DAMTP 2000/NA14

UOBYQA: unconstrained optimization by quadratic approximation

M.J.D. Powell

Abstract: UOBYQA is a new algorithm for general unconstrained optimization calculations, that takes account of the curvature of the objective function, F say, by forming quadratic models by interpolation. Therefore, because no first derivatives are required, each model is defined by $\frac{1}{2}(n+1)(n+2)$ values of F, where n is the number of variables, and the interpolation points must have the property that no nonzero quadratic polynomial vanishes at all of them. A typical iteration of the algorithm generates a new vector of variables, $\underline{\widetilde{x}}_t$ say, either by minimizing the quadratic model subject to a trust region bound, or by a procedure that should improve the accuracy of the model. Then usually $F(\underline{\widetilde{x}}_t)$ is obtained, and one of the interpolation points is replaced by $\underline{\widetilde{x}}_t$. Therefore the paper addresses the initial positions of the interpolation points, the adjustment of trust region radii, the calculation of $\underline{\widetilde{x}}_t$ in the two cases that have been mentioned, and the selection of the point to be replaced. Further, UOBYQA works with the Lagrange functions of the interpolation equations explicitly, so their coefficients are updated when an interpolation point is moved. The Lagrange functions assist the procedure that improves the model, and also they provide an estimate of the error of the quadratic approximation to F, which allows the algorithm to achieve a fast rate of convergence. These features are discussed and a summary of the algorithm is given. Finally, a Fortran implementation of UOBYQA is applied to several choices of F, in order to investigate accuracy, robustness in the presence of rounding errors, the effects of first derivative discontinuities, and the amount of work. The numerical results are very promising for $n \leq 20$, but larger values are problematical, because the routine work of an iteration is of fourth order in the number of variables.

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

December, 2000 (revised June, 2001).

1. Introduction

This paper describes the techniques that are used by the Fortran 77 software, namely UOBYQA, that the author has developed recently for unconstrained optimization calculations, when first derivatives of the objective function are not available. We use the notation $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, for the objective function. It is specified by a subroutine that calculates $F(\underline{x})$ for any vector of variables $\underline{x}$ in $\mathcal{R}^n$. The user also has to provide an initial vector of variables, $\underline{x}_b$ say, and initial and final values, $\rho_{\rm beg}$ and $\rho_{\rm end}$ say, of a trust region radius ρ. Values of ρ are chosen automatically that satisfy $\rho_{\rm beg} \geq \rho \geq \rho_{\rm end}$. For each one, typical distances between successive points at which F is calculated are of magnitude ρ, and ρ is not reduced until the objective function stops decreasing for such changes to the variables. Thus a highly useful feature of the software is that it is suitable for noisy objective functions. Indeed, the distances between the points provide some control over the contributions from the noise to estimates of first and second derivatives of F. On the other hand, noise is very harmful to algorithms that calculate derivative estimates from the differences between function values that occur when small changes are made to the variables.

Our derivative estimates are contained in a quadratic model

$$Q(\underline{x}) = c_Q + \underline{g}_Q^T(\underline{x}-\underline{x}_b) + \tfrac{1}{2}\,(\underline{x}-\underline{x}_b)^T G_Q\,(\underline{x}-\underline{x}_b), \qquad \underline{x} \in \mathcal{R}^n, \tag{1.1}$$

that is constructed by interpolation to values of the objective function. We let $\underline{x}_b$ be the initial vector that has been mentioned already in the case $\rho = \rho_{\rm beg}$, but, whenever ρ is reduced, $\underline{x}_b$ is changed to the vector of variables of the least calculated value of F so far. Here we are trying to prevent damage from computer rounding errors in the computation of $Q(\underline{x})$, by picking $\underline{x}_b$ so that $\|\underline{x}-\underline{x}_b\|$ becomes small for relevant vectors $\underline{x}$. Thus $\underline{x}_b$ is available when Q is constructed, so the parameters of the quadratic model are the real number c_Q, the vector $\underline{g}_Q \in \mathcal{R}^n$, and the $n \times n$ symmetric matrix G_Q. Hence the linear space of quadratic polynomials from $\mathcal{R}^n$ to $\mathcal{R}$ has dimension $m = \frac{1}{2}(n+1)(n+2)$. Therefore the parameters of Q are defined by a system of interpolation equations of the form

$$Q(\underline{x}_i) = F(\underline{x}_i), \qquad i = 1, 2, \ldots, m, \tag{1.2}$$

where the points $\underline{x}_i$, $i = 1, 2, \ldots, m$, are generated automatically by the algorithm in ways that will be specified. The UO part of the name UOBYQA denotes unconstrained optimization, BY is just by, and QA denotes quadratic approximation, because of the importance of the model (1.1).

For reasons given by Powell (2000), which will be obvious later, we require the Lagrange functions of the interpolation problem (1.2). For $j = 1, 2, \ldots, m$, the j-th Lagrange function is the quadratic polynomial ℓ_j from $\mathcal{R}^n$ to $\mathcal{R}$ that has the properties

$$\ell_j(\underline{x}_i) = \delta_{ij}, \qquad i = 1, 2, \ldots, m, \tag{1.3}$$

where δ_{ij} is the Kronecker delta. Our notation for its parameters is shown in the expression

$$\ell_j(\underline{x}) = c_j + \underline{g}_j^T(\underline{x}-\underline{x}_b) + \tfrac{1}{2}(\underline{x}-\underline{x}_b)^T G_j(\underline{x}-\underline{x}_b), \qquad \underline{x}\in\mathcal{R}^n, \tag{1.4}$$

which is analogous to equation (1.1). Because conditions (1.2) and (1.3) imply the identity

$$Q(\underline{x}) = \sum_{j=1}^{m} F(\underline{x}_j)\,\ell_j(\underline{x}), \qquad \underline{x}\in\mathcal{R}^n, \tag{1.5}$$

it follows from expression (1.4) that the parameters of Q have the values

$$c_Q = \sum_{j=1}^{m} F(\underline{x}_j)\,c_j, \quad \underline{g}_Q = \sum_{j=1}^{m} F(\underline{x}_j)\,\underline{g}_j \quad \text{and} \quad G_Q = \sum_{j=1}^{m} F(\underline{x}_j)\,G_j. \tag{1.6}$$

The Fortran software works explicitly with all the first and second derivative parameters of all the Lagrange functions, and also $\underline{g}_Q$ and G_Q are constructed, but there is no need to retain c_Q and c_j, $j=1,2,\ldots,m$.

The quadratic model is used in a trust region calculation. Specifically, $\underline{d}\in\mathcal{R}^n$ is set to an estimate of the solution of the problem

$$\text{minimize} \quad Q(\underline{x}_k+\underline{d}) \quad \text{subject to} \quad \|\underline{d}\| \le \Delta, \tag{1.7}$$

where k is the integer in $[1,m]$ such that $F(\underline{x}_k)$ is the least of the values $F(\underline{x}_i)$, $i=1,2,\ldots,m$, any ties being broken by preferring the $F(\underline{x}_k)$ that was calculated first. Further, the vector norm of the problem (1.7) is Euclidean, and Δ is another trust region radius that satisfies $\Delta\ge\rho$. The advantage of introducing Δ is to allow the lengths of the changes to the variables to exceed ρ, which helps to avoid some loss of efficiency that may occur otherwise, because ρ is never increased by the algorithm. We will find in Section 3 that Δ is adjusted in a way that is typical for trust region methods.

We do not allow ρ to increase, because increases would necessitate more decreases later, and usually it is onerous to make decreases, due to some tests that have to be satisfied. These tests respond to the question, raised in the opening paragraph, whether the objective function has stopped decreasing for the current value of ρ. We introduce the tests by considering the situation when $\underline{d}$ solves the problem (1.7) with $\Delta=\rho$, but the calculation of the objective function at $\underline{x}_k+\underline{d}$ reveals $F(\underline{x}_k+\underline{d})>F(\underline{x}_k)$. Then ρ should be reduced if Q is a good approximation to F in the region $\{\underline{x} : \|\underline{x}-\underline{x}_k\| \le \rho\}$, but we doubt the goodness of Q if one or more of the distances $\|\underline{x}_i-\underline{x}_k\|$, $i=1,2,\ldots,m$, is greater than 2ρ. Therefore the tests may require the interpolation point $\underline{x}_j$, say, to be moved into the neighbourhood $\{\underline{x} : \|\underline{x}-\underline{x}_k\| \le \rho\}$ before ρ is reduced. We call such a move a model step. It is usually followed by a step to $\underline{x}_k+\underline{d}$, where $\underline{d}$ is defined by the trust region calculation (1.7) for the current value of ρ.

The description of the details of the algorithm is divided into three sections. Two trust region subproblems have to be solved, and they are the subject of Section 2. One of them is the computation (1.7), and the other one is the problem

$$\text{maximize} \quad |\ell_j(\underline{x}_k+\underline{d})| \quad \text{subject to} \quad \|\underline{d}\| \leq \rho, \tag{1.8}$$

where ℓ_j is the Lagrange function (1.4). The reason for this calculation is that the resultant value of $\underline{x}_k+\underline{d}$ is a suitable new position for the interpolation point $\underline{x}_j$, when it is moved by a model step. Indeed, this choice maintains and assists the nonsingularity of the interpolation equations (1.2) (see Powell, 2000). After the initial quadratic model has been formed, which requires m function evaluations, then each vector of variables for a new value of F is generated by one of the two trust region subproblems. The ways of choosing between these alternatives, and of deciding which interpolation point to move in the case of a model step, are explained in Section 3. The adjustments of the trust region radii are also specified there. Then Section 4 describes the initialization procedure that provides the first m interpolation points, their Lagrange functions, and the first quadratic model. Another topic of Section 4 is the updating of the coefficients of the Lagrange functions and the quadratic model when an interpolation point is moved. These moves include not only model steps, but also the replacement of an interpolation point by $\underline{x}_k+\underline{d}$ after the calculation of $F(\underline{x}_k+\underline{d})$, where $\underline{d}$ is the solution of the problem (1.7). A summary of the complete algorithm is given in Section 5, with a few comments on its implementation. Finally, the performance of the algorithm in practice is shown by some numerical experiments that are presented and discussed in Section 6.

The idea of forming quadratic models by interpolation for optimization without derivatives was proposed by Winfield (1973). The present work is a development of a method that uses linear polynomial models for constrained calculations (Powell, 1994). An early version of this development is addressed in a survey by the author (Powell, 1998) of direct search methods for unconstrained optimization. Some of our techniques can also be found in the algorithm of Conn, Scheinberg and Toint (1997a, 1997b), but that algorithm does not employ Lagrange functions. Furthermore, the UOBYQA software includes a new way of achieving a fast rate of convergence, by making use of the bound on the error of the quadratic model that is given by Powell (2000).

2. The two trust region subproblems

The present version of UOBYQA does not take advantage of any sparsity in second derivatives of the objective function. Therefore we allow the solution of the subproblem (1.7) to employ $\mathcal{O}(n^3)$ computer operations, which is not excessive as the total number of coefficients of the Lagrange functions is $\mathcal{O}(n^4)$, and they are all updated when an interpolation point is moved. Specifically, we apply the trust region method of Moré and Sorensen (1983), because of its control of the accuracy that is achieved. Some details of the implementation of that method by the UOBYQA software are given in the first half of this section.

The vector $\underline{h}_Q = \underline{g}_Q + G_Q(\underline{x}_k - \underline{x}_b)$ is calculated, because expression (1.1) implies the identity

$$Q(\underline{x}_k + \underline{d}) = Q(\underline{x}_k) + \underline{h}_Q^T \underline{d} + \tfrac{1}{2} \underline{d}^T G_Q \underline{d}, \qquad \underline{d} \in \mathcal{R}^n. \tag{2.1}$$

It follows from the KKT conditions of the subproblem (1.7) that $\underline{d}$ should satisfy the equation

$$(G_Q + \theta I)\, \underline{d} = -\underline{h}_Q, \tag{2.2}$$

where I is the $n \times n$ unit matrix, and where θ is a nonnegative number such that $G_Q + \theta I$ is positive definite or semi-definite. Further, if θ is positive, then $\|\underline{d}\| = \Delta$ should hold. Therefore the method of Moré and Sorensen may require the solution of the system (2.2) for several trial values of θ.

This task demands very little effort if the matrix G_Q is tridiagonal. Moreover, the equations (2.2) are equivalent to the system

$$(\Omega\, G_Q \Omega^T + \theta I)\, (\Omega\, \underline{d}) = -\Omega\, \underline{h}_Q, \tag{2.3}$$

where Ω is any $n \times n$ orthogonal matrix. Therefore in the trust region calculation G_Q and $\underline{h}_Q$ are overwritten by $\Omega\, G_Q \Omega^T$ and $\Omega\, \underline{h}_Q$, respectively, where Ω is chosen to make the new G_Q tridiagonal. Finally, after calculating an acceptable $\underline{d}$ from the new equations (2.2), this $\underline{d}$ is Ω times the required $\underline{d}$, so it is overwritten by $\Omega^T \underline{d}$, which preserves the constraint $\|\underline{d}\| \le \Delta$ because Ω is orthogonal. The choice of Ω, the reduction of G_Q to tridiagonal form, and the multiplications by Ω and Ω^T can all be done in $\mathcal{O}(n^3)$ computer operations (see Section 7-4 of Parlett, 1980, for instance).

