Journal of Computational Mathematics, Vol.22, No.4, 2004, 489–500.

A DIRECT SEARCH FRAME-BASED CONJUGATE GRADIENTS METHOD *

I.D. Coope C.J. Price
(*Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand*)

Abstract

A derivative-free frame-based conjugate gradients algorithm is presented. Convergence is shown for C^1 functions, and this is verified in numerical trials. The algorithm is tested on a variety of low dimensional problems, some of which are ill-conditioned, and is also tested on problems of high dimension. Numerical results show that the algorithm is effective on both classes of problems. The results are compared with those from a discrete quasi-Newton method, showing that the conjugate gradients algorithm is competitive. The algorithm exhibits the conjugate gradients speed-up on problems for which the Hessian at the solution has repeated or clustered eigenvalues. The algorithm is easily parallelizable.

Mathematics subject classification: 90C56, 90C30, 65K05.
Key words: Conjugate gradients, Derivative-free, Frame-based methods, Numerical results.

1. Introduction

The linear conjugate gradient was developed by Hestenes and Stiefel [10], and extended to the minimization of general functions by Fletcher and Reeves [6]. A description of conjugate gradients methods for minimising general functions can be found in [5, 6, 7, 8, 9, 12] and elsewhere. In this paper we consider the application of conjugate gradients techniques to unconstrained minimization of C^1 functions in a derivative-free context. A method is described which conforms to the frame-based template in [2], thereby guaranteeing convergence under standard conditions. The problem may be formally stated as

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

where a local, but not necessarily a global minimizer is sought. Here we restrict attention to objective functions f which are continuously differentiable, but do not assume that gradient information is available. Second order optimality conditions are not useable as second derivatives may not exist. Consequently stationary points are accepted as solutions. The algorithm forms an estimate of the gradient at each iterate, but does not rely on the accuracy of these estimates to guarantee convergence. These gradient estimates are used to form conjugate gradients search directions, and line searches are conducted along these directions. Hence the algorithm mimics a conjugate gradients method when it can (that is to say, when its gradient estimates happen to be accurate), which makes it more effective in practice. Convergence is guaranteed by the frame-based nature of the algorithm, not the fact that it mimics a conjugate gradients method. The theoretical convergence properties are unaffected by the accuracy of gradient estimates, although inaccurate estimates will, in general, degrade the numerical performance of the algorithm.

* Received August 26, 2002; final revised June 4, 2003.

The conjugate gradients method used is that of Polak and Ribière [12], and Polyak [13] (hereafter PRP). Limited numerical comparisons between frame based PRP and Fletcher–Reeves methods indicated that the former was more promising. This preference seems to be in accord with the case when exact gradients are available [7]. The automatic reset property [14] of the PRP method is also very desirable for higher dimensional problems.

The algorithm generates a sequence of iterates $\{x^{(k)}\}$, where k is the iteration number. At each iteration the function values at a set of points $\Phi^{(k)}$ called a frame are calculated. Frames are defined precisely in the following section, and a template for frame-based algorithms is given in Section 3. Loosely speaking, the points in the frame surround $x^{(k)}$. The gradient at $x^{(k)}$ is estimated using points from the frame $\Phi^{(k)}$. This gradient estimate allows a derivative-free conjugate search direction to be formed, as described in Section 4. A line search is conducted along this search direction, yielding the next iterate. The process is repeated until an adequate approximation to a stationary point is obtained. The choice of frames yields a second order gradient estimate at each iterate. The line search uses parabolic interpolation to locate an approximation to each line local minimum. On a quadratic these gradient estimates and approximations to line local minima are exact, and so the algorithm exactly minimizes convex quadratics in a finite time. A description of the line search is given in Section 5. Numerical results and concluding remarks are presented in Sections 6 and 7.

2. Frames

A frame Φ is defined by a frame centre x, a frame size $h > 0$, and a positive basis $\mathcal{V}_+$. A positive basis [4] (see also [18]) is a set of vectors $\mathcal{V}_+$ with the following two properties:

(i) every vector in R^n is a linear combination of the members of $\mathcal{V}_+$, where all coefficients of the linear combination are non-negative; and

(ii) no proper subset of $\mathcal{V}_+$ satisfies (i).

A frame $\Phi(x, h, \mathcal{V}_+)$ around x is a set of points of the form $\Phi(x, h, \mathcal{V}_+) = \{x + hv : v \in \mathcal{V}_+\}$. It is shown in [4] that positive bases (and hence also frames) have at least $n + 1$ and at most $2n$ members. Positive bases and frames containing $2n$ members are called maximal. Maximal positive bases [4] are of the form $\mathcal{V}_+ = \{v_1, v_2, \ldots, v_n, -v_1, -v_2, \ldots, -v_n\}$ where $v_1, \ldots, v_n$ are a basis for R^n. Herein we use $v_i = e_i$, where e_i is the i^{th} unit vector. This yields a frame around a frame centre $x^{(k)}$ of the form:

$$\Phi^{(k)} = \{x^{(k)} + h^{(k)} e_i : \forall i = 1, \ldots, n\} \cup \{x^{(k)} - h^{(k)} e_i : \forall i = 1, \ldots, n\}. \tag{1}$$

Each such maximal frame contains enough information to form second order estimates of the gradient at $x^{(k)}$, and also the diagonal entries of the Hessian at $x^{(k)}$. These second derivative estimates are used to scale the decision variables at each reset.

