\alpha d_k\in S$ and $x_k+\gamma_m\alpha d_k\in S$ for all $\alpha>0$. We have:

$$f(x_k+\gamma_m\alpha d_k)\le f(x_k+\alpha d_k)-\frac{(a_k+b_k)^2}{2b_k}. \tag{5.1}$$

But, from uniform convexity we have the following quadratic upper bound on $f(x_k+\alpha d_k)$:

$$f(x_k+\alpha d_k)\le f(x_k)+\alpha g_k^T d_k+\frac{1}{2}M\alpha^2\|d_k\|^2.$$

Therefore,

$$\begin{aligned} f(x_k+\alpha d_k)&\le f(x_k)-\alpha t\|g_k\|^2+\frac{1}{2}Mc_2\alpha^2\|g_k\|^2\\ &=f(x_k)+\left[-t\alpha+\frac{1}{2}Mc_2\alpha^2\right]\|g_k\|^2. \end{aligned}$$

Observe that for $0\le\alpha\le t/(Mc_2)$, $-t\alpha+\frac{1}{2}Mc_2\alpha^2\le-\frac{t}{2}\alpha$ which follows from the convexity of $-t\alpha+(Mc_2/2)\alpha^2$. Using this result we get:

$$f(x_k+\alpha d_k)\le f(x_k)-\frac{t}{2}\alpha\|g_k\|^2\le f(x_k)-\rho t\alpha\|g_k\|^2, \tag{5.2}$$

since $\rho<1/2$.

From proposition 4.1 the Wolfe line search terminates with a value $\alpha\ge\omega>0$. Therefore, for $0\le\alpha\le t/(Mc_2)$, this provides a lower bound on the decrease in the function f, i.e.

$$f(x_k+\alpha d_k)\le f(x_k)-\rho t\omega\|g_k\|^2. \tag{5.3}$$

On the other hand,

$$\frac{(a_k+b_k)^2}{2b_k}\ge\frac{\left(-c\|g_k\|^2+\omega mc_3\|g_k\|^2\right)^2}{2Mc_2\|g_k\|^2}=\frac{(\omega mc_3-c)^2}{2Mc_2}\|g_k\|^2. \tag{5.4}$$

Considering (5.3) and (5.4) from (5.1) we get:

$$f(x_k+\gamma_m\alpha d_k)\le f(x_k)-\rho t\omega\|g_k\|^2-\frac{(\omega mc_3-c)^2}{2Mc_2}\|g_k\|^2. \tag{5.5}$$

Therefore

$$f(x_k) - f(x_k + \gamma_m \alpha d_k) \geq \left[\rho t \omega + \frac{(\omega m c_3 - c)^2}{2Mc_2} \right] \|g_k\|^2 .$$

But, $f(x_k) - f(x_{k+1}) \to 0$ and as a consequence g_k goes to zero, i.e. x_k converges to x^*. Having in view that $f(x_k)$ is a nonincreasing sequence, it follows that $f(x_k)$ converges to $f(x^*)$. From (5.5) we see that

$$f(x_{k+1}) \leq f(x_k) - \left[\rho t \omega + \frac{(\omega m c_3 - c)^2}{2Mc_2} \right] \|g_k\|^2 . \tag{5.6}$$

Combining this with $\|g_k\|^2 \geq 2m(f(x_k) - f^*)$ and subtracting f^* from both sides of (5.6) we conclude:

$$f(x_{k+1}) - f^* \leq \kappa(f(x_k) - f^*),$$

where

$$\kappa = 1 - 2m \left[\rho t \omega + \frac{(\omega m c_3 - c)^2}{2Mc_2} \right] < 1.$$

Therefore, $f(x_k)$ converges to f^* at least as fast as a geometric series with a factor that depends on the parameter ρ in the first Wolfe condition, the bounds m and M, and the parameter t introduced in the descent condition. Hence, the convergence of the acceleration scheme is at least linear. ■

