On the use of quadratic models in unconstrained minimization without derivatives[1]

M.J.D. Powell

Abstract: Quadratic approximations to the objective function provide a way of estimating first and second derivatives in iterative algorithms for unconstrained minimization. Therefore we address the construction of suitable quadratic models Q by interpolating values of the objective function F. On a typical iteration, the objective function is calculated at the point that minimizes the current quadratic model subject to a trust region bound, and we find that these values of F provide good information for the updating of Q, except that a few extra values are needed occasionally to avoid degeneracy. The number of interpolation points and their positions can be controlled adequately by deleting one of the current points to make room for each new one. An algorithm is described that works in this way. It is applied to some optimization calculations that have between 10 and 160 variables. The numerical results suggest that, if $m=2n+1$, then the number of evaluations of F is only of magnitude n, where m and n are the number of interpolation conditions of each model and the number of variables, respectively. This success is due to the technique that updates Q. It minimizes the Frobenius norm of the change to $\nabla^2 Q$, subject to the interpolation conditions that have been mentioned.

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

March, 2003.

[1]Presented at The First International Conference on Optimization Methods and Software (December, 2002), Hangzhou, China.

1. Least Frobenius norm updating of quadratic models

We consider the efficiency and use of quadratic models in iterative algorithms for unconstrained minimization calculations. Let $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, be the objective function whose minimum value is required, let k be the iteration number, and let $\underline{x}_k^*$ be the vector of variables such that $F(\underline{x}_k^*)$ is the least calculated value of the objective function so far at the beginning of the k-th iteration. Then the quadratic model of the k-th iteration is a quadratic polynomial $Q_k(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, that satisfies $Q_k(\underline{x}_k^*) = F(\underline{x}_k^*)$. Each quadratic model is formed automatically. We will give particular attention to the construction of Q_{k+1} from Q_k, which is done by the k-th iteration.

Usually the iteration generates a change to the variables that is based on the approximation

$$Q_k(\underline{x}) \;\approx\; F(\underline{x}), \qquad \underline{x} \in \mathcal{R}^n. \tag{1.1}$$

Further, in trust region algorithms it is assumed that this approximation is helpful only if $\underline{x}$ is sufficiently close to $\underline{x}_k^*$. Specifically, a positive parameter Δ_k, called the "trust region radius", is also available at the beginning of the iteration, and, until termination or some special action is required, the next trial vector of variables is $\underline{x}_k^* + \underline{d}_k$, where $\underline{d}_k$ is an estimate of the vector $\underline{d}$ that solves the subproblem

$$\text{Minimize} \quad Q_k(\underline{x}_k^* + \underline{d}) \quad \text{subject to} \quad \|\underline{d}\| \leq \Delta_k, \tag{1.2}$$

the vector norm being Euclidean. There are good ways of generating $\underline{d}_k$ that do not require the second derivative matrix $\nabla^2 Q_k$ to be positive definite (see Moré and Sorensen, 1983, for instance). Then the new function value $F(\underline{x}_k^* + \underline{d}_k)$ may be calculated, and $\underline{x}_{k+1}^*$ may be defined by the formula

$$\underline{x}_{k+1}^* \;=\; \begin{cases} \underline{x}_k^*, & F(\underline{x}_k^* + \underline{d}_k) \geq F(\underline{x}_k^*) \\ \underline{x}_k^* + \underline{d}_k, & F(\underline{x}_k^* + \underline{d}_k) < F(\underline{x}_k^*). \end{cases} \tag{1.3}$$

Equation (1.3) provides the best vector of variables so far, but many algorithms set $\underline{x}_{k+1}^*$ to $\underline{x}_k^* + \underline{d}_k$ only if the reduction $F(\underline{x}_k^*) - F(\underline{x}_k^* + \underline{d}_k)$ is sufficiently large. Originally this device was introduced to assist proofs of convergence, but now the use of sufficient reductions has become standard practice. In Sections 2 and 3, however, we are going to study a trust region algorithm that retains formula (1.3), because we welcome every decrease that occurs in the value of the objective function. The author does not know of cases where formula (1.3) prevents convergence, provided that the new trust region radius Δ_{k+1} is chosen carefully. For example, the value $\Delta_{k+1} = \frac{1}{2}\|\underline{d}_k\|$ may be suitable if the condition

$$F(\underline{x}_k^*) - F(\underline{x}_k^* + \underline{d}_k) \;<\; 0.1\,[\,Q_k(\underline{x}_k^*) - Q_k(\underline{x}_k^* + \underline{d}_k)\,] \tag{1.4}$$

holds, because then the iteration fails to achieve one tenth of the reduction in the objective function that is predicted by the quadratic model. On the other hand, if the change in F compares favourably with the predicted change, then the choice $\Delta_{k+1} = \max[\Delta_k, 2\|\underline{d}_k\|]$ may be made. Details of this kind are addressed in many publications, including the book of Conn, Gould and Toint (2000).

Several methods have been proposed for constructing the new quadratic model Q_{k+1} when first derivatives of the objective function are available (see Fletcher, 1987, for instance). They satisfy $Q_{k+1}(\underline{x}^*_{k+1}) = F(\underline{x}^*_{k+1})$ and $\underline{\nabla} Q_{k+1}(\underline{x}^*_{k+1}) = \underline{\nabla} F(\underline{x}^*_{k+1})$, so Q_{k+1} has the form

$$Q_{k+1}(\underline{x}^*_{k+1}+\underline{d}) \;=\; F(\underline{x}^*_{k+1}) + \underline{d}^T \underline{\nabla} F(\underline{x}^*_{k+1}) + \tfrac{1}{2}\, \underline{d}^T B_{k+1}\, \underline{d}, \qquad \underline{d}\in\mathcal{R}^n, \tag{1.5}$$

for some $n\times n$ symmetric matrix B_{k+1}. The usual choices of B_{k+1} depend on the remark that, if F is twice differentiable, then its second derivatives have the property

$$\Big\{ \int_0^1 \nabla^2 F(\underline{x}^*_k+\theta\, \underline{d}_k)\, d\theta \Big\}\, \underline{d}_k \;=\; \underline{\nabla} F(\underline{x}^*_k+\underline{d}_k) - \underline{\nabla} F(\underline{x}^*_k). \tag{1.6}$$

We see that the left hand side is the product $\nabla^2 F \underline{d}_k$ if F happens to be a quadratic polynomial, where $\nabla^2 F$ is the constant second derivative matrix of F. Therefore the difference in gradients on the right hand side is calculated, and B_{k+1} is required to satisfy the condition

$$B_{k+1}\, \underline{d}_k \;=\; \underline{\nabla} F(\underline{x}^*_k+\underline{d}_k) - \underline{\nabla} F(\underline{x}^*_k). \tag{1.7}$$

A good discussion of the merits of this technique can be found in Dennis and Schnabel (1983).

Equation (1.7) provides only n constraints on the new model (1.5). Because B_{k+1} is an $n\times n$ symmetric matrix, the number of remaining degrees of freedom is $\frac{1}{2}n(n-1)$, which are fixed by different methods in different ways. In particular, the well-known symmetric Broyden formula is relevant to our work. It takes up the freedom by minimizing the Frobenius norm

$$\|B_{k+1} - B_k\|_F \;=\; \|\nabla^2 Q_{k+1} - \nabla^2 Q_k\|_F \tag{1.8}$$

of the difference between the second derivative matrices of the old and new quadratic models. The Frobenius norm of a matrix is the square root of the sum of squares of its elements.