Frames which are called quasi-minimal are of particular interest. These frames have the property that no frame point is more than ϵ lower than the frame centre, where ϵ is a preselected non-negative constant. The convergence theory [2] shows that any method conforming to it will generate an infinite subsequence of quasi-minimal frames. The convergence theory also shows that, under mild conditions, the cluster points of this subsequence of quasi-minimal frame centres are stationary points of f.

At each iteration a positive constant $\epsilon^{(k)}$ is chosen. The frame $\Phi^{(k)}$ is called *quasi-minimal* if and only if

$$f\left(x^{(k)}\right) \leq f(x) + \epsilon^{(k)} \quad \forall x \in \Phi^{(k)}. \tag{2}$$

The constant $\epsilon^{(k)}$ is connected to the frame size via the following relation

$$\epsilon^{(k)} = N\left(h^{(k)}\right)^{\nu} \tag{3}$$

where $N > 0$ and $\nu > 1$ are constants. The restriction $\nu > 1$ ensures that $\epsilon = o(h)$ as $h \to 0$; a property which is vital to the convergence theory. Herein $\nu = 3/2$ and $N = 1$ were used.

3. A Convergent Template

In this section the template presented in [2] is described, and the relevant convergence theory is summarised. The choice of frame in equation (1) is much more restricted than in [2]. Corresponding simplifications have been made to the template and convergence theory.

The idea behind the template is either to generate a quasi-minimal frame or to locate a point of sufficient descent at each iteration. A point of sufficient descent at iteration k is any point x satisfying

$$f(x) < f^{(k)} - \epsilon^{(k)},$$

where $f^{(k)} \equiv f(x^{(k)})$. If the frame $\Phi^{(k)}$ around $x^{(k)}$ is completed then this frame is either quasi-minimal or it must contain a point of sufficient descent. Depending on which of these two outcomes occurs, h (and hence ϵ) is respectively decreased, or not decreased. The latter case may result in h being increased.

This ensures that an infinite subsequence of quasi-minimal frames occurs unless the sequence of function values is unbounded below. If the subsequence of quasi-minimal frames were finite, then h would be bounded away from zero. Equation (3) means ϵ would also have a strictly positive lower bound. Eventually each iteration would reduce f by at least this quantity, meaning that the sequence of function values must be unbounded below. Hence if the sequences $\{x^{(k)}\}$ and $\{f^{(k)}\}$ are bounded then the sequence of quasi-minimal iterates must be infinite, and it must also have cluster points. A final requirement of the method of adjusting h is that $h \to 0$ if it is reduced repeatedly without any increases occurring.

The stationarity of cluster points of the subsequence of quasi-minimal frame centres is a direct consequence of equation (3). For clarity, replace $\{x^{(k)}\}$ with a subsequence of itself which has a unique limit point x^* and consists entirely of quasi-minimal frame centres. Quasi-minimality of $\Phi^{(k)}$ implies

$$\frac{f\left(x^{(k)} \pm h^{(k)}e_i\right) - f\left(x^{(k)}\right)}{h^{(k)}} + \frac{\epsilon^{(k)}}{h^{(k)}} \geq 0 \qquad \forall\, i = 1, \ldots, n.$$

In the limit $k \to \infty$ equation (3) means that the second term vanishes, showing that the directional derivative of f in the direction $\pm e_i$ is non-negative at x^*. Since $\pm e_1, \ldots, \pm e_n$ is a positive basis, the only possibility is that x^* is a stationary point of f [2].

A simplified form of the frame-based template described in [2] is stated next. The simplifications are permitted because the conjugate gradients method uses the same positive basis for all iterations. The iteration counter i is used in place of k to highlight the fact that several iterations of the conjugate gradients method may be treated as one iteration of the template. This is discussed in detail later in this section.

Algorithm 1. 1. Initialize: Set $i = 1$. Choose $h^{(1)}$, $N > 0$, and $\nu > 1$.

2. Calculate $\epsilon^{(i)}$.

3. Execute any finite process which either chooses $x^{(i+1)}$ as a point of sufficient descent, or forms a quasi-minimal frame $\Phi^{(i)}$ around $x^{(i)}$.

4. If sufficient descent was obtained in step 3 then choose $h^{(i+1)} \geq h^{(i)}$, increment i and go to step 2.

5. Set $x^{(i+1)}$ as the lowest point found in step 3. Set $h^{(i+1)} < h^{(i)}$, increment i and go to step 2.

There is one point on which the conjugate gradient algorithm differs from the template given in Algorithm 1 above. If $\Phi^{(i)}$ is quasi-minimal, the template requires that $x^{(i+1)}$ be the lowest point found in that step. The conjugate gradients algorithm is not able to do this immediately because any such step along a frame direction destroys the conjugate directions property of the algorithm on convex quadratics. Instead the conjugate gradients method must wait until the next reset to make such a move. This can be accommodated in the template by regarding all iterations starting from the iteration in which a step along a frame direction should occur until the next reset as part of the finite process in step 3. Hence the counter i is similar, but not equivalent to k. The convergence properties of this template are stated in the following theorem.

