

A DIRECT SEARCH CONJUGATE DIRECTIONS  
ALGORITHM FOR UNCONSTRAINED MINIMIZATION

I. D. Coope and C. J. Price

*Department of Mathematics & Statistics,  
University of Canterbury,  
Private Bag 4800, Christchurch, New Zealand.*

**Report Number:** 188

November 1999

**Keywords:** derivative free, grid based optimization, positive basis, multidirectional search, conjugate directions



# A Direct Search Conjugate Directions Algorithm for Unconstrained Minimization.

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

## Abstract

A direct search algorithm for unconstrained minimization of smooth functions is described. The algorithm minimizes the function over a sequence of successively finer grids. Each grid is defined by a set of basis vectors. From time to time these basis vectors are updated to include available second derivative information by making some basis vectors mutually conjugate. Convergence to one or more stationary points is shown, and the finite termination property of conjugate direction methods on strictly convex quadratics is retained. Numerical results show that the algorithm is effective on a variety of problems including ill-conditioned problems.

**Key Words:** derivative free, grid based optimization, positive basis, multidirectional search, conjugate directions.

## 1 Introduction

There has been much recent interest in derivative free methods for unconstrained optimization [1, 6, 9]. It may be argued that methods such as discrete quasi-Newton methods which approximate derivatives with finite differences are derivative free, however these methods have not been proven to be convergent. In this paper interest is directed at algorithms for which convergence proofs are known.

A variety of provably convergent methods have been described, including ones based on line searches, trust regions, and on grids. The algorithm presented here is in the last category, and uses the convergence theory developed in [2]. The algorithm does not require  $C^2$  continuity, but can exploit it by using conjugate directions to form the grids. From time to time gradient estimates are available as a byproduct, and these are used to approximate a quasi-Newton step on each such occasion. These quasi-Newton steps are not needed to establish convergence.

A minimizer of a given  $C^1$  objective function  $f : R^n \rightarrow R$  is sought, where the gradient  $\nabla f$  of  $f$  is locally Lipschitz. The algorithm does not make explicit use of  $\nabla f$ , but minimizes

$f$  by examining it on a sequence  $\{\mathcal{G}^{(m)}\}_{m=1}^{\infty}$  of successively finer grids. Each grid  $\mathcal{G}^{(m)}$  is defined by a set of  $n$  linearly independent basis vectors  $\mathcal{V}^{(m)} = \{v_i^{(m)} : i \in 1, \dots, n\}$ . The points on the grid  $\mathcal{G}^{(m)}$  are

$$\mathcal{G}^{(m)} = \left\{ x \in \mathbb{R}^n : x = x_o^{(m)} + h^{(m)} \sum_{i=1}^n \eta_i v_i^{(m)} \text{ where } \eta_i \text{ is integer } \forall i \in 1, \dots, n \right\}$$

The parameter  $h^{(m)}$  is referred to as the mesh size, and is adjusted as  $m$  is increased in order to ensure that the meshes become finer in a manner needed to establish convergence. The point  $x_o^{(m)}$  is included to allow each grid to have a different origin to its predecessor. The grid points are referenced via  $\eta$  rather than  $x$  to avoid the accumulation of round off errors from repeated movements on  $\mathcal{G}^{(m)}$ .

The algorithm seeks to minimize  $f$  over each grid  $\mathcal{G}^{(m)}$ , where a minimiser of  $f$  over a grid is defined as follows:

**Definition 1** GRID LOCAL MINIMUM *A point  $x$  on the grid  $\mathcal{G}^{(m)}$  is defined as a grid local minimum if and only if*

$$f(x + v) \geq f(x) \quad \text{and} \quad f(x - v) \geq f(x) \quad \forall v \in \mathcal{V}^{(m)}$$

□

This definition is motivated by the observation that if

$$(\nabla f(x))^T v \geq 0 \quad \text{and} \quad (\nabla f(x))^T (-v) \geq 0 \quad \forall v \in \mathcal{V}^{(m)} \quad (1)$$

then  $x$  is a stationary point of  $f$  (see e.g. [2]). The conditions which define a grid local minimum are a finite difference approximation to this. In each main iteration of the algorithm, a grid  $\mathcal{G}^{(m)}$  is selected using previous information, and a grid local minimiser of  $f$  over  $\mathcal{G}^{(m)}$  is sought through a series of line searches along the directions in  $\mathcal{V}^{(m)}$ . In practice, a finite number of alterations to the grid are permitted during the line searches. An outline of the algorithm's form is as follows:

#### THE ALGORITHM OUTLINE

- (i) Initialize all variables.
- (ii) Execute any finite process.
- (iii) Search cyclically along the directions  $v_1, \dots, v_n$  for grid points which are lower than the current iterate. When a grid local minimum is found, proceed to the next step.
- (iv) Execute any finite process.
- (v) Form a new grid with its origin at the current lowest iterate. If stopping criteria are not satisfied, go to step (ii).

