Accelerated adaptive Perry conjugate gradient algorithms based on the self-scaling memoryless BFGS update

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

January 16, 2017

Abstract. An accelerated adaptive class of nonlinear conjugate gradient algorithms is suggested. The search direction in these algorithms is given by symmetrization of the scaled Perry conjugate gradient direction [A. Perry, *A modified conjugate gradient algorithm*. Operations Research, 26 (1978) 1073-1078], which depends by a positive parameter. The value of this parameter is determined by minimizing the distance between the symmetrical scaled Perry conjugate gradient search direction matrix and the self-scaling memoryless BFGS update by Oren in the Frobenius norm. Two variants of the parameter in the search direction are presented as those given by: Oren and Luenberger [S.S. Oren, D. G. Luenberger, *Self-scaling variable metric (SSVM) algorithms. I. Criteria and sufficient conditions for scaling a class of algorithms*. Management Sci., 20 (1973/74) 845-862] and Oren and Spedicato [S.S. Oren, E. Spedicato, *Optimal conditioning of self-scaling variable metric algorithms*. Math. Program., 10 (1976) 70-90]. The corresponding algorithm, ACGSSV, is equipped with a very well known acceleration scheme of conjugate gradient algorithms. The global convergence of the algorithm is given both for uniformly convex and general nonlinear functions under the exact or the Wolfe line search. Using a set of 800 unconstrained optimization test problems, of different structure and complexity, we prove that selection of the scaling parameter in self-scaling memoryless BFGS update leads to algorithms which substantially outperform the CG-DESCENT, SCALCG, and CONMIN conjugate gradient algorithms, being more efficient and more robust. However, the conjugate gradient algorithm ADCG based on clustering the eigenvalues of the iteration matrix defined by the search direction is more efficient and slightly more robust than our ACGSSV algorithm. By solving five applications from the MINPACK-2 test problem collection with 10^6 variables, we show that the adaptive Perry conjugate gradient algorithms based on the self-scaling memoryless BFGS update, endowed with the acceleration scheme, is top performer versus CG_DESCENT.

Key words: Unconstrained optimization; conjugate gradient algorithms; Wolfe conditions; self-scaling memoryless BFGS update; sufficient descent condition; conjugacy condition; Frobenius norm

MSC: 49M07; 49M10; 65K05; 90C06

1. Introduction

For solving large-scale unconstrained optimization problems

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

where $f : R^n \to R$ is a continuously differentiable function, bounded from below, one of the most elegant, simple and powerful method is the conjugate gradient method. This method is characterized by low memory requirements and strong local and global convergence properties. 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:

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

In (1.3) β_k is known as the conjugate gradient parameter and $g_k = \nabla f(x_k)$. Notice that the standard formulation of conjugate gradient method uses the search direction defined as $d_{k+1} = -g_k + \beta_k d_k$. However, in our paper we consider the search direction d_{k+1} given as in (1.3), where $s_k = x_{k+1} - x_k = \alpha_k d_k$. Since β_k is any scalar, this simple modification of the standard conjugate gradient method does not change the significance of the parameter β_k in (1.3).

Usually, the stepsize α_k is computed to satisfy some line search conditions [4]. In the convergence analyses and implementation of conjugate gradient algorithms the standard Wolfe conditions [5, 6]

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

$$g_{k+1}^T d_k \ge \sigma g_k^T d_k, \tag{1.5}$$

where d_k is a descent direction and $0 < \rho \le \sigma < 1$, often have been considered. Also, the strong Wolfe line search conditions consisting of (1.4) and

$$\left| g_{k+1}^T d_k \right| \le -\sigma g_k^T d_k, \tag{1.6}$$

can be used.

The search direction d_k, assumed to be a descent one plays the main role in these methods. Different conjugate gradient algorithms correspond to different choices for the scalar conjugate gradient parameter β_k [7]. On the other hand the stepsize α_k guarantees the global convergence in some cases and is very important in efficiency.

In an attempt to use quasi-Newton techniques in conjugate gradient algorithms essentially Perry [1] derived the conjugate gradient parameter β_k in (1.3) by equating the conjugate gradient search direction $-g_{k+1} + \beta_k s_k$ to the quasi-Newton direction $-B_{k+1}^{-1} g_{k+1}$, where B_{k+1} is an approximation of the Hessian, i.e.

$$-g_{k+1} + \beta_k s_k = -B_{k+1}^{-1} g_{k+1}. \tag{1.7}$$

After some simple algebraic manipulation from (1.7) we get the Perry's choice for β_k and the corresponding search direction as:

$$\beta_k = \frac{y_k^T g_{k+1} - s_k^T g_{k+1}}{y_k^T s_k}, \tag{1.8}$$

$$d_{k+1} = -\left[I - \frac{s_k y_k^T}{y_k^T s_k} + \frac{s_k s_k^T}{y_k^T s_k} \right] g_{k+1} \equiv -Q_{k+1}^P g_{k+1}. \tag{1.9}$$

Observe that the formal equality (1.7) is only a technical argument to get a value for the conjugate parameter β_k.

If in (1.2) an exact line search direction is performed, then (1.8) is identical to the Hestenes and Stiefel [8] conjugate gradient algorithm. Observe that Q_{k+1}^P is not symmetric and does not satisfy the quasi-Newton (secant) condition. However, the corresponding Perry's direction (1.9) satisfies the Dai and Liao [9] conjugacy condition, $d_{k+1}^T y_k = -u(g_{k+1}^T s_k)$, with $u = 1$. Now, it is worth saying that if the quasi-Newton direction $-B_{k+1}^{-1} g_{k+1}$ is contained into the cone

generated by $-g_{k+1}$ and s_k, then β_k cannot alone ensure the equality (1.7). It is clear that the above condition (1.7) guarantees that $-g_{k+1}+\beta_k s_k$ and the quasi-Newton direction $-B_{k+1}^{-1}g_{k+1}$ are „coincident" and not just collinear [10]. In order to skip over this limitation we introduce an appropriate scaling of the quasi-Newton direction and consider the equality:

$$-g_{k+1}+\beta_k s_k = -\eta_k B_{k+1}^{-1} g_{k+1}, \tag{1.10}$$

where η_k is a positive scalar parameter. As above, from (1.10) equality, after simple algebraic operations we get the *scaled Perry conjugate gradient parameter* and the corresponding search direction as:

$$\beta_k = \frac{y_k^T g_{k+1} - \eta_k s_k^T g_{k+1}}{y_k^T s_k}, \tag{1.11}$$

$$d_{k+1} = -\left[I - \frac{s_k y_k^T}{y_k^T s_k} + \eta_k \frac{s_k s_k^T}{y_k^T s_k} \right] g_{k+1} \equiv -\bar{P}_{k+1} g_{k+1}. \tag{1.12}$$

Observe that $\bar{P}_{k+1}$ in (1.12) is not symmetric and so the known quasi-Newton condition is not satisfied. Therefore, strictly speaking $\bar{P}_{k+1}$ is not a memoryless quasi-Newton update. Now, by adding in $\bar{P}_{k+1}$ from (1.12) the term $-(y_k s_k^T / y_k^T s_k) g_{k+1}$ we force the symmetry, thus obtaining a new search direction as:

$$d_{k+1} = -P_{k+1} g_{k+1}, \tag{1.13}$$

known as the *symmetrical scaled Perry conjugate gradient direction*, where

$$P_{k+1} = I - \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} + \eta_k \frac{s_k s_k^T}{y_k^T s_k}. \tag{1.14}$$

Observe that P_{k+1} is a symmetric matrix, but it does not satisfy the quasi-Newton condition.

In an effort to get efficient conjugate gradient algorithms by forcing the quasi-Newton condition to hold, Shanno [11] and Andrei [12] obtained high performances memoryless BFGS updates and scaled memoryless BFGS preconditioned updates, respectively. The Shanno computational scheme, analyzed in [13], has global convergence for convex functions and inexact line search [14], but in general, it may not converge, even when the line search is exact [15]. However, the Shanno algorithm is convergent if the restarts are used, but the speed of convergence can decrease. On the other hand the computational scheme by Andrei [12], further analyzed in [16], ensures the sufficient descent property for uniformly convex functions and global convergence for general functions under the exact line search. Both algorithms of Shanno and Andrei have good numerical performances being able to solve large-scale unconstrained optimization problems of different structure and complexity. It is worth mentioning here another way of developments for a class of new spectral conjugate gradient methods, which is a modification of the spectral Perry's conjugate gradient method such that it possesses sufficient descent property for any (inexact) line search, presented by Yu in [17] and by Yu, Guan and Chen in [18].

In this paper, we consider another way of developments by *not forcing the quasi-Newton condition to hold*. Instead, we suggest some adaptive choices for the parameter η_k in (1.14) in such a way to reduce the distance between the search direction matrix P_{k+1} and the self-scaled memoryless BFGS update, one of the best variant of the memoryless quasi-Newton methods.

The structure of the paper is as follows. In Section 2 we present a short review of the self-scaled memoryless BFGS update by Oren [19], with Oren and Luenberger [2] and Oren and Spedicato [3] formulae for scaling parameter computation. Section 3 presents accelerated adaptive symmetrical scaled Perry conjugate gradient algorithms based on minimizing the difference between the matrix P_{k+1} and the self-scaling memoryless BFGS update matrix. In Section 4 the global convergence of the algorithm is proved, both for uniformly convex and general nonlinear functions. Section 5 presents numerical results and comparisons of the suggested algorithms versus CG-DESCENT by Hager and Zhang [20], accelerated SCALCG by Andrei [12], CONMIN by Shanno and Phua [21] and ADCG by Andrei [22]. It is proved that this class of algorithms based on a symmetrization of the scaled Perry conjugate gradient direction and on minimizing the distance between this symmetrical scaled Perry conjugate gradient direction matrix and the self-scaling memoryless BFGS update is more efficient and more robust than the conjugate gradient algorithms considered in these studies, at least for this set of numerical experiments.