Theorem 1. *If the following conditions are satisfied*

1. *the sequence of iterates is bounded;*

2. *f is continuously differentiable with a locally Lipschitz gradient; and*

3. *$h^{(k)} \to 0$ as $k \to \infty$*

then the subsequence of quasi-minimal frame centres is infinite, and all cluster points of this subsequence are stationary points of f.

Proof. The result is a direct consequence of Theorems 4.1 and 4.2 of [2].

This theorem does not specify how $h^{(k)} \to 0$ is to be achieved in the limit $k \to \infty$. Theorem 4.3 of [2] shows that the following approach to choosing h will ensure $h \to 0$ in the limit $k \to \infty$, subject to the conditions in Theorem 1. If a quasi-minimal frame is located, then h must be scaled by a constant factor $c_{\text{dech}} \in (0, 1)$. Otherwise h may be kept constant, or scaled up by a factor which is bounded below by 1, and bounded above by a second constant $c_{\text{inch}} > 1$. The reasoning behind this is as follows: if h is bounded away from zero ($h \geq h_0$ say), then an infinite number of sufficient descent steps must occur. Each of these steps must reduce f by at least $N(h_0)^\nu$. This contradicts assumptions 1 and 2 of Theorem 1.

4. The Conjugate Gradients Algorithm

The algorithm performs a fixed number of direct search conjugate gradient steps, and then resets. At each reset the algorithm uses estimates of the pure second derivatives $\partial^2 f/\partial x_i^2$, $i = 1, \ldots, n$, to scale the decision variables. After each reset the algorithm starts from the lowest known point, which may be either a frame point or a point found in the latest line search. The next iteration searches along the steepest descent direction from this point.

The basic steps of one iteration of the algorithm are as follows. An iteration in which a reset does not occur is described first, after which the case when a reset occurs is discussed. Firstly a frame $\Phi^{(k)}$ is constructed about the current iterate $x^{(k)}$. Using the function values at these frame points a second order estimate $g^{(k)}$ of the gradient at $x^{(k)}$ is formed. The i^{th} element of $g^{(k)}$ is $(f(x^{(k)} + h^{(k)}e_i) - f(x^{(k)} - h^{(k)}e_i))/2h^{(k)}$. The next search direction $p^{(k)}$ is then calculated using $g^{(k)}$, $g^{(k-1)}$, and $p^{(k-1)}$. An accurate line search is conducted along the line $x^{(k)} + \alpha p^{(k)}$, $\alpha \in R$, for the next iterate $x^{(k+1)}$. The frame size h is then updated. Specifically, if the frame $\Phi^{(k)}$ is quasi-minimal then $h^{(k)}$ is reduced, which, subject to a lower bound, yields $h^{(k+1)}$. Otherwise, if the distance between $x^{(k)}$ and $x^{(k+1)}$ is large compared with

$h^{(k)}$ then $h^{(k)}$ is increased, yielding $h^{(k+1)}$. The constant $\epsilon^{(k+1)}$ which defines quasi-minimality is then recalculated using equation (3). This completes an iteration.

In an iteration in which a reset occurs, the following changes are made. First, the diagonal matrix $D^{(k)}$ is formed where the i^{th} diagonal entry D_i of D is the estimate $(f(x^{(k)} + h^{(k)}e_i) + f(x^{(k)} - h^{(k)}e_i) - 2f^{(k)})/(h^{(k)})^2$ of $\partial^2 f/\partial x_i^2$. An iteration is completed as described above, and the reset is then performed. The reset involves setting the current iterate to be the lowest known point, and updating the scaling for each coordinate direction. The next iteration searches along the steepest descent direction at $x^{(k+1)}$.

Each line search looks along the line $x + \alpha hp/\|p\|$, $\alpha \in R$, for a local minimizer of f on that line. The line search returns a value $\alpha^{(k)}$ which approximates a local minimizer of the function $\psi^{(k)}(\alpha)$, where

$$\psi^{(k)}(\alpha) = f\left(x^{(k)} + \alpha h^{(k)} p^{(k)} / \|p\|\right).$$

The line search uses safeguarded parabolic interpolation. It is described in Section 5.

4.1. Generating the Search Direction

Each frame provides second order estimates of the gradient and the pure second derivatives. The latter are used to scale the coordinate directions in order to improve the conditioning of the problem. The algorithm begins with all scale factors H_i set equal to one. At each reset new scale factors are calculated using

$$H_i = 1 / \max\{D_i, \tau_{2\text{nd}}\} \quad \forall i = 1, \ldots, n.$$

Here $\tau_{2\text{nd}}$ is a small positive constant used to ensure that each scale factor H_i is positive. For the numerical results presented herein $\tau_{2\text{nd}} = 10^{-4}$ was used. The decision variables x_i are then scaled for all iterations until the next reset occurs using $z_i = x_i/\sqrt{H_i}$. This scaling ensures that $\partial^2 f/\partial z_i^2 = 1$ for every co-ordinate direction along which $\partial^2 f/\partial x_i^2$ exceeds $\tau_{2\text{nd}}$. The next search direction is calculated by applying the conjugate direction formula to the z variables. This yields the following formula