It is shown in [2] that, under mild conditions, an algorithm with this framework generates a sequence of grid local minima which converge to one or more stationary points of  $f$ . For convenience this theorem is restated here, with a slight specialization to reflect the definition of a grid local minimum used herein.

**Theorem 1** *Given*

- (a) *The sequence of iterates  $\{x^{(k)}\}_{k=1}^{\infty}$  is bounded;*
- (b)  *$f(x)$  is continuously differentiable, and its gradient  $\nabla f(x)$  is Lipschitz in any bounded region of  $R^n$ ;*
- (c) *There exist positive constants  $K$  and  $\tau_{det}$  such that  $|\det(v_1^{(m)} \dots v_n^{(m)})| \geq \tau_{det}$  and  $\|v_i^{(m)}\| \leq K$  for all  $m$  and  $i$ ; and*
- (d)  *$h^{(m)} \rightarrow 0$  as  $m \rightarrow \infty$ ;*

*then each cluster point  $\hat{x}^{(\infty)}$  of the subsequence  $\{\hat{x}^{(m)}\} \subseteq \{x^{(k)}\}$  is a stationary point of  $f(x)$ . Here each  $\hat{x}^{(m)}$  is the grid local minimum of  $\mathcal{G}^{(m)}$  found by the algorithm.*

**Proof:** See [2]. □

## 2 General Description of the Algorithm

The members of  $\mathcal{V}^{(m)}$  are chosen to maintain any known second derivative information in the form of mutually conjugate directions. The set of directions  $\mathcal{V}^{(m)}$  is divided into two subsets:  $\mathcal{V}_c^{(m)} = \{v_1^{(m)}, \dots, v_c^{(m)}\}$  and  $\mathcal{V}_{nc}^{(m)} = \{v_{c+1}^{(m)}, \dots, v_n^{(m)}\}$ . The members of  $\mathcal{V}_c$  are regarded as mutually conjugate, whereas the members of  $\mathcal{V}_{nc}$  are not. These basis vectors form the columns of the matrix  $V^{(m)} = [v_1^{(m)} \dots v_n^{(m)}]$ . For convenience the matrices  $V_c^{(m)}$  and  $V_{nc}^{(m)}$  will be used to refer to the first  $c$  and the last  $n - c$  columns of  $V^{(m)}$  respectively.

The algorithm repeatedly conducts line searches along the directions in  $\mathcal{V}^{(m)}$  until a grid local minimum is found. Between grid local minima, existing members of  $\mathcal{V}_c^{(m)}$  are not changed during these line searches. Each member of  $\mathcal{V}_{nc}^{(m)}$  can be changed once between grid local minima. This occurs when  $v \in \mathcal{V}_{nc}^{(m)}$  is removed from  $\mathcal{V}_{nc}^{(m)}$ , and replaced by a new conjugate direction which is then included in the set  $\mathcal{V}_c^{(m)}$ . These new conjugate directions are generated using the parallel subspace theorem (see eg. [3, 7, 8]). This process continues until a grid local minimum is found. The directions in  $\mathcal{V}_c^{(m)}$  are then scaled so that they have unit estimated curvature along them. This ensures that, when  $c = n$ ,  $V_c V_c^T$  is the inverse Hessian on a strictly convex quadratic.

Each new conjugate direction changes the grid  $\mathcal{G}^{(m)}$ . Each such grid alteration removes a vector from  $\mathcal{V}_{nc}$ , hence only a finite number of such alterations can be made without locating a grid local minimum. These alterations are permitted as part of the finite process in step (ii) of the algorithm outline.

At each grid local minimum, if less than a full set of conjugate directions is known, then these are retained. Otherwise the members of  $\mathcal{V}^{(m)}$  are re-ordered, the conjugate directions are no longer regarded as such, and the process begins again with  $c = 1$ .

At each grid local minimizer, a second order estimate  $\hat{g}_v^{(m)}$  of  $V^T \nabla f$  is obtained. On noting  $VV^T$  approximates the inverse Hessian, the Newton step  $p = -(\nabla^2 f)^{-1} \nabla f$  can be estimated. The algorithm conducts a brief search along  $p$  for a lower point before selecting the next grid. This search forms part of the finite process in step (iv).

## 2.1 The Line and Ray Searches

The form of the algorithm requires that a search from an iterate  $x$  along  $v_i$  may be abandoned only after  $f$  has been calculated at the points  $x+v_i$  and  $x-v_i$ . Hence if the algorithm searches along all  $n$  directions  $v_1, \dots, v_n$  from  $x$  without finding a point lower than  $x$ , then  $x$  is a grid local minimum. If a lower point than  $x$  is located, then the algorithm searches further along that direction. More precisely, if  $f(x+v_i) < f(x)$  then a ray search along the ray  $x + \alpha v_i$ ,  $\alpha > 0$  is performed; otherwise if  $f(x-v_i) < f(x)$  a ray search along the ray  $x - \alpha v_i$ ,  $\alpha > 0$  is performed; otherwise the line search is terminated unsuccessfully.

