A View of Algorithms for Optimization without Derivatives[1]

M.J.D. Powell

Abstract: Let the least value of the function $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, be required, where $n \geq 2$. If the gradient $\underline{\nabla} F$ is available, then one can tell whether search directions are downhill, and first order conditions help to identify the solution. It seems in practice, however, that the vast majority of unconstrained calculations do not employ any derivatives. A view of this situation is given, attention being restricted to methods that are suitable for noisy functions, and that change the variables in ways that are not random. Particular topics include the McKinnon (1998) example of failure of the Nelder and Mead (1965) simplex method, some convergence properties of pattern search algorithms, and my own recent research on using quadratic models of the objective function. We find that least values of functions of more than 100 variables can be calculated.

Department of Applied Mathematics and Theoretical Physics,
Centre for Mathematical Sciences,
Wilberforce Road,
Cambridge CB3 0WA,
England.

April, 2007

[1]Presented as a William Benter Distinguished Lecture at the City University of Hong Kong.

1. Introduction

Many optimization problems occur naturally. A soap film on a wire frame, for example, takes the shape that has least area, and an atomic system may decay into a state that has least energy. They also arise from best ways of achieving objectives, such as changing the path of a space capsule to a required new orbit by firing rocket motors in the way that consumes least fuel, or designing the cheapest suspension bridge that will carry prescribed loads for a reasonable range of weather conditions. Medical applications include the treatment of malignant tumours by radiation, when the required total dose is collected from several sources outside the patient's body, the amount from each source being chosen to minimize the damage to healthy tissue. Other examples arise from data fitting, from the design of experiments and from financial mathematics, for instance.

The development of algorithms for optimization has been my main field of research for 45 years, but I have given hardly any attention to applications. It is very helpful, however, to try to solve some particular problems well, in order to receive guidance from numerical results, and in order not to be misled from efficiency in practice by a desire to prove convergence theorems. My particular problems were usually contrived, and often I let the objective function be a quadratic polynomial in the variables. Indeed, I have constructed several useful algorithms by seeking good performance in this case in a way that allows the objective function to be general.

I started to write computer programs in Fortran at Harwell in 1962. The optimization software that I developed there, until I left in 1976, was made available for general use by inclusion in the Harwell Subroutine Library (HSL). Occasionally people elsewhere would hear about these contributions from publications, from conferences and from contacts with other people. The procedures for obtaining the software were unrestricted, and I was always delighted to hear when my work had been useful. The change of orbit calculation is mentioned above, because I was told after the event that the DFP algorithm (Fletcher and Powell, 1963) had assisted the moon landings of the Apollo 11 Space Mission.

I made some more contributions to HSL after moving to Cambridge in 1976 and also I became a consultant for IMSL. One product they received from me was the TOLMIN package (Powell, 1989) for optimization subject to linear constraints, which requires first derivatives of the objective function. Their customers, however, prefer methods that are without derivatives, so IMSL forced my software to employ difference approximations instead, although this modification may lose much accuracy. I was not happy. The IMSL point of view was receiving much support then from the widespread popularity of simulated annealing and genetic algorithms. That was also sad, because those methods take many decisions randomly, instead of taking advantage of the precision that is available usually in calculated function values. Thus there was strong motivation to try to construct

some better algorithms.

At about that time, Westland Helicopters asked me to help with a constrained optimization problem that had only four variables. Therefore I developed the COBYLA software (Powell, 1994), which constructs linear polynomial approximations to the objective and constraint functions by interpolation at the vertices of simplices (a simplex in n dimensions is the convex hull of $n+1$ points, n being the number of variables). Even then, simplices had been in use for optimization without derivatives for more than 30 years. That work is the subject of Section 2, because some of the algorithms are employed often in practical calculations.

It is explained in Section 2 that MacKinnon (1998) discovered an example of failure of the Nelder and Mead (1965) simplex method, which adds to the imperfections of the techniques that are favoured by many users. Thus pattern search methods, which also had a history then of more than 30 years, received a big boost. A comprehensive review of recent work in that field is presented by Kolda, Lewis and Torczon (2003). It includes some ways of ensuring convergence that we address briefly in Section 3.

My own recent and current research is investigating the use of quadratic models of the objective function in unconstrained calculations, these approximations too being constructed from interpolation conditions without any derivatives. Good efficiency can be achieved using only $2n+1$ conditions at a time, although a quadratic polynomial has $\frac{1}{2}(n+1)(n+2)$ degrees of freedom. These findings, with a few numerical results, receive attention in Section 4.