2. Scaled memoryless BFGS update

As we know the quasi-Newton methods are one of the best methods for solving unconstrained optimization problems. They do not require explicit second order derivatives and they have very good local and global convergence properties [23]. Having an approximation $H_k \approx \nabla^2 f(x_k)^{-1}$ of the inverse Hessian, these methods determine a new approximation $H_{k+1} \approx \nabla^2 f(x_{k+1})^{-1}$ which satisfies the so called secant equation which includes the second order information. The best quasi-Newton method with strong theoretical properties and very favorable numerical performances is BFGS update [4]. This update is given by:

$$H_{k+1} = H_k - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{y_k^T s_k} + \left(1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right)\frac{s_k s_k^T}{y_k^T s_k}. \tag{2.1}$$

In order to improve the performances of this method, the so called *scaled quasi-Newton updates* have been developed [4]. The purpose of these methods is to improve the condition number of the successive approximations to the inverse Hessian by replacing H_k in (2.1) with $t_k H_k$, where $t_k > 0$ is known as the *scaling parameter*. Two very well known and effective formulae for t_k computation are those given by Oren and Luenberger [2]:

$$t_k = \frac{s_k^T H_k^{-1} s_k}{y_k^T s_k}, \tag{2.2}$$

or by Oren and Spedicato [3]:

$$t_k = \frac{y_k^T s_k}{y_k^T H_k y_k}. \tag{2.3}$$

The scaled BFGS update with one of the above parameters (2.2) or (2.3) is called *self-scaling BFGS update* [19].

In order to get an efficient method for solving large-scale problems, at every iteration, the matrix H_k is replaced by the identity matrix thus avoiding saving the matrix H_k. Therefore the *self-scaling memoryless BFGS update* is obtained as:

$$H_{k+1}^{t_k} = t_k I - t_k \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} + \left(1 + t_k \frac{\|y_k\|^2}{y_k^T s_k}\right)\frac{s_k s_k^T}{y_k^T s_k}, \tag{2.4}$$

able to solve large-scale unconstrained optimization problems of different structures and complexities. In this context, the *memoryless* versions of the scaling parameters (2.2) and (2.3) can be written, respectively as:

$$t_k = \frac{\|s_k\|^2}{y_k^T s_k}, \tag{2.5}$$

$$t_k = \frac{y_k^T s_k}{\|y_k\|^2}, \tag{2.6}$$

where $\|.\|$ stands for the Euclidian norm. The self-scaling memoryles BFGS update is given by (2.4), where the scaling parameter t_k is computed as in (2.5) or (2.6). Using another way of developments as those given by Shanno [11] and Andrei [12] in the following we consider an adaptive choice of the parameter η_k in (1.14) based on the self-scaling memoryless BFGS update.

3. Accelerated adaptive Perry conjugate gradient algorithms

Search direction. In this section we deal with an adaptive choice for the parameter η_k in the Perry symmetrical, scaled memoryless iteration matrix P_{k+1} given by (1.14). Having in view that the self-scaling memoryless BFGS update is one of the best quasi-Newton methods and observing the similarity between the structures of the search direction matrix P_{k+1} (1.14) and the self-scaling memoryless BFGS update (2.4), we suggest computing the parameter η_k as solution of the following minimization problem:

$$\min_{\eta_k > 0} \|D_{k+1}\|_F^2, \tag{3.1}$$

where $D_{k+1} = P_{k+1} - H_{k+1}^{t_k}$ and $\|.\|_F$ is the Frobenius matrix norm. Since $\|D_{k+1}\|_F^2 = tr(D_{k+1}^T D_{k+1})$, it follows that the minimization problem (3.1) is equivalent to:

$$\min_{\eta_k > 0} \left(tr(D_{k+1}^T D_{k+1}) \right). \tag{3.2}$$

From (1.14) and (2.4) we have

$$D_{k+1} = \eta_k \frac{s_k s_k^T}{y_k^T s_k} + A_k,$$

where

$$A_k = I - \frac{s_k y_k^T}{y_k^T s_k} - \frac{y_k s_k^T}{y_k^T s_k} - t_k I + t_k \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} - \left(1 + t_k \frac{\|y_k\|^2}{y_k^T s_k} \right) \frac{s_k s_k^T}{y_k^T s_k}.$$

Therefore,

$$\begin{aligned} D_{k+1}^T D_{k+1} &= \left(\eta_k \frac{s_k s_k^T}{y_k^T s_k} + A_k^T \right) \left(\eta_k \frac{s_k s_k^T}{y_k^T s_k} + A_k \right) \\ &= \eta_k^2 \frac{s_k s_k^T s_k s_k^T}{(y_k^T s_k)^2} + \eta_k \frac{s_k s_k^T}{y_k^T s_k} A_k + \eta_k A_k^T \frac{s_k s_k^T}{y_k^T s_k} + A_k^T A_k. \end{aligned}$$

Since $A_k^T A_k$ is independent by η_k, after some simple algebraic manipulations we get:

$$tr\left(D_{k+1}^T D_{k+1} \right) = \eta_k^2 \frac{\|s_k\|^4}{(y_k^T s_k)^2} + 2\eta_k t_k \frac{\|s_k\|^2}{y_k^T s_k} - 2\eta_k \frac{\|s_k\|^4}{(y_k^T s_k)^2} - 2\eta_k t_k \frac{\|y_k\|^2 \|s_k\|^4}{(y_k^T s_k)^3} - 2\eta_k \frac{\|s_k\|^2}{y_k^T s_k}.$$

Since the coefficient of η_k^2 is always positive, it follows that the second degree function defining $tr\left(D_{k+1}^T D_{k+1}\right)$ is always convex. Therefore, the unique solution of the minimization problem (3.2) is given by:

$$\eta_k = 1 + t_k \left[\frac{\|y_k\|^2}{y_k^T s_k} - \frac{y_k^T s_k}{\|s_k\|^2} \right] + \frac{y_k^T s_k}{\|s_k\|^2}. \tag{3.3}$$

Observe that if the line search satisfies the Wolfe conditions (1.4) and (1.5), then for any $k > 0,\ y_k^T s_k > 0.$ From the Wolfe conditions and the inequality $y_k^T s_k / \|s_k\|^2 \le \|y_k\|^2 / y_k^T s_k$, it follows that $\|s_k\|^2 \|y_k\|^2 / (y_k^T s_k)^2 > 1.$ Since $t_k > 0$ for any $k > 0,$ from (3.3) we have that $\eta_k > 1.$ Therefore, from (1.13) and (1.14) our algorithm is given by (1.2) where

$$d_{k+1} = -g_{k+1} + \left[\frac{y_k^T g_{k+1}}{y_k^T s_k} - \eta_k \frac{s_k^T g_{k+1}}{y_k^T s_k} \right] s_k + \frac{s_k^T g_{k+1}}{y_k^T s_k} y_k, \tag{3.4}$$

and η_k is computed as in (3.3). Observe that this is a three term search direction. For an exact line search, g_{k+1} is orthogonal to s_k, i.e. the search direction (3.4) reduces to the Hestenes and Stiefel direction [8].

The parameter η_k given by (3.3) defines a class of algorithms according to the choices of the scaling parameter t_k in (3.3). Selecting $t_k = 1$ we get:

$$\eta_k^1 = 1 + \frac{\|y_k\|^2}{y_k^T s_k}. \tag{3.5}$$

On the other hand, selecting t_k as in (2.5) (Oren-Luenberger) or (2.6) (Oren-Spedicato) respectively we get:

$$\eta_k^{OL} = \frac{\|s_k\|^2 \|y_k\|^2}{(y_k^T s_k)^2} + \frac{y_k^T s_k}{\|s_k\|^2}, \tag{3.6}$$

$$\eta_k^{OS} = 2 - \frac{(y_k^T s_k)^2}{\|s_k\|^2 \|y_k\|^2} + \frac{y_k^T s_k}{\|s_k\|^2}. \tag{3.7}$$

Observe that $\eta_k^1 \ge 1.$ Since, as we said, $\|s_k\|^2 \|y_k\|^2 / (y_k^T s_k)^2 > 1,$ it follows that $\eta_k^{OL} > 1$ and $\eta_k^{OS} > 1.$ Notice that for $t_k = 1,$ from (1.14) and (2.4), $P_{k+1} = H_{k+1}^1,$ i.e. for this value of the scaling parameter the symmetrical scaled Perry search direction matrix P_{k+1} is exactly the self-scaling memoryless BFGS update. Besides, for $\eta_k = \eta_k^1$ the search direction (3.4) satisfies the Dai and Liao [9] conjugacy condition, i.e. $y_k^T d_{k+1} = -(s_k^T g_{k+1}).$ An interesting result is given by the following proposition, showing an optimal property of $\eta_k^1.$

Proposition 3.1. If $t_k = 1,$ *then* $\eta_k = 1 + \dfrac{\|y_k\|^2}{y_k^T s_k}$ *is the unique minimizer of* $\|D_{k+1}\|.$

Proof. For $t_k = 1,$

$$D_{k+1} = \left(\eta_k - 1 - \frac{\|y_k\|^2}{y_k^T s_k} \right) \frac{s_k s_k^T}{y_k^T s_k}.$$

Therefore, having in view the definition of the induced matrix norm [24] we have:

$$\|D_{k+1}\| = \max_{x \in \Omega} \|D_{k+1} x\| = \frac{1}{|y_k^T s_k|} \left\| \left(\eta_k - 1 - \frac{\|y_k\|^2}{y_k^T s_k} \right) s_k \right\| \max_{x \in \Omega} |s_k^T x,|$$

where $\Omega = \{x \in \mathbb{R}^n : \|x\| = 1\}$. But, $\arg\max_{x \in \Omega} |s_k^T x| = \pm \frac{s_k}{\|s_k\|}$. Therefore,

$$\|D_{k+1}\| = \left\| \left(\eta_k - 1 - \frac{\|y_k\|^2}{y_k^T s_k} \right) s_k \right\| \frac{\|s_k\|}{|y_k^T s_k|}.$$

Hence, we can see that $\eta_k^1 = \arg\min \|D_{k+1}\|$. ■

As we have already seen the value of η_k given by (3.3) ensures that the scaled Perry symmetric iteration matrix P_{k+1} is as close as possible to the self-scaling memoryless BFGS update $H_{k+1}^{t_k}$. The search directions of the self-scaling memoryless BFGS algorithm satisfies the descent condition $g_k^T d_k < 0, \quad \forall k \geq 0,$ [4]. Unlike the quasi-Newton methods, in conjugate gradient algorithms the descent condition has a crucial role. Therefore, in the following theorem a value of η_k is determined in such a way that additionally to minimize the Frobenius norm of the D_{k+1} matrix it ensures the descent condition of the search direction (3.4).

