A DIRECT SEARCH OPTIMIZATION METHOD THAT MODELS THE OBJECTIVE AND CONSTRAINT FUNCTIONS BY LINEAR INTERPOLATION

M.J.D. POWELL
Department of Applied Mathematics and Theoretical Physics,
University of Cambridge, Silver Street,
Cambridge CB3 9EW, England.

Abstract. An iterative algorithm is proposed for nonlinearly constrained optimization calculations when there are no derivatives. Each iteration forms linear approximations to the objective and constraint functions by interpolation at the vertices of a simplex and a trust region bound restricts each change to the variables. Thus a new vector of variables is calculated, which may replace one of the current vertices, either to improve the shape of the simplex or because it is the best vector that has been found so far, according to a merit function that gives attention to the greatest constraint violation. The trust region radius ρ is never increased, and it is reduced when the approximations of a well-conditioned simplex fail to yield an improvement to the variables, until ρ reaches a prescribed value that controls the final accuracy. Some convergence properties and several numerical results are given, but there are no more than 9 variables in these calculations because linear approximations can be highly inefficient. Nevertheless, the algorithm is easy to use for small numbers of variables.

Key words: Direct search, Linear interpolation, Nonlinear constraints, Optimization without derivatives

1. Introduction

John Dennis has provided the best description of a direct search optimization calculation that I have encountered. It is to find the deepest point of a muddy lake, given a boat and a plumb line, when there is a price to be paid for each sounding. A specification of an algorithm that is suitable for solving this problem would probably appeal to geometric intuition, and probably the procedure would require widely spaced measurements to be taken, in order to smooth out any high frequency variations in the depth of the lake. Experience has shown that many computer users find such algorithms attractive for a wide range of optimization calculations.

In particular, the method of Nelder and Mead (1965) is used in very many fields to calculate the least value of a function $F(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, when there are no constraints on the variables. Confirmation of this assertion can be found in the CMCI CompuMath Citation Index, more than 200 different applications being listed during the last 10 years. An iteration of this method is given the value of F at $n+1$ points, $\{\underline{x}^{(j)} : j = 0,1,\ldots,n\}$ say, where the points have to satisfy the nondegeneracy condition that the volume of their convex hull in $\mathcal{R}^n$ is positive. Let $\underline{x}^{(\ell)}$ be a vertex of the convex hull at which F is greatest, so ℓ is determined by the equation

$$F(\underline{x}^{(\ell)}) = \max\{F(\underline{x}^{(j)}) : j=0,1,\ldots,n\}. \qquad (1)$$

S. Gomez and J-P. Hennart (eds.), Advances in Optimization and Numerical Analysis, 51–67.

Then, because $\underline{x}^{(\ell)}$ is a vertex at which the objective function is worst, the iteration replaces $\underline{x}^{(\ell)}$ by the point

$$\underline{x}^{(\ell)}_{\rm new} = -\theta\,\underline{x}^{(\ell)} + (1+\theta)\,n^{-1}\sum_{j=0,\,j\neq\ell}^{n}\underline{x}^{(j)}, \tag{2}$$

where the "reflection coefficient" θ is a constant from the open interval $(0,1)$. Further, if $F(\underline{x}^{(\ell)}_{\rm new})$ is the least calculated function value so far, then a larger θ may be used instead. We see that formula (2) defines $\underline{x}^{(\ell)}_{\rm new}$ by extrapolation along the straight line that joins $\underline{x}^{(\ell)}$ to the mean value of the other n points. Therefore it is elementary that, if F is a nonconstant linear function, then $F(\underline{x}^{(\ell)}_{\rm new})$ is less than the average of the numbers $\{F(\underline{x}^{(j)}) : j = 0,1,\ldots,n,\ j\neq\ell\}$, so we expect an iteration to be successful at reducing F in the general case. If the iterations fail to make progress, however, and if an acceptably small value of the objective function has not been found, then the algorithm shrinks the current simplex with the vertices $\{\underline{x}^{(j)} : j = 0,1,\ldots,n\}$ before continuing the sequence of iterations. This kind of technique when $n=2$ might be suitable for seeking the deepest point of the muddy lake. In any case, the method is so straightforward to understand and to program for computer calculations that it is applied frequently.

The Nelder and Mead algorithm was developed from the method of Spendley, Hext and Himsworth (1962), in which every simplex is regular, this property being sustained by the value $\theta=1$ in formula (2). It was found, however, that the other choices of θ that have been mentioned provide much better efficiency by adapting the shape of the simplex to the curvature of the objective function automatically. Further, the Nelder and Mead algorithm is sometimes used to solve constrained problems by the simple expedient of replacing $F(\underline{x})$ by $+\infty$ if and only if $\underline{x}$ is infeasible. Then the simplex flattens itself so that it tends to be close to the active constraint boundaries. A disadvantage of this approach, however, is that an excellent change to the variables may be discarded because it gives an infinite value of the objective function. Therefore it is recommended by Subrahmanyam (1989) that, if a trial $\underline{x}$ is infeasible, then the setting of $F(\underline{x})$ to $+\infty$ should be postponed until the beginning of the next iteration, the usefulness of this technique being shown by numerical examples.

We take the view, however, that it may be possible to develop direct search methods that provide much better efficiency by taking advantage of the available details of the constraints. Therefore we address constrained optimization calculations that are expressed in the form

$$\left.\begin{array}{ll}\text{minimize} & F(\underline{x}),\quad \underline{x}\in\mathcal{R}^n\\ \text{subject to} & c_i(\underline{x})\geq 0,\quad i=1,2,\ldots,m\end{array}\right\}, \tag{3}$$

and we assume that the objective and constraint functions can be calculated for every $\underline{x}$, but there are no smoothness assumptions. We take from the Nelder and Mead method the idea of generating the next vector of variables from function values at the vertices $\{\underline{x}^{(j)} : j=0,1,\ldots,n\}$ of a nondegenerate simplex in $\mathcal{R}^n$. In this case there are unique linear functions, $\hat{F}$ and $\{\hat{c}_i : i=1,2,\ldots,m\}$ say, that interpolate F

and $\{c_i : i=1,2,\ldots,m\}$ at the vertices, and we approximate the calculation (3) by the linear programming problem

$$\left.\begin{array}{ll}\text{minimize} & \hat{F}(\underline{x}), \quad \underline{x}\in\mathcal{R}^n \\ \text{subject to} & \hat{c}_i(\underline{x}) \geq 0, \quad i=1,2,\ldots,m\end{array}\right\}. \tag{4}$$

On most iterations the problem (4) guides the changes to the variables, but some iterations give higher priority to modifying the shape of the simplex, in order that interpolation at its vertices is likely to yield good linear models of the objective and constraint functions. We are encouraged by the fact that the use of linear approximations to constraints is highly successful in variable metric algorithms (see Powell, 1978, for instance), because such approximations often take up much of the freedom in the variables in a suitable way. Thus it can happen that constrained calculations are easier than unconstrained ones, even when the given constraints are nonlinear.

The iterative use of expression (4) puts our method in the class of "sequential linear programming algorithms" that originated from the work of Griffith and Stewart (1961). A good discussion of the merits and disadvantages of this class is given in Section 6.1 of Himmelblau (1972), in the case when the gradients of the linear approximations are calculated analytically or are difference approximations to derivatives. It is possible, however, that our construction of these gradients is new, because we derive them by interpolation at the vertices of simplices that are analogous to the ones that occur in the Nelder and Mead algorithm.

