IMA Journal of Numerical Analysis (2006) **26**, 604−627
doi:10.1093/imanum/drl006
Advance Access publication on March 24, 2006

The cyclic Barzilai–Borwein method for unconstrained optimization

YU-HONG DAI[†]
State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, PO Box 2719, Beijing 100080, People's Republic of China

WILLIAM W. HAGER[‡]
Department of Mathematics, University of Florida, Gainesville, FL 32611, USA

KLAUS SCHITTKOWSKI[§]
Department of Computer Science, University of Bayreuth, 95440 Bayreuth, Germany

AND

HONGCHAO ZHANG[¶]
Department of Mathematics, University of Florida, Gainesville, FL 32611, USA

[Received on 2 June 2005; revised on 11 January 2006]

In the cyclic Barzilai–Borwein (CBB) method, the same Barzilai–Borwein (BB) stepsize is reused for m consecutive iterations. It is proved that CBB is locally linearly convergent at a local minimizer with positive definite Hessian. Numerical evidence indicates that when $m > n/2 \geqslant 3$, where n is the problem dimension, CBB is locally superlinearly convergent. In the special case $m = 3$ and $n = 2$, it is proved that the convergence rate is no better than linear, in general. An implementation of the CBB method, called adaptive cyclic Barzilai–Borwein (ACBB), combines a non-monotone line search and an adaptive choice for the cycle length m. In numerical experiments using the CUTEr test problem library, ACBB performs better than the existing BB gradient algorithm, while it is competitive with the well-known PRP+ conjugate gradient algorithm.

Keywords: unconstrained optimization; gradient method; convex quadratic programming; non-monotone line search.

1. Introduction

In this paper, we develop a cyclic version of the Barzilai–Borwein (BB) gradient type method (Barzilai & Borwein, 1988) for solving an unconstrained optimization problem

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

[†] Email: dyh@lsec.cc.ac.cn
[‡] Email: hager@math.ufl.edu
[§] Email: klaus.schittkowski@uni-bayreuth.de
[¶] Email: hzhang@math.ufl.edu

where f is continuously differentiable. Gradient methods start from an initial point x_0 and generate new iterates by the rule

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

$k \geqslant 0$, where $g_k = \nabla f(x_k)^{\mathrm{T}}$ is the gradient, viewed as a column vector, and α_k is a stepsize computed by some line search algorithm.

In the steepest descent (SD) method, which can be traced back to Cauchy (1847), the 'exact steplength' is given by

$$\alpha_k \in \arg\min_{\alpha \in \mathbb{R}} f(x_k - \alpha g_k). \tag{1.3}$$

It is well known that SD can be very slow when the Hessian of f is ill-conditioned at a local minimum (see Akaike, 1959; Forsythe, 1968). In this case, the iterates slowly approach the minimum in a zigzag fashion. On the other hand, it has been shown that if the exact SD step is reused in a cyclic fashion, then the convergence is accelerated. Given an integer $m \geqslant 1$, which we call the cycle length, cyclic SD can be expressed as

$$\alpha_{m\ell+i} = \alpha_{m\ell+1}^{\mathrm{SD}} \quad \text{for } i = 1, \ldots, m, \tag{1.4}$$

$\ell = 0, 1, \ldots$, where α_k^{SD} is the exact steplength given by (1.3). Formula (1.4) was first proposed in Friedlander *et al.* (1999), while the particular choice $m = 2$ was also investigated in Dai (2003) and Raydan & Svaiter (2002). The analysis in Dai & Fletcher (2005a) shows that if $m > \frac{n}{2}$, cyclic SD is likely R-superlinearly convergent. Hence, SD is accelerated when the stepsize is repeated.

Let BB denote the original method of Barzilai & Borwein (1988). In this paper, we develop a cyclic Barzilai–Borwein (CBB) method. The basic idea of Barzilai and Borwein is to regard the matrix $D(\alpha_k) = \frac{1}{\alpha_k} I$ as an approximation of the Hessian $\nabla^2 f(x_k)$ and impose a quasi-Newton property on $D(\alpha_k)$:

$$\alpha_k = \arg\min_{\alpha \in \mathbb{R}} \|D(\alpha) s_{k-1} - y_{k-1}\|_2, \tag{1.5}$$

where $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = g_k - g_{k-1}$ and $k \geqslant 2$. The proposed stepsize, obtained from (1.5), is

$$\alpha_k^{\mathrm{BB}} = \frac{s_{k-1}^{\mathrm{T}} s_{k-1}}{s_{k-1}^{\mathrm{T}} y_{k-1}}. \tag{1.6}$$

Other possible choices for the stepsize α_k include Dai (2003), Dai & Fletcher (2006), Dai & Yang (2001, 2003), Friedlander *et al.* (1999), Grippo & Sciandrone (2002), Raydan & Svaiter (2002) and Serafini *et al.* (2005). In this paper, we refer to (1.6) as the BB formula. The gradient method (1.2) corresponding to the BB stepsize (1.6) is called the BB method.

Due to their simplicity, efficiency and low memory requirements, BB-like methods have been used in many applications. Glunt *et al.* (1993) present a direct application of the BB method in chemistry. Birgin *et al.* (1999) use a globalized BB method to estimate the optical constants and the thickness of thin films, while in Birgin *et al.* (2000) further extensions are given, leading to more efficient projected gradient methods. Liu & Dai (2001) provide a powerful scheme for solving noisy unconstrained optimization problems by combining the BB method and a stochastic approximation method. The projected BB-like method turns out to be very useful in machine learning for training support vector machines (see Serafini *et al.*, 2005; Dai & Fletcher, 2006). Empirically, good performance is observed on a wide variety of classification problems.

The superior performance of cyclic SD, compared to the ordinary SD, as shown in Dai & Fletcher (2005a), leads us to consider the CBB method:

$$\alpha_{m\ell+i} = \alpha^{\mathrm{BB}}_{m\ell+1} \quad \text{for } i = 1, \ldots, m, \tag{1.7}$$

where $m \geqslant 1$ is again the cycle length. An advantage of the CBB method is that for general non-linear functions, the stepsize is given by the simple formula (1.5) in contrast to the non-trivial optimization problem associated with the SD step (1.3).

In Friedlander *et al.* (1999), the authors obtain global convergence of CBB when f is a strongly convex quadratic. Dai (2003) establishes the R-linear convergence of CBB for a strongly convex quadratic. In Section 2, we prove the local R-linear convergence for the CBB method at a local minimizer of a general non-linear function. In Section 3, numerical evidence for strongly convex quadratic functions indicates that the convergence is superlinear if $m > n/2 \geqslant 3$. In the special case $m = 3$ and $n = 2$, we prove that the convergence is at best linear, in general.

In Section 4, we propose an adaptive method for computing an appropriate cycle length, and we obtain a globally convergent non-monotone scheme by using a modified version of the line search developed in Dai & Zhang (2001). This new line search, an adaptive analogue of Toint's (1997) scheme for trust region methods, accepts the original BB stepsize more often than does Raydan's (Raydan, 1997) strategy for globalizing the BB method. We refer to Raydan's globalized BB implementation as the GBB method. Numerical comparisons with the PRP+ algorithm and the SPG2 algorithm (Birgin *et al.*, 2000) (one version of the GBB method) are given in Section 4 using the CUTEr test problem library (Bongartz *et al.*, 1995).

Throughout this paper, we use the following notations. $\|\cdot\|$ is the Euclidean norm of a vector. The subscript k is often associated with the iteration number in an algorithm. The letters i, j, k, ℓ, m and n, either lower or upper case, designate integers. The gradient $\nabla f(x)$ is a row vector, while $g(x) = \nabla f(x)^{\mathrm{T}}$ is a column vector; here T denotes transpose. The gradient at the iterate x_k is $g_k = g(x_k)$. We let $\nabla^2 f(x)$ denote the Hessian of f at x. The ball with centre x and radius ρ is denoted $B_\rho(x)$.

2. Local linear convergence

In this section, we prove R-linear convergence for the CBB method. In Liu & Dai (2001), it is proposed that R-linear convergence for the BB method applied to a general non-linear function could be obtained from the R-linear convergence results for a quadratic by comparing the iterates associated with a quadratic approximation to the general non-linear iterates. In our R-linear convergence result for the CBB method, we make such a comparison.

The CBB iteration can be expressed as

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

where

$$\alpha_k = \frac{s_i^{\mathrm{T}} s_i}{s_i^{\mathrm{T}} y_i}, \quad i = \nu(k), \quad \text{and} \quad \nu(k) = m\lfloor (k-1)/m \rfloor, \tag{2.2}$$

$k \geqslant 1$. For $r \in \mathbb{R}$, $\lfloor r \rfloor$ denotes the largest integer j such that $j \leqslant r$. We assume that f is two times Lipschitz continuously differentiable in a neighbourhood of a local minimizer x^* where the Hessian $H = \nabla^2 f(x^*)$ is positive definite. The second-order Taylor approximation $\hat{f}$ to f around x^* is given by

$$\hat{f}(x) = f(x^*) + \frac{1}{2}(x - x^*)^{\mathrm{T}} H (x - x^*). \tag{2.3}$$

We will compare an iterate x_{k+j} generated by (2.1) to a CBB iterate $\hat{x}_{k,j}$ associated with $\hat{f}$ and the starting point $\hat{x}_{k,0} = x_k$. More precisely, we define

$$\begin{aligned} \hat{x}_{k,0} &= x_k, \\ \hat{x}_{k,j+1} &= \hat{x}_{k,j} - \hat{\alpha}_{k,j}\hat{g}_{k,j}, \qquad j \geqslant 0, \end{aligned} \tag{2.4}$$

where

$$\hat{\alpha}_{k,j} = \begin{cases} \alpha_k & \text{if } \nu(k+j) = \nu(k) \\ \frac{\hat{s}_i^{\mathrm{T}}\hat{s}_i}{\hat{s}_i^{\mathrm{T}}\hat{y}_i}, & i = \nu(k+j), \text{ otherwise.} \end{cases}$$

Here $\hat{s}_{k+j} = \hat{x}_{k,j+1} - \hat{x}_{k,j}$, $\hat{g}_{k,j} = H(\hat{x}_{k,j} - x^*)$ and $\hat{y}_{k+j} = \hat{g}_{k,j+1} - \hat{g}_{k,j}$.

