JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 113, No. 1, pp. 5–19, April 2002 (©2002)

A Convergent Variant of the Nelder–Mead Algorithm

C. J. Price,[1] I. D. Coope,[2] and D. Byatt[3]

Communicated by Y. Zhang

Abstract. The Nelder–Mead algorithm (1965) for unconstrained optimization has been used extensively to solve parameter estimation and other problems. Despite its age, it is still the method of choice for many practitioners in the fields of statistics, engineering, and the physical and medical sciences because it is easy to code and very easy to use. It belongs to a class of methods which do not require derivatives and which are often claimed to be robust for problems with discontinuities or where the function values are noisy. Recently (1998), it has been shown that the method can fail to converge or converge to nonsolutions on certain classes of problems. Only very limited convergence results exist for a restricted class of problems in one or two dimensions. In this paper, a provably convergent variant of the Nelder–Mead simplex method is presented and analyzed. Numerical results are included to show that the modified algorithm is effective in practice.

Key Words. Nelder–Mead algorithm, derivative free optimization, positive basis methods, simplex, polytope, frame based methods.

1. Introduction

The Nelder–Mead algorithm (Ref. 1) for unconstrained optimization has been used extensively to solve parameter estimation and other problems since it was first proposed in 1965. Despite its age, it is still the method of choice for many practitioners in the fields of statistics, engineering, and the physical and medical sciences (Ref. 2) because it is easy to code and very easy to use. It belongs to a class of methods which do not require derivatives

[1]Lecturer, Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand.

[2]Senior Lecturer, Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand.

[3]Postgraduate Student, Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand.

and which are often claimed to be robust for problems with discontinuities or where the function values are noisy. However, only recently (Ref. 3) it has been shown that the method can fail to converge or converge to non-solutions on certain classes of problems. Only very limited convergence results exist for a restricted class of problems in one and two dimensions (Ref. 4), which assume that f is strictly convex among other things. Accordingly, if the Nelder–Mead method is to possess acceptable convergence properties, then it must be altered in some way. This paper describes such a variant of the Nelder–Mead algorithm and lists numerical results which show that the modified method is effective in practice.

The Nelder–Mead algorithm is designed to solve unconstrained optimization problems, which are of the form

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

Here, acceptable solutions are stationary points of f not higher than any point on the initial simplex.

The Nelder–Mead algorithm generates a sequence of simplices, where each simplex is defined by $n+1$ distinct points $x_0, \dots, x_n$, with corresponding function values $f_0, \dots, f_n$. It is assumed that the points are sorted so that the function values are in increasing order.

At each iteration, the Nelder–Mead method examines a small number of points along the line $\bar{x} + \zeta(\bar{x} - x_n)$, $\zeta \in R$, where $\bar{x}$ is the centroid of the points $x_0, \dots, x_{n-1}$. At each iteration, one or more of four ζ values are considered. These four values, denoted α, $\pm\beta$, γ, are required to satisfy

$$0 < \beta < 1, \qquad 0 < \alpha < \gamma, \quad \text{where } \gamma > 1. \tag{1}$$

Typical values (Refs. 1 and 4) for these parameters are $\alpha = 1, \beta = 0.5, \gamma = 2$, and these values are used herein. The values $\alpha, \gamma, \beta, -\beta$ respectively yield the reflection x_r, expansion x_e, outer contraction x_c, and inner contraction x_{cc} points. The function values at these four points are denoted by f_r, f_e, f_c, f_{cc}. The algorithm first tries the reflected point x_r in order to locate a point that is acceptable (i.e., lower than x_{n-1}). If unsuccessful or if x_r is the best known point, the algorithm then respectively either tries one of the two contraction points or the expansion point x_e. If an acceptable point is found, then that point replaces x_n, producing a new simplex. If no acceptable point is found the algorithm shrinks the points $x_1, \dots, x_n$ toward the lowest x_0 which yields a new simplex. The shrink replaces each x_i by $x_0 + \sigma(x_i - x_0)$ for $i = 1, \dots, n$, where $0 < \sigma < 1$. Typically (Refs. 1 and 4), $\sigma = 0.5$. A new iteration is then started. The algorithm is precisely expressed as Algorithm 1.1, which is a restatement of the version given in Ref. 4.