Our procedure is specified in Section 2. We will find that it has the following properties. Changes to the variables are restricted by a trust region bound, which gives the user some control over the steps that are taken automatically and which responds satisfactorily to the fact that there may be no finite solution to the linear programming problem (4). The trust region radius remains constant until predicted improvements to the objective function and feasibility conditions fail to occur, although the simplex has a good shape. Then the trust region radius is reduced until it reaches a final value that has to be set by the user. The lengths of the trial steps are usually equal to the current trust region bound, in order that little damage is done to the early iterations by any high frequency fluctuations in the objective and constraint functions that are of small amplitude. The shapes of successive simplices can vary greatly, because most changes to the variables would satisfy any linear constraints, so there is often a tendency for the simplices to be squashed onto the constraint boundaries, which we remove explicitly. Indeed, as mentioned already, some iterations pick changes to the variables whose primary purpose is to improve the shape of the simplex. We employ a merit function of the form

$$\Phi(\underline{x}) = F(\underline{x}) + \mu\,[\max\{-c_i(\underline{x}) : i=1,2,\ldots,m\}]_+, \quad \underline{x}\in\mathcal{R}^n, \tag{5}$$

in order to compare the goodness of two different vectors of variables. Here μ is a parameter that is adjusted automatically, and the subscript "+" means that the expression in square brackets is replaced by zero if and only if its value is negative, so we have $\Phi(\underline{x}) = F(\underline{x})$ whenever $\underline{x}$ is feasible. We take the view that $\underline{x}\in\mathcal{R}^n$ is better than $\underline{y}\in\mathcal{R}^n$ if and only if the inequality $\Phi(\underline{x}) < \Phi(\underline{y})$ holds. Moreover, it is not difficult to implement the given rules for adjusting the variables.

Our knowledge of the convergence properties of the algorithm is the subject of Section 3. Then Section 4 discusses some of the details of Section 2 and presents a few numerical results. We conclude that the proposed method is suitable for a range of optimization calculations, but that the final accuracy is sometimes severely limited by the use of linear approximations to nonlinear functions. A Fortran implementation of the algorithm is available from the author at the e-mail address mjdp@amtp.cam.ac.uk.

2. The Algorithm

The algorithm includes several strategies, and is summarised in Figure 1. First we consider the vector of variables that is calculated from the linear programming problem (4) by the "Generate $\underline{x}^{(*)}$" box of the figure. This task requires the vertices $\{\underline{x}^{(j)} : j=0,1,\ldots,n\}$ of a nondegenerate simplex, a positive trust region radius ρ, and the current value of the parameter μ of the merit function (5). The vertices have already been ordered so that $\underline{x}^{(0)}$ is optimal, which means that the inequalities

$$\Phi(\underline{x}^{(0)}) \leq \Phi(\underline{x}^{(j)}), \quad j=1,2,\ldots,n, \tag{6}$$

are satisfied. Then the trust region condition on the new vector of variables, $\underline{x}^{(*)}$ say, is the bound

$$\|\underline{x}^{(*)}-\underline{x}^{(0)}\|_2 \leq \rho. \tag{7}$$

If possible, we let $\underline{x}^{(*)}$ minimize the linear approximation $\hat{F}(\underline{x}^{(*)})$ to the objective function subject to the inequality (7) and to the linear constraints

$$\hat{c}_i(\underline{x}^{(*)}) \geq 0, \quad i=1,2,\ldots,m, \tag{8}$$

of the problem (4), picking the $\underline{x}^{(*)}$ that gives the least value of $\|\underline{x}^{(*)}-\underline{x}^{(0)}\|_2$ if these conditions admit more than one $\underline{x}^{(*)}$. Alternatively, it can happen that the inequalities (7) and (8) are contradictory. Then we define $\underline{x}^{(*)}$ by minimizing the greatest of the constraint violations $\{-\hat{c}_i(\underline{x}^{(*)}) : i=1,2,\ldots,m\}$ subject to the trust region bound. Further, any remaining freedom in $\underline{x}^{(*)}$ is used to minimize $\hat{F}(\underline{x}^{(*)})$ and, if some freedom still remains, then we remove the ambiguity by again making $\|\underline{x}^{(*)}-\underline{x}^{(0)}\|_2$ as small as possible. The calculation of $\underline{x}^{(*)}$ has been implemented by imagining that ρ is increased continuously from zero to the current value. The sequence of values of $\underline{x}^{(*)}$ that would occur for this range of ρ is a continuous trajectory that is composed of straight line pieces. It is convenient to follow the trajectory from $\underline{x}^{(0)}$ to the required $\underline{x}^{(*)}$ by identifying and updating the active sets of linear constraints that define the linear pieces.

Next we describe the adjustment of μ, because it depends on the $\underline{x}^{(*)}$ that has just been specified. We set $\mu = 0$ initially, but in this case, when choosing the optimal vertex, it is assumed that μ is a tiny positive number whose value need not be specified. Later we take the view that it is unreasonable to expect the reduction $\Phi(\underline{x}^{(*)}) < \Phi(\underline{x}^{(0)})$ in the merit function (5) if the value of μ does not provide the condition $\hat{\Phi}(\underline{x}^{(*)}) < \hat{\Phi}(\underline{x}^{(0)})$, where $\hat{\Phi}$ is the approximation

$$\hat{\Phi}(\underline{x}) = \hat{F}(\underline{x}) + \mu\,[\max\{-\hat{c}_i(\underline{x}) : i=1,2,\ldots,m\}]_+, \quad \underline{x}\in\mathcal{R}^n, \tag{9}$$