We exploit the following result established in Dai (2003, Theorem 3.2):

LEMMA 2.1 Let $\{\hat{x}_{k,j}\colon j \geqslant 0\}$ be the CBB iterates associated with the starting point $\hat{x}_{k,0} = x_k$ and the quadratic $\hat{f}$ in (2.3), where H is positive definite. Given two arbitrary constants $C_2 > C_1 > 0$, there exists a positive integer N with the following property: For any $k \geqslant 1$ and

$$\hat{\alpha}_{k,0} \in [C_1, C_2], \tag{2.5}$$

$$\|\hat{x}_{k,N} - x^*\| \leqslant \frac{1}{2}\|\hat{x}_{k,0} - x^*\|.$$

In our next lemma, we estimate the distance between $\hat{x}_{k,j}$ and x_{k+j}. Let $B_\rho(x)$ denote the ball with centre x and radius ρ. Since f is two times Lipschitz continuously differentiable and $\nabla^2 f(x^*)$ is positive definite, there exist positive constants ρ, λ and $\Lambda_2 > \Lambda_1$ such that

$$\|\nabla f(x) - H(x - x^*)\| \leqslant \lambda\|x - x^*\|^2 \quad \text{for all } x \in B_\rho(x^*) \tag{2.6}$$

and

$$\Lambda_1 \leqslant \frac{y^{\mathrm{T}}\nabla^2 f(x)y}{y^{\mathrm{T}}y} \leqslant \Lambda_2 \quad \text{for all } y \in \mathbb{R}^n \text{ and } x \in B_\rho(x^*). \tag{2.7}$$

Note that if x_i and $x_{i+1} \in B_\rho(x^*)$, then the fundamental theorem of calculus applied to $y_i = g_{i+1} - g_i$ yields

$$\frac{1}{\Lambda_2} \leqslant \frac{s_i^{\mathrm{T}}s_i}{s_i^{\mathrm{T}}y_i} \leqslant \frac{1}{\Lambda_1}. \tag{2.8}$$

Hence, when the CBB iterates lie in $B_\rho(x^*)$, condition (2.5) of Lemma 2.1 is satisfied with $C_1 = 1/\Lambda_2$ and $C_2 = 1/\Lambda_1$. If we define $g(x) = \nabla f(x)^{\mathrm{T}}$, then the fundamental theorem of calculus can also be used to deduce that

$$\|g(x)\| = \|g(x) - g(x^*)\| \leqslant \Lambda_2\|x - x^*\| \tag{2.9}$$

for all $x \in B_\rho(x^*)$.

LEMMA 2.2 Let $\{x_j : j \geqslant k\}$ be a sequence generated by the CBB method applied to a function f with a local minimizer x^*, and assume that the Hessian $H = \nabla^2 f(x^*)$ is positive definite with (2.7) satisfied. Then for any fixed positive integer N, there exist positive constants δ and γ with the following property: For any $x_k \in B_\delta(x^*)$, $\alpha_k \in [\Lambda_2^{-1}, \Lambda_1^{-1}]$, $\ell \in [0, N]$ with

$$\|\hat{x}_{k,j} - x^*\| \geqslant \frac{1}{2}\|\hat{x}_{k,0} - x^*\| \quad \text{for all } j \in [0, \max\{0, \ell - 1\}], \tag{2.10}$$

we have

$$x_{k+j} \in B_\rho(x^*) \quad \text{and} \quad \|x_{k+j} - \hat{x}_{k,j}\| \leqslant \gamma\,\|x_k - x^*\|^2 \tag{2.11}$$

for all $j \in [0, \ell]$.

Proof. Throughout the proof, we let c denote a generic positive constant, which depends on fixed constants, such as N, Λ_1, Λ_2 or λ, but not on k, the choice of $x_k \in B_\delta(x^*)$ or the choice of $\alpha_k \in [\Lambda_2^{-1}, \Lambda_1^{-1}]$. To facilitate the proof, we also show that

$$\|g(x_{k+j}) - \hat{g}(\hat{x}_{k,j})\| \leqslant c\|x_k - x^*\|^2, \tag{2.12}$$

$$\|s_{k+j}\| \leqslant c\|x_k - x^*\|, \tag{2.13}$$

$$|\alpha_{k+j} - \hat{\alpha}_{k,j}| \leqslant c\|x_k - x^*\|, \tag{2.14}$$

for all $j \in [0, \ell]$, where $\hat{g}(x) = \nabla \hat{f}(x)^{\mathrm{T}} = H(x - x^*)$.

The proof of (2.11)–(2.14) is by induction on ℓ. For $\ell = 0$, we take $\delta = \rho$. Relation (2.11) is trivial since $\hat{x}_{k,0} = x_k$. By (2.6), we have

$$\|g(x_k) - \hat{g}(\hat{x}_{k,0})\| = \|g(x_k) - \hat{g}(x_k)\| \leqslant \lambda\|x_k - x^*\|^2,$$

which gives (2.12). Since $\delta = \rho$ and $x_k \in B_\delta(x^*)$, it follows from (2.9) that

$$\|s_k\| = \|\alpha_k g_k\| \leqslant \frac{\Lambda_2}{\Lambda_1}\|x_k - x^*\|,$$

which gives (2.13). Relation (2.14) is trivial since $\hat{\alpha}_{k,0} = \alpha_k$.

Now, proceeding by induction, suppose that there exist $L \in [1, N)$ and $\delta > 0$ with the property that if (2.10) holds for any $\ell \in [0, L - 1]$, then (2.11)–(2.14) are satisfied for all $j \in [0, \ell]$. We wish to show that for a smaller choice of $\delta > 0$, we can replace L by $L + 1$. Hence, we suppose that (2.10) holds for all $j \in [0, L]$. Since (2.10) holds for all $j \in [0, L - 1]$, it follows from the induction hypothesis and (2.13) that

$$\|x_{k+L+1} - x^*\| \leqslant \|x_k - x^*\| + \sum_{i=0}^{L} \|s_{k+i}\| \leqslant c\|x_k - x^*\|. \tag{2.15}$$

Consequently, by choosing δ smaller, if necessary, we have $x_{k+L+1} \in B_\rho(x^*)$ when $x_k \in B_\delta(x^*)$.

By the triangle inequality,

$$\begin{aligned}\|x_{k+L+1} - \hat{x}_{k,L+1}\| &= \|x_{k+L} - \alpha_{k+L}\,g(x_{k+L}) - [\hat{x}_{k,L} - \hat{\alpha}_{k,L}\,\hat{g}(\hat{x}_{k,L})]\| \\ &\leqslant \|x_{k+L} - \hat{x}_{k,L}\| + |\hat{\alpha}_{k,L}|\|g(x_{k+L}) - \hat{g}(\hat{x}_{k,L})\| \\ &\quad + |\alpha_{k+L} - \hat{\alpha}_{k,L}|\|g(x_{k+L})\|. \end{aligned} \tag{2.16}$$

We now analyse each of the terms in (2.16). By the induction hypothesis, the bound (2.11) with $j = L$ holds, which gives

$$\|x_{k+L} - \hat{x}_{k,L}\| \leqslant c\|x_k - x^*\|^2. \tag{2.17}$$

By the definition of $\hat{\alpha}$, either $\hat{\alpha}_{k,L} = \alpha_k \in [\Lambda_2^{-1}, \Lambda_1^{-1}]$ or

$$\hat{\alpha}_{k,L} = \frac{\hat{s}_i^{\mathrm{T}}\hat{s}_i}{\hat{s}_i^{\mathrm{T}}\hat{y}_i}, \quad i = \nu(k+L).$$

In this latter case,

$$\frac{1}{\Lambda_2} \leqslant \frac{\hat{s}_i^{\mathrm{T}}\hat{s}_i}{\hat{s}_i^{\mathrm{T}}H\hat{s}_i} = \frac{\hat{s}_i^{\mathrm{T}}\hat{s}_i}{\hat{s}_i^{\mathrm{T}}\hat{y}_i} \leqslant \frac{1}{\Lambda_1}.$$

Hence, in either case, $\hat{\alpha}_{k,L} \in [\Lambda_2^{-1}, \Lambda_1^{-1}]$. It follows from (2.12) with $j = L$ that

$$|\hat{\alpha}_{k,L}|\|g(x_{k+L}) - \hat{g}(\hat{x}_{k,L})\| \leqslant \frac{1}{\Lambda_1}\|g(x_{k+L}) - \hat{g}(\hat{x}_{k,L})\| \leqslant c\|x_k - x^*\|^2. \tag{2.18}$$

Also, by (2.14) with $j = L$ and (2.9), we have

$$|\alpha_{k+L} - \hat{\alpha}_{k,L}|\|g(x_{k+L})\| \leqslant c\|x_k - x^*\|\|x_{k+L} - x^*\|.$$

Utilizing (2.15) (with L replaced by $L - 1$) gives

$$|\alpha_{k+L} - \hat{\alpha}_{k,L}|\|g(x_{k+L})\| \leqslant c\|x_k - x^*\|^2. \tag{2.19}$$

We combine (2.16)–(2.19) to obtain (2.11) for $j = L + 1$. Note that in establishing (2.11), we exploited (2.12)–(2.14). Consequently, to complete the induction step, each of these estimates should be proved for $j = L + 1$.

Focusing on (2.12) for $j = L + 1$, we have

$$\begin{aligned}
\|g(x_{k+L+1}) - \hat{g}(\hat{x}_{k,L+1})\| &\leqslant \|g(x_{k+L+1}) - \hat{g}(x_{k+L+1})\| + \|\hat{g}(x_{k+L+1}) - \hat{g}(\hat{x}_{k,L+1})\| \\
&= \|g(x_{k+L+1}) - \hat{g}(x_{k+L+1})\| + \|H(x_{k+L+1} - \hat{x}_{k,L+1})\| \\
&\leqslant \|g(x_{k+L+1}) - H(x_{k+L+1} - x^*)\| + \Lambda_2\|x_{k+L+1} - \hat{x}_{k,L+1}\| \\
&\leqslant \|g(x_{k+L+1}) - H(x_{k+L+1} - x^*)\| + c\|x_k - x^*\|^2,
\end{aligned}$$

since $\|H\| \leqslant \Lambda_2$ by (2.7). The last inequality is due to (2.11) for $j = L + 1$, which was just established. Since we chose δ small enough that $x_{k+L+1} \in B_\rho(x^*)$ (see (2.15)), (2.6) implies that

$$\|g(x_{k+L+1}) - H(x_{k+L+1} - x^*)\| \leqslant \lambda\|x_{k+L+1} - x^*\|^2 \leqslant c\|x_k - x^*\|^2.$$

Hence, $\|g(x_{k+L+1}) - \hat{g}(\hat{x}_{k,L+1})\| \leqslant c\|x_k - x^*\|^2$, which establishes (2.12) for $j = L + 1$.