Algorithm 1.1.

```
Step 1. Set $j = 1$. Get the initial simplex and calculate its function
        values. Choose $\alpha$, $\beta$, $\sigma$, $\gamma$ subject to (1).
Step 2. Sort the vertices $x_0^{(j)}, \dots, x_n^{(j)}$ of the current simplex so that
        their function values $f_0^{(j)}, \dots, f_n^{(j)}$ are in ascending order.
Step 3. Calculate $x_r^{(j)}$ and $f_r^{(j)}$. Let $x_{\text{new}}^{(j)}$ be undefined.
Step 4. Depending on $f_r$, do one of the four following cases:
        (a) If $f_r < f_0$, then first calculate $f_e = f(x_e)$; second, if $f_e < f_r$,
            then set $x_{\text{new}} = x_e$; otherwise, set $x_{\text{new}} = x_r$.
        (b) If $f_0 \leq f_r < f_{n-1}$, then set $x_{\text{new}} = x_r$.
        (c) If $f_{n-1} \leq f_r < f_n$, then first calculate $f_c = f(x_c)$; second, if
            $f_c \leq f_n$ then set $x_{\text{new}} = x_c$.
        (d) If $f_n \leq f_r$, then first calculate $f_{cc} = f(x_{cc})$; second, if
            $f_{cc} \leq f_n$ then set $x_{\text{new}} = x_{cc}$
Step 5. If $x_{\text{new}}$ is undefined, then shrink the simplex; otherwise, set
        $x_i^{(j+1)} = x_i^{(j)}$ for $i = 0, \dots, n-1$ and set $x_n^{(j+1)} = x_{\text{new}}^{(j)}$.
Step 6. Let $j = j + 1$. If the stopping conditions are not satisfied, go to
        Step 2.
```

The (j) superscripts indicate quantities for the jth simplex. These superscripts have been omitted in Step 4 for clarity. The case chosen in Step 4 dictates at which of the four points x_r, x_e, x_c, x_{cc} the algorithm calculates f. If x_r is the best known point, then the algorithm tries the expansion point x_e in the hope that it may lead to further improvement. Otherwise, if x_r is higher than x_{n-1}, then the algorithm calculates f at whichever of the two contraction points x_c and x_{cc} is closer to the lower of x_r and x_n. Stopping conditions for Algorithm 1.1 are satisfied when acceptably small values are obtained for $|f_n^{(j)} - f_0^{(j)}|$ and for $\|x_i^{(j)} - x_0^{(j)}\|_\infty$, $i = 1, \dots, n$.

The original Nelder–Mead algorithm differs from Algorithm 1.1 on three points: x_e is accepted if it is lower than x_0; x_r is accepted if $f_r = f_{n-1}$; and x_c is accepted when $f_r = f_n$.

Currently known convergence results for the Nelder–Mead algorithm (Refs. 4 and 5) are poor and do not provide the kind of guarantees that one would normally expect. Indeed, the Nelder–Mead algorithm has been shown to not converge on some C^1 functions (Ref. 3). Kelley (Ref. 5) establishes the following partial convergence result: if the Nelder–Mead algorithm satisfies a sufficient descent condition at all but a finite number of iterations, and if no shrink steps are performed, then the cluster points of the sequence of iterates are stationary points of f. This result is only partial in that there is no control over whether or not sufficient descent is obtained, and shrink steps are avoided. Kelley proposes an oriented restart instead of a shrink, but this provides no guarantee of convergence.

Rykov (Ref. 6) has also developed a class of convergent variants of the Nelder–Mead algorithm. Rykov's modifications are much more extensive than the ones reported here and result in a class of algorithms which are similar to and predate the multidirectional search algorithm of Dennis and Torczon (Ref. 7). Rykov's methods map simplex vertices through a number of different centroids, where each such centroid is the centroid of some subset of the simplex vertices. The best known point is then taken as the new iterate: no sufficient descent condition is employed. In contrast to Ref. 6, the algorithm presented herein is very similar to the original Nelder–Mead algorithm; in the authors' opinion, it may be as similar as any convergent algorithm can be to the original Nelder–Mead algorithm.

The approach herein is along the lines of Ref. 5 in that a sufficient descent condition is used. However, when an iteration of the Nelder–Mead algorithm does not yield sufficient descent, then a fragment of a grid called a frame is completed, thereby guaranteeing convergence.

Frames are defined in terms of positive bases, both of which are described in Section 2 along with the convergence theory of frame based methods (Ref. 8). Section 3 describes the modified algorithm and embeds it in the framework of the convergence theory, thereby establishing convergence for C^1 objective functions under mild conditions. Gradient information is not required in order to execute the modified algorithm. Section 4 presents numerical comparisons between the modified and unmodified Nelder–Mead algorithms. Concluding remarks are made in Section 5.

2. Positive Bases and Frames

The modified Nelder–Mead algorithm uses the convergence theory developed in Ref. 8, which is expressed in terms of positive bases and frames. These concepts are discussed first, followed by a brief description of the convergence theory. A positive basis $\mathscr{V}_+$ satisfies the defining properties (Ref. 9) that:

(i) Every vector in R^n is a nonnegative combination of the members of $\mathscr{V}_+$.

(ii) Any proper subset of $\mathscr{V}_+$ is not a positive basis.