After making the system (2.2) tridiagonal, the algorithm tries $\theta = 0$. If a Cholesky factorization shows that G_Q is positive definite, and if the resultant $\underline{d}$ satisfies $\|\underline{d}\| \le \Delta$, then the required $\underline{d}$ has been found. In this case, however, an estimate of the least eigenvalue of G_Q, λ_Q say, may be needed later. Therefore λ_Q is computed with a relative error of at most 0.01, using the tridiagonal form of G_Q.

Otherwise, the final $\underline{d}$ will have the property $\|\underline{d}\| = \Delta$. Let $\underline{d} = \underline{d}(\theta)$ denote the solution of the system (2.2) for any θ such that $G_Q + \theta I$ is positive definite. Then $\|\underline{d}(\theta)\|$ decreases monotonically as θ increases, and $\underline{d} = \underline{d}(\theta)$ is acceptable if θ satisfies the equation

$$1 \,/\, \|\underline{d}(\theta)\| \;=\; 1 \,/\, \Delta. \tag{2.4}$$

Useful properties of this form are that, if we regard $1/\|\underline{d}(\theta)\|$ as a function of θ for $\theta > -\lambda_Q$, then it increases monotonically, it is concave, and its first derivative is bounded below by $1/\|\underline{h}_Q\|$. The algorithm takes advantage of these properties in a version of the rule of false position for adjusting θ, that tries to find a solution of the nonlinear equation (2.4).

Now Lemma 3.4 of Moré and Sorensen (1983) shows that, if the optimal $\underline{d}$ is on the trust region boundary, if θ has any value greater than $-\lambda_Q$, if $\underline{z}$ is any vector in $\mathcal{R}^d$ such that $\|\underline{d}(\theta)+\underline{z}\|=\Delta$, and if η is the ratio

$$\eta = \underline{z}^T (G_Q + \theta I)\, \underline{z} \Big/ [\, \underline{d}(\theta)^T (G_Q + \theta I)\, \underline{d}(\theta) + \theta\, \Delta^2\,], \tag{2.5}$$

then $\widehat{\underline{d}}=\underline{d}(\theta)+\underline{z}$ achieves the condition

$$Q(\underline{x}_k+\widehat{\underline{d}}) - Q(\underline{x}_k) \;\le\; (1-\eta)\; \min\{Q(\underline{x}_k+\underline{d}) - Q(\underline{x}_k) : \|\underline{d}\| \le \Delta\}. \tag{2.6}$$

In other words, if η is small, then the reduction in Q that occurs when the vector of variables is changed from $\underline{x}_k$ to $\underline{x}_k+\widehat{\underline{d}}$ is close to the greatest reduction that is allowed by the trust region constraint, the relative difference between these reductions being bounded above by η. Therefore UOBYQA accepts $\widehat{\underline{d}}$ as a suitable $\underline{d}$ if the ratio (2.5) is at most 0.01.

Only two choices of $\underline{z}$ are considered when this technique is applied. Usually $\underline{z}$ is a multiple of $\underline{d}(\theta)$, so $\widehat{\underline{d}}$ is the vector

$$\widehat{\underline{d}} = \Delta\; \underline{d}(\theta) \,/\, \|\underline{d}(\theta)\|, \tag{2.7}$$

and the condition for ending the trust region calculation simplifies to the inequality

$$\left(\frac{\Delta}{\|\underline{d}(\theta)\|} - 1 \right)^2 \le\; 0.01 \left(1 + \frac{\theta\, \Delta^2}{\underline{d}(\theta)^T (G_Q + \theta I)\, \underline{d}(\theta)} \right). \tag{2.8}$$

A different $\underline{z}$ may be necessary for good efficiency, however, in the hard case when $G_Q+\theta I$ has to be nearly (or exactly) singular. Then the Cholesky factorization of $G_Q+\theta I$ may break down due to a negative pivot for some of the values of θ that are used. When this happens, a vector $\underline{v}$ is constructed with the property $\underline{v}^T (G_Q+\theta I) \underline{v} < 0$, that tends to be an eigenvector of G_Q with eigenvalue λ_Q as $\theta \to -\lambda_Q$. A multiple of this vector is the alternative choice of $\underline{z}$. Numerical experiments have confirmed that it is very suitable for achieving the termination condition $\eta \le 0.01$ in pathologically hard cases.

The other trust region subproblem (1.8) can be solved by two applications of the method that has just been considered, because we can put $Q=\ell_j$ and then $Q=-\ell_j$ in the calculation (1.7) with $\Delta=\rho$. The amount of work of this approach, however, may be unacceptable. Indeed, $\underline{x}_j$ is a candidate for a move by a model step when the subproblem (1.8) occurs, but the move is taken only if the resultant value of $|\ell_j(\underline{x}_k+\underline{d})|$ is sufficiently large. Thus several values of j may be tried before a move is made, and there are about $\frac{1}{2}n^2$ possible choices of j.

Fortunately, a crude solution of the subproblem is adequate. Therefore UOBYQA applies a procedure that makes $|\ell_j(\underline{x}_k+\underline{d})|$ relatively large subject to $\|\underline{d}\|\le\rho$ in only $\mathcal{O}(n^2)$ operations, as described in the remainder of this section.

The integer j in the subproblem (1.8) is always different from k, so the Lagrange conditions (1.3) include $\ell_j(\underline{x}_k)=0$. Therefore, recalling the notation (1.4), and letting $\underline{h}_j$ be the vector $\underline{g}_j+G_j(\underline{x}_k-\underline{x}_b)$ which is calculated, the analogue of expression (2.1) is that the subproblem can be written in the form

$$\text{maximize} \quad |\underline{h}_j^T\underline{d}+\tfrac{1}{2}\,\underline{d}^TG_j\,\underline{d}| \;=\; |\ell_j(\underline{x}_k+\underline{d})| \quad \text{subject to} \quad \|\underline{d}\|\le\rho. \tag{2.9}$$

Further, because the trust region constraint allows $\underline{d}$ to be replaced by $-\underline{d}$, it is equivalent to consider the calculation

$$\text{maximize} \quad |\underline{h}_j^T\underline{d}|+\tfrac{1}{2}\,|\underline{d}^TG_j\,\underline{d}| \quad \text{subject to} \quad \|\underline{d}\|\le\rho. \tag{2.10}$$

Now, if $\widehat{\underline{d}}$ and $\widetilde{\underline{d}}$ are values of $\underline{d}$ that maximize $|\underline{h}_j^T\underline{d}|$ and $|\underline{d}^TG_j\,\underline{d}|$, respectively, subject to $\|\underline{d}\|\le\rho$, then $\underline{d}$ may be an adequate estimate of the solution of the problem (2.9), if it is the choice between $\pm\widehat{\underline{d}}$ and $\pm\widetilde{\underline{d}}$ that gives the largest value of the objective function of the problem. Indeed, for every feasible $\underline{d}$, including the exact solution of the present calculation, we find the elementary bound

$$\begin{aligned} |\underline{h}_j^T\underline{d}|+\tfrac{1}{2}\,|\underline{d}^TG_j\,\underline{d}| \;&\le\; |\underline{h}_j^T\widehat{\underline{d}}|+\tfrac{1}{2}\,|\widetilde{\underline{d}}^TG_j\,\widetilde{\underline{d}}|+|\underline{h}_j^T\widetilde{\underline{d}}|+\tfrac{1}{2}\,|\widehat{\underline{d}}^TG_j\,\widehat{\underline{d}}| \\ &\le\; 2\,\max\Big[|\underline{h}_j^T\widehat{\underline{d}}|+\tfrac{1}{2}\,|\widehat{\underline{d}}^TG_j\,\widehat{\underline{d}}|,\; |\underline{h}_j^T\widetilde{\underline{d}}|+\tfrac{1}{2}\,|\widetilde{\underline{d}}^TG_j\,\widetilde{\underline{d}}|\Big]. \end{aligned} \tag{2.11}$$

It follows that the proposed choice of $\underline{d}$ gives a value of $|\ell_j(\underline{x}_k+\underline{d})|$ that is at least half of the optimal value. Now $\widehat{\underline{d}}$ is the vector $\pm\rho\,\underline{h}_j/\|\underline{h}_j\|$, while $\widetilde{\underline{d}}$ is an eigenvector of an eigenvalue of G_j of largest modulus, which would be too expensive to calculate. Therefore the UOBYQA software retains $\widehat{\underline{d}}$, but generates a different $\widetilde{\underline{d}}$ by the method of the next paragraph, which includes some features of the power method for obtaining large eigenvalues.

Because $|\widetilde{\underline{d}}^TG_j\,\widetilde{\underline{d}}|$ is large only if $\|G_j\,\widetilde{\underline{d}}\|$ is substantial, the technique begins by finding a column of G_j, $\underline{w}$ say, that has the greatest Euclidean norm. Hence, letting $\underline{v}_1,\underline{v}_2,\ldots,\underline{v}_n$ be the columns of the symmetric matrix G_j, we deduce the bound

$$\begin{aligned} \|G_j\,\underline{w}\| \;&\ge\; \|\underline{w}\|^2 \;=\; \max\{\|\underline{v}_k\|:k=1,2,\ldots,n\}\,\|\underline{w}\| \\ &\ge\; n^{-1/2}\left[\textstyle\sum_{k=1}^n\|\underline{v}_k\|^2\right]^{1/2}\,\|\underline{w}\| \;\ge\; n^{-1/2}\,\sigma(G_j)\,\|\underline{w}\|, \end{aligned} \tag{2.12}$$

where $\sigma(G_j)$ is the spectral radius of G_j. It may be disastrous, however, to set $\widetilde{\underline{d}}$ to a multiple of $\underline{w}$, because $\underline{w}^TG_j\,\underline{w}$ is zero in the case

$$G_j=\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -2/3 & -2/3 \\ 1 & -2/3 & -1 & -2/3 \\ 1 & -2/3 & -2/3 & -1 \end{pmatrix}, \tag{2.13}$$

for example. Therefore the algorithm picks $\underline{\widetilde{d}}$ from the two dimensional linear subspace of $\mathcal{R}^n$ that is spanned by $\underline{w}$ and $G_j\,\underline{w}$. Specifically, $\underline{\widetilde{d}}$ has the form $\alpha\,\underline{w}+\beta\,G_j\,\underline{w}$, where the ratio α/β is calculated to maximize the expression

$$|(\alpha\,\underline{w}+\beta\,G_j\,\underline{w})^T G_j\,(\alpha\,\underline{w}+\beta\,G_j\,\underline{w})|\,/\,\|\alpha\,\underline{w}+\beta\,G_j\,\underline{w}\|^2, \tag{2.14}$$

which determines the direction of $\underline{\widetilde{d}}$. Then the length of $\underline{\widetilde{d}}$ is defined by $\|\underline{\widetilde{d}}\|=\rho$, the sign of $\underline{\widetilde{d}}$ being unimportant. This construction gives the optimal $\underline{\widetilde{d}}$ in the case (2.13), because two eigenvectors of G_j are in the span of $\underline{w}$ and $G_j\,\underline{w}$, and they include the eigenvector of the eigenvalue that has modulus $\sigma(G_j)$.

This choice of $\underline{\widetilde{d}}$ is never very bad, because it achieves the property

$$|\underline{\widetilde{d}}^T G_j\,\underline{\widetilde{d}}| \;\geq\; \tfrac{1}{2}\,n^{-1/2}\,\sigma(G_j)\,\rho^2. \tag{2.15}$$

A proof of this assertion begins with the remark that, by construction, the maximum over α and β of expression (2.14) is just $|\underline{\widetilde{d}}^T G_j\,\underline{\widetilde{d}}|/\rho^2$. Therefore condition (2.15) holds if there exist values of α and β such that the ratio (2.14) is at least $\frac{1}{2}n^{-1/2}\sigma(G_j)$. We consider the case

$$\alpha = \|G_j\,\underline{w}\| \quad \text{and} \quad \beta = \pm\|\underline{w}\|, \tag{2.16}$$

where the $\pm$ sign is chosen so that the signs of $(\alpha^2\underline{w}^T G_j\,\underline{w}+\beta^2\underline{w}^T G_j^3\underline{w})$ and $\alpha\beta\,\underline{w}^T G_j^2\underline{w}$ are the same. Thus the numerator of expression (2.14) is bounded below by $|2\alpha\beta\,\underline{w}^T G_j^2\underline{w}|=|2\alpha^3\beta|$. Moreover, the values (2.16) and Cauchy–Schwarz imply the upper bound

$$\|\alpha\,\underline{w}+\beta\,G_j\,\underline{w}\|^2 \leq \alpha^2\,\|\underline{w}\|^2+|2\alpha\beta|\,\|\underline{w}\|\,\|G_j\,\underline{w}\|+\beta^2\,\|G_j\,\underline{w}\|^2 = 4\alpha^2\beta^2. \tag{2.17}$$

Hence expression (2.14) is at least $\frac{1}{2}|\alpha/\beta| = \frac{1}{2}\|G_j\,\underline{w}\|/\|\underline{w}\|$ in the case (2.16), which is bounded below by $\frac{1}{2}n^{-1/2}\sigma(G_j)$ because of condition (2.12). The proof of property (2.15) is complete.