Each ray search from  $x$  along  $v_o$  calculates  $f(x + \alpha v_o)$  at successively larger integer values of  $\alpha$  as long as a decreasing sequence of function values is obtained. When the last value is not lower than the second to last value, then the ray search is terminated, and the penultimate  $\alpha$  value determines the new iterate. The first two values are  $\alpha = 1$  and  $\alpha = 2$ , unless  $v_o = -v_i$ , in which case the first and second values are  $\alpha = -1$  and  $\alpha = 1$ . Each subsequent  $\alpha$  value is calculated using the formula  $\alpha = \max(\alpha + 1, \min(8\alpha, \lfloor \alpha_q + 0.5 \rfloor))$ . Here  $\lfloor \cdot \rfloor$  denotes the floor function, and  $\alpha_q$  is defined as the minimizer of the one dimensional quadratic interpolating the last three points on the line  $x + \alpha v_o$  at which  $f$  was calculated. If the interpolating quadratic is not strictly convex, then  $\alpha_q = 8\alpha$  is used.

## 3 The Main Algorithm

The basic structure of the algorithm is as follows

### THE MAIN ALGORITHM

1. Initialize  $m = k = c = 1$ ,  $i = 0$ , starting point  $x^{(0)}$ . Set  $x_b = \text{'unknown'}$ ,  $h^{(0)} = \infty$ ,  $h^{(1)} = 1$ , and  $V^{(1)} = I_n$ .
2. (a) Set  $i = i + 1$ . If  $i > n$ , set  $i = 1$ . If  $i = 1$  set  $x_{\text{old}} = x^{(k)}$ .  
 (b) execute a line search along the direction  $v_i^{(m)}$  from  $x^{(k)}$ .  
 (c) if  $i = c$ ,  $c < n$ , and  $x_b \neq \text{'unknown'}$ , then augment the set of conjugate directions as described in section 3.2.  
 (d) if a grid local minimum has been found go to step 3, otherwise alter  $h$  as specified in section 3.1.

- (e) if  $i = n$  do a ray search along  $x^{(k)} + \alpha(x^{(k)} - x_{\text{old}})$ ,  $\alpha > 0$ . Go to step 2(a).
3. Calculate  $\hat{g}_v^{(m)}$  and scale each member of  $\mathcal{V}_c$  so the estimated curvature along each direction is unity.
  4. Perform a 2 point line search along the quasi-Newton direction.
  5. If  $f(x_e) < f(x^{(k+1)})$ , then set  $x^{(k+1)} = x_e$ .
  6. Choose  $h^{(m+1)} = h^{(m)}/s_r$  and update  $s_r$ .
  7. If  $c \geq n$  set  $c = 1$ , and  $x_b = \text{'unknown'}$ . Set  $v_1^{(m+1)} = v_n^{(m)}$ , set  $v_i^{(m+1)} = v_{i-1}^{(m)}$  for all  $i = 2, \dots, n$ . Orthogonalize  $V$ .
  8. Set  $i = 0$ , increment  $m$ , and go to step 2.

Here  $i$  is the index of the direction being used in the line search.

The quantity  $\hat{g}_v^{(m)} \approx V^T \nabla f(\hat{x}^{(m)})$  is the estimated gradient of  $f(x + h^{(m)}V\eta)$  with respect to  $h\eta$ . At each grid local minimiser  $\hat{x}^{(m)}$ , the function value is known at each of the points  $\hat{x}^{(m)} \pm h^{(m)}v_i^{(m)}$ ,  $i = 1, \dots, n$ , and so central difference estimates along each  $v_i^{(m)}$  directly yield each element of  $\hat{g}_v^{(m)}$ .

In step 7 the matrix  $V$  is orthogonalized by post-multiplying it by an orthogonal matrix  $Q$ , where  $Q$  is chosen so that  $Q^T V^T V Q$  is a diagonal matrix. Orthogonalizing  $V$  in this way leaves the estimate  $VV^T$  of the inverse Hessian unaltered.

### 3.1 Choosing the mesh size

Each time a new grid is selected in step 6,  $h^{(m)}$  is divided by a scale down factor  $s_r$ , and  $s_r$  is then updated via the following process: if the number of line searches on the previous grid exceeds  $4n + n^2/2$  then  $s_r$  is reduced according to the formula

$$s_r = \max(1 + [s_r - 1]/4, s_{\min})$$

Otherwise, if the number of line searches on the previous grid is less than  $2n$  then  $s_r$  is increased using the formula:

$$s_r = \min(1 + 2(s_r - 1), s_{\max})$$

Here  $s_{\max} \geq s_{\min} \geq 1$  is required. The values  $s_{\min} = 1.01$  and  $s_{\max} = 8$  were used to generate the numerical results presented herein. The reason for this adaptive strategy for reducing  $h$  is to allow grids to become fine quickly when grid local minima are being found quickly, but to avoid grids that are too fine. In the latter event, if the grid is poorly oriented then many line searches may be made before a grid local minimum is found, and until a grid local minimum is found there is only limited scope for re-orienting the grid. The ray search in step 2(e) is also used to speed up the location of a grid local minimum on each grid.