$$p^{(k+1)} = -Hg^{(k+1)} + \beta^{(k)} p^{(k)} \quad \text{where} \quad \beta^{(k)} = \max\left\{0, \frac{g^T H\left(g^{(k+1)} - g\right)}{g^T H g}\right\} \tag{4}$$

and where the (k) superscripts have been omitted from g. The matrix H is a diagonal matrix with diagonal entries $H_1, \ldots, H_n$. Here (4) uses the PRP update [12, 13], with Powell's modification [15] that negative values of β are replaced with zero. This modification is preferred for several reasons. It has been shown [16] that the unmodified PRP algorithm can cycle without ever approaching a solution point. Gilbert and Nocedal [7] show that Powell's modification gives convergence. Their numerical results show that these two methods are comparable in practice. Also, much of the theory behind conjugate gradients methods is developed for quadratic objective functions. On such functions the Fletcher–Reeves [6] and PRP formulas for β are equal and necessarily non-negative as the former is the ratio $\|g^{(k+1)}\|^2/\|g^{(k)}\|^2$. This suggests that any negative values for $\beta^{(k)}$ should be replaced with 0.

Numerical testing showed that the algorithm's performance was significantly improved by scaling of each variable at each reset, particularly on the ill-conditioned problems. Spacing resets $n + 3$ iterations apart rather than n was also shown to be more effective by numerical testing, and is justified by the fact that the PRP conjugate gradients method tends to reset automatically when progress is poor [5, 14].

A listing of the algorithm is now given. For clarity some (k) superscripts on g, h, and p have been omitted. The variable j counts the number of iterations until the next reset. Stopping conditions and the method of updating h are described in the following two subsections.

Algorithm 2. *Data* $x^{(1)}$, τ_{acc}

1. Initialize: Set $k = 1$, $j = n$, $h = \alpha^{(0)} = 1$, and $H = I$. Choose $N > 0$, $\tau_{\min} > 0$, $h_{\min} > 0$, and $\nu > 1$.

2. **while** (stopping conditions do not hold) **do**

   (a) Calculate the function values at the frame points, and form the gradient estimate $g^{(k)}$. If $j = 1$ then also form the pure second derivative estimate D.

   (b) Check stopping conditions.

   (c) Calculate the new search direction $p^{(k)}$ via equation (4).

   (d) Select the initial values ψ_0' and α_{init} of the line search using $\psi_0' = hp^T g/\|p\|$ and $\alpha_{\text{init}} = \alpha^{(k-1)}$. Using these initial values, find a local minimizer $\alpha^{(k)}$ of $\psi^{(k)}(\alpha)$.

   (e) **If** $j = 1$ **then** update H, set $x^{(k+1)}$ equal to the lowest known point, and set $j = n+3$, **else** decrement j and set $x^{(k+1)} = x^{(k)} + \alpha^{(k)} hp/\|p\|$.

   (f) Update $h^{(k)}$ to get $h^{(k+1)}$, and increment k.

   end.

The algorithm keeps a record of the lowest known point, and, at each reset, sets the current iterate equal to this best point. This is vital for convergence purposes because it allows a direct search using the frames to take over when the line searches are rendered useless through inaccurate gradient estimates.

4.2. Adjusting the Frame Size

The main intention of the method of selecting h is not to get accurate estimates of the gradients, but to get acceptable descent. This is a fundamental part of the frame-based nature of the method. It ensures convergence in exact arithmetic even if every gradient estimate is complete rubbish, chosen randomly, set to zero, or chosen as maliciously as possible. The only requirements imposed by the convergence theory on the line searches is that they are finite processes which do not accept ascent steps. This is much weaker than the conditions imposed by standard conjugate gradients theory (See e.g. [1]).

In essence, the updating strategy for h increases it if the algorithm is making good progress, and decreases it whenever a quasi-minimal frame is located. The size of h is only loosely linked to the accuracy of the gradient estimates. Inaccurate gradients tend to result in poor progress being made, which usually makes quasi-minimal frames more frequent, and results in a smaller h. For accurate gradients the opposite is true, and h tends to be increased more often. One could perhaps think that, for accurate gradients, one should retain the same h rather than increase it and risk a loss of gradient accuracy. However the convergence theory applies only to the subsequence of quasi-minimal frame centres, not the whole sequence of iterates. An algorithm which allows h to fluctuate around a range of values giving accurate gradients will produce more quasi-minimal frames than one which keeps h fixed. Moreover, keeping h larger increases the chance that the direct search implicit in the frames will also find better points from time to time, particularly in the earlier iterations.

The frame size is altered as follows. If the current frame is quasi-minimal then h is reduced by a factor of 4, or set to a lower bound $h_{\min}$, whichever is the greater. Otherwise, if the line search accepts a step significantly larger than h, then h is increased. Specifically, if $\alpha^{(k)} > 2 + 2\sqrt{n}$ then $h^{(k+1)} = 5h^{(k)}/2$ is used. The square root term ensures that a step of length h along each axis is not considered significantly larger than h. In this paper

$$h_{\min} = \max\left\{10^{-10}, 10^{-5}\tau_{\text{acc}}\right\}$$

was used for all numerical test runs.