Having generated $\underline{\widehat{d}}$ and $\underline{\widetilde{d}}$ in the ways that have been described, the algorithm sets $\underline{d}$ to a linear combination of these vectors, but the choice is not restricted to $\pm\underline{\widehat{d}}$ or $\pm\underline{\widetilde{d}}$ as suggested in the paragraph that includes inequality (2.11), unless $\underline{\widehat{d}}$ and $\underline{\widetilde{d}}$ are nearly or exactly parallel. Instead, vectors $\underline{\widehat{u}}$ and $\underline{\widetilde{u}}$ of length ρ are found in the span of $\underline{\widehat{d}}$ and $\underline{\widetilde{d}}$ that satisfy the conditions $\underline{\widehat{u}}^T\underline{\widetilde{u}}=0$ and $\underline{\widehat{u}}^T G_j\,\underline{\widetilde{u}}=0$, which is a 2×2 matrix eigenvalue problem. Then $\underline{d}$ has the form $\underline{d}=\cos\phi\,\underline{\widehat{u}}+\sin\phi\,\underline{\widetilde{u}}$. Further, ϕ is chosen to provide a relatively large value of the function

$$\begin{aligned}|\ell_j(\underline{x}_k+\underline{d})| \;=\; |\underline{h}_j^T\underline{d}+\tfrac{1}{2}\,\underline{d}^T G_j\,\underline{d}| \;=\; &|\underline{h}_j^T\underline{\widehat{u}}\,\cos\phi+\underline{h}_j^T\underline{\widetilde{u}}\,\sin\phi \\ &+\tfrac{1}{2}\,\underline{\widehat{u}}^T G_j\,\underline{\widehat{u}}\,\cos^2\phi+\tfrac{1}{2}\,\underline{\widetilde{u}}^T G_j\,\underline{\widetilde{u}}\,\sin^2\phi|, \quad 0\leq\phi\leq 2\pi.\end{aligned} \tag{2.18}$$

The ϕ that maximizes this expression can be derived from a quartic polynomial equation, but for convenience the choice of ϕ by the algorithm is restricted to integer multiples of $\pi/4$. Thus the final value of $|\ell_j(\underline{x}_k+\underline{d})|$ is the quantity

$$\max\Big[\, |\underline{h}_j^T\,\widehat{\underline{u}}| + \tfrac{1}{2}\,|\widehat{\underline{u}}^T G_j\,\widehat{\underline{u}}|,\ |\underline{h}_j^T\,\widetilde{\underline{u}}| + \tfrac{1}{2}\,|\widetilde{\underline{u}}^T G_j\,\widetilde{\underline{u}}|,\\ 2^{-1/2}\,(|\underline{h}_j^T\,\widehat{\underline{u}}| + |\underline{h}_j^T\,\widetilde{\underline{u}}|) + \tfrac{1}{4}\,|\widehat{\underline{u}}^T G_j\,\widehat{\underline{u}} + \widetilde{\underline{u}}^T G_j\,\widetilde{\underline{u}}|\,\Big]. \qquad (2.19)$$

This quantity is never much less than the maximum value of the function (2.18). Specifically, the author has found analytically that the ratio of expression (2.19) to the maximum value is bounded below by $(12+2\sqrt{2})/17 \approx 0.872$.

During the numerical testing in the development of the UOBYQA software, comparisons were made between the true solution of the problem (1.8) and the given approximate solution that requires only $\mathcal{O}(n^2)$ computer operations. In these tests the ratio of expression (2.19) to the optimal value $\max\{|\ell_j(\underline{x}_k+\underline{d})| : \|\underline{d}\| \le \rho\}$ was calculated whenever the problem (1.8) was solved. For example, in five runs of the experiment of Section 6 that minimized the function (6.1) with $n=20$, the problem (1.8) occurred 8466 times altogether. The ratio was never less than 0.5, and the numbers of times it was less than 0.6, 0.7, 0.8 and 0.9 were 3, 65, 544 and 2519, respectively. The figures are better for smaller values of n. For example, in the same experiment with $n=5$, the ratio was never less than 0.8, and it was less than 0.9 on only 6 out of 194 occasions. Therefore our procedure for the subproblem (1.8) seems to be suitable, there being no need for high precision in the new position of an interpolation point that is moved by a model step.

3. The changes to the variables

Our algorithm is iterative, and we begin this description of the changes that are made to the variables by defining an iteration in a convenient way. The definition is based on the information that is required at the start of an iteration. That information includes the current interpolation points $\underline{x}_i$, $i=1,2,\ldots,m$, the vector $\underline{x}_b$ of expressions (1.1) and (1.4), the gradient $\underline{g}_j$ and the second derivative matrix G_j of each of the current Lagrange functions ℓ_j, $j=1,2,\ldots,m$, and the gradient g_Q and the second derivative matrix G_Q of the current quadratic model, which satisfies the interpolation equations (1.2). It includes also the current values of the trust region radii Δ and ρ where $\Delta \ge \rho$, and the integer k that is introduced in expression (1.7), so $F(\underline{x}_k)$ is the least of the function values $F(\underline{x}_i)$, $i=1,2,\ldots,m$. Furthermore, a key ingredient of the information is a decision that has been taken already between two alternatives, namely whether the iteration will calculate $\underline{d} \in \mathcal{R}^n$ by solving the problem (1.7), or whether an interpolation point will be moved by a model step. In the latter case, the index j of the point that will be moved and the new position of the point, namely $\underline{x}_k+\underline{d}$, are available. A few other numbers are required too, which will receive attention later.

We define an iteration to be the work of carrying out the given decision between the alternatives, followed by all the operations that provide the information that has been mentioned for the next iteration, except that the final iteration ends the computation because a termination condition is satisfied. Therefore one new value of the objective function is calculated on most iterations, and it is included in the quadratic model by the updating methods that are given in Section 4. In this section we address not only the changes that are made to the variables, but also the revision of $\underline{x}_b$, Δ and ρ, the termination condition, the way of deciding between the alternatives for the next iteration, and the selection of the index j of each interpolation point that will be moved by a model step.

There is little to say about iterations that apply a model step, because they do not alter $\underline{x}_b$, Δ and ρ, and always their decision between the alternatives is that the next iteration will solve the problem (1.7). Of course the function value $F(\underline{x}_k+\underline{d})$ is calculated for the given vector $\underline{d}$, and then the interpolation point $\underline{x}_j$ is replaced by $\underline{x}_k+\underline{d}$, which requires the first and second derivative coefficients of the Lagrange functions and the quadratic model to be updated. Further, k is not altered in the case $F(\underline{x}_k+\underline{d}) \geq F(\underline{x}_k)$, but otherwise it is changed to j, because the new $F(\underline{x}_j)$ is the least calculated value of the objective function so far. Another task of a model step iteration is explained in the next paragraph. It occurs just before the updating, and is important to the fast rate of convergence that is mentioned at the end of Section 1.

The task is concerned with the following theorem that is given in Powell (2000). If the objective function has third derivatives that are bounded by a constant, M say, then the difference between the quadratic model and the objective function satisfies the condition

$$|Q(\underline{x}) - F(\underline{x})| \leq \tfrac{1}{6} M \sum_{i=1}^{m} |\ell_i(\underline{x})| \, \|\underline{x}-\underline{x}_i\|^3, \qquad \underline{x} \in \mathcal{R}^n. \tag{3.1}$$

One can take the view that this property of quadratic interpolation is of academic interest only, because the third derivative assumption is too restrictive, because functions are not differentiable in the presence of computer rounding errors, and because M is not available. In practice, however, the algorithm employs inequality (3.1) in a way that can be implemented for general objective functions, and that is highly successful at reducing the number of iterations in comparison with earlier versions of UOBYQA. Therefore we let $\frac{1}{6}M$ be a nonnegative parameter of the software, that corresponds to $\frac{1}{6}M$ in condition (3.1), and that is included in the information at the beginning of each iteration. Then, when $F(\underline{x}_k+\underline{d})$ is obtained by a model step iteration, it is convenient to put $\underline{x} = \underline{x}_k+\underline{d}$ into the bound (3.1), in order to test whether $\frac{1}{6}M$ seems to be large enough. Specifically, the value of this parameter is overwritten by the number

$$\max \Big[\tfrac{1}{6} M, \; |Q(\underline{x}_k+\underline{d}) - F(\underline{x}_k+\underline{d})| \Big/ \sum_{i=1}^{m} |\ell_i(\underline{x}_k+\underline{d})| \, \|\underline{x}_k+\underline{d}-\underline{x}_i\|^3 \Big], \tag{3.2}$$

where the functions and vectors in this expression are the ones that are given at the beginning of the iteration. The description of the work that is done by a model step iteration is complete.

Alternatively, an iteration of trust region type solves the problem (1.7) by the method that is described in the first half of Section 2, which gives the trial step $\underline{d}$. Further, that method also provides an estimate of the least eigenvalue of the second derivative matrix G_Q if $\underline{d}$ is a Newton–Raphson step. We let $\overline{\lambda}_Q$ be this estimate in the case $\|\underline{d}\| < \frac{1}{2}\rho$, but otherwise $\overline{\lambda}_Q$ is set to zero. The number

$$\varepsilon = \tfrac{1}{2}\,\rho^2\,\overline{\lambda}_Q \tag{3.3}$$

is stored, because, if the next iteration is going to move $\underline{x}_j$ by a model step, then ε is required in the selection of j.

We recall from the opening paragraph of Section 1 that typical distances between successive points at which F is calculated are of magnitude ρ. Therefore the current iteration computes $F(\underline{x}_k+\underline{d})$ if and only if $\|\underline{d}\| \geq \frac{1}{2}\rho$ holds. Attention will be given later to the case $\|\underline{d}\| < \frac{1}{2}\rho$. When $F(\underline{x}_k+\underline{d})$ is obtained, we take the opportunity of revising the parameter $\frac{1}{6}M$ again to the value (3.2). Then the value of Δ is updated in a way that depends on the ratio

$$r = [\, F(\underline{x}_k) - F(\underline{x}_k+\underline{d})\,] \,/\, [\, Q(\underline{x}_k) - Q(\underline{x}_k+\underline{d})\,]\,. \tag{3.4}$$

Specifically, the details of the updating are typical of trust region methods, because we assume that the trust region is too conservative, adequate or overambitious in the cases $r \geq 0.7$, $0.1 < r < 0.7$ or $r \leq 0.1$, respectively. Therefore Δ is overwritten by the new trust region radius

$$\begin{cases} \max\,[\,\Delta,\, \frac{5}{4}\,\|\underline{d}\|,\, \rho + \|\underline{d}\|\,], & r \geq 0.7, \\ \max\,[\,\frac{1}{2}\,\Delta,\, \|\underline{d}\|\,], & 0.1 < r < 0.7, \\ \frac{1}{2}\,\|\underline{d}\|, & r \leq 0.1, \end{cases} \tag{3.5}$$

except that the new value is set to ρ if it would satisfy $\Delta \leq \frac{3}{2}\rho$ otherwise, because $\Delta \geq \rho$ is mandatory, and we will find that not allowing Δ to be slightly larger than ρ is helpful occasionally.

Next, because $F(\underline{x}_k+\underline{d})$ is available, it is usual to replace one of the points $\underline{x}_i$, $i = 1, 2, \ldots, m$, by $\underline{x}_k+\underline{d}$, a replacement being obligatory in the case

$$F(\underline{x}_k+\underline{d}) < F(\underline{x}_k), \tag{3.6}$$

in order to retain the least calculated value of F. The details of this part of the algorithm are given in the next section. Let t be the index of the new interpolation point if a replacement is made and let t be zero otherwise. Of course Q and ℓ_i, $i = 1, 2, \ldots, m$, are updated if $t \neq 0$. Further, the value of the index k of the best interpolation point is altered to t when the reduction (3.6) is achieved.

When inequality (3.6) is satisfied for the old value of k, it seems that trust region steps are helpful to the main calculation. In this case, therefore, the values of $\underline{x}_b$ and ρ are not changed, and the decision is taken that the next iteration will also solve the problem (1.7), which completes the work of the current iteration. We expect this strategy to be advantageous on average, but, if a long sequence of trust region steps lies in a linear subspace of $\mathcal{R}^n$ that has dimension less than n, then some important features of the objective function may be ignored for many iterations. The algorithm also perseveres with trust region steps whenever $t \neq 0$ and $\|\underline{d}\| > 2\rho$ occur, because the model has been revised, and, if inequality (3.6) failed, then the new value of Δ is at most $\frac{1}{2}\|\underline{d}\|$.