For the same reason, every time  $n^2 + 8n$  consecutive line searches are executed without leaving step 2 the algorithm attempts to increase  $h$  at the end of step 2(d) according to the formula

$$h^{(m)} = \min \left( 2h^{(m)}, h^{(m-1)} / s_{\min} \right)$$

The use of  $h^{(0)} = \infty$  allows the algorithm to scale the initial grid up as much as is necessary to obtain a grid local minimum. These alterations are part of the finite process in step (ii) of the algorithm outline.

### 3.2 Generating the Set of Conjugate Directions

When  $f$  is a strictly convex quadratic, the searches along the directions in  $\mathcal{V}_c^{(m)}$  allow the minimizer  $x_b$  of  $f$  over the manifold  $\mathcal{M}$  to be calculated, where  $\mathcal{M} = \{x_b + V_c \zeta : \exists \zeta \in R^c\}$ . Provided a non-zero step occurs in the following  $n - c$  line searches along the directions in  $\mathcal{V}_{nc}^{(m)}$ , the sequence of iterates is translated off  $\mathcal{M}$ . The next searches along the directions in  $\mathcal{V}_c^{(m)}$  then allow the minimizer  $x_e$  of  $f$  on a manifold parallel to  $\mathcal{M}$  to be calculated. The direction  $x_e - x_b$  is conjugate to all members of  $\mathcal{V}_c^{(m)}$  (see eg. [7, 3, 8]).

Using  $h^{(m)} V^{(m)} \eta_{\text{new}} = x_e - x_b$ , the new conjugate direction  $x_e - x_b$  replaces the direction  $v_j$  in  $\mathcal{V}_{nc}^{(m)}$  for which the absolute value of the  $j^{\text{th}}$  component  $(\eta_{\text{new}})_j$  of  $\eta_{\text{new}}$  is maximal. The order of the remaining members of  $\mathcal{V}_{nc}^{(m)}$  is retained, the new conjugate direction is transferred from  $\mathcal{V}_{nc}^{(m)}$  to  $\mathcal{V}_c^{(m)}$ , and  $c$  is incremented.

If  $(\eta_{\text{new}})_j = 0$  for each  $j = c + 1, \dots, n$  then no displacement off the manifold  $\mathcal{M}$  has occurred, in which case the update is abandoned, and  $x_b$  is set to  $x_e$ . If the update is successful, then  $x_b$  is reset to ‘unknown.’

The ability to calculate the location of  $x_b$  stems from the fact that each line search provides function values at three or more points along the line in question. This allows the step to that line’s exact minimizer to be calculated for a strictly convex quadratic, by minimizing the one dimensional quadratic interpolating the last three points at which  $f$  was calculated on the line. The form of the line search guarantees this interpolating quadratic is strictly convex except when all three interpolated function values are equal. In the latter case the middle interpolated point is taken as the line’s minimiser. The contiguity of the searches along the members of  $\mathcal{V}_c$ , and conjugacy means that the sum of these steps to each line’s minimiser is the step to the minimiser  $x_b$ .

It can be shown that each update to  $V$  is via either by scaling of columns, or post-multiplication by a rank 1 matrix. Hence the determinant  $|\det(V)|$  in condition (c) of theorem 1 can be updated from iteration to iteration.

### 3.3 Scaling the members of $\mathcal{V}_c$

At each grid local minimum, the directions in  $\mathcal{V}_c^{(m)}$  are scaled to incorporate curvature information from the line searches along elements in  $\mathcal{V}_c^{(m)}$ . Let the estimate of the second derivative of  $f$  at  $\hat{x}^{(m)}$  along the direction  $v_i^{(m)}$  be  $H_i^{(m)}$ . Then

$$v_i^{(m+1)} = v_i^{(m)} \left[ \max \left( \epsilon, H_i^{(m)} \right) \right]^{-\frac{1}{2}} \quad \forall i = 1, \dots, c \quad (2)$$

so that the estimate of the second derivative of  $f$  at  $\hat{x}^{(m)}$  along each new direction  $v_i^{(m+1)}$  is 1, for  $i = 1, \dots, c$ . Here  $\epsilon$  is a small positive constant ( $10^{-8}$ ) used to avoid divide by zero problems. Although the form of the line search means that  $H_i < 0$  is impossible,  $H_i = 0$  can occur when  $f(x) = f(x + v) = f(x - v)$ .

The scaling of  $v_i^{(m+1)}$  in (2) may result in the violation of the bound  $\|v\| \leq K$  in condition (c) of theorem 1, in which case  $v_i^{(m+1)}$  is scaled so that  $\|v_i^{(m+1)}\| = K$ .

