DAMTP 2002/NA02

On trust region methods for unconstrained minimization without derivatives[1]

M.J.D. Powell

Abstract: We consider some algorithms for unconstrained minimization without derivatives that form linear or quadratic models by interpolation to values of the objective function. Then a new vector of variables is calculated by minimizing the current model within a trust region. Techniques are described for adjusting the trust region radius, and for choosing positions of the interpolation points that maintain not only nonsingularity of the interpolation equations but also the adequacy of the model. Particular attention is given to quadratic models with diagonal second derivative matrices, because numerical experiments show that they are often more efficient than full quadratic models for general objective functions. Finally, some recent research on the updating of full quadratic models is described briefly, using fewer interpolation equations than before. The resultant freedom is taken up by minimizing the Frobenius norm of the change to the second derivative matrix of the model. A preliminary version of this method provides some very promising numerical results.

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

February, 2002 (revised August, 2002).

[1]Presented at NTOC 2001, Kyoto, Japan.

1. Introduction

Let the minimum of a function $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, be required, where F is defined by a subroutine that provides the value of $F(\underline{x})$ for any vector of variables $\underline{x}$. We assume that no derivatives of the objective function are available. The algorithms that have been developed for this calculation vary greatly in the use that is made of function values when deciding on changes to the variables. In simulated annealing, for example, each new vector of variables is a random move from a point where F is known, but, in order to achieve some global convergence properties, that point may not be where the least value of the objective function has been calculated so far. Moreover, if an algorithm employs a line search, then several consecutive new vectors of variables may be collinear, and also the choice of search direction may demand some extra function evaluations, especially if a gradient vector has to be estimated by finite differences. We are going to restrict attention to trust region methods, however, where an approximation to $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, is constructed from available function values. Then the next vector of variables is generated usually by seeking the minimum of the approximation in a suitable part of $\mathcal{R}^n$. The approximation is called the "model function".

We reserve the notation $Q(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, for the current model function, because often it is a quadratic polynomial. We take the view until Section 3, however, that any linear space $\mathcal{M}$ of functions from $\mathcal{R}^n$ to $\mathcal{R}$ can be prescribed, the dimension of $\mathcal{M}$ being the finite number m. Then every model function is an element of $\mathcal{M}$. The algorithm of Conn, Scheinberg and Toint (1997) begins the iterations before m values of F have been calculated, and some other algorithms pick each model function from $\mathcal{M}$ by weighted least squares fitting to all the values of F that are available. In the trust region methods that we consider until Section 5, however, each Q is an element of $\mathcal{M}$ that is defined by interpolation conditions of the form

$$Q(\underline{x}_i) \;=\; F(\underline{x}_i), \qquad i=1,2,\ldots,m, \tag{1.1}$$

the right hand sides being known. The points $\underline{x}_i \in \mathcal{R}^n$, $i=1,2,\ldots,m$, have to be in positions that ensure the nonsingularity of the system (1.1), a convenient way of satisfying this condition being described in Section 2.

When choosing the linear space $\mathcal{M}$, attention should be given to the amount of work that arises from solving a system of the form (1.1) on every iteration. In the trust region methods of the author, the only calculated values of F that are retained at the beginning of the current iteration are the m right hand sides of expression (1.1). Then the iteration generates at most one new value of F. Therefore at least $m-1$ of the current interpolation conditions are carried forward to the next iteration. It follows that, by applying updating techniques, each new model function Q after the first one can be generated in only $\mathcal{O}(m^2)$ operations. Unfortunately, however, if $\mathcal{M}$ contains all quadratic polynomials, then m has the value $\frac{1}{2}(n+1)(n+2)$, so the routine work of each iteration is $\mathcal{O}(n^4)$. Thus the use of full quadratic models becomes intolerable for more than about 50 variables. On the other hand, if $\mathcal{M}$ is the space of linear and constant polynomials, which has dimension $n+1$, then the construction of model functions is not expensive in comparison with other operations of trust region methods, but such models seem to be unsuitable for unconstrained optimization, because linear polynomials have no curvature.

Therefore we consider the idea of letting $\mathcal{M}$ be the space of quadratic polynomials that have diagonal second derivative matrices. In this case the task of updating Q is relatively easy, because the dimension of $\mathcal{M}$ is only $m=2n+1$. Further, we expect the presence of diagonal curvature to provide substantial improvements over the use of linear polynomial model functions. We also expect a trust region method with the new $\mathcal{M}$ to require more iterations than a trust region method with the full quadratic model, because less information about the objective function is present in Q. These questions are investigated by applying methods with the types of model that have been mentioned to several examples of unconstrained optimization calculations. We will find in Section 4 that some of the best numerical results are given by the new choice of $\mathcal{M}$.

All of the software that is employed in the experiments was written in Fortran 77 by the author. Specifically, the COBYLA (Powell, 1994) and UOBYQA (Powell, 2002) packages treat the cases when $\mathcal{M}$ is composed of all polynomials of degree at most one and all polynomials of degree at most two, respectively, and the results for the new $\mathcal{M}$ were computed by a modification of UOBYQA. Including the modification was straightforward, because already the author had developed a version of UOBYQA where the elements of $\mathcal{M}$ are quadratic polynomials whose second derivative matrices have any given sparsity structure, subject to symmetry and unrestricted diagonal elements. Then the same sparsity conditions occurred in the objective function $F(\underline{x})$, $\underline{x}\in\mathcal{R}^n$, but now we run the software for the new $\mathcal{M}$ even if all the second derivatives of F are nonzero. The name COBYLA is an acronym for Constrained Optimization BY Linear Approximation. That package is intended for calculations with constraints on the variables that provide compensation for the lack of curvature in the model function Q, but COBYLA can also be applied to unconstrained problems, because then a typical change to the variables is a multiple of the steepest descent direction of the current model function. On the other hand, UOBYQA is designed for unconstrained optimization.

Section 2 gives some details of these trust region methods that are valid when $\mathcal{M}$ is a general linear space. Some properties of our particular choices of $\mathcal{M}$ are addressed in Section 3. The results of the numerical experiments that have been mentioned are presented and discussed in Section 4. They provide strong reasons for the development of some new algorithms. Therefore another way of updating full quadratic models is considered briefly in Section 5. It minimizes the Frobenius norm of the change to $\nabla^2 Q$ when there are fewer than $\frac{1}{2}(n+1)(n+2)$ interpolation conditions.