4.3. Stopping Conditions

The stopping conditions are similar to those proposed for unconstrained optimization by Gill, Murray and Wright [8]. Specifically, the algorithm halts when

$$\|g\| \leq \min(1, (1+|f|)\tau_{\text{acc}}) \quad \text{and} \quad h < 5\max(\tau_{\text{acc}}, h_{\min}). \tag{5}$$

The accuracy parameter τ_{acc} was set to 10^{-5} on all test runs except where stated otherwise.

The condition on h in (5) ensures that the algorithm does not halt prematurely if a near-zero gradient estimate occurs with a large h. An example of this is the function

$$f(x) = x^2 + \left(1 + x - x^3\right) / \left(1 + x^2\right) \quad x \in R$$

with $x^{(k)} = 0$ and $h^{(k)} = 1$. Both frame points have $f = 3/2$ whereas $f(0) = 1$. This gives a minimal frame with the gradient estimate $g^{(k)} = 0$. The point $x = 0$ is clearly not stationary, and the condition on h prevents the algorithm from accepting it without further examination.

The algorithm also halted when it was unable to find a lower function value and the minimum frame size $h_{\min}$ had been reached. Specifically, the algorithm stops if all of the following three conditions are satisfied: $h \leq h_{\min}(1+\tau_{\min})$; $|\alpha| < \tau_{\min}$; and the current frame is quasi-minimal. Here $\tau_{\min}$ is a small positive constant ($\tau_{\min} = 10^{-8}$ was used herein).

5. The Line Search Algorithm

A safeguarded parabolic line search designed to locate a line local minimizer is used. For convenience the line search is described in terms of the one dimensional optimization problem:

$$\min_{\alpha \in R} \psi(\alpha) \quad \text{where} \quad \psi(\alpha) = f(x + \alpha h p/\|p\|).$$

The line search is given an estimate ψ_0' of the derivative of ψ with respect to α at $\alpha = 0$, and an initial step α_{init}. The former is calculated using $\psi_0' = hp^T g/\|p\|$. The latter is chosen as the final value $\alpha^{(k-1)}$ from the previous line search, with the convention that $\alpha^{(0)} = 1$.

The line search has three phases: The first generates three distinct points a, b, and c, sorted in increasing order, and then calculates ψ at these three points. The second phase searches for a bracket, where a bracket is three points a, b, and c for which $\psi(b) \leq \min(\psi(a), \psi(c))$ and $a < b < c$. The existence of a bracket ensures that a line local minimum lies in the interval $[a, c]$. Once a bracket has been found the third phase reduces the size of the bracketing interval $[a, c]$ until an acceptable approximation to the line local minimum has been found.

The first phase is provided with the function value $\psi(0)$, an estimate ψ_0' of the first derivative ψ' at $\alpha = 0$, and an initial step α_{init}. First α_1 is chosen as the closest point to α_{init} in the interval $[\kappa_1, \kappa_2]$, where $0 < \kappa_1 < \kappa_2$. Using this information, the algorithm calculates the minimizer α_2 of the quadratic interpolating ψ and ψ_0' at 0, and ψ at α_1. If this minimizer does not exist then $\alpha_2 = \alpha_1/2$ is used. If α_2 is within $\rho_{\min}$ of either 0 or α_1, then α_2 is chosen as $2\alpha_1$ if $\psi(\alpha_1) \leq \psi(0)$, or $-\alpha_1$ otherwise. Here $\rho_{\min}$ is a small positive constant such that two α values are regarded as indistinguishable if they differ by less than $\rho_{\min}$. The values α_0, α_1, and α_2 are then sorted into ascending order and relabelled a, b, and c.

If a, b, and c form a bracket then the aim of the second phase has already been achieved. If not, then the second phase extends the triple of points to the left if $\psi(a) < \psi(c)$, otherwise the triple is extended rightward. To extend leftwards, the minimizer α_q of the quadratic interpolating ψ at a, b, and c is first calculated, if it exists. Otherwise $\alpha_q = b$ is used. The point c

is then replaced by b, b is replaced by a, and a is replaced by

$$\max\left(a - 20(c-a), \min\left(a - 2(c-a), \alpha_q\right)\right).$$

This process of extending leftward is repeated until a bracket is found. The process of extending rightward is similar.

The third phase reduces the size of the bracket length $c - a$ until a sufficiently accurate estimate of a line local minimizer is obtained. Each reduction is carried out by first calculating the minimizer α_q of the quadratic interpolating ψ at a, b, and c. This minimizer exists unless $\psi(a) = \psi(b) = \psi(c)$, in which case α_q is chosen to bisect the longer of the intervals $[a, b]$ and $[b, c]$. In order to ensure that the bracket's length is reduced significantly, α_q is bounded away from a and c by reassigning it as follows:

$$\alpha_q \longleftarrow \max\left(a + \rho(c-a), \min\left(c - \rho(c-a), \alpha_q\right)\right)$$

where $\rho \in (0, \frac{1}{2})$. The new bracket is chosen as the first of the triples of points a, α_q, b, and b, α_q, c, which is actually a bracket. The algorithm always reduces the bracket at least twice, and terminates when $|\alpha_q - b| < \rho_{\text{acc}}\kappa_3/(\kappa_3 + |b|)$, where κ_3 is a positive constant. The factor scaling ρ_{acc} is used to ensure that ρ_{acc} is a relative measure of accuracy when the line search large step is large, and that it is an absolute measure otherwise. At least two reductions of the bracket are required to ensure the termination condition (usually) compares minimizers of consecutive quadratic fits.