Theorem 3.1. *If* $\eta_k \geq 2 \frac{\|y_k\|^2}{y_k^T s_k}$, *then the search direction (3.4) satisfies the descent condition, i.e.* $g_{k+1}^T d_{k+1} < 0, \ \forall k \geq 0.$

Proof. From (3.4) we have:

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 + 2 \frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{y_k^T s_k} - \eta_k \frac{(s_k^T g_{k+1})^2}{y_k^T s_k}. \tag{3.8}$$

Now, using the classical inequality $u^T v \leq \frac{1}{2}\left[\|u\|^2 + \|v\|^2 \right]$, where $u, v \in R^n$ are arbitrary vectors and considering:

$$u = \frac{1}{\sqrt{2}} (y_k^T s_k) g_{k+1}, \qquad v = \sqrt{2} (s_k^T g_{k+1}) y_k, \tag{3.9}$$

we get:

$$\frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{y_k^T s_k} = \frac{(y_k^T g_{k+1})(y_k^T s_k)(s_k^T g_{k+1})}{(y_k^T s_k)^2} = \frac{[1/\sqrt{2}(y_k^T s_k) g_{k+1}]^T [\sqrt{2}(s_k^T g_{k+1}) y_k]}{(y_k^T s_k)^2}$$

$$\leq \frac{\frac{1}{2}\left[\frac{1}{2}(y_k^T s_k)^2 \|g_{k+1}\|^2 + 2(s_k^T g_{k+1})^2 \|y_k\|^2 \right]}{(y_k^T s_k)^2} = \frac{1}{4}\|g_{k+1}\|^2 + \frac{(s_k^T g_{k+1})^2}{(y_k^T s_k)^2} \|y_k\|^2.$$

Hence,

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

Therefore, if $\eta_k \ge 2\|y_k\|^2 / y_k^T s_k$, then $g_{k+1}^T d_{k+1} < 0$, i.e. the search direction satisfies the descent condition. ■

Therefore, using the Theorem 3.1 the following simple adaptive strategy for computing the search direction in our algorithm can be presented. Using (3.3) compute:

$$\bar{\eta}_k = 1 + t_k\left[\frac{\|y_k\|^2}{y_k^T s_k} - \frac{y_k^T s_k}{\|s_k\|^2}\right] + \frac{y_k^T s_k}{\|s_k\|^2}, \tag{3.11}$$

where $t_k = 1$ or it is given by (2.5) or (2.6).

The value of η_k is computed in an adaptive manner as follows:

$$\eta_k = \begin{cases} \bar{\eta}_k, & \text{if } \bar{\eta}_k > 2\|y_k\|^2 / y_k^T s_k, \\ 2\|y_k\|^2 / y_k^T s_k, & \text{otherwise.} \end{cases} \tag{3.12}$$

Stepsize computation and acceleration scheme. In ACGSSV the stepsize α_k is computed using the Wolfe line search conditions (1.4) and (1.5). Conjugate gradient algorithms are characterized by the fact that the stepsize may differ from 1 by two order of magnitude in a very unpredictable manner [25]. They can be larger or smaller than 1 depending on how the problem is scaled. This behavior of the stepsize in conjugate gradient algorithms is in sharp contrast to the Newton, quasi-Newton, or the limited memory quasi-Newton methods, which accept the unit stepsize for most of the final iterations. Therefore, in conjugate gradient algorithms there is more room to change the stepsize given by the Wolfe line search conditions. This is done by the so called the acceleration scheme. A description of the acceleration scheme for the conjugate gradient algorithms is presented in [26]. The idea is as follows. In order to improve the reduction of the function values along the iterations, the acceleration scheme modifies in a multiplicative manner the stepsize α_k computed by the Wolfe line search conditions (1.4) and (1.5). In accelerated algorithm instead of (1.2) the new estimation of the minimum point of (1.1) is computed as

$$x_{k+1} = x_k + \xi_k \alpha_k d_k, \tag{3.13}$$

where

$$\xi_k = -\frac{a_k}{b_k}, \tag{3.14}$$

$a_k = \alpha_k g_k^T d_k$, $b_k = -\alpha_k (g_k - g_z)^T d_k$, $g_z = \nabla f(z)$ and $z = x_k + \alpha_k d_k$. Therefore, if $b_k \ne 0$, then the new estimation of the solution is computed as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, otherwise $x_{k+1} = x_k + \alpha_k d_k$. In [26] it is proved that the acceleration scheme is linear convergent.

With these, using the definitions of g_k, s_k, y_k and the above developments we can present the following class of conjugate gradient algorithms based on the self-scaling memoryless BFGS update.

Algorithm ACGSSV

Step 1. Select the initial starting point x_0 and compute: $f_0 = f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $k = 0$. Select a value for the parameter ε.

Step 2. Test a criterion for stopping the iterations. For example, if $\|g_k\|_\infty \le \varepsilon$, then stop; otherwise continue with step 3.

Step 3. Using the Wolfe line search conditions (1.4) and (1.5) determine the stepsize 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 $\xi_k = -a_k / b_k$ and update the variables as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, 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. Select a variant of the algorithm, i.e. compute the value of the parameter t_k as: $t_k = 1$ or $t_k = \|s_k\|^2 / y_k^T s_k$ or $t_k = y_k^T s_k / \|y_k\|^2$.

Step 8. Compute η_k as in (3.12), where $\bar{\eta}_k$ is computed as in (3.11).

Step 9. Compute the search direction d_{k+1} as in (3.4).

Step 10. Restart criterion. If the restart criterion of Powell $\left|g_{k+1}^T g_k\right| > 0.2\|g_{k+1}\|^2$ is satisfied, then set $d_{k+1} = -g_{k+1}$.

Step 11. Compute the initial guess $\alpha_k = \alpha_{k-1}\|d_{k-1}\| / \|d_k\|$, set $k = k+1$ and continue with step 2. ■

Observe that the algorithm ACGSSV includes three variants according to the value of the scaling parameter η_k, computed as in (3.5), (3.6) or (3.7).

When the Powell restart condition [27] is satisfied, then we restart the algorithm with the negative gradient $-g_{k+1}$. Some more sophisticated reasons for restarting the conjugate gradient algorithms have been proposed in the literature [28]. However, in this paper we are interested in the performance of a conjugate gradient algorithm that uses this restart criterion of Powell associated to a direction determined on the basis of the self-scaling memoryless BFGS update.

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 stepsize crucially affects the practical behavior of the algorithm. At every iteration $k \ge 1$ the starting guess for the stepsize α_k in the line search procedure is computed as $\alpha_{k-1}\|d_{k-1}\| / \|d_k\|$.

ACGSSV *is defined* by the search direction (3.4) and (3.12) where $t_k = 1$ or it is given by (2.5) or (2.6), as well as by the Wolfe line search conditions (1.4) and (1.5) and by the acceleration scheme (3.13) and (3.14). Intensive numerical experiments, presented in Section 5, proved that the acceleration scheme improves the performances of the algorithm subject to the number of iterations, to the number of function and its gradient evaluations, or computing time metrics. It is worth saying that the above acceleration scheme is working in conjugate gradient methods and it is independent by the conjugate gradient search direction [26]. However, it is quite possible that without acceleration a given conjugate gradient algorithm performs poorly, no matter how elaborated the search direction is. Therefore, to get ACGSSV as an efficient and

robust conjugate gradient algorithm the stepsize α_k, computed by means of the Wolfe line search conditions, is modified by the above acceleration procedure.

4. Global convergence analysis

The global convergence analysis of the above algorithms is based on bounding the norm of the search direction (see [29] or [30]). In this section we prove the global convergence of the above algorithms under the following basic *assumptions*:

(i) *The level set* $S = \{x \in R^n : f(x) \le f(x_0)\}$ *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* $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$, *for all* $x, y \in N$.

Since $\{f(x_k)\}$ is a decreasing sequence, it is clear that the sequence $\{x_k\}$ generated by the proposed algorithm ACGSSV is contained in S. Under these assumptions on f there exists a constant $\Gamma \ge 0$ such that $\|\nabla f(x)\| \le \Gamma$ for all $x \in S$. Notice that the assumption that the function f is bounded below is weaker than the usual assumption that the level set is bounded.

The following proposition shows that the Wolfe line search always gives a lower bound for the stepsize α_k.

Proposition 4.1. *Suppose that* d_k *is a descent direction and the gradient* ∇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*

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

Proof. Subtracting $g_k^T d_k$ from both sides of (1.5) and using the Lipschitz continuity we get

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

Since d_k is a descent direction and $\sigma < 1$, we get the conclusion of the proposition ■

The following proposition, often called the Zoutendijk condition, is used to prove the global convergence of the nonlinear conjugate gradient algorithms. Originally, it was obtained by Wolfe [5, 6] and Zoutendijk [31] under the Wolfe line search (1.4) and (1.5). In the following, we shall prove that the Zoutendijk condition holds under the Wolfe line search.