6. Numerical results and comparisons

In this section we report some numerical results obtained with an implementation of the ACGSYS algorithm. The code is written in Fortran and compiled with f77 (default compiler settings) on a Workstation Intel Pentium 4 with 1.8 GHz. We selected a number of 75 large-scale unconstrained optimization test functions in generalized or extended form [1] (some from CUTE library [8]). For each test function we have taken ten numerical experiments with the number of variables $n = 1000, 2000, ..., 10000$. The algorithm implements the Wolfe line search conditions with $\rho = 0.0001$, $\sigma = \|g_{k+1}\|^2 / (|y_k^T g_{k+1}| + \|g_{k+1}\|^2)$, and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector. If $\sigma < \rho$, then we set $\sigma = 0.8$. If $\Delta_k \geq \varepsilon_m$, where ε_m is epsilon machine, then θ_k and β_k are computed as in (2.6) and (2.7), respectively. Otherwise, set $\theta_k = 1$ and $\beta_k = \|g_{k+1}\|^2 / y_k^T s_k$, i.e. we consider the Dai-Yuan conjugate gradient algorithm [13]. In ACGSYS we set $t = 7/8$ and $u = 0.01$. In all the algorithms we considered in this numerical study the maximum number of iterations is limited to 10000.

The comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal value found by ALG1 and ALG2, for problem $i = 1, \ldots, 750$, respectively. We say that, in the particular problem i, the performance of ALG1 was better than the performance of ALG2 if:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{6.1}$$

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

In the first set of numerical experiments we compare ACGSYS versus Hestenes and Stiefel (HS) ($\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T s_k}$) [19], Dai and Yuan (DY) ($\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$) [13] and versus Polak-Ribière-Polyak (PRP) ($\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{g_k^T g_k}$) [24, 25], conjugate gradient algorithms. Figures 1-3 present the Dolan and Moré [14] CPU performance profile of ACGSYS versus HS, DY and PRP, respectively.

An attractive feature of the Hestenes and Stiefel conjugate gradient algorithm is that the pure conjugacy condition $y_k^T d_{k+1} = 0$ always is satisfied, independent of the line search. However, for an exact line search the convergence properties of the HS method are similar to the convergence properties of the PRP method. Therefore, by Powell's example [26], the HS method with exact line search may not converge for a general nonlinear function. On the other hand, the DY method always generates descent directions, and in [10] Dai established a remarkable property for the DY conjugate gradient algorithm, relating the descent directions to the sufficient descent condition. It is shown that if there exist constants γ_1 and γ_2 such that $\gamma_1 \le \|g_k\| \le \gamma_2$ for all k, then for any $p \in (0,1)$, there exists a constant $c > 0$ such that the sufficient descent condition $g_i^T d_i \le -c\|g_i\|^2$ holds for at least $\lfloor pk \rfloor$ indices $i \in [0, k]$, where $\lfloor j \rfloor$ denotes the largest integer $\le j$. However, the DY method does not satisfy the conjugacy condition.

In contrast, observe that in ACGSYS the search directions are always descent directions and the conjugacy condition always is satisfied independent of the accuracy of the line search.