Now the quadratic model is assumed to be adequate for the current ρ if the interpolation points satisfy the conditions

$$\|\underline{x}_i - \underline{x}_k\| \leq 2\rho, \qquad i = 1, 2, \ldots, m. \tag{3.7}$$

This property is enjoyed by the new position of $\underline{x}_t$ when $t \neq 0$ and $\|\underline{d}\| \leq 2\rho$, so the iteration has made some useful progress if the distance $\|\underline{x}_t - \underline{x}_k\|$ was greater than 2ρ before the updating altered $\underline{x}_t$. Therefore in this case too the current iteration is complete, and the next iteration will try another trust region step without changing $\underline{x}_b$ and ρ.

The other possibilities are considered in the remainder of this section, including the situation $\|\underline{d}\| < \frac{1}{2}\rho$ when $F(\underline{x}_k + \underline{d})$ is not calculated. The notation will refer to current functions and points, which differ from those at the beginning of the iteration if some updating has been done. We will find that the iteration prepares for a model step if the quadratic model seems to be inadequate. Otherwise, if $\|\underline{d}\| > \rho$ holds, then again the next iteration takes a trust region step with the new Δ for the current ρ. In the remaining case, when the quadratic model seems to be suitable, and either condition (3.6) fails or $F(\underline{x}_k + \underline{d})$ is not calculated, then no further progress may be possible for the current ρ. Therefore ρ is reduced, except that termination occurs if ρ has already reached the prescribed lower bound $\rho_{\rm end}$. The details of all these operations are as follows.

When the parameter (3.3) has the value $\varepsilon = 0$, and also in the earlier versions of UOBYQA that did not take advantage of the bound (3.1), the quadratic model is unacceptable if one or more of the conditions (3.7) fail. If unacceptability of the model occurs, then the next iteration will move the point $\underline{x}_j$ by a model step, where j is chosen from the set

$$\mathcal{J} = \{i : \|\underline{x}_i - \underline{x}_k\| > 2\rho\}. \tag{3.8}$$

Usually the algorithm picks the least integer $j \in \mathcal{J}$ that has the property

$$\|\underline{x}_j - \underline{x}_k\| = \max\{\|\underline{x}_i - \underline{x}_k\| : i \in \mathcal{J}\}. \tag{3.9}$$

Further, the move $\underline{d}$ for the model step is calculated by applying the method in the second half of Section 2 to the problem (1.8).

Many calculations of the objective function can be saved, however, by using the bound (3.1) to relax the tests for the acceptability of the quadratic model, in a way that preserves the procedure of the previous paragraph in the case $\varepsilon = 0$. In order to be specific, we construe the decision not to calculate $F(\underline{x}_k + \underline{d})$ in the case $\|\underline{d}\| < \frac{1}{2}\rho$ as giving up an opportunity to reduce the objective function by an amount that is predicted to be of magnitude ε. Further, if such possible changes to the objective function are being neglected, then errors of ε in the quadratic model should be tolerable. These errors are indicated in expression (3.1), which suggests that, if $\underline{x}$ is constrained by the trust region bound $\|\underline{x} - \underline{x}_k\| \leq \rho$, then the contribution to the error of the model from the position of $\underline{x}_j$ is approximately the quantity

$$\begin{aligned} &\tfrac{1}{6} M \max\{|\ell_j(\underline{x})|\, \|\underline{x} - \underline{x}_j\|^3 : \|\underline{x} - \underline{x}_k\| \leq \rho\} \\ &\qquad\qquad \approx \tfrac{1}{6} M\, \|\underline{x}_j - \underline{x}_k\|^3 \max\{|\ell_j(\underline{x}_k + \underline{d})| : \|\underline{d}\| \leq \rho\}. \end{aligned} \tag{3.10}$$

Therefore we prefer not to move $\underline{x}_j$ if it satisfies the condition

$$\tfrac{1}{6} M\, \|\underline{x}_j - \underline{x}_k\|^3 \max\{|\ell_j(\underline{x}_k + \underline{d})| : \|\underline{d}\| \leq \rho\} \;\leq\; \varepsilon. \tag{3.11}$$

We are ignoring the dependence of the other Lagrange functions on $\underline{x}_j$, however, in the hope of finding a useful technique that can be implemented cheaply.

We are going to combine this idea with the operations of the paragraph that includes equations (3.8) and (3.9). Moreover, we are concerned by the possibility that the parameter $\frac{1}{6}M$ may be too small, because initially it is set to zero, so the value (3.3) is replaced by $\varepsilon = 0$ if $\frac{1}{6}M$ has been overwritten by the number (3.2) fewer than ten times. Then the selection of an integer j for a model step iteration begins as before by forming the set (3.8), and, if $\mathcal{J}$ is nonempty, by seeking a j that has the property (3.9). Further, we retain the calculation of $\underline{d}$ from the problem (1.8), and now we also require the number $\ell_j(\underline{x}_k + \underline{d})$, because the choice of $\underline{d}$ makes $|\ell_j(\underline{x}_k + \underline{d})|$ close to the maximum value in expression (3.11). It follows that, if the inequality

$$\tfrac{1}{6} M\, \|\underline{x}_j - \underline{x}_k\|^3\, |\ell_j(\underline{x}_k + \underline{d})| \;\leq\; \varepsilon \tag{3.12}$$

holds, then the position of $\underline{x}_j$ seems to be adequate, so j is deleted from $\mathcal{J}$, and another j is found that satisfies equation (3.9) if the diminished set $\mathcal{J}$ is nonempty. This procedure continues recursively until $j \in \mathcal{J}$ fails the test (3.12), or until $\mathcal{J}$ is exhausted. In the former case, the work of the present iteration is complete, and the next iteration has to move $\underline{x}_j$ to $\underline{x}_k + \underline{d}$ by a model step.

Otherwise, the tests for the acceptability of the quadratic model are achieved, so we ask whether further calculations are required for the current value of ρ. The answer is affirmative if and only if $\|\underline{d}\| > \rho$ holds, where $\underline{d}$ is now the solution of the problem (1.7) at the beginning of the present iteration. Then the decision is taken to solve the trust region problem (1.7) on the next iteration. The resetting of Δ to ρ immediately after expression (3.5) helps to provide $\|\underline{d}\| \leq \rho$.

In the remaining situation, no more iterations are required for the present value of ρ, because we have a good quadratic model, but it seems that steps of length ρ fail to decrease the objective function. Therefore, if $\rho > \rho_{\text{end}}$, the algorithm reduces ρ from ρ_{old} to ρ_{new}, say, by applying the formula

$$\rho_{\text{new}} = \begin{cases} \rho_{\text{end}}, & \rho_{\text{end}} < \rho_{\text{old}} \leq 16\,\rho_{\text{end}}, \\ \sqrt{\rho_{\text{old}}\,\rho_{\text{end}}}, & 16\,\rho_{\text{end}} < \rho_{\text{old}} \leq 250\,\rho_{\text{end}}, \\ 0.1\,\rho_{\text{old}}, & \rho_{\text{old}} > 250\,\rho_{\text{end}}, \end{cases} \tag{3.13}$$

which is designed to provide reductions by about a factor of ten that achieve $\rho = \rho_{\text{end}}$ eventually. Further, $\underline{x}_b$ is overwritten by $\underline{x}_b + \underline{x}_k$, which makes no difference to the second derivative matrices G_Q and G_j, but the gradient vectors of the quadratic model and Lagrange functions become $\underline{g}_Q + G_Q\,\underline{x}_k$ and $\underline{g}_j + G_j\,\underline{x}_k$, $j = 1, 2, \ldots, m$, respectively. Then Δ is set to $\max[\frac{1}{2}\rho_{\text{old}}, \rho_{\text{new}}]$, which has the advantage of allowing a trust region step that satisfied $\|\underline{d}\| < \frac{1}{2}\rho$ for $\rho = \rho_{\text{old}}$. We are now ready to begin the iterations with $\rho = \rho_{\text{new}}$, and the decision between the alternatives for the first of them is that the problem (1.7) will be solved.

Of course termination occurs when $\rho = \rho_{\text{end}}$ and no more iterations are required, but another value of the objective function may be calculated. Specifically, if the solution of the problem (1.7) satisfied $\|\underline{d}\| < \frac{1}{2}\rho$, then the value of $F(\underline{x}_k + \underline{d})$ has not been computed, so the algorithm does that computation now. Further, $\underline{x}_k$ is overwritten by $\underline{x}_k + \underline{d}$ if inequality (3.6) is achieved, in order that $\underline{x}_k$ can be returned as the optimal vector of variables. This device often provides a substantial improvement to the vector of variables, because, when it is applied, the quadratic model is good and $\underline{d}$ is a Newton–Raphson step.

4. Initialization and updating

The initialization procedure depends on the data $\underline{x}_b \in \mathcal{R}^n$ and $\rho_{\text{beg}} > 0$ that have to be provided by the user of UOBYQA. It is usually helpful if the given $\underline{x}_b$ is close to the required vector of variables, while numbers of magnitude ρ_{beg} are assumed to be suitable as distances between any two of the interpolation points $\underline{x}_i$, $i = 1, 2, \ldots, m$, of the initial quadratic model. The positions of these interpolation points, which are taken from Powell (2000), are as follows.

They include $\underline{x}_1 = \underline{x}_b$ and $\underline{x}_{2j} = \underline{x}_b + \rho_{\text{beg}}\,\underline{e}_j$, $j = 1, 2, \ldots, n$, where $\underline{e}_j$ is the j-th coordinate vector in $\mathcal{R}^n$. The choice of $\underline{x}_{2j+1}$ depends on $F(\underline{x}_{2j})$, however, in order to provide a bias towards low function values. Specifically, defining σ_j to be -1 or $+1$ in the cases $F(\underline{x}_{2j}) \geq F(\underline{x}_b)$ or $F(\underline{x}_{2j}) < F(\underline{x}_b)$, respectively, UOBYQA applies the formula

$$\underline{x}_{2j+1} = \begin{cases} \underline{x}_b - \rho_{\rm beg}\, \underline{e}_j & \text{if} \quad \sigma_j = -1, \\ \underline{x}_b + 2\rho_{\rm beg}\, \underline{e}_j & \text{if} \quad \sigma_j = +1, \end{cases} \qquad j=1,2,\ldots,n. \tag{4.1}$$

Thus $\underline{x}_{2j+1}$ is on the positive side of $\underline{x}_b$ along the j-th coordinate direction if and only if σ_j is positive. Further, letting $i(p,q)$ have the value

$$i(p,q) = 2n+1+p+\tfrac{1}{2}\,(q-1)\,(q-2), \qquad 1 \leq p < q \leq n, \tag{4.2}$$

the remaining initial interpolation points are assigned the positions

$$\underline{x}_{i(p,q)} = \underline{x}_b + \rho_{\rm beg}\,(\sigma_p\, \underline{e}_p + \sigma_q\, \underline{e}_q), \qquad 1 \leq p < q \leq n. \tag{4.3}$$

The notation (4.2) provides the property that the subscripts of the vectors (4.3) run through the integers in the interval $[2n+2, m]$.

These choices make it easy to derive the parameters of the first quadratic model (1.1) from the conditions (1.2). Firstly, the coincidence $\underline{x}_1 = \underline{x}_b$ gives $c_Q = F(\underline{x}_1)$, and then, for $j=1,2,\ldots,n$, the fact that $\underline{x}_{2j} - \underline{x}_b$ and $\underline{x}_{2j+1} - \underline{x}_b$ are different nonzero multiples of $\underline{e}_j$ allows $(\underline{g}_Q)_j$ and $(G_Q)_{jj}$ to be deduced from $F(\underline{x}_{2j}) - F(\underline{x}_b)$ and $F(\underline{x}_{2j+1}) - F(\underline{x}_b)$, where $(\underline{g}_Q)_j$ and $(G_Q)_{pq}$ denote the j-th component of $\underline{g}_Q$ and the (p,q)-th element of G_Q, respectively. Finally, the off-diagonal elements of the symmetric matrix G_Q are obtained from the remark that equations (1.1), (1.2) and (4.3) imply the identity

$$\begin{aligned} c_Q + \rho_{\rm beg}\,[\sigma_p\,(\underline{g}_Q)_p + \sigma_q\,(\underline{g}_Q)_q] + \tfrac{1}{2}\rho_{\rm beg}^2\,[(G_Q)_{pp} + 2\sigma_p\sigma_q\,(G_Q)_{pq} + (G_Q)_{qq}] \\ = F(\underline{x}_{i(p,q)}), \qquad 1 \leq p < q \leq n. \end{aligned} \tag{4.4}$$

Indeed, the required matrix element $(G_Q)_{pq}$ is the only unknown quantity in this relation for each p and q. Thus the positions of the interpolation points reduce the work of solving the initial $m \times m$ system (1.2) to only $\mathcal{O}(n^2)$ operations.