2. Details of the methods

At the beginning of any iteration of our trust region methods, the model function $Q(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, defined by the interpolation conditions (1.1), is available. Further, we assume without loss of generality that $\underline{x}_1$ is the best of the interpolation points, which means that it has the property

$$F(\underline{x}_1) \;\le\; F(\underline{x}_i), \qquad i=1,2,\ldots,m. \tag{2.1}$$

If more than one of the function values $F(\underline{x}_i)$, $i=1,2,\ldots,m$, is least, we split the tie by letting $\underline{x}_1$ be the point at which the value $F(\underline{x}_1)$ was calculated first. The "suitable part of $\mathcal{R}^n$", mentioned in the first paragraph of Section 1, has the form

$$\mathcal{S} \;=\; \{\underline{x} :\; \|\underline{x}-\underline{x}_1\| \le \Delta\}, \tag{2.2}$$

for some positive parameter Δ, where the vector norm is Euclidean. The set $\mathcal{S} \subset \mathcal{R}^n$ is called the "trust region".

Many papers have addressed the calculation of the point $\underline{x}_\Delta$, say, in $\mathcal{S}$ that minimizes the model function $Q(\underline{x})$, $\underline{x} \in \mathcal{S}$. If Q is a linear polynomial, then $\underline{x}_\Delta$ is where the steepest descent direction of Q from $\underline{x}_1$ meets the boundary of the trust region, and, if Q is a quadratic polynomial, then the author prefers to generate $\underline{x}_\Delta$ by the method of Moré and Sorensen (1983). That method is iterative, the iterations being stopped in practice when an element $\underline{\widehat{x}}_\Delta$ of $\mathcal{S}$ is found that satisfies the condition

$$F(\underline{\widehat{x}}_\Delta) - F(\underline{x}_1) \;\le\; (1-\eta)\,[F(\underline{x}_\Delta) - F(\underline{x}_1)], \tag{2.3}$$

where η is a prescribed positive tolerance. Thus the choice $\eta = 0.01$, which is typical, ensures that the estimate $\underline{\widehat{x}}_\Delta \approx \underline{x}_\Delta$ provides at least 99% of the greatest reduction in F from $F(\underline{x}_1)$ that can be achieved within the trust region. We ignore the difference between $\underline{\widehat{x}}_\Delta$ and $\underline{x}_\Delta$ from now on, letting $\underline{x}_\Delta$ denote the calculated element of $\mathcal{S}$ that gives an acceptably small value of the model function.

The iterations of our trust region methods that generate $\underline{x}_\Delta$ are called "trust region iterations". Usually they calculate the function value $F(\underline{x}_\Delta)$, and then it is also usual for one of the interpolation points $\underline{x}_i$, $i=1,2,\ldots,m$, to be replaced by $\underline{x}_\Delta$. Further, Q is updated in order to satisfy the new interpolation equations (1.1). Details of these operations are given later. No other new values of the objective function are found by a trust region iteration, but a new vector of variables that is different from $\underline{x}_\Delta$ may be required. For example, we consider the possibility that the first component of the step $\underline{x}_\Delta - \underline{x}_1$ is zero on every iteration. In that case, if only trust region iterations were applied, and if the system of equations (1.1) remained nonsingular throughout the sequence of iterations, then some of the interpolation equations of the first model function would be retained. On the other hand, in order to achieve enough accuracy in the approximation $Q \approx F$, it may be necessary for all the interpolation points to be sufficiently close to $\underline{x}_1$. Therefore our methods also include some iterations that are called "model iterations". Each model iteration calculates the objective function at a new point, $\underline{x}_Q \in \mathcal{R}^n$ say, that is chosen to correct any tendencies towards singularity in the system (1.1), which assists the suitability of Q as an approximation to $F(\underline{x})$, $\underline{x} \in \mathcal{S}$.

The first iteration is always a trust region iteration, and a model iteration is always followed by a trust region iteration. Therefore the decision that the next iteration will be a model iteration is taken during a trust region iteration. Further, if that decision is made, then the trust region iteration picks the point $\underline{x}_Q$ and the integer, t say, from $[2, m]$, such that $\underline{x}_t$ will be rejected from the set $\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_m\}$ to make room for $\underline{x}_Q$. Thus the only operations of a model iteration are the calculation of $F(\underline{x}_Q)$, the updating of Q that is required because $Q(\underline{x}_t) = F(\underline{x}_t)$ is replaced by $Q(\underline{x}_Q) = F(\underline{x}_Q)$ in the system (1.1), all other interpolation conditions being retained, and exchanging $\underline{x}_1$ with $\underline{x}_Q$ if $F(\underline{x}_Q)$ is less than $F(\underline{x}_1)$.

It is possible for the vector $\underline{x}_\Delta$ of a trust region iteration to be the point $\underline{x}_1$, because $Q(\underline{x}_1) = F(\underline{x}_1)$ can be the least value of $Q(\underline{x})$, $\underline{x} \in \mathcal{S}$. In that case the calculation of $F(\underline{x}_\Delta)$ would be superfluous. Further, the use of $F(\underline{x}_\Delta)$ may be disadvantageous if the distance $\|\underline{x}_\Delta - \underline{x}_1\|$ is small, because then a model function that interpolates both $F(\underline{x}_1)$ and $F(\underline{x}_\Delta)$ tends to be sensitive to errors in the objective function, especially if the errors cause substantial discontinuities. Therefore, after generating $\underline{x}_\Delta$, each trust region iteration tests the condition

$$\|\underline{x}_\Delta - \underline{x}_1\| \;\geq\; \tfrac{1}{2}\,\rho \qquad (2.4)$$

for a choice of ρ that is addressed below. The function value $F(\underline{x}_\Delta)$ is calculated on the current iteration if and only if this condition holds. The parameter ρ is intended to provide large steps in the space of the variables during the early iterations, its initial value being prescribed. Further, when no more progress seems to be possible with the current value, ρ is reduced, except that termination occurs if ρ has reached its final value, which is also prescribed. Typical reductions are by a factor of ten, and ρ is never increased. The trust region radius Δ is either set to ρ on every iteration, or is adjusted in a usual way (see Fletcher, 1987, for instance), subject to the bound $\Delta \geq \rho$. The first and second of these alternatives are employed by COBYLA and UOBYQA, respectively.

If $F(\underline{x}_\Delta)$ is calculated on a trust region iteration, then the inequality

$$F(\underline{x}_\Delta) \;\leq\; F(\underline{x}_1) - 0.1\,[Q(\underline{x}_1) - Q(\underline{x}_\Delta)] \qquad (2.5)$$

is tested. In other words, we ask whether the step from $\underline{x}_1$ to $\underline{x}_\Delta$ reduces the objective function by at least one tenth of the amount that is predicted by the model, this amount being positive because of condition (2.4). The factor 0.1 on the right hand side of expression (2.5) can be altered to any other constant from the open interval $(0, 1)$. If the reduction (2.5) is achieved, then the next iteration is also a trust region iteration, which is begun after the usual updating that may revise Δ, that causes the new model function to satisfy $Q(\underline{x}_\Delta)=F(\underline{x}_\Delta)$, and that reorders the interpolation points so that $\underline{x}_\Delta$ is the new $\underline{x}_1$.