Observe that α_{k+L+1} equals either $\alpha_k \in [\Lambda_2^{-1}, \Lambda_1^{-1}]$ or $(s_i^{\mathrm{T}} s_i)/(s_i^{\mathrm{T}} y_i)$, where $k + L \geqslant i = \nu(k + L + 1) > k$. In this latter case, since $x_{k+j} \in B_\rho(x^*)$ for $0 \leqslant j \leqslant L + 1$, it follows from (2.8) that

$$\alpha_{k+L+1} \leqslant \frac{1}{\Lambda_1}.$$

Combining this with (2.9), (2.15) and the bound (2.13) for $j \leqslant L$, we obtain

$$\|s_{k+L+1}\| = \|\alpha_{k+L+1} g(x_{k+L+1})\| \leqslant \frac{\Lambda_2}{\Lambda_1} \|x_{k+L+1} - x^*\| \leqslant c\|x_k - x^*\|.$$

Hence, (2.13) is established for $j = L + 1$.

Finally, we focus on (2.14) for $j = L + 1$. If $\nu(k + L + 1) = \nu(k)$, then $\hat{\alpha}_{k,L+1} = \alpha_{k+L+1} = \alpha_k$, so we are done. Otherwise, $\nu(k + L + 1) > \nu(k)$, and there exists an index $i \in (0, L]$ such that

$$\alpha_{k+L+1} = \frac{s_{k+i}^{\mathrm{T}} s_{k+i}}{s_{k+i}^{\mathrm{T}} y_{k+i}} \quad \text{and} \quad \hat{\alpha}_{k,L+1} = \frac{\hat{s}_{k+i}^{\mathrm{T}} \hat{s}_{k+i}}{\hat{s}_{k+i}^{\mathrm{T}} \hat{y}_{k+i}}.$$

By (2.11) and the fact that $i \leqslant L$, we have

$$\|s_{k+i} - \hat{s}_{k+i}\| \leqslant c\|x_k - x^*\|^2.$$

Combining this with (2.13), and choosing δ smaller, if necessary, gives

$$|s_{k+i}^{\mathrm{T}} s_{k+i} - \hat{s}_{k+i}^{\mathrm{T}} \hat{s}_{k+i}| = |2s_{k+i}^{\mathrm{T}}(s_{k+i} - \hat{s}_{k+i}) - \|\hat{s}_{k+i} - s_{k+i}\|^2| \leqslant c\|x_k - x^*\|^3. \tag{2.20}$$

Since $\hat{\alpha}_{k,i} \in [\Lambda_2^{-1}, \Lambda_1^{-1}]$, we have

$$\|\hat{s}_{k+i}\| = \|\hat{\alpha}_{k,i} \hat{g}_{k,i}\| \geqslant \frac{1}{\Lambda_2} \|H(\hat{x}_{k,i} - x^*)\| \geqslant \frac{\Lambda_1}{\Lambda_2} \|\hat{x}_{k,i} - x^*\|.$$

Furthermore, by (2.10), it follows that

$$\|\hat{s}_{k+i}\| \geqslant \frac{\Lambda_1}{2\Lambda_2} \|\hat{x}_{k,0} - x^*\| = \frac{\Lambda_1}{2\Lambda_2} \|x_k - x^*\|. \tag{2.21}$$

Hence, combining (2.20) and (2.21) gives

$$\left| 1 - \frac{s_{k+i}^{\mathrm{T}} s_{k+i}}{\hat{s}_{k+i}^{\mathrm{T}} \hat{s}_{k+i}} \right| = \frac{|s_{k+i}^{\mathrm{T}} s_{k+i} - \hat{s}_{k+i}^{\mathrm{T}} \hat{s}_{k+i}|}{\hat{s}_{k+i}^{\mathrm{T}} \hat{s}_{k+i}} \leqslant c\|x_k - x^*\|. \tag{2.22}$$

Now let us consider the denominators of α_{k+i} and $\hat{\alpha}_{k,i}$. Observe that

$$\begin{aligned} s_{k+i}^{\mathrm{T}} y_{k+i} - \hat{s}_{k+i}^{\mathrm{T}} \hat{y}_{k+i} &= s_{k+i}^{\mathrm{T}}(y_{k+i} - \hat{y}_{k+i}) + (s_{k+i} - \hat{s}_{k+i})^{\mathrm{T}} \hat{y}_{k+i} \\ &= s_{k+i}^{\mathrm{T}}(y_{k+i} - \hat{y}_{k+i}) + (s_{k+i} - \hat{s}_{k+i})^{\mathrm{T}} H \hat{s}_{k+i}. \end{aligned} \tag{2.23}$$

By (2.11) and (2.13), we have

$$\begin{aligned} |(s_{k+i} - \hat{s}_{k+i})^{\mathrm{T}} H \hat{s}_{k+i}| &= |(s_{k+i} - \hat{s}_{k+i})^{\mathrm{T}} H s_{k+i} - (s_{k+i} - \hat{s}_{k+i})^{\mathrm{T}} H (s_{k+i} - \hat{s}_{k+i})| \\ &\leqslant c\|x_k - x^*\|^3 \end{aligned} \tag{2.24}$$

for δ sufficiently small. Also, by (2.12) and (2.13), we have

$$|s_{k+i}^{\mathrm{T}}(y_{k+i} - \hat{y}_{k+i})| \leqslant \|s_{k+i}\|(\|g_{k+i+1} - \hat{g}_{k,i+1}\| + \|g_{k+i} - \hat{g}_{k,i}\|) \leqslant c\|x_k - x^*\|^3. \tag{2.25}$$

Combining (2.23)–(2.25) yields

$$|s_{k+i}^{\mathrm{T}} y_{k+i} - \hat{s}_{k+i}^{\mathrm{T}} \hat{y}_{k+i}| \leqslant c\|x_k - x^*\|^3. \tag{2.26}$$

Since x_{k+i} and $x_{k+i+1} \in B_\rho(x^*)$, it follows from (2.7) that

$$s_{k+i}^{\mathrm{T}} y_{k+i} = s_{k+i}^{\mathrm{T}}(g_{k+i+1} - g_{k+i}) \geqslant \Lambda_1\|s_{k+i}\|^2 = \Lambda_1|\alpha_{k+i}|^2\|g_{k+i}\|^2. \tag{2.27}$$

By (2.8) and (2.7), we have

$$|\alpha_{k+i}|^2\|g_{k+i}\|^2 \geqslant \frac{1}{\Lambda_2^2}\|g_{k+i}\|^2 = \frac{1}{\Lambda_2^2}\|g(x_{k+i}) - g(x^*)\|^2 \geqslant \frac{\Lambda_1^2}{\Lambda_2^2}\|x_{k+i} - x^*\|^2. \tag{2.28}$$

Finally, (2.10) gives

$$\|x_{k+i} - x^*\|^2 \geqslant \frac{1}{4}\|x_k - x^*\|^2. \tag{2.29}$$

Combining (2.27)–(2.29) yields

$$s_{k+i}^{\mathrm{T}} y_{k+i} \geqslant \frac{\Lambda_1^3}{4\Lambda_2^2}\|x_k - x^*\|^2. \tag{2.30}$$

Combining (2.26) and (2.30) gives

$$\left|1 - \frac{\hat{s}_{k+i}^{\mathrm{T}}\hat{y}_{k+i}}{s_{k+i}^{\mathrm{T}} y_{k+i}}\right| = \frac{|s_{k+i}^{\mathrm{T}} y_{k+i} - \hat{s}_{k+i}^{\mathrm{T}}\hat{y}_{k+i}|}{s_{k+i}^{\mathrm{T}} y_{k+i}} \leqslant c\|x_k - x^*\|. \tag{2.31}$$

Observe that

$$\begin{aligned}
|\alpha_{k+L+1} - \hat{\alpha}_{k,L+1}| &= \left|\frac{s_{k+i}^{\mathrm{T}} s_{k+i}}{s_{k+i}^{\mathrm{T}} y_{k+i}} - \frac{\hat{s}_{k+i}^{\mathrm{T}}\hat{s}_{k+i}}{\hat{s}_{k+i}^{\mathrm{T}}\hat{y}_{k+i}}\right| = \hat{\alpha}_{k,L+1}\left|1 - \left(\frac{s_{k+i}^{\mathrm{T}} s_{k+i}}{\hat{s}_{k+i}^{\mathrm{T}}\hat{s}_{k+i}}\right)\left(\frac{\hat{s}_{k+i}^{\mathrm{T}}\hat{y}_{k+i}}{s_{k+i}^{\mathrm{T}} y_{k+i}}\right)\right| \\
&\leqslant \frac{1}{\Lambda_1}\left|1 - \left(\frac{s_{k+i}^{\mathrm{T}} s_{k+i}}{\hat{s}_{k+i}^{\mathrm{T}}\hat{s}_{k+i}}\right)\left(\frac{\hat{s}_{k+i}^{\mathrm{T}}\hat{y}_{k+i}}{s_{k+i}^{\mathrm{T}} y_{k+i}}\right)\right| \\
&= \frac{1}{\Lambda_1}|a(1-b) + b| \leqslant \frac{1}{\Lambda_1}(|a| + |b| + |ab|),
\end{aligned} \tag{2.32}$$

where

$$a = 1 - \frac{s_{k+i}^{\mathrm{T}} s_{k+i}}{\hat{s}_{k+i}^{\mathrm{T}}\hat{s}_{k+i}} \quad \text{and} \quad b = 1 - \frac{\hat{s}_{k+i}^{\mathrm{T}}\hat{y}_{k+i}}{s_{k+i}^{\mathrm{T}} y_{k+i}}.$$