Thus the symmetric Broyden formula enjoys the highly useful property that is given in the theorem below. The theorem is presented in a way that is also applicable to the method for unconstrained minimization without derivatives that is going to be studied in Sections 2 and 3. Specifically, we write Q_{k+1} in the form

$$Q_{k+1}(\underline{x}) \;=\; \sum_{j=1}^{m^*} \mu_j\, b_j(\underline{x}), \qquad \underline{x}\in\mathcal{R}^n, \tag{1.9}$$

where b_j, $j=1,2,\ldots,m^*$, is a basis of the linear space of polynomials of degree at most two from $\mathcal{R}^n$ to $\mathcal{R}$, the value of m^* being $\frac{1}{2}(n+1)(n+2)$. Then the condition

(1.7) on $B_{k+1} = \nabla^2 Q_{k+1}$ provides n linear equality constraints on the coefficients μ_j, $j=1,2,\ldots,m^*$. Further, the properties

$$Q_{k+1}(\underline{x}^*_{k+1}) \;=\; F(\underline{x}^*_{k+1}) \quad \text{and} \quad \underline{\nabla} Q_{k+1}(\underline{x}^*_{k+1}) \;=\; \underline{\nabla} F(\underline{x}^*_{k+1}) \tag{1.10}$$

are also linear equality constraints on the coefficients of expression (1.9). We let $\mathcal{S}$ be the set of polynomials of the form (1.9) that satisfy these constraints, so $\mathcal{S}$ is a linear manifold. It follows from equations (1.6) and (1.10) that, if F happens to be a quadratic polynomial, then F is also an element of $\mathcal{S}$.

Theorem: Let $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, be a quadratic polynomial, and let the old quadratic model Q_k be available. The new model Q_{k+1} has to satisfy the linear constraints $Q_{k+1} \in \mathcal{S}$, as stated in the previous paragraph, and the remaining freedom in Q_{k+1} has to make the Frobenius norm $\|\nabla^2 Q_{k+1} - \nabla^2 Q_k\|_F$, $Q_{k+1} \in \mathcal{S}$, as small as possible. Then, if F is in $\mathcal{S}$, this construction provides the relation

$$\|\nabla^2 Q_{k+1} - \nabla^2 F\|_F^2 \;=\; \|\nabla^2 Q_k - \nabla^2 F\|_F^2 \;-\; \|\nabla^2 Q_{k+1} - \nabla^2 Q_k\|_F^2. \tag{1.11}$$

Proof: Equation (1.11) is a well-known consequence of orthogonal projection within an inner product space. Specifically, we employ the inner product that is induced by the semi-norm $\|Q\| = \|\nabla^2 Q\|_F$, $Q \in \mathcal{Q}$, where $\mathcal{Q}$ is the linear space of polynomials of degree at most two. Thus Q_{k+1} is the orthogonal projection of Q_k onto the linear manifold $\mathcal{S} \subset \mathcal{Q}$, which gives the required result. $\Box$

The theorem implies the inequality

$$\|\nabla^2 Q_{k+1} - \nabla^2 F\|_F \;\leq\; \|\nabla^2 Q_k - \nabla^2 F\|_F, \tag{1.12}$$

when F is a quadratic function. Thus, if the symmetric Broyden formula is employed on every iteration, then the Frobenius norm of the error of the approximation $\nabla^2 Q_k \approx \nabla^2 F$ decreases monotonically during the iterative procedure. This property is welcome, but at first sight does not cause much enthusiasm, because the assumption that F is quadratic seems to be very restrictive, and because $\|\nabla^2 Q_k - \nabla^2 F\|_F$ may stay bounded away from zero as $k \rightarrow \infty$. On the other hand, the theorem also provides the limit

$$\lim_{k\rightarrow\infty} \|\nabla^2 Q_{k+1} - \nabla^2 Q_k\|_F \;=\; 0, \tag{1.13}$$

which causes $\underline{x}_k$, $k=1,2,3,\ldots$, to converge to the optimal vector of variables at a superlinear rate, when the objective function is quadratic, for many of the usual choices of the changes $\underline{d}_k$ to the variables.

These changes are derived from the quadratic models, and then the standard way of proving superlinear convergence, due to Broyden, Dennis and Moré (1973), requires the condition

$$\lim_{k\rightarrow\infty} \|\underline{\nabla} F(\underline{x}^*_k + \underline{d}_k) - \underline{\nabla} Q_k(\underline{x}^*_k + \underline{d}_k)\| \,/\, \|\underline{d}_k\| \;=\; 0. \tag{1.14}$$

In other words, as k becomes large, the approximation (1.1) has to provide a good prediction of the new gradient $\underline{\nabla} F(\underline{x}_k^*+\underline{d}_k)$. Now the property $\underline{\nabla} Q_k(\underline{x}_k^*)=\underline{\nabla} F(\underline{x}_k^*)$, the constraint (1.7) on $B_{k+1}=\nabla^2 Q_{k+1}$, and $Q_k \in \mathcal{Q}$ imply the identity

$$\begin{aligned}
&\|\underline{\nabla} F(\underline{x}_k^*+\underline{d}_k) - \underline{\nabla} Q_k(\underline{x}_k^*+\underline{d}_k)\| \,/\, \|\underline{d}_k\| \\
&\quad = \|\{\underline{\nabla} F(\underline{x}_k^*+\underline{d}_k) - \underline{\nabla} F(\underline{x}_k^*)\} - \{\underline{\nabla} Q_k(\underline{x}_k^*+\underline{d}_k) - \underline{\nabla} Q_k(\underline{x}_k^*)\}\| \,/\, \|\underline{d}_k\| \\
&\quad = \|(\nabla^2 Q_{k+1} - \nabla^2 Q_k)\, \underline{d}_k\| \,/\, \|\underline{d}_k\|. \qquad (1.15)
\end{aligned}$$

It follows from equation (1.13) that the limit (1.14) is achieved. This argument is also valid for many other objective functions that have continuous second derivatives, because then equation (1.11) holds if $\nabla^2 F$ is the matrix inside the braces of expression (1.6), but careful attention has to be given to the changes to $\nabla^2 F$ that occur in expression (1.11) during the sequence of iterations. The purpose of these remarks is to point out that the least Frobenius norm method for updating quadratic models may provide fast convergence, even if $\|\nabla^2 Q_k - \nabla^2 F(\underline{x}_k^*)\|_F$ does not become small as k is increased.

Throughout the remainder of this paper, we study trust region methods when $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, is a general function, and its derivatives are not available. Therefore the quadratic models are derived from interpolation equations. We let the conditions on Q_{k+1} be the constraints

$$Q_{k+1}(\underline{x}_i^+) \;=\; F(\underline{x}_i^+), \qquad i=1,2,\ldots,m, \qquad (1.16)$$

where m is a prescribed integer that is independent of k, but the positions of the interpolation points $\underline{x}_i^+ \in \mathcal{R}^n$, $i=1,2,\ldots,m$, are generated automatically. They include the point $\underline{x}_{k+1}^*$, in order that $Q_{k+1}(\underline{x}_{k+1}^*) = F(\underline{x}_{k+1}^*)$ holds, and also they must have the following two properties. **(1)** The equations (1.16) can be satisfied by a model $Q_{k+1} \in \mathcal{Q}$ for any right hand sides, and **(2)** the interpolation points do not all lie in a hyperplane $\{\underline{x} : \pi(\underline{x})=0\}$, where π is a linear polynomial. It follows that, given the old model Q_k, there is a unique new model Q_{k+1} that minimizes $\|\nabla^2 Q_{k+1} - \nabla^2 Q_k\|_F$, $Q_{k+1} \in \mathcal{Q}$, subject to the constraints (1.16). We address algorithms for unconstrained minimization without derivatives that update quadratic models in this way.