Herein, each positive basis $\mathscr{V}_+^{(k)}$ is defined in terms of a basis $\mathscr{V}^{(k)}$, where

$$\mathscr{V}^{(k)} = \{v_i^{(k)} \in R^n : i = 1, \ldots, n\},$$

and k is the frame number. Each such basis $\mathcal{V}^{(k)}$ is required to satisfy the conditions

$$|\det([v_1^{(k)}, v_2^{(k)}, \dots, v_n^{(k)}])| > \tau \quad \text{and} \quad \|v_i^{(k)}\| \leq K_0, \qquad \forall i \in 1, \dots, n, \tag{2}$$

where τ and K_0 are positive constants independent of k. The positive bases used herein are of a particular form: the first n members of $\mathcal{V}_+^{(k)}$ are those of $\mathcal{V}^{(k)}$ in the same order; the final member of $\mathcal{V}_+^{(k)}$ is

$$v_{n+1}^{(k)} = -[(\gamma - \alpha)/\alpha n] \sum_{i=1}^{n} v_i^{(k)}, \tag{3}$$

which, for $\gamma > \alpha$, yields ordered positive bases of the form

$$\mathcal{V}_+^{(k)} = \{v_1^{(k)}, v_2^{(k)}, \dots, v_n^{(k)}, -[(\gamma - \alpha)/\alpha n] \sum_{i=1}^{n} v_i^{(k)}\}. \tag{4}$$

Equation (4) imposes a specific order on the members of $\mathcal{V}_+^{(k)}$, and so these positive bases will be referred to as ordered positive bases from now on. The bound on each $\|v_i^{(k)}\|$ in (2) can be extended to all members of each $\mathcal{V}_+^{(k)}$ by setting K equal to the larger of K_0 and $(\gamma - \alpha)K_0/\alpha$, yielding

$$\|v_i^{(k)}\| \leq K, \qquad \forall k \text{ and } \forall i = 1, \dots, n+1. \tag{5}$$

Herein, a frame Φ consists of $n+1$ points arranged around a central point called the frame center. The frame $\Phi(x, \mathcal{V}_+^{(k)}, h^{(k)})$ is specified in terms of a frame center x, a positive basis $\mathcal{V}_+^{(k)}$, and a frame size $h^{(k)}$ as follows:

$$\Phi = \{x + h^{(k)} v: v \in \mathcal{V}_+^{(k)}\}.$$

The frame size $h^{(k)}$ is adjusted from time to time in a manner that guarantees convergence under appropriate conditions.

Property (i) of positive bases is useful in formulating stopping conditions for derivative-free algorithms via the following result (see e.g. Refs. 8, 10, 11), reproduced here for convenience.

Theorem 2.1. If the set of vectors $\mathcal{V}_+$ is a positive basis, then

$$g^T v \geq 0, \qquad \forall v \in \mathcal{V}_+ \Rightarrow g = 0. \tag{6}$$

Proof. See e.g. Ref. 8 or Ref. 10. □

A discrete approximation to condition (6) gives the following definition.

Definition 2.1. Minimal Frame. A frame $\Phi(x, \mathcal{V}_+, h)$ is minimal iff

$$f(x + hv) \geq f(x), \qquad \forall v \in \mathcal{V}_+.$$

For convergence purposes, it is convenient to work with frames that are only nearly minimal; this concept is defined precisely below.

Definition 2.2. Quasiminimal Frame and Quasiminimal Point. A frame $\Phi = \Phi(x, \mathcal{V}_+, h)$ which satisfies the weaker condition

$$f(x+hv)+\epsilon \geq f(x), \qquad \forall v \in \mathcal{V}_+,$$

is called ϵ-quasiminimal, and the corresponding point x will be referred to as an ϵ-quasiminimal point.

When the value of the constant ϵ is not in doubt, the shorter term "quasiminimal" will be used. In practice a useful choice for ϵ is $\epsilon = Nh^{\nu}$, where $\nu > 1$ and N is a positive constant. In determining whether or not an iterate is quasiminimal, ϵ acts effectively as a measure of sufficient descent.

These definitions allow the formation of a generic template for frame based algorithms (Ref. 8), a slightly simplified version of which is given below. Here, $f^{(k)} = f(x^{(k)})$ is used, where k counts the number of frames. In contrast, the $x_i^{(j)}$ are points on the jth simplex. The variables m and $z^{(m)}$ count the number of quasiminimal frames and denote the quasiminimal points.

Algorithm 2.1.

Step 1. Initialize $m = k = 0$, and let the initial point be $x^{(1)}$. Choose $\nu > 1$ and $N > 0$.

Step 2. Choose h. Set $\epsilon = Nh^{\nu}$.