The line search method also halts when the difference between two α values in the bracket falls below a small positive constant $\rho_{\min} = \min\{\rho_{\text{acc}}, \tau_{\min}\}$. When this happens the two α values are regarded as indistinguishable, and the bracket no longer identifies a region containing a local minimum of ψ. Finally, an upper limit on the number of function evaluations the line search may use is imposed. Herein a limit of 20 was used.

The following line search parameter values were used to generate all of the numerical results listed in this paper: $\rho = 0.1$, $\kappa_1 = 2$, $\kappa_2 = \kappa_3 = 100$, and $\rho_{\text{acc}} = 10^{-5}$.

6. Numerical Results and Discussion

Extensive numerical trials were performed using two categories of test problems. The first category consists of the test problems solved by Sun in [17], and the first 20 problems in [11]. Results for these problems are presented in Tables 1 and 2. A comparison between Algorithm 2 and Sun's quasi-Newton algorithm is presented in Table 2. The second category consists of three high dimensional problems which were solved in a selection of dimensions up to 1000. Results for these three problems appear in Table 3. The legend for these tables is as follows: n is the dimension of the problem; 'nf' and 'itns' are the number of function evaluations and iterations performed; and 'qmf' is the number of quasi-minimal frames generated. The quantities $f^\sharp$, $\|g^\sharp\|$, and $h^\sharp$ are respectively the function value, norm of the gradient estimate, and frame size at the final frame centre. Test runs with $\tau_{\text{acc}} < 10^{-5}$ are marked with a dagger (†). The symbol (§) marks runs for which the stopping condition on h delayed the termination of Algorithm 2. Problems marked with a double dagger (‡) were not solved to the required accuracy, but the algorithm obtained the optimal function value to six significant figures.

The measure of accuracy is on the *estimated* gradient g at the final iterate, not the exact gradient. Failure to obtain the required accuracy could be a result of an inaccurate g estimate rather than the final iterate not adequately approximating the solution point. This is clearly the case with Meyer's function, where the final function value agrees with that obtained by methods using gradients to at least six significant figures. For the other function (Osborne 2), lowering $h_{\min}$ to 10^{-18} allowed the algorithm to obtain the requested accuracy in 3088 function evaluations, with a final h value of 4e–11.

Table 1: Results for the low dimensional problems. Problems are listed in the order they appear in [11].

Problem	n	nf	qmf	itns	$f^\sharp$	$\|g^\sharp\|$	$h^\sharp$
Freudenstein & Roth	2	117	9	14	48.9843	8.9e–5	1e–5
Powell badly scaled	2	1984	76	124	2.365e–19	4.4e–8	1e–10
Jennrich & Sampson	2	214	13	26	124.362	1.5e–6	9e–8
Bard	3	228	15	21	8.21488e–3	1.5e–6	4e–8
Gauss §	3	88	9	9	1.1279e–8	3.1e–8	4e–6
Meyer ‡	3	5193	136	390	87.9459	197.309	1e–10
Gulf	3	585	27	48	3.539e–11	1.7e–6	2e–9
Box	3	259	18	22	9.148e–7	5.8e–6	6e–10
Kowalik and Osborne	4	409	21	29	3.07506e–4	7.1e–8	3e–10
Osborne 1	5	2286	56	147	5.47371e–5	6.6e–6	1e–8
Biggs exponential 6	6	523	20	30	5.65565e–3	5.2e–6	9e–9
Osborne 2 ‡	11	2443	44	88	0.0401377	1.1e–5	1e–10
Watson	6	1741	42	100	3.50122e–5	8.0e–6	4e–8
Penalty function I	4	401	19	27	2.27303e–5	6.3e–6	2e–9
—	10	1047	25	38	7.10066e–5	6.3e–6	1e–10
Modified Cragg †	4	3893	223	225	1.704e–24	9.1e–21	1e–10

There were three functions on which the algorithm located only a poor approximation to the actual solution. These were the modified Cragg function and Penalty Function I in dimensions 4 and 10. For the penalty function increasing the required accuracy to 10^{-7} enabled the algorithm to solve the problem accurately in both cases. The Modified Cragg function is highly ill-conditioned, with a zero Hessian at the solution. In order to obtain a final iterate accurate to 3 digits it was necessary to increase the required accuracy to 10^{-20}. These three results are marked with a '†' in Tables 1 and 2.

The use of the lower limit on h is justified on the grounds that if h gets too close to machine precision then the gradient estimates will become completely inaccurate. All problems were also solved using an $h_{\min}$ value of 10^{-18}, which is below machine precision. This changed the results for only a few functions: Powell Badly Scaled, Meyer's, Extended Powell in 64 dimensions, Osborne 2, Broyden's Tridiagonal function in 200–1000 dimensions, and Modified Cragg with a required accuracy of 10^{-20}. This change allowed the algorithm to solve the Osborne 2 problem to the required accuracy, but not Meyer's function. In most cases this change meant that the algorithm took slightly more function evaluations to solve each problem, and in a few cases less function evaluations. For the extended Powell, and Broyden tridiagonal functions there were no changes in the number of function evaluations needed to solve the problem, and the differences in the solutions found were inconsequential. The solution found for the Modified Cragg function with a required accuracy of 10^{-20} and $h_{\min} = 10^{-10}$ was much more accurate than that found with $h_{\min} = 10^{-18}$. This was because $h_{\min} = 10^{-18}$ permitted h to fall below machine precision. Once this occurs all frame points are identical in finite precision arithmetic. Such a frame yields an estimate of 0 for the gradient, which immediately causes the algorithm to terminate.