Condition **(2)** of the last paragraph implies $m \geq n+1$. It also implies that, if $m=n+1$ is chosen, then there is enough freedom in the constant and linear terms of Q_{k+1} to satisfy the equations (1.16). Thus the least Frobenius norm updating method yields $\nabla^2 Q_{k+1} = \nabla^2 Q_k$, so equation (1.11) becomes trivial. Moreover, condition **(1)** implies $m \leq m^* = \frac{1}{2}(n+1)(n+2)$. Further, if $m=m^*$ is chosen, then all of the freedom in the new model is taken up by the interpolation conditions. Hence Q_{k+1} is independent of Q_k, and $\nabla^2 Q_{k+1} = \nabla^2 F$ occurs if F is quadratic, so again equation (1.11) becomes trivial. We assume that m is picked from the interval $[n+2, m^*]$, in order that some changes are made to the second derivatives of the current quadratic model during the minimization calculation.

We now let $\mathcal{S}$ be the linear manifold of functions of the form (1.9) that satisfy the linear equality constraints (1.16). It follows from the theorem of this section that, when F is a quadratic function, the updating method provides the limit (1.13). Therefore, in view of the remarks above on convergence when first derivatives of F are available, we are optimistic that a version of the method without derivatives can be developed that enjoys excellent convergence properties in practice. Theoretical analysis is more difficult than before, however, because $\nabla^2 Q_{k+1} = \nabla^2 Q_k$ is possible when a change to the second derivatives is expected, an example being given below. Therefore we are going to draw our conclusions from numerical experiments in Section 3, using the algorithm that is the subject of Section 2.

The algorithm retains only m values of F for the beginning of the k-th iteration, namely the right hand sides of the interpolation equations

$$Q_k(\underline{x}_i) \;=\; F(\underline{x}_i), \qquad i=1,2,\ldots,m, \tag{1.17}$$

that are satisfied by Q_k. Further, except on a few iterations that do not change the quadratic model, the k-th iteration calculates just one new function value, namely $F(\underline{x}_k^*+\underline{d}_k)$, where $\underline{x}_k^*+\underline{d}_k$ is not in the set $\{\underline{x}_i : i=1,2,\ldots,m\}$. Therefore at most one interpolation is moved, say $\underline{x}_t$, and the new points have the positions

$$\underline{x}_i^+ = \underline{x}_i, \quad i\in\{1,2,\ldots,m\}\backslash\{t\} \qquad \text{and} \qquad \underline{x}_t^+ = \underline{x}_k^*+\underline{d}_k, \tag{1.18}$$

for some integer t in $[1,m]$. It follows that the updating technique provides the new model

$$Q_{k+1}(\underline{x}) \;=\; Q_k(\underline{x}) + \{F(\underline{x}_t^+)-Q_k(\underline{x}_t^+)\}\,\ell_t^+(\underline{x}), \qquad \underline{x}\in\mathcal{R}^n, \tag{1.19}$$

where ℓ_t^+ is the element of $\mathcal{Q}$ such that $\|\nabla^2 \ell_t^+\|_F$ is least subject to the Lagrange conditions

$$\ell_t^+(\underline{x}_i^+) \;=\; \delta_{it}, \qquad i=1,2,\ldots,m. \tag{1.20}$$

We end this section with the example that demonstrates $\nabla^2 Q_{k+1} = \nabla^2 Q_k$. We let n and m have the values 2 and 4, respectively, and we let the points $\underline{x}_i$, $i=1,2,3$, lie on the line $\{\underline{x} : \pi(\underline{x})=0\}$, where π is a linear polynomial, but $\pi(\underline{x}_4)$ is nonzero, due to condition **(2)** on the old interpolation points. We also assume $\pi(\underline{x}_k^*+\underline{d}_k) \neq 0$, which permits $t=4$ in formula (1.18). This choice would give $\nabla^2 Q_{k+1} = \nabla^2 Q_k$, however, because the function ℓ_t^+ in expression (1.19) would be the linear polynomial

$$\ell_t^+(\underline{x}) \;=\; \pi(\underline{x}) \,/\, \pi(\underline{x}_t^+), \qquad \underline{x}\in\mathcal{R}^2. \tag{1.21}$$

On the other hand, each of the choices $t=1$, $t=2$ and $t=3$ is also allowed by conditions **(1)** and **(2)** on the new interpolation points, and in these cases $F(\underline{x}_k^*+\underline{d}_k) \neq Q_k(\underline{x}_k^*+\underline{d}_k)$ would imply $\nabla^2 Q_{k+1} \neq \nabla^2 Q_k$. The point of the example is that the algorithm of Section 2 is designed to be robust when the objective function is noisy. Therefore it includes a feature that keeps the interpolation points well separated automatically. Thus $t=4$ is mandatory in the example if $\underline{x}_k^*+\underline{d}_k$ is sufficiently close to $\underline{x}_4$. Then the replacement of one of the collinear interpolation points has to wait until a later iteration, which postpones any improvement to the second derivative matrix of the quadratic model.

2. An algorithm for unconstrained minimization without derivatives

Recently the author developed the algorithm that has the name UOBYQA, which is an acronym for Unconstrained Optimization BY Quadratic Approximation (Powell, 2002a). It has the form that is described in Section 1, except that the least Frobenius norm updating method is not employed, because on each iteration the new quadratic model Q_{k+1} is defined uniquely by the interpolation conditions (1.16), the value of m being $m=m^*=\frac{1}{2}(n+1)(n+2)$. It follows that the routine work of each iteration is of magnitude n^4, which is tolerable for $n \leq 25$, but the total amount of computation becomes prohibitive for $n \geq 50$. Therefore the author investigated the possibility of forming useful quadratic models from only $\mathcal{O}(n)$ function values. That task is straightforward if $\nabla^2 F$ is known to have only $\mathcal{O}(n)$ nonzero elements, because most of the freedom in each new model Q_{k+1} can be taken up by including the sparsity pattern of $\nabla^2 F$ in $\nabla^2 Q_{k+1}$, so m is less than m^* by the number of independent sparsity conditions. One can also impose a sparsity pattern on $\nabla^2 Q_{k+1}$ that is not present in $\nabla^2 F$. Indeed, at a conference in Kyoto in 2001, the author presented numerical results for a version of UOBYQA with $m=2n+1$, that value being appropriate because the second derivative matrix of every quadratic model is forced to be diagonal, even if none of the elements of $\nabla^2 F$ is zero (Powell, 2002b). Thus the amount of work of each iteration is reduced to only $\mathcal{O}(n^2)$, including the approximate solution of the trust region subproblem (1.2). The algorithm was applied successfully to some unconstrained minimization calculations without derivatives that have more than 100 variables.

In all of those methods, the number of interpolation conditions is the number of degrees of freedom in each new quadratic model, and the construction of Q_{k+1} does not depend on Q_k. Indeed, the least Frobenius norm updating method was not tried by the author until January, 2002. It gave some results that seemed to deserve publicity as soon as possible, so they are included in the author's contribution to the proceedings of the Kyoto conference (Powell, 2002b), although the work was done after the meeting. Those activities are mentioned, in order to explain that much software for trust region methods with quadratic models was available at the beginning of the work on least Frobenius norm updating. Therefore many features of the algorithm of this section are taken from UOBYQA, and the details of our implementation of the updating method are also published elsewhere (Powell, 2002c). Moreover, our new techniques may not yet be in their final form. Therefore only an outline is given below of the algorithm that calculated the numerical results of Section 3.