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

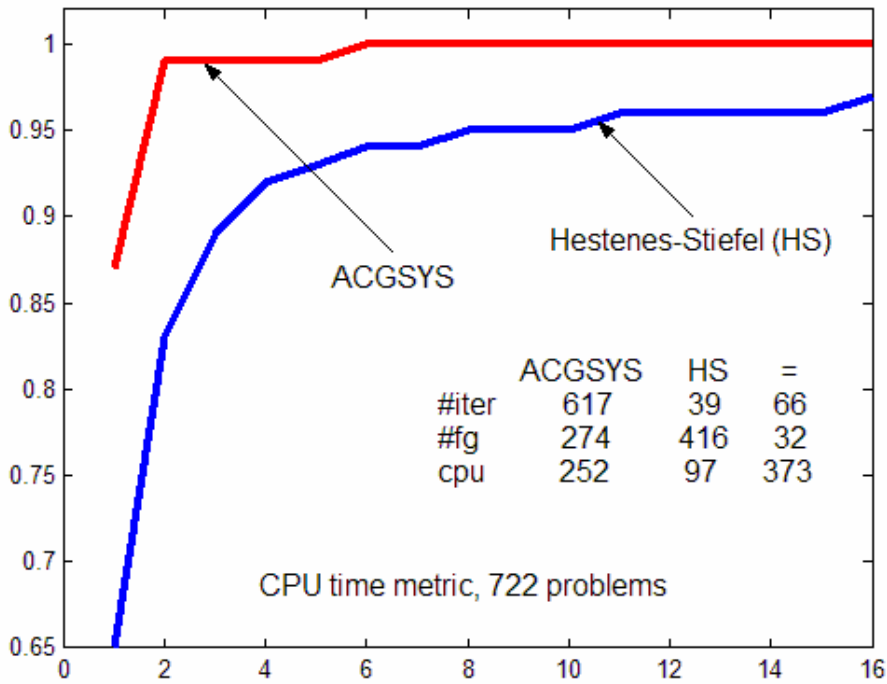

Fig. 1. ACGSYS versus Hestenes-Stiefel.

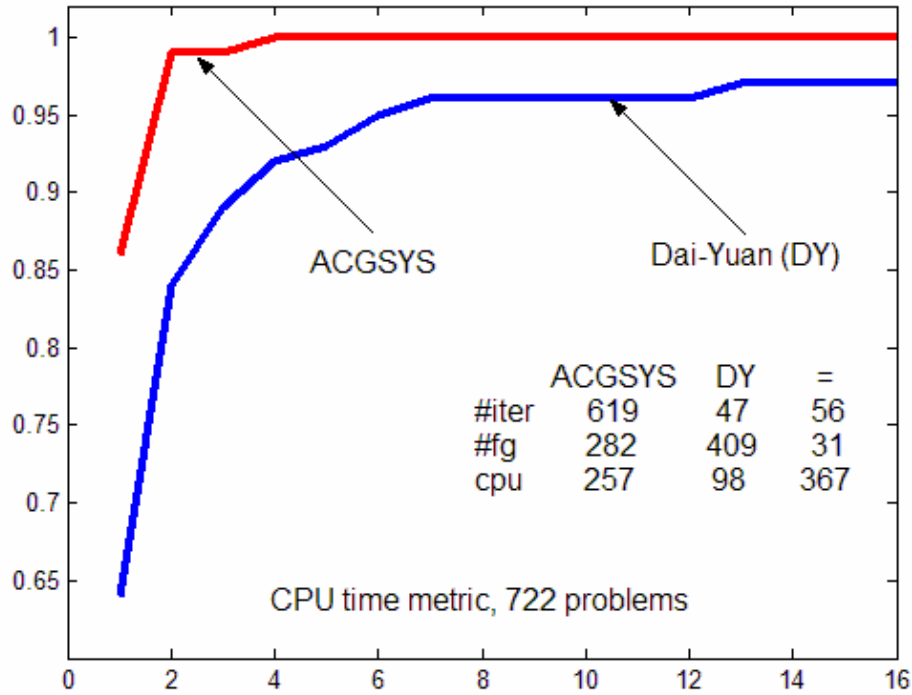

Fig. 2. ACGSYS versus Dai-Yuan.