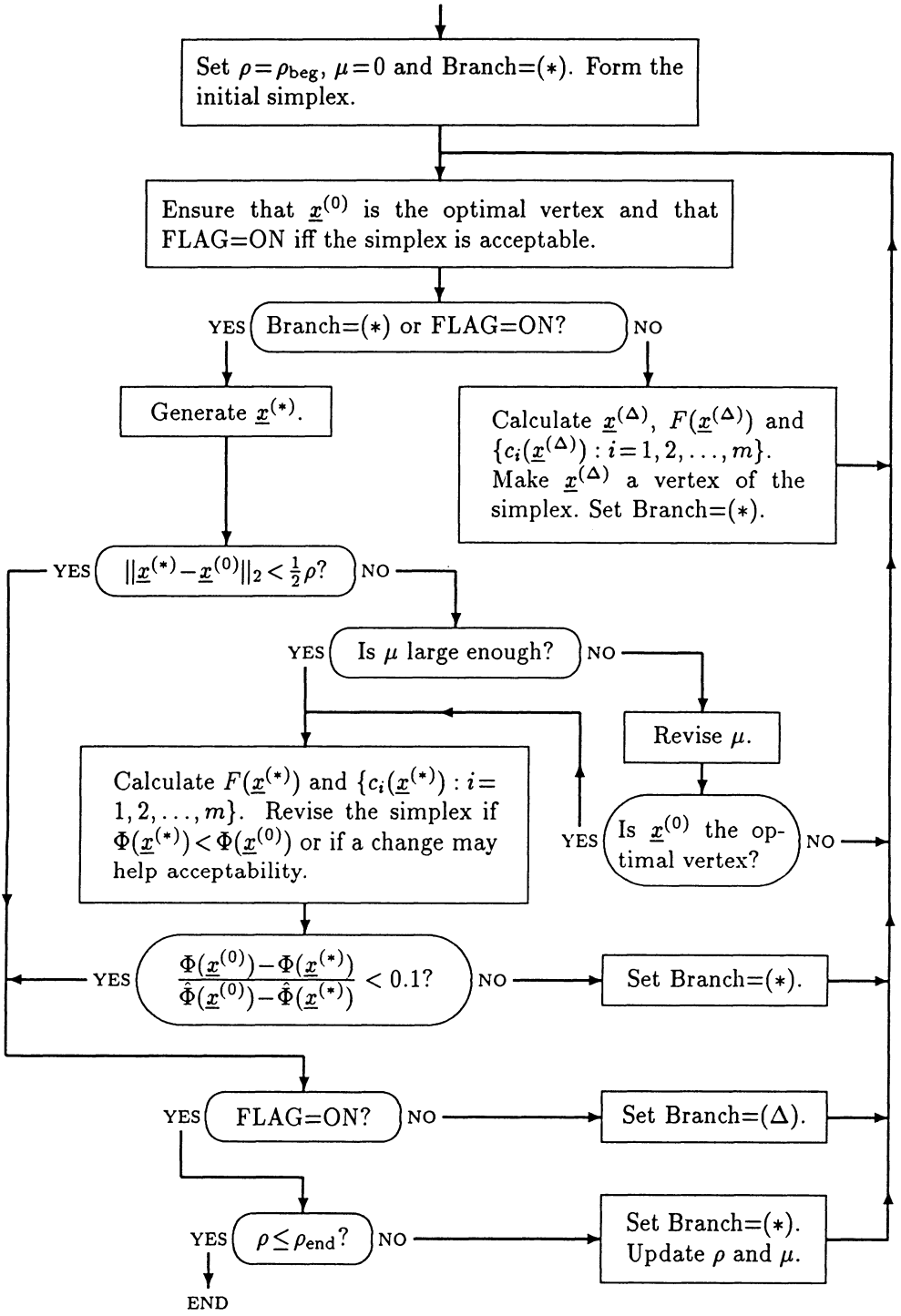

Figure 1: A summary of the algorithm

to Φ that is obtained by replacing F and $\{c_i : i = 1,2,\ldots,m\}$ by their current linear approximations. Therefore the following strategy is employed by the "Is μ large enough" and "Revise μ" boxes of the figure. Let $\bar{\mu}$ be the least nonnegative value of μ that would yield $\hat{\Phi}(\underline{x}^{(*)}) \leq \hat{\Phi}(\underline{x}^{(0)})$, the existence of $\bar{\mu}$ being an elementary consequence of the definition of $\underline{x}^{(*)}$. We leave μ unchanged if it satisfies $\mu \geq \frac{3}{2}\bar{\mu}$, but otherwise we increase μ to $2\bar{\mu}$. The choice of the factors $\frac{3}{2}$ and 2 in this technique was guided by guesswork and numerical calculations. Many similar factors will occur later, some of their values being discussed in Section 4.

A possible consequence of increasing μ is that condition (6) no longer holds. In this case the optimality of $\underline{x}^{(0)}$ is restored by exchanging two vertices of the simplex. Then the calculation of $\underline{x}^{(*)}$ and any necessary further adjustments to μ are repeated until $\underline{x}^{(0)}$ is optimal and the value of μ is acceptable. This procedure cannot cycle, because each change to $\underline{x}^{(0)}$ must reduce the value of the term $[\max\{-c_i(\underline{x}^{(0)}) : i = 1,2,\ldots,m\}]_+$.

The strategy for adjusting the trust region radius borrows from the Nelder and Mead algorithm the principle that one should continue to use the current ρ until the iterations fail to provide satisfactory reductions in the merit function, and then ρ should be decreased. We have the complication, however, that many different shapes of simplex can occur, so we also require the current simplex to be "acceptable" before decreasing ρ, in case unsuitable shapes have caused the linear programming problem (4) to be a very poor approximation to the main calculation, where "acceptable" is defined later. Therefore the bottom two boxes of Figure 1 are reached if we have an acceptable simplex, and if either the left hand side of inequality (7) is less than $\frac{1}{2}\rho$ or we find the condition

$$\Phi(\underline{x}^{(0)}) - \Phi(\underline{x}^{(*)}) < 0.1\left[\hat{\Phi}(\underline{x}^{(0)}) - \hat{\Phi}(\underline{x}^{(*)})\right], \tag{10}$$

which means that changing the variables from $\underline{x}^{(0)}$ to $\underline{x}^{(*)}$ fails to provide a tenth of the improvement in the merit function that is predicted by the approximation (9). It can be argued that in both of these cases there is a need for a reduction in the trust region radius.

The initial and final values of ρ, namely ρ_{beg} and ρ_{end}, are given by the user. We recommend that ρ_{beg} be a reasonable change to make to the variables for a coarse exploration of the calculation, while ρ_{end} should be approximately the required distance from the final vector of variables to the solution of the optimization problem. The following action is taken when the conditions for reducing ρ are satisfied. If $\rho \leq \rho_{\text{end}}$, then the iterative procedure is terminated, the final vector of variables being the current $\underline{x}^{(0)}$, except that $\underline{x}^{(*)}$ is preferred instead if $\Phi(\underline{x}^{(*)})$ is available and satisfies $\Phi(\underline{x}^{(*)}) < \Phi(\underline{x}^{(0)})$. Alternatively, when $\rho > \rho_{\text{end}}$, the trust region radius is set to the value

$$\rho_{\text{new}} = \begin{cases} \frac{1}{2}\rho, & \rho > 3\rho_{\text{end}} \\ \rho_{\text{end}}, & \rho \leq 3\rho_{\text{end}}. \end{cases} \tag{11}$$

Further, because it is shown in Section 3 that the merit function parameter μ can become very large, we estimate whether it would be advantageous to decrease μ when ρ is reduced. Specifically, we take the view that the i-th constraint is important to

the merit function if i is in the set

$$\mathcal{I} = \{i : c_i^{(\min)} < \tfrac{1}{2} c_i^{(\max)}\} \cap \{1, 2, \ldots, m\}, \tag{12}$$

where $c_i^{(\min)}$ and $c_i^{(\max)}$ are the least and greatest values of $c_i(\underline{x})$ at the vertices of the current simplex. Then μ is set to zero if $\mathcal{I}$ is empty, but otherwise we replace μ by the number

$$\left[\max_{j=0,1,\ldots,n} F(\underline{x}^{(j)}) - \min_{j=0,1,\ldots,n} F(\underline{x}^{(j)})\right] \Big/ \min\left\{[c_i^{(\max)}]_+ - c_i^{(\min)} : i \in \mathcal{I}\right\}, \tag{13}$$

provided that this change reduces μ. An explanation of the ratio (13) is given in Section 4.

A major difference between our algorithm and the method of Nelder and Mead is that we retain the current simplex when ρ is decreased. Therefore any immediate change to $\underline{x}^{(*)}$ is due to the new right hand side of inequality (7). Further, the current simplex will certainly be revised if it is not "acceptable" for the new value of ρ, the definition of "acceptability" being as follows.