Together, (2.22), (2.31) and (2.32) yield

$$|\alpha_{k+L+1} - \hat{\alpha}_{k,L+1}| \leqslant c\|x_k - x^*\|$$

for δ sufficiently small. This completes the proof of (2.11)–(2.14). □

THEOREM 2.3 Let x^* be a local minimizer of f, and assume that the Hessian $\nabla^2 f(x^*)$ is positive definite. Then there exist positive constants δ and γ and a positive constant $c < 1$ with the property that for all starting points $x_0, x_1 \in B_\delta(x^*)$, $x_0 \neq x_1$, the CBB iterates generated by (2.1) and (2.2) satisfy

$$\|x_k - x^*\| \leqslant \gamma c^k \|x_1 - x^*\|.$$

Proof. Let $N > 0$ be the integer given in Lemma 2.1, corresponding to $C_1 = \Lambda_1^{-1}$ and $C_2 = \Lambda_2^{-1}$, and let δ_1 and γ_1 denote the constants δ and γ given in Lemma 2.2 Let γ_2 denote the constant c in (2.13). In other words, these constant δ_1, γ_1 and γ_2 have the property that whenever $\|x_k - x^*\| \leqslant \delta_1$, $\alpha_k \in [\Lambda_2^{-1}, \Lambda_1^{-1}]$ and

$$\|\hat{x}_{k,j} - x^*\| \geqslant \frac{1}{2}\|\hat{x}_{k,0} - x^*\| \quad \text{for } 0 \leqslant j \leqslant \ell - 1 < N,$$

we have

$$\|x_{k+j} - \hat{x}_{k,j}\| \leqslant \gamma_1 \|x_k - x^*\|^2, \tag{2.33}$$

$$\|s_{k+j}\| \leqslant \gamma_2 \|x_k - x^*\|, \tag{2.34}$$

$$x_{k+j} \in B_\rho(x^*), \tag{2.35}$$

for all $j \in [0, \ell]$. Moreover, by the triangle inequality and (2.34), it follows that

$$\|x_{k+j} - x^*\| \leqslant (N\gamma_2 + 1)\|x_k - x^*\| = \gamma_3 \|x_k - x^*\|, \quad \gamma_3 = (N\gamma_2 + 1), \tag{2.36}$$

for all $j \in [0, \ell]$. We define

$$\delta = \min\{\delta_1, \rho, (4\gamma_1)^{-1}\}. \tag{2.37}$$

For any x_0 and $x_1 \in B_\delta(x^*)$, we define a sequence $1 = k_1 < k_2 < \cdots$ in the following way: Starting with the index $k_1 = 1$, let $j_1 > 0$ be the smallest integer with the property that

$$\left\|\hat{x}_{k_1,j_1} - x^*\right\| \leqslant \frac{1}{2}\left\|\hat{x}_{k_1,0} - x^*\right\| = \frac{1}{2}\|x_1 - x^*\|.$$

Since x_0 and $x_1 \in B_\delta(x^*) \subset B_\rho(x^*)$, it follows from (2.8) that

$$\hat{\alpha}_{1,0} = \alpha_1 = \frac{s_0^{\mathrm{T}} s_0}{s_0^{\mathrm{T}} y_0} \in [\Lambda_2^{-1}, \Lambda_1^{-1}].$$

By Lemma 2.1, $j_1 \leqslant N$. Define $k_2 = k_1 + j_1 > k_1$. By (2.33) and (2.37), we have

$$\begin{aligned}
\left\|x_{k_2} - x^*\right\| &= \left\|x_{k_1+j_1} - x^*\right\| \leqslant \left\|x_{k_1+j_1} - \hat{x}_{k_1,j_1}\right\| + \left\|\hat{x}_{k_1,j_1} - x^*\right\| \\
&\leqslant \gamma_1 \left\|x_{k_1} - x^*\right\|^2 + \frac{1}{2}\left\|\hat{x}_{k_1,0} - x^*\right\| \\
&= \gamma_1 \left\|x_{k_1} - x^*\right\|^2 + \frac{1}{2}\left\|x_{k_1} - x^*\right\| \leqslant \frac{3}{4}\left\|x_{k_1} - x^*\right\|.
\end{aligned} \tag{2.38}$$

Since $\|x_1 - x^*\| \leqslant \delta$, it follows that $x_{k_2} \in B_\delta(x^*)$. By (2.35), $x_j \in B_\rho(x^*)$ for $1 \leqslant j \leqslant k_1$.

Now, proceed by induction. Assume that k_i has been determined with $x_{k_i} \in B_\delta(x^*)$ and $x_j \in B_\rho(x^*)$ for $1 \leqslant j \leqslant k_i$. Let $j_i > 0$ be the smallest integer with the property that

$$\left\| \hat{x}_{k_i, j_i} - x^* \right\| \leqslant \frac{1}{2} \left\| \hat{x}_{k_i, 0} - x^* \right\| = \frac{1}{2} \left\| x_{k_i} - x^* \right\| .$$

Set $k_{i+1} = k_i + j_i > k_i$. Exactly as in (2.38), we have

$$\left\| x_{k_{i+1}} - x^* \right\| \leqslant \frac{3}{4} \left\| x_{k_i} - x^* \right\| .$$

Again, $x_{k_{i+1}} \in B_\delta(x^*)$ and $x_j \in B_\rho(x^*)$ for $j \in [1, k_{i+1}]$.

For any $k \in [k_i, k_{i+1})$, we have $k \leqslant k_i + N - 1 \leqslant Ni$, since $k_i \leqslant N(i-1) + 1$. Hence, $i \geqslant k/N$. Also, (2.36) gives

$$\begin{aligned} \|x_k - x^*\| \leqslant \gamma_3 \left\| x_{k_i} - x^* \right\| &\leqslant \gamma_3 \left(\frac{3}{4}\right)^{i-1} \left\| x_{k_1} - x^* \right\| \\ &\leqslant \gamma_3 \left(\frac{3}{4}\right)^{(k/N)-1} \|x_1 - x^*\| \\ &= \gamma c^k \|x_1 - x^*\|, \end{aligned}$$

where

$$\gamma = \left(\frac{4}{3}\right) \gamma_3 \quad \text{and} \quad c = \left(\frac{3}{4}\right)^{1/N} < 1.$$

This completes the proof. □

3. The CBB method for convex quadratic programming

In this section, we give numerical evidence which indicates that when m is sufficiently large, the CBB method is superlinearly convergent for a quadratic function

$$f(x) = \frac{1}{2} x^{\mathrm{T}} A x - b^{\mathrm{T}} x, \tag{3.1}$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric and positive definite and $b \in \mathbb{R}^n$. Since CBB is invariant under an orthogonal transformation and since gradient components corresponding to identical eigenvalues can be combined (see, e.g. Dai & Fletcher, 2005b), we assume without loss of generality that A is diagonal:

$$A = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \quad \text{with } 0 < \lambda_1 < \lambda_2 < \cdots < \lambda_n. \tag{3.2}$$

In the following subsections, we give an overview of the experimental convergence results; we then show in the special case $m = 2$ and $n = 3$ that the convergence rate is no better than linear, in general. Finally, we show that the convergence rate for CBB is strictly faster than that of SD. We obtain some further insights by applying our techniques to cyclic SD.

3.1 *Asymptotic behaviour and cycle number*

In the quadratic case, it follows from (1.2) and (3.1) that

$$g_{k+1} = (I - \alpha_k A) g_k. \tag{3.3}$$

TABLE 1 *Transition to superlinear convergence*

n	2	3	4	5	6	8	10	12	14
Superlinear m	1	3	2	4	4	5	6	7	8
Linear m		2	1	3	3	4	5	6	7

If $g_k^{(i)}$ denotes the ith component of the gradient g_k, then by (3.3) and (3.2), we have

$$g_{k+1}^{(i)} = (1 - \alpha_k \lambda_i) g_k^{(i)}, \quad i = 1, 2, \ldots, n. \tag{3.4}$$

We assume that $g_k^{(i)} \neq 0$ for all sufficiently large k. If $g_k^{(i)} = 0$, then by (3.4), component i remains zero during all subsequent iterations; hence, it can be discarded. In the BB method, starting values are needed for x_0 and x_1 in order to compute α_1. In our study of CBB, we treat α_1 as a free parameter. In our numerical experiments, α_1 is the exact stepsize (1.3).

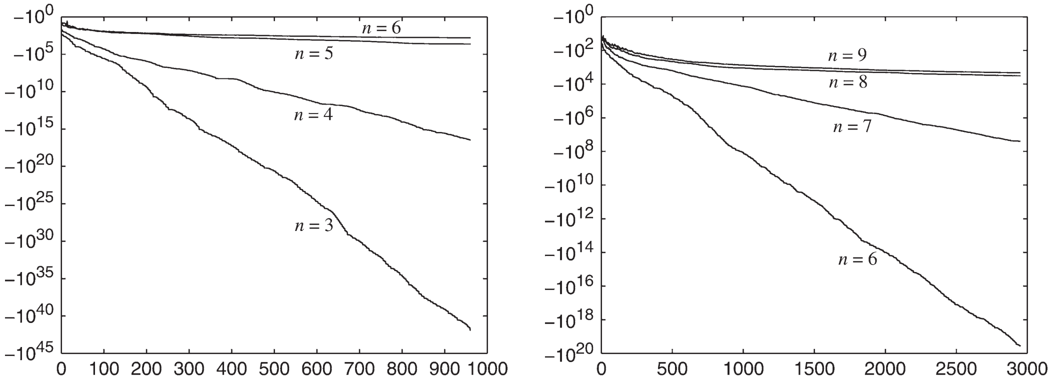
For different choices of the diagonal matrix (3.2) and the starting point x_1, we have evaluated the convergence rate of CBB. By the analysis given in Friedlander *et al.* (1999) for positive definite quadratics or by the result given in Theorem 2.3 for general non-linear functions, the convergence rate of the iterates is at least linear. On the other hand, for m sufficiently large, we observe experimentally that the convergence rate is superlinear. For fixed dimension n, the value of m where the convergence rate makes a transition between linear and non-linear is shown in Table 1. More precisely, for each value of n, the convergence rate is superlinear when m is greater than or equal to the integer given in the second row of Table 1. The convergence is linear when m is less than or equal to the integer given in the third row of Table 1.

The limiting integers appearing in Table 1 are computed in the following way: For each dimension, we randomly generate 30 problems, with eigenvalues uniformly distributed on $(0, n]$, and 50 starting points—a total of 1500 problems. For each test problem, we perform $1000n$ CBB iterations, and we plot $\log(\log(\|g_k\|_\infty))$ versus the iteration number. We fit the data with a least squares line, and we compute the correlation coefficient to determine how well the linear regression model fits the data. If the correlation coefficient is 1 (or -1), then the linear fit is perfect, while a correlation coefficient of 0 means that the data are uncorrelated. A linear fit in a plot of $\log(\log(\|g_k\|_\infty))$ versus the iteration number indicates superlinear convergence. For m large enough, the correlation coefficients are between -1.0 and -0.98, indicating superlinear convergence. As we decrease m, the correlation coefficient abruptly jumps to the order of -0.8. The integers shown in Table 1 reflect the values of m where the correlation coefficient jumps.