If $F(\underline{x}_\Delta)$ is calculated on a trust region iteration, but inequality (2.5) fails, then Δ is decreased if it exceeds ρ, and usually Q is revised, in order to include the new value of F in the next model function. Further, $\underline{x}_\Delta$ is exchanged with $\underline{x}_1$ whenever the reduction $F(\underline{x}_\Delta)<F(\underline{x}_1)$ occurs, although many other trust region algorithms move the centre of the region (2.2) only if the reduction in the objective function is sufficiently large, which means that $F(\underline{x}_\Delta)$ satisfies an inequality of the form (2.5). An advantage of preserving the conditions (2.1) is that, if $F(\underline{x}_\Delta)$ is calculated, then the strict inequalities

$$Q(\underline{x}_\Delta) \;<\; Q(\underline{x}_1) \;=\; F(\underline{x}_1) \;\le\; F(\underline{x}_i), \qquad i=1,2,\ldots,m, \tag{2.6}$$

hold. Thus the equations (1.1) ensure that $\underline{x}_\Delta$ is not one of the interpolation points $\underline{x}_i$, $i=1,2,\ldots,m$.

If condition (2.4) or (2.5) fails on a trust region iteration, then it is assumed that the next iteration will be a model iteration, so the algorithm makes a provisional choice of the index t of the interpolation condition that will be replaced. Specifically, this trial value of t is an integer from $[1,m]$ that has the property

$$\|\underline{x}_t-\underline{x}_1\| \;=\; \max\,\{\|\underline{x}_i-\underline{x}_1\| :\; i=2,3,\ldots,m\}, \tag{2.7}$$

and the ratio $\|\underline{x}_t-\underline{x}_1\|/\rho$ is compared with a prescribed constant $\beta>1$. In the early versions of our algorithms, the decision to employ a model iteration next is always taken if the ratio exceeds β. Each model iteration replaces $\underline{x}_t$ by $\underline{x}_Q$, as mentioned already, the choice of $\underline{x}_Q$ being given later. The UOBYQA software, however, tries to avoid the following disadvantage of the earlier versions. Because of the test (2.4), we expect most of the distances $\|\underline{x}_i-\underline{x}_1\|$, $i=2,3,\ldots,m$, to exceed $\frac{1}{2}\rho$ when ρ is going to be decreased. Moreover, the usual reductions in ρ are by a factor that is greater than 2β. Thus, after the reduction, most of these distances exceed $\beta\rho$. It follows that at least of magnitude m iterations are required to achieve the condition $\|\underline{x}_t-\underline{x}_1\|\le\beta\rho$ for the new value of ρ. This disadvantage is tolerable for COBYLA but not for UOBYQA, because the values of m are $n+1$ and $\frac{1}{2}(n+1)(n+2)$, respectively. Therefore, for each integer i in $[2,m]$, UOBYQA can generate a number, θ_i say, that is an estimate of the contribution to the error $F(\underline{x})-Q(\underline{x})$, $\underline{x}\in\mathcal{S}$, from the position of $\underline{x}_i$, and it calculates θ_*, say, which is an estimate of the reduction in F that is excluded by condition (2.4), θ_* being zero if the second derivative matrix of Q is not positive definite. Then a model iteration is applied next if and only if both $\|\underline{x}_t-\underline{x}_1\|>\beta\rho$ and $\theta_t>\theta_*$ hold for an interpolation point $\underline{x}_t$, where t can be different from the original trial value. Details are given in the description of UOBYQA (Powell, 2002).

The only remaining situation that can happen on a trust region iteration is that condition (2.4) or (2.5) fails, and it is found that there is no need to improve

Q by a model iteration. Then we ask whether the work using the current value of ρ is complete. The answer is negative if the distance $\|\underline{x}_\Delta - \underline{x}_1\|$ exceeds ρ, which is possible when $\Delta > \rho$ is allowed. In that case the current iteration will have calculated $F(\underline{x}_\Delta)$, and will have decreased Δ to a value that satisfies $\rho \leq \Delta < \|\underline{x}_\Delta - \underline{x}_1\|$, so a trust region iteration with the new Δ is performed next. Otherwise, the value of ρ seems to be preventing or impairing progress. Therefore termination occurs if ρ has reached its final value, or ρ is reduced and, because the tests of the previous paragraph suggest that the quadratic model is good, the next iteration is also a trust region iteration.

We now turn our attention to the Lagrange functions of the system (1.1), because they are highly useful to COBYLA and UOBYQA for maintaining nonsingularity of the system, as shown in the next paragraph, and for updating Q when one of the equations (1.1) is replaced by a new interpolation condition. The definition of the Lagrange function $\ell_i(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, where i is any integer from $[1, m]$, is that it is the element of the space $\mathcal{M}$ of model functions that satisfies the equations

$$\ell_i(\underline{x}_j) \;=\; \delta_{ji}, \qquad j=1,2,\ldots,m, \tag{2.8}$$

δ_{ji} being the Kronecker delta. These functions are also important to the calculation of the numbers θ_i, $i=2,3,\ldots,m$, by UOBYQA, mentioned in the paragraph that includes expression (2.7).

It is elementary that the system (1.1) is singular if and only if a nonzero element of $\mathcal{M}$ vanishes at all the points $\underline{x}_i$, $i=1,2,\ldots,m$. The positions of the interpolation points of the first iteration have to be chosen in a way that provides nonsingularity, this requirement being addressed in Section 3, and we find by induction how to preserve nonsingularity. Assume that singularity occurs for the first time when the interpolation point $\underline{x}_t$ is removed from the set $\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_m\}$, and let $\underline{x}_t^*$ be the new interpolation point, which is $\underline{x}_\Delta$ or $\underline{x}_Q$ on a trust region or a model iteration, respectively. Then, if the function $\ell^* \in \mathcal{M}$, say, vanishes on the new set of points, it must vanish on the set $\{\underline{x}_1, \underline{x}_2 \ldots, \underline{x}_m\} \backslash \{\underline{x}_t\}$, which implies that ℓ^* is a multiple of the old Lagrange function $\ell_t(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, because the old system of equations is nonsingular. It follows that the singularity of the new system is equivalent to the condition $\ell_t(\underline{x}_t^*) = 0$. Therefore our algorithms pick t and $\underline{x}_t^*$ in ways that ensure that $\ell_t(\underline{x}_t^*)$ is nonzero. Further, by relating Lagrange functions to ratios of determinants of matrices of systems of equations, it can be shown that relatively large values of $|\ell_t(\underline{x}_t^*)|$ are advantageous.