### 3.4 Stopping Conditions

The numerical results presented herein were generated using the simple test

$$\|\hat{g}_v^{(m)}\|_2 \leq \tau_{\text{acc}} \quad (3)$$

where the stopping tolerance  $\tau_{\text{acc}}$  was set at  $10^{-5}$ . The use of  $g_v$  in (3) is preferred because, given  $VV^T \approx G_*^{-1}$ ,

$$\|\hat{g}_v\|_2^2 \approx g^T G_*^{-1} g \approx (\hat{x} - x_*)^T G_* (\hat{x} - x_*)$$

where the Taylor series approximation  $g(x) = G_*(x - x_*)$  has been used, and where  $G_* = \nabla^2 f(x^*)$ . Clearly, (3) provides an estimate of the difference between the least known and optimal values of  $f$ .

In addition to (3), the algorithm halted whenever  $h$  fell below  $0.01\tau_{\text{acc}}$ . Such a limit is needed because, if  $h$  were allowed to become too small then integer increments to  $\eta$  may produce no change to  $x + hV\eta$  in finite precision arithmetic.

More sophisticated tests [4] may be applied to the sequence of grid local minima, but the ‘infrequent’ nature of this sequence reduces the value of such tests.

## 4 Exact Termination on a Quadratic

It has been shown in theorem 1 that the subsequence of grid local minimizers converges to a stationary point. It is now shown that the algorithm possesses the property of finite termination on strictly convex quadratics.

**Theorem 2** *Let  $f$  be a strictly convex quadratic of the form*

$$f(x) = \frac{1}{2}x^T Gx + a^T x \quad (4)$$

*then the algorithm finds the exact minimiser  $x^*$  of  $f(x)$  in a finite number of function evaluations.*

**Proof:** First, it is shown that the algorithm generates a full set of conjugate directions unless it selects  $x^*$  as an iterate before this process is complete. Let  $\mathcal{V}_c$  be the set of conjugate directions at the  $j^{\text{th}}$  iteration, where  $x^{(j)}$  has been obtained from a search along  $v_c$ . Let  $\mathcal{M} = \{x^{(j)} + V_c\zeta : \exists \zeta \in R^c\}$ , and let  $x_b$  minimise  $f$  over  $\mathcal{M}$ . Although the searches along the members of  $\mathcal{V}_c$  do not select  $x_b$  as an iterate, they do provide enough information

to calculate  $x_b$  exactly when  $f$  is of the form (4). It is first shown that either (a)  $x_b = x^*$ ; or (b) a direction  $v_{\text{new}}$  conjugate to every member of  $\mathcal{V}_c$  is generated.

To show (b) occurs it is sufficient to show that the algorithm performs a set of line searches along the directions in  $\mathcal{V}_c$  from an iterate  $x^{(k)} \notin \mathcal{M}$ , for some  $k > j$ . Together with the parallel subspace theorem, the first such set of searches yields  $v_{\text{new}}$ .

If the algorithm takes a non-zero step along a direction in  $\mathcal{V}_{\text{nc}}$ , the linear independence of  $\mathcal{V}$  ensures the subsequent set of searches along the directions in  $\mathcal{V}_c$  are completed, and take place off  $\mathcal{M}$ . Otherwise the searches for  $\mathcal{V}_{\text{nc}}$  make no movement, and conjugacy ensures that one set of searches along the directions in  $\mathcal{V}_c$  will locate a grid local minimum. Steps 4 and 5 are then executed, ensuring the next iterate  $x$  satisfies  $f(x) \leq f(x_b)$ . If this inequality is strict, then  $x \notin \mathcal{M}$ . Otherwise  $x = x_b$ , and the next  $n$  line searches will either return  $x_b$  as a grid local minimum or move to a lower iterate (necessarily not on  $\mathcal{M}$ ). In the former case, the algorithm executes step 4 at  $x_b$ . If  $x_b = x^*$ , the solution has been found, otherwise  $\nabla f(x_b)$  is non-zero (because  $x_b \neq x^*$ ), and orthogonal to  $\mathcal{M}$ . The use of central differences means that  $\nabla f(x_b)$  is known exactly. Now  $V$  is of full rank, and so  $p = -VV^T \nabla f(x_b)$  is a non-zero direction of descent. The line  $\ell(\alpha) = x_b + \alpha p$ ,  $\alpha \in \mathbb{R}$  intersects  $\mathcal{M}$  at  $x_b$  only. Step 4 of the algorithm looks at two points on the line. These are  $x_b + p$  and  $x_b + \alpha_p p$  where the latter is the minimizer of  $f$  over  $\ell(\alpha)$ . Now because  $p$  is a descent direction at  $x_b$  it follows that neither  $x_b + p$  nor  $x_b + \alpha_p p$  lie on  $\mathcal{M}$ . Hence step 4 moves the sequence of iterates off  $\mathcal{M}$ .