Based on Table 1, the convergence rate is conjectured to be superlinear for $m > n/2 \geqslant 3$. For $n < 6$, the relationship between m and n at the transition between linear and superlinear convergence is more complicated, as seen in Table 1. Graphs illustrating the convergence appear in Fig. 1. The horizontal axis in these figures is the iteration number, while the vertical axis gives $\log(\log(\|g_k\|_\infty))$. Here $\|\cdot\|_\infty$ represents the sup–norm. In this case, straight lines correspond to superlinear convergence—the slope of the line reflects the convergence order. In Fig. 1, the bottom two graphs correspond to superlinear convergence, while the top two graphs correspond to linear convergence—for these top two examples, a plot of $\log(\|g_k\|_\infty)$ versus the iteration number is linear.

3.2 *Analysis for the case $m = 2$ and $n = 3$*

The theoretical verification of the experimental results given in Table 1 is not easy. We have the following partial result in connection with the column $m = 2$.



(a) (b)

FIG. 1. Graphs of $\log(\log(\|g_k\|_\infty))$ versus k; (a) $3 \leqslant n \leqslant 6$ and $m = 3$; (b) $6 \leqslant n \leqslant 9$ and $m = 4$.

THEOREM 3.1 For $n = 3$, there exists a choice for the diagonal matrix (3.2) and a starting guess x_1 with the property that $\alpha_{k+8} = \alpha_k$ for each k, and the convergence rate of CBB with $m = 2$ is at most linear.

Proof. To begin, we treat the initial stepsize α_1 as a variable. For each k, we define the vector u_k by

$$u_k^{(i)} = \frac{(g_k^{(i)})^2}{\|g_k\|^2}, \quad i = 1, \ldots, n. \tag{3.5}$$

The above definition is important and is used for some other gradient methods (see Forsythe, 1968; Dai & Yang, 2001). For the case $m = 2$, we can obtain by (2.1), (2.2) and (3.4) and the definition of u_k that

$$u_{2k+1}^{(i)} = \frac{(1 - \alpha_{2k-1}\lambda_i)^4 u_{2k-1}^{(i)}}{\sum_{\ell=1}^n (1 - \alpha_{2k-1}\lambda_\ell)^4 u_{2k-1}^{(\ell)}} \tag{3.6}$$

for all $k \geqslant 1$ and $i = 1, \ldots, n$. In the same fashion, we have

$$\alpha_{2k+1} = \frac{\sum_{i=1}^n (1 - \alpha_{2k-1}\lambda_i)^2 u_{2k-1}^{(i)}}{\sum_{i=1}^n \lambda_i (1 - \alpha_{2k-1}\lambda_i)^2 u_{2k-1}^{(i)}}. \tag{3.7}$$

We want to force our examples to satisfy

$$u_9 = u_1 \quad \text{and} \quad \alpha_9 = \alpha_1. \tag{3.8}$$

For $k \geqslant 1$, a subsequent iteration of the method is uniquely determined by u_{2k-1} and α_{2k-1}. It follows from (3.8) that $u_{8k+1} = u_1$ and $\alpha_{8k+1} = \alpha_1$ for all $k \geqslant 1$, and hence a cycle occurs.

For any i and j, let b_{ij} be defined by

$$b_{ij} = 1 - \alpha_{2i-1}\lambda_j. \tag{3.9}$$

Henceforth, we focus on the case $n = 3$ specified in the statement of the Theorem 3.1. To satisfy relation (3.8), we impose the following condition on the stepsizes $\{\alpha_1, \alpha_3, \alpha_5, \alpha_7\}$:

$$\left|\prod_{i=1}^4 b_{ij}\right| = \tau, \quad j = 1, 2, 3, \tag{3.10}$$

where $\tau > 0$ is a positive number. By (3.6) and (3.10), we know that the first equation of (3.8) is satisfied. On the other hand, (3.6), (3.7), $\alpha_9 = \alpha_1$ and the definition of (3.9) imply the following system of linear equations for u_1:

$$Tu_1 = \begin{bmatrix} b_{11}^2 b_{21} & b_{12}^2 b_{22} & b_{13}^2 b_{23} \\ b_{11}^4 b_{21}^2 b_{31} & b_{12}^4 b_{22}^2 b_{32} & b_{13}^4 b_{23}^2 b_{33} \\ b_{11}^4 b_{21}^4 b_{31}^2 b_{41} & b_{12}^4 b_{22}^4 b_{32}^2 b_{42} & b_{13}^4 b_{23}^4 b_{33}^2 b_{43} \\ b_{11}^5 b_{21}^4 b_{31}^4 b_{41}^2 & b_{12}^5 b_{22}^4 b_{32}^4 b_{42}^2 & b_{13}^5 b_{23}^4 b_{33}^4 b_{43}^2 \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_1^{(2)} \\ u_1^{(3)} \end{bmatrix} = 0. \tag{3.11}$$

The above system has three variables and four equations. Multiplying the jth column by $b_{1j}^{-2} b_{2j}^{-1} b_{4j}$ for $j = 1, 2, 3$ and using condition (3.10), it follows that the rank of the coefficient matrix T is the same as the rank of the 4×3 matrix B with entries b_{ij}. By the definition of b_{ij}, the rank of T is at most 2; hence, the linear system (3.11) has a non-zero solution u_1.

To complete the construction, u_1 should satisfy the constraints

$$u_1^{(i)} > 0, \quad i = 1, 2, 3, \tag{3.12}$$

and

$$u_1^{(1)} + u_1^{(2)} + u_1^{(3)} = 1. \tag{3.13}$$

The above conditions are fulfilled if we look for a solution $\{\alpha_1, \alpha_3, \alpha_5, \alpha_7\}$ of (3.10) such that

$$\alpha_1^{-1}, \alpha_3^{-1} \in (\lambda_1, \lambda_2) \quad \text{and} \quad \alpha_5^{-1}, \alpha_7^{-1} \in (\lambda_2, \lambda_3). \tag{3.14}$$

In this case, we may choose

$$u_1 = t\left[b_{11}^{-2} b_{21}^{-1}\left(\frac{b_{13}}{b_{43}} - \frac{b_{12}}{b_{42}}\right), b_{12}^{-2} b_{22}^{-1}\left(\frac{b_{11}}{b_{41}} - \frac{b_{13}}{b_{43}}\right), b_{13}^{-2} b_{23}^{-1}\left(\frac{b_{12}}{b_{42}} - \frac{b_{11}}{b_{41}}\right)\right]^{\mathrm{T}}, \tag{3.15}$$

where $t > 0$ is a scaling factor such that (3.13) holds. Therefore, if we choose $\{\alpha_1, \alpha_3, \alpha_5, \alpha_7\}$ satisfying (3.10) and (3.14) and furthermore u_1 from (3.15), relation (3.8) holds. Hence, we have that $u_{8+i} = u_i$ and $\alpha_{8+i} = \alpha_i$ for all $i \geqslant 1$.

Now we discuss a possible choice of $\tau > 0$ in (3.10). Specifically, we are interested in the maximal value τ^* of τ such that (3.10) and (3.14) hold. By continuity assumption, we know that suitable solutions exist for any $\tau \in (0, \tau^*)$. This leads to the maximization problem

$$\max\left\{\tau\colon \prod_{i=1}^{4} b_{ij} = \tau\,(j = 1, 2, 3);\, \alpha_1^{-1}, \alpha_3^{-1} \in (\lambda_1, \lambda_2), \alpha_5^{-1}, \alpha_7^{-1} \in (\lambda_2, \lambda_3)\right\}. \tag{3.16}$$

To solve (3.16), we consider the Lagrangian function

$$L(\tau, \alpha_1, \alpha_3, \alpha_5, \alpha_7, \mu_1, \mu_2, \mu_3) = \tau + \sum_{j=1}^{3} \mu_j \left[\tau - \prod_{i=1}^{4}(1 - \alpha_{2i-1}\lambda_j)\right], \tag{3.17}$$

where $\{\mu_j\}$ are the multipliers corresponding to equality constraints. Since at a KKT point of (3.16) the partial derivatives of L are zero, we require $\{\mu_i\}$ to satisfy relation (3.10), $\mu_1+\mu_2+\mu_3=1$ and

$$\sum_{j=1}^{3}\mu_j\lambda_j\prod_{\substack{\ell=1\\ \ell\neq i}}^{4}(1-\alpha_{2\ell-1}\lambda_j)=0 \quad (i=1,2,3,4). \tag{3.18}$$

Dividing each relation in (3.18) by τ and using (3.10), we obtain the following linear equations for $\mu=(\mu_1,\mu_2,\mu_3)^{\mathrm{T}}$:

$$H\mu=0, \quad \text{where } H\in\mathbb{R}^{4\times 3} \text{ with } h_{ij}=\lambda_j b_{ij}^{-1}. \tag{3.19}$$

To guarantee that system (3.19) has a non-zero solution μ, the rank of the coefficient matrix H must be at most 2. Let $H_{3,3}$ denote the submatrix formed by the first three rows of H. By direct calculation, we obtain

$$\det(H_{3,3})=\frac{\lambda_1\lambda_2\lambda_3(\lambda_1-\lambda_2)(\lambda_2-\lambda_3)(\lambda_3-\lambda_1)(\alpha_1-\alpha_3)(\alpha_3-\alpha_5)(\alpha_5-\alpha_1)}{\prod_{i,j\in\{1,2,3\}}b_{ij}}. \tag{3.20}$$

Thus, $\det(H_{3,3})=0$ and inequality constraints (3.14) lead to $\alpha_1=\alpha_3$. Similarly, we can get $\alpha_5=\alpha_7$. From (3.10), we know that (3.16) achieves its maximum

$$\tau^*=\frac{(\lambda_1-\lambda_2)^2(\lambda_2-\lambda_3)^2}{(\lambda_1\lambda_2+\lambda_2\lambda_3+\lambda_3\lambda_1-\lambda_2^2)^2} \tag{3.21}$$

at

$$\alpha_1^*=\alpha_3^*=(\bar\lambda-\bar\xi)^{-1}, \quad \alpha_5^*=\alpha_7^*=(\bar\lambda+\bar\xi)^{-1}, \tag{3.22}$$

where $\bar\lambda=\frac{\lambda_1+\lambda_3}{2}$, $\bar\xi=\sqrt{\frac{\xi_1+\xi_2}{2}}$, $\xi_1=(\bar\lambda-\lambda_1)^2$ and $\xi_2=(\bar\lambda-\lambda_2)^2$. From the continuity argument, we know that there exist cyclic examples of the CBB method with $m=2$ for any $\tau\in(0,\tau^*)$. For example, we may consider the following symmetric subfamily of examples with $\eta\in(0,\frac{1}{2}]$:

$$\alpha_1,\alpha_5=[\bar\lambda\mp\sqrt{\eta\xi_1+(1-\eta)\xi_2}]^{-1}, \quad \alpha_3,\alpha_7=[\bar\lambda\mp\sqrt{(1-\eta)\xi_1+\eta\xi_2}]^{-1}. \tag{3.23}$$

It is easy to check that the above $\{\alpha_i\}$ satisfies (3.10) and (3.14). When η moves from 0 to $\frac{1}{2}$, we can see that the value τ moves from 0 to τ^*. □

Now we present some numerical examples. Suppose that $\lambda_1=1$, $\lambda_2=5$ and $\lambda_3=8$. Because of (3.22), we choose $\alpha_1^*=\alpha_3^*=\frac{1}{2}$ and $\alpha_5^*=\alpha_7^*=\frac{1}{7}$ from where the maximizer $\tau^*=\frac{9}{49}$ is found. From (3.11), we get $u_1=(\frac{972}{1001},\frac{28}{1001},\frac{1}{1001})^{\mathrm{T}}$. By the definition of u_1, the previous discussions and choosing $g_1=\bar t(\pm 18\sqrt{3},\pm 2\sqrt{7},\pm 1)^{\mathrm{T}}$ with any $\bar t>0$ and $\alpha_1=\frac{1}{2}$, the CBB method with $m=2$ produces cycling of the sequence given by $\{u_i\}$ and $\{\alpha_i\}$.