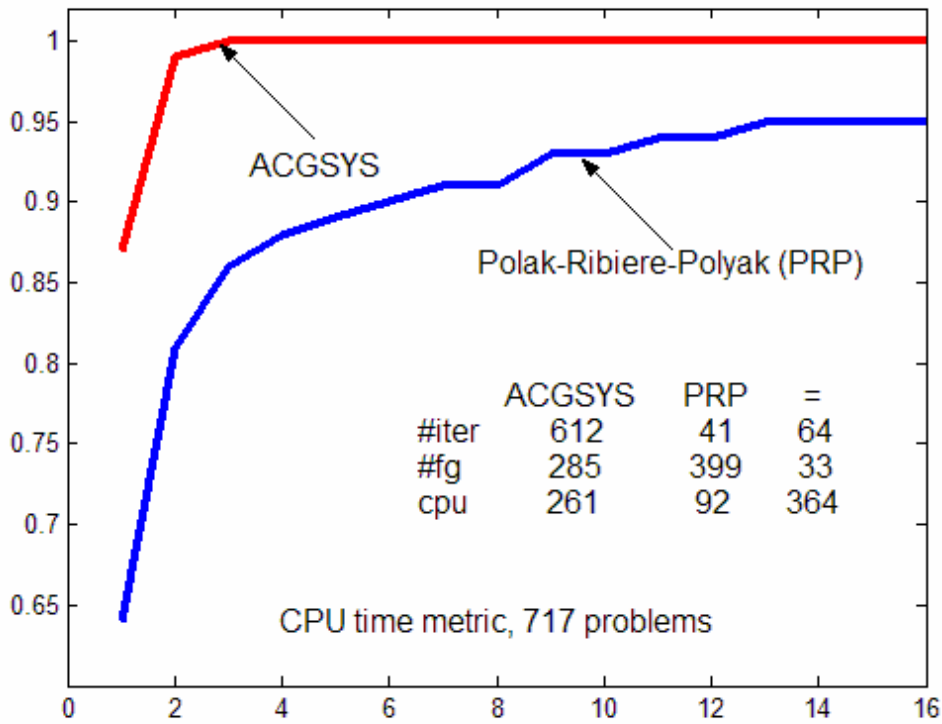

Fig. 3. ACGSYS versus Polak-Ribière-Polyak.

When comparing ACGSYS with these conjugate gradient algorithms subject to CPU time metric we see that ACGSYS is top performer, i.e. the accelerated conjugate gradient algorithm with guaranteed descent and conjugacy conditions is more successful and more

robust than the considered conjugate gradient algorithms. For example, when comparing ACGSYS with Hestene-Stiefel (HS) (see Figure 1), subject to the number of iterations, we see that ACGSYS was better in 617 problems (i.e. it achieved the minimum number of iterations in 617 problems). HS was better in 39 problems and they achieved the same number of iterations in 66 problems, etc. Out of 750 problems, only for 722 problems does the criterion (6.1) hold. Therefore, ACGSYS appears to generate the best search direction and the best steplength, on average.

In the second set of numerical experiments we compare ACGSYS versus hybrid Dai-Yuan ($\beta_k^{hDY} = \max\left\{-c\beta_k^{DY}, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, c = (1-\sigma)/(1+\sigma), \quad \sigma = 0.8$) [13]. The hDY method reduces to the Fletcher and Reeves method [16] if f is a strictly convex quadratic function and the line search is exact. For a standard Wolfe line search, Dai and Yuan [13] proved that it produces descent directions at every iteration and they established the global convergence of their hybrid conjugate gradient algorithm when the Lipschitz assumption holds. However, the hDY conjugate gradient algorithm does not satisfy the conjugacy condition. Figure 4 presents the Dolan and Moré CPU time performance profile of ACGSYS versus hDY. The best performance, relative to the CPU time metric, again was obtained by ACGSYS, the top curve in Figure 4.