The above argument shows the algorithm either encounters  $x^*$ , or generates a full set of conjugate directions. In the latter case  $g_v = V^T \nabla f$ , and, when  $c = n$ , the inverse Hessian  $(\nabla^2 f)^{-1} = VV^T$  because of the scaling in step 3. Hence  $p = -Vg_v$  is the exact step to  $x^*$ , and step 4 of the algorithm ensures that this step will be taken.  $\square$

## 5 Numerical Results

The algorithm was tested on a variety of general test problems, and on a family of quadratics.

### 5.1 Results for the full algorithm

The algorithm was tested on the first 19 test problems listed in [5]. The results for these problems are listed in table 1, where ‘# fcn’ denotes the number of function evaluations performed, and  $f^\sharp$  is the function value at the final iterate. The legends  $\|g_v^\sharp\|$ ,  $m^\sharp$ , and  $h^\sharp$  denote the final values for the norm of the gradient with respect to  $h^\sharp \eta$ , the number of meshes, and the final mesh size respectively. For all of these problems the algorithm was able to locate the optimal point, and terminated after satisfying the stopping condition (3). The second, starred, set of results for Powell’s badly scaled two dimensional function use a required accuracy of  $\tau_{\text{acc}} = 10^{-8}$  rather than  $10^{-5}$ . The latter, looser tolerance is achieved by points far from the solution. For  $\tau_{\text{acc}} = 10^{-5}$  the final iterate was  $x^\sharp = (1.33 * 10^{-5}, 7.52)$ , whereas for  $10^{-8}$  the final iterate was  $x^\sharp = (1.1 * 10^{-5}, 9.106)$ , which is the solution. The

Problem	$n$	# fcn	$f^\sharp$	$\ g_v^\sharp\ $	$m^\sharp$	$h^\sharp$
Rosenbrock	2	380	3.6e-11	8.2e-6	21	5.6e-4
Freudenstein & Roth	2	111	3.193	7.5e-6	5	7.4e-3
Powell badly scaled	2	734	1.9e-7	1.2e-6	29	1.3e-4
Powell badly scaled*	2	1784	6.7e-18	5.0e-9	72	1.1e-7
Brown badly scaled	2	58	1.4e-20	1.6e-10	4	0.056
Beale	2	87	5.6e-13	1.1e-6	8	5.2e-5
Jennrich & Sampson	2	154	124.4	1.6e-6	7	1.4e-3
Helical valley	3	11	0	0	1	1
Helical valley*	3	303	4.2e-11	8.5e-6	11	5.1e-4
Bard	3	200	17.43	1.5e-13	4	0.033
Gaussian	3	47	1.1e-8	1.2e-6	5	1.0e-3
Meyer	3	9070	87.95	1.0e-6	62	1.9e-4
Gulf Research	3	655	1.8e-13	1.9e-6	21	4.7e-6
Box 3-dimensional	3	227	0.01409	1.4e-6	10	5.4e-6
Powell singular	4	242	2.6e-11	5.4e-6	8	3.3e-5
Wood	4	315	4.9e-12	3.1e-6	11	1.1e-5
Kowalik and Osborne	4	317	3.1e-4	7.7e-6	9	1.2e-5
Brown and Dennis	4	232	85822	4.1e-6	6	0.021
Osborne 1	5	1413	5.5e-5	2.0e-6	15	1.7e-5
Biggs exp6	6	3403	1.9e-11	6.1e-6	36	8.0e-5
Osborne 2	11	2341	0.04014	1.7e-6	21	3.7e-7

Table 1: Numerical Results on 19 standard test functions for the standard algorithm. Here  $n$  is the dimension of the problem and ‘# fcn’ is the number of function evaluations performed. The quantities in the right hand four columns are respectively the final function value, the magnitude of the final gradient estimate  $g_v$ , the number of grids used, and the final grid size.

$n$	# fcn	$f^\sharp$	$\ g_v^\sharp\ $	$\ x^\sharp - x^*\ $
2	19	0	0	0
4	67	2.5e-32	2.0e-16	1.0e-16
6	121	1.2e-31	5.8e-16	7.3e-16
8	235	2.8e-30	2.4e-15	2.1e-15
10	353	1.7e-30	1.9e-15	1.4e-15
20	1156	1.4e-20	1.7e-10	8.7e-11
30	2317	2.4e-20	2.2e-20	3.0e-10

Table 2: Numerical Results on a family of quadratics.

second, starred, set of results for the helical valley problem use  $h^{(1)} = 0.9$  rather than  $h^{(1)} = 1$ . With the latter choice the solution  $x^*$  is a grid local minimizer of the initial grid, and so the algorithm locates it artificially fast.