By assuming that the Hessian matrix is $A=\mathrm{diag}(1,5,8)$, we also compute the sequences $\{u_{2k-1}\}$ and $\{\alpha_{2k-1}\}$ generated by (3.6) and (3.7), respectively. Initial values for u_1 and α_1 are obtained by an SD step at u_0, i.e.

$$\alpha_1=\alpha_0=\frac{u_0^{\mathrm{T}}u_0}{u_0^{\mathrm{T}}Au_0}, \qquad u_1^{(i)}=\frac{(1-\alpha_0\lambda_i)^2(u_0^{(i)})^2}{\sum_\ell(1-\alpha_0\lambda_\ell)^2(u_0^{(\ell)})^2} \quad (i=1,2,3).$$

For different u_0, we see that different cycles are obtained, which are numerically stable. In Table 2, the index $\bar{k}$ can be different for each vector u_0 so that $\alpha_{\bar{k}+1}^{-1}, \alpha_{\bar{k}+3}^{-1} \in (\lambda_1, \lambda_2)$.

3.3 *Comparison with SD*

The analysis in Section 3.2 shows that CBB with $m = 2$ is at best linearly convergent. By (3.4) and (3.10), we obtain

$$\|g_{k+8}\|_2 = \tau \|g_k\|_2 \quad \text{for all } k \geqslant 1, \tag{3.24}$$

where τ is the parameter in (3.10). The above relation implies that the convergence rate of the method only depends on the value τ. Furthermore, Table 2 tells us that this value of τ is related to the starting point. It may be very small or relatively large. The maximal possible value of τ is the τ^* in (3.21). In the 3D case, we get

$$\|g_{k+1}\|_2 \leqslant \frac{\lambda_3 - \lambda_1}{\lambda_3 + \lambda_1} \|g_k\|_2 \tag{3.25}$$

for the SD method (see Akaike, 1959). It is not difficult to show that

$$\tau^* < \left[\frac{\lambda_3 - \lambda_1}{\lambda_3 + \lambda_1}\right]^4. \tag{3.26}$$

Thus, we see that CBB with $m = 2$ is faster than the SD method if $n = 3$. This result could be extended to the arbitrary dimensions since we observe that CBB with $m = 2$ generates similar cycles for higher-dimensional quadratics.

The examples provided in Section 3.2 for CBB with $m = 2$ are helpful in understanding and analysing the behaviour of other non-monotone gradient methods. For example, we can also use the same technique to construct cyclic examples for the alternate step (AS) gradient method, at least theoretically. The AS method corresponds to the cyclic SD method (1.4) with $m = 2$. In fact, if we define u_k as in (3.5), we obtain for all $k \geqslant 1$

$$\alpha_{2k-1} = \frac{\sum_\ell u_{2k-1}^{(\ell)}}{\sum_\ell \lambda_\ell u_{2k-1}^{(\ell)}}, \quad u_{2k+1}^{(i)} = \frac{(1 - \alpha_{2k-1}\lambda_i)^4 u_{2k-1}^{(i)}}{\sum_\ell (1 - \alpha_{2k-1}\lambda_\ell)^4 u_{2k-1}^{(\ell)}} \tag{3.27}$$

TABLE 2 *Different choices of u_0 generate different cycles*

u_0^{T}	$\alpha_{\bar{k}+1}^{-1}$	$\alpha_{\bar{k}+3}^{-1}$	$\alpha_{\bar{k}+5}^{-1}$	$\alpha_{\bar{k}+7}^{-1}$	τ
(1, 2, 3)	4.9103	1.0000	8.0000	5.0008	4.2186×10^{-6}
(1, 3, 2)	3.2088	1.3409	6.9100	7.2058	1.2890×10^{-1}
(2, 1, 3)	1.1099	1.2764	5.0197	7.9938	1.5024×10^{-2}
(2, 3, 1)	1.5797	2.0807	5.7248	7.7683	1.3706×10^{-1}
(3, 1, 2)	4.9846	1.0026	7.9086	7.7458	1.6018×10^{-3}
(3, 2, 1)	1.0015	4.9912	7.8776	7.8866	9.4127×10^{-4}

for $i = 1, \dots, n$. For any n with $u_{2n+1} = u_1$ and $\alpha_{2n+1} = \alpha_1$, we require the stepsizes $\{\alpha_{2k-1}\colon k = 1, \dots, n-1\}$ to satisfy

$$\left|\prod_{i=1}^{n-1} b_{ij}\right| = \tau, \quad j = 1, \dots, n, \tag{3.28}$$

where b_{ij} is given by (3.9). At the same time, we obtain the following linear equations for u_1:

$$Tu_1 = 0, \quad \text{where } T \in \mathbb{R}^{(n-1)\times n} \text{ with } T_{ij} = b_{ij} \prod_{\ell=1}^{i-1} b_{\ell j}^4. \tag{3.29}$$

The above system (3.29) has n variables, but $n-1$ equations. If there is a positive solution $\bar{u}_1$, then we may scale the vector and obtain another positive solution $u_1 = c\bar{u}_1$ with $\sum_\ell u_1^{(\ell)} = 1$, which completes the construction of a cyclic example. Here we present a 5D example. We first fix $\alpha_1 = 1$, $\alpha_3 = 0.1$, $\alpha_5 = 0.2$ and $\alpha_7 = 0.0625$, and then choose

$$\lambda = (0.73477, 1.3452, 4.2721, 10.554, 16.154)$$

which are five roots of the equation $\prod_{k=1}^4 (1 - \alpha_{2k-1} w) = 0.2$. Therefore, we get the matrix

$$T = \begin{pmatrix} 0.26523 & -0.34515 & -3.2721 & -9.5537 & -15.154 \\ 0.00458 & 0.01228 & 65.659 & -461.26 & -32451 \\ 0.00311 & 0.00582 & 1.7964 & -0.08696 & -16870 \\ 0.00184 & 0.00208 & 0.00406 & 0.04056 & -1800.5 \end{pmatrix}.$$

The system $Tu_1 = 0$ has the positive solution

$$\bar{u}_1 = (5.6163 \times 10^5, 3.3397 \times 10^5, 7.3848 \times 10^3, 9.9533 \times 10^2, 1.0)^{\mathrm{T}}$$

which leads to

$$u_1 = (6.2128 \times 10^{-1}, 3.6945 \times 10^{-1}, 8.1693 \times 10^{-3}, 1.1011 \times 10^{-3}, 1.1062 \times 10^{-6})^{\mathrm{T}}.$$

Therefore, if we choose the above initial vector u_1, we get $u_{10k+1} = u_1$ and $\alpha_{10k+1} = \alpha_1$ for all $k \geqslant 1$, and hence, the AS method falls into a cycle. Unlike CBB with $m = 2$, we have not found any cyclic example for the AS method which are numerically stable.

4. An adaptive cyclic Barzilai–Borwein method

In this section, we examine the convergence speed of CBB for different values of $m \in [1, 7]$, using quadratic programming problems of the form:

$$f(x) = \frac{1}{2} x^{\mathrm{T}} A x, \quad A = \mathrm{diag}(\lambda_1, \dots, \lambda_n). \tag{4.1}$$

We will see that the choice for m has a significant impact on performance. This leads us to propose an adaptive choice for m. The BB algorithm with this adaptive choice for m and a non-monotone line search is called adaptive cyclic Barzilai–Borwein (ACBB). Numerical comparisons with SPG2 and with conjugate gradient codes using the CUTEr test problem library are given later in Section 4.

4.1 *A numerical investigation of CBB*

We consider the test problem (4.1) with four different condition numbers C for the diagonal matrix, $C = 10^2$, $C = 10^3$, $C = 10^4$ and $C = 10^5$, and with three different dimensions $n = 10^2$, $n = 10^3$ and $n = 10^4$. We let $\lambda_1 = 1$, $\lambda_n = C$, the condition number. The other diagonal elements λ_i, $2 \leqslant i \leqslant n-1$, are randomly generated on the interval $(1, \lambda_n)$. The starting points $x_1^{(i)}$, $i = 1, \dots, n$, are randomly generated on the interval $[-5, 5]$. The stopping condition is

$$\|g_k\|_2 \leqslant 10^{-8}.$$

For each case, 10 runs are made and the average number of iterations required by each algorithm is listed in Table 3 (under the columns labelled BB and CBB). The upper bound for the number of iterations is 9999. If this upper bound is exceeded, then the corresponding entry in Table 3 is F.

In Table 3, we see that $m = 2$ gives the worst numerical results—in Section 3, we saw that as m increases, convergence became superlinear. For each case, a suitably chosen m drastically improves the efficiency of the BB method. For example, in case of $n = 10^2$ and cond $= 10^5$, CBB with $m = 7$ only requires one-fifth of the iterations of the BB method. The optimal choice of m varies from one test case to another. If the problem condition is relatively small (cond $= 10^2$, 10^3), a smaller value of m (3 or 4) is preferred. If the problem condition is relatively large (cond $= 10^4$, 10^5), a larger value of m is more efficient. This observation is the motivation for introducing an adaptive choice for m in the CBB method.