The maximum value on the final h in (5) together with the facts that $h^{(0)} = 1$ and each $h^{(k+1)} \geq h^{(k)}/4$ means that the algorithm must perform at least *nine* iterations before terminating. This prevented the algorithm from terminating more quickly on a number of problems: Gauss, Hilbert, Tridiagonal in 10 dimensions, and both Powell and Variably Dimensioned for

Table 2: A comparison between our algorithm and Sun's [17].

		Algorithm 2			Sun		
Problem	n	nf	itns	$f^\sharp$	nf	itns	$f^\sharp$
Beale	2	96	12	1.774e–12	81	14	3.39e–11
Brown badly scaled	2	161	17	2.468e–23	58	10	5.75e–15
Brown and Dennis	4	244	20	85822.2	209	18	85822.2
Broyden Tridiagonal	10	485	23	1.001e–12	845	39	1.58e–12
Modified Cragg	4	214	14	1.901e–7	455	49	7.26e–11
Dixon	10	1869	73	4.261e–17	971	45	3.53e–11
Extended Powell	4	388	29	9.509e–9	387	41	5.43e–11
—	32	2496	36	4.817e–9	3062	46	5.88e–11
—	64	6541	49	1.903e–9	6327	48	6.80e–11
Helical valley	3	277	27	2.448e–16	314	42	4.08e–12
Hilbert	4	118 §	10	2.085e–32	47	4	5.98e–12
Penalty function I	4	747 †	47	2.24998e–5	653	70	2.24998e–5
—	10	1568 †	55	7.08765e–5	4231	189	7.08765e–5
Powell	20	445 §	10	6.409e–31	1480	35	6.99e–11
—	50	1045 §	10	0	819	7	1.83e–14
Rosenbrock	2	300	34	5.234e–11	124	22	1.48e–11
Tridiagonal	10	414 §	17	2.640e–29	255	11	1.88e–12
—	50	5523	53	5.727e–14	5053	49	6.47e–19
Trigonometric	5	372	23	2.160e–9	355	31	8.45e–12
Variably Dimensioned	20	445 §	10	2.312e–29	896	16	2.10e–13
—	50	1045 §	10	9.785e–28	1871	15	2.10e–13
Woods	4	496	39	2.234e–13	354	37	2.66e–12

all values of n. These problems are identified with the '§' symbol in the relevant tables. One could immediately set h below $5\tau_{\text{acc}}$ once the required accuracy in $\|g\|$ is achieved, and doing so enabled the algorithm to solve these problems using less than half the number of function evaluations, on average. However, not reducing h rapidly makes the algorithm more robust, and this was considered more important by the authors.

Table 2 lists results for Algorithm 2 and for Sun's discrete quasi-Newton algorithm [17]. A direct comparison shows our algorithm is significantly faster on eight of the 22 problems, and nearly equal on one. A less crude comparison is made by dividing each problem's function count by the smaller of the function counts for both methods. These normalized function counts are then totalled for each method, yielding totals of 29.6 for Algorithm 2, and 30.06 for Sun's method. Together these comparisons show that our method is often somewhat slower, but occasionally *much* faster than Sun's. The stopping condition used by Sun differs from ours: Sun's algorithm halted when a function value within 10^{-10} of the optimal function value was encountered, whereas we do not assume knowledge of the optimal function value. Lack of knowledge of f^*, and the upper bound on the final h in (5) mean that Algorithm 2 sometimes performs several iterations after the solution has been accurately found.

The algorithm was also tested on three high dimensional problems: the extended Rosenbrock's function, the Variably Dimensioned function and Broyden's Tridiagonal function. These three problems were solved in 200 to 1000 dimensions. For each of these three problems, the Hessian at the solution has either clustered, or only two distinct eigenvalues. Hence these problems test the algorithm's ability to obtain the speed-up expected of a conjugate gradients

Table 3: Results for three high dimensional problems.