The algorithm was also tested on a family of quadratics of the form

$$f(x) = (x - \mathbf{1})^T G_n (x - \mathbf{1}) \quad \text{where } \mathbf{1} = (1, 1, \dots, 1)^T \quad \text{and} \quad x^{(0)} = \pi \left( 1, \frac{1}{2}, \frac{1}{3}, \dots, \frac{1}{n} \right)^T$$

Here  $G_n$  is the  $n \times n$  tridiagonal matrix with all diagonal elements equal to 2, and all super- and sub-diagonal elements equal to 1. Results are listed in table 2, where the  $\sharp$  superscript denotes the value of the quantity taken at the final iterate  $x^\sharp$ , and  $x^*$  is the solution.

The results show that the algorithm is effective on a wide variety of problems which includes ill-conditioned problems. The property of exact termination on strictly convex quadratics is verified by the numerical results. The stopping condition is satisfied when  $f \approx 10^{-5}$ , yet the final function values are many orders of magnitude smaller than this.

## 5.2 Results for variations on the algorithm

Six variants of the algorithm were also tested on the 19 general test problems. These variations were obtained by deleting one or more parts of the algorithm. The first variant omits the ray search in step 2(e); the second omits the orthogonalization of  $V$  in step 7; and the third omits both the orthogonalization of  $V$  and the ray search in step 2(e). Results are presented in table 3. The fourth variant adjusts  $h$  only after a grid local minimum is found, and halves  $h$  on each such occasion. The fifth and sixth variants respectively omit step 4, and steps 4 and 5 of the algorithm. Results for these three variants are listed in table 4. The second, starred, sets of results for Powell's badly scaled function and the helical valley function are for the reasons described above. Each variant of the algorithm obtained the solution of the Powell badly scaled function with an accuracy of  $10^{-8}$ , but stopped short of the solution when the required accuracy was  $10^{-5}$ . This was due to the nature of Powell's badly scaled function, rather than the algorithm.

There are three ways the algorithm can terminate: by achieving the required accuracy; by reaching the minimum mesh size limit; and by reaching the maximum number of it-

Problem	$n$	Number of function evaluations			
		Full	no step 2(e)	no orthog.	no 2(e)/orthog.
Rosenbrock	2	380	481	601	1801
Freudenstein & Roth	2	75	68	76	71
Powell badly scaled	2	734	1356	808	779
Powell badly scaled*	2	1784	1740	1908	9059
Brown badly scaled	2	58	54	53	49
Beale	2	87	80	95	86
Jennrich & Sampson	2	154	181	159	286
Helical valley	3	11	11	11	11
Helical valley*	3	303	524	352	1513
Bard	3	200	196	200	196
Gaussian	3	47	46	47	46
Meyer	3	9070	30413†	28527†	151128†
Gulf Research	3	655	819	668	$> 10^6$ ‡ ◊
Box 3-dimensional	3	227	215	185	177
Powell singular	4	242	327	539	872
Wood	4	315	345	299	333
Kowalik and Osborne	4	317	387	340	438
Brown and Dennis	4	232	229	193	183
Osborne 1	5	1413	752◊	1410	9077
Biggs exp6	6	3403	8105	3178	1052◊
Osborne 2	11	2341	25019	2992◊	25182◊

Table 3: Numerical Results on 19 standard test functions for several variants of the algorithm. Column 4 lists results when the ray search in step 2(e) is omitted. The results in column 5 were generated with the orthogonalization of  $V$  in step 7 omitted, and the results in column 6 are for when both the orthogonalization of  $V$  and step 2(e) were omitted.

Problem	$n$	Number of function evaluations			
		Full	no step 4	no step 4,5	$h^+ = \frac{1}{2}h$
Rosenbrock	2	380	274	319	341
Freudenstein & Roth	2	75	78	88	81
Powell badly scaled	2	734	868	1060	1055†
Powell badly scaled*	2	1784	1679	2067	2544
Brown badly scaled	2	58	43	111	58
Beale	2	87	79	143	90
Jennrich & Sampson	2	154	150	171	192
Helical valley	3	11	11	11	11
Helical valley*	3	303	288	513	280
Bard	3	200	174	247	196
Gaussian	3	47	49	87	93
Meyer	3	9070	9340	8625†	16493†◇
Gulf Research	3	655	642	790	660
Box 3-dimensional	3	227	198	240	243
Powell singular	4	242	381	380	319
Wood	4	315	447	564	299
Kowalik and Osborne	4	317	308	376	338
Brown and Dennis	4	232	214	408	232
Osborne 1	5	1413	3263	5929	3984
Biggs exp6	6	3403	8578	6653	2442
Osborne 2	11	2341	3637	$> 10^6$ ‡ ◇	2192

Table 4: Numerical Results on 19 standard test functions for several variants of the algorithm. The fourth column lists results for the algorithm with the quasi-Newton step removed. The fifth column lists results when both the quasi-Newton step and the step to the estimated minimum over the manifold  $\mathcal{M}$  were omitted. The sixth lists results for  $h$  kept constant except when a grid local minimum is found; immediately after this occurs  $h$  is reduced by a factor of 2.

erations. Results for which the algorithm terminated for the second or third reasons are marked with a † and ‡ respectively. In each case the lower limit on the mesh size  $h$  was set at  $0.01\tau_{\text{acc}}$ . Entries marked with a  $\diamond$  terminated before the optimal function value was attained.