Step 3. Perform the following process repeatedly until a quasiminimal frame is found: Execute any finite process which first increments k and then either generates a quasiminimal frame $\Phi(x^{(k+1)}, \mathcal{V}_+^{(k)}, h)$ which satisfies $f^{(k+1)} \leq f^{(k)}$ or locates a point $x^{(k+1)}$ satisfying

$$f^{(k+1)} < f^{(k)} - \epsilon. \tag{7}$$

Step 4. Increment m. Let $z^{(m)} = x^{(k+1)}$ [$z^{(m)}$ is quasiminimal]. If the stopping conditions are not satisfied, go to Step 2.

The mode of operation of Algorithm 2.1 is as follows. The outer loop (Steps 2–4) generates a sequence of quasiminimal iterates, which must converge under mild conditions to one or more stationary points of f. The inner loop (Step 3) generates a sequence of points until a quasiminimal point is found. Each iteration of the inner loop examines the points which form the frame around the current frame center, as well as a finite number of other arbitrarily chosen points. If descent of at least ϵ is not forthcoming, then the inner loop terminates and the current frame center is quasiminimal. The purpose of the inner loop is to locate a quasiminimal point within a finite number of function evaluations.

This template is a direct specialization of the template presented in Ref. 8 except in one way. Here, Step 3 is required to either obtain sufficient descent or locate a quasiminimal frame centered on a point $x^{(k+1)}$, where $x^{(k+1)}$ is not higher than $x^{(k)}$. In contrast, the template in Ref. 8 requires that the frame be centered on $x^{(k)}$. The template in Ref. 8 generates a sequence of frames of the form $\Phi(x^{(k)}, \mathscr{V}_+^{(k)}, h^{(k)})$ such that an infinite subsequence of these frames is quasiminimal. Algorithm 2.1 generates a sequence of frames of the form $\Phi(x^{(k+1)}, \mathscr{V}_+^{(k)}, h^{(k)})$ which has an infinite subsequence of quasiminimal frames. The conditions on the sequences $\mathscr{V}_+^{(k)}\}$ and $\{h^{(k)}\}$ are invariant under finite shifts of the form $h^{(k)} \rightarrow h^{(k+1)}$. Hence, by relabeling these sequences, the convergence theory of Ref. 8 may be directly applied to the sequence $\{\Phi(x^{(k+1)}, \mathscr{V}_+^{(k+1)}, h^{(k+1)})\}$. This is restated here in the following theorem.

Theorem 2.2. Assume the following:

(a) the sequence of iterates $\{x^{(k)}\}$ is bounded;
(b) f is continuously differentiable and its gradient ∇f is Lipschitz in any bounded region of R^n;
(c) $\exists\, K, \tau > 0$ such that $|\det([v_1^{(k)} \cdots v_n^{(k)}])| > \tau$, for all k, and $\|v_i^{(k)}\| \leq K$, for all k and for all $i = 1, \ldots, n+1$;
(d) $h^{(k)} \rightarrow 0$ as $k \rightarrow \infty$.

Then, each cluster point $z^{(\infty)}$ of the sequence of quasiminimal iterates $\{z^{(m)}\}$ is a stationary point of $f(x)$.

Proof. See Ref. 8. It is shown in Ref. 11 that the Lipschitz constant in (b) is superfluous: C^1 continuity of f is sufficient. Reference 8 allows N to be replaced by a sequence $\{N^{(k)}\}$ for which a fixed upper bound $N_{\max}$ exists such that $0 < N^{(k)} \leq N_{\max}$ for all k. □

The monotonicity of $\{f^{(k)}\}$ means that the sequence $\{x^{(k)}\}$ converges to a set of points on which f is constant. In the usual case when $\{x^{(k)}\}$ converges to a unique point, that limit point is a stationary point of f.

3. Modified Nelder–Mead Algorithm

The Nelder–Mead method is embedded in the convergent algorithm template as follows. First, let $x_{\text{low}}^{(k)}$ be the lowest point in the current simplex at the start of Step 3. The finite process in Step 3 is used to execute a finite number of Nelder–Mead iterations. This finite process repeatedly performs Nelder–Mead steps, provided each such step replaces $x_n^{(j)}$ with a point not higher than $f_n^{(j)} - \epsilon$. After a finite number of iterations, either a point at least ϵ lower than $x_{\text{low}}^{(k)}$ is found, or the highest simplex function value is not reduced by at least ϵ. This completes one iteration of the inner loop. When sufficient descent has not been obtained, the algorithm forms a frame around the lowest known point. This frame either yields sufficient descent or is quasiminimal. In the latter case, Step 3 terminates.

Each execution of Step 3 may produce a finite number of intermediate simplices before terminating with a simplex containing either a point at least ϵ lower than $x_{\text{low}}^{(k)}$ or locating a quasiminimal frame. In the former case, this lower point becomes the new iterate $x^{(k+1)}$; in the latter case, $x^{(k+1)}$ is the center of this quasiminimal frame.