All the nonzero coefficients of all the initial Lagrange functions can also be computed in $\mathcal{O}(n^2)$ operations, because, for $1 \leq p < q \leq n$, the Lagrange function $\ell_{i(p,q)}$ has a very simple form. It is derived from the remark that, if $p \neq q$, then, because of the positions of the interpolation points, only the $i(p,q)$-th of the products $(\underline{x}_i - \underline{x}_b)_p\,(\underline{x}_i - \underline{x}_b)_q$, $i=1,2,\ldots,m$, is nonzero, where the notation $(\underline{x} - \underline{x}_b)_j$ denotes the j-th component of $\underline{x} - \underline{x}_b$. Hence we find the formula

$$\ell_{i(p,q)}(\underline{x}) = (\sigma_p\sigma_q/\rho_{\rm beg}^2)\,(\underline{x} - \underline{x}_b)_p\,(\underline{x} - \underline{x}_b)_q, \qquad \underline{x} \in \mathcal{R}^n, \quad 1 \leq p < q \leq n, \tag{4.5}$$

which shows that $(G_{i(p,q)})_{pq} = \sigma_p\sigma_q/\rho_{\rm beg}^2$ is the only nonzero coefficient of $\ell_{i(p,q)}$.

Next we identify the nonzero coefficients of ℓ_k, $k=2,3,\ldots,2n+1$, by making use of the quadratic polynomials

$$\left.\begin{aligned} \widehat{\ell}_{2j}(\underline{x}) &= \frac{(\underline{x} - \underline{x}_b)_j\,(\underline{x} - \underline{x}_{2j+1})_j}{(\underline{x}_{2j} - \underline{x}_b)_j\,(\underline{x}_{2j} - \underline{x}_{2j+1})_j} \\ \widehat{\ell}_{2j+1}(\underline{x}) &= \frac{(\underline{x} - \underline{x}_b)_j\,(\underline{x} - \underline{x}_{2j})_j}{(\underline{x}_{2j+1} - \underline{x}_b)_j\,(\underline{x}_{2j+1} - \underline{x}_{2j})_j} \end{aligned}\right\}, \quad \underline{x} \in \mathcal{R}^n, \quad j=1,2,\ldots,n. \tag{4.6}$$

They satisfy the Lagrange conditions $\widehat{\ell}_k(\underline{x}_i) = \delta_{ik}$, $k = 2, 3, \ldots, 2n+1$, for most integers i in $[1, m]$, the exceptions being $i = i(p, j)$, $1 \leq p < j$, and $i = i(j, q)$, $j < q \leq n$, where j is the integer in $[1, n]$ such that k is equal to $2j$ or $2j+1$. It follows from the Lagrange properties of formula (4.5) that the function

$$\ell_k(\underline{x}) = \widehat{\ell}_k(\underline{x}) - \sum_{p=1}^{j-1} \widehat{\ell}_k(\underline{x}_{i(p,j)})\, \ell_{i(p,j)}(\underline{x}) - \sum_{q=j+1}^{n} \widehat{\ell}_k(\underline{x}_{i(j,q)})\, \ell_{i(j,q)}(\underline{x}), \quad \underline{x} \in \mathcal{R}^n, \quad (4.7)$$

does achieve $\ell_k(\underline{x}_i) = \delta_{ik}$, $i = 1, 2, \ldots, m$, as required, where the first or second sum is suppressed in the case $j = 1$ or $j = n$, respectively. Expressions (4.5), (4.6) and (4.7) show that the nonzero off-diagonal elements of $G_k = \nabla^2 \ell_k$ are confined to the j-th row and column of G_k. They also imply that $(G_k)_{jj}$ is the only nonzero diagonal element of G_k. Similarly, only the j-th component of $\underline{g}_k$ can be nonzero. Therefore it is easy to calculate the parameters of the initial Lagrange functions ℓ_k, $k = 2, 3, \ldots, 2n+1$. The parameters of ℓ_1, however, can all be nonzero. The algorithm extracts their values from the elementary identity

$$\ell_1(\underline{x}) = 1 - \sum_{i=2}^{m} \ell_i(\underline{x}), \qquad \underline{x} \in \mathcal{R}^n. \quad (4.8)$$

Furthermore, the initialization procedure sets k for the first iteration to the least integer in $[1, m]$ such that $F(\underline{x}_k)$ is the least of the function values $F(\underline{x}_i)$, $i = 1, 2, \ldots, m$, and it picks the initial values $\frac{1}{6}M = 0$, $\rho = \rho_{\text{beg}}$ and $\Delta = \rho_{\text{beg}}$. The decision between the alternatives for the first iteration of UOBYQA is that the problem (1.7) will be solved.

We now turn to the updating of the coefficients of the Lagrange functions when the interpolation point $\underline{x}_t$ is moved to the new position $\widetilde{\underline{x}}_t$, say, but the positions of the other interpolation points are preserved. We let the old and new Lagrange functions be ℓ_i, $i = 1, 2, \ldots, m$, and $\widetilde{\ell}_i$, $i = 1, 2, \ldots, m$, respectively. The point $\widetilde{\underline{x}}_t$ has to satisfy the condition

$$\ell_t(\widetilde{\underline{x}}_t) \neq 0, \quad (4.9)$$

because otherwise the nonzero quadratic polynomial ℓ_t would vanish on the new set of interpolation points, so the new system of equations (1.2) would be singular. Moreover, for every integer i in $[1, m]$, the difference $\widetilde{\ell}_i - \ell_i$ has to be a multiple of $\widetilde{\ell}_t$, in order that $\widetilde{\ell}_i$ agrees with ℓ_i at all the old interpolation points that are retained. Further, for each i, the multiplying factor is defined by the equation $\widetilde{\ell}_i(\widetilde{\underline{x}}_t) = \delta_{it}$. Thus we deduce the formulae

$$\widetilde{\ell}_t(\underline{x}) \;=\; \ell_t(\underline{x}) \,/\, \ell_t(\widetilde{\underline{x}}_t), \qquad \underline{x} \in \mathcal{R}^n, \quad (4.10)$$

and

$$\widetilde{\ell}_i(\underline{x}) = \ell_i(\underline{x}) - \ell_i(\widetilde{\underline{x}}_t)\, \widetilde{\ell}_t(\underline{x}), \qquad \underline{x} \in \mathcal{R}^n, \quad i \neq t. \quad (4.11)$$

Therefore the algorithm updates the coefficients of the Lagrange functions in the following way. The coefficients of $\widetilde{\ell}_t$ are set to the corresponding coefficients of ℓ_t divided by $\ell_t(\underline{\widetilde{x}}_t)$. Then, for every integer i in $[1,m]$ that is different from t, the coefficients of $\widetilde{\ell}_i$ are set to the coefficients of ℓ_i minus the corresponding coefficients of $\widetilde{\ell}_t$ multiplied by $\ell_i(\underline{\widetilde{x}}_t)$. The numbers $\ell_i(\underline{\widetilde{x}}_t)$, $i=1,2,\ldots,m$, are available, because $\underline{\widetilde{x}}_t$ is always the vector $\underline{x}_k+\underline{d}$ that occurred in the most recent use of expression (3.2). These remarks specify the updating method in a very convenient form.

Fortunately, as explained in Powell (2000), the formulae (4.10) and (4.11) have some excellent stability properties. In particular, if ℓ_t is any quadratic polynomial, and if $\underline{\widetilde{x}}_t$ is any point of $\mathcal{R}^n$ that obeys the constraint (4.9), then equation (4.10) gives the identity $\widetilde{\ell}_t(\underline{\widetilde{x}}_t)=1$. Thus expression (4.11) provides $\widetilde{\ell}_i(\underline{\widetilde{x}}_t)=0$, $i\neq t$, even if the old Lagrange functions ℓ_i, $i=1,2,\ldots,m$, fail to satisfy any Lagrange conditions. Usually, however, the old Lagrange functions were generated by the updating method on the previous iteration. It follows from the present argument that, if $\underline{x}_j$ is the interpolation point that was moved, then the values

$$\ell_i(\underline{x}_j)=\delta_{ij}, \qquad i=1,2,\ldots,m, \tag{4.12}$$

have been achieved already. Therefore we assume that the conditions (4.12) hold for an integer j that is different from t. In this case, formula (4.10) shows that $\widetilde{\ell}_t(\underline{x}_j)=0$ is inherited from $\ell_t(\underline{x}_j)=0$, so the function (4.11) satisfies $\widetilde{\ell}_i(\underline{x}_j)=\ell_i(\underline{x}_j)=\delta_{ij}$, $i\neq t$. We draw the following conclusions by applying these remarks recursively. Any failure in the Lagrange conditions at an interpolation point is corrected by the updating method when the interpolation point is moved to a new position. Then any further failures at that point are caused only by computer rounding errors, even if there are large discrepancies in the Lagrange conditions at the other interpolation points. We do not expect any large discrepancies to occur, however, and this view is corroborated very well by numerical experiments, as shown in Section 6.

The quadratic model has to be revised too, when $\underline{x}_t$ is moved to the new position $\underline{\widetilde{x}}_t$ and the other interpolation points are not disturbed. Then, letting Q and $\widetilde{Q}$ be the old and new models, the equations $\widetilde{Q}(\underline{x}_i)=Q(\underline{x}_i)$, $i\neq t$, should hold. It follows that the difference $\widetilde{Q}-Q$ is the multiple of the Lagrange function (4.10) that provides the value $\widetilde{Q}(\underline{\widetilde{x}}_t)=F(\underline{\widetilde{x}}_t)$. Thus $\widetilde{Q}$ is the quadratic polynomial

$$\widetilde{Q}(\underline{x})=Q(\underline{x})+[F(\underline{\widetilde{x}}_t)-Q(\underline{\widetilde{x}}_t)]\,\widetilde{\ell}_t(\underline{x}), \qquad \underline{x}\in\mathcal{R}^n. \tag{4.13}$$

Therefore its coefficients are generated by adding to the coefficients of Q the corresponding coefficients of $\widetilde{\ell}_t$ multiplied by the factor

$$F(\underline{\widetilde{x}}_t)-Q(\underline{\widetilde{x}}_t)=F(\underline{x}_k+\underline{d})-Q(\underline{x}_k+\underline{d}), \tag{4.14}$$

which is also available from the most recent use of expression (3.2). Equation (4.13) implies $\widetilde{Q}(\underline{\widetilde{x}}_t)=F(\underline{\widetilde{x}}_t)$ for any function $Q(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, and it also implies

$\widetilde{Q}(\underline{x}_i) = Q(\underline{x}_i)$, $i \neq t$. Therefore any failure in the condition $Q(\underline{x}_t) = F(\underline{x}_t)$ is corrected when $\underline{x}_t$ is moved, and any further failures are due to computer rounding errors. Thus the stability properties of the updating of Q are similar to those of the previous paragraph.

Finally, we address the selection of t, when $F(\underline{x}_k+\underline{d})$ has been calculated for a trial step $\underline{d}$ obtained from the problem (1.7). We recall from Section 3 that t may be positive or zero, $t=0$ being reserved for the case when no updating is done by the current iteration, but otherwise the interpolation point $\underline{x}_t$ is moved to the position $\widetilde{\underline{x}}_t = \underline{x}_k+\underline{d}$. The algorithm picks a value of t from the set $\{1, 2, \ldots, m\}$ in the following way, and then decides later whether to overwrite the choice by $t=0$.