For $j = 1, 2, \ldots, n$, let $\sigma^{(j)}$ be the Euclidean distance from the vertex $\underline{x}^{(j)}$ to the opposite face of the current simplex, and let $\eta^{(j)}$ be the length of the edge between $\underline{x}^{(j)}$ and $\underline{x}^{(0)}$. We say that the simplex is "acceptable" if and only if the inequalities

$$\left.\begin{aligned} \sigma^{(j)} &\geq \alpha\rho \\ \eta^{(j)} &\leq \beta\rho \end{aligned}\right\}, \quad j = 1, 2, \ldots, n, \tag{14}$$

hold, where α and β are constants that satisfy the conditions $0 < \alpha < 1 < \beta$. The software picks the values $\alpha = \frac{1}{4}$ and $\beta = 2.1$. Thus the lengths of the edges and the volume of an acceptable simplex are of magnitudes ρ and ρ^n respectively, so they are appropriate to the current trust region radius.

The initial simplex is constructed in the following way from ρ_{beg} and an initial vector of variables, which have to be provided by the user. We let $\underline{x}^{(0)}$ be the given vector and then we cycle through the indices $j = 1, 2, \ldots, n$. For each j we set $\underline{x}^{(j)} = \underline{x}^{(0)} + \rho_{\text{beg}} \underline{e}_j$, where $\underline{e}_j$ is the j-th coordinate vector. Further, $\underline{x}^{(j)}$ is exchanged with $\underline{x}^{(0)}$ before proceeding to the next value of j if and only if the condition $F(\underline{x}^{(j)}) < F(\underline{x}^{(0)})$ is satisfied. Thus $\underline{x}^{(0)}$ becomes the optimal vertex of the initial simplex, and we do not worry about the possibility that this simplex may not be "acceptable".

The vector $\underline{x}^{(*)}$ is not calculated on every iteration, because it is clear sometimes that priority should be given to trying to satisfy the conditions (14). Specifically, this priority is imposed if and only if the previous iteration would have reduced the current value of ρ if its simplex were "acceptable", the priority being initiated by the "Set Branch = Δ" box of Figure 1. In the next paragraph we define a vector $\underline{x}^{(\Delta)}$ that is an alternative new vector of variables that is chosen to improve acceptability. Therefore the current iteration calculates $\underline{x}^{(*)}$ instead of $\underline{x}^{(\Delta)}$ if and only if at least one of the following five conditions holds. (C1) There is no previous iteration. (C2) The previous iteration reduced ρ. (C3) The previous iteration calculated $\underline{x}^{(\Delta)}$. (C4) The previous iteration calculated $\underline{x}^{(*)}$ and reduced the merit function by at least

one tenth of the predicted reduction. (C5) The current simplex is "acceptable". The first four conditions are mutually exclusive, but usually condition (C3) or (C4) holds when (C5) is achieved. Exceptions can occur, however, because sometimes the previous iteration will have replaced a vertex of the simplex by a vector $\underline{x}^{(*)}$ that does not satisfy condition (10).

When none of the above conditions holds, the vector $\underline{x}^{(\Delta)}$ is defined as follows. If any of the numbers $\{\eta^{(j)} : j=1,2,\ldots,n\}$ of expression (14) is greater than $\beta\rho$, we let ℓ be the least integer from $[1,n]$ that satisfies the equation

$$\eta^{(\ell)} = \max\{\eta^{(j)} : j=1,2,\ldots,n\}. \tag{15}$$

Otherwise we obtain ℓ from the formula

$$\sigma^{(\ell)} = \min\{\sigma^{(j)} : j=1,2,\ldots,n\}, \tag{16}$$

the inequality $\sigma^{(\ell)} < \alpha\rho$ being implied by the failure of condition (C5). The iteration is going to replace the vertex $\underline{x}^{(\ell)}$ by $\underline{x}^{(\Delta)}$, so we require $\underline{x}^{(\Delta)}$ to be well away from the face of the simplex that is opposite the vertex $\underline{x}^{(\ell)}$. Therefore we let $\underline{v}^{(\ell)}$ be the vector of unit length that is perpendicular to this face, and we make the choice

$$\underline{x}^{(\Delta)} = \underline{x}^{(0)} \pm \gamma\rho\,\underline{v}^{(\ell)}, \tag{17}$$

where the $\pm$ sign is chosen to minimize the approximation $\hat{\Phi}(\underline{x}^{(\Delta)})$ to the new value of the merit function, and where γ is a constant from the interval $(\alpha,1)$ that is set to $\gamma=\frac{1}{2}$ by the software. Then the next iteration is given the simplex that has the vertices $\{\underline{x}^{(j)} : j=0,1,\ldots,n,\ j\neq\ell\}$ and $\underline{x}^{(\Delta)}$. The description of an iteration that calculates $\underline{x}^{(\Delta)}$ is complete.

Alternatively, when an iteration forms $\underline{x}^{(*)}$, we have to choose between three options, namely reducing ρ, or preserving ρ for another iteration that will calculate $\underline{x}^{(*)}$, or preserving ρ for another iteration that will give priority to improving the "acceptability" of the simplex. The rules that govern the choice have been specified already. Because we employ the test (10) if and only if $\underline{x}^{(*)}$ satisfies the condition

$$\|\underline{x}^{(*)}-\underline{x}^{(0)}\|_2 \geq \tfrac{1}{2}\rho, \tag{18}$$

the function values $F(\underline{x}^{(*)})$ and $\{c_i(\underline{x}^{(*)}) : i=1,2,\ldots,m\}$ are calculated only when inequality (18) holds. Then we may include these function values in future linear approximations by letting $\underline{x}^{(*)}$ replace one of the vertices $\{\underline{x}^{(j)} : j=1,2,\ldots,n\}$ of the current simplex, any change to the simplex being made in the following way.

The numbers $\{\bar{\sigma}^{(j)} : j=1,2,\ldots,n\}$ are found, where $\bar{\sigma}^{(j)}$ is defined to be the distance from $\underline{x}^{(*)}$ to the face of the current simplex that is opposite $\underline{x}^{(j)}$. These numbers are useful, because some elementary geometry shows that, if $\underline{x}^{(j)}$ is replaced by $\underline{x}^{(*)}$, then the volume of the simplex is multiplied by the factor $\bar{\sigma}^{(j)}/\sigma^{(j)}$. Therefore we take the view that the nonsingularity of the interpolation conditions will not be damaged if j is in the set

$$\mathcal{J} = \{j : \bar{\sigma}^{(j)} \geq \sigma^{(j)}\} \cup \{j : \bar{\sigma}^{(j)} \geq \alpha\rho\}, \tag{19}$$

where α is introduced in expression (14). Moreover, we expect the optimal vertex of the next iteration to be the point

$$\underline{\bar{x}}^{(0)} = \begin{cases} \underline{x}^{(*)}, & \Phi(\underline{x}^{(*)}) < \Phi(\underline{x}^{(0)}) \\ \underline{x}^{(0)}, & \Phi(\underline{x}^{(*)}) \geq \Phi(\underline{x}^{(0)}). \end{cases} \tag{20}$$

Therefore, if $\mathcal{J}$ is nonempty, we let ℓ be the least element of $\mathcal{J}$ that has the property

$$\|\underline{x}^{(\ell)} - \underline{\bar{x}}^{(0)}\|_2 = \max\{\|\underline{x}^{(j)} - \underline{\bar{x}}^{(0)}\|_2 : j \in \mathcal{J}\}. \tag{21}$$