The Nelder–Mead method is now expressed in terms of frames, which allows various modifications to be discussed more easily. The side vectors v_i of the simplex $x_0, \ldots, x_n$ are defined as

$$v_i = (x_i - x_0)/h, \qquad \forall i = 1, \ldots, n.$$

These side vectors form the first n members of a positive basis which, together with (4), completely define a positive basis for the simplex $x_0, \ldots, x_n$. Clearly, each simplex is one point short of being a frame, and each simplex point other than x_0 satisfies

$$x_i = x_0 + hv_i.$$

To complete the frame, a new point x_p (called the pseudo-expand point) is added, where

$$x_p = x_0 + hv_{n+1} = x_0 - h\,[(\gamma - \alpha)/\alpha n] \sum_{i=1}^{n} v_i.$$

This point is an entirely natural one for a Nelder–Mead like algorithm to use.

Consider the simplex $x_1, \ldots, x_n, H$, called the ghost simplex, where the point

$$H = \bar{g} - (x_0 - \bar{g})/\alpha$$

and $\bar{g}$ is the centroid of the points $x_1, \ldots, x_n$. Let H be the highest point on the ghost simplex. The Nelder–Mead algorithm applied to the ghost simplex will immediately reflect H through $\bar{g}$ and calculate f at the point x_0. Now, x_0 is the lowest point in $\{x_0, \ldots, x_n\}$, and so the Nelder–Mead method will then attempt an expansion by examining the point $\bar{g} + \gamma(x_0 - \bar{g})/\alpha$ which is the pseudo-expand point x_p. In particular, when $\alpha = 1$ and $\gamma = 2$,

$$x_p = x_0 + (x_0 - \bar{g}).$$

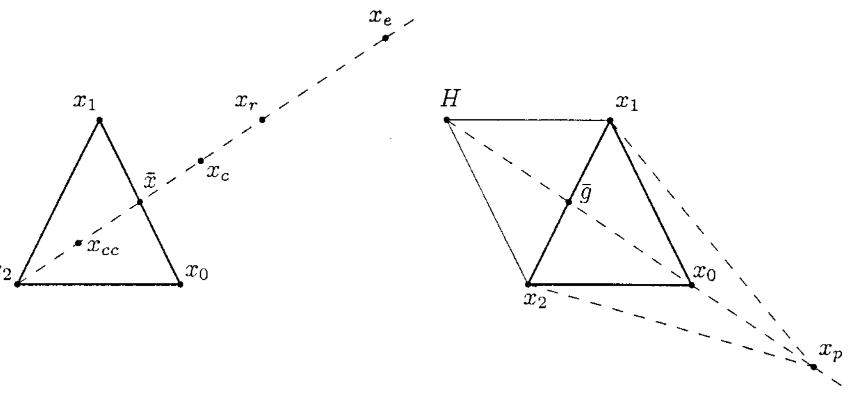

Fig. 1. Trial points from the Nelder–Mead algorithm (left) and the pseudo-expand point from the ghost simplex for the modified Nelder–Mead algorithm (right).

Figure 1 shows the positions of the standard Nelder–Mead points (left-hand image) and those arising from the ghost simplex (right-hand image) for a 2-dimensional simplex x_0, x_1, x_2.

Hence, if sufficient descent is not forthcoming from a Nelder–Mead iteration a frame can be completed by assuming a fictitious history for the algorithm. If the pseudo-expand point yields sufficient descent, then the algorithm continues with another Nelder–Mead step; otherwise, the frame is quasiminimal and the algorithm shrinks the frame and may also reorient the frame.

Algorithm 3.1.

```
Step 1. Let $m=j=1$, and choose the initial simplex. Choose $v>1, N>0, \kappa>1, h>0$. Set $f_n^{(0)}=\infty$. Let $\epsilon=Nh^v$.
Step 2. While $f_n^{(j-1)}-f_n^{(j)}>\epsilon$ and the stopping conditions do not hold, execute iterations of the Nelder–Mead method without shrinks, and increment $j$ after each iteration.
Step 3. If the basis violates (2) or (5), reshape the basis. Set $j=j+1$.
Step 4. Calculate $x_p^{(j)}$ and $f_p^{(j)}$.
Step 5. Repeat the following steps until a frame that is not quasiminimal is found:
    (a) Set $z^{(m)}=x_0^{(j)}$ and increment $m$.
    (b) If the basis has not been reshaped, reshape the basis; otherwise, set $h=h/\kappa$, set $\epsilon=Nh^v$, and reverse the basis directions.
    (c) Set $j=j+1$. Calculate the function values on the new frame.
```

Step 6. Choose the new simplex as $\{x_1^{(j)}, \ldots, x_n^{(j)}\}$ and the lower of $x_0^{(j)}$ and $x_p^{(j)}$. Go to Step 2.