Our adaptive idea arises from the following considerations. If a stepsize is used infinitely often in the gradient method, namely, $\alpha_k \equiv \alpha$, then under the assumption that the function Hessian A has no multiple eigenvalues, the gradient g_k must approximate an eigenvector of A, and $g_k^{\mathrm{T}} A g_k / g_k^{\mathrm{T}} g_k$ tends to the corresponding eigenvalue of A (see Dai, 2003). Thus, it is reasonable to assume that repeated use of a BB stepsize leads to good approximations of eigenvectors of A. First, we define

$$\nu_k = \frac{g_k^{\mathrm{T}} A g_k}{\|g_k\| \|A g_k\|}. \tag{4.2}$$

TABLE 3 *Comparing CBB(m) method with an ACBB method*

			CBB						Adaptive	
n	Cond	BB	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$	$\overline{M} = 5$	$\overline{M} = 10$
10^2	10^2	147	219	156	145	150	160	166	136	134
	10^3	505	2715	468	364	376	395	412	367	349
	10^4	1509	F	1425	814	852	776	628	878	771
	10^5	5412	F	5415	3074	1670	1672	1157	2607	1915
10^3	10^2	147	274	160	158	162	166	181	150	145
	10^3	505	1756	548	504	493	550	540	481	460
	10^4	1609	F	1862	1533	1377	1578	1447	1470	1378
	10^5	5699	F	6760	4755	3506	3516	2957	4412	3187
10^4	10^2	156	227	162	166	167	170	187	156	156
	10^3	539	3200	515	551	539	536	573	497	505
	10^4	1634	F	1823	1701	1782	1747	1893	1587	1517
	10^5	6362	F	6779	5194	4965	4349	4736	4687	4743

If g_k is exactly an eigenvector of A, we know that $\nu_k = 1$. If $\nu_k \approx 1$, then g_k can be regarded as an approximation of an eigenvector of A and $\alpha_k^{\mathrm{BB}} \approx \alpha_k^{\mathrm{SD}}$. In this case, it is worthwhile to calculate a new BB stepsize α_k^{BB} so that the method accepts a step close to the SD step. Therefore, we test the condition

$$\nu_k \geqslant \beta, \tag{4.3}$$

where $\beta \in (0, 1)$ is constant. If the above condition holds, we calculate a new BB stepsize. We also introduce a parameter $\overline{M}$, and if the number of cycles $m > \overline{M}$, we calculate a new BB stepsize. Numerical results for this ACBB with $\beta = 0.95$ are listed under the column Adaptive of Table 3, where two values of $\overline{M} = 5$, 10 are tested.

From Table 3, we see that the adaptive strategy makes sense. The performance with $\overline{M} = 5$ or $\overline{M} = 10$ is often better than that of the BB method. This motivates the use of a similar strategy for designing an efficient gradient algorithms for unconstrained optimization.

4.2 *Non-monotone line search and cycle number*

As mentioned in Section 1, the choice of the stepsize α_k is very important for the performance of a gradient method. For the BB method, function values do not decrease monotonically. Hence, when implementing BB or CBB, it is important to use a non-monotone line search.

Assuming that d_k is a descent direction at the kth iteration ($g_k^{\mathrm{T}} d_k < 0$), a common termination condition for the steplength algorithm is

$$f(x_k + \alpha_k d_k) \leqslant f_r + \delta \alpha_k g_k^{\mathrm{T}} d_k, \tag{4.4}$$

where f_r is the so-called 'reference function value' and $\delta \in (0, 1)$ a constant. If $f_r = f(x_k)$, then the line search is monotone since $f(x_{k+1}) < f(x_k)$. The non-monotone line search proposed in Grippo *et al.* (1986) chooses f_r to be the maximum function value for the M most recent iterates. That is, at the kth iteration, we have

$$f_r = f_{\max} = \max_{0 \leqslant i \leqslant \min\{k, M-1\}} f(x_{k-i}). \tag{4.5}$$

This non-monotone line search is used by Raydan (1997) to obtain GBB. Dai & Schittkowski (2005) extended the same idea to a sequential quadratic programming method for general constrained nonlinear optimization. An even more adaptive way of choosing f_r is proposed by Toint (1997) for trust region algorithms and then extended by Dai & Zhang (2001). Compared with (4.5), the new adaptive way of choosing f_r allows big jumps in function values, and is therefore very suitable for the BB algorithm (see Dai & Fletcher, 2005b, 2006; Dai & Zhang, 2001).

The numerical results which we report in this section are based on the non-monotone line search algorithm given in Dai & Zhang (2001). The line search in our paper differs from the line search in Dai & Zhang (2001) in the initialization of the stepsize. Here, the starting guess for the stepsize coincides with the prior BB step until the cycle length has been reached; at which point, we recompute the step using the BB formula. In each subsequent subiteration, after computing a new BB step, we replace (4.4) with

$$f(x_k + \bar{\alpha}_k d_k) \leqslant \min\{f_{\max}, f_r\} + \delta \bar{\alpha}_k g_k^{\mathrm{T}} d_k,$$

where f_r is the reference value given in Dai & Zhang (2001) and $\bar{\alpha}_k$ is the initial trial stepsize (the previous BB step). It is proved in Dai & Zhang (2001, Theorem 3.2) that the criteria given for choosing the non-monotone stepsize ensures convergence in the sense that

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

We now explain how we decided to terminate the current cycle, and recompute the stepsize using the BB formula. Note that the re-initialization of the stepsize has no effect on convergence, it only affects the initial stepsize used in the line search. Loosely, we would like to compute a new BB step in any of the following cases:

R1. The number of times m the current BB stepsize has been reused is sufficiently large: $m \geqslant \overline{M}$, where $\overline{M}$ is a constant.

R2. The following non-quadratic analogue of (4.3) is satisfied:

$$\frac{s_k^{\mathrm{T}} y_k}{\|s_k\|_2 \|y_k\|_2} \geqslant \beta, \tag{4.6}$$

where $\beta < 1$ is near 1. We feel that Condition (4.6) should only be used in a neighbourhood of a local minimizer, where f is approximately quadratic. Hence, we only use Condition (4.6) when the stepsize is sufficiently small:

$$\|s_k\|_2 < \min\left\{\frac{c_1 f_{k+1}}{\|g_{k+1}\|_\infty}, 1\right\}, \tag{4.7}$$

where c_1 is a constant.

R3. The current step s_k is sufficiently large:

$$\|s_k\|_2 \geqslant \max\left\{c_2 \frac{f_{k+1}}{\|g_{k+1}\|_\infty}, 1\right\}, \tag{4.8}$$

where c_2 is a constant.

R4. In the previous iteration, the BB step was truncated in the line search. That is, the BB step had to be modified by the non-monotone line search routine to ensure convergence.

Nominally, we recompute the BB stepsize in any of the cases R1–R4. One case where we prefer to retain the current stepsize is the case where the iterates lie in a region where f is not strongly convex. Note that if $s_k^{\mathrm{T}} y_k < 0$, then there exists a point between x_k and x_{k+1} where the Hessian of f has negative eigenvalues. In detail, our rules for terminating the current cycle and re-initializing the BB stepsize are the following conditions.

4.2.1 *Cycle termination/stepsize initialization.*

T1. If any of the conditions R1 through R4 are satisfied and $s_k^{\mathrm{T}} y_k > 0$, then the current cycle is terminated and the initial stepsize for the next cycle is given by

$$\alpha_{k+1} = \max\left\{\alpha_{\min}, \min\left\{\frac{s_k^{\mathrm{T}} s_k}{s_k^{\mathrm{T}} y_k}, \alpha_{\max}\right\}\right\},$$

where $\alpha_{\min} < \alpha_{\max}$ are fixed constants.

T2. If the length m of the current cycle satisfies $m \geqslant 1.5\overline{M}$, then the current cycle is terminated and the initial stepsize for the next cycle is given by

$$\alpha_{k+1} = \max\{1/\|g_{k+1}\|_\infty, \alpha_k\}.$$

Condition T2 is a safeguard for the situation where $s_k^{\mathrm{T}} y_k < 0$ in a series of iterations.

4.3 *Numerical results*

In this subsection, we compare the performance of our ACBB stepsize algorithm, denoted ACBB, with the SPG2 algorithm of Birgin *et al.* (2000, 2001), with the PRP+ conjugate gradient code developed by Gilbert & Nocedal (1992) and with the CG_DESCENT code of Hager & Zhang (2005b, to appear). The SPG2 algorithm is an extension of Raydan's (1997) GBB algorithm which was downloaded from the TANGO web page maintained by Ernesto Birgin. In our tests, we set the bounds in SPG2 to infinity. The PRP+ code is available at http://www.ece.northwestern.edu/∼nocedal/software.html. The CG_DESCENT code is found at http://www.math.ufl.edu/∼hager/papers/CG. The line search in the PRP+ code is a modification of subroutine CSRCH of Moré & Thuente (1994), which employs various polynomial interpolation schemes and safeguards in satisfying the strong Wolfe conditions. CG_DESCENT employs an 'approximate Wolfe' line search. All codes are written in Fortran and compiled with f77 under the default compiler settings on a Sun workstation. The parameters used by CG_DESCENT are the default parameter values given in Hager & Zhang (2006) for version 1.1 of the code. For SPG2, we use parameter values recommended on the TANGO web page. In particular, the steplength was restricted to the interval $[10^{-30}, 10^{30}]$, while the memory in the non-monotone line search was 10.

The parameters of the ACBB algorithm are $\alpha_{\min} = 10^{-30}$, $\alpha_{\max} = 10^{30}$, $c_1 = c_2 = 0.1$ and $\overline{M} = 4$. For the initial iteration, the starting stepsize for the line search was $\alpha_1 = 1/\|g_1\|_\infty$. The parameter values for the non-monotone line search routine from Dai & Zhang (2001) were $\delta = 10^{-4}$, $\sigma_1 = 0.1$, $\sigma_2 = 0.9$, $\beta = 0.975$, $L = 3$, $M = 8$ and $P = 40$.

Our numerical experiments are based on the entire set of 160 unconstrained optimization problem available from CUTEr in the fall of 2004. As explained in Hager & Zhang (2006), we deleted problems that were small or problems where different solvers converged to different local minimizers. After the deletion process, we were left with 111 test problems with dimension ranging from 50 to 10^4.