Proposition 4.2. *Suppose that* d_k *is a descent direction and the gradient* ∇f *satisfies the Lipschitz condition* $\|\nabla f(x) - \nabla f(x_k)\| \le L\|x - x_k\|$, *where* L *is a positive constant. If the line search satisfies the Wolfe conditions (1.4) and (1.5), then*

$$\sum_{k \ge 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < +\infty. \tag{4.2}$$

Proof. Using (4.1), from (1.4) we get

$$f(x_k) - f(x_k + \alpha_k d_k) \geq \frac{\rho(1-\sigma)(g_k^T d_k)^2}{L\|d_k\|^2}.$$

Now, combining this inequality with assumption (i) we obtain (4.2), known as Zoutendijk condition. ■

For *uniformly convex functions* we can prove that the norm of the direction d_{k+1} computed as in (3.4) where η_k is computed as in (3.5) or (3.6) or (3.7) is bounded above.

Theorem 4.1. *Suppose that the assumptions (i) and (ii) hold. Consider the algorithm ACGSSV where the search direction d_k is given by (3.4) and η_k is computed as in (3.5) or (3.6) or (3.7). Suppose that α_k is computed by the Wolfe line search (1.4) and (1.5). Suppose that f is a uniformly convex function on S, i.e. there exists a constant $\mu > 0$ such that*

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2 \tag{4.3}$$

for all $x, y \in N$. If the search direction d_k satisfies the descent condition $g_k^T d_k < 0$, $\forall k \geq 0$, then there exists a positive constant M such that for any $k \geq 0$,

$$\|d_k\| \leq M. \tag{4.4}$$

Proof. From Lipschitz continuity we have $\|y_k\| \leq L\|s_k\|$. On the other hand, from uniform convexity it follows that $y_k^T s_k \geq \mu \|s_k\|^2$. Now, from (3.5), (3.6), (3.7), Lipschitz continuity, uniform convexity and the Cauchy-Schwarz inequality we have:

$$\eta_k^1 = 1 + \frac{\|y_k\|^2}{y_k^T s_k} \leq 1 + \frac{L^2 \|s_k\|^2}{\mu \|s_k\|^2} = 1 + \frac{L^2}{\mu},$$

$$\eta_k^{OL} = \frac{\|s_k\|^2 \|y_k\|^2}{(y_k^T s_k)^2} + \frac{y_k^T s_k}{\|s_k\|^2} \leq \frac{\|s_k\|^2 L^2 \|s_k\|^2}{\mu^2 \|s_k\|^4} + \frac{L\|s_k\|^2}{\|s_k\|^2} = \frac{L^2}{\mu^2} + L,$$

$$\eta_k^{OS} = 2 - \frac{(y_k^T s_k)^2}{\|s_k\|^2 \|y_k\|^2} + \frac{y_k^T s_k}{\|s_k\|^2} \leq 2 + \frac{\|y_k\|^2 \|s_k\|^2}{\|s_k\|^2 \|y_k\|^2} + \frac{L\|s_k\|^2}{\|s_k\|^2} = 3 + L.$$

Therefore, η_k in (3.4) is bounded above. From (3.4) using Lipschitz continuity, uniform convexity and the Cauchy-Schwarz inequality we have: $\|d_{k+1}\| \leq \Gamma + 2\frac{L\Gamma}{\mu} + |\eta_k|\frac{\Gamma}{\mu}$. Since η_k in (3.4) is bounded above it follows that $\|d_{k+1}\|$ is bounded above, i.e. $\|d_{k+1}\| \leq M$, where $M \equiv \Gamma + 2\frac{L\Gamma}{\mu} + |\eta_k|\frac{\Gamma}{\mu}$. Therefore, (4.4) is true. ■

Theorem 4.2. *Suppose that the assumptions (i) and (ii) hold. Consider the three term conjugate gradient algorithm (1.2), where for all $k \geq 0$, the search direction d_k given by (3.4) is a descent direction and the stepsize α_k is determined by the strong Wolfe line search conditions. If*

$$\sum_{k \geq 0} \frac{1}{\|d_k\|^2} = \infty, \tag{4.5}$$

then the algorithm converges in the sense that

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{4.6}$$

Proof. We proceed by contradiction. Suppose that (4.6) does not hold, i.e. there exists a constant $\gamma > 0$, such that $\|g_k\| \ge \gamma$ for all $k \ge 0$. From (3.4) we can write:

$$d_{k+1} = -g_{k+1} + \beta_k s_k + \delta_k y_k, \tag{4.7}$$

where

$$\beta_k = \frac{y_k^T g_{k+1}}{y_k^T s_k} - \eta_k \frac{s_k^T g_{k+1}}{y_k^T s_k}, \qquad \delta_k = \frac{s_k^T g_{k+1}}{y_k^T s_k}. \tag{4.8}$$

From (4.7), using Lipschitz continuity and uniform convexity, we have

$$\begin{aligned}\|d_{k+1}\| &= \|-g_{k+1} + \beta_k s_k + \delta_k y_k\| \ge \|\beta_k s_k\| - \|-g_{k+1} + \delta_k y_k\| \\ &\ge \|\beta_k s_k\| - \|g_{k+1}\| - \|\delta_k y_k\| \ge \|\beta_k s_k\| - \|g_{k+1}\| - \left\|\frac{s_k^T g_{k+1}}{y_k^T s_k} y_k\right\| \\ &\ge \|\beta_k s_k\| - \|g_{k+1}\| - \frac{L}{\mu}\|g_{k+1}\| = \|\beta_k s_k\| - \left(1 + \frac{L}{\mu}\right)\|g_{k+1}\|.\end{aligned}$$

Therefore

$$\|\beta_k s_k\| \le \|d_{k+1}\| + \left(1 + \frac{L}{\mu}\right)\|g_{k+1}\|. \tag{4.9}$$

From (4.7) we also have

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 + \beta_k s_k^T g_{k+1} + \delta_k y_k^T g_{k+1}.$$

Therefore, using again Lipschitz continuity and uniform convexity, we have

$$\begin{aligned}\|g_{k+1}\|^2 &= \beta_k s_k^T g_{k+1} + \delta_k y_k^T g_{k+1} - g_{k+1}^T d_{k=1} \le \left|g_{k+1}^T d_{k+1}\right| + \left|\beta_k s_k^T g_{k+1}\right| + \left|\delta_k y_k^T g_{k+1}\right| \\ &= \left|g_{k+1}^T d_{k+1}\right| + \left|\beta_k s_k^T g_{k+1}\right| + \left|\frac{s_k^T g_{k+1}}{y_k^T s_k} y_k^T g_{k+1}\right| \le \left|g_{k+1}^T d_{k+1}\right| + \left|\beta_k s_k^T g_{k+1}\right| + \frac{L}{\mu}\|g_{k+1}\|^2.\end{aligned}$$

Therefore

$$\left(1 - \frac{L}{\mu}\right)\|g_{k+1}\|^2 \le \left|g_{k+1}^T d_{k+1}\right| + \left|\beta_k s_k^T g_{k+1}\right|. \tag{4.10}$$

Now, let us define

$$t_k = \frac{\left|g_k^T d_k\right|}{\|d_k\|}. \tag{4.11}$$

Hence, from (4.10) we obtain

$$\left(1 - \frac{L}{\mu}\right)\frac{\|g_{k+1}\|^2}{\|d_{k+1}\|} \le t_{k+1} + |\beta_k|\alpha_k \frac{\left|d_k^T g_{k+1}\right|}{\|d_k\|}\frac{\|d_k\|}{\|d_{k+1}\|}.$$

Now, using the strong Wolfe line search (1.6), we get

$$\left(1 - \frac{L}{\mu}\right)\frac{\|g_{k+1}\|^2}{\|d_{k+1}\|} \le t_{k+1} + \sigma t_k \frac{\|\beta_k s_k\|}{\|d_{k+1}\|}. \tag{4.12}$$

Using (4.9) in (4.12) we get

$$\left(1 - \frac{L}{\mu}\right)\frac{\|g_{k+1}\|^2}{\|d_{k+1}\|} \le t_{k+1} + \sigma t_k \frac{\|d_{k+1}\| + (1 + L/\mu)\|g_{k+1}\|}{\|d_{k+1}\|}.$$

After some simple algebraic manipulations we get

$$\left[\left(1-\frac{L}{\mu}\right)-\sigma t_k\left(1+\frac{L}{\mu}\right)\frac{1}{\|g_{k+1}\|}\right]\frac{\|g_{k+1}\|^2}{\|d_{k+1}\|}\le t_{k+1}+\sigma t_k. \tag{4.13}$$

From Zoutendijk condition (4.2) it follows that

$$\sum_{k\ge 0} t_k^2 < +\infty.$$

Therefore,

$$\lim_{k\to\infty} t_k = 0. \tag{4.14}$$

From Theorem 4.1 we know that $\|d_k\|$ is bounded. Moreover, since $\|g_k\|\ge\gamma$ for all $k\ge 0,$ from (4.13) it follows that

$$\lim_{k\to\infty}\left[1-\frac{L}{\mu}-\sigma t_k\left(1+\frac{L}{\mu}\right)\frac{1}{\|g_{k+1}\|}\right]=0.$$

Therefore, there exists an integer k_0 such that

$$\sigma t_k\left(1+\frac{L}{\mu}\right)\le\frac{1}{2}\left(1-\frac{L}{\mu}\right)\|g_{k+1}\|, \tag{4.15}$$

for all $k\ge k_0.$ Hence, from (4.13) we get

$$\frac{\|g_{k+1}\|^2}{\|d_{k+1}\|}\le\frac{2\mu}{\mu-L}(t_{k+1}+\sigma t_k).$$

Now, from (4.14), we have

$$\sum_{k\ge 0}\frac{\|g_{k+1}\|^2}{\|d_{k+1}\|}\le\frac{2\mu}{\mu-L}\sum_{k\ge 0}(t_{k+1}+\sigma t_k)\le+\infty. \tag{4.16}$$

Since $\|g_{k+1}\|\ge\gamma,$ it follows that (4.16) contradicts (4.5), i.e. $\liminf_{k\to\infty}\|g_k\|=0.$ ■

The following theorem ensures the sufficient descent property of the iterative method (1.2) and (3.4) for the *general nonlinear functions under the exact line search*. This result is necessary to complete the convergence analysis of the algorithm given by (1.2) where the search direction is computed as in (3.4).