The user of the algorithm has to provide an initial vector of variables, $\underline{x}_0^*$ say, the initial and final values of the trust region radius, $\rho_{\rm beg}$ and $\rho_{\rm end}$ say, the number m of interpolation conditions, and a subroutine that calculates $F(\underline{x})$ for any vector of variables $\underline{x} \in \mathcal{R}^n$. Let $\underline{x}_1$ be $\underline{x}_0^*$ and, for $j=1,2,\ldots,n$, let $\underline{x}_{j+1}$ and $\underline{x}_{j+n+1}$ be $\underline{x}_0^* + \rho_{\rm beg}\underline{e}_j$ and $\underline{x}_0^* - \rho_{\rm beg}\underline{e}_j$, respectively, where $\underline{e}_j$ is the j-th coordinate vector in $\mathcal{R}^n$. If $m \leq 2n+1$, the initial interpolation points are the first m points of the sequence $\underline{x}_i$, $i=1,2,\ldots,2n+1$. Otherwise, the sequence is augmented by $m-2n-1$ points of the form $\underline{x}_0^* \pm \rho_{\rm beg}\underline{e}_\alpha \pm \rho_{\rm beg}\underline{e}_\beta$, where α and β are different integers from $[1,n]$, and where the $m-2n-1$ choices of the pairs $\{\alpha,\beta\}$ are all different. The values $F(\underline{x}_i)$, $i=1,2,\ldots,m$, are computed. Then the quadratic model Q_1 of the first iteration is formed by minimizing $\|\nabla^2 Q_1\|_F$, $Q_1 \in \mathcal{Q}$, subject to the equations

$$Q_1(\underline{x}_i) \;=\; F(\underline{x}_i), \qquad i=1,2,\ldots,m, \tag{2.1}$$

which requires only $\mathcal{O}(m)$ operations, because of the positions of the initial interpolation points. Further, $\underline{x}_1^*$ is set to the first point in the sequence $\underline{x}_i$, $i = 1,2,\ldots,m$, that provides the least value of the objective function so far, and $\Delta_1 = \rho = \rho_{\rm beg}$ is set too. Moreover, an $(m+n+1)\times(m+n+1)$ symmetric matrix H is constructed, which gives the coefficients of the Lagrange functions for use in formula (1.19) as described later. These operations generate the data that are needed for the first iteration of the algorithm.

The purpose of the parameter ρ is taken from UOBYQA. Specifically, interpolation points are kept apart by calculating the new function value $F(\underline{x}_k^* + \underline{d}_k)$ only if $\|\underline{d}_k\| \geq \frac{1}{2}\rho$ holds for the current value of ρ. Moreover, ρ is never increased, and it is not reduced until the restriction on new function evaluations seems to be preventing further progress. A typical change in ρ is by a factor of ten, and $\rho_{\rm beg}$ and $\rho_{\rm end}$ are its initial and final values, so we require $\rho_{\rm end} \leq \rho_{\rm beg}$. These devices are intended to make the algorithm robust when the objective function is noisy. Indeed, we wish to postpone until late in the calculation the situation when a substantial discontinuity in F occurs between two close interpolation points, because then the model may become useless, due to the magnitudes of its first and second derivatives. An analogous situation arises from computer rounding errors, if F would be smooth in exact arithmetic. This happens in the numerical experiments of Section 3, and we find that the algorithm provides good accuracy in the final value of each objective function.

The trust region parameter Δ_k of the subproblem (1.2) allows $\|\underline{d}_k\|$ to be much larger than ρ occasionally, because the choice $\Delta_{k+1} = 2\|\underline{d}_k\|$, mentioned in Section 1, may have been made on several previous iterations. Thus some inefficiencies can be avoided if $\rho_{\rm beg}$ is too small, or if the variables move into a region of $\mathcal{R}^n$ where reducing the objective function becomes less difficult. We prefer not to increase ρ, because of the work that is required whenever ρ is reduced, due to the conditions on the acceptability of the quadratic model that are given below. Therefore it is advantageous to work with both ρ and Δ_k. The usual ways of adjusting trust region radii can provide both increases and decreases in the sequence Δ_k, $k=1,2,3,\ldots$, but, because of the remarks of the previous paragraph, we depart from normal practice by including the constraint $\Delta_{k+1} \geq \rho$ when Δ_{k+1} is set by the k-th iteration.

The iterations are of a usual form for trust region algorithms, Δ_k being revised automatically and ρ being constant, until the choice of $\underline{d}_k$ gives one of the conditions

$$\|\underline{d}_k\| < \tfrac{1}{2}\rho \quad \text{or} \quad F(\underline{x}_k^*+\underline{d}_k) > F(\underline{x}_k^*) - \tfrac{1}{2}\,[\,Q_k(\underline{x}_k^*) - Q_k(\underline{x}_k^*+\underline{d}_k)\,]. \tag{2.2}$$

Then $\|\underline{x}_k^*-\underline{x}_{k-1}^*\|$ may be much larger than ρ, and the sequence $\underline{x}_j^*$, $j=1,2,\ldots,k$, may have followed a narrow valley of the objective function in $\mathcal{R}^n$. Thus it is not unusual for the interpolation equations (1.17) to be nearly linearly dependent, especially if ρ_{beg} is very small. Therefore, in addition to the trust region iterations, the algorithm also employs "model iterations", which correct tendencies towards linear dependence. Each model iteration alters the position of an interpolation point that is furthest from the best point so far.

Specifically, if the k-th iteration is a model iteration, then the new interpolation points are defined by formula (1.18), where t is the least integer in $[1,m]$ that satisfies the equation

$$\|\underline{x}_t-\underline{x}_k^*\| \;=\; \max\{\|\underline{x}_i-\underline{x}_k^*\| \;:\; i=1,2,\ldots,m\}. \tag{2.3}$$

Further, letting ℓ_t be the t-th Lagrange function of the current interpolation points, which is the element of $\mathcal{Q}$ that minimizes $\|\nabla^2 \ell_t\|_F$ subject to $\ell_t(\underline{x}_i)=\delta_{it}$, $i=1,2,\ldots,m$, the vector $\underline{d}_k$ of formula (1.18) is chosen to be a crude estimate of the solution to the subproblem

$$\text{Maximize} \quad |\ell_t(\underline{x}_k^*+\underline{d})| \quad \text{subject to} \quad \|\underline{d}\| \;\leq\; \max\,[\,0.1\,\Delta_k,\, \rho\,]\,. \tag{2.4}$$

The reason for seeking a large value of $|\ell_t(\underline{x}_k^*+\underline{d}_k)|$ is explained later. It is an extension of the remark that, if $\ell_t(\underline{x}_k^*+\underline{d}_k)$ is nonzero, then the required conditions **(1)** and **(2)**, given in the paragraph that includes expression (1.16), are inherited by the new interpolation points. The choice of the right hand side $\max[0.1\Delta_k, \rho]$ takes into consideration the possibility that Δ_k is much larger than ρ and the objective function has some narrow valleys.

We now address the rules that determine whether an iteration is of trust region or model type, the main difference between these types being that $\underline{d}_k$ is derived from subproblem (1.2) or (2.4), respectively. Another difference is that the integer t of formula (1.18) is chosen before $\underline{d}_k$ in a model iteration, instead of the other way round. Moreover, a few of the trust region iterations do not alter the set of interpolation points. The rules also determine when the calculations with the current value of ρ are complete. Then the iterations are continued with a smaller value of ρ, unless ρ has reached its final value $\rho_{\rm end}$, which is the condition for termination.