Our remarks on condition (4.9) show that it is important for t to have the property

$$\ell_t(\underline{x}_k + \underline{d}) \neq 0. \tag{4.15}$$

Further, we wish to move an interpolation point that seems to be making a relatively large contribution to the bound (3.1) on the error of the quadratic model. Both of these objectives are observed by developing the idea of letting t be a value of i that maximizes the product $|\ell_i(\underline{x}_k+\underline{d})| \, \|\underline{x}_k+\underline{d}-\underline{x}_i\|^3$, $i=1, 2, \ldots, m$. The first term of the product is welcome in view of condition (4.15), and the second one is useful in the case (3.6), because it promotes the replacement of an interpolation point that is far from the current best vector of variables. We strengthen this aim by changing the product to $|\ell_i(\underline{x}_k+\underline{d})| \, \|\underline{x}_i-\widehat{\underline{x}}_k\|^3$, where $\widehat{\underline{x}}_k$ is the value of $\underline{x}_k$ for the next iteration, which is $\underline{x}_k$ instead of $\underline{x}_k+\underline{d}$ if $F(\underline{x}_k+\underline{d}) \geq F(\underline{x}_k)$ occurs. Moreover, the position of $\underline{x}_i$ is close enough to $\widehat{\underline{x}}_k$ if it satisfies $\|\underline{x}_i-\widehat{\underline{x}}_k\| \leq \rho$, so then we give priority to the term $|\ell_i(\underline{x}_k+\underline{d})|$, except that $\underline{x}_k$ must not be moved if it is the best vector of variables so far. Specifically, the algorithm combines these ingredients by setting t to a value of i that maximizes the expression

$$|\ell_i(\underline{x}_k+\underline{d})| \, \max[\,1, \, \|\underline{x}_i-\widehat{\underline{x}}_k\|^3 / \rho^3\,], \qquad i \in \{1, 2, \ldots, m\} \backslash \mathcal{K}, \tag{4.16}$$

where $\mathcal{K}$ is empty or $\{k\}$ if $\widehat{\underline{x}}_k = \underline{x}_k+\underline{d}$ or $\widehat{\underline{x}}_k = \underline{x}_k$, respectively. This value of t is retained whenever condition (3.6) is achieved, and whenever the greatest of the products (4.16) exceeds one, but otherwise $t=0$ is preferred. Therefore a positive integer t that satisfies $|\ell_t(\underline{x}_k+\underline{d})| > 1$ is never rejected. Thus the updating tends to be beneficial to the interpolation equations (1.2), because, if the equations are written in matrix form by introducing any basis of the space of quadratic polynomials, then the replacement of $\underline{x}_t$ by $\widetilde{\underline{x}}_t = \underline{x}_k+\underline{d}$ multiplies the determinant of the matrix by the factor $\ell_t(\underline{x}_k+\underline{d})$. Furthermore, the details of this method for selecting t were influenced by numerical results.

5. Summary of the algorithm

The following summary of the algorithm is divided into steps, where each step refers to the relevant part of the material of the previous three sections. Every iteration begins at Step 3. Here a zero value of the integer variable j indicates that the decision between the alternatives is that $\underline{d}$ will be calculated by solving the problem (1.7). This use of j complements the one that is mentioned in the first paragraph of Section 3, because j is positive when a model step is required. Some further details of the implementation are given after the summary.

Step 1: The user supplies the data that are specified in the opening paragraph of Section 1, namely the initial vector of variables $\underline{x}_b \in \mathcal{R}^n$, the parameters ρ_{beg} and ρ_{end} that determine the choices of ρ, and the subroutine that provides the value $F(\underline{x})$ of the objective function for any $\underline{x}$ in $\mathcal{R}^n$.

Step 2: The initialization procedure, explained in the first half of Section 4, generates the initial set of interpolation points, with the coefficients of the initial quadratic model and Lagrange functions. It also sets the values $\frac{1}{6}M = 0$, $\rho = \rho_{\text{beg}}$, $\Delta = \rho_{\text{beg}}$ and $j = 0$, and k becomes the least integer in $[1, m]$ that has the property

$$F(\underline{x}_k) \;=\; \min\,\{F(\underline{x}_i) : i = 1, 2, \ldots, m\}. \tag{5.1}$$

Step 3: If $j = 0$, then the problem (1.7) is solved by the method that is described in the first half of Section 2, which provides the trial step $\underline{d}$. Further, the number $\texttt{DNORM} = \|\underline{d}\|$ is noted, and there is a branch to Step 8 if $\texttt{DNORM} < \frac{1}{2}\rho$ occurs.

Step 4: The new value of the objective function $F(\underline{x}_k + \underline{d})$ is calculated, $\underline{d}$ being available at the beginning of the iteration if j is positive. The numbers $Q(\underline{x}_k + \underline{d})$ and $\ell_i(\underline{x}_k + \underline{d})$, $i = 1, 2, \ldots, m$, are computed too, using a technique that is given later in this section. Then the parameter $\frac{1}{6}M$ is overwritten by expression (3.2), and the value $\texttt{FOLD} = F(\underline{x}_k)$ is noted.

Step 5: If $j = 0$, then Δ is updated in the way that is the subject of the paragraph that includes expression (3.5). Further, t is selected by the method in the last paragraph of Section 4. Alternatively, if $j > 0$, then t is set to j.

Step 6: If $t > 0$, then the interpolation point $\underline{x}_t$ is moved to the position $\underline{\widetilde{x}}_t = \underline{x}_k + \underline{d}$, using the updating formulae (4.10), (4.11) and (4.13) to revise the coefficients of the Lagrange functions and the quadratic model. Moreover, the value of k is changed to t if $F(\underline{\widetilde{x}}_t) < \texttt{FOLD}$ occurs, which preserves equation (5.1). Let $\texttt{DMOVE}$ be the distance between the old position of $\underline{x}_t$ and $\underline{x}_k$ for the new value of k.

Step 7: The tests that are stated in the two complete paragraphs after inequality (3.6) are tried. Specifically, if $t > 0$, and if at least one of the four conditions

$$j > 0, \quad F(\underline{\widetilde{x}}_t) < \texttt{FOLD}, \quad \texttt{DNORM} > 2\rho \quad \text{and} \quad \texttt{DMOVE} > 2\rho \tag{5.2}$$

holds, then j is set to zero and there is a branch to Step 3, in order to begin an iteration that calculates a trust region step.

Step 8: The procedure in the paragraph that includes expression (3.12) is employed to seek a positive integer j for a model step. If one is found, then the model step $\underline{d}$ will have been calculated by applying the method in the second half of Section 2 to the problem (1.8). Otherwise, either the conditions (3.7) are satisfied or the search for $j>0$ has exhausted the set $\mathcal{J}$, so j is set to zero.

Step 9: As mentioned soon after expression (3.12), there is a branch to Step 3 for a new iteration either if j is positive or if both $j=0$ and $\texttt{DNORM}>\rho$ occur.

Step 10: If $\rho>\rho_{\text{end}}$, then the algorithm performs the operations of the paragraph that includes equation (3.13). They decrease ρ and Δ and revise $\underline{x}_b$ before branching to Step 3 for the next iteration. Since j is already zero, the next iteration will generate a trust region step by solving the problem (1.7).

Step 11: The iterations are now complete, but one more value of F may be required before termination. Indeed, we recall from the last paragraph of Section 3 that $F(\underline{x}_k+\underline{d})$ is calculated if $\texttt{DNORM}<\frac{1}{2}\rho$, and then $\underline{x}_k$ is overwritten by $\underline{x}_k+\underline{d}$ if the reduction (3.6) is achieved. Finally, the current $\underline{x}_k$ is returned to the user as the best estimate of the optimal vector of variables. $\square$

Unfortunately, the amount of work of both Step 4 and Step 6 is $\mathcal{O}(n^4)$, because every coefficient of every Lagrange function is relevant. Therefore these expensive parts of the algorithm are simplified as much as possible. Specifically, the required coefficients of the quadratic model are held in a vector $\underline{v}_Q\in\mathcal{R}^{n(n+3)/2}$, whose first n and last $\frac{1}{2}n(n+1)$ entries are the components of $\underline{g}_Q$ and the elements of the lower triangular and diagonal parts of G_Q, respectively. Further, for each integer i in $[1,m]$, the required coefficients of the Lagrange function ℓ_i are stored similarly in a single vector $\underline{v}_i\in\mathcal{R}^{n(n+3)/2}$. Then the updating formula (4.10) requires $\underline{v}_t$ to be multiplied by $[\ell_t(\underline{x}_k+\underline{d})]^{-1}$, and, using this new $\underline{v}_t$, formulae (4.11) and (4.13) require $\underline{v}_i$, $i\neq t$, and $\underline{v}_Q$ to be overwritten by the vectors

$$\underline{v}_i-\ell_i(\underline{x}_k+\underline{d})\,\underline{v}_t \quad\text{and}\quad \underline{v}_Q+[F(\underline{x}_k+\underline{d})-Q(\underline{x}_k+\underline{d})]\,\underline{v}_t, \tag{5.3}$$

respectively. Moreover, in Step 4 we make use of the observation that equation (1.1), the symmetry of G_Q and the definition of $\underline{v}_Q$ give the identity

$$Q(\underline{x}_k+\underline{d})-Q(\underline{x}_k)=\underline{g}_Q^T\underline{d}+\underline{d}^TG_Q\,(\underline{x}_k-\underline{x}_b)+\tfrac{1}{2}\,\underline{d}^TG_Q\,\underline{d}=\underline{v}_Q^T\underline{w}, \tag{5.4}$$

where $\underline{w}\in\mathcal{R}^{n(n+3)/2}$ does not depend on $\underline{g}_Q$ and G_Q. Its components are calculated from the remark that, if $j(p,q)$ is the integer in $[n+1,\frac{1}{2}n(n+3)]$ that is defined by $(\underline{v}_Q)_{j(p,q)}=(G_Q)_{pq}$, $1\le p\le q\le n$, then they have the values

$$\left.\begin{array}{rcll} w_j &=& d_j, & j=1,2,\ldots,n, \\ w_{j(p,q)} &=& d_p\,(\underline{x}_k-\underline{x}_b)_q+d_q\,(\underline{x}_k-\underline{x}_b)_p+d_p\,d_q, & 1\le p<q\le n, \\ w_{j(p,p)} &=& d_p\,(\underline{x}_k-\underline{x}_b)_p+\frac{1}{2}\,d_p^2, & p=1,2,\ldots,n. \end{array}\right\} \tag{5.5}$$

Thus $\underline{w}$ is formed in $\mathcal{O}(n^2)$ operations, and then $Q(\underline{x}_k+\underline{d})-Q(\underline{x}_k)$ is just the scalar product $\underline{v}_Q^T\underline{w}$. Now equation (5.4) remains true with no change to $\underline{w}$ if Q is replaced by ℓ_i on the left hand side and if the subscript Q is replaced by i elsewhere. Hence we find the formulae

$$\ell_i(\underline{x}_k+\underline{d}) - \ell_i(\underline{x}_k) = \underline{v}_i^T\underline{w}, \qquad i=1,2,\ldots,m. \tag{5.6}$$

Therefore the $\mathcal{O}(n^4)$ part of Step 4 is only the calculation of the scalar products (5.6), where $\underline{w}$ has the components (5.5).

The method of the previous paragraph is the reason for the statement, made in Section 1, that there is no need to retain the coefficients c_Q and c_j, $j=1,2,\ldots,m$, of the functions (1.1) and (1.4). Indeed, the values of $Q(\underline{x}_k+\underline{d})$ and $\ell_i(\underline{x}_k+\underline{d})$, $i=1,2,\ldots,m$, are obtained from the equations

$$\left.\begin{array}{rcccl} Q(\underline{x}_k+\underline{d}) &=& Q(\underline{x}_k)+\underline{v}_Q^T\underline{w} &=& F(\underline{x}_k)+\underline{v}_Q^T\underline{w} \qquad \text{and} \\ \ell_i(\underline{x}_k+\underline{d}) &=& \ell_i(\underline{x}_k)+\underline{v}_i^T\underline{w} &=& \delta_{ik}+\underline{v}_i^T\underline{w}, \qquad i=1,2,\ldots,m. \end{array}\right\} \tag{5.7}$$

Furthermore, we see that $F(\underline{x}_k)$ is the only one of the function values $F(\underline{x}_i)$, $i=1,2,\ldots,m$, that has to be available at the start of an iteration of the algorithm, because we do not work with the system (1.2) explicitly. Therefore $F(\underline{x}_k)$ is one of the other numbers that are mentioned at the end of the opening paragraph of Section 3. It is updated occasionally. Specifically, Step 6 of the algorithm reduces $F(\underline{x}_k)$ from `FOLD` to $F(\underline{\tilde{x}}_t)$ when the value of k is changed to t, because $F(\underline{\tilde{x}}_t)<$ `FOLD` occurs.

Only one more quantity is present among the other numbers that are required at the beginning of each iteration, namely `NF`, which is the number of values of $F(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, that have been calculated so far. Of course it is set to m in Step 2 of the algorithm, and it is increased by one in Step 4. One purpose of `NF` is that the choice of j for a model step, given in the paragraph that includes inequality (3.12), depends on the number of updates of $\frac{1}{6}M$ that have been made. The algorithm employs the remark that this number has the value `NF` $-\,m$. Furthermore, the user may prescribe an upper bound, `NFMAX` say, on the number of calls of the subroutine that generates values of the objective function. Then there is a return from Step 4 if `NF`=`NFMAX` holds at the beginning of the step, the current $\underline{x}_k$ being the final vector of variables.

One other situation may cause an early return. It is due to the fact that the method for revising Δ in Step 5 is suitable only if the denominator $Q(\underline{x}_k)$–$Q(\underline{x}_k+\underline{d})$ of the ratio (3.4) is positive. This happens in theory, because $\underline{d}$ is a solution of the problem (1.7) that has the property $\|\underline{d}\|\geq\frac{1}{2}\rho$, but damage may be caused by computer rounding errors. Therefore, if j is zero at the beginning of Step 5, there is a check on the computed value of the scalar product (5.4). The calculations are terminated if $\underline{v}_Q^T\underline{w}\geq 0$ occurs, and again the current $\underline{x}_k$ is the final vector of variables. An example of this early termination is shown in the numerical results of the next section.