Then the algorithm gives attention to the second of the acceptability requirements (14) by replacing $\underline{x}^{(\ell)}$ by $\underline{x}^{(*)}$ if we have the inequality $\|\underline{x}^{(\ell)} - \underline{\bar{x}}^{(0)}\|_2 > \delta\rho$, where δ is a constant satisfying $1 < \delta \leq \beta$ that is set to $\delta = 1.1$ by the software. Otherwise the new simplex is determined by the rule that its volume is to be maximized, subject to the condition that updating is mandatory when $\Phi(\underline{x}^{(*)})$ is less than $\Phi(\underline{x}^{(0)})$. In other words, if one (or both) of the conditions $\Phi(\underline{x}^{(*)}) < \Phi(\underline{x}^{(0)})$ and $\bar{\sigma}^{(\ell)} > \sigma^{(\ell)}$ holds, then $\underline{x}^{(\ell)}$ is replaced by $\underline{x}^{(*)}$, where now the integer ℓ is derived from the equation

$$\bar{\sigma}^{(\ell)}/\sigma^{(\ell)} = \max\{\bar{\sigma}^{(j)}/\sigma^{(j)} : j = 1, 2, \ldots, n\}. \tag{22}$$

Thus the simplex is revised by most iterations that calculate the objective and constraint functions at $\underline{x}^{(*)}$, the only exception being when, in addition to all the inequalities $\{\bar{\sigma}^{(j)} \leq \sigma^{(j)} : j = 1, 2, \ldots, n\}$ and $\Phi(\underline{x}^{(*)}) \geq \Phi(\underline{x}^{(0)})$, we find that the distance $\|\underline{x}^{(j)} - \underline{\bar{x}}^{(0)}\|_2$ is bounded above by $\delta\rho$ for every j in $\mathcal{J}$.

The description of our algorithm is now complete. Further details of the implementation are available in the Fortran listing that is mentioned at the end of Section 1.

3. Convergence Properties

Our knowledge of the convergence properties of the algorithm is slight. If we tried to establish a global convergence theorem by standard methods, then we would address the following four assertions. (A1) The parameter μ of the merit function (5) remains finite. (A2) The number of reductions in the trust region radius ρ is also finite. (A3) If μ and ρ remain constant, then any optimal vertex $\underline{x}^{(0)}$ cannot be retained for an infinite number of iterations. (A4) If μ and ρ remain constant, then the number of replacements of the optimal vertex is finite. These assertions would imply termination, because, for each ρ, the algorithm has the property that every change to the merit function parameter multiplies μ by at least the factor 4/3.

The following simple example suggests, however, that it would be difficult to make assumptions that provide assertion (A1) without ruling out some optimization calculations that the algorithm should solve. Let $n = 1$ and let the calculation be the problem

$$\left.\begin{array}{ll} \text{minimize} & F(x) = -|x-3|, \quad x \in \mathcal{R}, \\ \text{subject to} & c(x) = \frac{1}{4} - |x| \geq 0 \end{array}\right\}, \tag{23}$$

whose solution is $x = -\frac{1}{4}$. Further, let x and ρ satisfy $x > 3$ and $\rho = 1$ initially. Then on the first iteration the approximation (9) to the merit function is the expression

$$\hat{\Phi}(x) = -x + 3 + \mu[x - \tfrac{1}{4}]_+, \quad x \in \mathcal{R}, \tag{24}$$

so the algorithm sets the first value of the merit function parameter to $\mu=2$. It is now straightforward to deduce the action of an iteration if the optimal vertex of the current simplex satisfies $x^{(0)}>\frac{1}{4}$. Specifically, one can show by induction that $x^{(1)}$ and $x^{(*)}$ have the values $x^{(0)}+\rho$ and $x^{(0)}-\rho$ respectively, that $\mu=2$ is preserved, and that the inequality $\Phi(x^{(*)})<\Phi(x^{(0)})$ holds, which causes the next simplex to have the vertices $x^{(0)}-\rho$ and $x^{(0)}$. Further, the trust region radius is not reduced until inequality (10) is satisfied, which requires the condition $x^{(0)}<0.325$, mainly because $\rho=1$, $\mu=2$ and $0.325\leq x^{(0)}\leq 3$ imply the relation

$$\begin{aligned}\Phi(x^{(0)})-\Phi(x^{(*)}) &= [\,F(x^{(0)})-F(x^{(*)})\,]+2\,\{-c(x^{(0)})-[-c(x^{(0)}-1)\,]_+\}\\ &= 1+2\,\{x^{(0)}-\tfrac{1}{4}-[\,|x^{(0)}-1|-\tfrac{1}{4}\,]_+\}\\ &\geq 1+2\,\{0.325-\tfrac{1}{4}-[\,0.675-\tfrac{1}{4}\,]\}=0.3\\ &\geq 0.1\,[\,\hat{\Phi}(x^{(0)})-\hat{\Phi}(x^{(*)})\,]. \qquad (25)\end{aligned}$$

Therefore an iteration can begin with $x^{(0)}=\frac{1}{2}-\epsilon$, and can generate a simplex with the vertices $x^{(0)}=-\frac{1}{2}-\epsilon$ and $x^{(1)}=\frac{1}{2}-\epsilon$, where ϵ is a very small positive number. In this case the approximation to the merit function on the next iteration has the form

$$\hat{\Phi}(x)=x-3+\mu\,[\,\tfrac{1}{4}-2\epsilon x-2\epsilon^2\,]_+, \quad x\in\mathcal{R}. \qquad (26)$$

It now follows that we decrease the infeasibility of the linear approximation to the constraint function by *increasing* x. Further, we see that this change to x provides the required reduction in $\hat{\Phi}$ only if μ is made larger than $1/(2\epsilon)$.

Hence, even in the simple case (23), there is no *a priori* upper bound on μ. The difficulty is that the values of a function at the vertices of a simplex can be tiny perturbations of a constant, although the function itself varies substantially for most changes to the variables. Thus the linear approximations that are made by the algorithm may be highly misleading. On the other hand, the numerical results of the next section show that the given algorithm solves a range of nonlinear optimization calculations fairly well. It therefore seems futile to impose restrictions that would allow assertion (A1) to be established analytically. Moreover, a proof of assertion (A4) may be even more elusive, so we ignore this challenge too, at least in the case when the precision of the computer arithmetic is infinite.

On the other hand, the validity of assertion (A2) is an easy consequence of formula (11) and the condition $\rho_{\text{end}}>0$. Further, the following argument establishes that assertion (A3) is also true.

Lemma 1. *Any sequence of iterations of the given algorithm that does not change either ρ or the optimal vertex $\underline{x}^{(0)}$ is finite.*

Proof: We recall that, if a simplex is acceptable at the beginning of an iteration, and if the operations of the iteration do not alter the optimal vertex, then either a reduction in ρ or termination occurs at the end of the iteration. Therefore it is sufficient to prove that the conditions (14) are satisfied after a finite number of iterations of the given sequence.

Now Figure 1 shows that an iteration would change $\underline{x}^{(0)}$ if the previous iteration had calculated an $\underline{x}^{(*)}$ that satisfied $\Phi(\underline{x}^{(*)})<\Phi(\underline{x}^{(0)})$. Therefore alternate iterations in the sequence revise the simplex by replacing one of the vertices

$\{\underline{x}^{(j)} : j=1,2,\ldots,n\}$ by the point (17). Further, if the set $\{j : \|\underline{x}^{(j)}-\underline{x}^{(0)}\|_2 > \beta\rho\}$ is nonempty, then the replacement reduces the number of elements in this set by one. Further, none of the other changes to the simplex can increase the number of elements in this set, because every $\underline{x}^{(*)}$ satisfies inequality (7). It follows that the second of the conditions (14) holds for every j after at most n iterations of the given sequence have used formula (17) to introduce a new vertex, which happens on alternate iterations.