A simple version of the rules is presented first. Then we note that it may give a rate of convergence that is highly inefficient, so the description is completed by a modification to the simple version. Rule 1 is that the k-th iteration is not allowed to be a model iteration if the current interpolation points satisfy $\|\underline{x}_i - \underline{x}_k^*\| \le 2\Delta_k$, $i=1,2,\ldots,m$, because the purpose of a model iteration is to improve the position of an interpolation point that is well outside the current trust region. Rule 2 is that every model iteration is followed by a trust region iteration, because we take the optimistic view that the change to $\underline{x}_t$ by the model iteration is going to be helpful. Rule 3 is that, if the k-th iteration is a trust region iteration, then it is always followed by another trust region iteration if $\underline{d}_k$ satisfies $\|\underline{d}_k\| \ge \frac{1}{2}\rho$, and if the new vector of variables achieves the reduction

$$F(\underline{x}_k^* + \underline{d}_k) \;\le\; F(\underline{x}_k^*) - \tfrac{1}{2}\,[\, Q_k(\underline{x}_k^*) - Q_k(\underline{x}_k^* + \underline{d}_k)\,] \tag{2.5}$$

in the objective function. Rule 4 applies to the remaining situation, where the k-th iteration is a trust region iteration and one of the conditions (2.2) holds. Then, if $F(\underline{x}_k^* + \underline{d}_k)$ is calculated, the value of Δ_{k+1} and the new interpolation points are set in the usual way, while, in the alternative case $\|\underline{d}_k\| < \frac{1}{2}\rho$, the lack of a new value of F implies $Q_{k+1} = Q_k$ and $\underline{x}_i^+ = \underline{x}_i$, $i=1,2,\ldots,m$, but $\Delta_{k+1} = \frac{1}{2}\Delta_k$ or $\Delta_{k+1} = \rho$ is set if $\Delta_k > 3\rho$ or $\Delta_k \le 3\rho$, respectively. Further, Rule 4 makes the following choice between three possibilities. The next iteration is a model iteration unless Rule 1 would be violated. The next iteration is a trust region iteration if a model iteration is not allowed and if the current value of $\max[\Delta_{k+1}, \|\underline{d}_k\|]$ exceeds ρ. Otherwise there are no more iterations with the current value of ρ, because the rules imply not only that condition (2.2) holds with $\|\underline{d}_k\| \le \rho$, but also that the interpolation points for the next iteration have the property

$$\|\underline{x}_i^+ - \underline{x}_{k+1}^*\| \;\le\; 2\rho, \qquad i=1,2,\ldots,m. \tag{2.6}$$

After a reduction in ρ, from $\rho_{\rm old}$ to $\rho_{\rm new}$ say, a trust region iteration is performed next, with the value $\Delta_k = \max[\frac{1}{2}\rho_{\rm old}, \rho_{\rm new}]$. The relation $\Delta_k \ge \frac{1}{2}\rho_{\rm old}$ brings the advantage that, if $F(\underline{x}_k^* + \underline{d}_k)$ was not calculated on the previous iteration because of the condition $\|\underline{d}_k\| < \frac{1}{2}\rho_{\rm old}$, then that $\underline{d}_k$ is within the new trust region. We now expose the inefficiency that has been mentioned, by assuming that the current value of $\underline{x}_k^*$ happens to be the optimal vector of variables, which implies that $\underline{x}_k^*$ remains fixed for the remainder of the calculation. On the first iteration after a reduction in ρ, it is usual for every interpolation point $\underline{x}_i$ except $\underline{x}_k^*$ to satisfy $\|\underline{x}_i - \underline{x}_k^*\| > 2\rho_{\rm new}$. We have noted, however, that the new value of ρ is retained until all the conditions (2.6) are achieved. Therefore we expect the number of evaluations of the objective function to be at least $m-1$ for each value of ρ. This situation is particularly intolerable if F is a strictly convex quadratic function and ρ is reduced many times. Indeed, about m function values would be usual for each decimal place of accuracy, although the theorem of Section 1 suggests that the least Frobenius norm updating technique can provide a superlinear rate of convergence.

Any technique that corrects this inefficiency has to give up condition (2.6) occasionally when deciding that the work with the current value of ρ is complete. The algorithm may do so on a trust region iteration if $\underline{d}_k$ satisfies $\|\underline{d}_k\| < \frac{1}{2}\rho$. Then the decision not to calculate $F(\underline{x}_k^* + \underline{d}_k)$ implies a willingness to ignore a reduction in the objective function that has the predicted value

$$Q_k(\underline{x}_k^*) - Q_k(\underline{x}_k^* + \underline{d}_k) \;=\; \tfrac{1}{2}\, \underline{d}_k^T \nabla^2 Q_k\, \underline{d}_k, \qquad (2.7)$$

the right hand side being due to $\underline{\nabla} Q_k(\underline{x}_k^* + \underline{d}_k) = 0$. Expression (2.7) is relatively large, however, if $\underline{d}_k$ is a direction of high curvature of Q_k. Therefore we take the view that the number

$$\varepsilon_k \;=\; \tfrac{1}{2}\, \rho^2 \max\,[0, \lambda_1(\nabla^2 Q_k)] \qquad (2.8)$$

provides an indication of changes to F that can be ignored, where $\lambda_1(\nabla^2 Q_k)$ is an estimate of the least eigenvalue of $\nabla^2 Q_k$, which becomes available from the subproblem (1.2) when $\underline{d}_k$ is selected. Moreover, the differences

$$\eta_j \;=\; F(\underline{x}_j^* + \underline{d}_j) - Q_j(\underline{x}_j^* + \underline{d}_j), \qquad j \in \mathcal{J}, \qquad (2.9)$$

can be used to assess the accuracy of quadratic models, where the iteration number j is in $\mathcal{J}$ if and only if $F(\underline{x}_j^* + \underline{d}_j)$ is calculated. These remarks suggest that a model iteration is unnecessary if recent values of $|\eta_j|$ compare favourably with ε_k. Therefore the modification to the given rules is as follows.

Only Rule 4 is changed. The modification is applied only when $\|\underline{d}_k\| < \frac{1}{2}\rho$ occurs on a trust region iteration, and when there are at least three elements in the set $[1, k-1] \cap \mathcal{J}$. Then the algorithm tests the conditions

$$\max\,[\,|\eta_\alpha|, |\eta_\beta|, |\eta_\gamma|\,] \;<\; \varepsilon_k \qquad \text{and} \qquad \Delta_\alpha = \Delta_\beta = \Delta_\gamma = \Delta_k = \rho, \qquad (2.10)$$

where α, β and γ are the greatest integers in $\mathcal{J}$ that satisfy $\alpha < \beta < \gamma < k$. The new version of Rule 4 states that the calculations with the current value of ρ are complete if all the conditions (2.6) or all the conditions (2.10) are achieved. Otherwise, the next iteration is a model iteration as before. The indices of $\mathcal{J}$ are taken from both trust region and model iterations. The use of three indices in the test (2.10) on the accuracy of recent quadratic models was found to be suitable in numerical experiments, including the examples of Section 3.