2. Simplex methods

Let the least value of $F(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, be required, and let the function values $F(\underline{x}_i)$, $i=0,1,\ldots,n$, be available, where $F(\underline{x}_0)\le F(\underline{x}_1)\le\cdots\le F(\underline{x}_n)$. We assume that the volume of the simplex with the vertices $\underline{x}_i\in\mathcal{R}^n$, $i=0,1,\ldots,n$, is nonzero. An iteration of the original simplex method (Spendley, Hext and Himsworth, 1962) calculates the new function value $F(\hat{\underline{x}})$, where $\hat{\underline{x}}$ is the point

$$\hat{\underline{x}} \;=\; (2/n)\,\textstyle\sum_{i=0}^{n-1}\underline{x}_i \;-\; \underline{x}_n. \qquad (1)$$

If $F(\hat{\underline{x}})<F(\underline{x}_{n-1})$ is achieved, then $\hat{\underline{x}}$ replaces $\underline{x}_n$ as a vertex of the simplex. Otherwise, a "contraction" is preferred, $\underline{x}_0$ being retained, but $\underline{x}_i$ being replaced by $\frac{1}{2}(\underline{x}_0+\underline{x}_i)$ for $i=1,2,\ldots,n$, and then n more function values have to be computed for the next iteration. In both cases, the new set of function values is arranged into ascending order as before.

The point $\bar{\underline{x}}=(1/n)\sum_{i=0}^{n-1}\underline{x}_i$ is the centroid of the face of the simplex that is opposite the vertex $\underline{x}_n$, and equation (1) puts $\hat{\underline{x}}$ on the line from $\underline{x}_n$ through $\bar{\underline{x}}$, the step $\hat{\underline{x}}-\bar{\underline{x}}$ being the same as $\bar{\underline{x}}-\underline{x}_n$. Thus the volumes of the old and new simplices are the same unless contraction occurs. The test $F(\hat{\underline{x}})<F(\underline{x}_{n-1})$ for accepting $F(\hat{\underline{x}})$ is employed instead of the test $F(\hat{\underline{x}})<F(\underline{x}_n)$, in order that the $\hat{\underline{x}}$ of the next iteration will be different from the old $\underline{x}_n$.

Usually this method reduces $F(\underline{x}_n)$ steadily as the iterations proceed, but its main contribution now to optimization algorithms is as a source of ideas. In particular, if there are only two variables, and if the level sets $\{\underline{x} : F(\underline{x}) \leq F(\underline{x}_n)\}$ are bounded, one can prove as follows that contractions occur regularly. We let $\underline{y}_0$, $\underline{y}_1$ and $\underline{y}_2$ be the vertices of the first simplex, and we assume without loss of generality that $\underline{y}_0$ is at the origin. Then equation (1) with $n=2$ implies that every $\hat{\underline{x}}$ until the first contraction has the form $\alpha\,\underline{y}_1+\beta\,\underline{y}_2$, where α and β are integers. Thus the possible positions of vertices are confined to a two dimensional grid before and between contractions. Each vertex is visited at most once, because every iteration without a contraction reduces either $F(\underline{x}_n)$ or the number of function values at vertices that take the value $F(\underline{x}_n)$. It follows from the boundedness of the level sets that there is going to be another contraction. This kind of argument for establishing convergence, which can be applied to the original simplex method only for $n \leq 2$, is fundamental to the construction of the recent pattern search methods of Section 3.

We consider the original simplex method when $n=2$, when F is the function $F(\underline{x})=x_1^2+100x_2^2$, $\underline{x}\in\mathcal{R}^2$, and when the current simplex has the vertices

$$\underline{x}_0 = \begin{pmatrix} 198 \\ 0.1 \end{pmatrix}, \quad \underline{x}_1 = \begin{pmatrix} 200 \\ 0 \end{pmatrix} \quad \text{and} \quad \underline{x}_2 = \begin{pmatrix} 199 \\ 2.1 \end{pmatrix}. \tag{2}$$

The values $F(\underline{x}_0) = 198^2+1$, $F(\underline{x}_1) = 200^2$ and $F(\underline{x}_2) = 199^2+441$ satisfy the ordering condition $F(\underline{x}_0)<F(\underline{x}_1)<F(\underline{x}_2)$. Therefore equation (1) picks the point $\hat{\underline{x}}=\underline{x}_0+\underline{x}_1-\underline{x}_2$, which has the coordinates $\hat{x}_1=199$ and $\hat{x}_2=-2$. A contraction occurs, because the new function value has the property $F(\hat{\underline{x}})=(200-1)^2+400=F(\underline{x}_1)+1$. We see in expression (2) that the distance from the simplex to the origin is about 200, but that the lengths of the edges of the simplex are in the interval (2.0, 2.4). Thus it seems that the contraction has added about 200 to the number of iterations that are needed to move the simplex close to the origin, which is where the objective function is least.