Theorem 4.3. *Suppose that the assumptions (i) and (ii) hold for the objective function* f *in (1.1). If in the iterative method (1.2) and (3.4) the exact line search is used, then the search directions satisfy the sufficient descent condition* $g_k^T d_k<-\|g_k\|^2,\ \forall k\ge 0.$

Proof. For $k=0,$ $d_0=-g_0$ and so $g_0^T d_0=-\|g_0\|^2.$ If the exact line search is applied, then $s_k^T g_{k+1}=0,\ \forall k\ge 0.$ Therefore, from (3.8) we have that $g_k^T d_k<-\|g_k\|^2,$ for any $k\ge 1.$ ■

For *general nonlinear functions under Wolfe line search conditions* (1.4) and (1.5) we can prove the convergence by establishing the sufficient descent property of (1.2) and (3.4).

Theorem 4.4. *Suppose that the assumptions (i) and (ii) hold for the objective function* f *in (1.1). Consider the iterative method (1.2) and (3.4), where* $\eta_k=2\|y_k\|^2/y_k^T s_k,$ *and the stepsize* α_k *is determined by the Wolfe line search conditions (1.4) and (1.5). Then,* P_{k+1} *is a nonsingular*

matrix and the search direction (3.4) satisfies the sufficient descent condition $g_k^T d_k \le (-1/2)\|g_k\|^2$.

Proof. Observe that $d_{k+1} = -P_{k+1} g_{k+1}$, where P_{k+1} is given by (1.14). To establish the theorem, at first we show that for all $k \ge 0$, the eigenvalues of P_{k+1} are bounded below by a positive constant. From the second Wolfe condition (1.5) we have that $s_k^T y_k > 0$, and consequently, $s_k \ne 0$ and $y_k \ne 0$. Therefore, there exists a set of mutually orthogonal unit vectors $\{u_k^i\}_{i=1}^{n-2}$ such that

$$s_k^T u_k^i = y_k^T u_k^i = 0, \quad i = 1,\ldots,n-2,$$

which leads to

$$P_{k+1} u_k^i = u_k^i, \quad i = 1,\ldots,n-2.$$

Thus, the vectors u_k^i, $i = 1,\ldots,n-2$, are the eigenvectors of P_{k+1} which correspond to the eigenvalue 1. Now, let λ_{n-1}^k and λ_n^k be the two remaining eigenvalues of P_{k+1}. Since the trace of a square matrix is equal to the sum of its eigenvalues, from (1.14) we have that

$$tr(P_{k+1}) = (n-2) + \eta_k \frac{\|s_k\|^2}{y_k^T s_k} \equiv (n-2) + \lambda_{n-1}^k + \lambda_n^k.$$

Therefore

$$\lambda_{n-1}^k + \lambda_n^k = \eta_k b_k, \text{ where } b_k = \|s_k\|^2 / y_k^T s_k. \tag{4.17}$$

On the other hand, the determinant of square matrix is equal to the product of its eigenvalues. Using the formula of algebra [32]: $\det(I + pq^T + uv^T) = (1 + q^T p)(1 + v^T u) - (p^T v)(q^T u)$, where

$$p = -\frac{y_k - \eta_k s_k}{y_k^T s_k}, \quad q = s_k, \quad u = -\frac{s_k}{y_k^T s_k} \text{ and } v = y_k,$$

from (1.14), after some simple algebraic operations, we have that

$$\det(P_{k+1}) = \eta_k \frac{\|s_k\|^2}{y_k^T s_k} - \frac{\|s_k\|^2 \|y_k\|^2}{(y_k^T s_k)^2} \equiv \lambda_{n-1}^k \lambda_n^k.$$

Therefore

$$\lambda_{n-1}^k \lambda_n^k = \eta_k b_k - a_k, \text{ where } a_k = \|s_k\|^2 \|y_k\|^2 / (y_k^T s_k)^2. \tag{4.18}$$

Observe that $a_k > 1$. If $\eta_k \ge 2\|y_k\|^2 / y_k^T s_k$, then $\det(P_{k+1}) \ge a_k > 1$, i.e. the matrix P_{k+1} is nonsingular.

Now, if $\eta_k = 2\|y_k\|^2 / y_k^T s_k$, then

$$\lambda_{n-1}^k + \lambda_n^k = 2a_k > 0, \tag{4.19}$$

$$\lambda_{n-1}^k \lambda_n^k = a_k > 0. \tag{4.20}$$

Therefore, all the eigenvalues of P_{k+1} are strictly positive. Moreover, since $\det(P_{k+1}) = \lambda_{n-1}^k \lambda_n^k = a_k > 1$, it follows that the matrix P_{k+1} is nonsingular. From the above relations (4.19) and (4.20) the eigenvalues λ_{n-1}^k and λ_n^k satisfy the quadratic equation $\lambda^2 - 2a_k \lambda + a_k = 0$, i.e.

$$\lambda_{n-1}^k = a_k - \sqrt{a_k(a_k - 1)},$$
$$\lambda_n^k = a_k + \sqrt{a_k(a_k - 1)}.$$

Observe that $\lambda_n^k > \lambda_{n-1}^k$. Besides,

$$\lambda_n^k = \frac{a_k}{\lambda_{n-1}^k} \geq \frac{a_k}{\lambda_{n-1}^k + \lambda_n^k} = \frac{a_k}{2a_k} = \frac{1}{2}. \tag{4.21}$$

Now, from (4.21), for all $k \geq 0$ we have

$$g_{k+1}^T d_{k+1} = -g_{k+1}^T P_{k+1} g_{k+1} \leq -\lambda_n^k \|g_{k+1}\|^2 \leq -\frac{1}{2}\|g_{k+1}\|^2 .$$

Therefore, the search direction (3.4) with $\eta_k = 2\|y_k\|^2 / y_k^T s_k$ satisfies the sufficient descent condition $g_{k+1}^T d_{k+1} \leq -c\|g_{k+1}\|^2$, where $c = 1/2$. ■

The Theorem 4.4 is necessary to complete the convergence analysis of the proposed algorithm for general nonlinear functions, as described in [30]. Observe that the Theorem 3.1 proves only the descent character of the search direction (3.4) with $\eta_k \geq 2\|y_k\|^2 / y_k^T s_k$. The Theorem 4.4 proves the sufficient descent character of the search direction (3.4) with $\eta_k = 2\|y_k\|^2 / y_k^T s_k$.

5. Numerical results and comparisons

The ACGSSV algorithm was implemented in double precision Fortran using loop unrolling of depth 5 and compiled with f77 (default compiler settings) and run on a Workstation Intel Pentium 4 with 1.8 GHz. We selected a number of 80 large-scale unconstrained optimization test functions in generalized or extended form presented in [33]. Some of the problems from this collection are taken from [34]. For each test function we have considered 10 numerical experiments with the number of variables increasing as $n = 1000, 2000, \ldots, 10000$. The algorithms compared in this section use the Wolfe line search conditions with cubic interpolation [32], $\rho = 0.0001$, $\sigma = 0.8$ and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector.

When the algorithms are compared we can consider at least two points of view: the first is based on the optimal point generated by the algorithm, and the second one is using the objective function value in this point. Since all the algorithms used and compared in this paper generate local solutions, we compare them by using the point of view based on the objective function value in the point determined by each of the algorithms. Therefore, the comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal value found by ALG1 and ALG2, for problem $i = 1, \ldots, 800$, respectively. We say that, in the particular problem i, the performance of ALG1 was better than the performance of ALG2 if:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{5.1}$$

and the number of iterations (#iter), or the number of function-gradient evaluations (#fg), or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively. Possibly, some other points of view for comparing the algorithms can be used, but in this paper we consider this one. Of course, the test problems where the algorithms do not converge to the same function value, according to criterion (5.1), are discarded from comparisons.

By using the acceleration scheme, ACGSSV algorithm is different from many other conjugate gradient algorithms. We know that the line search acceleration technique can improve the numerical behavior of conjugate gradient algorithms remarkable. It is quite possible that the efficiency and robustness of the accelerated conjugate gradient algorithms are mainly determined

by the acceleration technique and less by the search direction. However, this is not an impediment in comparing algorithms, because we do not compare separately search directions techniques or stepsize procedures. We compare algorithms as a whole including both search directions as well as stepsize computations. Besides, when an algorithm is designed the idea is to endow it with the best known procedures, which implement the algebraic operations for search direction and stepsize computations. In other words, even if from the computational point of view the acceleration scheme change the context of comparisons, we are interested to see the performances of the accelerated adaptive Perry conjugate gradient algorithms ACGSSV versus some other conjugate gradient algorithms with acceleration scheme (SCALCG and ADCG) or without acceleration scheme (CONMIN and CG-DESCENT).

Since, CG-DESCENT [35] is among the best nonlinear conjugate gradient algorithms proposed in the literature, but not necessarily the best, in the first set of numerical experiments we compare our algorithm ACGSSV versus CG-DESCENT (version 1.4, Wolfe line search, default settings, $\|\nabla f(x_k)\|_\infty \le 10^{-6}$,). Fig. 1 presents the Dolan and Moré's [36] performance profile of ACGSSV versus CG-DESCENT for $t_k = 1$ subject to CPU time metric. Fig. 2 presents the same performance profile of ACGSSV versus CG-DESCENT for $t_k = \|s_k\|^2 / y_k^T s_k$. Similarly, Fig. 3 presents the performance profile of ACGSSV versus CG-DESCENT for $t_k = y_k^T s_k / \|y_k\|^2$.

