Algorithm 851: CG_DESCENT, a Conjugate Gradient Method with Guaranteed Descent

WILLIAM W. HAGER and HONGCHAO ZHANG
University of Florida

Recently, a new nonlinear conjugate gradient scheme was developed which satisfies the descent condition $\mathbf{g}_k^{\mathsf{T}}\mathbf{d}_k \le -\frac{7}{8}\|\mathbf{g}_k\|^2$ and which is globally convergent whenever the line search fulfills the Wolfe conditions. This article studies the convergence behavior of the algorithm; extensive numerical tests and comparisons with other methods for large-scale unconstrained optimization are given.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Fortran 77*; G.1.6 [**Numerical Analysis**]: Optimization—*Gradient methods*; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Conjugate gradient method, convergence, line search, unconstrained optimization, Wolfe conditions

1. INTRODUCTION

In Hager and Zhang [2005] we introduce a new nonlinear conjugate gradient method for solving an unconstrained optimization problem

$$\min \{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}, \tag{1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable. Here, we investigate the numerical performance of the algorithm and show how it relates to previous conjugate gradient research. The iterates $\mathbf{x}_k$, $k \ge 0$, in conjugate gradient methods satisfy the recurrence

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k,$$

This material is based upon work supported by the National Science Foundation under Grant No. CCR-0203270.
Authors' addresses: W. Hager, H. Zhang, Department of Mathematics, University of Florida, PO Box 118105, Gainesville, FL 32611-8105; email: {hager,hzhang}@math.ufl.edu.

where the stepsize α_k is positive, and the directions $\mathbf{d}_k$ are generated by the rule:

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k, \quad \mathbf{d}_0 = -\mathbf{g}_0. \tag{2}$$

Here, $\mathbf{g}_k = \nabla f(\mathbf{x}_k)^\mathsf{T}$ where the gradient $\nabla f(\mathbf{x}_k)$ of f at $\mathbf{x}_k$ is a row vector, and $\mathbf{g}_k$ is a column vector. There are many different versions of the conjugate gradient method corresponding to different choices of β_k. When f is quadratic and α_k is chosen to minimize f in the search direction $\mathbf{d}_k$, these choices are all equivalent, but for a general nonlinear function, different choices have quite different convergence properties.

The history of the conjugate gradient method, surveyed in Golub and O'leary [1989], begins with the research of Cornelius Lanczos, Magnus Hestenes, and others (Forsythe, Motzkin, Rosser, and Stein) at the Institute for Numerical Analysis (National Applied Mathematics Laboratories of the United States National Bureau of Standards in Los Angeles), and with the independent research of Eduard Stiefel at Eidg. Technische Hochschule Zürich. In the seminal work of Hestenes and Stiefel [1952], the algorithm is presented as an approach to solve symmetric, positive definite linear systems. Advantages of the conjugate gradient method are its low memory requirements and its convergence speed.

In 1964, the domain of application of conjugate gradient methods was extended to nonlinear problems, starting with the seminal research of Fletcher and Reeves [1964]. In their work, the stepsize α_k is obtained by a line search in the search direction $\mathbf{d}_k$, and the update parameter β_k is given by

$$\beta_k^{FR} = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2},$$

where $\|\cdot\|$ denotes the Euclidean norm. Daniel [1967] gave the following choice for the update parameter:

$$\beta_k^{D} = \frac{\mathbf{g}_{k+1}^\mathsf{T} \nabla^2 f(\mathbf{x}_k)\mathbf{d}_k}{\mathbf{d}_k^\mathsf{T} \nabla^2 f(\mathbf{x}_k)\mathbf{d}_k},$$

where $\nabla^2 f$ is the Hessian of f. This choice for β_k requires evaluation of both the gradient and the Hessian in each step, which can be impractical in many applications, while the Fletcher-Reeves formula only requires the gradient in each step.

Global convergence of the Fletcher-Reeves scheme was established for an exact line search [Powell 1984] or for an inexact line search [Al-Baali 1985]. However, Powell [1977] observed that in some cases jamming occurred; that is, the search directions $\mathbf{d}_k$ became nearly orthogonal to the gradient $\mathbf{g}_k$. Polak and Ribière [1969] and Polyak [1969] gave a modification of the Fletcher-Reeves update which addressed the jamming phenomenon that would be pointed out in Powell [1977]. Their choice for the update parameter was

$$\beta_k^{PRP} = \frac{\mathbf{y}_k^\mathsf{T} \mathbf{g}_{k+1}}{\|\mathbf{g}_k\|^2},$$

where $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$. When jamming occurs $\mathbf{g}_{k+1} \approx \mathbf{g}_k$, $\beta_k^{PRP} \approx 0$, and $\mathbf{d}_{k+1} \approx -\mathbf{g}_{k+1}$. In other words, when jamming occurs, the search direction is no longer orthogonal to the gradient, but aligned with the negative gradient. This built-in restart feature of the PRP method often gave more rapid convergence when compared to the FR scheme.

The work of Hestenes and Stiefel [1952] presents a choice for β_k closely related to the PRP scheme:

$$\beta_k^{HS} = \frac{\mathbf{y}_k^\mathsf{T}\mathbf{g}_{k+1}}{\mathbf{y}_k^\mathsf{T}\mathbf{d}_k}. \tag{3}$$

If α_k is obtained by an exact line search, then by (2) and the orthogonality condition $\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k = 0$, we have

$$\mathbf{y}_k^\mathsf{T}\mathbf{d}_k = (\mathbf{g}_{k+1} - \mathbf{g}_k)^\mathsf{T}\mathbf{d}_k = -\mathbf{g}_k^\mathsf{T}\mathbf{d}_k = \mathbf{g}_k^\mathsf{T}\mathbf{g}_k.$$

Hence, $\beta_k^{HS} = \beta_k^{PRP}$ when α_k is obtained by an exact line search.

More recent nonlinear conjugate gradient algorithms include the conjugate descent algorithm of Fletcher [1987], the scheme of Liu and Storey [1991], and the scheme of Dai and Yuan [1999] (see the survey article [Hager and Zhang 2006]). The scheme of Dai and Yuan, which is used in the numerical experiments of Section 4, corresponds to the following choice for the update parameter:

$$\beta_k^{DY} = \frac{\|\mathbf{g}_{k+1}\|^2}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}. \tag{4}$$

Powell [1984] showed that the PRP method, with α_k obtained by an exact line search, can cycle infinitely without approaching a stationary point. Thus, the PRP method addressed the jamming of the FR method, but Powell's example shows that in some cases PRP does not converge at all, even when the line search is exact. To cope with possible convergence failure in the PRP scheme, Powell [1986] suggested that β_k be replaced by

$$\beta_k^{PRP+} = \max\,\{\beta_k^{PRP}, 0\}.$$

In other words, when the β_k given by PRP is negative, it is replaced by zero. Gilbert and Nocedal [1992] proved global convergence of this PRP+ scheme. Other ways to restrict β_k in the PRP scheme are developed in Han et al. [2001] and Wang et al. [2000].

Although the PRP+ scheme addresses both the jamming of the FR method and the possibility of convergence failure, it interferes with the n-step convergence property of the conjugate gradient method for strongly convex quadratic functions. That is, when the conjugate gradient method is applied to a quadratic with an exact line search, the successive iterates minimize f over an expanding sequence of subspaces, leading to rapid convergence. In this case, $\beta_k > 0$ for each k, however, due to rounding errors we can have $\beta_k^{PRP} < 0$, which implies that $\beta_k^{PRP+} = 0$. Each time β_k is set to zero, the conjugate gradient method is restarted, and the expanding sequence of subspaces reinitiates with a one-dimensional space, leading to slower convergence than would be achieved if there was no restart.

Another important issue related to the performance of conjugate gradient methods is the line search, which requires sufficient accuracy to ensure that the search directions yield descent [Hager 1989]. Common criteria for line search accuracy are the Wolfe conditions [Wolfe 1969, 1971]:

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) - f(\mathbf{x}_k) \le \delta \alpha_k \mathbf{g}_k^\mathsf{T} \mathbf{d}_k, \tag{5}$$

$$\mathbf{g}_{k+1}^\mathsf{T} \mathbf{d}_k \ge \sigma \mathbf{g}_k^\mathsf{T} \mathbf{d}_k, \tag{6}$$

where $0 < \delta \le \sigma < 1$. In the "strong Wolfe" conditions, (6) is replaced by $|\mathbf{g}_{k+1}^\mathsf{T} \mathbf{d}_k| \le -\sigma \mathbf{g}_k^\mathsf{T} \mathbf{d}_k$. It has been shown [Dai and Yuan 2000] that for the FR scheme, the strong Wolfe conditions may not yield a direction of descent unless $\sigma \le 1/2$, even for the function $f(\mathbf{x}) = \lambda \|\mathbf{x}\|^2$, where $\lambda > 0$ is a constant. In typical implementations of the Wolfe conditions, it is often most efficient to choose σ close to one. Hence, the constraint $\sigma \le 1/2$, needed to ensure descent, represents a significant restriction in the choice of the line search parameters. For the PRP scheme, the strong Wolfe conditions may not yield a direction of descent for any choice of $\sigma \in (0, 1)$.