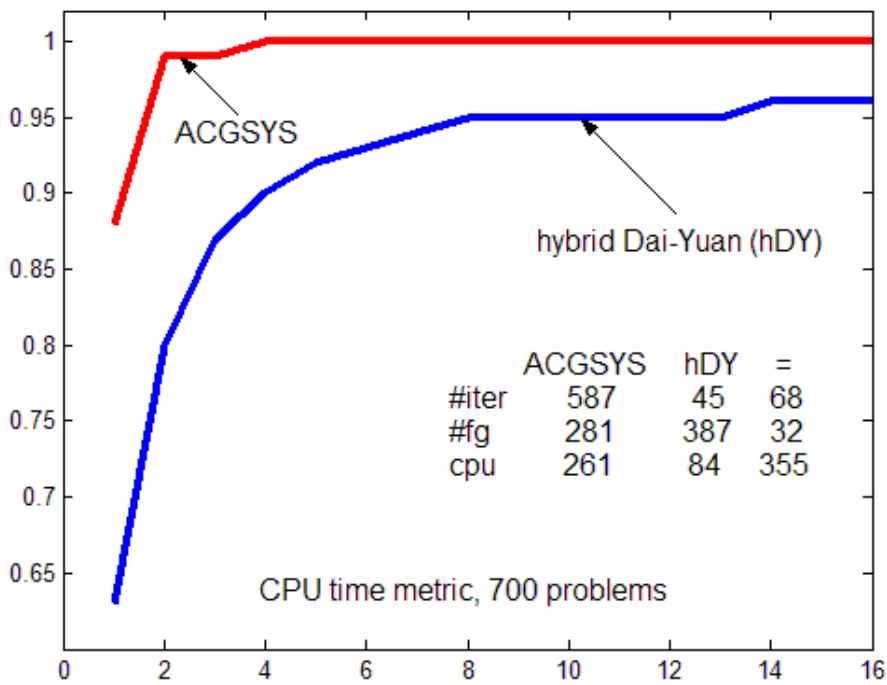

Fig. 4. ACGSYS versus hybrid Dai-Yuan.

In the third set of numerical experiments we compare ACGSYS versus CG_DESCENT by Hager and Zhang [17]. Figure 5 presents the Dolan and Moré CPU time performance profile of ACGSYS versus CG_DESCENT with Wolfe line search. CG_DESCENT was devised in order to ensure sufficient descent, independent of the accuracy of the line search. Hager and Zhang [17] proved that the direction d_k in their algorithm satisfies the sufficient descent condition $g_k^T d_k \le -(7/8)\|g_k\|^2$. This is the main reason we have considered $t = 7/8$ in all our numerical experiments.

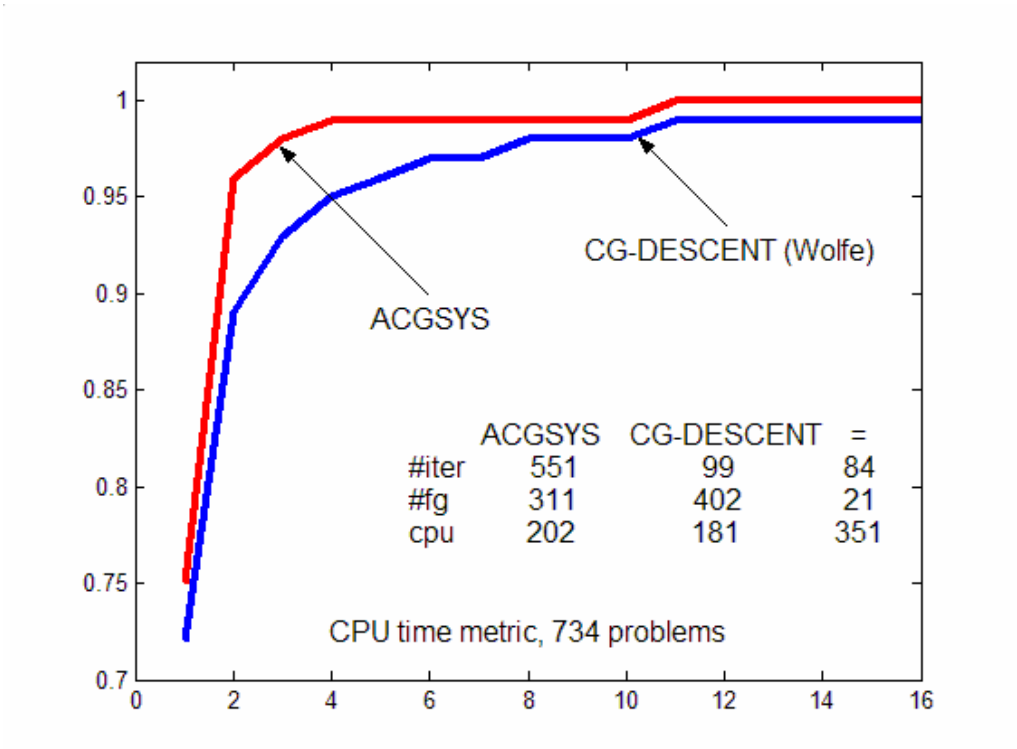

Fig. 5. ACGSYS versus CG_DESCENT by Hager and Zhang.