In these figures, for every $\tau \ge 1$, the performance profile gives the fraction of the test problems that each considered algorithmic variant has a performance within a factor of τ from the best. The left-hand side of the figures gives the percentage of the test problems for which an algorithm is the fastest; the right-hand side gives the percentage of the test problems that are successfully solved by these algorithms. Mainly, the left-hand side is a measure of the efficiency of an algorithm; the right-hand side is a measure of the robustness of an algorithm. Clearly, the top curve corresponds to the algorithm that solved the most problems in a time that was within a factor τ of the best time.

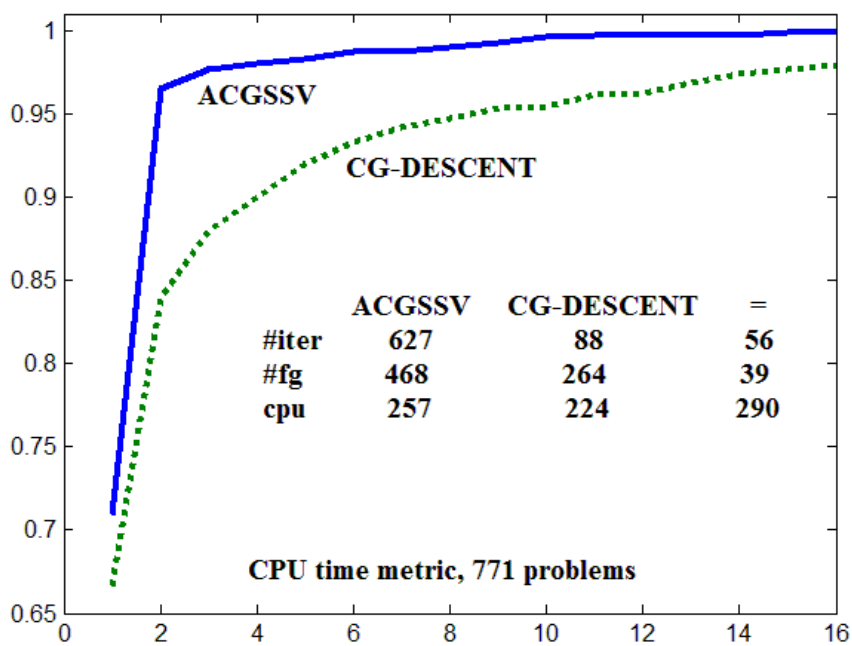

Fig. 1. ACGSSV with $t_k = 1$ versus CG-DESCENT.

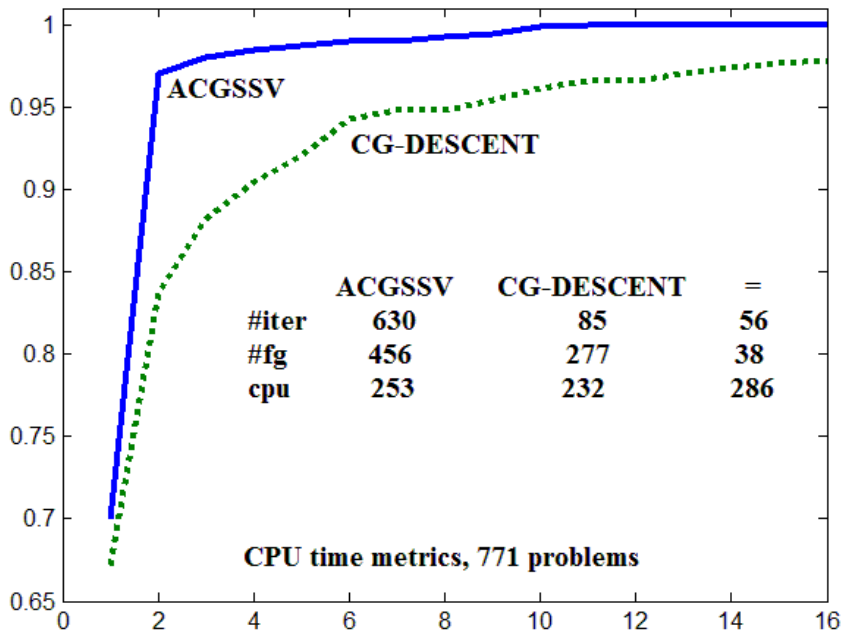

Fig. 2. ACGSSV with $t_k = \|s_k\|^2 / y_k^T s_k$ versus CG-DESCENT.

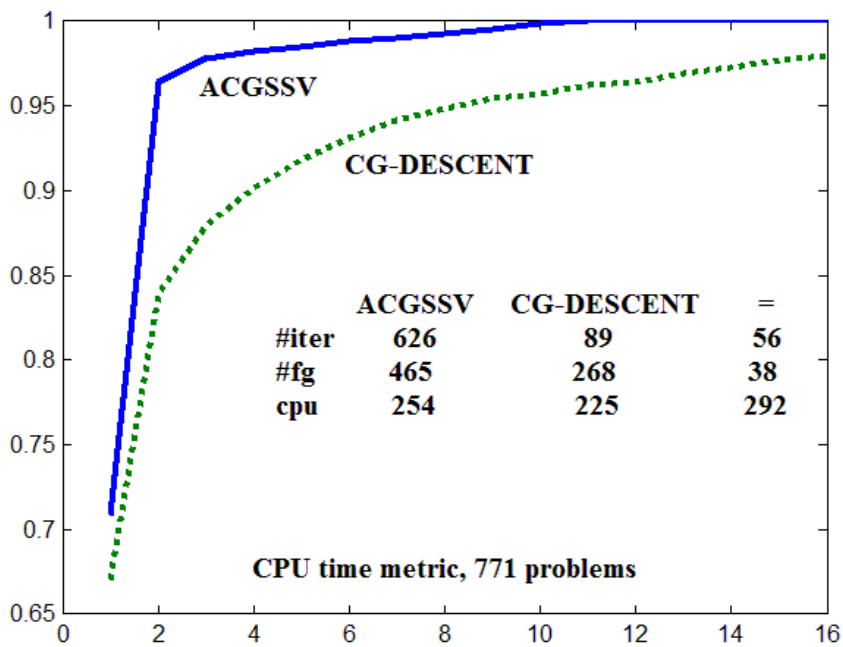

Fig. 3. ACGSSV with $t_k = y_k^T s_k / \|y_k\|^2$ versus CG-DESCENT.

When comparing ACGSSV versus CD-DESCENT for $t_k = 1$, from Fig. 1, we see that subject to the number of iterations ACGSSV was better in 627 problems (i.e. it achieved the minimum number of iterations for solving 627 problems), CG-DESCENT was better in 88 problems and they achieved the same number of iterations in 56 problems, etc. Out of 800 problems considered in these numerical experiments, only for 771 problems does the criterion (5.1) hold. From Figs. 2 and 3 we see the same behavior of ACGSSV versus CG-DESCENT. In fact, the differences among these three variants of the algorithm ACGSSV determined by η_k^1, η_k^{OL} or η_k^{OS} are very small. However, intensive numerical experiments show that the variant of ACGSSV with η_k^1 is slightly more efficient and the variant of the algorithm with η_k^{OL} is slightly more robust. We see that this computational scheme based on the minimizing the distance between the symmetrical scaled Perry conjugate gradient search direction matrix and the self-scaling memoryless BFGS update lead us to algorithms which substantially outperform the CG-DESCENT, being way more efficient and more robust.

Since all these three variants of ACGSSV algorithm (with η_k^1, η_k^{OL} or η_k^{OS}) have similar performances, in the second set of numerical experiments we compare ACGSSV with $t_k = \|s_k\|^2 / y_k^T s_k$ versus SCALCG (spectral, accelerated) [12] and versus CONMIN [21], respectively. In Figures 4 and 5 we present the Dolan and Moré performance profiles of ACGSSV versus the BFGS memoryless preconditioned conjugate gradient algorithms SCALCG and CONMIN.

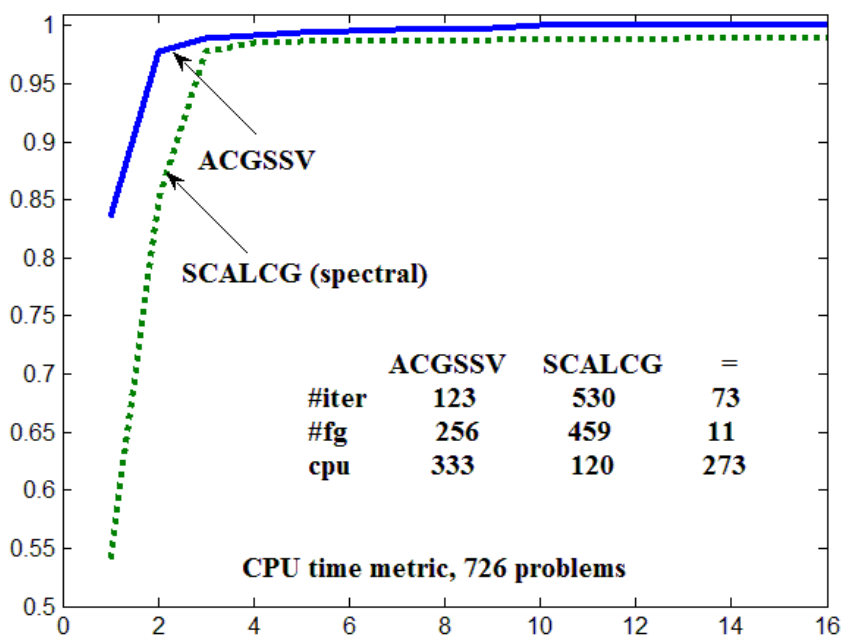

Fig. 4. ACGSSV with $t_k = \|s_k\|^2 / y_k^T s_k$ versus SCALCG.