We recall that, when the decision is taken on a trust region iteration that a model iteration will be performed next, the integer $t \in [2, m]$ has been selected, but a new interpolation point $\underline{x}_Q = \underline{x}_t^*$ is required. It is chosen from the region

$$\mathcal{N} \;=\; \{\underline{x} : \|\underline{x} - \underline{x}_1\| \leq \rho\}, \tag{2.9}$$

and the remark at the end of the last paragraph suggests that it is optimal to let $\underline{x}_Q$ be the vector that maximizes $|\ell_t(\underline{x})|$, $\underline{x} \in \mathcal{N}$, which is done in both COBYLA and UOBYQA. This task is straightforward because the functions ℓ_t and $-\ell_t$ are both elements of $\mathcal{M}$, and the required $\underline{x}_Q$ minimizes one of these functions on $\mathcal{N}$. Therefore we can calculate $\underline{x}_Q$ by two applications of the procedure that is available already for minimizing the model function $Q(\underline{x})$, $\underline{x} \in \mathcal{S}$.

On a trust region iteration that calculates $F(\underline{x}_\Delta)$, the new interpolation condition is usually included in the model function, by replacing the point $\underline{x}_t$ by $\underline{x}_\Delta$, where the integer t has to be chosen. Nonsingularity is preserved by requiring $\ell_t(\underline{x}_\Delta)$ to be nonzero, and in principal we seek a large value of $|\ell_t(\underline{x}_\Delta)|$. On the other hand, we wish to remove interpolation points that are far from $\underline{x}_1$, and the Lagrange functions of such points are relatively small in the region $\mathcal{S}$, because of their zeros at the interpolation points in $\mathcal{S}$. Therefore t is set to an integer in $[1, m]$ that has the property

$$\Omega(\|\underline{x}_t - \underline{x}_*\|)\,|\ell_t(\underline{x}_\Delta)| \;=\; \max\,\{\Omega(\|\underline{x}_i - \underline{x}_*\|)\,|\ell_i(\underline{x}_\Delta)| : \; i = 1, 2, \ldots, m\}, \qquad (2.10)$$

where Ω is a weighting function, and where $\underline{x}_*$ is the choice between $\underline{x}_1$ and $\underline{x}_\Delta$ that is the next vector $\underline{x}_1$. In UOBYQA, for example, Ω is the function $\Omega(r) = \max[1, (r/\rho)^3]$, $r \geq 0$. We do not expect the updating to improve the quadratic model, however, if $|\ell_t(\underline{x}_\Delta)| \leq 1$, $\|\underline{x}_t - \underline{x}_*\| \leq \rho$ and $\underline{x}_\Delta \neq \underline{x}_*$ hold for this choice of t, which is the unusual case when Q is not updated. Otherwise, $\underline{x}_\Delta$ replaces $\underline{x}_t$ in the interpolation equations (1.1).

The updating of Q is simple if the Lagrange functions are available. Indeed, when the interpolation equation $Q(\underline{x}_t) = F(\underline{x}_t)$ is replaced by $Q(\underline{x}_t^*) = F(\underline{x}_t^*)$, the change to Q has to be a multiple of ℓ_t, in order to preserve the other interpolation conditions. Further, the multiplying factor is defined by $Q_{\text{new}}(\underline{x}_t^*) = F(\underline{x}_t^*)$, where Q_{new} is the new model function. These remarks provide the formula

$$Q_{\text{new}}(\underline{x}) \;=\; Q_{\text{old}}(\underline{x}) \;+\; \frac{F(\underline{x}_t^*) - Q_{\text{old}}(\underline{x}_t^*)}{\ell_t(\underline{x}_t^*)}\,\ell_t(\underline{x}), \qquad \underline{x} \in \mathcal{R}^n, \qquad (2.11)$$

for generating Q_{new} from the current model function Q_{old}. Our trust region methods store the coefficients of Q_{old} and all the Lagrange functions explicitly, the total number of coefficients being about m^2. Then the coefficients of Q_{new} are obtained from formula (2.11) in only $\mathcal{O}(m)$ operations.

All the Lagrange functions are updated too, by formulae that are analogous to expression (2.11). Specifically, the functions

$$\left.\begin{array}{rcl} \ell_t^*(\underline{x}) & = & \ell_t(\underline{x}) \,/\, \ell_t(\underline{x}_t^*) \\ \ell_i^*(\underline{x}) & = & \ell_i(\underline{x}) \;-\; \ell_i(\underline{x}_t^*)\,\ell_t^*(\underline{x}), \quad i \neq t \end{array}\right\}, \qquad \underline{x} \in \mathcal{R}^n, \qquad (2.12)$$

are the new Lagrange functions, as they satisfy the Lagrange conditions of the new interpolation points. Thus the work of updating all the coefficients is $\mathcal{O}(m^2)$, which may be optimal, because, if the system (1.1) is written in matrix form, then the matrix has m^2 elements. The values $\ell_i(\underline{x}_t^*)$, $i=1,2,\ldots,m$, have been calculated already on a trust region iteration, because they are the numbers $\ell_i(\underline{x}_\Delta)$, $i=1,2,\ldots,m$, of expression (2.10). Moreover, the updating method (2.12) has a stability property that prevents damage from an accumulation of computer rounding errors over a sequence of iterations (Powell, 2001), the property being derived from the remark that the method gives the Lagrange conditions $\ell_i^*(\underline{x}_t^*)=\delta_{it}$, $i=1,2,\ldots,m$, for arbitrary functions $\ell_i \in \mathcal{M}$, provided that $\ell_t(\underline{x}_t^*)$ is nonzero. On the other hand, when $\mathcal{M}$ is the space of quadratic polynomials, the fact that m is of magnitude n^2 is very unwelcome.