Now, let us consider our new conjugate gradient scheme which corresponds to the following choice for the update parameter:

$$\bar{\beta}_k^N = \max\left\{\beta_k^N, \eta_k\right\}, \quad \text{where} \tag{7}$$

$$\eta_k = \frac{-1}{\|\mathbf{d}_k\| \min\{\eta, \|\mathbf{g}_k\|\}}, \tag{8}$$

$$\beta_k^N = \frac{1}{\mathbf{d}_k^\mathsf{T} \mathbf{y}_k} \left(\mathbf{y}_k - 2\mathbf{d}_k \frac{\|\mathbf{y}_k\|^2}{\mathbf{d}_k^\mathsf{T} \mathbf{y}_k} \right)^\mathsf{T} \mathbf{g}_{k+1}. \tag{9}$$

Here, $\eta > 0$ is a constant. The maximization in (7) plays the role of the truncation operation in the PRP+ scheme.

We were led to the new scheme associated with (7) by deleting a term from the search direction for the memoryless quasiNewton scheme of Perry [1977] and Shanno [1978]. More precisely, the direction $\mathbf{d}_{k+1}$ generated by (2) with the update parameter given by (9) is related to the direction $\mathbf{d}_{k+1}^{PS}$ of the Perry/Shanno scheme in the following way:

$$\mathbf{d}_{k+1}^{PS} = \frac{\mathbf{y}_k^\mathsf{T} \mathbf{s}_k}{\|\mathbf{y}_k\|^2} \left(\mathbf{d}_{k+1} + \frac{\mathbf{d}_k^\mathsf{T} \mathbf{g}_{k+1}}{\mathbf{d}_k^\mathsf{T} \mathbf{y}_k} \mathbf{y}_k \right), \tag{10}$$

where $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$. We show in Hager and Zhang [2005] that the $\mathbf{d}_{k+1}$ term in (10) dominates the $\mathbf{y}_k$ term to the right when the cosine of the angle between $\mathbf{d}_k$ and $\mathbf{g}_{k+1}$ is sufficiently small and f is strongly convex. In this case, the directions generated by (2) with update parameter (9) are approximately multiples of $\mathbf{d}_{k+1}^{PS}$.

As shown in a survey article [Hager and Zhang 2006], the update formula (9) is one member of a family of conjugate gradient methods with guaranteed descent; each member of the family corresponds to a parameter $\theta \in [1/4, \infty)$. Different choices for the parameter correspond to differences in the relative importance of conjugacy versus descent. The scheme (9) corresponds to the intermediate parameter value $\theta = 2$.

Observe that when η_k is sufficiently small, $\max\{\beta_k^N, \eta_k\} = \beta_k^N$ and the direction (2) with update parameter (9) coincides with the direction (2) with update parameter (7). We establish in Hager and Zhang [2005] a global convergence result for the scheme corresponding the direction (2) and update parameter (9) when f is strongly convex. For the truncated scheme corresponding to the direction (2) and the update parameter (7), a global convergence result is established for general functions.

The Perry/Shanno scheme, analyzed further in Powell [1977], Shanno and Phua [1980], and Shanno [1985], has global convergence for convex functions and an inexact line search [Shanno 1978], but in general it does not necessarily converge, even when the line search is exact [Powell 1984]. Of course, the Perry/Shanno scheme is convergent if restarts are employed, however, the speed of convergence can decrease. Han et al. [1997] proved that if a standard Wolfe line search is employed, then convergence to a stationary point is achieved when $\lim_{k\to\infty} \|\mathbf{y}_k\|_2 = 0$ and the gradient of f is Lipschitz continuous.

The new scheme (7)–(9) addresses limitations in previous conjugate gradient schemes in the following ways:

—For a line search satisfying the second Wolfe condition (6), and for any choice of f, the iterates satisfy the descent condition

$$\mathbf{g}_k^\mathsf{T}\mathbf{d}_k \le -\frac{7}{8}\|\mathbf{g}_k\|^2. \tag{11}$$

—Jamming is avoided, essentially due to the $\mathbf{y}_k^\mathsf{T}\mathbf{g}_{k+1}$ term in the definition of β_k^N, the same term that appears in the PRP scheme.

—If our scheme is implemented with a line search that satisfies the standard (not strong) Wolfe conditions, then the iterates are globally convergent in the sense that $\liminf_{k\to\infty} \|\mathbf{g}_k\| = 0$.

—For k large, we have $\bar{\beta}_k^N = \beta_k^N$, assuming $\mathbf{d}_k$ is bounded, if $\mathbf{g}_k$ tends to zero. Hence, there is no restart of the conjugate gradient iteration, at least for large k, and the n-step convergence property for strongly convex quadratics is retained.

The theorem concerning the descent properties of the new conjugate gradient scheme given in Hager and Zhang [2005] is repeated here for convenience:

THEOREM 1. *If* $\mathbf{d}_k^\mathsf{T}\mathbf{y}_k \neq 0$ *and*

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \tau\mathbf{d}_k, \quad \mathbf{d}_0 = -\mathbf{g}_0, \tag{12}$$

for any $\tau \in [\beta_k^N, \max\{\beta_k^N, 0\}]$, *then*

$$\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_{k+1} \le -\frac{7}{8}\|\mathbf{g}_{k+1}\|^2. \tag{13}$$

PROOF. Since $\mathbf{d}_0 = -\mathbf{g}_0$, we have $\mathbf{g}_0^\mathsf{T}\mathbf{d}_0 = -\|\mathbf{g}_0\|^2$, which satisfies (13). Suppose $\tau = \beta_k^N$. Multiplying (12) by $\mathbf{g}_{k+1}^\mathsf{T}$, we have

$$\begin{aligned}\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_{k+1} &= -\|\mathbf{g}_{k+1}\|^2 + \beta_k^N\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k \\ &= -\|\mathbf{g}_{k+1}\|^2 + \mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k\left(\frac{\mathbf{y}_k^\mathsf{T}\mathbf{g}_{k+1}}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k} - 2\frac{\|\mathbf{y}_k\|^2\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k}{(\mathbf{d}_k^\mathsf{T}\mathbf{y}_k)^2}\right)\end{aligned}$$

$$= \frac{\mathbf{y}_k^\mathsf{T}\mathbf{g}_{k+1}(\mathbf{d}_k^\mathsf{T}\mathbf{y}_k)(\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k) - \|\mathbf{g}_{k+1}\|^2(\mathbf{d}_k^\mathsf{T}\mathbf{y}_k)^2 - 2\|\mathbf{y}_k\|^2(\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k)^2}{(\mathbf{d}_k^\mathsf{T}\mathbf{y}_k)^2}. \quad (14)$$

We apply the inequality

$$\mathbf{u}^\mathsf{T}\mathbf{v} \le \frac{1}{2}(\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2)$$

to the first term in (14) with

$$\mathbf{u} = \frac{1}{2}(\mathbf{d}_k^\mathsf{T}\mathbf{y}_k)\mathbf{g}_{k+1} \quad \text{and} \quad \mathbf{v} = 2(\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k)\mathbf{y}_k$$

to obtain (13). On the other hand, if $\tau \neq \beta_k^N$, then $\beta_k^N \le \tau \le 0$. After multiplying (12) by $\mathbf{g}_{k+1}^\mathsf{T}$, we have

$$\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_{k+1} = -\|\mathbf{g}_{k+1}\|^2 + \tau\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k.$$

If $\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k \ge 0$, then (13) follows immediately, since $\tau \le 0$. If $\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k < 0$, then

$$\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_{k+1} = -\|\mathbf{g}_{k+1}\|^2 + \tau\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k \le -\|\mathbf{g}_{k+1}\|^2 + \beta_k^N\mathbf{g}_{k+1}^\mathsf{T}\mathbf{d}_k$$

since $\beta_k^N \le \tau \le 0$. Hence, (13) follows by our previous analysis. □

By taking $\tau = \beta_k^N$, we see that the directions generated by (2) and (9) are descent directions when $\mathbf{d}_k^\mathsf{T}\mathbf{y}_k \neq 0$. Since η_k in (7) is negative, it follows that

$$\bar{\beta}_k^N = \max\left\{\beta_k^N, \eta_k\right\} \in [\beta_k^N, \max\{\beta_k^N, 0\}]. \quad (15)$$

Hence, the direction given by (2) and (7) is a descent direction when $\mathbf{d}_k^\mathsf{T}\mathbf{y}_k \neq 0$. [Dai and Yuan 1999, 2001] present conjugate gradient schemes with the property that $\mathbf{d}_{k+1}^\mathsf{T}\mathbf{g}_{k+1} < 0$ when $\mathbf{d}_k^\mathsf{T}\mathbf{y}_k > 0$. If f is strongly convex or the line search satisfies the Wolfe conditions, then $\mathbf{d}_k^\mathsf{T}\mathbf{y}_k > 0$ and the Dai/Yuan schemes yield descent. Note that in (13) we bound $\mathbf{d}_{k+1}^\mathsf{T}\mathbf{g}_{k+1}$ by $-(7/8)||\mathbf{g}_{k+1}||^2$, while for the schemes of [Dai and Yuan 1999, 2001], the negativity of $\mathbf{d}_{k+1}^\mathsf{T}\mathbf{g}_{k+1}$ is established.

The assumption $\mathbf{d}_k^\mathsf{T}\mathbf{y}_k \neq 0$ in Theorem 1 is always fulfilled when f is strongly convex or the line search satisfies the Wolfe condition (6). That is, by (6)

$$\mathbf{y}_k^\mathsf{T}\mathbf{d}_k = (\mathbf{g}_{k+1} - \mathbf{g}_k)^\mathsf{T}\mathbf{d}_k \ge (\sigma - 1)\mathbf{g}_k^\mathsf{T}\mathbf{d}_k.$$

For $k = 0$, we have

$$\mathbf{g}_0^\mathsf{T}\mathbf{d}_0 = -\|\mathbf{g}_0\|^2 \le -\frac{7}{8}\|\mathbf{g}_0\|^2.$$

Utilizing Theorem 1 and proceeding by induction,

$$\mathbf{y}_k^\mathsf{T}\mathbf{d}_k \ge \frac{7}{8}(1 - \sigma)\|\mathbf{g}_k\|^2 \quad (16)$$

for each $k \ge 0$. Hence, $\mathbf{y}_k^\mathsf{T}\mathbf{d}_k > 0$ if $\mathbf{g}_k \neq \mathbf{0}$.