Finally, we mention a technique that reduces the work of Step 8. It depends on the elementary bound

$$\begin{aligned} |\ell_j(\underline{x}_k+\underline{d})| &= |\,\delta_{jk} + \underline{g}_k^T\underline{d} + \underline{d}^T G_j\,(\underline{x}_k-\underline{x}_b) + \tfrac{1}{2}\,\underline{d}^T G_j\,\underline{d}\,| \\ &\leq \|\underline{d}\|\,\|\,\underline{g}_k + G_j\,(\underline{x}_k-\underline{x}_b)\,\| + \tfrac{1}{2}\,\|\underline{d}\|^2\,\|G_j\|_F, \qquad j\neq k, \end{aligned} \tag{5.8}$$

on the function (1.4), where $\|G_j\|_F$ is the Frobenius norm $[\sum_{p=1}^n \sum_{q=1}^n (G_j)_{pq}^2]^{1/2}$. Thus, for $j\neq k$, the solution $\underline{d}$ of the problem (1.8) achieves the condition (3.12) if the coefficients of ℓ_j have the property

$$\tfrac{1}{6} M\,\|\underline{x}_j-\underline{x}_k\|^3 \left[\rho\,\|\,\underline{g}_k + G_j\,(\underline{x}_k-\underline{x}_b)\,\| + \tfrac{1}{2}\,\rho^2\,\|G_j\|_F\right] \leq \varepsilon. \tag{5.9}$$

Now testing this inequality requires much less effort than the approximate solution of the problem (1.8), although the complexity of both tasks is $\mathcal{O}(n^2)$. Therefore, when $\varepsilon>0$, and when the algorithm is asking whether j should be removed from $\mathcal{J}$, inequality (5.9) is tried first. Of course j is discarded from $\mathcal{J}$ if the inequality holds, but otherwise the problem (1.8) is solved and condition (3.12) is tested as described already. We recall from the end of Section 2 that in some numerical experiments the calculation (1.8) occurred 8466 and 194 times for $n=20$ and $n=5$, respectively. Those counts would increase to 16434 and 785, however, if the technique of this paragraph were not included in the algorithm.

6. Numerical results and discussion

The development of the UOBYQA software was guided by numerical experiments, using objective functions of the form

$$F(\underline{x}) = \sum_{i=1}^n \Big[a_i - \sum_{j=1}^n (S_{ij}\sin x_j + C_{ij}\cos x_j)\Big]^2, \qquad \underline{x}\in\mathcal{R}^n. \tag{6.1}$$

The way of generating the parameters of F is taken from Fletcher and Powell (1963), and is as follows. The elements of the $n\times n$ matrices S and C are random integers from the interval $[-100,100]$, and a vector $\underline{x}_*$ is chosen whose components are random numbers from $[-\pi,\pi]$. Then the parameters a_i, $i=1,2,\ldots,n$, are defined by the equation $F(\underline{x}_*)=0$, and the starting vector $\underline{x}_b$ is formed by adding random perturbations from $[-0.1\pi, 0.1\pi]$ to the components of $\underline{x}_*$. All distributions of random numbers are uniform. The remaining data for Step 1 in Section 5 are $\rho_{\text{beg}}=0.1$ and $\rho_{\text{end}}=10^{-8}$. The number of variables is restricted severely by the $\mathcal{O}(n^4)$ work of each iteration, the calculations being done on a Sun Sparc 2 or Sparc 10 workstation. The values $n=3$, $n=5$, $n=10$ and $n=20$ were selected for most of the trials.

An advantage of the random numbers is that it is easy to generate many different objective functions. We are going to consider 20 of them, 5 for each of

n	Values of `NFTOT`				
3	37	46	35	115*	45
5	75	74	110	69	63
10	229	455	222	398	645*
20	1150	760	1502	1480	1074

Table 1: UOBYQA applied to functions of the form (6.1)

the values of n that have been mentioned. These test functions provide two other features that are also helpful to learning by experiments. Firstly, because the number of terms in the sum of squares (6.1) is equal to the number of variables, it happens often that the second derivative matrix $\nabla^2 F$ is ill-conditioned at the required solution. Secondly, because F is periodic, it has many saddle points and maxima. The UOBYQA software responds to these challenges very well. Indeed, the greatest final value of F in the 20 trials is 2.48×10^{-14}, and in that case the greatest modulus of a difference between a variable and the corresponding component of $\underline{x}_*$ is only 6.56×10^{-10}, which is substantially less than ρ_{end}. In two of the trials, however, the algorithm finds an optimal vector of variables that is different from $\underline{x}_*$.

The total number of calculations of F in each of the 20 trials, `NFTOT` say, is given in Table 1. The asterisks indicate the two problems in which the final $\underline{x}$ is far from $\underline{x}_*$. The entries in the table suggest that those two problems are relatively hard. and the variations in the entries for each n show that the different random numbers provide several degrees of difficulty. On the other hand, the table is not suitable for estimating the dependence of `NFTOT` on n. Nevertheless, our results can be compared with those of Powell (1964) for a conjugate direction method in the case (6.1). One finds that the UOBYQA software requires fewer function evaluations, which is some compensation for the huge amount of routine work that occurs in the construction of quadratic models by interpolation.

The usefulness of inequality (3.12) to the speed of convergence of UOBYQA was investigated numerically. Those studies are reported in Powell (2000), but the main conclusions are repeated now with some additional comments because of their importance. That work was also addressed by the author in his talk at the 17th Symposium of the Mathematical Programming Society in Atlanta. We begin the present discussion by recalling from Steps 8 and 9 of the summary of Section 5 that, if the computations of UOBYQA with the current ρ are complete, then one or both of the conditions

$$\|\underline{x}_j - \underline{x}_k\| \le 2\rho \quad \text{and} \quad \tfrac{1}{6} M \, \|\underline{x}_j - \underline{x}_k\|^3 \, |\ell_j(\underline{x}_k + \underline{d})| \le \varepsilon \tag{6.2}$$

$n=5$		ρ_{old}	$n=20$	
NF	FBEST		NF	FBEST
46	6.6×10^{-1}	10^{-1}	488	1.0×10^{0}
68	4.5×10^{-4}	10^{-2}	984	9.5×10^{-4}
90	7.1×10^{-6}	10^{-3}	1259	1.8×10^{-6}
92	2.1×10^{-8}	10^{-4}	1352	1.5×10^{-8}
98	1.4×10^{-9}	10^{-5}	1393	2.5×10^{-10}
102	5.4×10^{-12}	10^{-6}	1424	1.9×10^{-12}
105	1.2×10^{-16}	10^{-7}	1493	3.5×10^{-14}
110	7.2×10^{-20}	10^{-8}	1502	2.9×10^{-18}

Table 2: The calculations so far when ρ is reduced

must hold for every integer j in $[1,m]$, where $\underline{d}$ is an approximation to the solution of the problem (1.8). When ρ is reduced by formula (3.13), however, then it is usual for the interpolation points to satisfy $\|\underline{x}_j-\underline{x}_k\|>2\rho$, $j\neq k$, for the new value of ρ, because of the techniques that keep the interpolation points apart for the old ρ. Therefore, if the bounds $\|\underline{x}_j-\underline{x}_k\|\leq 2\rho$, $j=1,2,\ldots,m$, have to be achieved eventually for the new ρ, then the algorithm may have to move all but one of the interpolation points, which would require $m-1=\frac{1}{2}n(n+3)$ new values of the objective function. The purpose of the alternative test for the acceptability of $\underline{x}_j$, namely inequality (3.12), which is the second of the conditions (6.2), is to reduce the number of points that have to be moved for each new ρ. It follows that the use of this inequality is successful if UOBYQA calculates fewer than $\frac{1}{2}n^2$ new values of F for most of its choices of ρ.

The UOBYQA software is highly successful in this way throughout the numerical experiments of Table 1. For example, some details are given in Table 2 for the $n=5$ and $n=20$ calculations that require 110 and 1502 values of the objective function, respectively. Each row of the table states the number of function values so far, namely `NF`, and the least value of F so far, namely `FBEST`, when the iterations with $\rho=\rho_{\text{old}}$ are complete. We see that a substantial improvement in accuracy is achieved after each reduction in ρ, and that the criterion for success, explained at the end of the previous paragraph, is satisfied easily in the last five rows of the table. Therefore the algorithm makes very good use of the bound (3.1) on the error of the quadratic model.

If one ignores computer rounding errors, then it may be possible to show that the excellent results in Table 2 as ρ_{old} decreases are due to superlinear convergence. In other words, the average number of new values of the objective function for each ρ may tend to zero as $\rho \to 0$. We consider this conjecture briefly, assuming that the number of reductions in ρ by a factor of ten is infinite, that F has bounded third derivatives, that the relevant vectors of variables converge to a local minimum $\underline{x}_*$ of the objective function, that $\nabla^2 F(\underline{x}_*)$ is positive definite, and that the techniques for adjusting the positions of the interpolation points provide the property

$$\max \{|\ell_j(\underline{x}_k+\underline{d})| : \|\underline{d}\| \le \rho\} \;\le\; c, \qquad j=1,2,\ldots,m, \tag{6.3}$$

where $\underline{x}_k$ satisfies equation (5.1) as usual, and where c is a positive constant. These assumptions imply that Δ and $\|\underline{x}_k-\underline{x}_*\|$ are of magnitude ρ and bounded above by a constant multiple of ρ, respectively. It follows from the second of the conditions (6.2) that, if an iteration picks $\underline{x}_j$ for a move by a model step, and if ε is at least a positive multiple of ρ^2, which is a likely consequence of the definition (3.3), then the magnitude of $\|\underline{x}_j-\underline{x}_*\|$ is at least $\rho^{2/3}$. Hence the new position $\underline{\widetilde{x}}_t$ of an interpolation point in Step 6 of Section 5 is not going to be disturbed by a model step move, until ρ has decreased from its present value, $\rho_{\rm old}$ say, to one of magnitude $\rho_{\rm old}^{3/2}$, which implies that the average number of model step moves for each ρ does tend to zero. This property is also expected for the number of calculations of $F(\underline{x}_k+\underline{d})$ when the trial step $\underline{d}$ solves the problem (1.7). Indeed, because improvements to the quadratic model should cause any Newton–Raphson steps to converge superlinearly, we have only to consider solutions of the problem (1.7) that are prevented from being Newton–Raphson steps by the bound $\|\underline{d}\| \le \Delta$. Now the assumptions should imply that, after a finite number of iterations, all the trust region steps give the decrease (3.6) in the objective function. Therefore the reductions in ρ occur because a Newton–Raphson step satisfies $\|\underline{d}\| < \frac{1}{2}\rho$. Then the first choice of Δ for the new ρ, namely $\max[\frac{1}{2}\rho_{\rm old}, \rho_{\rm new}]$, ensures that the next solution of problem (1.7) provides the same $\underline{d}$. Further, because of the goodness of the quadratic model, formula (3.5) should not reduce Δ. It follows that all the trust region steps may become Newton–Raphson steps eventually, giving superlinear convergence. It would be better, however, to establish this property without some of the assumptions that have been made.

The calculation should terminate after a finite number of iterations if $\rho_{\rm end}$ is positive, but the only analysis of this question so far depends on the limited precision of computer arithmetic. Specifically, because the total number of different values of the objective function is finite in practice, a least calculated value must occur, so the point $\underline{x}_k$ that satisfies equation (5.1) becomes fixed. Then, for each ρ, every trust region iteration is followed by a model step iteration, and the trust region iterations reduce Δ until $\Delta = \rho$ holds. Now, when $\underline{x}_k$ is fixed, the trust region steps with $\Delta = \rho$ do not decrease the number of integers $j \in [1, m]$ that satisfy $\|\underline{x}_j - \underline{x}_k\| \le 2\rho$, while every model step increases this number. Thus the number of iterations for each ρ is finite. Further, the number of reductions in ρ is also finite, which completes the proof of termination.

n	Values of NFTOT				
3	37	64	43	137*	54
5	92	93	155	87	83
10	290	590	285	560	728*
20	1361	964	1939	2010	1365

Table 3: Examples of deterioration due to rounding errors

n	Values of NFTOT				
3	115	112	122	127	117
5	396	245	330	393	270
10	978	1120	1266	1564	1074
20	5385	6022	5269	4517	4821

Table 4: Examples with first derivative discontinuities

We investigate the effects of computer rounding errors by repeating the calculations of Table 1, after adding the constant 10^4 to the objective function (6.1), but $\rho_{\text{beg}}=0.1$ and $\rho_{\text{end}}=10^{-8}$ are retained. The new values of NFTOT are given in Table 3. The additional work in comparison with Table 1 can be attributed to the impossibility of constructing good quadratic models by interpolation, when typical distances between interpolation points are of magnitude $\rho=10^{-7}$ or $\rho=10^{-8}$. Indeed, when the new objective function is tried in the Table 2 calculations, the new values of NF in the $\rho_{\text{old}}=10^{-6}$ row are 102 and 1430 for $n=5$ and $n=20$, respectively, and the corresponding computed values of FBEST-10^4 are 3.6×10^{-12} and 1.8×10^{-11}, so the damage to changes in the variables from rounding errors occurs for $\rho\leq10^{-7}$. Moreover, in all the test problems of Table 3, a computed value of FBEST-10^4 is exactly zero before ρ reaches its final value. Therefore all the work of UOBYQA with $\rho=10^{-8}$ makes no difference to the final vector of variables. Further, the computer output shows that FBEST-10^4 is always an integer multiple of 1.8189894×10^{-12} in practice. Such rounding errors cause $\frac{1}{6}M$ to become huge, due to the difficulties of estimating third derivatives from function values. Thus the second of the conditions (6.2), which is important to superlinear convergence in theory, is ineffective when ρ is too small. These remarks may be helpful to the choice of ρ_{end}.