On each trust region iteration, the choice of the integer t for formula (1.18) depends on the values $\ell_j(\underline{x}_k^* + \underline{d}_k)$, $j = 1, 2, \ldots, m$, of the Lagrange functions, and also Lagrange functions occur in expressions (1.19) and (2.4). Therefore the coefficients of these functions are calculated and updated by the algorithm, using the following method that is studied by Powell (2002c). Fortunately, $\nabla^2 \ell_j$ has the form

$$\nabla^2 \ell_j \;=\; \sum_{s=1}^m H_{sj}\, (\underline{x}_s - \underline{x}_0)\, (\underline{x}_s - \underline{x}_0)^T, \qquad j=1,2,\ldots,m, \tag{2.11}$$

for any choice of $\underline{x}_0 \in \mathcal{R}^n$, which is derived from the minimization of $\|\nabla^2 \ell_j\|_F$, subject to the Lagrange conditions

$$\ell_j(\underline{x}_i) \;=\; \delta_{ij}, \qquad 1 \le i, j \le m. \tag{2.12}$$

Thus the $m \times m$ matrix H provides all the second derivatives of all the Lagrange functions. Further, the variational problem that gives equation (2.11) also gives the constraints

$$\sum_{s=1}^m H_{sj} = 0 \quad \text{and} \quad \sum_{s=1}^m H_{sj}\, \underline{x}_s = \sum_{s=1}^m H_{sj} (\underline{x}_s - \underline{x}_0) = 0, \quad j=1,2,\ldots,m. \tag{2.13}$$

Therefore $\underline{x}_0$ is irrelevant to the sum (2.11) in exact arithmetic, but the algorithm adjusts $\underline{x}_0$ occasionally, after setting $\underline{x}_0 = \underline{x}_0^*$ initially, because smaller values of $\|\underline{x}_s - \underline{x}_0\|$, $s=1,2,\ldots,m$, reduce the contributions from computer rounding errors to expression (2.11).

Having chosen $\underline{x}_0$, we let ℓ_j be the quadratic polynomial

$$\ell_j(\underline{x}) \;=\; c_j + (\underline{x} - \underline{x}_0)^T \underline{g}_j + \tfrac{1}{2}\, (\underline{x} - \underline{x}_0)^T\, \nabla^2 \ell_j\, (\underline{x} - \underline{x}_0), \qquad \underline{x} \in \mathcal{R}^n, \tag{2.14}$$

which has $m+n+1$ parameters, namely c_j, the components of $\underline{g}_j$, and the coefficients H_{sj}, $s=1,2,\ldots,m$, of formula (2.11). These parameters are constrained by equations (2.12) and (2.13) for the choice of j. By using formulae (2.11) and (2.14), we write the Lagrange conditions (2.12) as the identities

$$c_j + (\underline{x}_i - \underline{x}_0)^T \underline{g}_j + \tfrac{1}{2} \sum_{s=1}^m H_{sj}\, \{(\underline{x}_i - \underline{x}_0)^T (\underline{x}_s - \underline{x}_0)\}^2 \;=\; \delta_{ij}, \quad 1 \le i, j \le m. \tag{2.15}$$

It follows that the parameters of ℓ_j satisfy the square system of linear equations

$$\left(\begin{array}{c|c} A & \underline{e} \quad X^T \\ \hline \underline{e}^T & \\ X & 0 \end{array} \right) \left(\begin{array}{c} H_{1j} \\ \vdots \\ H_{mj} \\ \hline c_j \\ \underline{g}_j \end{array} \right) \;=\; W \left(\begin{array}{c} H_{1j} \\ \vdots \\ H_{mj} \\ \hline c_j \\ \underline{g}_j \end{array} \right) \;=\; \underline{e}_j, \tag{2.16}$$

where A is the $m \times m$ matrix with the elements $\frac{1}{2}\{(\underline{x}_i - \underline{x}_0)^T (\underline{x}_s - \underline{x}_0)\}^2$, $1 \le i, s \le m$, where $\underline{e}$ is the vector of ones in $\mathcal{R}^m$, where X is the $n \times m$ matrix with the columns $\underline{x}_i - \underline{x}_0$, $i=1,2,\ldots,m$, where W is defined to be the $(m+n+1) \times (m+n+1)$ matrix on the left hand side, and where $\underline{e}_j$ on the right hand side is the j-th coordinate vector in $\mathcal{R}^{m+n+1}$. It can be shown that W, which is symmetric and independent of j, is nonsingular, because of conditions **(1)** and **(2)** of Section 1. Therefore the parameters of ℓ_j are the elements of the j-th column of W^{-1}. The algorithm generates all the elements of W^{-1} explicitly, so from now on we change the definition of H to $H = W^{-1}$, which preserves formula (2.11). We note that H is symmetric. Moreover, the matrix H for the first iteration is constructed in only $\mathcal{O}(m^2)$ operations, by selecting $\underline{x}_0 = \underline{x}_0^*$ and by taking advantage of the sparsity in W that is given by the first set of interpolation points, where we are recalling the procedure of the third paragraph of this section.

Another advantage of the availability of H is that, for any $\underline{x}$ in $\mathcal{R}^n$, all the values $\ell_i(\underline{x})$, $i = 1, 2, \ldots, m$, can also be calculated in only $\mathcal{O}(m^2)$ operations. Specifically, we let $\underline{v} = \underline{v}(\underline{x})$ be the vector in $\mathcal{R}^n$ that has the components

$$\left.\begin{array}{rl} v_s = \frac{1}{2}\{(\underline{x}-\underline{x}_0)^T(\underline{x}_s-\underline{x}_0)\}^2, & s=1,2,\ldots,m, \\ v_{m+1} = 1 \quad \text{and} \quad v_{m+i+1} = (\underline{x}-\underline{x}_0)_i, & i=1,2,\ldots,n. \end{array}\right\} \tag{2.17}$$

Then, because of equations (2.11) and (2.14), the required numbers $\ell_i(\underline{x})$, $i = 1, 2, \ldots, m$, are the first m components of the product $H\underline{v}$. For example, if $\underline{x}$ is the interpolation point $\underline{x}_j$, then $\underline{v} = \underline{v}(\underline{x}_j)$ is the j-th column of W, so the construction provides the vector $H\underline{v} = HW\underline{e}_j = \underline{e}_j$, which agrees with the Lagrange conditions $\ell_i(\underline{x}_j) = \delta_{ij}$, $i = 1, 2, \ldots, m$.

The matrix H is updated as follows when the change (1.18) is made to the interpolation points. Let W^+ and $H^+ = (W^+)^{-1}$ be the new versions of W and H. Fortunately, because only the interpolation point $\underline{x}_t$ is moved, the nonzero elements of the difference $W^+ - W$ are confined to its t-th row and column. Hence the rank of $W^+ - W$ is at most two, so the calculation of H^+ from H requires only $\mathcal{O}(m^2)$ operations. We put $\underline{x} = \underline{x}_t^+ = \underline{x}_k^* + \underline{d}_k$ into expression (2.17), because the resultant vector $\underline{v} = \underline{v}(\underline{x}_k^* + \underline{d}_k)$ is the t-th column of W^+, except for the diagonal element

$$W_{tt}^+ \;=\; \tfrac{1}{2}\, \|\underline{x}_t^+ - \underline{x}_0\|^4 \;=\; \tfrac{1}{2}\, \|\underline{x}_k^* + \underline{d}_k - \underline{x}_0\|^4. \tag{2.18}$$