Another algorithm related to our new conjugate gradient scheme is the scheme of Dai and Liao [2001], where the update parameter is given by

$$\beta_k^{DL} = \frac{1}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}(\mathbf{y}_k - t\mathbf{s}_k)^\mathsf{T}\mathbf{g}_{k+1}. \quad (17)$$

Here, $t > 0$ is a constant and $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$. Numerical results are reported in Dai and Liao [2001] for $t = 0.1$ and $t = 1$; for different choices of t, the numerical results are quite different. The method (2) and (7) can be viewed as an adaptive version of (17) corresponding to $t = 2\|\mathbf{y}_k\|^2/\mathbf{s}_k^\mathsf{T}\mathbf{y}_k$.

Our article is organized as follows: In Section 2 we summarize the convergence results established in Hager and Zhang [2005] for the new conjugate gradient scheme, while Section 3 discusses our implementation. The line search utilizes a special secant step to achieve rapid convergence, while high accuracy is achieved by replacing the Wolfe conditions by an approximation which can be checked more reliably than ordinary Wolfe conditions in a neighborhood of a local minimum. Finally, in Section 4 we compare the performance of the new conjugate gradient scheme to the L-BFGS quasiNewton method [Liu and Nocedal 1989, Nocedal 1980], the PRP+ method [Gilbert and Nocedal 1992], and the schemes in Dai and Yuan [1999, 2001]. We use the unconstrained problems in the CUTEr [Bongartz et al. 1995] test problem library, and the performance profiles of Dolan and Moré [2002].

2. REVIEW OF THE CONVERGENCE ANALYSIS

In Hager and Zhang [2005] we prove two types of results. For strongly convex functions, it is shown that for the scheme with $\beta_k = \beta_k^N$, we have

$$\lim_{k\to\infty} \mathbf{g}_k = \mathbf{0}. \tag{18}$$

For more general smooth functions and $\beta_k = \bar{\beta}_k^N$, we have

$$\liminf_{k\to\infty} \|\mathbf{g}_k\| = 0. \tag{19}$$

More precisely, for the strong convergence result (18), we assume that the line search satisfies either the Wolfe's conditions (5)–(6), or Goldstein's conditions [Goldstein 1965], and that there exist constants L and $\mu > 0$ such that

$$\begin{aligned} \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| &\le L\|\mathbf{x}-\mathbf{y}\| \text{ and} \\ \mu\|\mathbf{x}-\mathbf{y}\|^2 &\le (\nabla f(\mathbf{x}) - \nabla f(\mathbf{y}))(\mathbf{x}-\mathbf{y}), \end{aligned} \tag{20}$$

for all $\mathbf{x}$ and $\mathbf{y} \in \mathcal{L}$, where

$$\mathcal{L} = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \le f(\mathbf{x}_0)\}. \tag{21}$$

To obtain the weak convergence result (19), we drop the strong convexity assumption. Instead, we require that the line search satisfies the Wolfe conditions, that the level set $\mathcal{L}$ is bounded, and that f satisfies the Lipschitz condition (20).

3. NUMERICAL IMPLEMENTATION

Our numerical implementation of the conjugate gradient scheme, based on the update parameter (7), is called CG_DESCENT. Each step of the algorithm involves the following two operations:

—We perform a line search to move from the current iterate $\mathbf{x}_k$ to the next iterate $\mathbf{x}_{k+1}$; and

—We evaluate the new search direction $\mathbf{d}_{k+1}$ in (2) using the choice (7) for the update parameter.

In this section we explain how the stepsize α_k is computed. Work focusing on the development of efficient line search algorithms includes Lemaréchal [1981], Al-Baali and Fletcher [1984], Moré and Sorensen [1984], Hager [1989], and Moré and Thuente [1994]. New features of our line search include:

—an approximation to the Wolfe conditions that can be evaluated with greater precision,
—a special secant routine that leads to a rapid reduction in the width of the interval bracketing α_k (an acceptable step), and
—a quadratic step that retains the n-step quadratic convergence property of the conjugate gradient method.

Let α denote a scalar, and define the function

$$\phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k).$$

In principle, we would like to satisfy the Wolfe conditions (5)–(6). In terms of ϕ, these conditions are equivalent to

$$\delta\phi'(0) \geq \frac{\phi(\alpha_k) - \phi(0)}{\alpha_k} \quad \text{and} \quad \phi'(\alpha_k) \geq \sigma\phi'(0), \tag{22}$$

where $0 < \delta \leq \sigma < 1$. Numerically, the first condition in (22) is difficult to satisfy in a neighborhood of a local minimum since $\phi(\alpha) \approx \phi(0)$, and the subtraction $\phi(\alpha_k) - \phi(0)$ is relatively inaccurate [Hager 1988]. More precisely, we show in Hager and Zhang [2005] that a local minimizer is evaluated with accuracy on the order of the square root of the machine epsilon when using (22).

This leads us to introduce the *approximate Wolfe conditions*:

$$(2\delta - 1)\phi'(0) \geq \phi'(\alpha_k) \geq \sigma\phi'(0), \tag{23}$$

where $0 < \delta < .5$ and $\delta \leq \sigma < 1$. The second inequality in (23) is the same as the second inequality in (22), which is equivalent to the second Wolfe condition (6). Since the condition (6) is included in the approximate Wolfe conditions, it follows from Theorem 1, (15), and (16) that the directions generated by (2) and (9), or by (2) and (7) are descent directions when the approximate Wolfe conditions are used in the line search.

The first inequality in (23) is obtained by replacing ϕ by a quadratic interpolant $q(\cdot)$ that matches $\phi(\alpha)$ at $\alpha = 0$ and $\phi'(\alpha)$ at $\alpha = 0$ and $\alpha = \alpha_k$. Evaluating the finite difference quotient in (22) using q in place of ϕ, we have

$$\frac{\phi(\alpha_k) - \phi(0)}{\alpha_k} \approx \frac{q(\alpha_k) - q(0)}{\alpha_k} = \frac{\phi'(\alpha_k) + \phi'(0)}{2}, \tag{24}$$

and after making this substitution for the difference quotient in (22), we obtain the first inequality in (23). Since the subtraction $q(\alpha_k) - q(0)$ is done exactly, we circumvent the errors inherent in the original finite difference $\phi(\alpha_k) - \phi(0)$. When f is quadratic, the approximation (24) is exact and gives a more accurate way to implement the standard Wolfe conditions.

The code CG_DESCENT allows the user to choose between three different objectives in the line search:

V1. Compute a point satisfying the Wolfe conditions (22).

V2. Compute a point satisfying the approximate Wolfe conditions (23).

V3. For the initial iterations, compute a point satisfying the Wolfe conditions (22). If at iteration k the following condition is satisfied, then switch permanently to the approximate Wolfe conditions:

$$|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| \le \omega C_k, \tag{25}$$

where

$$\begin{cases} Q_k = 1 + Q_{k-1}\Delta, & Q_{-1} = 0, \\ C_k = C_{k-1} + (|f(\mathbf{x}_k)| - C_{k-1})/Q_k, & C_{-1} = 0. \end{cases} \tag{26}$$

Here, $\Delta \in [0, 1]$ is a parameter used in the averaging of the previous absolute function values, and $\omega \in [0, 1]$ is a parameter used to determine when to switch from the Wolfe to the approximate Wolfe conditions. As Δ approaches 0, we give more weight to the most recent function values. As the iteration difference $|f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})|$ in (25) tends to zero, we cannot satisfy the first Wolfe condition (5) due to numerical errors. Hence, when the difference is sufficiently small, we switch to the approximate Wolfe conditions.

There is a fundamental difference between the computation of a point satisfying the Wolfe conditions, and the computation of a point satisfying the approximate Wolfe conditions. Moré and Thuente [1994] develop an algorithm for finding a point satisfying the Wolfe conditions that is based on computing a local minimizer of the function ψ defined by

$$\psi(\alpha) = \phi(\alpha) - \phi(0) - \alpha\delta\phi'(0).$$

Since $\psi(0) = 0$, it is required that the local minimizer α^* satisfy

$$\psi(\alpha^*) < 0 \quad \text{and} \quad \psi'(\alpha^*) = 0.$$

Together, these two relations imply that the Wolfe conditions hold in a neighborhood of α^* if $\delta < \sigma$.

In contrast to the Wolfe conditions (22), the approximate Wolfe conditions (23) are satisfied at a minimizer of ϕ. Hence, when trying to satisfy the approximate Wolfe condition we focus on minimizing ϕ; when trying to satisfy the usual Wolfe condition we focus on minimizing ψ. Although there is no theory to guarantee convergence when using the approximate Wolfe conditions, we pointed out earlier that there is a numerical advantage in using the approximate Wolfe conditions—we can compute local minimizers with accuracy on the order of the machine epsilon, rather than with accuracy on the order of the square root of the machine epsilon. We now observe that there can be a speed advantage associated with the approximate Wolfe conditions.