The Nelder and Mead (1965) version of the simplex method, NM say, tries to avoid such inefficiencies by applying the given contraction only in the case $\min[F(\hat{\underline{x}}), F(\check{\underline{x}})] \geq F(\underline{x}_{n-1})$, where $\check{\underline{x}}$ is another point on the straight line through $\underline{x}_n$, $\bar{\underline{x}}$ and $\hat{\underline{x}}$. Most implementations of NM make the choice

$$\check{\underline{x}} \;=\; \begin{cases} 2\,\hat{\underline{x}} - \bar{\underline{x}}, & F(\hat{\underline{x}}) < F(\underline{x}_0), \\ \frac{1}{2}\,(\bar{\underline{x}} + \hat{\underline{x}}), & F(\underline{x}_0) \leq F(\hat{\underline{x}}) < F(\underline{x}_{n-1}), \\ \frac{1}{2}\,(\underline{x}_n + \bar{\underline{x}}), & F(\underline{x}_{n-1}) \leq F(\hat{\underline{x}}). \end{cases} \tag{3}$$

The first line of this formula puts $\check{\underline{x}}$ beyond $\hat{\underline{x}}$ when the move from $\underline{x}_n$ to $\hat{\underline{x}}$ reduces the objective function very well. The second line keeps $\check{\underline{x}}$ on the $\hat{\underline{x}}$ side of $\bar{\underline{x}}$ when it is known that the current iteration is going to be without a contraction. The third line puts $\check{\underline{x}}$ on the $\underline{x}_n$ side of $\bar{\underline{x}}$ when $\hat{\underline{x}}$ is unacceptable as a vertex of the

next simplex. In the example of the previous paragraph, formula (3) gives the point $\underline{\check{x}} = \frac{1}{2}(\underline{x}_n + \underline{\bar{x}})$. It has the coordinates $\check{x}_1 = 199$ and $\check{x}_2 = 1.075$, and the ordering $F(\underline{\check{x}}) < F(\underline{x}_{n-1}) < F(\underline{\hat{x}})$ occurs. Therefore $\underline{\check{x}}$ replaces $\underline{x}_n$ as a vertex and $\underline{x}_{n-1}$ is retained, which provides contraction only in the x_2 direction. Thus further progress can be made by future iterations, without increasing substantially the number of iterations that are needed to move the simplex close to the origin.

I was told in the 1970s that, in applications that require unconstrained minimization without derivatives, the NM method is employed more than any other algorithm. It is still very popular, although simulated annealing seems to have taken over first place. Therefore many users should pay careful attention to the examples of MacKinnon (1998), one of them being as follows.

We retain $n=2$, and we ask whether, on every iteration, formula (3) can give $\underline{\check{x}}=\frac{1}{2}(\underline{x}_n+\underline{\bar{x}})$, with $\underline{x}_0$ staying fixed and with $\underline{x}_1$ and $\underline{x}_2$ being replaced by $\underline{\check{x}}$ and $\underline{x}_1$, respectively, when the vertices of the simplex are updated for the next iteration. We let the fixed vertex $\underline{x}_0$ be at the origin, and we consider the vectors $\underline{x}_1$ that would occur on three consecutive iterations. Letting superscripts denote iteration numbers, we would have $\underline{x}_2^{(k)} = \underline{x}_1^{(k-1)}$ and $\underline{\check{x}} = \underline{x}_1^{(k+1)}$. Thus, because the last line of formula (3) becomes $\underline{\check{x}}=\frac{1}{2}\underline{x}_2+\frac{1}{4}\underline{x}_1$, due to $\underline{x}_0=0$ and $n=2$, we obtain the three term recurrence relation

$$\underline{x}_1^{(k+1)} \;=\; \tfrac{1}{2}\,\underline{x}_1^{(k-1)} + \tfrac{1}{4}\,\underline{x}_1^{(k)}, \qquad k=2,3,4,\ldots, \tag{4}$$

which has many solutions, one of them being the sequence

$$\underline{x}_1^{(k)} \;=\; \begin{pmatrix} \left[\,(1+\sqrt{33})\,/\,8\,\right]^k \\ \left[\,(1-\sqrt{33})\,/\,8\,\right]^k \end{pmatrix} \;\approx\; \begin{pmatrix} (0.8431)^k \\ (-0.5931)^k \end{pmatrix}, \qquad k=1,2,3,\ldots\,. \tag{5}$$

This sequence is generated automatically if the objective function satisfies $F(\underline{\check{x}}) < F(\underline{x}_1) < F(\underline{\hat{x}})$ on every iteration. In other words, because the right hand side of equation (1) is the vector $\underline{x}_0+\underline{x}_1-\underline{x}_2=\underline{x}_1^{(k)}-\underline{x}_1^{(k-1)}$, we require the conditions

$$F(\underline{x}_1^{(k+1)}) \;<\; F(\underline{x}_1^{(k)}) \;<\; F(\underline{x}_1^{(k)}-\underline{x}_1^{(k-1)}), \qquad k=2,3,4,\ldots\,. \tag{6}$$