3. The spaces of model functions

The COBYLA software was written by the author for constrained minimization calculations (Powell, 1994). It was applied to the test problems of Section 4, which are unconstrained, because it was available and easy to use. Some of the results of those experiments are reported and discussed later, because they are instructive. We recall that COBYLA is a trust region method of the type that is being considered, and that its space $\mathcal{M}$ of model functions is composed of all linear and constant polynomials from $\mathcal{R}^n$ to $\mathcal{R}$. Therefore the dimension m of $\mathcal{M}$ is just $n+1$, and all the procedures of Section 2 are so easy to implement that the amount of routine work of each iteration is only of magnitude n^2. At the beginning of the calculation, one of the interpolation points, $\underline{x}_1$ say, is given by the user, and ρ is set to its prescribed initial value. Then the other interpolation points of the first iteration are $\underline{x}_{j+1} = \underline{x}_1 + \rho\,\underline{e}_j$, $j=1,2,\ldots,n$, where $\underline{e}_j$ is the j-th coordinate vector in $\mathcal{R}^n$. After calculating all the function values $F(\underline{x}_i)$, $i=1,2,\ldots,m$, two of the points $\underline{x}_i$ are exchanged if necessary so that condition (2.1) holds. Hence all the nonzero coefficients of the Lagrange functions of the first iteration are found in only $\mathcal{O}(n)$ operations. These remarks show that, when COBYLA is employed for unconstrained optimization, then it is usual for most of the time of the computation to be spent on the calculation of values of the objective function.

It has been mentioned, however, that linear polynomial models are unsuitable for unconstrained calculations. The following example shows a serious deficiency of this choice of $\mathcal{M}$. Let F be the quadratic function

$$F(\underline{x}) \;=\; x_1^2 + 4\,\Gamma\, x_2^2, \qquad \underline{x} \in \mathcal{R}^2, \tag{3.1}$$

where Γ is a very large positive constant, and let the interpolation points be the vectors

$$\underline{x}_1 = \begin{pmatrix} \Gamma\rho \\ 0 \end{pmatrix}, \quad \underline{x}_2 = \begin{pmatrix} \Gamma\rho - \frac{1}{2}\rho \\ \frac{1}{2}\rho \end{pmatrix} \quad \text{and} \quad \underline{x}_3 = \begin{pmatrix} \Gamma\rho - \frac{1}{2}\rho \\ -\frac{1}{2}\rho \end{pmatrix}, \tag{3.2}$$

for the current value of ρ. The positions of these points have been chosen so that there is no need for a model iteration, and the choice of F satisfies condition (2.1), the relevant function values being $F(\underline{x}_1) = \Gamma^2\rho^2$ and $F(\underline{x}_2) = F(\underline{x}_3) = \Gamma^2\rho^2 + \frac{1}{4}\rho^2$. Thus, due to symmetry about the x_1-axis and $F(\underline{x}_2) > F(\underline{x}_1)$, the interpolation equations (1.1) define a linear polynomial Q that decreases monotonically as x_1 increases. Further, the minimization of Q within the trust region (2.2) provides $\underline{x}_\Delta = \underline{x}_1 + \Delta\underline{e}_1$, so $F(\underline{x}_\Delta)$ exceeds $F(\underline{x}_1)$ on a trust region iteration. Therefore, in the usual case $\Delta = \rho$, either termination or a reduction in ρ occurs. Termination is unwelcome, because the distance from $\underline{x}_1$ to the solution of the unconstrained problem is $\Gamma\rho$. Alternatively, if $\Delta = \rho$ persists as in COBYLA, then the number of iterations for the new value of ρ is typically of magnitude $\sigma\Gamma$, where σ is the factor by which ρ is reduced. Both of these situations are unsatisfactory when Γ is very large.

Therefore most of our attention is given to model functions that are quadratic polynomials. The use of interpolation equations of the form (1.1) for defining a quadratic model is proposed by Winfield (1973), but he does not provide a robust way of maintaining nonsingularity in the system of equations. Nonsingularity is intimately related to values of Lagrange functions, as shown in formulae (2.11) and (2.12). These relations were the reason for beginning the development of UOBYQA about ten years ago. Two major changes have been made to the original version, namely the introduction of $\Delta \geq \rho$ instead of the single trust region radius $\Delta = \rho$, and performing fewer model iterations, by adding the condition $\theta_t > \theta_*$ to $\|\underline{x}_t - \underline{x}_1\| > \beta\rho$, as stated soon after expression (2.7). The material of Section 2 is based on the techniques that are employed by the current version of UOBYQA. It follows that, because $\mathcal{M}$ contains all quadratic polynomials from $\mathcal{R}^n$ to $\mathcal{R}$, the amount of routine work of an iteration is $\mathcal{O}(n^4)$. The user has to pick an interpolation point for the first iteration, $\underline{x}_1$ say, and then the other interpolation points of the first quadratic model are generated automatically, by taking steps of magnitude ρ from $\underline{x}_1$ in the space of the variables. There are two steps along each coordinate direction, which define the components of the gradient vector $\underline{\nabla} Q(\underline{x}_1)$ and the diagonal elements of the second derivative matrix $\nabla^2 Q$. Further, each off-diagonal element $(\nabla^2 Q)_{ij}$, $i \neq j$, is obtained from a single step of the form $\pm\rho\underline{e}_i \pm \rho\underline{e}_j$, the choice of signs and other details being given in Powell (2002). That paper also presents some numerical results to demonstrate the accuracy and efficiency of UOBYQA for objective functions of up to 20 variables. A few experiments with larger values of n are included in the next section.

If the objective function has a sparse second derivative matrix, which happens in many applications, then it is advantageous to let $\mathcal{M}$ be the linear space of quadratic polynomials whose second derivatives satisfy the sparsity conditions of $\nabla^2 F$. Thus the dimension m of $\mathcal{M}$ becomes the sum of $n+1$ and the number of matrix elements that may be nonzero on the diagonal and in the lower triangular part of $\nabla^2 F$, every second derivative matrix being symmetric. Further, no sparsity conditions are allowed on the diagonal of $\nabla^2 F$ in unconstrained calculations. The techniques of Section 2 remain valid. In particular, the application of formulae (2.11) and (2.12) to update Q and all the coefficients of all the Lagrange functions still takes $\mathcal{O}(m^2)$ operations. Therefore this task may now be less expensive than the solution of the trust region subproblem at the beginning of Section 2 by the method of Moré and Sorensen (1983). The interpolation points of the first quadratic model are generated in the way that is described in the previous paragraph, except that the step from $\underline{x}_1$ of the form $\pm\rho\underline{e}_i \pm\rho\underline{e}_j$ is omitted if and only if $(\nabla^2 Q)_{ij}$ is required to be zero. A version of UOBYQA that includes this sparsity, namely UOBSQA (Unconstrained Optimization By Sparse Quadratic Approximation) is employed in some of the numerical experiments of the next section. Then the question under investigation is not the reduction in work on each iteration, but the decrease that occurs in the total number of iterations, due to the extra information about the objective function that is given by the sparsity conditions.