Recall that the conjugate gradient method has an n-step quadratic convergence property when α_k is the minimum of $\phi(\cdot)$ (see [Cohen 1972], also [Hirst 1989] for a result concerning the required accuracy in the line search minimum). However, if the line search is based on the Wolfe conditions and the function ψ is minimized instead of ϕ, then the n-step quadratic convergence property is lost.

For a general function f, the minimization of ψ when implementing the Wolfe conditions represents an approximate minimization of f in the search direction (due to the linear term in the definition of ψ). By focusing on the function ϕ, as we do in the approximate Wolfe conditions, we usually obtain better approximations to the minimum of f, since we are minimizing the actual function we wish to minimize rather than an approximation to it.

We now give a detailed description of the algorithm used to generate a point satisfying the approximate Wolfe conditions (23). The line search algorithm represents an approach for computing a local minimizer of $\phi(\cdot)$ on the interval $[0, \infty)$. The line search is terminated whenever an iterate α_k is generated with the following property:

T1. Either the original Wolfe conditions (22) are satisfied, or

T2. the approximate Wolfe conditions (23) are satisfied and

$$\phi(\alpha_k) \le \phi(0) + \epsilon_k, \tag{27}$$

where $\epsilon_k \ge 0$ is an estimate for the error in the value of f at iteration k.

In our code, we incorporate the following possible expressions for the error in the function value:

$$\epsilon_k = \epsilon C_k \quad \text{or} \quad \epsilon_k = \epsilon, \tag{28}$$

where ϵ is a (small) user-specified constant and C_k is an estimate for the function value size generated by the recurrence (26). We would like to satisfy the original Wolfe conditions, so we terminate the line search in T1 whenever they are satisfied. However, numerically, f is flat in a neighborhood of a local minimum and the first inequality in (22) is never satisfied. In this case, we terminate the line search when the approximate Wolfe conditions are satisfied. Due to the constraint (27) in T2, we only terminate the line search using the approximate Wolfe conditions when the value of f at the accepted step is not much larger than the value of f at the previous iterate—due to numerical errors, the value of f at an acceptable step may be larger than the value of f at the previous iterate. The parameter ϵ_k in (27) allows for a small growth in the value of f.

Our algorithm for computing a point that satisfies either T1 or T2 generates a nested sequence of (bracketing) intervals. A typical interval $[a, b]$ in the nested sequence satisfies the following *opposite slope condition*:

$$\phi(a) \le \phi(0) + \epsilon_k, \quad \phi'(a) < 0, \quad \phi'(b) \ge 0. \tag{29}$$

In Moré and Thuente [1994], the bracketing intervals used in the computation of a point satisfying the Wolfe conditions satisfy the following relations:

$$\psi(a) \le \psi(b), \quad \psi(a) \le 0, \quad \psi'(a)(b - a) < 0.$$

We prefer to use the relations (29) to describe the bracketing interval since we view the function derivative, in finite precision arithmetic, as more reliable than the function value in computing a local minimizer.

Given a bracketing interval $[a, b]$ satisfying (29) and given an update point c, we now explain how we will update the bracketing interval. The input of this procedure is the current bracketing interval $[a, b]$ and a point c generated by either a secant step, or a bisection step, as explained shortly. The output of the procedure is the updated bracketing interval $[\bar{a}, \bar{b}]$. In the *interval update rules* appearing below, θ denotes a parameter in $(0, 1)$ ($\theta = 1/2$ corresponding to the bisection method):[1]

$$[\bar{\mathbf{a}}, \bar{\mathbf{b}}] = \mathbf{update}\ (\mathbf{a}, \mathbf{b}, \mathbf{c})$$

```
U0. If $c \notin (a, b)$, then $\bar{a} = a$, $\bar{b} = b$, and return.
U1. If $\phi'(c) \geq 0$, then $\bar{a} = a$, $\bar{b} = c$, and return.
U2. If $\phi'(c) < 0$ and $\phi(c) \leq \phi(0) + \epsilon_k$, then $\bar{a} = c$, $\bar{b} = b$, and return.
U3. If $\phi'(c) < 0$ and $\phi(c) > \phi(0) + \epsilon_k$, then $\bar{a} = a$, $\bar{b} = c$, and do the following:
    a. Set $d = (1 - \theta)\bar{a} + \theta\bar{b}$; if $\phi'(d) \geq 0$, then $\bar{b} = d$ and return.
    b. If $\phi'(d) < 0$ and $\phi(d) \leq \phi(0) + \epsilon_k$, then $\bar{a} = d$ and go to a.
    c. If $\phi'(d) < 0$ and $\phi(d) > \phi(0) + \epsilon_k$, then $\bar{b} = d$ and go to a.
```

After completing U1–U3, we obtain a new interval $[\bar{a}, \bar{b}] \subset [a, b]$ whose endpoints satisfy (29). The loop embedded in U3a–U3c should terminate since the interval width $\bar{b} - \bar{a}$ tends to zero, and at the endpoints, the following conditions hold:

$$\begin{aligned} \phi'(\bar{a}) &< 0, \quad \phi(\bar{a}) \leq \phi(0) + \epsilon_k \\ \phi'(\bar{b}) &< 0, \quad \phi(\bar{b}) > \phi(0) + \epsilon_k \end{aligned}$$

The input c for the update routine is generated by a special secant step. If c is obtained from a secant step based on function values at a and b, then we write

$$c = \text{ secant } (a, b) = \frac{a\phi'(b) - b\phi'(a)}{\phi'(b) - \phi'(a)}.$$

The secant step used in our line search, denoted secant^2, is defined in the following way:

$$[\bar{\mathbf{a}}, \bar{\mathbf{b}}] = \mathbf{secant^2}(\mathbf{a}, \mathbf{b})$$

```
S1. $c =$ secant $(a, b)$ and $[A, B] =$ update $(a, b, c)$.
S2. If $c = B$, then $\bar{c} =$ secant $(b, B)$.
S3. If $c = A$, then $\bar{c} =$ secant $(a, A)$.
S4. If $c = A$ or $c = B$, then $[\bar{a}, \bar{b}] =$ update $(A, B, \bar{c})$. Otherwise, $[\bar{a}, \bar{b}] = [A, B]$.
```

As explained in Hager and Zhang [2005], statement S1 typically updates one side of the bracketing interval, while S4 updates the other side. The convergence rate of secant^2 is $1 + \sqrt{2} \approx 2.4$ [Hager and Zhang 2005, Thm. 3.1].

The following routine is used to generate an initial interval $[a, b]$ satisfying the opposite slope condition (29), beginning with the initial guess $[0, c]$.

$$[\mathbf{a}, \mathbf{b}] = \mathbf{bracket}\ (\mathbf{c})$$

```
B0. Initialize $j = 0$ and $c_0 = c$.
```

[1]The termination rules T1–T2, the update rule U0–U3, and the secant rules S1–S4 also appear in Hager and Zhang [2005]; they are repeated here for continuity.

B1. If $\phi'(c_j) \geq 0$, then $b = c_j$ and $a = c_i$, where $i < j$ is the largest integer such that $\phi(c_i) \leq \phi(0) + \epsilon_k$, and return.

B2. If $\phi'(c_j) < 0$ and $\phi(c_j) > \phi(0) + \epsilon_k$, then return after generating a and b using U3a–c with the initialization $\bar{a} = 0$ and $\bar{b} = c_j$.

B3. Otherwise, set $c_{j+1} = \rho c_j$, increment j, and go to B1.

Here, $\rho > 0$ is the factor by which c_j grows in each step of the bracket routine. We continue to let c_j expand until either the slope $\phi'(c_j)$ becomes nonnegative, activating B1, or the function value $\phi(c_j)$ is large enough to activate B2.

Next, we give the rules used to generate the starting guess c used by the bracket routine. The parameter QuadStep in I1 is explained later, while $\|\cdot\|_\infty$ stands for the sup-norm (maximum absolute component of the vector).

$$[\mathbf{c}] = \textbf{initial}\ (\mathbf{k})$$

I0. If $k = 0$ and the user does not specify the starting point in the line search, then it is generated by the following rules:
  (a) If $\mathbf{x}_0 \neq \mathbf{0}$, then $c = \psi_0 \|\mathbf{x}_0\|_\infty / \|\mathbf{g}_0\|_\infty$ and return.
  (b) If $f(\mathbf{x}_0) \neq 0$, then $c = \psi_0 |f(\mathbf{x}_0)| / \|\mathbf{g}_0\|^2$ and return.
  (c) Otherwise, $c = 1$ and return.

I1. If QuadStep is true, $\phi(\psi_1 \alpha_{k-1}) \leq \phi(0)$, and the quadratic interpolant $q(\cdot)$ that matches $\phi(0)$, $\phi'(0)$, and $\phi(\psi_1 \alpha_{k-1})$ is strongly convex with a minimizer α_q, then $c = \alpha_q$ and return.

I2. Otherwise, $c = \psi_2 \alpha_{k-1}$.

The rationale for the starting guesses given in I0 is the following: If $f(\mathbf{x}) \approx d_0 + d_2 \mathbf{x}^\mathsf{T}\mathbf{x}$, where $d_2 > 0$, then the minimum is attained at $\mathbf{x} = \mathbf{0}$. The step that yields this minimum is $\alpha = \|\mathbf{x}_0\|_\infty / \|\mathbf{g}_0\|_\infty$. Since this estimate is obviously crude, we multiply by a small scalar ψ_0 in order to keep the starting guess close to $\mathbf{x}_0$. If $\mathbf{x}_0 = \mathbf{0}$, then this estimate is unsuitable, and we consider the approximation $f(\mathbf{x}) \approx d_2 \mathbf{x}^\mathsf{T}\mathbf{x}$ and the corresponding optimal step $\alpha = 2|f(\mathbf{x}_0)| / \|\mathbf{g}_0\|^2$. Again, we multiply this estimate by a small scalar ψ_0 in I0b. If $f(\mathbf{x}_0) = 0$, then this guess is unsuitable, and we simply take $c = 1$.