The variable j counts the number of Nelder–Mead like iterations, whereas k, which is not used explicitly, counts the number of template iterates. Ties in the sorting process in the Nelder–Mead iterations are resolved on an oldest-is-last basis in the list $x_0^{(j)}, \ldots, x_n^{(j)}$. The stopping conditions are identical to those for Algorithm 1.1.

Although the convergence analysis requires C^1 continuity, gradient information is not required in order to execute Algorithm 3.1. The process of checking that the bounds in (2) are satisfied and the reshaping of the bases when these bounds are violated are discussed in Sections 3.1 and 3.2.

3.1. Calculating the Determinant. Satisfaction of (2) requires that the determinant of the matrix $V = [v_1 \cdots v_n]$ be calculated. This can either be calculated directly or more efficiently by the following process. If the volume of the jth simplex is known, then the volume of each subsequent simplex can be calculated easily because the four operations of reflection, expansion, contraction, and shrink scale the volume of the simplex by $\gamma, \alpha, \beta, \sigma^n$ respectively. For convenience, the ratio of these volumes is denoted by μ, where the final volume is μ times the original. Elementary calculations show that the volume of a simplex $x_0, \ldots, x_n$ is

$$(1/n!)\,|\det[\hat{v}_1, \ldots, \hat{v}_n]| \prod_{i=1}^{n} \|v_i\|,$$

and hence,

$$|\det[\hat{v}_1^{(j)} \cdots \hat{v}_n^{(j)}]| = \mu |\det[\hat{v}_1, \ldots, \hat{v}_n]| \prod_{i=1}^{n} \|v_i\|/\|v_i^{(j)}\|,$$

where each $\hat{v}_i$ is the unit vector satisfying $v_i = \|v_i\|\hat{v}_i$ and all unmarked norms are 2-norms. Hence, the determinant in (2) can be calculated very efficiently.

3.1. Reshaping the Frame. If the side vectors $v_1, \ldots, v_n$ violate the bounds in (2) or (5), then the simplex must be reshaped or the convergence theory is invalidated. First, $v_1, \ldots, v_n$ are sorted into decreasing order of length. The matrix $V = [v_1, \ldots, v_n]$ is formed and then factors Q (orthogonal) and R (upper triangular) are calculated such that

$$QR = V.$$

The columns $q_1, \ldots, q_n$ of Q form the directions of the new basis, where each such q_i is scaled by a factor D_i, calculated as follows:

$$D_i = \operatorname{sign}(R_{ii}) \min(K_0, \max(|R_{ii}|, \bar{R}/10)), \quad \text{where } \bar{R} = (1/n) \sum |R_{ii}|.$$

This scaling ensures that (5) is also satisfied and generates a simplex with orthogonal side vectors no longer than the previous longest side vector v_1. The orthogonality means that the volume of this simplex is easily found. Other reshapes are possible, and several are discussed in Ref. 11.

4. Numerical Results

The modified Nelder–Mead algorithm was compared against the MATLAB R11.1 implementation of the Nelder–Mead algorithm (FMINSEARCH) on a Sun Enterprise 450. This version of the Nelder–Mead algorithm will be referred to as the standard Nelder–Mead algorithm from now on. The modified and standard Nelder–Mead algorithms were tested on a variety of problems, most of which are listed in Ref. 12. The remaining two test problems are the standard quadratic $f(x) = x^T x$ with the starting point $x^{(1)} = (2, 1, 1, \dots, 1)^T$, and the McKinnon (Ref. 3) function

$$f(x_1, x_2) = \begin{Bmatrix} \theta\phi|x_1|^\omega + x_2 + x_2^2, & x_1 < 0, \\ \theta x_1^\omega + x_2 + x_2^2, & x_1 \geq 0, \end{Bmatrix}$$

with the parameter values

$$\theta = 6, \qquad \phi = 60, \qquad \omega = 2.$$

Two sets of results for the McKinnon function are presented. The first uses the standard starting simplex[4] (used by FMINSEARCH), and the second (marked with an asterisk) uses the starting simplex given by McKinnon. The latter simplex is chosen so the Nelder–Mead method fails by converging to the origin through an infinite sequence of inner contractions. For all other test problems, the initial simplex was that used by FMINSEARCH. This initial simplex is obtained by displacing a given initial point along each axis in turn, where each displacement is 5% of the initial point corresponding coordinate value. A displacement of 0.00025 is used when a coordinate value is zero.

The results are listed in Tables 1 and 2, where n is the dimension of the problem, Minimum is the final function value attained by the relevant algorithm, and FE denotes the number of function evaluations required to reach this function value. The column marked MS(%) for the modified method lists the percentage of steps that were not standard Nelder–Mead steps. The symbol † marks entries for which the final iterate only poorly approximates the solution. Similarly, ‡ marks entries for which the standard

[4]This was suggested by L. Pfeffer, Stanford University.