Further, according to formula (4.14) of Powell (2002c), H^+ is the matrix

$$\begin{aligned} H^+ \;=\; H + \frac{1}{\sigma_t}\Big[& \alpha_t\, (\underline{e}_t - H\,\underline{v})\, (\underline{e}_t - H\,\underline{v})^T - \beta_t\, H\,\underline{e}_t\, \underline{e}_t^T H \\ & + \tau_t \Big\{ H\,\underline{e}_t\, (\underline{e}_t - H\,\underline{v})^T + (\underline{e}_t - H\,\underline{v})\, \underline{e}_t^T H \Big\}\Big], \end{aligned} \tag{2.19}$$

where the parameters α_t, β_t, τ_t and σ_t are given by the notation

$$\left.\begin{array}{ll} \alpha_i = \underline{e}_i^T H\,\underline{e}_i = H_{ii}, & \beta_i = \frac{1}{2}\, \|\underline{x}_k^* + \underline{d}_k - \underline{x}_0\|^4 - \underline{v}^T H\,\underline{v} \\ \tau_i = \underline{e}_i^T H\,\underline{v} = \ell_i(\underline{x}_k^* + \underline{d}_k) & \text{and} \quad \sigma_i = \alpha_i \beta_i + \tau_i^2 \end{array}\right\}, \quad i = 1, 2, \ldots, m. \tag{2.20}$$

Equation (2.19) implies that W^+ inherits nonsingularity from W if σ_t is nonzero, and then conditions **(1)** and **(2)** are satisfied by the new interpolation points. The property $\sigma_t > 0$ can usually be achieved without difficulty, because σ_t is defined to be $\alpha_t\beta_t + \tau_t^2$, and α_t and β_t are both nonnegative in exact arithmetic. Therefore a large value of $|\tau_t| = |\ell_t(\underline{x}_k^* + \underline{d}_k)|$ is welcome, which is the reason for deriving $\underline{d}_k$ from the subproblem (2.4) on a model iteration.

We recall, however, that a trust region iteration chooses $\underline{d}_k$ before t. A large value of $|\ell_t(\underline{x}_k^* + \underline{d}_k)|$ is still attractive. On the other hand, we may wish to drop the interpolation condition $Q_k(\underline{x}_t) = F(\underline{x}_t)$ if $\underline{x}_t$ is relatively far from $\underline{x}_{k+1}^*$, because the new quadratic model is going to be used in the region $\{\underline{x} : \|\underline{x} - \underline{x}_{k+1}^*\| \leq \Delta_{k+1}\}$. Therefore we consider $|\ell_t(\underline{x}_k^* + \underline{d}_k)|$ when $\|\underline{x}_t - \underline{x}_k^*\|$ is much greater than Δ_k, but we assume that most of the interpolation points $\underline{x}_i$ satisfy $\|\underline{x}_i - \underline{x}_k^*\| \leq \Delta_k$. In general, this assumption causes $\|\underline{\nabla}\ell_t(\underline{x}_k^*)\|$ to be relatively small. Therefore the approximate behaviour of ℓ_t on the line segment from $\underline{x}_k^*$ to $\underline{x}_t$ is that it grows quadratically from zero to one. Thus, in the usual case $\|\underline{d}_k\| = \Delta_k$, we expect $|\ell_t(\underline{x}_k^* + \underline{d}_k)|$ to be of magnitude $(\Delta_k / \|\underline{x}_t - \underline{x}_k^*\|)^2$, which is much less than one, but the Lagrange functions have the elementary property

$$\sum_{j=1}^{m} \ell_j(\underline{x}_k^* + \underline{d}_k) \;=\; \sum_{j=1}^{m} \ell_j(\underline{x}) \;=\; 1, \qquad \underline{x} \in \mathcal{R}^n. \tag{2.21}$$

It follows that a relatively small value of $|\tau_t| = |\ell_t(\underline{x}_k^* + \underline{d}_k)|$ may be advantageous occasionally. Therefore, when t is required by a trust region iteration, its choice depends on the products $\omega_i\tau_i$, $i = 1, 2, \ldots, m$, where ω_i is the weight

$$\omega_i \;=\; \{\, \max[\,1,\ \|\underline{x}_i - \underline{x}_k^*\| \,/\, \max(0.1\Delta_k, \rho)\,]\,\}^3, \qquad i = 1, 2, \ldots, m. \tag{2.22}$$

The purpose of the cube is to assist the removal of remote interpolation points, and the term $\max(0.1\Delta_k, \rho)$ is taken from the problem (2.4). The algorithm also pays attention to the $\alpha_t\beta_t$ part of the denominator σ_t of formula (2.19). Specifically, t is the value of i that provides the largest of the products

$$\omega_i^2\, |\sigma_i| \;=\; \omega_i^2\, |\alpha_i\,\beta_i + \tau_i^2|, \qquad i = 1, 2, \ldots, m, \tag{2.23}$$

except that the best interpolation point so far is never discarded. This technique for selecting t also requires $\mathcal{O}(m^2)$ operations, because the vector $\underline{v} = \underline{v}(\underline{x}_k^* + \underline{d}_k)$ and the parameter β_i of expression (2.20) are independent of i.

Another task that is important in practice is the changes that are made to the vector $\underline{x}_0$ of expression (2.14), in order to reduce the effects of rounding errors. The author has been studying this question for about six months, without finding an answer that is satisfactory in difficult cases. Fortunately, however, the current version of the algorithm has produced the very promising numerical results that are presented and discussed in the next section. All of the techniques that we have considered may be included in the final version of the algorithm. They require only $\mathcal{O}(n^2)$ computer operations per iteration when $m = 2n+1$, for example. Therefore further attention will also be given to the generation of $\underline{d}_k$ from the subproblem (1.2), because the cost of that calculation at present is $\mathcal{O}(n^3)$.

3. Numerical results and conclusions

The author had intended to present numerical results for several test problems at the Hangzhou conference in December, 2002, but, as mentioned already, the development of the algorithm of Section 2 is not yet complete. Further, he switched from development to producing numerical results less than one week before leaving Cambridge for that conference. In the development work the number of variables n was only three, but in the experiments the range of n is from $n=10$ to $n=160$. Fortunately, most of the numbers of iterations and all the final accuracies remained satisfactory when n was increased in this way, so no modifications were made in December to the current version of the algorithm. The development work with $n=3$ has been resumed since the conference, however, and it has caused a few recent changes, which are included in the details of Section 2. Therefore, because the results of this section were calculated by the newer version of the algorithm, the entries in the tables below are different from the similar numerical results that were presented in Hangzhou. These differences are unimportant to the main conclusion of this section.

Only one test problem with more than three variables has been tried so far, because the author is still determined to reduce the damage from computer rounding errors in difficult cases with $n=3$. This test problem has also been used to demonstrate some features of the UOBYQA software (Powell, 2002a). Its objective function has the form

$$F(\underline{x}) \;=\; \sum_{i=1}^{2n} \Big\{ b_i - \sum_{j=1}^{n} \Big(S_{ij} \sin(\theta_j x_j) + C_{ij} \cos(\theta_j x_j) \Big) \Big\}^2 , \qquad \underline{x} \in \mathcal{R}^n, \tag{3.1}$$

where b_i, S_{ij}, C_{ij} and θ_j are parameters whose values depend on random numbers. Specifically, each element of the $2n \times n$ matrices S and C is picked independently from the uniform distribution of random integers on $[-100, 100]$, each scaling factor θ_j, $j=1,2,\ldots,n$, is from the logarithmic distribution of random numbers on $[0.1, 1]$, and the parameters b_i, $i=1,2,\ldots,2n$, are defined by the condition $F(\underline{x}^*)=0$, where $\underline{x}^*$ has the components $x_j^* = \hat{x}_j^*/\theta_j$, $j=1,2,\ldots,n$, the numerators $\hat{x}_j^*$, $j=1,2,\ldots,n$, being picked independently from the uniform distribution on $[-\pi, \pi]$. Thus $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, is bounded and periodic, with a least value of zero, which occurs at $\underline{x} = \underline{x}^*$. The presence of $2n$ terms in the sum of squares (3.1) causes $\nabla^2 F(\underline{x}^*)$ to be well-conditioned in general, the condition number being about 100, because of the factors θ_j, $j=1,2,\ldots,n$. The initial vector of variables, namely $\underline{x}_0^*$, is given the components $(\hat{x}_j^* + 0.1 \hat{y}_j^*)/\theta_j$, $j=1,2,\ldots,n$, where each $\hat{y}_j^*$ is also from the uniform distribution on $[-\pi, \pi]$. Further, the values of ρ_{beg} and ρ_{end} are set to 10^{-1} and 10^{-6}, respectively, and the choices of m and n are stated below.