For $k > 0$, we exploit the previous step α_{k-1} in determining the new initial step, and we utilize a result of Hirst [1989], which is roughly the following: For a line search done with "quadratic accuracy," the conjugate gradient method retains the n-step local quadratic convergence property established by Cohen [1972]. More precisely, let $q(\cdot)$ denote the quadratic interpolant that matches $\phi(0)$, $\phi'(0)$, and $\phi(R)$, and let α_q denote the minimizer of q, assuming it exists. Hirst [1989] shows that if

$$\phi(R) \leq \phi(0) \quad \text{and} \quad \phi(\alpha_k) \leq \phi(\alpha_q),$$

then the FR, PRP, HS, and D conjugate gradient schemes all preserve the n-step quadratic convergence property. Although our new scheme was not known at the time of Hirst's work, we anticipate that Hirst's analysis will apply to the new scheme (since it is related to both the PRP and HS schemes that Hirst did analyze).

If the parameter QuadStep is true, then we attempt to find a point that complies with the conditions needed in Hirst's analysis of n-step quadratic convergence. By Theorem 1, the current search direction is a descent direction. Hence, a small positive R should satisfy the requirement $\phi(R) \le \phi(0)$. More precisely, we try $R = \psi_1\alpha_{k-1}$ in I1 where $0 < \psi_1 < 1$. If the quadratic interpolant in I1 has no minimum or QuadStep is false, then we simply take $c = \psi_2\alpha_{k-1}$, where $\psi_2 > 1$ since we wish to avoid expanding c further in the bracket routine.

We now give a complete statement of the line search procedure, beginning with a list of the parameters.

Line Search/CG_DESCENT Parameters

$\delta-$	range $(0, .5)$, used in the Wolfe conditions (22) and (23)
$\sigma-$	range $[\delta, 1)$, used in the Wolfe conditions (22) and (23)
$\epsilon-$	range $[0, \infty)$, used in the approximate Wolfe termination (T2)
$\omega-$	range $[0, 1]$, used in switching from Wolfe to approximate Wolfe conditions
$\Delta-$	range $[0, 1]$, decay factor for Q_k in the recurrence (26)
$\theta-$	range $(0, 1)$, used in the update rules when the potential intervals $[a, c]$ or $[c, b]$ violate the opposite slope condition contained in (29)
$\gamma-$	range $(0, 1)$, determines when a bisection step is performed (L2 below)
$\eta-$	range $(0, \infty)$, enters into the lower bound for β_k^N in (7) through η_k
$\rho-$	range $(1, \infty)$, expansion factor used in the bracket rule B3.
ψ_0-	range $(0, 1)$, small factor used in starting guess I0.
ψ_1-	range $(0, 1)$, small factor used in I1.
ψ_2-	range $(1, \infty)$, factor multiplying previous step α_{k-1} in I2.

Line Search Algorithm

```
L0. $c =$ initial $(k)$, $[a_0, b_0] =$ bracket $(c)$, and $j = 0$.
L1. $[a, b] =$ secant$^2(a_j, b_j)$.
L2. If $b - a > \gamma(b_j - a_j)$, then $c = (a + b)/2$ and $[a, b] =$ update $(a, b, c)$.
L3. Increment $j$, set $[a_j, b_j] = [a, b]$, and go to L1.
```

The line search is terminated whenever a point is generated satisfying T1/T2.

4. NUMERICAL COMPARISONS

In this section we compare the performance of the new conjugate gradient method, denoted CG_DESCENT, to the L-BFGS limited memory quasiNewton method of Nocedal [1980] and Liu and Nocedal [1989], and to other conjugate gradient methods. We considered both the PRP+ version of the conjugate gradient method developed by Gilbert and Nocedal [1992] where the β_k associated with the Polak-Ribière-Polyak conjugate gradient method [Polak and Ribière 1969; Polyak 1969] is kept nonnegative, and versions of the conjugate gradient method developed by [Dai and Yuan 1999, 2001], denoted CGDY and DYHS, which achieve descent for any line search that satisfies the Wolfe conditions

(5)–(6). The hybrid conjugate gradient method DYHS uses

$$\beta_k = \max\left\{0, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\},$$

where β_k^{HS} is the choice of Hestenes-Stiefel (3) and β_k^{DY} is defined in (4). The test problems are the unconstrained problems in the CUTEr [Bongartz et al. 1995] test problem library.

The L-BFGS and PRP+ codes were obtained from Jorge Nocedal's web page. The L-BFGS code is authored by Jorge Nocedal, while the PRP+ code is co-authored by Guanghui Liu, Jorge Nocedal, and Richard Waltz. In the documentation for the L-BFGS code, it is recommended that between three and seven vectors be used for the memory. Hence, we chose five vectors for the memory. The line search in both codes is a modification of subroutine CSRCH of Moré and Thuente [1994], which employs various polynomial interpolation schemes and safeguards in satisfying the strong Wolfe line search conditions.

We also manufactured a new L-BFGS code by replacing the Moré/Thuente line search by the new line search presented in our article. We call this new code L-BFGS*. The new line search would need to be modified for use in the PRP+ code to ensure descent. Hence, we retained the Moré/Thuente line search in the PRP+ code. Since the conjugate gradient algorithms of Dai and Yuan achieve descent for any line search that satisfies the Wolfe conditions, we are able to use the new line search in our experiments with CGDY and with DYHS. All codes were written in Fortran and compiled with f77 (default compiler settings) on a Sun workstation.

Our line search algorithm uses the following values for the parameters:

$$\delta = .1, \quad \sigma = .9, \quad \epsilon = 10^{-6}, \quad \theta = .5, \quad \gamma = .66, \quad \eta = .01, \quad \rho = 5,$$
$$\omega = 10^{-3}, \quad \Delta = .7, \quad \psi_0 = .01, \quad \psi_1 = .1, \quad \psi_2 = 2.$$

Our rationale for these choices was the following: The constraints on δ and σ are $0 < \delta \le \sigma < 1$, and $\delta < .5$. As δ approaches 0 and σ approaches 1, the line search terminates more quickly. The chosen values $\delta = .1$ and $\sigma = .9$ represent a compromise between our desire for rapid termination and our desire to improve the function value. When using the approximate Wolfe conditions, we would like to achieve decay in the function value if numerically possible. Hence, we made the small choice $\epsilon = 10^{-6}$, and we used the first estimate in (28) for the function error. When restricting β_k in (7), we would like to avoid truncation if possible, since the fastest convergence for a quadratic function is obtained when there is no truncation at all. The choice $\eta = .01$ leads to infrequent truncation of β_k. The choice $\gamma = .66$ ensures that the length of the interval $[a, b]$ decreases by a factor of 2/3 in each iteration of the line search algorithm. The choice $\theta = .5$ in the update procedure corresponds to bisection. As ω tends to zero, the line search becomes a Wolfe line search as opposed to an approximate Wolfe line search. Taking $\omega = 10^{-3}$, we switch to the approximate Wolfe line search when the cost function converges to three digits. If the cost function vanishes at a minimizer, then an estimate of the form $\epsilon f(\mathbf{x}_k)$ is often a poor approximation for the error in the value of f. By taking $\Delta = .7$ in (26), the parameter C_k in (26) goes to zero more slowly than $f(\mathbf{x}_k)$, and hence, we often obtain a less poor estimate for the error in the function value. In the routine to initialize the line search, we take $\psi_0 = .01$ to keep the initial step close to the starting point. We would like the guess $\alpha = \psi_1 \alpha_{k-1}$ in I1 to satisfy the descent condition $\phi(\psi_1 \alpha_{k-1}) \le \phi(0)$, but we do not want the guess to be a poor approximation to a minimizer of ϕ. Hence, we take $\psi_1 = .1$. When the line search does not start with a quadratic interpolation step, we take $\psi_2 = 2$, in which case the initial guess in I2 is twice the previous stepsize α_{k-1}. We take $\psi_2 > 1$ in an effort to avoid future expansions in the bracket routine.

The first set of experiments use CUTEr unconstrainted test problems with dimensions between 50 and 10^4. We downloaded all the unconstrained test problems and then deleted a problem in any of the following cases:

(D1) The problem was small (dimension less than 50).
(D2) The problem could be solved in, at most, .01 seconds by any of the solvers.
(D3) The cost function seemed to have no lower bound.
(D4) The cost function generated a "NaN" for what seemed to be a reasonable choice for the input.
(D5) The problem could not be solved by any of the solvers (apparently due to nondifferentiability of the cost function).
(D6) Different solvers converged to different local minimizers (that is, the optimal costs were different).

In the Fall of 2004, there are 160 unconstrained optimization problems in the CUTEr test. After the deletion process, we were left with 106 problems. For some problems the dimension could be chosen arbitrarily. In these cases, we often ran two versions of the test problem, one with twice as many variables as the other.