However, in CG_DESCENT the directions do not satisfy the conjugacy condition (1.11). Again, the best performance, relative to the CPU time metric, was obtained by ACGSYS, the top curve in Figure 5.

In the last set of numerical experiments we present a sensitivity study of the ACGSYS subject to the variation of parameter u. Table 1 presents the total number of iterations (#itert), the total number of function and its gradient evaluations (#fgt) and the total CPU time (#cput) for solving the above set of 750 unconstrained optimization problems for $t = 7/8$ and for different values of u. For example, for solving the set of 750 problems with $t = 7/8$ and $u = 0.001$, the total number of iteration is 284678, the total number of function and its gradient evaluations is 696921 and the total CPU time is 322.82 seconds, etc.

Table 1. Sensitivity of the algorithm subject to u. $t = 7/8$.

u	#itert	#fgt	#cput
0.001	284678	696921	322.82
0.005	281919	702663	325.40
0.01	281197	714022	345.83
0.05	288053	684277	304.03
0.1	279370	687177	313.06
0.5	286722	685759	316.16
0	297382	721890	329.38
1	300556	713043	334.45
5	285184	677952	301.60
10	281560	677557	306.82
50	283533	663013	304.46

In section 3 we argued that the ACGSYS algorithm is not sensitive to the variation of u. In Table 1 we have a computational evidence of this behavior of ACGSYS corresponding to a set of 11 numerical experiments. For example, subject to the CPU time metric we see that the average of the total CPU time corresponding to these 11 numerical experiments for solving 750 problems in each experiment is 3504/11=318.546 seconds. The largest deviation is of

27.29 seconds and corresponds to the numerical experiment in which $u = 0.01$. Therefore, in all these 11 numerical experiments the maximum deviation is of 27.29/750=0.0363 seconds per problem.

7. Conclusions

For solving large scale unconstrained optimization problems we have presented a new conjugate gradient algorithm that for all $k \geq 0$ both the descent and the conjugacy conditions are guaranteed. In our algorithm the search direction is selected as a linear combination of $-g_{k+1}$ and s_k, where the coefficients in this linear combination are selected in such a way that both the descent and the conjugacy condition are satisfied at every iteration. Besides, in our algorithm the step length is modified by an acceleration scheme which proved to be very efficient in reducing the values of the minimizing function along the iterations. For a test set consisting of 750 problems with dimensions ranging between 1000 and 10,000, the CPU time performance profiles of ACGSYS was higher than those of HS, PRP, DY, hDY and CG_DESCENT with Wolfe line search. At present ACGSYS is the fastest conjugate gradient algorithm.

References

[1] N. Andrei, *An unconstrained optimization test functions collection.* Advanced Modeling and Optimization, 10 (2008), pp. 147-161.

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

[3] N. Andrei, *Scaled conjugate gradient algorithms for unconstrained optimization*. Computational Optimization and Applications, 38 (2007), pp. 401-416.

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

[5] N. Andrei, *A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization*. Applied Mathematics Letters, 20 (2007), 645-650.

[6] N. Andrei, *Numerical comparison of conjugate gradient algorithms for unconstrained optimization.* Studies in Informatics and Control, 16 (2007), pp.333-352.

[7] E. Birgin and J.M. Martínez, *A spectral conjugate gradient method for unconstrained optimization*, Applied Math. and Optimization, 43, pp.117-128, 2001.

[8] 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.

[9] S. Boyd, L. Vandenberghe, *Convex optimization.* Cambridge University Press, 2004.