Expression (5) shows that, if the conditions (6) hold, then $\underline{x}_1^{(k)}$ tends to the zero vector as $k \to \infty$. Further, the edges of the simplex tend to be parallel to the first coordinate direction, which allows the possibility that the gradient $\underline{\nabla} F$ has a nonzero second component at the origin. All of these things happen in the examples of MacKinnon (1998), which include the convex objective function

$$F(\xi,\eta) \;=\; \begin{cases} 360\,\xi^2+\eta+\eta^2, & \xi \le 0, \\ 6\,\xi^2+\eta+\eta^2, & \xi \ge 0, \end{cases} \qquad (\xi,\eta) \in \mathcal{R}^2. \tag{7}$$

Let ξ_k and η_k be the components of $\underline{x}_1^{(k)}$, $k \geq 1$. Equation (5) provides $\xi_{k+1} \approx 0.8431\,\xi_k > 0$ and $|\eta_{k+1}| \approx 0.5931\,|\eta_k|$, so the choice (7) gives the property

$$F(\underline{x}_1^{(k+1)}) - F(\underline{x}_1^{(k)}) \;<\; -1.7350\,\xi_k^2 + 1.5931\,|\eta_k| - 0.6482\,\eta_k^2 \;<\; 0, \tag{8}$$

where the last part depends on $|\eta_k| < \xi_k^2$, which is shown in equation (5). Moreover, $\underline{x}_1^{(k)} - \underline{x}_1^{(k-1)}$ has the components $-0.1861\,\xi_k$ and $2.6861\,\eta_k$, approximately, which provide the bound

$$F(\underline{x}_1^{(k)}) - F(\underline{x}_1^{(k)} - \underline{x}_1^{(k-1)}) \;<\; -6.46\,\xi_k^2 + 3.69\,|\eta_k| - 6.21\,\eta_k^2 \;<\; 0. \tag{9}$$

Therefore the conditions (6) are achieved. Thus the NM method finds incorrectly that the least value of the function (7) is at the origin.

The COBYLA software (Powell, 1994) also employs the function values $F(\underline{x}_i)$, $i = 0, 1, \ldots, n$, at the vertices of a simplex, as mentioned in Section 1. They are interpolated by a unique linear polynomial $L(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, say, and also each iteration requires a trust region radius $\Delta > 0$, which is adjusted automatically. Except in the highly unusual case $\underline{\nabla} L = 0$, the next vector of variables $\hat{\underline{x}}$ is obtained by minimizing $L(\hat{\underline{x}})$ subject to $\|\hat{\underline{x}} - \underline{x}_0\| \leq \Delta$, where $F(\underline{x}_0)$ is still the least value of the objective function that has been found so far, and where the vector norm is Euclidean. Thus, in the unconstrained version of COBYLA, $\hat{\underline{x}}$ is the point

$$\hat{\underline{x}} \;=\; \underline{x}_0 \;-\; (\Delta \,/\, \|\underline{\nabla} L\|)\, \underline{\nabla} L, \tag{10}$$

which is different from all the vertices of the current simplex due to the relations $L(\hat{\underline{x}}) < L(\underline{x}_0) = F(\underline{x}_0) \leq F(\underline{x}_i) = L(\underline{x}_i)$, $i = 0, 1, \ldots, n$. The value $F(\hat{\underline{x}})$ is calculated, and then the simplex for the next iteration is formed by replacing just one of the vertices of the old simplex by $\hat{\underline{x}}$. Choosing which vertex to drop is governed mainly by staying well away from the degenerate situation where the volume of the simplex collapses to zero.

In the usual case when calculated values of F include errors that are not negligible, the gradient $\underline{\nabla} L$ becomes misleading if the distances between vertices are too small. Therefore this situation is avoided by imposing a lower bound, ρ say, on Δ, except that ρ is made small eventually if high accuracy is requested. The initial and final values of ρ should be set to typical changes to the variables and to the required final accuracy in the variables, respectively. The software never increases ρ, and ρ is reduced only when the current value seems to be preventing further progress. Therefore, if $\|\underline{x}_t - \underline{x}_0\|$ is much larger than ρ, where $\|\underline{x}_t - \underline{x}_0\|$ is the greatest of the distances $\|\underline{x}_i - \underline{x}_0\|$, $i = 1, 2, \ldots, n$, then the vertex $\underline{x}_t$ may be moved to $\check{\underline{x}}$, say, before reducing ρ, the position of $\check{\underline{x}}$ being obtained by maximizing the volume of the new simplex subject to $\|\check{\underline{x}} - \underline{x}_0\| = \rho$. This "alternative iteration", which aims to improve the simplex without considering F, is relevant to some of the techniques of the next two sections.