Nominally, our termination criterion was the following:

$$\|\nabla f(\mathbf{x}_k)\|_\infty \le \max\{10^{-6}, 10^{-12}\|\nabla f(\mathbf{x}_0)\|_\infty\}. \qquad (30)$$

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 (30) is satisfied. As a result, different solvers obtain completely different values for the cost, and the test problem would be discarded by (D6). By changing the convergence criterion to $\|\nabla f(\mathbf{x}_k)\|_\infty \le 10^{-6}$, the computed costs all agreed 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 on William Hager's webpage.[2] Here, we analyze the performance data using the profiles of Dolan and Moré [2002]. That is, for subsets of the methods being analyzed, we plot the fraction P of problems for which any given method is within a factor τ of the best time. In a performance profile plot, 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

[2]http://www.math.ufl.edu/~hager/papers/CG

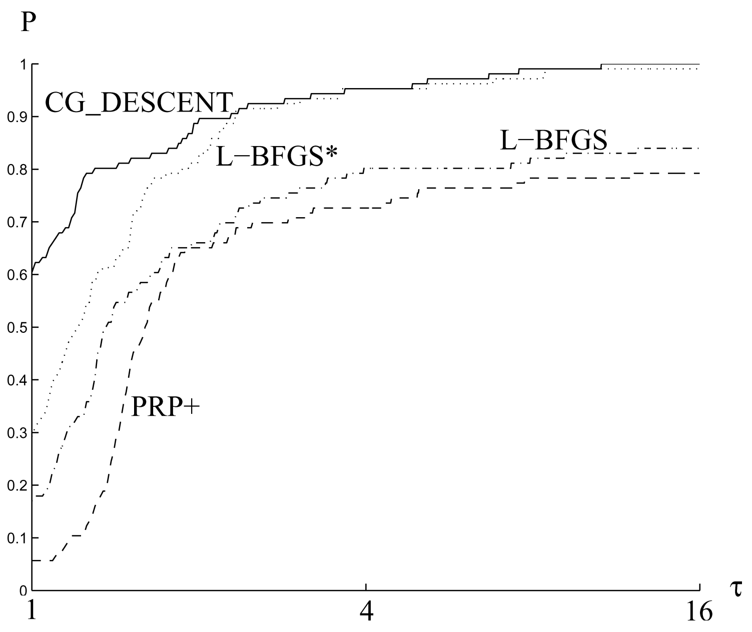

Fig. 1. Performance based on CPU time for CG and L-BFGS codes.

Table I. Number of Times Each Method was Fastest (time metric, stopping criterion (30))

Method	Fastest
CG_DESCENT	64
L-BFGS*	31
L-BFGS	18
PRP+	6

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 Figure 1, we use CPU time to compare the performance of CG_DESCENT to that of L-BFGS, L-BFGS*, and PRP+. Both the CG_DESCENT and L-BFGS* codes use the approximate Wolfe line search. Since the L-BFGS* curve lies above the L-BFGS curve, the L-BFGS* algorithm benefited from the new line search. The best performance, relative to the CPU time metric, was obtained by CG_DESCENT, the top curve in Figure 1. For this collection of methods, the number of times any method achieved the best time is shown in Table I. The column total in Table I exceeds 106 due to ties for some test problems.

In Figure 2, we use CPU time to compare the performance of the conjugate gradient codes CG_DESCENT, CGDY, DYHS, and PRP+. Figure 2 indicates that, relative to the CPU time metric, CG_DESCENT is fastest, then DYHS, then CGDY, and then PRP+. Since the three fastest codes use the same line search, these codes only differ in their choice of the search direction (as dictated

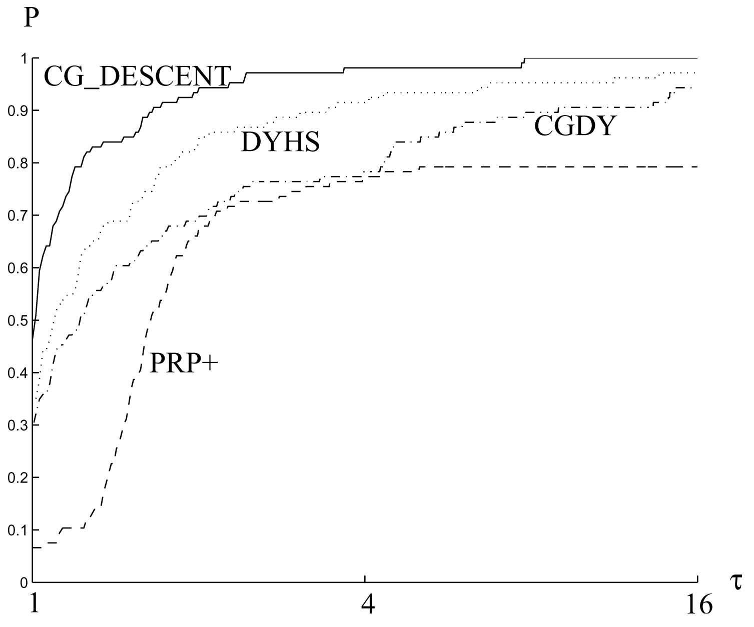

Fig. 2. Performance based on CPU time for CG codes.

Table II. Number of Times Each Method was Fastest (time metric, stopping criterion (30))

Method	Fastest
CG_DESCENT	49
DYHS	36
CGDY	31
PRP+	7

by their choice for β_k). Hence, CG_DESCENT appears to generate the best search directions, on average. For this collection of methods, the number of times each method achieved the best time appears in Table II.

In Figure 3, we compare the performance of CG_DESCENT for various choices of its parameters. The dashed curve corresponds to the approximate Wolfe line search and the parameter QuadStep = .false. Clearly, skipping the quadratic interpolation step at the start of each iteration increases the CPU time. On the other hand, if we wait long enough we still solve the problems, since the dashed curves eventually reaches the same limit as the top two curves. The top dotted curve in Figure 3 corresponds to a hybrid scheme in which a Wolfe line search is employed until (25) holds, with $\omega = 10^{-3}$, followed by an approximate Wolfe line search; the performance of the pure approximate Wolfe line search (top solid curve) is slightly below the performance of the hybrid scheme. The bottom solid curve, corresponding to a pure Wolfe line search, is inferior to either an approximate Wolfe line search or the hybrid scheme.

For these variations of CG_DESCENT, the number of times each variation achieved the best time appears in Table III. Except for Figure 3, all plots in this article are for a pure approximate Wolfe line search.

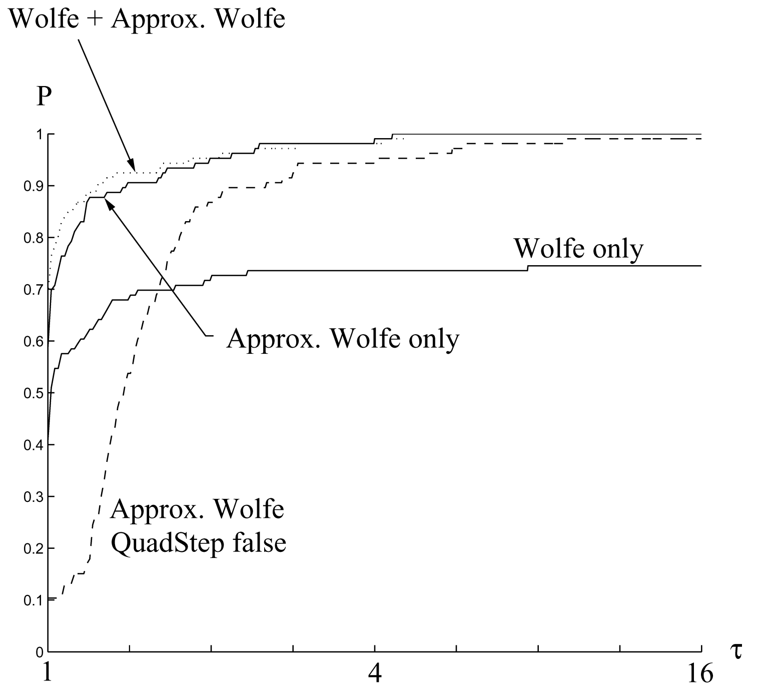

Fig. 3. Performance based on CPU time for various choices of CG_DESCENT parameters.

Table III. Number of Times Each Variation of CG_DESCENT was Fastest

Method	Fastest
Wolfe followed by Approximate Wolfe	73
Approximate Wolfe only	62
Wolfe only	43
Approximate Wolfe, QuadStep false	11

In the next series of experiments, we use a stopping criterion of the form:

$$\|\nabla f(\mathbf{x}_k)\|_\infty \le 10^{-6}(1 + |f(\mathbf{x}_k)|). \quad (31)$$

Except in cases where f vanishes at an optimum, this criterion often leads to quicker termination than the previous criterion (30) and to less accurate solutions. In fact, for some problems, where f is large at the starting point, many of the algorithms terminated almost immediately, far from the optimum. For example, if $f(x) = x^2$ is a scalar function, then $f'(x) = 2x$ and for any large x, (31) is satisfied. The problems where we encounter quick termination due to large f at a starting point are DQRTIC, PENALTY1, QUARTC, VARDIM, WOODS, ARGLINB, ARGLINC, PENALTY2, and NONCVXUN. These problems were resolved using the previous stopping criterion.