The main purpose of the numerical experiments is to investigate the dependence on m and n of the total numbers of function evaluations that occur. Therefore we let n take the values 10, 20, 40, 80 and 160, and we compare $m=2n+1$ with

m	n	Counts of F values	Final values of F	Times in seconds
21	10	334 – 494	$1\times d^{-10}$ – $4\times d^{-10}$	0.37 – 0.55
37	10	276 – 361	$3\times d^{-11}$ – $7\times d^{-11}$	0.50 – 0.66
66	10	209 – 281	$1\times d^{-12}$ – $7\times d^{-11}$	0.67 – 0.91
41	20	1018 – 1290	$2\times d^{-9}$ – $3\times d^{-9}$	5.4 – 6.8
97	20	907 – 1148	$2\times d^{-10}$ – $3\times d^{-9}$	10.7 – 13.3
231	20	606 – 792	$2\times d^{-12}$ – $3\times d^{-10}$	18.7 – 25.6
81	40	2071 – 2408	$9\times d^{-9}$ – $6\times d^{-8}$	57 – 65
264	40	2103 – 2741	$2\times d^{-9}$ – $3\times d^{-8}$	198 – 264
861	40	3652 – 12222	$1\times d^{-10}$ – $6\times d^{-9}$	1998 – 7340
161	80	3922 – 4254	$2\times d^{-8}$ – $2\times d^{-7}$	667 – 735
731	80	4794 – 4989	$2\times d^{-8}$ – $3\times d^{-8}$	4869 – 5264
321	160	7621 – 8150	$1\times d^{-7}$ – $3\times d^{-6}$	9284 – 10046

Table 1: Numerical results for objective functions of the form (3.1)

$m=\frac{1}{2}(n+1)(n+2)$, because the first choice requires much less work per iteration, and in the second case the interpolation conditions take up all the freedom in every quadratic model, which may reduce the number of iterations. Further, the intermediate value of m in Table 1 is about the geometric mean of the other two values, because this choice may be suitable when the work per iteration is $\mathcal{O}(n^3)$. The largest m is dropped for $n=80$, however, and only the smallest m is tried with $n=160$, in order to avoid very long computations. We generate five objective functions of the form (3.1) for each n by choosing different random numbers, and also one starting point $\underline{x}_0^*$ is picked automatically for each F. Thus the same five minimization problems are solved for the different values of m, which provides five minimization calculations for each row of Table 1. The headings of the table state the quantities that are recorded for each of these problems, and the main part of the table gives the least and greatest values of these quantities for the five calculations that have been mentioned. Here a "Counts of F values" figure is the total number of evaluations of F by the algorithm in one experiment, the notation $\alpha\times d^{-\beta}$ stands for $\alpha\times 10^{-\beta}$, and every number in the last column is the total computation time in seconds for the work of one application of the algorithm. All the calculations were run in Fortran on a Sun Ultra 10 workstation.

The entries in the $n=10$ and $n=20$ rows of Table 1 are not very interesting. Indeed, the third column suggests that an increase in m reduces the number of evaluations of the objective function, the fourth column shows that good accuracy is achieved, a typical value of F for random variables being about $n^2\times 10^4$, and the timings in the last column indicate that smaller values of m are more efficient if the calculation of $F(\underline{x})$ for any $\underline{x}$ requires only $\mathcal{O}(n^2)$ operations. A remarkable feature of the $n=40$ rows, however, is that not only the least computation times but also the fewest numbers of function evaluations occur for the smallest choice of m. This observation receives further attention below. We note also that, for $n=40$ and $m=861$, the range 3652–12222 of the "Counts of F values" is huge. This remark is embarrassing to the author, because he has found that these figures are highly sensitive to changes in the Fortran implementation of the algorithm, even when the changes would not alter the results in exact arithmetic, which provides another good reason for the current research on reducing the damage from computer rounding errors. Further, the UOBYQA algorithm is analogous to our method with $m=\frac{1}{2}(n+1)(n+2)$, but, when UOBYQA was applied to five cases of the test problem of this section with $n = 40$, in order to calculate one of the results in Table 4 of Powell (2002b), the total numbers of function values ranged from 1970 to only 2275.

The ranges of the "Counts of F values" in the other rows of our table are so benign, however, that their contributions from computer rounding errors may be negligible in comparison to the variances that arise from the randomness in the construction of F. We can also take the view that, if rounding errors have a strong influence on the numbers of iterations, then an increase in the amount of work is more likely than a decrease. Therefore the given results suggest that, when $m=2n+1$ is chosen and n is large, then the solution of our test problem requires at most about $50n$ values of F. Further, regardless of any assumptions, we see that five unconstrained minimization calculations with 160 variables have been solved, and that in each case the objective function was calculated no more than 8150 times. On the other hand, UOBYQA would require 13041 values of F to generate its first quadratic model, because a quadratic function of 160 variables has this number of independent parameters. Therefore, although the algorithm is not in its final form, we can conclude already that, instead of seeking accuracy in the approximation $\nabla^2 Q_k \approx \nabla^2 F$, it may be much more efficient to update $\nabla^2 Q_k$ by the least Frobenius norm technique of this paper.

Our discussion of the theorem in Section 1 also suggests that fewer than $\mathcal{O}(n^2)$ values of the objective function may be sufficient to achieve high precision in unconstrained calculations without derivatives. Indeed, we recall that the limit (1.13), which is due to equation (1.11), does not require $\|\nabla^2 Q_k - \nabla^2 F\|_F$ to tend to zero. Therefore, when convergence properties are deduced from the limit, the number of parameters of a quadratic model is irrelevant. Further, if n is huge, then the value of n itself may also be irrelevant, except for the construction of the first quadratic model. It follows from these remarks that it may become possible to calculate least values of functions of hundreds of variables to high accuracy when derivatives are not available.

Acknowledgements

This paper was written at the University of Minnesota, where the author participated in the IMA Program on Optimization for two months, and was a Senior Ordway Visitior in the School of Mathematics. He is very grateful for the excellent facilities, hospitality and support that he received. He also thanks John Dennis for several helpful comments on the first draft of this work.

References

C.G. Broyden, J.E. Dennis and J.J. Moré (1973), “On the local and superlinear convergence of quasi-Newton methods”, *J. Inst. Math. Appl.*, Vol. 12, pp. 223–245.

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

J.E. Dennis and R.B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall (Englewood Cliffs).

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

J.J. Moré and D.C. Sorensen (1983), “Computing a trust region step”, *SIAM J. Sci. Stat. Comput.*, Vol. 4, pp. 553–572.

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

M.J.D. Powell (2002b), “On trust region methods for unconstrained minimization without derivatives”, Report No. DAMTP 2002/NA02, University of Cambridge.

M.J.D. Powell (2002c), “Least Frobenius norm updating of quadratic models that satisfy interpolation conditions”, Report No. DAMTP 2002/NA08, University of Cambridge.