3. Pattern search methods

Pattern search methods also make use of values of the objective function $F(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, without derivatives, and they are designed to have the following convergence property. If all level sets of the form $\{\underline{x} : F(\underline{x})\leq F(\underline{x}_j)\}$ are bounded, and if F has a continuous first derivative, then, assuming exact arithmetic and an infinite number of iterations, the condition

$$\liminf\{\,\|\underline{\nabla}F(\underline{x}_j)\| : j=1,2,3,\ldots\}\;=\;0 \tag{11}$$

is achieved, where $\{\underline{x}_j : j=1,2,3,\ldots\}$ is the set of points at which F is calculated.

The ingredients of a basic pattern search method include a grid size Δ that is reduced to zero, typically by halving when a reduction is made. For each Δ, the points $\underline{x}_j$ that are tried are restricted to a regular grid of mesh size Δ, the number of grid points being finite within any bounded region of $\mathcal{R}^n$. The condition for reducing Δ depends on a "generating set", $\mathcal{G}(\Delta)$, say, which is a collection of moves between grid points in $\mathcal{R}^n$, such that the origin is a strictly interior point of the convex hull of the elements of $\mathcal{G}(\Delta)$, and such that the longest move in $\mathcal{G}(\Delta)$ tends to zero as $\Delta\to 0$. For example, the vector $\underline{x}\in\mathcal{R}^n$ may be a grid point if and only if its components are integral multiples of Δ, and $\mathcal{G}(\Delta)$ may contain the $2n$ moves $\pm\Delta\,\underline{e}_i$, $i=1,2,\ldots,n$, where $\underline{e}_i$ is the i-th coordinate vector in $\mathcal{R}^n$.

Let $\underline{x}^{(k)}$ be the best vector of variables at the beginning of the k-th iteration, which means that it is the point of the current grid at which the least value of F has been found so far. One or more values of F at grid points are calculated on the k-th iteration, and we let $F(\underline{x}_{j(k)})$ be the least of them. The vector $\underline{x}^{(k+1)}$ for the next iteration is set to $\underline{x}_{j(k)}$ or $\underline{x}^{(k)}$ in the case $F(\underline{x}_{j(k)})<F(\underline{x}^{(k)})$ or $F(\underline{x}_{j(k)})\geq F(\underline{x}^{(k)})$, respectively. Particular attention is given to moves from $\underline{x}^{(k)}$ that are in the generating set, because, if $F(\underline{x}_{j(k)})<F(\underline{x}^{(k)})$ does not occur, then Δ is reduced for the next iteration if and only if it is known that $\underline{x}^{(k)}$ has the property

$$F(\underline{x}^{(k)})\;\leq\;F(\underline{x}^{(k)}+\underline{d}),\qquad \underline{d}\in\mathcal{G}(\Delta). \tag{12}$$

The main step in establishing the limit (11) concerns the iterations that reduce Δ. We assemble their iteration numbers in the set $\mathcal{K}$, say, and we pick an algorithm that forces $|\mathcal{K}|$ to be infinite. It is sufficient to calculate enough function values at grid points on each iteration to ensure that either condition (12) holds, or the strict reduction $F(\underline{x}^{(k+1)})<F(\underline{x}^{(k)})$ is achieved. Then, for each Δ, the condition $k\in\mathcal{K}$ is going to be satisfied eventually, because the number of occurrences of $F(\underline{x}^{(k+1)})<F(\underline{x}^{(k)})$ with the current grid is finite, due to the bounded level sets of F.

Now, for "suitable" generating sets $\mathcal{G}(\Delta)$, the property (12) implies that, when F is continuously differentiable, the infinite sequence $\|\underline{\nabla}F(\underline{x}^{(k)})\|$, $k\in\mathcal{K}$, tends to zero, "suitable" being slightly stronger than the conditions above on $\mathcal{G}(\Delta)$,

as explained in Kolda *et al* (2003). Moreover, every $\underline{x}^{(k)}$ is in the set $\{\underline{x}_j : j = 1, 2, 3, \ldots\}$ of points at which F is calculated. Therefore, if the set $\{\underline{x}^{(k)} : k \in \mathcal{K}\}$ contains an infinite number of different elements, then the convergence property (11) is obtained. Otherwise, the vectors $\underline{x}^{(k)}$, $k \in \mathcal{K}$, are all the same for sufficiently large k, and this single vector, $\underline{x}^*$ say, satisfies $\underline{\nabla} F(\underline{x}^*) = 0$. In this case, the test (12) implies that all of the values $F(\underline{x}^* + \underline{d})$, $\underline{d} \in \mathcal{G}(\Delta)$, are calculated for sufficiently small Δ. Thus vectors of the form $\underline{x}^* + \underline{d}$ provide an infinite subsequence of $\{\underline{x}_j : j = 1, 2, 3, \ldots\}$ that converges to $\underline{x}^*$. It follows from $\underline{\nabla} F(\underline{x}^*) = 0$ and the continuity of $\underline{\nabla} F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, that the limit (11) is achieved, which completes the proof.