[10] Y.H. Dai, *New properties of a nonlinear conjugate gradient method.* Numer. Math., 89 (2001), pp.83-98.

[11] Y.H. Dai and L.Z. Liao, *New conjugacy conditions and related nonlinear conjugate gradient methods*. Applied Mathematical Optimization, 43 (2001), pp. 87-101.

[12] Y.H. Dai, Han, J.Y., Liu, G.H., Sun, D.F., Yin, .X. and Yuan, Y., *Convergence properties of nonlinear conjugate gradient methods*. SIAM Journal on Optimization 10 (1999), 348-358.

[13] Y.H. Dai, Y. Yuan, *An efficient hybrid conjugate gradient method for unconstrained optimization,* Ann. Oper. Res., 103 (2001) 33-47.

[14] Dolan, E.D. and Moré, J.J., *Benchmarking optimization software with performance profiles*, Math. Programming **91**, 201-213 (2002)

[15] R. Fletcher, *Practical Optimization:Vol. 1: Unconstrained optimization.* John Wiley and Sons, Chichester, 1980.

[16] R. Fletcher and C.M. Reeves, *Function minimization by conjugate gradients.* Computer Journal 7 (1964), pp. 149-154.

[17] W.W. Hager and H. Zhang, "*A new conjugate gradient method with guaranteed descent and an efficient line search*", SIAM Journal on Optimization, 16 (2005) 170-192.

[18] W.W. Hager, H. Zhang, *A survey of nonlinear conjugate gradient methods.* Pacific journal of Optimization, 2 (2006) 35-58.
[19] M.R. Hestenes, E.L. Stiefel, *Methods of conjugate gradients for solving linear systems,* J. Research Nat. Bur. Standards, 49 (1952) 409-436.
[20] D.C. Liu, J. Nocedal, *On the limited memory BFGS method for large scale optimization methods.* Mathematical Programming, 45 (1989), pp. 503-528.
[21] J. Nocedal, *Conjugate gradient methods and nonlinear optimization.* In Linear and nonlinear Conjugate Gradient related methods, L. Adams and J.L. Nazareth (eds.), SIAM, 1996, pp.9-23.
[22] A. Perry, *A modified conjugate gradient algorithm.* Operations Research 26 (1978), pp. 1073-1078.
[23] E. Polak, *Computational methods in optimization: A unified approach.* Academic Press, New York, 1971.
[24] E. Polak, G. Ribière, *Note sur la convergence de directions conjuguée,* Rev. Francaise Informat Recherche Operationelle, 3e Année 16 (1969) 35-43.
[25] B.T. Polyak, *The conjugate gradient method in extreme problems.* USSR Comp. Math. Math. Phys., 9 (1969) 94-112.
[26] M.J.D. Powell, *Nonconvex minimization calculations and the conjugate gradient method.* Numerical Analysis (Dundee, 1983), Lecture Notes in Mathematics, vol. 1066, Springer-Verlag, Berlin, 1984, pp. 122-141.
[27] M.J.D. Powell, *Some convergence properties of the conjugate gradient method.* Mathematical Ptrogramming, 11 (1976), pp.42-49.
[28] D.F. Shanno, *Conjugate gradient methods with inexact searches.* Mathematics of Operations Research 3 (1978), pp. 244-256.
[29] D.F. Shanno, K.H. Phua, *Algorithm 500, Minimization of unconstrained multivariate functions,* ACM Trans. on Math. Soft., 2 (1976) 87-94.
[30] Y. Yuan, *Analysis on the conjugate gradient method.* Technical Report, Computing Center, Academia Sinica, China, 1990.
[31] Y. Yuan, J. Stoer, *A subspace study on conjugate gradient algorithms.* Z. Angew. Math. Mech., 75 (1995), pp. 69-77.
[32] P. Wolfe, *Convergence conditions for ascent methods,* SIAM Rev. 11 (1969) 226-235.
[33] P. Wolfe, *Convergence conditions for ascent methods II: some corrections*, SIAM Rev. 13 (1971) 185-188.

December 10, 2008