The conditions $\|\underline{x}^{(*)}-\underline{x}^{(0)}\|_2 \leq \rho$ and $\|\underline{x}^{(\Delta)}-\underline{x}^{(0)}\|_2 \leq \rho$ also imply that the given iterations never add any elements to the set $\{j : \|\underline{x}^{(j)}-\underline{x}^{(0)}\|_2 > \delta\rho\}$, where δ is introduced soon after equation (21). Therefore this set does not change after a finite number of iterations. It follows from some details in Section 2 that eventually none of the iterations under consideration reduces the volume of the current simplex. Further, the alternate iterations that replace a vertex $\underline{x}^{(\ell)}$ by the point (17) multiply the volume by a factor that exceeds γ/α. Thus, if the lemma were false, the volume would become unbounded, which would contradict the second of the conditions (14). The proof is complete. ■

The formula (17) that defines $\underline{x}^{(\Delta)}$ also has the following interesting property.

Lemma 2. *Let an iteration replace the vertex $\underline{x}^{(\ell)}$ of the current simplex by $\underline{x}^{(\Delta)}$, and let the distances from the vertices (excluding $\underline{x}^{(0)}$) to their opposite faces in the old and new simplices be $\{\sigma_{old}^{(j)} : j=1,2,\ldots,n\}$ and $\{\sigma_{new}^{(j)} : j=1,2,\ldots,n\}$ respectively. Then, in addition to the equation $\sigma_{new}^{(\ell)}=\gamma\rho$, we have the inequalities*

$$\sigma_{new}^{(j)} \geq \sigma_{old}^{(j)}, \quad j=1,2,\ldots,n, \; j\neq\ell. \tag{27}$$

Proof: The equation $\sigma_{\rm new}^{(\ell)} = \gamma\rho$ follows from the choice (17) and the definition of $\underline{v}^{(\ell)}$. Further, this definition implies the relation

$$(\underline{x}^{(\Delta)}-\underline{x}^{(0)}, \underline{x}^{(i)}-\underline{x}^{(0)}) = 0, \quad i=1,2,\ldots,n, \; i\neq\ell, \tag{28}$$

the left hand side being a scalar product. Let j be any integer from $[1,n]$ that is different from ℓ. Then the closest point to $\underline{x}^{(j)}$ in the opposite face of the new simplex can be expressed in the form

$$\underline{x}^{(0)} + \sum_{\substack{i=1\\ i\neq j,\ell}}^{n} \theta_i(\underline{x}^{(i)}-\underline{x}^{(0)}) + \theta_\ell(\underline{x}^{(\Delta)}-\underline{x}^{(0)}) \tag{29}$$

for some multipliers $\{\theta_i : i=1,2,\ldots,n, \; i\neq j\}$, which gives the bound

$$[\sigma_{\rm new}^{(j)}]^2 = \|(\underline{x}^{(j)}-\underline{x}^{(0)}) - \sum_{\substack{i=1\\ i\neq j,\ell}}^{n} \theta_i(\underline{x}^{(i)}-\underline{x}^{(0)}) - \theta_\ell(\underline{x}^{(\Delta)}-\underline{x}^{(0)})\|_2^2$$

$$\geq \|(\underline{x}^{(j)}-\underline{x}^{(0)}) - \sum_{\substack{i=1\\ i\neq j,\ell}}^{n} \theta_i(\underline{x}^{(i)}-\underline{x}^{(0)})\|_2^2, \tag{30}$$

where the last line depends on all the equations (28). We see that this right hand side is the square of the distance from $\underline{x}^{(j)}$ to a point in the opposite face of the *old* simplex. Thus the required inequality (27) is a consequence of the definition of $\sigma_{\rm old}^{(j)}$. ■

We complete this section by establishing termination under the crude assumption that, due to the limited precision of computer arithmetic, only a finite number of different values of the functions F and $\{c_i : i=1,2,\ldots,m\}$ can occur. In this case assertion (A4) of the opening paragraph of this section holds, because every change to the optimal vertex has to provide a strict reduction in the merit function, which now cannot happen infinitely often. Further, we have noted already that assertions (A2) and (A3) are valid. Therefore it is straightforward to deduce termination if the number of changes to μ is finite, which should be another consequence of the computer arithmetic. Nevertheless, we will respond to the challenge of allowing μ to be any nonnegative real number.

Since every increase in μ is by at least the factor 4/3, and since we have already treated the case when μ is changed finitely often, we can assume without loss of generality that μ is greater than any positive constant Ω. We set $\Omega = 1$ if all calculable values of F are the same, or if $m = 0$, or if all calculable values of the constraint violation function

$$\Gamma(\underline{x}) = [\max\{-c_i(\underline{x}) : i=1,2,\ldots,m\}]_+, \quad \underline{x}\in\mathcal{R}^n, \tag{31}$$

are equal. Otherwise, we let Ω be the greatest change that can occur in F divided by the smallest positive change that can occur in Γ, this ratio being well-defined because of the crude assumption in the previous paragraph. Then, if μ exceeds Ω, the reduction $\Phi(\underline{x}^{(*)}) < \Phi(\underline{x}^{(0)})$ is found in practice if and only if we have either $\Gamma(\underline{x}^{(*)}) < \Gamma(\underline{x}^{(0)})$ or $\Gamma(\underline{x}^{(*)}) = \Gamma(\underline{x}^{(0)})$ and $F(\underline{x}^{(*)}) < F(\underline{x}^{(0)})$, the value of F being immaterial in the former case due to the choice of Ω. Thus the conditions that define an optimal vertex become independent of μ when the merit function parameter is sufficiently large. Hence the number of reductions in the merit function that can be achieved by updating the optimal vertex is finite. Therefore assertion (A4) is valid even if the condition that μ remains constant is replaced by the requirement that μ be sufficiently large, so there is no need for assertion (A1). It follows that the presumed finite precision of the computer arithmetic implies termination. Further, one could take advantage of this argument in practice by forcing a coarse discretization on the values of F and Γ when testing whether a vertex is optimal.

4. Discussion and Numerical Results

The main influence on the design of the algorithm was the belief that linear approximations to nonlinear constraints are highly useful. Further, because one can express a general objective function as a linear objective function subject to an inequality constraint by introducing a slack variable, it is consistent to make a linear approximation to the objective function too. We picked the easiest way of defining these linear approximations, namely interpolation at the vertices of a simplex. It was

then necessary to impose a trust region bound in order that each linear programming subproblem has a finite solution. The shape of the trust region was chosen to be spherical, in order to preserve rotational symmetry when one takes a geometrical view of the steps of the algorithm, and the changes to the trust region radius are monotonic, in order to avoid all the careful attention to details that would arise from a strategy that allowed ρ to increase. We picked a merit function that employs the greatest constraint violation because this is often what users want. Here we have in mind that 1000 constraint violations of 10^{-6} are usually preferable to a single constraint violation of 10^{-3}, but a 1-norm merit function would not distinguish between these two cases. On the other hand, a smooth merit function would provide so many advantages that this subject deserves some research.