Table 1. Results for problems of dimensions 2 to 4.

		FMINSEARCH		Modified N–M		
Function	n	FE	Minimum	FE	Minimum	MS(%)
Rosenbrock	2	219	1.099e–18	285	1.391e–17	3.0
Freudenstein and Roth	2	172	48.9843	217	48.9843	8.2
Powell badly scaled	2	754	1.111e–25	969	4.240e–25	1.2
Brown badly scaled	2	335	7.039e–18	498	7.998e–17	6.7
Beale	2	162	6.114e–18	191	2.078e–18	13.2
Jennrich and Sampson	2	133	124.362	157	124.362	7.8
McKinnon	2	290	–0.25000	426	–0.25000	2.4
McKinnon*	2	359	0.00000†	351	–0.25000	1.2
Helical Valley	3	428	4.785e–17	342	9.832e–16	4.4
Bard	3	100004‡	17.4287	1134	17.4287	1.8
Gaussian	3	216	1.1279e–8	194	1.1279e–8	6.2
Meyer	3	100004‡	87.9459	2801	87.9459	0.6
Gulf Research	3	687	1.140e–22	529	5.445e–19	1.9
Box	3	701	3.057e–22	478	8.705e–21	4.3
Powell singular	4	956	3.564e–28	1045	6.735e–26	4.2
Woods	4	572	1.564e–17	656	2.574e–16	5.0
Kowalik and Osborne	4	398	3.07506e–4	653	3.07506e–4	2.7
Brown and Dennis	4	100001‡	85822.2	603	85822.2	4.9
Standard quadratic	4	326	4.529e–17	440	2.154e–17	4.2
Penalty I	4	1371	2.24998e–5	1848	2.24998e–5	1.3
Penalty II	4	3730	9.37629e–6	4689	9.37629e–6	0.4

method failed to terminate properly due to roundoff error preventing movement of the simplex. In these cases, the algorithm was halted when the maximum number of function evaluations exceeded 10^5.

On each run, the algorithm was required to attain an accuracy of

$$\max_{i=1,\dots,n} \|x_i - x_0\|_\infty \le x_{\text{tol}} \quad \text{and} \quad \max_{i=1,\dots,n} |f_i - f_0| \le f_{\text{tol}},$$

where $x_{\text{tol}} = 10^{-8}$ and $f_{\text{tol}} = 10^{-12}$ were used. These values are tighter than the default settings of 10^{-4} for both used by FMINSEARCH. Test runs showed that the default settings allowed both algorithms to halt prematurely.

The standard values

$$\alpha = 1, \beta = \sigma = 1/2, \gamma = 2$$

were used for the Nelder–Mead parameters. Other parameters were given the following values in the test runs:

$$N = [(f_n^{(1)} - f_0^{(1)})/N_0 n]\, h^{-v} \quad \text{and} \quad K_0 = 10^3,$$

where N_0 is an arbitrary positive parameter. Herein, $N_0 = 100$ was used. Finally $\nu = 4.5$, $\tau = 10^{-18}$, $\kappa = 4$ were also used. These parameter values were chosen after limited numerical trials (Ref. 11).

Table 2. Results for problems of dimensions greater than 4.

		FMINSEARCH		Modified N–M		
Function	*n*	FE	Minimum	FE	Minimum	MS(%)
Osborne 1	5	1098	5.46489e–5	1488	5.46489e–5	0.7
Brown almost linear	5	782	1.459e–18	648	1.087e–18	4.8
Biggs exp 6	6	1130	5.565565e–3	4390	1.161e–20	0.7
Extended Rosenbrock	6	7015	2.791e–17	3110	1.358e–14	0.9
Brown almost linear	7	1819	9.721e–18	1539	1.512e–17	3.3
Standard quadratic	8	1519	2.933e–16	1002	8.075e–17	6.4
Extended Rosenbrock	8	5958	6.664e–1†	5314	3.279e–17	2.0
Variably dimensional	8	3780	2.085e–16	2563	1.248e–15	3.6
Extended Powell	8	2513	5.132e-7†	7200	6.438e–24	0.7
Watson	9	3229	3.985e-3†	5256	1.39976e–6	1.1
Extended Rosenbrock	10	6684	9.72338†	7629	2.221e–16	0.8
Penalty I	10	5479	7.567e-5†	9200	7.08765e–5	2.6
Penalty II	10	6783	2.978e-4†	32768	2.93661e–4	0.4
Trigonometric	10	3105	2.79506e–5	2466	2.79506e–5	1.3
Osborne 2	11	4926	0.0401377	6416	0.0401377	1.0
Extended Powell	12	6607	5.525e–6†	20076	1.111e–20	0.3
Standard Quadratic	16	8543	7.704e–16	2352	1.415e–16	2.7
Standard Quadratic	24	100000†	0.5042†	4766	1.217e–15	1.1

The numerical results in Tables 1 and 2 show that the modified algorithm solved all problems whereas the standard Nelder–Mead algorithm failed on 9 of the test problems. The numerical performances of both algorithms were similar when the standard algorithm worked well. When the standard algorithm performed poorly or failed, the modified algorithm still worked well and located an accurate approximation to the solution with a number of function evaluations consistent with its performance on other problems. Eight of the problems on which the standard algorithm failed have dimension of at least 8; the ninth is the McKinnon problem.