One other method is also considered in the experiments of Section 4, namely UOBDQA (Unconstrained Optimization By Diagonal Quadratic Approximation), which has been introduced already in Section 1. In this method, we let $\mathcal{M}$ be the $2n+1$ dimensional space of quadratic polynomials that have diagonal second derivative matrices, even if there is no sparsity in $\nabla^2 F$. All the techniques of Section 2 are applied, and the routine work of each iteration takes only $\mathcal{O}(n^2)$ operations, including the algorithm of Moré and Sorensen for solving the trust region subproblem, because $\nabla^2 Q$ is a diagonal matrix. A cause of concern, however, is that the estimate of the error of the approximation $Q(\underline{x}) \approx F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, that is employed by UOBYQA and that is inherited by UOBSQA, is no longer valid, because the method of estimation requires the error to be zero whenever F is a quadratic polynomial (Powell, 2001). We continue, however, to use the formula for the estimate as if it were true, in order to discover experimentally whether UOBDQA may be useful in practice. Therefore, if UOBDQA and the diagonal version of UOBSQA are applied to the same objective function, with the same initial vector of variables and the same initial and final values of ρ, then the same sequences of vectors of variables are calculated, including the interpolation points of the first quadratic model. Our distinction between these two methods is that, in the tables of numerical results that are given later, the name UOBSQA is reserved for the case when the sparsity structure of $\nabla^2 Q$, $Q \in \mathcal{M}$, is the same as the sparsity structure of $\nabla^2 F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$.

When these structures are different, then the performance of UOBDQA may be inefficient in ways that are similar to the deficiencies of COBYLA that are shown in the second paragraph of this section. For example, we let F be the quadratic function

$$F(\underline{x}) \;=\; (x_1+x_2)^2 + 6\,\Gamma\,(x_1-x_2)^2, \qquad \underline{x} \in \mathcal{R}^2, \tag{3.3}$$

where Γ is still a very large constant, and we let $\underline{x}_1$, $\underline{x}_2$, $\underline{x}_3$, $\underline{x}_4$ and $\underline{x}_5$ be the points

$$\begin{pmatrix} \Gamma\rho \\ \Gamma\rho \end{pmatrix}, \quad \begin{pmatrix} \Gamma\rho-\rho \\ \Gamma\rho \end{pmatrix}, \quad \begin{pmatrix} \Gamma\rho \\ \Gamma\rho-\rho \end{pmatrix}, \quad \begin{pmatrix} \Gamma\rho+\rho \\ \Gamma\rho \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \Gamma\rho \\ \Gamma\rho+\rho \end{pmatrix}, \tag{3.4}$$

respectively, for the current value of ρ. Again there is no need for a model iteration and condition (2.1) holds. Indeed, in addition to $F(\underline{x}_1) = 4\,\Gamma^2\rho^2$, we find the function values

$$\left.\begin{array}{l} F(\underline{x}_2) \;=\; F(\underline{x}_3) \;=\; 4\,\Gamma^2\rho^2 \;+\; (1+2\,\Gamma)\,\rho^2 \\ F(\underline{x}_4) \;=\; F(\underline{x}_5) \;=\; 4\,\Gamma^2\rho^2 \;+\; (1+10\,\Gamma)\,\rho^2 \end{array}\right\}. \tag{3.5}$$

Hence UOBDQA generates a quadratic model that can be written in the form

$$Q(\underline{x}_1+\underline{d}) \;=\; F(\underline{x}_1) \;+\; 4\,\Gamma\rho\,(d_1+d_2) \;+\; (1+6\,\Gamma)\,(d_1^2+d_2^2), \quad \underline{d}\in\mathcal{R}^2. \tag{3.6}$$

Thus the solution $\underline{x}_\Delta$ of the trust region subproblem is the vector of variables that minimizes $Q(\underline{x})$, $\underline{x}\in\mathcal{R}^2$, both components of $\underline{x}_\Delta-\underline{x}_1$ being $-2\,\Gamma\rho/(1+6\,\Gamma)$. It follows that condition (2.4) fails, so either termination or a reduction in ρ occurs, although the distance from $\underline{x}_1$ to the optimal vector of variables is $\sqrt{2}\,\Gamma\rho$. Therefore, unfortunately, some of the severe disadvantages of COBYLA apply also to UOBDQA.

4. Numerical results

The methods of Section 3 were applied to three unconstrained optimization problems with sparse second derivative matrices, that allow the number of variables n to be arbitrarily large. These problems are called ARWHEAD, BDQRTIC and CHROSEN, the first two being Test Problems 55 and 61 in the Appendix of Conn *et al* (1994), and the last one being taken from Toint (1978), except that some of his parameters are set to one. The name ARWHEAD indicates that the nonzero elements of $\nabla^2 F$ have an arrowhead structure, F being the function

$$F(\underline{x}) \;=\; \sum_{i=1}^{n-1} \Big\{ (x_i^2+x_n^2)^2 - 4x_i + 3 \Big\}, \qquad \underline{x}\in\mathcal{R}^n. \tag{4.1}$$

The BDQRTIC problem has the objective function

$$F(\underline{x}) \;=\; \sum_{i=1}^{n-4} \Big\{ (x_i^2+2x_{i+1}^2+3x_{i+2}^2+4x_{i+3}^2+5x_n^2)^2 - 4x_i + 3 \Big\}, \quad \underline{x}\in\mathcal{R}^n, \tag{4.2}$$

which is an extension of ARWHEAD that adds a band matrix of width seven to the previous arrowhead structure of $\nabla^2 F$. Moreover, the name CHROSEN

Test Problem	COBYLA	UOBYQA	UOBSQA	UOBDQA
ARWHEAD, $n=10$	280	219	118	105
ARWHEAD, $n=15$	522	458	170	164
ARWHEAD, $n=20$	678	837	225	260
ARWHEAD, $n=25$	900	1320	296	277
BDQRTIC, $n=10$	1106	434	350	288
BDQRTIC, $n=15$	2323	834	594	385
BDQRTIC, $n=20$	3616	1541	855	534
BDQRTIC, $n=25$	5619	2302	1016	705
CHROSEN, $n=10$	4661	454	247	3652
CHROSEN, $n=15$	6935	1064	431	4590
CHROSEN, $n=20$	8912	1897	553	5871
CHROSEN, $n=25$	10861	2565	736	5943

Table 1: Numbers of values of F in the cases when $\nabla^2 F$ is sparse

denotes Chained Rosenbrock, which does not require an explanation, F being the function

$$F(\underline{x}) \;=\; \sum_{i=2}^{n} \left\{ 4(x_{i-1}-x_i^2)^2 + (1-x_i)^2 \right\}, \qquad \underline{x}\in\mathcal{R}^n, \tag{4.3}$$

so $\nabla^2 F$ is a tridiagonal matrix. We see that all three objective functions are quartic polynomials, and that, when UOBSQA is applied, the dimension m of the space $\mathcal{M}$ is $3n$, $6n-9$ or $3n$ for the choice (4.1), (4.2) or (4.3), respectively, assuming $n\geq 5$ in BDQRTIC. For each test problem and each trust region method, the numbers of variables $n=10$, $n=15$, $n=20$ and $n=25$ were tried. In every case, we let the initial and final values of ρ be 0.5 and 10^{-6}. Further, as in the references cited above, the initial vector of variables was set to $\underline{e}$, $\underline{e}$ or $-\underline{e}$ for ARWHEAD, BDQRTIC or CHROSEN, respectively, where all the components of $\underline{e}\in\mathcal{R}^n$ are one.