The example in the second paragraph of Section 4 is worrying because of the severe loss of efficiency that can occur if μ becomes huge. Indeed, consider the simple case when $n=2$ and $m=1$, when F and c_1 are the linear functions

$$F(\underline{x}) = x_2, \quad c_1(\underline{x}) = -x_1, \quad \underline{x} \in \mathcal{R}^2, \tag{32}$$

when $\rho = 1$, and when the vertices of the current simplex have the components $\underline{x}^{(0)} = (0,1)$, $\underline{x}^{(1)} = (\epsilon, 0)$ and $\underline{x}^{(2)} = (1,1)$, where the number ϵ is very small and positive. If μ satisfied the condition $\mu > 1/\epsilon$, then $\underline{x}^{(0)}$ would be the optimal vertex and $\underline{x}^{(*)}$ would have the coordinates $(0,0)$. Further, the current iteration of our algorithm would update the simplex by replacing $\underline{x}^{(1)}$ by $\underline{x}^{(*)}$, although this change to the variables is tiny. Instead, therefore, it might be better to let $\underline{x}^{(*)}$ solve the linear programming problem (4) subject to the trust region bound $\|\underline{x}^{(*)} - \underline{x}^{(1)}\|_2 \leq \rho$, which would give the trial vector $\underline{x}^{(*)} = (0, -[1-\epsilon^2]^{1/2})$. This choice, however, is also unsatisfactory, because, if c_1 were replaced by a mildly nonlinear function that satisfied $c_1(\underline{x}) = -x_1$ at the vertices of the current simplex, then it is likely that we would find the increase $\Phi(\underline{x}^{(*)}) > \Phi(\underline{x}^{(0)})$ in the merit function. Thus the large value of μ would cause a reduction in ρ, although there is no evidence that the variables are nearly optimal.

It is difficult, however, to devise techniques that increase and decrease a merit function parameter and that are guaranteed not to cycle in general optimization calculations. Therefore we include the partial remedy of only reducing μ if it seems to be too large when ρ is decreased, which happens finitely often. The procedure that is given in Section 2 is derived from the magnitudes of the two terms on the right hand side of the definition (5) of the merit function. Clearly the numerator of expression (13) is approximately a typical change to F, and it is reasonable to exclude from consideration the constraints whose indices are not in $\mathcal{I}$. Further, the term $[c_i^{(\max)}]_+ - c_i^{(\min)}$ is a typical change to the i-th constraint function if $c_i^{(\max)}$ is nonnegative and otherwise it is a change that has to be made to achieve feasibility, but the choice of "min" rather than "max" in the denominator of the ratio (13) is debatable. Here we take the view that, if μ is decreased, then we want each of the relevant constraints to make a contribution to the merit function that is not much less than a typical change to F. It is therefore helpful if the user scales the constraint functions so that they have similar magnitudes.

The values of the parameters α, β, γ and δ that are mentioned in Section 2 were guided by some numerical tests that did not provide clear answers. We picked $\alpha = \frac{1}{4}$

for the first of the acceptability conditions (14), because $\alpha=\frac{1}{8}$ gave relatively poor results in the numerical experiments, and because the proof of Lemma 1 suggests that there should be enough scope for γ/α to be substantially greater than one. There was little difference in practice between $\gamma=\frac{1}{2}$ and $\gamma=1$, so we preferred the smaller value, because formula (17) tends to move $\underline{x}^{(\Delta)}$ away from any subspace that contains the most successful vectors of variables. We require $\beta>1$ in expression (14). Further, the condition $\beta\geq 2$ is advantageous, because otherwise it would be usual for every edge of the current simplex to be too long immediately after halving ρ, and then "acceptability" might demand the updating of all the suboptimal vertices before the next reduction in ρ. On the other hand, an example in the penultimate paragraph of this section will show that too large a value of β can be highly inefficient. Therefore $\beta=2.1$ was selected. The parameter δ that occurs soon after equation (21) should certainly satisfy $1<\delta\leq\beta$. We picked $\delta=1.1$, because a value that is only a little larger than one helps the newly calculated function values at $\underline{x}^{(*)}$ to be included in the linear approximations of the next iteration.

The algorithm was applied to the following ten problems using single precision arithmetic on a Sparc 2 workstation. In every case the initial trust region radius was $\rho=\frac{1}{2}$ and all components of the initial vector of variables were set to 1. Firstly, some easy tests of the effects of nonlinearity were made by the calculations

$$\text{minimize } F(\underline{x})=10(x_1+1)^2+x_2^2, \quad \underline{x}\in\mathcal{R}^2, \tag{A}$$

$$\left.\begin{array}{l}\text{minimize } F(\underline{x})=x_1x_2, \quad \underline{x}\in\mathcal{R}^2,\\ \text{subject to } x_1^2+x_2^2\leq 1\end{array}\right\}, \tag{B}$$

and

$$\left.\begin{array}{l}\text{minimize } F(\underline{x})=x_1x_2x_3, \quad \underline{x}\in\mathcal{R}^3,\\ \text{subject to } x_1^2+2x_2^2+3x_3^2\leq 1\end{array}\right\}, \tag{C}$$

there being no constraints in problem (A). The next two calculations were also unconstrained, being the mild versions

$$\text{minimize } F(\underline{x})=(x_1^2-x_2)^2+(1+x_1)^2, \quad \underline{x}\in\mathcal{R}^2, \tag{D}$$

and

$$\text{minimize } F(\underline{x})=10(x_1^2-x_2)^2+(1+x_1)^2, \quad \underline{x}\in\mathcal{R}^2, \tag{E}$$

of the well-known problem of Rosenbrock (1960). We took the constrained calculations

$$\left.\begin{array}{l}\text{minimize } F(\underline{x})=-x_1-x_2, \quad \underline{x}\in\mathcal{R}^2,\\ \text{subject to } x_1^2\leq x_2 \text{ and to } x_1^2+x_2^2\leq 1\end{array}\right\}, \tag{F}$$

and

$$\left.\begin{array}{l}\text{minimize } F(\underline{x})=x_3, \quad \underline{x}\in\mathcal{R}^3,\\ \text{subject to } 5x_1-x_2+x_3\geq 0, \quad -5x_1-x_2+x_3\geq 0\\ \text{and to } \quad x_1^2+x_2^2+4x_2\leq x_3\end{array}\right\} \tag{G}$$

from Fletcher's (1987) book. Finally, problems (H)–(J) are the ones with the numbers 43, 100 and 108 in Hock and Schittkowski (1980), so they have 4, 7 and 9

Problem number	Function values	Final $F(\underline{x})$	Final $\Gamma(\underline{x})$	Final $\|\|\underline{x}-\underline{x}^{(\mathrm{opt})}\|\|_2$
(A)	37	1.8×10^{-5}	0	3.3×10^{-3}
(B)	37	−0.5000	2.0×10^{-6}	1.3×10^{-3}
(C)	45	−0.0786	4.7×10^{-6}	1.4×10^{-3}
(D)	100	3.1×10^{-5}	0	1.3×10^{-2}
(E)	347	4.0×10^{-3}	0	1.4×10^{-1}
(F)	30	−1.4142	3.0×10^{-6}	1.2×10^{-4}
(G)	29	−3.0000	1.3×10^{-4}	5.9×10^{-5}
(H)	74	−44.0000	2.9×10^{-6}	1.4×10^{-3}
(I)	198	680.6303	5.7×10^{-5}	5.9×10^{-3}
(J)	143	−0.8660	1.0×10^{-6}	8.9×10^{-4}