It is important to the efficency of pattern search methods to include some other techniques. In particular, if the starting point $\underline{x}^{(k)}$ is far from the nearest grid point at which Δ can be reduced, then it may be very helpful to try some steps from $\underline{x}^{(k)}$ that are much longer than the elements of $\mathcal{G}(\Delta)$. One possibility, taken from Hooke and Jeeves (1961), is often suitable when $\|\underline{x}^{(k)} - \underline{x}^{(k-1)}\|$ is relatively large. It is to begin the search for $\underline{x}^{(k+1)}$ not at $\underline{x}^{(k)}$ but at the point $2\,\underline{x}^{(k)} - \underline{x}^{(k-1)}$, in order to follow the trend in the objective function that is suggested by the previous iteration. Details are given in Section 4.2 of Kolda *et al* (2003). That paper is a mine of information on pattern search methods, including developments for constraints on the variables.

The reduction in Δ by a pattern search method when the test (12) is satisfied is analogous to the reduction in ρ by COBYLA that is the subject of the last paragraph of Section 2. Both methods make reductions only when they seem to be necessary for further progress, which helps to keep apart the points in $\mathcal{R}^n$ at which F is calculated. Furthermore, decreases in Δ are guaranteed in pattern search methods, but, if $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, is a continuous function with bounded level sets, then it may be possible for COBYLA to make an infinite number of strict reductions in the best calculated value of F without decreasing ρ. Therefore, when asked about the convergence properties of COBYLA, I am unable to give a favourable answer within the usual conventions of convergence theory.

Instead, I prefer to draw the following conclusion from the finite precision of computer arithmetic. In practice, the number of different values of $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, is bounded above by the number of different real numbers that can be produced by a computer. Let us assume there are about 10^{18} different numbers in the interval $[10^\ell, 10^{\ell+1}]$, where ℓ is any integer from $[-200, 200]$. Then, not forgetting zero and plus and minus signs, there are no more than 10^{21} different values of $F(\underline{x})$, which is less than the number of lattice points of the unit cube when $n = 20$ and $\Delta = 0.1$. Thus we have two examples of theoretical bounds on the amount of work that are not useful. I am without practical experience of how the efficiency of pattern search methods depends on n.

4. Quadratic models

We recall the example of Section 2 with the vertices (2). It demonstrates that contractions can occur in the original simplex method when the current simplex is far from the required solution. A similar disadvantage exists in COBYLA, which is shown by retaining $F(\underline{x})=x_1^2+100\,x_2^2$, $\underline{x}\in\mathcal{R}^2$, and by letting the current simplex have the vertices

$$\underline{x}_0=\begin{pmatrix}50\\0\end{pmatrix},\quad \underline{x}_1=\begin{pmatrix}49\\1\end{pmatrix}\quad\text{and}\quad \underline{x}_2=\begin{pmatrix}49\\-1\end{pmatrix}. \tag{13}$$

We see that the edges of this simplex are much shorter than the distance from the simplex to the required solution at the origin, and that the simplex is well-conditioned, being a triangle with angles of $\pi/4$, $\pi/4$ and $\pi/2$. Therefore we expect the linear polynomial $L(\underline{x})=2500+(50-x_1)$, $\underline{x}\in\mathcal{R}^2$, which interpolates F at the vertices, to be helpful. The gradient $\underline{\nabla}L$ in formula (10), however, is such that the step from $\underline{x}_0$ to $\hat{\underline{x}}$ is directly away from the solution at the origin.

This example illustrates the well-known remark that, if F is smooth, then algorithms for unconstrained optimization are hardly ever efficient unless some attention is given to the curvature of F. One way of doing so without derivatives of F is to replace the linear polynomial $L(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, of each iteration of COBYLA, by a quadratic model of the form

$$Q(\underline{x})\;=\;F(\underline{x}_0)+(\underline{x}-\underline{x}_0)^T\underline{g}_0+\tfrac{1}{2}\,(\underline{x}-\underline{x}_0)^T\,\nabla^2 Q\,(\underline{x}-\underline{x}_0),\qquad \underline{x}\in\mathcal{R}^n, \tag{14}$$

whose parameters are also defined by interpolation. The value $F(\underline{x}_0)$ is available, because we continue to let it be the least calculated value of the objective function so far. Furthermore, after choosing the quadratic model Q, the analogue of formula (10) is to let $\hat{\underline{x}}=\underline{x}_0+\underline{d}$ be the next trial vector of variables, where $\underline{d}$ is obtained by solving (approximately) the "trust region subproblem"

$$\text{Minimize}\quad Q(\underline{x}_0+\underline{d})\quad\text{subject to}\quad \|\underline{d}\|\leq\Delta, \tag{15}$$

which is studied in many publications, including the compendious work of Conn, Gould and Toint (2000).