In Figure 4, we compare the time performance of the solvers using the criterion (31). In CG_DESCENT we used the approximate Wolfe line search and QuadStep is true. Observe that with this weaker stopping criterion, PRP+

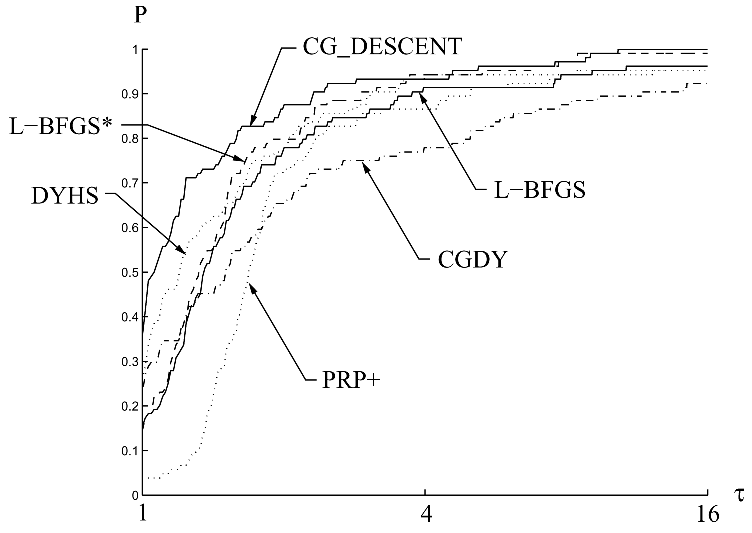

Fig. 4. Performance based on CPU time for stopping criterion (31).

Table IV. Number of Times Each Method was Fastest (time metric, stopping criterion (31))

Method	Fastest	Method	Fastest
CG_DESCENT	37	L-BFGS	16
DYHS	27	L-BFGS*	15
CGDY	24	PRP+	4

solves a larger fraction of the test problems. Again, the best performance in this test set is obtained by CG_DESCENT. In Table IV we give the number of times each method achieved the best time.

In the next experiment, we compare performance based on the number of function and gradient evaluations, using the stopping criterion (30). For our CUTEr test set, we found that, on average, the CPU time to evaluate the derivative of f was about 3 times the CPU time to evaluate f itself. Figure 5 gives the performance profiles based on the metric

$$\mathrm{NF} + 3\mathrm{NG}, \tag{32}$$

where NF is the number of function evaluations and NG is the number of gradient evaluations. Observe that relative to this metric, L-BFGS* (L-BFGS with the new line search) achieved the top performance, followed by CG_DESCENT. In Table V we give the number of times each method achieved the best time in the function evaluation metric (32).

In Figure 6 we compare performance based on number of iterations and the stopping criterion (30). Notice that relative to the number of iterations, CG_DESCENT and L-BFGS* have almost identical performance. Also, it is interesting to observe in Figure 6 that the PRP+ code is the top performer, relative to the iteration metric, for values of τ near 1. In Table VI we give the number of

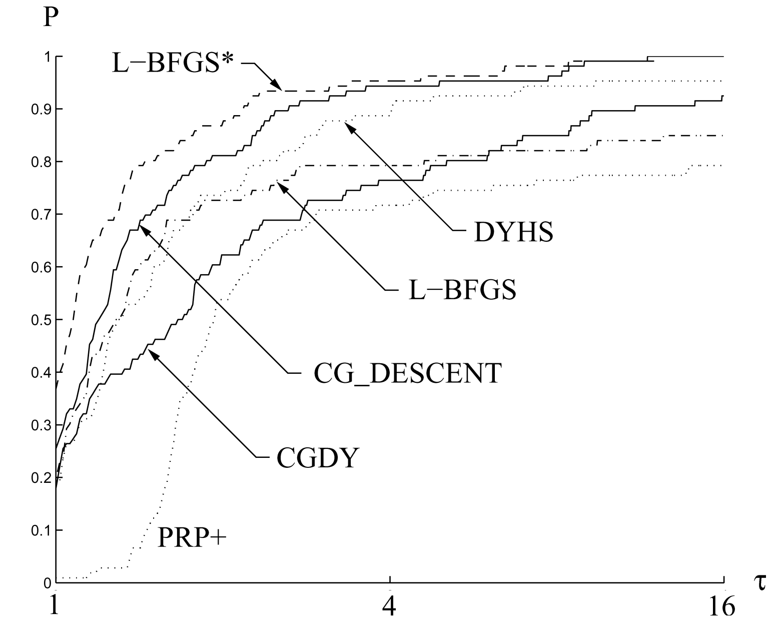

Fig. 5. Performance based on number of function evaluations and stopping criterion (30).

Table V. Number of Times Each Method was Fastest (function evaluation metric, stopping criterion (30))

Method	Fastest	Method	Fastest
L-BFGS*	39	DYHS	20
CG_DESCENT	27	CGDY	19
L-BFGS	21	PRP+	1

times each method achieved the best time in the iteration metric. Since PRP+ performs well in the iteration metric for τ near 1, we conclude that the overall poor performance of PRP+ in the time and function evaluation metrics is connected with the poor performance of the line search. In particular, the line search in the PRP+ scheme must achieve sufficient accuracy to ensure descent, while with CG_DESCENT the search directions are always descent directions, independent of the accuracy in the line search. Hence, with CG_DESCENT the line search terminates as soon as either the Wolfe or approximate Wolfe conditions are satisfied, without having to further improve accuracy to achieve a descent direction.

Together, Figures 1, 5, and 6 seem to imply the following: CG_DESCENT and L-BFGS* require, on average, almost the same number of iterations to achieve a given error tolerance. In the line search more function evaluations are needed by CG_DESCENT to achieve the stopping criterion, while with L-BFGS* the initial step $\alpha = 1$ is acceptable quite often. On the other hand, the linear algebra in the L-BFGS code to update the search direction is more time consuming than the linear algebra in CG_DESCENT. Hence, the reduction in

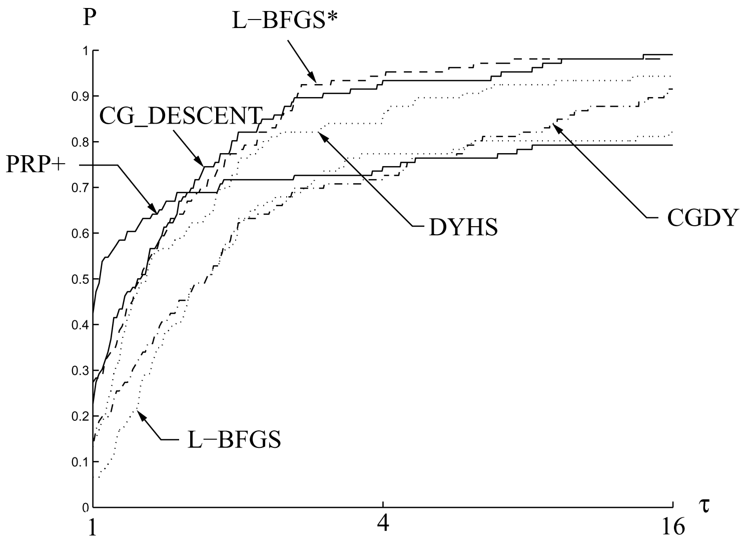

Fig. 6. Performance based on iterations and stopping criterion (30).

Table VI. Number of Times Each Method was Fastest (iteration metric, stopping criterion (30))

Method	Fastest	Method	Fastest
PRP+	45	DYHS	16
L-BFGS*	29	CGDY	14
CG_DESCENT	24	L-BFGS	7

the number of function evaluations seen in Figure 5 for L-BFGS* is dominated in Figure 1 by the cost of the linear algebra.

As discussed in Section 1, the truncation of β_k in (7) for CG_DESCENT can be controlled through the parameters η_k and η. First, $\mathbf{g}_k$ tends to zero, helping to make η_k small, and second, by taking η near zero we can make η_k small even when $\|\mathbf{g}_k\|$ is large. Powell's example [1984] reveals that truncation may be needed to ensure convergence, but from the practical viewpoint truncation impedes the convergence of the conjugate gradient method. In this set of test problems we found that there were 33 problems where CG_DESCENT truncated a step, and 64 problems where PRP+ truncated a step. Altogether, there were 89 truncation steps with CG_DESCENT and 172 truncation steps with PRP+. Hence, formula (7) reduced the number of truncations when compared to PRP+. We also tried $\eta = 10^{-6}$, in which case there were no truncations at all in CG_DESCENT. With this smaller value for η there were 51 problems where the convergence speed improved and 33 problems where the convergence was slower. Hence, by decreasing η, there were fewer truncations and a slight improvement in convergence speed.

In Table VII we illustrate the accuracy of the algorithms and line search. We solved problem CURLY10 in the CUTEr library with dimension 1000 and with various tolerances. The top two methods, CG_DESCENT and L-BFGS*, use the new line search based on the approximate Wolfe conditions (23), while

Table VII. Solution Time in Seconds versus Tolerance

Algorithm	Tolerance $\|\mathbf{g}_k\|_\infty$					
Name	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
CG_DESCENT	10.04	17.13	25.26	27.49	32.03	39.79
L-BFGS*	14.80	19.46	31.12	36.30	46.86	54.43
L-BFGS	16.48	22.63	33.36	F	F	F
PRP+	17.80	24.13	F	F	F	F

the bottom two methods, L-BFGS and PRP+, use the Moré/Thuente line search based on the usual Wolfe conditions (22). An F in the table means that the line search terminated before the convergence tolerance for $\|\mathbf{g}_k\|_\infty$ was satisfied. According to the documentation for the line search in the L-BFGS and PRP+ codes, "rounding errors prevent further progress. There may not be a step which satisfies the sufficient decrease and curvature conditions. Tolerances may be too small."