Table 1: Problems (A)–(J) when $\rho_{\mathrm{end}}=10^{-3}$

Problem number	Function values	Final $F(\underline{x})$	Final $\Gamma(\underline{x})$	Final $\|\|\underline{x}-\underline{x}^{(\mathrm{opt})}\|\|_2$
(A)	65	1.2×10^{-7}	0	2.8×10^{-4}
(B)	44	−0.5000	6.0×10^{-8}	6.1×10^{-5}
(C)	60	−0.0786	0	9.2×10^{-6}
(D)	173	6.4×10^{-7}	0	1.7×10^{-3}
(E)	698	9.5×10^{-5}	0	2.2×10^{-2}
(F)	41	−1.4142	1.5×10^{-7}	4.6×10^{-5}
(G)	33	−3.0000	0	2.4×10^{-8}
(H)	87	−44.0000	2.2×10^{-6}	1.2×10^{-3}
(I)	212	680.6303	0	5.3×10^{-3}
(J)	173	−0.8660	1.2×10^{-7}	9.5×10^{-5}

Table 2: Problems (A)–(J) when $\rho_{\mathrm{end}}=10^{-4}$

variables respectively. This last calculation is intended to maximize the area of a hexagon of unit diameter, but it provides the value $\frac{1}{2}\sqrt{3}\approx0.866$, although the area of a circle of unit diameter is only $\frac{1}{4}\pi\approx0.785$. The reason for the incorrect result is that the formulation of the problem allows one circuit of the hexagon to degenerate to two circuits of an equilateral triangle, the area of the triangle being counted twice. This example is also interesting because it includes some local minima where the area has the value 0.5.

The results are presented in Tables 1 and 2, the difference between the tables being that the final trust region radii are 10^{-3} and 10^{-4}, respectively. The columns of each table display the problem number, the number of calculations of F and $\{c_i : i=1,2,\ldots,m\}$, the final value of the objective function, the final value of the maximum constraint violation (31), and the final value of $\|\underline{x}-\underline{x}^{(\mathrm{opt})}\|_2$, which is the

Euclidean distance from the calculated vector of variables to a true solution. We see that the amount of computation is not excessive for an easy-to-use algorithm that does not require any derivatives, although the Nelder and Mead (1965) method is sometimes much more efficient when there are no constraints, because it adapts the shape of the simplex to the curvature of the objective function. Further, a comparison of the two tables shows that the accuracy in the calculated variables can be controlled approximately by the final value of ρ, but some of the entries in the last column of the tables are rather large.

The reason for these large entries is that linear approximations to nonlinear functions are often misleading near a solution to an optimization calculation. For example, we consider the unconstrained minimization of the quadratic function

$$F(\underline{x}) = x_1^2 + Mx_2^2, \quad \underline{x} \in \mathcal{R}^2, \tag{33}$$

where M is a positive constant, and we let θ be a positive parameter. Then the points

$$\underline{x}^{(0)} = \begin{pmatrix} \theta \\ 0 \end{pmatrix}, \quad \underline{x}^{(1)} = \begin{pmatrix} \theta - \frac{1}{4}\rho \\ (\frac{15}{16})^{1/2}\rho \end{pmatrix} \quad \text{and} \quad \underline{x}^{(2)} = \begin{pmatrix} \theta - \frac{1}{4}\rho \\ -(\frac{15}{16})^{1/2}\rho \end{pmatrix} \tag{34}$$

provide an acceptable simplex, and we have the function values

$$F(\underline{x}^{(0)}) = \theta^2, \quad F(\underline{x}^{(1)}) = F(\underline{x}^{(2)}) = \theta^2 - \tfrac{1}{2}\theta\rho + (\tfrac{1}{16} + \tfrac{15}{16}M)\rho^2. \tag{35}$$

Thus $\underline{x}^{(0)}$ is an optimal vertex if the inequality

$$\theta < (\tfrac{1}{8} + \tfrac{15}{8}M)\rho \tag{36}$$

holds. Further, the algorithm makes the linear approximation

$$\hat{F}(\underline{x}) = \theta^2 + 2[\theta - (\tfrac{1}{8} + \tfrac{15}{8}M)\rho](x_1 - \theta), \quad \underline{x} \in \mathcal{R}^2, \tag{37}$$

to the objective function, so $\underline{x}^{(*)}$ has the components $(\theta + \rho, 0)$ when $\underline{x}^{(0)}$ is the optimal vertex. In this case, unfortunately, we find the inequality $F(\underline{x}^{(*)}) > F(\underline{x}^{(0)})$, so either termination occurs or ρ is reduced, although the distance from $\underline{x}^{(0)}$ to the true solution can exceed $\frac{15}{8}M\rho$. Therefore it is possible for large errors to occur in the calculated solution when the second derivative matrix $\nabla^2 F$ is positive definite and only mildly ill-conditioned. Further, if ρ is halved when $\theta = \frac{15}{8}M\rho$, and if the final variables are nearly optimal, then at least another $\frac{15}{4}M$ iterations are needed. In the light of this example, the numerical results of Tables 1 and 2 are as good as can be expected.

Further, this example suggests that it may be possible to develop a very useful new algorithm by using quadratic instead of linear approximations to the objective and constraint functions. Then each simplex of $(n+1)$ points would have to be replaced by a suitable set of $\frac{1}{2}(n+1)(n+2)$ points, in order that the coefficients of the new approximations can be calculated by interpolation. The author intends to investigate this approach, at least in the unconstrained case, because such research should yield another algorithm that is more suitable for the minimization of noisy functions than the usual methods that employ difference approximations to derivatives.

Acknowledgements

This work began when David Ingram of Westland Helicopters asked for my advice on the technique of extending a version of the Nelder and Mead (1965) method to constrained optimization by forcing the objective function to be infinite at infeasible points. Thus he provided the motivation for the given algorithm. I am also very grateful to the Mathematics Department of the University of Canterbury, New Zealand, because it provided excellent facilities for this research while I was there on sabbatical leave. Further, I offer my thanks to a referee who suggested several improvements to the presentation.

References

R. Fletcher (1987), *Practical Methods of Optimization*, John Wiley and Sons (Chichester).

R.E. Griffith and R.A. Stewart (1961), "A nonlinear programming technique for the optimization of continuous processing systems", *Management Sci.*, Vol. 7, pp. 379–392.

D.M. Himmelblau (1972), *Applied Nonlinear Programming*, McGraw-Hill (New York).

W. Hock and K. Schittkowski (1980), *Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems 187*, Springer-Verlag (Berlin).

J.A. Nelder and R. Mead (1965), "A simplex method for function minimization", *Comput. J.*, Vol. 7, pp. 308–313.

M.J.D. Powell (1978), "A fast algorithm for nonlinearly constrained optimization calculations", in *Numerical Analysis, Dundee 1977, Lecture Notes in Mathematics 630*, ed. G.A. Watson, Springer-Verlag (Berlin), pp. 144–157.

H.H. Rosenbrock (1960), "An automatic method for finding the greatest or least value of a function", *Comput. J.*, Vol. 3, pp. 175–184.

W. Spendley, G.R. Hext and F.R. Himsworth (1962), "Sequential application of simplex designs in optimisation and evolutionary operation", *Technometrics*, Vol. 4, pp. 441–461.

M.B. Subrahmanyam (1989), "An extension of the simplex method to constrained optimization", *J. Optim. Theory Appl.*, Vol. 62, pp. 311–319.