The numbers of values of F that occurred in these calculations are reported in Table 1. A comparison of the UOBYQA and UOBSQA columns shows the reductions that can be achieved in the numbers of iterations by taking advantage of sparsity in $\nabla^2 F$. Indeed, it is possible that these numbers are of magnitude n^2 and n for UOBYQA and UOBSQA, respectively. No more attention will be given to UOBSQA in this section. The performance of COBYLA is better than the author expected, especially on the ARWHEAD test problem, and the number of iterations seems to be proportional to n in the CHROSEN experiments. The success of UOBDQA on both ARWHEAD and BDQRTIC is staggering and unexplained. These calculations were the first applications of UOBDQA that were tried by the author, so they gave much encouragement for further work. The poor results of this method for CHROSEN become less bad as n increases, but some damage occurs from the disadvantages that are the subject of the last paragraph of Section 3. Specifically, for every n in the CHROSEN experiments, the distance from the final vector of variables of UOBDQA to the optimal vector, namely $\underline{e}$, is in the interval $[5.2\times 10^{-5}, 7.4\times 10^{-5}]$, although the final value of ρ is 10^{-6}. The corresponding interval for COBYLA is $[9.0\times 10^{-5}, 1.1\times 10^{-4}]$. Good accuracy is obtained, however, in all the other calculations of Table 1.

We also try some objective functions $F(\underline{x})$, $\underline{x} \in \mathcal{R}^n$, that are periodic, so they have maxima and saddle points in addition to minima. Indeed, we apply COBYLA, UOBYQA and UOBDQA to several cases of the trigonometric function

$$F(\underline{x}) \;=\; \sum_{i=1}^{\nu} \Big\{ b_i - \sum_{j=1}^{n} \left(S_{ij} \sin x_j + C_{ij} \cos x_j\right) \Big\}^2, \qquad \underline{x} \in \mathcal{R}^n, \tag{4.4}$$

that is taken from Fletcher and Powell (1963). Here ν is an integer that is at least n, and the parameters S_{ij} and C_{ij}, $1 \le i \le \nu$, $1 \le j \le n$, are independent random integers from the interval $[-100, 100]$. Further, a vector $\underline{\check{x}}$ is chosen randomly from $[-\pi, \pi]^n \subset \mathcal{R}^n$, and then the parameters b_i, $1 \le i \le \nu$, are defined by the equation $F(\underline{\check{x}}) = 0$, so $\underline{\check{x}}$ is an optimal vector of variables for the unconstrained minimization of F. The required initial vector of variables is generated by making a random perturbation to every component of $\underline{\check{x}}$, each perturbation being from the distribution that is uniform on $[-0.1\pi, 0.1\pi]$. From now on, we let the initial and final values of ρ be 0.1 and 10^{-6}, respectively. Thus, apart from the random numbers, each problem and the data for the trust region methods are defined by n and ν. The effects of randomness are tested by generating five different sets of random numbers for every n and ν. Solving these five different versions of the optimization problem by any one of our methods usually provides five different values of the number of times the objective function is calculated. Just the least and greatest of these five values will be reported Tables 2, 3 and 5. The choices of random numbers were preserved, in order to apply the three different trust region methods to the same test problems. We let the number of variables be $n = 3$, $n = 5$, $n = 10$ and $n = 20$, and we compare the dependence of the minimization of the function (4.4) on two different choices of ν, namely $\nu = n$ and $\nu = 2n$.

The results of these experiments are given in Table 2. We see that the performance of UOBYQA is satisfactory for all the test problems, but that the worst results of COBYLA and UOBDQA when $\nu = n$ are remarkably bad. The reason is that a large value of Γ in example (3.1) or (3.3) is analogous to severe ill-conditioning in the second derivative matrix $\nabla^2 F(\underline{\check{x}})$. Moreover, because the terms inside the braces of expression (4.4) are zero at $\underline{x} = \underline{\check{x}}$, we find the identity $\nabla^2 F(\underline{\check{x}}) = 2Z^T Z$, where Z is the $\nu \times n$ matrix that has the elements

$$Z_{ij} \;=\; S_{ij} \cos \check{x}_j - C_{ij} \sin \check{x}_j, \qquad 1 \le i \le \nu, \quad 1 \le j \le n. \tag{4.5}$$

Test Problem	COBYLA	UOBYQA	UOBDQA
$\nu=n,\ n=3$	89–18792	34–112	105–36553
$\nu=n,\ n=5$	482–171206	60–103	501–38506
$\nu=n,\ n=10$	4521–53942	209–625	6017–97136
$\nu=n,\ n=20$	19533–67726	736–1448	21473–190196
$\nu=2n,\ n=3$	92–387	33–37	129–587
$\nu=2n,\ n=5$	165–450	51–56	233–512
$\nu=2n,\ n=10$	360–616	139–158	539–1036
$\nu=2n,\ n=20$	867–1404	416–483	1158–2019

Table 2: Numbers of values of F for the trigonometric function (4.4)

Therefore, letting $\underline{z}_i^T$ denote the i-th row of Z, the least eigenvalue of the positive definite matrix $\nabla^2 F(\check{\underline{x}})$ is the quantity

$$2 \min \Big\{ \sum_{i=1}^{\nu} (\underline{z}_i^T \underline{v})^2 : \ \underline{v} \in \mathcal{R}^n, \ \|\underline{v}\| = 1 \Big\} . \tag{4.6}$$

It follows that COBYLA and UOBDQA tend to be highly inefficient if a vector $\underline{v} \in \mathcal{R}^n$ is nearly orthogonal to all the rows of Z. This can happen easily due to the random numbers in the case $\nu = n$, but, in the alternative case $\nu = 2n$, the probability that $\nabla^2 F(\check{\underline{x}})$ is well-conditioned is high. Thus the entries in the second half of Table 2 depend less strongly on the random numbers. Further, if the most difficult of the five test problems with $\nu = n = 3$ is deleted, then the COBYLA and UOBDQA results in the first row of Table 2 become 89–3778 and 105–1207, respectively. Similarly, if the third of the five problems with $\nu = n = 5$ is deleted, then the COBYLA and UOBDQA entries in the second row of the table become 482–1135 and 501–1400. It seems, however, that UOBYQA is good at coping with ill-conditioning when the number of variables is small.