Nominally, our stopping criterion was the following:

$$\|\nabla f(x_k)\|_\infty \leqslant \max\{10^{-6}, 10^{-12}\|\nabla f(x_0)\|_\infty\}. \tag{4.9}$$

In a few cases, this criterion was too lenient. For example, with the test problem PENALTY1, the computed cost still differs from the optimal cost by a factor of 10^5 when Criterion (4.9) is satisfied. As a result, different solvers obtain completely different values for the cost, and the test problem would be discarded. By changing the convergence criterion to $\|\nabla f(x_k)\|_\infty \leqslant 10^{-6}$, the computed costs all agreed to six digits. The problems for which the convergence criterion was strengthened were DQRTIC, PENALTY1, POWER, QUARTC and VARDIM.

The CPU time in seconds and the number of iterations, function evaluations and gradient evaluations for each of the methods are posted at the following web site: http://www.math.ufl.edu/∼hager/papers/CG. Here we analyse the performance data using the profiles of Dolan & Moré (2002). That is, we plot the fraction p of problems for which any given method is within a factor τ of the best time. In a plot of

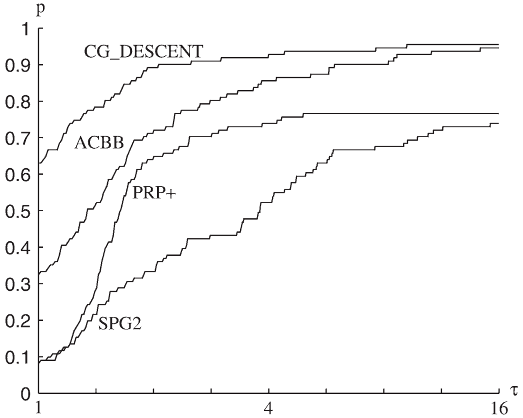


FIG. 2. Performance based on CPU time.

TABLE 4 *Number of times each method was fastest (time metric, stopping criterion* (4.9))

Method	Fastest
CG_DESCENT	70
ACBB	36
PRP+	9
SPG2	9

performance profiles, the top curve is 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 each of the methods. In essence, the right side is a measure of an algorithm's robustness.

In Fig. 2, we use CPU time to compare the performance of the four codes ACBB, SPG2, PRP+ and CG_DESCENT. Note that the horizontal axis in Fig. 2 is scaled proportional to $\log_2(\tau)$. The best performance, relative to the CPU time metric, was obtained by CG_DESCENT, the top curve in Fig. 2, followed by ACBB. The horizontal axis in the figure stops at $\tau = 16$ since the plots are essentially flat for larger values of τ. For this collection of methods, the number of times any method achieved the best time is shown in Table 4. The column total in Table 4 exceeds 111 due to ties for some test problems.

The results of Fig. 2 indicate that ACBB is much more efficient than SPG2, while it performed better than PRP+, but not as well as CG_DESCENT. From the experience in Raydan (1997), the GBB algorithm, with a traditional non-monotone line search (Grippo *et al.*, 1986), may be affected significantly by nearly singular Hessians at the solution. We observe that nearly singular Hessians do not affect ACBB significantly. In fact, Table 3 also indicates that ACBB becomes more efficient as the problem becomes more singular. Furthermore, since ACBB does not need to calculate the BB stepsize at every iteration, CPU time is saved, which can be significant when the problem dimension is large. For this test set, we found that the average cycle length for ACBB was 2.59. In other words, the BB step is re-evaluated after two or three iterations, on average. This memory length is smaller than the memory length that works

TABLE 5 *CPU times for selected problems*

Problem	Dimension	ACBB	CG_DESCENT
FLETCHER	5000	9.14	989.55
FLETCHER	1000	1.32	27.27
BDQRTIC	1000	0.37	3.40
VARDIM	10000	0.05	2.13
VARDIM	5000	0.02	0.92

well for quadratic function. When the iterates are far from a local minimizer of a general non-linear function, the iterates may not behave like the iterates of a quadratic. In this case, better numerical results are obtained when the BB stepsize is updated more frequently.

Even though ACBB did not perform as well as CG_DESCENT for the complete set of test problems, there were some cases where it performed exceptionally well (see Table 5). One important advantage of the ACBB scheme over conjugate gradient routines such as PRP+ or CG_DESCENT is that in many cases, the stepsize for ACBB is either the previous stepsize or the BB stepsize (1.5). In contrast, with conjugate gradient routines, each iteration requires a line search. Due to the simplicity of the ACBB stepsize, it can be more efficient when the iterates are in a regime where the function is irregular and the asymptotic convergence properties of the conjugate gradient method are not in effect. One such application is bound-constrained optimization problems—as components of x reach the bounds, these components are often held fixed, and the associated partial derivative change discontinuously. In Hager & Zhang (2005a) ACBB is combined with CG_DESCENT to obtain a very efficient active set algorithm for box-constrained optimization problems.

5. Conclusion and discussion

In this paper, we analyse the CBB method. For general non-linear functions, we prove linear convergence. For convex quadratic functions, our numerical results indicate that when $m > n/2 \geqslant 3$, CBB is likely to be R-superlinear. For the special case $n = 3$ and $m = 2$, the convergence rate, in general, is no better than linear. By utilizing non-monotone line search techniques, we develop an ACBB stepsize algorithm for general non-linear unconstrained optimization problems.

The test results in Fig. 2 indicate that ACBB is significantly faster than SPG2. Since the mathematical foundations of ACBB and the conjugate gradient algorithms are completely different, the performance seems to depend on the problem. Roughly speaking, if the objective function is ‘close’ to quadratic, the conjugate gradient routines seem to be more efficient; if the objective function is highly non-linear, then ACBB is comparable to or even better than conjugate gradient algorithms.

Acknowledgements

Constructive and detailed comments by the referees are gratefully acknowledged and appreciated. Y-HD was supported by the Alexander von Humboldt Foundation under grant CHN/1112740 STP and Chinese National Science Foundation grants 10171104 and 40233029. WWH and HZ were supported by US National Science Foundation grant no. 0203270.

REFERENCES

AKAIKE, H. (1959) On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Ann. Inst. Stat. Math. Tokyo*, **11**, 1–17.

BARZILAI, J. & BORWEIN, J. M. (1988) Two point step size gradient methods. *IMA J. Numer. Anal.*, **8**, 141–148.

BIRGIN, E. G., CHAMBOULEYRON, I. & MARTÍNEZ, J. M. (1999) Estimation of the optical constants and the thickness of thin films using unconstrained optimization. *J. Comput. Phys.*, **151**, 862–880.

BIRGIN, E. G., MARTÍNEZ, J. M. & RAYDAN, M. (2000) Nonmonotone spectral projected gradient methods for convex sets. *SIAM J. Optim.*, **10**, 1196–1211.

BIRGIN, E. G., MARTÍNEZ, J. M. & RAYDAN, M. (2001) Algorithm 813: SPG—software for convex-constrained optimization. *ACM Trans. Math. Softw.*, **27**, 340–349.

BONGARTZ, I., CONN, A. R., GOULD, N. I. M. & TOINT, P. L. (1995) CUTE: constrained and unconstrained testing environments. *ACM Trans. Math. Softw.*, **21**, 123–160.

CAUCHY, A. (1847) Méthode générale pour la résolution des systèms d'equations simultanées. *Comp. Rend. Sci. Paris*, **25**, 46–89.

DAI, Y. H. (2003) Alternate stepsize gradient method. *Optimization*, **52**, 395–415.

DAI, Y. H. & FLETCHER, R. (2005a) On the asymptotic behaviour of some new gradient methods. *Math. Prog.*, **103**, 541–559.

DAI, Y. H. & FLETCHER, R. (2005b) Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numer. Math.*, **100**, 21–47.

DAI, Y. H. & FLETCHER, R. (2006) New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Math. Prog.*, **106**, 403–421.

DAI, Y. H. & SCHITTKOWSKI, K. (2005) A sequential quadratic programming algorithm with non-monotone line search. *Technical Report*. Department of Mathematics, University of Bayreuth (submitted).

DAI, Y. H. & YANG, X. Q. (2006) A new gradient method with an optimal stepsize property. *Comput. Optim. Appl.*, **33**, 73–88.

DAI, Y. H. & YUAN, Y. (2003) Alternate minimization gradient method. *IMA J. Numer. Anal.*, **23**, 377–393.

DAI, Y. H. & ZHANG, H. (2001) An adaptive two-point stepsize gradient algorithm. *Numer. Algorithms*, **27**, 377–385.

DOLAN, E. D. & MORÉ, J. J. (2002) Benchmarking optimization software with performance profiles. *Math. Program.*, **91**, 201–213.

FORSYTHE, G. E. (1968) On the asymptotic directions of the s-dimensional optimum gradient method. *Numer. Math.*, **11**, 57–76.

FRIEDLANDER, A., MARTÍNEZ, J. M., MOLINA, B. & RAYDAN, M. (1999) Gradient method with retards and generalizations. *SIAM J. Numer. Anal.*, **36**, 275–289.

GILBERT, J. C. & NOCEDAL, J. (1992) Global convergence properties of conjugate gradient methods for optimization. *SIAM J. Optim.*, **2**, 21–42.

GLUNT, W., HAYDEN, T. L. & RAYDAN, M. (1993) Molecular conformations from distance matrices. *J. Comput. Chem.*, **14**, 114–120.

GRIPPO, L., LAMPARIELLO, F. & LUCIDI, S. (1986) A nonmonotone line search technique for Newton's method. *SIAM J. Numer. Anal.*, **23**, 707–716.

GRIPPO, L. & SCIANDRONE, M. (2002) Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. *Comput. Optim. Appl.*, **23**, 143–169.

HAGER, W. W. & ZHANG, H. (2005a) A new active set algorithm for box constrained optimization. *SIAM J. Optim.* (to appear).

HAGER, W. W. & ZHANG, H. (2005b) A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM J. Optim.*, **16**, 170–192.

HAGER, W. W. & ZHANG, H. (2006) Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent. *ACM Trans. Math. Softw.*, **32**.

LIU, W. B. & DAI, Y. H. (2001) Minimization algorithms based on supervisor and searcher cooperation. *J. Optim. Theory Appl.*, **111**, 359–379.

MORÉ, J. J. & THUENTE, D. J. (1994) Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, **20**, 286–307.

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

RAYDAN, M. & SVAITER, B. F. (2002) Relaxed steepest descent and Cauchy-Barzilai-Borwein method. *Comput. Optim. Appl.*, **21**, 155–167.

SERAFINI, T. ZANGHIRATI, G. & ZANNI, L. (2005) Gradient projection methods for quadratic programs and applications in training support vector machines. *Optim. Methods Softw.*, **20**, 353–378.

TOINT, P. L. (1997) A non-monotone trust region algorithm for nonlinear optimization subject to convex constraints. *Math. Prog.*, **77**, 69–94.