The numerical results in Table 1 show that the modified method might be slightly slower than the standard method in low dimensions (2 to 4). However, McKinnon's example shows that the standard method is not robust, even in 2 dimensions.

Of the 9 problems of dimension greater than 4 on which the standard Nelder–Mead algorithm worked, the modified algorithm was faster on seven. This suggests that the modified algorithm is not only more robust than the standard Nelder–Mead algorithm, but it is also faster on problems with more than four variables. The results for the Biggs 6-dimensional exponential function for the two algorithms are not comparable as the standard algorithm found a stationary point whereas the modified method found a global minimizer.

The standard Nelder–Mead algorithm obtained final function values which may appear acceptable for Penalty function II ($n = 10$) and for the extended Powell function ($n = 12$), but in both cases halted at a point far from the actual minimum point. The modified algorithm found accurate approximations to the minimizers on these ill-conditioned problems. Interestingly, the modified algorithm executed standard Nelder–Mead steps over 99.5% of the time on both problems.

The results for the standard quadratic in 24 dimensions shows that the failure of the standard Nelder–Mead method is not caused by ill conditioning. The modified method succeeded on this problem and performed only a small number of modified steps in order to obtain convergence.

5. Conclusions

A convergent variant of the Nelder–Mead algorithm has been presented, and convergence on C^1 functions has been shown under mild conditions. The modifications made to the Nelder–Mead method are minor and only infrequently invoked. They do not impede the Nelder–Mead method when it is working well, and intercede only when sufficient progress is not being made.

The convergence properties of the modified algorithm have been verified in extensive numerical testing in Ref. 11 and summarized in the tables. This testing revealed several functions on which the standard Nelder–Mead algorithm fails, and showed that the failure of the standard Nelder–Mead algorithm is not a product of ill conditioning. The results also show that the modified method is somewhat faster on problems of dimension greater than about four.

References

1. Nelder, J. A., and Mead, R., *A Simplex Method for Function Minimization*, Computer Journal, Vol. 7, pp. 308–313, 1965.
2. Wright, M. H., *Direct Search Methods: Once Scorned, Now Respectable*, Numerical Analysis 1995, Edited by D. F. Griffiths and G. A. Watson, Pitman Research Notes in Mathematics, Addison-Wesley Longman, London, England, Vol. 344, 1996.
3. Mckinnon, K. I. M., *Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point*, SIAM Journal on Optimization, Vol. 9, pp 148–158, 1998.
4. Lagarius, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E., *Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions*, SIAM Journal on Optimization, Vol. 9, pp. 112–147, 1998.

5. KELLEY, C. T., *Detection and Remediation of Stagnation in the Nelder–Mead Algorithm Using a Sufficient Decrease Condition*, SIAM Journal on Optimization, Vol. 10, pp. 43–55, 2000.
6. RYKOV, A. S., *Simplex Algorithms for Unconstrained Minimization*, Problems of Control and Information Theory, Vol. 12, pp. 195–208, 1983.
7. DENNIS, J. E., and TORCZON, V., *Direct Search Methods on Parallel Machines*, SIAM Journal on Optimization, Vol. 1, pp. 448–474, 1991.
8. COOPE, I. D., and PRICE, C. J., *Frame Based Methods for Unconstrained Optimization*, Journal of Optimization Theory and Applications, Vol. 107, pp 261–274, 2000.
9. DAVIS, C., *Theory of Positive Linear Dependence*, American Journal of Mathematics, Vol. 76, pp. 733–746, 1954.
10. COOPE, I. D., and PRICE, C. J., *On the Convergence of Grid Based Methods for Unconstrained Minimization*, SIAM Journal on Optimization, Vol. 11, pp. 859–69, 2001.
11. BYATT, D., *Convergent Variants of the Nelder–Mead Algorithm*, University of Canterbury, Christchurch, New Zealand, MSc Thesis, 2000.
12. MORÉ, J. J., GARBOW, B. S., and HILLSTROM, K. E., *Testing Unconstrained Optimization Software*, ACM Transactions on Mathematical Software, Vol. 7, pp. 17–41, 1981.