We tried applying UOBYQA to some functions with discontinuous first deriva-

n	Values of NFTOT					$q(n)$
3	36	36	35	31	35	34
5	64	54	59	67	64	63
10	161	159	161	185	157	164
20	509	473	488	486	467	486

Table 5: Some well-conditioned least squares calculations

tives. In the first of these experiments, the objective function is formed by replacing the sum of squares of expression (6.1) by the sum of moduli of the terms in square brackets. Then the 20 calculations of Table 1 are repeated, without changing the values of the parameters. Unfortunately, the distance from the final $\underline{x}_k$ to a local minimum is less than ρ_{end} in only two cases with $n=3$. For $n=5$, these distances are 3×10^{-2}, 1×10^{-3}, 1×10^{-1}, 7×10^{-4} and 6×10^{-6}, while only one of the $n=10$ and one of the $n=20$ trials yields $\|\underline{x}_k-\underline{x}_*\|_\infty<0.1$. It seems that the ill-conditioning, mentioned in the second paragraph of this section, is too severe. Therefore UOBYQA was also applied to the function

$$F(\underline{x}) = \sum_{i=1}^{2n} \Big| a_i - \sum_{j=1}^{n} (S_{ij} \sin x_j + C_{ij} \cos x_j) \Big| , \qquad \underline{x}\in\mathcal{R}^n, \tag{6.4}$$

S and C being matrices of size $2n\times n$. Their elements and $\underline{x}_*\in\mathcal{R}^n$ are generated as before. Then a_i, $i=1,2,\ldots,2n$, are defined by $F(\underline{x}_*)=0$, and as usual the components of $\underline{x}_b$ differ from those of $\underline{x}_*$ by random numbers from $[-0.1\pi, 0.1\pi]$. We retain $\rho_{\text{beg}}=0.1$ and $\rho_{\text{end}}=10^{-8}$, and again five test problems are generated for each of the choices $n=3$, $n=5$, $n=10$ and $n=20$. The numbers of calculations of the objective function that occurred are reported in Table 4. They are large because quadratic functions are unsuitable for modelling first derivative discontinuities. On the other hand, the accuracy that is achieved is excellent. Indeed, one of the final values of $\|\underline{x}_k-\underline{x}_*\|_\infty$ is 2.03×10^{-8}, and the final vectors of variables of all the other 19 cases are better because they satisfy $\|\underline{x}_k-\underline{x}_*\|_\infty<\rho_{\text{end}}$.

We employ the smooth least squares form

$$F(\underline{x}) = \sum_{i=1}^{2n} \Big[a_i - \sum_{j=1}^{n} (S_{ij} \sin x_j + C_{ij} \cos x_j) \Big]^2 , \qquad \underline{x}\in\mathcal{R}^n, \tag{6.5}$$

of the function (6.4) to study the dependence of the amount of work of UOBYQA on n. The numbers of calculations of the objective function in this case for the usual 20 test problems are given in Table 5, where all the parameters are taken from the experiments of Table 4, including $\rho_{\text{beg}}=0.1$ and $\rho_{\text{end}}=10^{-8}$. We see

n	Total time in seconds	Percentage of total time spent on			
		Prob. (1.7)	Prob. (1.8)	Eqn. (5.7)	Updating
3	0.020–0.022	13%–16%	4%–6%	10%–13%	12%–16%
5	0.10–0.13	10%–12%	3%–5%	15%–17%	17%–20%
10	1.81–2.25	6%–7%	2%–4%	24%–25%	28%–30%
20	48.4–56.7	3%–4%	1%–3%	32%–35%	49%–50%

Table 6: Some timings of the Table 5 examples

that the variations in the values of `NFTOT` in each row are much less than before. Further, these entries can be fitted quite well by the quadratic polynomial

$$q(n) \;=\; 0.8\,n^2 + 8.2\,n + 2, \qquad n\!=\!3,5,10,20, \tag{6.6}$$

as shown by the rounded values in the last column of the table. Moreover, after every Table 5 calculation, the vector of variables $\underline{x}_k$ satisfies $\|\underline{x}_k - \underline{x}_*\|_\infty < \rho_{\rm end}$. Thus we find that expression (6.5) provides some straightforward examples of successful applications of UOBYQA. Therefore we use the Table 5 calculations to illustrate computation times, which are measured in seconds by the Fortran function DTIME on a Sun Sparc 10 workstation. Special attention is given to the two trust region subproblems of Section 2, and to the tasks of an iteration that require $\mathcal{O}(n^4)$ operations, namely formula (5.7) for generating $\ell_i(\underline{x}_k+\underline{d})$, $i\!=\!1,2,\ldots,m$, and the updating in Step 6 of Section 5. The times of each task were obtained by comparing UOBYQA with a version that was programmed to perform the particular task under consideration more than once. The results of these investigations are reported in Table 6, the range of each entry in the body of the table being as follows. The value of the stated quantity was found for each of the five trials in the relevant row of Table 5, and then the range gives the least and greatest values that occurred. The lengths of the ranges depend not only on the random parameters of the objective functions, but also on some roughness in the method of DTIME. We see in Table 6 that the work of solving the subproblems of Section 2 is not excessive, and that, if $n\!\geq\!10$, then more than half of the time is taken by formula (5.7) and the updating. These expensive operations are so simple that it would be easy to perform them efficiently on a parallel machine.

Finally, we consider a few well-known test problems that are without random numbers. Many researchers have employed Rosenbrock's function

$$F(\underline{x}) \;=\; 100(x_2-x_1^2)^2 + (1-x_1)^2, \qquad \underline{x}\!\in\!\mathcal{R}^2, \tag{6.7}$$

and the "singular" function

$$F(\underline{x}) = (x_1\!+\!10x_2)^2 + 5(x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4, \qquad \underline{x}\!\in\!\mathcal{R}^4. \tag{6.8}$$

Rosenbrock's function (6.7)		ρ_{old}	The singular function (6.8)	
NF	FBEST		NF	FBEST
39	3.5×10^{-1}	10^{-1}	75	4.0×10^{-4}
82	1.3×10^{-6}	10^{-2}	123	7.1×10^{-9}
87	5.2×10^{-8}	10^{-3}	160	9.3×10^{-12}
93	3.3×10^{-13}	10^{-4}	214	2.9×10^{-16}
94	3.3×10^{-13}	10^{-5}	262	9.0×10^{-21}
96	6.5×10^{-15}	10^{-6}	295	3.1×10^{-23}
98	2.7×10^{-21}	10^{-7}	348	9.0×10^{-29}
100	7.1×10^{-23}	10^{-8}	386	4.5×10^{-34}

Table 7: UOBYQA applied to the functions (6.7) and (6.8)

We do so too, with the usual starting points $(-1.2, 1.0)$ and $(3.0, -1.0, 0.0, 1.0)$, respectively, and we retain $\rho_{\text{beg}}=0.1$ and $\rho_{\text{end}}=10^{-8}$. Table 7 is the analogue of Table 2 for these trials. We see that the second part of expression (6.2) allows an excellent rate of convergence in the case (6.7), but there are more than 30 new calculations of F for each ρ in the other case, because of the singularity of $\nabla^2 F$ at the solution. The early termination, mentioned in the penultimate paragraph of Section 5, occurred when UOBYQA was applied to the function (6.8), so the tiny value of the objective function at termination, namely $F=4.5\times10^{-34}$, shows that excellent accuracy is achieved before further progress is prevented by computer rounding errors.

The Chebyquad objective functions of Fletcher (1965) were also minimized by UOBYQA. They are sums of squares, and their variables are n points of a quadrature formula over the interval $[0,1]$, the starting values of the variables being $x_i = i/(n+1)$, $i=1,2,\ldots,n$. Therefore $\rho_{\text{beg}} = 0.2/(n+1)$ is suitable, and we try $\rho_{\text{beg}}=0.1$ and $\rho_{\text{beg}}=0.01$ too. We persevere with the choice $\rho_{\text{end}}=10^{-8}$, and n takes the values 2, 4, 6 and 8. The middle three columns of Table 8 show the total number of function calculations that UOBYQA requires in each of these cases. Good accuracy occurs, every final value of $F(\underline{x}_k)$ being within 2×10^{-17} of the least value of the objective function, F_* say, which is zero for $n=2$, 4 and 6, and about 0.0035 for $n=8$. Fletcher (1965) uses the Chebyquad examples to compare three old algorithms for unconstrained minimization without derivatives, the best results being obtained by the conjugate direction method of Powell (1964) that has been mentioned already. The entries in the last column of Table 8 are taken from Fletcher (1965). They are the number of function evaluations when the method of Powell (1964) reduces the objective function to about F_*+10^{-13}.

n	Choices of $\rho_{\rm beg}$			Powell (1964)
	0.1	$0.2/(n+1)$	0.01	
2	24	25	29	41
4	59	73	82	91
6	186	135	155	288
8	394	244	263	537

Table 8: Values of `NFTOT` for the Chebyquad examples

We see that the values of `NFTOT` in the middle three columns of Table 8 depend quite strongly on $\rho_{\rm beg}$. In fact the differences between the columns are mainly due to the calculations with $\rho \geq 10^{-4}$. Hence one can take the view that much of the efficiency has to be achieved from good strategies instead of from the fine details of quadratic models. Moreover, Broyden, Dennis and Moré (1973) prove that close estimates of all second derivatives are not necessary for the superlinear convergence of quasi-Newton methods for unconstrained optimization, although some aspects of $\nabla^2 F$ have to be quite accurate. These remarks suggest that it may be possible to advance the techniques of UOBYQA in a way that preserves good performance, and that avoids the need for all the parameters of all the Lagrange functions to be available. Now, however, we infer from Table 6 that UOBYQA becomes prohibitively expensive for more than 50 variables, the cause of the ceiling being the $\frac{1}{2}(n+1)(n+2)$ degrees of freedom in the quadratic model Q. There are many applications of unconstrained optimization with no more than 20 variables. Furthermore, any sparsity in $\nabla^2 F$ can be inherited by the quadratic model and the Lagrange functions, which may provide huge reductions in the number of degrees of freedom, but software for this straightforward extension to UOBYQA has not been written yet. The three main advantages of the present version are that it is easy to apply, it usually gives good accuracy, even in some cases when F has discontinuous first derivatives, and, in the small range of comparisons that have been made, it seems to require fewer function evaluations than other algorithms. The Fortran 77 code was developed for general use, and is available free of charge from the author at the e-mail address mjdp@cam.ac.uk.

Acknowledgement

The author is very grateful to three referees for their careful consideration of this paper. They made several good suggestions that have improved the presentation.

References

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

A.R. Conn, K. Scheinberg and Ph.L. Toint (1997a), "On the convergence of derivative-free methods for unconstrained optimization", in *Approximation Theory and Optimization*, eds. M.D. Buhmann and A. Iserles, Cambridge University Press (Cambridge), pp. 83–108.

A.R. Conn, K. Scheinberg and Ph.L. Toint (1997b), "Recent progress in unconstrained nonlinear optimization without derivatives", *Math. Programming*, Vol. 79, pp. 397–414.

R. Fletcher (1965), "Function minimization without evaluating derivatives—a review", *Comput. J.*, Vol. 8, pp. 33–41.

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

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

B.N. Parlett (1980), *The Symmetric Eigenvalue Problem*, Prentice-Hall (Englewood Cliffs, N.J.).

M.J.D. Powell (1964), "An efficient method for finding the minimum of a function of several variables without calculating derivatives", *Comput. J.*, Vol. 7, pp. 155–162.

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 (1998), "Direct search algorithms for optimization calculations", *Acta Numerica*, Vol. 7, pp. 287–336.

M.J.D. Powell (2000), "On the Lagrange functions of quadratic models that are defined by interpolation", Report No. DAMTP 2000/NA10, University of Cambridge (to be published in *Optimization Methods and Software*).

D. Winfield (1973), "Function minimization by interpolation in a data table", *J. Inst. Maths Applics*, Vol. 12, pp. 339–347.