This problem was chosen since it illustrates the typical performance that we saw in the test problems. That is, first the PRP+ scheme fails, and shortly thereafter, the L-BFGS scheme fails. Much later, the codes using the approximate Wolfe line search fail. For the CURLY10 test problem, we continued to reduce the convergence tolerance to 10^{-12} without failure. The solution time was 78.31 seconds for CG_DESCENT and 101.36 seconds for L-BFGS*. Additional comparisons, like the one given in Table VII, appear in Hager and Zhang [2005]. Roughly, a line search based on the Wolfe conditions can compute a solution with accuracy on the order of the square root of the machine epsilon, while a line search that also includes the approximate Wolfe conditions can compute a solution with accuracy on the order of the machine epsilon.

5. CONCLUSIONS

We have presented the recently introduced conjugate gradient algorithm, which we call CG_DESCENT, for solving unconstrained optimization problems. Although the update formula (7)–(9) is more complicated than previous formulas, the scheme is relatively robust in numerical experiments. We prove [Hager and Zhang 2005] that it satisfies the descent condition $\mathbf{g}_k^\mathsf{T}\mathbf{d}_k \le -\frac{7}{8}\|\mathbf{g}_k\|^2$ for either a Wolfe or an approximate Wolfe line search (see Theorem 1 in Section 1). We prove [Hager and Zhang 2005] global convergence under the standard (not strong) Wolfe conditions. A new line search was introduced that utilizes the approximate Wolfe conditions; this approximation provides a more accurate way to check the usual Wolfe conditions when the iterates are near a local minimizer. Our line search algorithm exploits a double secant step, denoted secant^2, that is designed to achieve rapid decay in the width of the interval which brackets an acceptable step. For a test set consisting of 106 problems from the CUTEr library with dimensions ranging between 50 and 10,000, the CPU time performance profile for CG_DESCENT was higher than those of CGDY, DYHS, PRP+, L-BFGS, and L-BFGS* (L-BFGS implemented using our new line search). The second best performance in the time metric was achieved by either L-BFGS* or DYHS. In the function evaluation metric, L-BFGS* had the top performance, followed by CG_DESCENT. On average, CG_DESCENT requires slightly more function/gradient evaluations in the line search, while the L-BFGS* line search frequently terminates at the initial step $\alpha = 1$. The better time performance of CG_DESCENT is due to the fact that the update of the search direction is less time consuming than the corresponding update in L-BFGS.

In the iteration metric, PRP+ had the best performance for τ near 1, followed by CG_DESCENT and L-BFGS*, whose performances were very similar. The latter two had the top performance in the iteration metric for larger values of τ. The better performance of CG_DESCENT relative to PRP+ in the time and function evaluation metrics was connected with the line search. With PRP+, the line search accuracy may need to be increased to achieve descent. The increased accuracy requires more evaluations of the cost function and its gradient; with CG_DESCENT, the search directions are always descent directions, independent of the line search accuracy. Since our implementation of CGDY and DYHS use the same line search as we use in CG_DESCENT, the better performance of CG_DESCENT is due to the generation of better search directions, on average.

A copy of the code CG_DESCENT, along with the User's Guide [Hager and Zhang 2004], are posted on William Hager's webpage.

ACKNOWLEDGMENTS

Initial experimentation with a version of the line search algorithm was done by Anand Ramasubramaniam in his Master's thesis [2000]. Constructive comments by the referees and by Tim Hopkins and Nicholas I. M. Gould are gratefully acknowledged.

REFERENCES

AL-BAALI, M. 1985. Decent property and global convergence of the Fletcher-Reeves method with in exact line search. *IMA J. Numer. Anal. 5*, 121–124.

AL-BAALI, M. AND FLETCHER, R. 1984. An efficient line search for nonlinear least squares. *J. Optim. Theory Appl. 48*, 359–377.

BONGARTZ, I., CONN, A. R., GOULD, N. I. M., AND TOINT, P. L. 1995. CUTE: Constrained and unconstrained testing environments. *ACM Trans. Math. Soft. 21*, 123–160.

COHEN, A. I. 1972. Rate of convergence of several conjugate gradient algorithms. *SIAM J. Numer. Anal. 9*, 248–259.

DAI, Y. H. AND LIAO, L. Z. 2001. New conjugate conditions and related nonlinear conjugate gradient methods. *Appl. Math. Optim. 43*, 87–101.

DAI, Y. H. AND YUAN, Y. 1999. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. Optim. 10*, 177–182.

DAI, Y. H. AND YUAN, Y. 2000. *Nonlinear Conjugate Gradient Methods*. Shang Hai Science and Technology, Beijing.

DAI, Y. H. AND YUAN, Y. 2001. An efficient hybrid conjugate gradient method for unconstrained optimization. *Ann. Oper. Res. 103*, 33–47.

DANIEL, J. W. 1967. The conjugate gradient method for linear and nonlinear operator equations. *SIAM J. Numer. Anal. 4*, 10–26.

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

FLETCHER, R. 1987. *Practical Methods of Optimization vol. 1: Unconstrained Optimization*. Wiley & Sons, New York.

FLETCHER, R. AND REEVES, C. 1964. Function minimization by conjugate gradients. *Comput. J. 7*, 149–154.

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

GOLDSTEIN, A. A. 1965. On steepest descent. *SIAM J. Control 3*, 147–151.

GOLUB, G. H. AND O'LEARY, D. P. 1989. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Rev. 31*, 50–100.

HAGER, W. W. 1988. *Applied Numerical Linear Algebra*. Prentice-Hall, Englewood Cliffs, N.J.

HAGER, W. W. 1989. A derivative-based bracketing scheme for univariate minimization and the conjugate gradient method. *Comput. Math. Appl. 18*, 779–795.

HAGER, W. W. AND ZHANG, H. 2004. CG_DESCENT user's guide. Tech. Rep., Dept. Math., Univ. Fla.

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

HAGER, W. W. AND ZHANG, H. 2006. A survey of nonlinear conjugate gradient methods. *Pacific J. Optim. 2*, 35–58.

HAN, J., LIU, G., SUN, D., AND YIN, H. 2001. Two fundamental convergence theorems for nonlinear conjugate gradient methods and their applications. *Acta Math. Appl. Sinica 17*, 38–46.

HAN, J. Y., LIU, G. H., AND YIN, H. X. 1997. Convergence of Perry and Shanno's memoryless quasi-Newton method for nonconvex optimization problems. *OR Trans. 1*, 22–28.

HESTENES, M. R. AND STIEFEL, E. L. 1952. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards 49*, 409–436.

HIRST, H. 1989. n-step quadratic convergence in the conjugate gradient method. Ph.D. thesis, Dept. Math., Penn. State Univ., State College, Penn.

LEMARÉCHAL, C. 1981. A view of line-searches. In *Optimization and Optimal Control*. vol. 30. Springer Verlag, Heidelberg, 59–79.

LIU, D. C. AND NOCEDAL, J. 1989. On the limited memory BFGS method for large scale optimization. *Math. Program. 45*, 503–528.

LIU, Y. AND STOREY, C. 1991. Efficient generalized conjugate gradient algorithms, part 1: Theory. *J. Optim. Theory Appl. 69*, 129–137.

MORÉ, J. J. AND SORENSEN, D. C. 1984. Newton's method. In *Studies in Numerical Analysis*, G. H. Golub, ed. Mathematical Association of America, Washington, D.C., 29–82.

MORÉ, J. J. AND THUENTE, D. J. 1994. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Soft. 20*, 286–307.

NOCEDAL, J. 1980. Updating quasi-Newton matrices with limited storage. *Math. Comp. 35*, 773–782.

PERRY, J. M. 1977. A class of conjugate gradient algorithms with a two step variable metric memory. Tech. Rep. 269, Center for Mathematical Studies in Economics and Management Science, Northwestern University.

POLAK, E. AND RIBIÈRE, G. 1969. Note sur la convergence de méthodes de directions conjuguées. *Rev. Fran caise Informat. Recherche Opérationnelle 3*, 35–43.

POLYAK, B. T. 1969. The conjugate gradient method in extremal problems. *USSR Comp. Math. Math. Phys. 9*, 94–112.

POWELL, M. J. D. 1977. Restart procedures for the conjugate gradient method. *Math. Program. 12*, 241–254.

POWELL, M. J. D. 1984. Nonconvex minimization calculations and the conjugate gradient method. In *Lecture Notes in Mathematics*. vol. 1066. Springer Verlag, Berlin, 122–141.

POWELL, M. J. D. 1986. Convergence properties of algorithms for nonlinear optimization. *SIAM Rev. 28*, 487–500.

RAMASUBRAMANIAM, A. 2000. Unconstrained optimization by a globally convergent high precision conjugate gradient method. M.S. thesis, Dept. Math., Univ. Florida.

SHANNO, D. F. 1978. On the convergence of a new conjugate gradient algorithm. *SIAM J. Numer. Anal. 15*, 1247–1257.

SHANNO, D. F. 1985. On the convergence of a new conjugate gradient algorithm. *Math. Program. 33*, 61–67.

SHANNO, D. F. AND PHUA, K. H. 1980. Remark on algorithm 500. *ACM Trans. Math. Soft. 6*, 618–622.

Wang, C., Han, J., and Wang, L. 2000. Global convergence of the Polak-Ribière and Hestenes-Stiefel conjugate gradient methods for the unconstrained nonlinear optimization. *OR Trans. 4*, 1–7.

Wolfe, P. 1969. Convergence conditions for ascent methods. *SIAM Rev. 11*, 226–235.

Wolfe, P. 1971. Convergence conditions for ascent methods II: Some corrections. *SIAM Rev. 13*, 185–188.

Received January 2004; revised March 2005; accepted June 2005