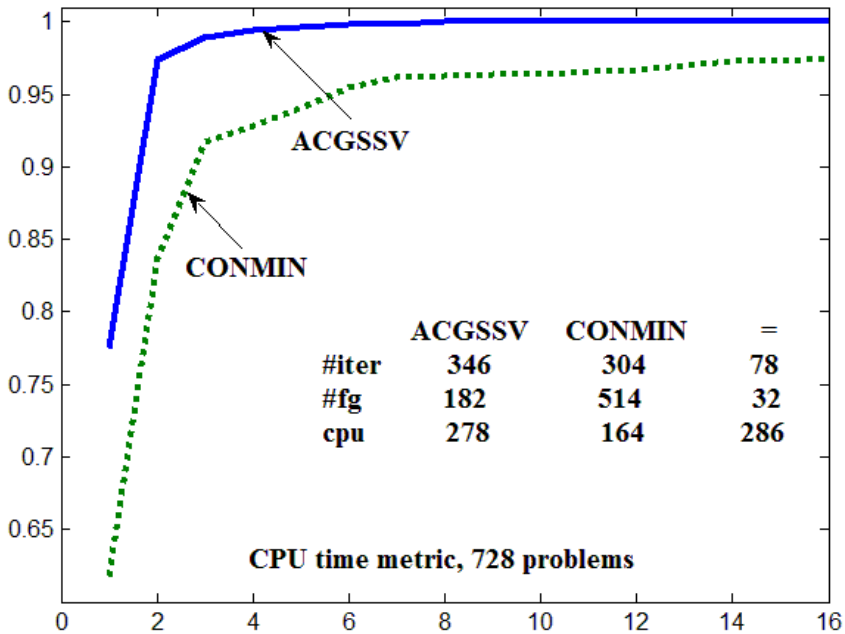

Fig. 5. ACGSSV with $t_k = \|s_k\|^2 / y_k^T s_k$ versus CONMIN.

The search direction in CONMIN by Shanno [11] is exactly the Beale-Powell restart memoryless BFGS quasi-Newton method, where the approximation to the inverse Hessian is reinitialized as the identity matrix at every step. On the other hand, the search direction in SCALCG by Andrei [12] is a scaled memoryless BFGS preconditioned conjugate gradient algorithm, which mainly is a double, positive definite quasi-Newton update scheme using the restart philosophy of Beale-Powell. Both SCALCG and CONMIN are very close to the memoryless quasi-Newton methods. In Figures 4 and 5 we have the numerical evidence that the symmetrical scaled Perry conjugate gradient algorithm ACGSSV based on the self-scaling memoryless BFGS update is far away more efficient and more robust.

In the third set of numerical experiments we compare ACGSSV versus the adaptive conjugate gradient algorithm ADCG [22]. The search direction in ADCG is computed as

$$d_{k+1}^{ADCG} = -g_{k+1} + \left[\frac{y_k^T g_{k+1}}{y_k^T s_k} - t_k \frac{s_k^T g_{k+1}}{y_k^T s_k} \right] s_k - \frac{s_k^T g_{k+1}}{y_k^T s_k} y_k, \tag{5.2}$$

where the parameter t_k is computed in an adaptive manner as:

$$t_k = \begin{cases} 2\sqrt{\tau - 1}\dfrac{\|y_k\|}{\|s_k\|}, & \text{if} \quad \dfrac{\|y_k\|^2 \|s_k\|^2}{(y_k^T s_k)^2} \geq \tau, \\ 0, & \text{otherwise}, \end{cases} \tag{5.3}$$

by clustering the eigenvalues of the matrix defining it, and $\tau > 1$ is a positive constant. The stepsize is computed using the classical Wolfe line search conditions, and is modified by an acceleration technique exactly as in ACGSSV algorithm. Figure 6 presents the performance profile of ACGSSV with $t_k = \|s_k\|^2 / y_k^T s_k$ versus ADCG with $\tau = 3$.

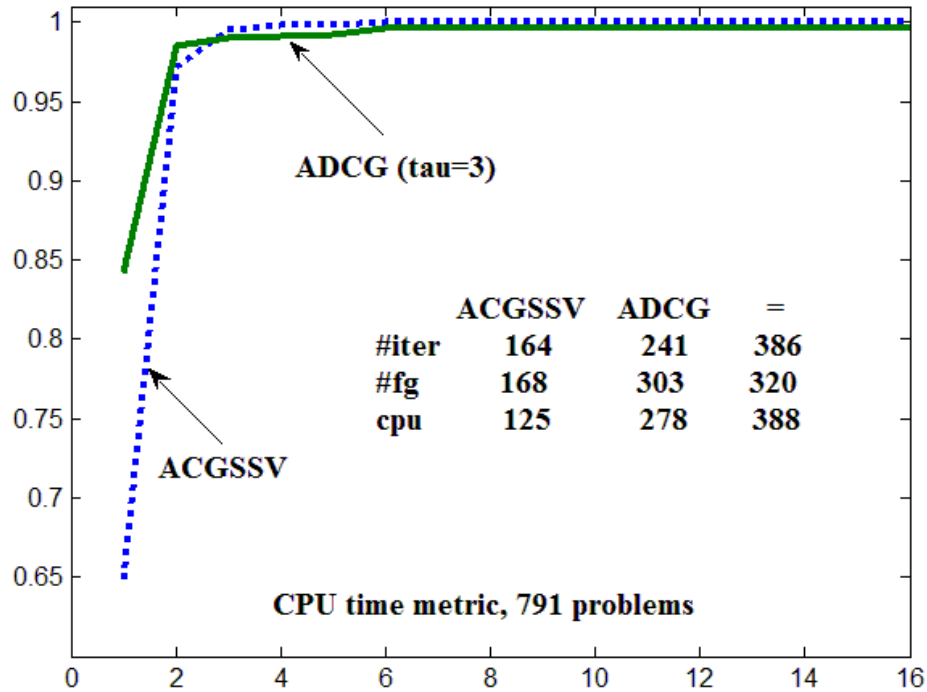

Fig.6. ACGSSV with $t_k = \|s_k\|^2 / y_k^T s_k$ versus ADCG with $\tau = 3$.

Observe that the search direction of the ACGSSV conjugate gradient algorithm given by (3.4) is very similar to the search direction corresponding to ADCG. The sign of the third term in (5.2) is modified in order to ensure the descent property of d_{k+1}^{ADCG}. However, the parameter η_k in (3.4) is computed by minimizing the difference between the symmetric matrix P_{k+1} of Perry (1.14) and the self-scaling memoryless BFGS update matrix (2.4). On the other hand, ADCG belongs to another class of conjugate gradient algorithms. The parameter t_k in (5.2) is computed by clustering the eigenvalues of the matrix defining the search direction (5.2). From Figure 6 we have computational evidence that ADCG conjugate gradient algorithm based on clustering the eigenvalues of the iteration matrix defined by the search direction is clearly more efficient than ACGSSV which is based on the self-scaling memoryless BFGS update. However, ACGSSV is slightly more robust than ADCG. The difference between ACGSSV and ADCG is substantial. ACGSSV tries to improve the condition number of the successive approximations to the inverse Hessian H_{k+1} by scaling this matrix as $t_k H_k$, where $t_k > 0$ is the scaling parameter given by Oren and Luenberger [2], or Oren and Spedicato [3]. On the other hand, ADCG tries to improve the condition number of H_{k+1} by clustering its eigenvalues. Clustering the eigenvalues of the iteration matrix determined by the search direction is one of the most important ingredients in designing efficient conjugate gradient algorithms [22, 37].

In the last set of numerical experiments we present comparisons between ACGSSV with $t_k = \|s_k\|^2 / y_k^T s_k$ versus CG-DESCENT conjugate gradient algorithms for solving some applications from the MINPACK-2 test problem collection [38]. In Table 1 we present these applications, as well as the values of their parameters.

Table 1
Applications from the MINPACK-2 collection.

A1	Elastic–plastic torsion [39, pp. 41–55], $c=5$
A2	Pressure distribution in a journal bearing [40], $b=10,\ \varepsilon=0.1$
A3	Optimal design with composite materials [41], $\lambda=0.008$
A4	Steady-state combustion [42, pp. 292–299], [43], $\lambda=5$
A5	Minimal surfaces with Enneper conditions [44, pp. 80–85]

The infinite-dimensional version of these problems is transformed into a finite element approximation by triangulation. Thus a finite-dimensional minimization problem is obtained whose variables are the values of the piecewise linear function at the vertices of the triangulation. The discretization steps are $nx=1{,}000$ and $ny=1{,}000$, thus obtaining minimization problems with 1,000,000 variables. A comparison between ACGSSV (Powell restart criterion, $\|\nabla f(x_k)\|_\infty \le 10^{-6}$, $\rho=0.0001$, $\sigma=0.8$, $t_k=\|s_k\|^2/y_k^T s_k$) and CG-DESCENT (version 1.4, Wolfe line search, default settings, $\|\nabla f(x_k)\|_\infty \le 10^{-6}$,) for solving these applications is given in Table 2.

Table 2
Performance of ACGSSV versus CG-DESCENT.
1,000,000 variables. $t_k=\|s_k\|^2/y_k^T s_k$. CPU seconds.