The extra costs of steps 2(e), 3, and 4 are in terms of extra function evaluations, and so the work saved in omitting these steps is reflected in the listings in tables 3 and 4. In contrast, the savings in omitting the orthogonalization of  $V$  in step 7 take the form of reduced overheads, and so are not reflected in the tabulated figures.

Table 3 shows that deleting one of the skewer search in step 2(e) or the orthogonalization in step 7 either makes little difference, or worsens the algorithm's performance. Deleting both steps 2(e) and 7 significantly worsens the algorithm's performance on over half the problems listed.

A danger with any grid method is that the grid local minimizer lies along a narrow valley which does not lie along any axis of the grid. Any significant movement along the valley requires many short movements along each of the grid axes in turn. Between grid local minimizers, opportunities to re-orient the grid are limited, and so it is possible that the algorithm will get forced into a very long zig-zagging search on one grid. The orthogonalization of  $V$  in step 7 and the skewer search in step 2(e) have been included to reduce the risk of this occurring, but they do not provide immunity. On both occasions when the algorithm exceeded the function evaluation limit, a very large number of line searches had been performed on one grid — indicating that zig-zagging was occurring.

Steps 4 and 5 perform similar functions in that both represent a step to the minimizer of an approximating quadratic on some subspace of  $R^n$ . The results listed in table 4 show that omitting step 4 improved performance on a few problems such as Rosenbrock's function, but worsened performance on others. In particular, the problems in higher dimensions required more function evaluations to solve. Deleting both steps 4 and 5 worsened performance on most problems, particularly those of higher dimension.

The algorithm was terminated by the limit on  $h$  on six runs: five of these were for the Meyer problem, and the sixth for Powell's badly scaled problem. On all but one of these runs the algorithm obtained the optimal function value. For the Meyer function with  $h^{(k+1)} = h^{(k)}/2$  only, the algorithm stopped before the optimal function value was achieved. A simple calculation shows that this variant of the algorithm is limited to 25 grids — essentially the algorithm ran out of grids before reaching the solution. The same variant also ran out of grids before satisfying (3) on Powell's badly scaled problem.

The full algorithm was not the fastest variant on most of the problems, although for many problems the difference was marginal. However the full algorithm and the variant with step 4 omitted were the only two to solve all problems in a reasonable amount of time. The results indicate that the full algorithm is the more effective of these two variants in dimensions greater than about 3 or 4.

## 6 Conclusion

A provably convergent derivative free conjugate directions algorithm has been presented. Numerical results for general unconstrained problems show that the algorithm effective in practice, even on problems which are ill-conditioned. The algorithm is based on a sequence of grids which are chosen to incorporate known second derivative information generated by use of the parallel subspace theorem. Consequently the algorithm retains the property of exact termination on strictly convex quadratics. This property is verified by numerical results for the family of tridiagonal quadratics. The algorithm is capable of making use of the continuity of second derivatives, but convergence is guaranteed under the weaker requirement of a  $C^1$  locally Lipschitz objective function. A number of anti-zigzagging features were included in the algorithm. These features are not required by the convergence theory, but improved the algorithm's performance on the set of general test problems.

## References

- [1] Conn, A. R., K. Scheinberg, and P. Toint, *On the convergence of derivative free methods for unconstrained optimization*, in *Approximation Theory and Optimization*, M. D. Buhmann and A. Iserles, eds, Cambridge, 1997, Cambridge University Press, pp 83–108.
- [2] Coope, I.D. and C.J. Price, *On the convergence of grid based methods for unconstrained optimization*, Research Report 180, Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand.
- [3] Fletcher, R., *Practical Methods of Optimization*, ©1987, Wiley.
- [4] Gill, P. E., W. Murray, and M. H. Wright, *Practical Optimization*, ©1981, Academic Press.
- [5] Moré, J. J., B. S. Garbow, and K. E. Hillstom, *Testing unconstrained optimization software*, ACM Trans. Math. Software 7 (1981), pp 17–41.
- [6] Powell, M. J. D., *Direct search algorithms for optimization calculations*, Acta Numerica 7, pp 287–336, Cambridge University Press (1998).
- [7] Powell, M. J. D., *An efficient method of finding the minimum of a function of several variables without calculating derivatives*, Computer J. 7 (1964), pp 155–162.
- [8] Smith, C. S., *The automatic computation of maximum likelihood estimates*, N. C. B. Sci. Dept. Report SC846/MR/40 (1962).
- [9] Torczon, V., *On the convergence of pattern search algorithms*, SIAM J. Optimization 7 (1997), pp 1–25.