The parameters of the quadratic model (14), besides $F(\underline{x}_0)$, are the components of $\underline{g}_0\in\mathcal{R}^n$ and the elements of the $n\times n$ symmetric matric $\nabla^2 Q$. We obtain the correct number of interpolation conditions by replacing the points $\{\underline{x}_i\,:\,i=0,1,\ldots,n\}$ of COBYLA by the set $\{\underline{x}_i\,:\,i=0,1,\ldots,m-1\}$, where $m=\frac{1}{2}(n+1)(n+2)$. The new points must have the "nonsingularity property" that the parameters of Q are defined uniquely by the equations

$$Q(\underline{x}_i)\;=\;F(\underline{x}_i),\qquad i=0,1,\ldots,m-1, \tag{16}$$

which is analogous to each simplex of COBYLA having a nonzero volume. We retain the feature that each iteration changes only one of the interpolation points.

Let $\underline{x}_t$ be replaced by $\hat{\underline{x}}$. The "nonsingularity property" is preserved well if $|\ell_t(\hat{\underline{x}})|$ is relatively large, where $\ell_t(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, is the quadratic polynomial that satisfies the "Lagrange conditions" $\ell_t(\underline{x}_i)=\delta_{it}$, $i=0,1,\ldots,m-1$, the right hand side being the Kronecker delta. I have assembled these ingredients and more into the UOBYQA Fortran software (Powell, 2002), where UOBYQA denotes Unconstrained Optimization BY Quadratic Approximation. By employing updating techniques, the amount of routine work of each iteration is only $\mathcal{O}(m^2)$. This software has been applied successfully to many test problems, including some with first derivative discontinuities.

The $\mathcal{O}(n^4)$ work of the iterations of UOBYQA prevents n from being more than about 100 in practice. Therefore I have investigated the idea of reducing the number m of interpolation conditions (16) from $\frac{1}{2}(n+1)(n+2)$ to only $m=2n+1$. Then there are enough data to define quadratic models with diagonal second derivative matrices, which is done before the first iteration. On each iteration, however, when updating the quadratic model from $Q^{(\rm old)}$ to $Q^{(\rm new)}$, say, I take up the freedom in $Q^{(\rm new)}$ by minimizing $\|\nabla^2Q^{(\rm new)}-\nabla^2Q^{(\rm old)}\|_F$, which is the Frobenius norm of the change to the second derivative matrix of Q. Of course $\nabla^2Q^{(\rm new)}$ has to be symmetric, and $Q^{(\rm new)}$ has to satisfy the interpolation conditions (16), after the current iteration has moved just one of the interpolation points. This technique is analogous to the "symmetric Broyden method" when first derivatives are available, and it has two very welcome features. One is that Q can be updated in only $\mathcal{O}(m^2)$ operations, even in the case $m=2n+1$, because the change to ∇^2Q can be written in the form

$$\nabla^2Q^{(\rm new)}-\nabla^2Q^{(\rm old)}\;=\;\sum_{i=1}^{m-1}\mu_i\,(\underline{x}_i-\underline{x}_0)\,(\underline{x}_i-\underline{x}_0)^T, \qquad (17)$$

where the values of the parameters μ_i, $i=1,2,\ldots,m-1$, have to be calculated. The other feature is that the use of Frobenius norms provides a least squares projection operator in the space of symmetric matrices. Thus, if $F(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, is itself a quadratic function, the updating provides the property

$$\|\nabla^2Q^{(\rm new)}-\nabla^2F\|_F^2\;=\;\|\nabla^2Q^{(\rm old)}-\nabla^2F\|_F^2-\|\nabla^2Q^{(\rm new)}-\nabla^2Q^{(\rm old)}\|_F^2, \qquad (18)$$

which suggests that the approximations $\nabla^2Q\approx\nabla^2F$ become more accurate as the iterations proceed. This updating method is at the heart of my NEWUOA Fortran software (Powell, 2006), which has superseded UOBYQA. The value of m is chosen by the user from the interval $[n+2,\,\frac{1}{2}(n+1)(n+2)]$.