Problem	n	nf	qmf	itns	$f^\sharp$	$\|g^\sharp\|$	$h^\sharp$
Extended Rosenbrock	200	8142	12	20	1.531e–12	4.9e–6	2e–6
—	400	21775	17	27	1.549e–17	4.0e–8	4e–8
—	600	26542	14	22	1.104e–12	9.6e–7	4e–7
—	800	40174	15	25	4.025e–14	1.9e–6	6e–7
—	1000	48183	14	24	1.694e–15	4.9e–6	6e–6
Broyden Tridiagonal	200	10519	19	26	8.433e–13	9.2e–6	1e–10
—	400	20917	19	26	1.058e–12	1.0e–5	1e–10
—	600	33729	20	28	1.033e–12	9.4e–6	1e–10
—	800	44928	19	28	4.767e–13	6.7e–6	1e–10
—	1000	58130	19	29	5.928e–13	7.5e–6	1e–10
Variably Dimensioned §	200	4045	9	10	4.819e–27	2.2e–10	4e–6
— §	400	8045	9	10	5.164e–27	4.9e–10	4e–6
— §	600	12045	9	10	1.841e–23	7.3e–8	4e–6
— §	800	16045	9	10	3.841e–23	1.6e–7	4e–6
— §	1000	20045	9	10	2.415e–22	5.8e–7	4e–6

method on a problem when the Hessian at the solution has repeated or clustered eigenvalues. The numerical results clearly show that this speed-up was obtained on all three problems.

The parameter values used to generate all numerical results were $N = 1$, $\nu = 1.5$, $\tau_{\min} = 10^{-8}$, and $\tau_{\text{acc}} = 10^{-5}$, except that other values of τ_{acc} were used on runs marked with the '†' symbol. Line search parameter values are listed at the end of the previous section.

Results were also generated using the formula for β of Dai and Yuan [3] for the 38 problems listed in Tables 1 and 2. The PRP formula was faster on 24 problems, and equally fast on two. The Dai-Yuan formula was faster on the remaining 12. Comparing these two methods in the same way comparisons were made for Sun's method yields scores of 40.2 for PRP and 46.4 for Dai-Yuan. Hence PRP is somewhat faster. Nevertheless, the Dai-Yuan method was robust and performed well.

7. Conclusion

A direct search conjugate gradients algorithm employing a safeguarded parabolic line search has been presented. A frame-based approach conforming to the template described in [2] is used, which shows that the method is provably convergent on continuously differentiable functions under mild conditions. The algorithm uses Powell's modification of the PRP formula, with resets. The resets are essential for the convergence theory because, at each reset, the algorithm may step to the lowest known point without interfering with the conjugate gradient nature of the method. The frames provide second order estimates of gradients, and, at each reset, estimates of the pure second order derivatives. These second order derivatives are used to scale each decision variable at each reset in order to improve the conditioning of the problem.

When the problem's dimension is large most function evaluations are performed at frame points, and only a few in the line searches. The function values at the frame points can be calculated in parallel, which would dramatically reduce computation times.

Extensive numerical testing shows that the algorithm is effective on a wide variety of problems, including ill-conditioned problems. The results for the three high dimensional problems show that the algorithm obtains the usual 'conjugate gradients' speed-up for problems with clustered or repeated eigenvalues of the Hessian at the solution. The comparison with the nu-

merical results of Sun's discrete quasi-Newton algorithm shows that Algorithm 2 is efficient as well as theoretically sound.

This paper shows that convergent direct search conjugate gradients methods are effective in practice. It also confirms that many effective direct search frame-based methods exist.

References

[1] M. Al Baali, Descent property and global convergence of the Fletcher–Reeves method with inexact line search, *IMA Journal of Numerical Analysis*, **5** (1985), 121–124.

[2] I. D. Coope and C. J. Price, Frame based methods for unconstrained optimization, *J. Optimization Theory Applic.*, **107** (2000), 261–274.

[3] Y. H. Dai and Y. Yuan, A nonlinear conjugate gradient method with a strong global convergence property, *SIAM Journal on Optimization*, **10** (1999), 177–182.

[4] C. Davis, Theory of positive linear dependence, *American Journal of Mathematics*, **76** (1954), 733–746.

[5] R. Fletcher, Practical methods of optimization, 2nd edition, Wiley, 1987.

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

[7] J. C. Gilbert and J. Nocedal, Global convergence of conjugate gradients methods for optimization, *SIAM Journal on Optimization*, **2**:1 (1992), 21–42.

[8] P. E. Gill, W. Murray, and M. H. Wright, Practical Optimization, Academic Press, 1981.

[9] L. Grippo and S. Lucidi, A globally convergent version of the Polak–Ribière conjugate gradient method, *Math. Prog.*, **78** (1997), 375–391.

[10] M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards*, **49** (1952), 409–436.

[11] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Software*, **7**:1 (1981), 17–41.

[12] E. Polak, Computational Methods in Optimization: A Unified Approach, Academic Press, 1971.

[13] Polyak, B. T., The conjugate gradient method in extremum problems, *USSR Comp. Math. Math. Phys.,* **9** (1969), 94–112.

[14] Powell, M. J. D., Restart procedures for the conjugate gradient method, *Math. Prog.,* **12** (1977), 251–254.

[15] M. J. D. Powell, Convergence properties of algorithms for nonlinear optimization, *SIAM Review*, **28** (1986), 487–500.

[16] M. J. D. Powell, Nonconvex minimization and the conjugate gradient method, Lecture Notes in Mathematics, 1066, Springer-Verlag, Berlin 1984, 122–141.

[17] L.-P. Sun, A quasi-Newton algorithm without calculating derivatives for unconstrained minimization, *J. Comp. Math.*, **12**:4 (1994), 380–386.

[18] W.-C. Yu, Positive basis and a class of direct search techniques, *Scientia Sinica (Zhongguo Kexue)*, Special Issue 1 on Mathematics (1979), 53–68.