	ACGSSV			**CG-DESCENT**		
	#iter	#fg	cpu	#iter	#fg	Cpu
A1	1113	2257	354.95	1145	2291	474.64
A2	2845	5718	1159.18	3370	6741	1835.51
A3	5906	12087	3609.19	4814	9630	3949.71
A4	1413	2864	2023.27	1802	3605	3786.25
A5	1608	3361	755.75	1225	2451	753.75
TOTAL	12885	26287	**7902.34**	12356	24718	10799.86

From Table 2, we see that, subject to the CPU time metric, the ACGSSV algorithm is top performer and the difference is significant, about **2897.52 seconds** for solving all these five applications. It is worth saying that intensive numerical experiments for solving the applications from MINPACK-2 collection with different values of the parameter t_k (i.e. $t_k=1$ or $t_k=y_k^T s_k/\|y_k\|^2$) mainly yield similar results concerning the numerical performances of ACGSSV algorithm. In all cases, for all these numerical experiments, ACGSSV was top performer versus CG-DESCENT.

The ACGSSV and CG-DESCENT algorithms (and codes) are different in many respects. Both of them use the Wolfe line search (implemented in different manners). However, these algorithms differ in their choice of the search direction and in the acceleration scheme used by ACGSSV. The search direction d_{k+1} given by (3.4) where the parameter η_k is computed as in (3.12) is more elaborate: it is descent and it is as close as possible by the search direction corresponding to the self-scaling memoryless BFGS update. On the other hand, the search direction in CG-DESCENT is a simple ad-hoc modification of the Hestenes and Stiefel algorithm.

6. Conclusions

In the panoply of the conjugate gradient algorithms we placed a new one based on symmetrization of the scaled Perry conjugate gradient direction which depends by a positive parameter. The value of this parameter is obtained by minimizing the distance between the symmetrical scaled Perry conjugate gradient search direction matrix and the self-scaling memoryless BFGS update. The scaling parameter in self-scaling memoryless BFGS update by Oren [19] is selected as those given by Oren and Luenberger [2] or Oren and Spedicato [3]. On the other hand, the parameter in scaled symmetrical Perry search direction matrix is computed in an adaptive manner in such a way to minimize the distance between this direction matrix and the self-scaling memoryless BFGS matrix, and to satisfy the descent condition. To improve the performances of the algorithm an acceleration scheme is included. The global convergence of the corresponding ACGSSV algorithm is proved both for uniformly convex functions and for general nonlinear functions. The numerical experience with ACGSSV, using 800 unconstrained optimization test problems and 5 large-scale applications from MINPACK-2 collection, prove that this adaptive computational scheme is more efficient and more robust than the well known CG-DESCENT [35], SCALCG [12] and CONMIN [21] conjugate gradient algorithms. On the other hand, comparisons of ACGSSV versus the adaptive conjugate gradient algorithm ADCG [22], based on clustering the eigenvalues of the corresponding iteration matrix, show that ADCG is more efficient and slightly more robust. Both these algorithms, ACGSSV and ADCG, are adaptive algorithms in a similar way. However, in ADCG algorithm the parameter is selected to cluster the eigenvalues of the iteration matrix. On the other hand in ACGSSV the parameter is computed to get the search direction as close as possible to the self-scaled memoryless BFGS approximation to the iteration matrix. Both these approaches based on closeness to the self-scaled BFGS approximation to the iteration matrix, and clustering the eigenvalues of the iteration matrix determined by the search direction are important ingredients in designing efficient conjugate gradient algorithms. In the same way of development another conjugate gradient algorithm can be obtained by using a hybridization of the scaling parameter given by Oren and Spedicato [3] and that suggested by Babaie-Kafaki [45].

References

[1] A. Perry, A modified conjugate gradient algorithm. Operations Research 26 (1978) 1073-1078.

[2] S.S. Oren, D. G. Luenberger, Self-scaling variable metric (SSVM) algorithms. I. Criteria and sufficient conditions for scaling a class of algorithms. Management Science 20 (1973/74) 845-862.

[3] S.S. Oren, E. Spedicato, Optimal conditioning of self-scaling variable metric algorithms. Mathematical Programming 10 (1976) 70-90.

[4] N. Andrei, Criticism of the Unconstrained Optimization Algorithms Reasoning. Editura Academiei Române, Bucureşti, 2009.

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

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

[7] W.W. Hager, H. Zhang, A survey of nonlinear conjugate gradient methods. Pacific Journal of Optimization 2 (2006) 35-58.

[8] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradients for solving linear systems, J. Research Nat. Bur. Standards 49 (1952) 409-436.

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

[10] H.D. Sherali, O. Ulular, Conjugate gradient methods using quasi-Newton updates with inexact line search. Journal of Mathematical Analysis and Applications 150 (1990) 359-377.

[11] D.F. Shanno, Conjugate gradient methods with inexact searches. Math. Oper. Res. 3 (1978) 244-256.
[12] N. Andrei, Scaled conjugate gradient algorithms for unconstrained optimization. Computational Optimization and Applications 38 (2007) 401-416.
[13] D.F. Shanno, Globally convergent conjugate gradient algorithms. Mathematical Programming 33 (1985) 61-67.
[14] D.F. Shanno, On the convergence of a new conjugate gradient algorithm. SIAM J. Numer. Anal. 15 (1978) 1247-1257.
[15] M.J.D. Powell, Nonconvex minimization calculations and the conjugate gradient method. In: Lecture Notes in Mathematics vol. 1066, Springer-Verlag, Berlin (1984) 122-141.
[16] S. Babaie-Kafaki, A note on the global convergence theorem of the scaled conjugate gradient algorithms proposed by Andrei. Comput. Optim. Appl. 52 (2012) 409-414.
[17] Yu, G.H., Nonlinear self-scaling conjugate gradient methods for large-scale optimization problems. Ph.D. Thesis, Sun Yat-Sen University, 2007.
[18] Yu, G.H., Guan, L., Chen, W., Spectral conjugate gradient methods with sufficient descent property for large-scale unconstrained optimization. Optimization Methods and Software 23 (2) (2008) 275-293.
[19] S.S. Oren, Self-scaling variable metric (SSVM) algorithms. II. Implementation and experiments. Management Science 20 (1974) 863-874.
[20] W.W. Hager, H. Zhang, A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM Journal on Optimization 16 (2005) 170-192.
[21] D.F. Shanno, K.H. Phua, Algorithm 500, minimization of unconstrained multivariate functions. ACM Trans. Math. Soft. 2 (1976) 87-94
[22] N. Andrei, An adaptive conjugate gradient algorithm for large-scale unconstrained optimization. Journal of Computational and Applied Mathematics 292 (2016) 83-91.
[23] J. Nocedal, S.J. Wright, Numerical optimization. (2nd ed.). Springer Series in Optimizations Research. Springer Science+Business Media, New York, 2006.
[24] C.D. Meyer, Matrix analysis and applied linear algebra. SIAM, Philadelphia, 2000.
[25] 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 9-23.
[26] N. Andrei, Acceleration of conjugate gradient algorithms for unconstrained optimization. Applied Mathematics and Computation 213(2009) 361-369.
[27] M.J.D. Powell, Restart procedures of the conjugate gradient method. Mathematical Programming 2 (1977) 241-254.
[28] Y.H. Dai, L.Z. Liao, D. Li, On restart procedures for the conjugate gradient method. Numerical Algorithms 35 (2004) 249-260.
[29] J.C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization. SIAM Journal on Optimization 2 (1992) 21-42.
[30] Y. Dai, J. Han, G. Liu, D. Sun, H. Yin, Y-X, Yuan. Convergence properties of nonlinear conjugate gradient methods. SIAM Journal on Optimization 10 (1999) 345-358.
[31] Zoutendijk, G., Nonlinear programming, computational methods. In J. Abadie (Ed.), Integer and Nonlinear Programming, North-Holland, (1970), 37-86.
[32] W. Sun, Y.X. Yuan, Optimization Theory and Methods. Nonlinear Programming. Springer Science + Business Media, New York, 2006.
[33] N. Andrei, An unconstrained optimization test functions collection. Advanced Modeling and Optimization 10 (2008) 147-161.
[34] N.I.M. Gould, D. Orban, P.L. Toint, CUTEr: A constrained and unconstrained testing environment, revised. ACM Transactions on Mathematical Software 29 (2003) 373-394.
[35] W.W. Hager, H. Zhang, Algorithm 851: CG-DESCENT, a conjugate gradient method with guaranteed descent. ACM Trans. Math. Softw. 32 (2006) 113-137.

[36] E. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles. Mathematical Programming 91 (2002) 201-213.
[37] N. Andrei, Eigenvalues versus singular values study in conjugate gradient algorithms for large-scale unconstrained optimization. Optimization Methods and Software, 2016 [DOI: 10.1080/10556788.2016.1225211].
[38] B.M. Averick, R.G. Carter, J.J. Moré, G.L. Xue, The MINPACK-2 test problem collection, Mathematics and Computer Science Division, Argonne National Laboratory, Preprint MCS-P153-0692, June 1992.
[39] R. Glowinski, Numerical Methods for Nonlinear Variational Problems, Springer-Verlag, Berlin, 1984.
[40] G. Cimatti, On a problem of the theory of lubrication governed by a variational inequality, Applied Mathematics and Optimization 3 (1977) 227–242.
[41] J. Goodman, R. Kohn, L. Reyna, Numerical study of a relaxed variational problem from optimal design, Computer Methods in Applied Mechanics and Engineering 57 (1986) 107–127.
[42] R. Aris, The Mathematical Theory of Diffusion and Reaction in Permeable Catalysts, Oxford, 1975.
[43] J. Bebernes, D. Eberly, Mathematical Problems from Combustion Theory. In: Applied Mathematical Sciences, 83, Springer-Verlag, 1989.
[44] J.C.C. Nitsche, Lectures on Minimal Surfaces, Vol. 1, Cambridge University Press, 1989.
[45] S. Babaie-Kafaki, A modified scaling parameter for the memoryless BFGS updating formula. Numerical Algorithms 72 (2016) 425-433.