I had not expected NEWUOA to require fewer calculations of the objective function than UOBYQA. In many cases when n is large, however, NEWUOA completes its task using less than the $\frac{1}{2}(n+1)(n+2)$ values of F that are required by the first iteration of UOBYQA. My favourite example is the sum of squares

$$F(\underline{x})\;=\;\sum_{i=1}^{2n}\left\{b_i-\sum_{j=1}^{n}\left[\,S_{ij}\,\sin(x_j/\sigma_j)+C_{ij}\,\cos(x_j/\sigma_j)\,\right]\right\}^2, \qquad \underline{x}\in\mathcal{R}^n, \qquad (19)$$

n	Numbers of calculations of F ($\#F$)				
	Case 1	Case 2	Case 3	Case 4	Case 5
10	373	370	365	420	325
20	1069	1131	1083	1018	952
40	2512	2202	1965	2225	2358
80	4440	4168	4168	4185	4439
160	6989	7541	7133	7237	7633
320	12816	14077	13304	13124	12523

Table 1: NEWUOA applied to the test problem (19)

where all the elements S_{ij} and C_{ij} are random integers from $[-100, 100]$, where each σ_j is chosen randomly from $[1, 10]$, and where each b_i is defined by $F(\underline{x}^*)=0$, for a vector $\underline{x}^* \in \mathcal{R}^n$ that is also chosen randomly. Thus the objective function is periodic, with local maxima and saddle points and with a global minimum at $\underline{x}=\underline{x}^*$. The initial and final values of ρ (see the last paragraph of Section 2) are set to 0.1 and 10^{-6}, respectively, and NEWUOA is given a starting vector $\underline{x}^{\rm o}$, which is picked by letting the weighted differences $(x_j^{\rm o}-x_j^*)/\sigma_j$, $j=1,2,\ldots,n$, be random numbers from $[-\pi/10, \pi/10]$. For each choice of n, five test problems were generated randomly. For each case, the total number of calculations of F is shown in Table 1. All the final values of the error $\|\underline{x}-\underline{x}^*\|_\infty$ were found to be less than 6×10^{-6}. We see that the growth of $\#F$ as n increases is no faster than linear, although the model Q has $\mathcal{O}(n^2)$ parameters. I believe that this stunning success is due mainly to the indication in equation (18) that $\|\nabla^2 Q^{(\rm new)}-\nabla^2 Q^{(\rm old)}\|_F$ tends to zero.

Four of my Fortran packages, namely TOLMIN, COBYLA, UOBYQA and NEWUOA, have been mentioned in this paper. They are all available for general use free of charge. I am always pleased to provide copies of them by e-mail, my address being mjdp@cam.ac.uk.

Acknowledgement

I had the honour of presenting a William Benter Distinguished Lecture at the Liu Bie Ju Centre for Mathematical Sciences of the City University of Hong Kong on February 7th, 2007. This paper describes the material of the lecture and was written after the lecture had been delivered, mainly during my stay in Hong Kong. I am very grateful for the excellent support, facilities and hospitality that I enjoyed there, both from the Liu Bie Ju Centre and from the Mathematics Department of City University.

References

A.R. Conn, N.I.M. Gould and Ph.L. Toint (2000), *Trust-Region Methods*, MPS/SIAM Series on Optimization, SIAM (Philadelphia).

R. Fletcher and M.J.D. Powell (1963), "A rapidly convergent descent method for minimization", *Comput. J.*, Vol. 6, pp. 163–168.

R. Hooke and T.A. Jeeves (1961), "Direct search solution of numerical and statistical problems", *J. Assoc. Comput. Mach.*, Vol. 8, pp. 212–229.

T.G. Kolda, R.M. Lewis and V. Torczon (2003), "Optimization by direct search: new perspectives on some classical and modern methods", *SIAM Review*, Vol. 45, pp. 385–482.

K.I.M. McKinnon (1998), "Convergence of the Nelder–Mead simplex method to a nonstationary point", *SIAM J. Optim.*, Vol. 9, pp. 148–158.

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

M.J.D. Powell (1989), "A tolerant algorithm for linearly constrained optimization calculations", *Math. Programming*, Vol. 45, pp. 547–566.

M.J.D. Powell (1994), "A direct search optimization method that models the objective and constraint functions by linear interpolation", in *Advances in Optimization and Numerical Analysis*, eds. S. Gomez and J-P. Hennart, Kluwer Academic (Dordrecht), pp. 51–67.

M.J.D. Powell (2002), "UOBYQA: unconstrained optimization by quadratic approximation", *Math. Programming B.*, Vol. 92, pp. 555–582.

M.J.D. Powell (2006), "The NEWUOA software for unconstrained optimization without derivatives", in *Large-Scale Nonlinear Optimization*, eds. G. Di Pillo and M. Roma, Springer (New York), pp. 255–297.

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.