Table 2 shows that COBYLA is more efficient than UOBDQA at minimizing the function (4.4) with $\nu = 2n$. This conclusion was unexpected, because COBYLA employs linear model functions, while UOBDQA pays some attention to curvature. On the other hand, linear models provide suitable search directions for unconstrained minimization if $\nabla^2 F$ is close to a multiple of the $n \times n$ unit matrix, and there is a tendency for the matrix $2Z^T Z$ of the previous paragraph to have this property if ν is increased for fixed n. Therefore we consider some other test problems where the general diagonal quadratic model of UOBDQA is particularly useful. Specifically, we let F_{old} be the function (4.4) with $\nu = 2n$, we let D be an $n \times n$ diagonal matrix whose diagonal elements are random numbers from the distribution that is logarithmic on $[1, 10]$, and we employ the objective

Test Problem	COBYLA	UOBYQA	UOBDQA
$\nu=2n,\ n=3$	500–5329	34–63	155–818
$\nu=2n,\ n=5$	991–4686	74–101	258–944
$\nu=2n,\ n=10$	4069–10948	198–249	708–1613
$\nu=2n,\ n=20$	8101–16662	596–665	1526–2762

Table 3: Numbers of values of F for the scaled trigonometric function

function

$$F_{\rm new}(\underline{x}) \;=\; F_{\rm old}(D^{-1}\underline{x}), \qquad \underline{x}\in\mathcal{R}^n. \tag{4.7}$$

The only other change to the $\nu=2n$ calculations of Table 2 is that, having chosen an initial vector of variables, $\underline{\widehat{x}}$ say, in the way that is described soon after equation (4.4), we let the initial vector of variables for the new calculation be $D\underline{\widehat{x}}$, which allows for the scaling that has been introduced. The initial and final values of ρ remain at 0.1 and 10^{-6}, however, because of any diagonal elements of D that are close to one. Table 3 presents the results of these new experiments that correspond to the $\nu=2n$ entries of Table 2. We see that UOBYQA and UOBDQA require a few more iterations than before, but that the efficiency of COBYLA has been destroyed by the introduction of some mild diagonal scaling. Indeed, these results restore the belief of the author that COBYLA is unsuitable for unconstrained minimization calculations.

We recall that the main advantage of UOBDQA over UOBYQA is that the amount of routine work of each iteration is only $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^4)$. Thus UOBDQA is faster than UOBYQA at solving the $n=20$ problems of Table 3, although it requires about three times as many values of F. Details are given in Table 4, which compares the computation times and numbers of function evaluations of these two trust region methods, using the objective function and initial data that are described in the previous paragraph. The results are not very sensitive to the random numbers that occur. Therefore, for each value of n, the entries in Table 4 are averages for the five test problems that are generated by different choices of the random numbers. The given times were measured in seconds, by the Fortran DTIME instruction, from the call of UOBYQA or UOBDQA until the return from the subroutine. All the calculations were run on a Sun Ultra 10 workstation, which allows the inclusion of problems with 40 variables, but the amount of work of UOBYQA for $n=80$ would be prohibitive. We see in the table that UOBYQA becomes impractical as n increases, but that UOBDQA may be useful for the larger values of n. On the other hand, we have found already that UOBDQA can be very inefficient, as in the $\nu=n$ calculations of Table 2 and in the example of the last paragraph of Section 3. Therefore another trust region method is considered briefly in the next section. Sometimes it requires fewer func-

Test Problem	UOBYQA		UOBDQA	
	Seconds	$\#F$	Seconds	$\#F$
$\nu=2n,\ n=3$	0.008	47	0.032	313
$\nu=2n,\ n=5$	0.042	83	0.119	523
$\nu=2n,\ n=10$	0.774	225	0.823	1067
$\nu=2n,\ n=20$	19.91	637	5.96	2040
$\nu=2n,\ n=40$	1086.6	2119	78.2	6831

Table 4: Averages of timings and $\#F$ for the scaled trigonometric function

tion values than UOBYQA, which is important when the amount of computation is dominated by evaluations of the objective function.

5. Least Frobenius norm updating

The main disadvantages of UOBYQA and UOBDQA are that the routine work of each iteration of UOBYQA takes of magnitude n^4 operations, while the efficiency of UOBDQA is often impaired by restricting the model functions to quadratic polynomials with diagonal second derivative matrices. The amount of work of UOBYQA occurs because each quadratic model depends on $\frac{1}{2}(n+1)(n+2)$ values of F, so now we prefer to employ the interpolation conditions

$$Q(\underline{x}_i) \;=\; F(\underline{x}_i), \qquad i=1,2,\ldots,\widehat{m}, \tag{5.1}$$

where $\widehat{m}$ is a given integer that is only of magnitude n. On the other hand, we also prefer to include all polynomials of degree at most two from $\mathcal{R}^n$ to $\mathcal{R}$ in the space $\mathcal{M}$, as in UOBYQA, so the dimension of $\mathcal{M}$ is $m=\frac{1}{2}(n+1)(n+2)$.

Ways of constructing quadratic models from fewer than m conditions are usual in algorithms for unconstrained optimization when first derivatives of F are available. Then a typical iteration changes the best vector of variables so far from $\underline{x}_k$ to $\underline{x}_{k+1}$, say, where k is the iteration number for the moment. The gradient vectors $\underline{\nabla} F(\underline{x}_k)$ and $\underline{\nabla} F(\underline{x}_{k+1})$ are calculated. Further, letting Q_{old} and Q_{new} be the quadratic models at the beginning and end of the iteration, Q_{new} is given the curvature information that is provided by the change in gradients. The procedure that takes up the remaining freedom in Q_{new} can often be expressed as the minimization of some measure of the difference between Q_{new} and Q_{old}. In particular, the symmetric Broyden formula (see Fletcher, 1987, for instance) generates the second derivative matrix $\nabla^2 Q_{\text{new}}$ by minimizing the Frobenius norm $\|\nabla^2 Q_{\text{new}} - \nabla^2 Q_{\text{old}}\|_F$, subject to symmetry and the quasi-Newton equation

$$(\nabla^2 Q_{\text{new}})\,(\underline{x}_{k+1} - \underline{x}_k) \;=\; \underline{\nabla} F(\underline{x}_{k+1}) - \underline{\nabla} F(\underline{x}_k). \tag{5.2}$$

In other words, the sum of squares

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left\{ (\nabla^2 Q_{\rm new})_{ij} - (\nabla^2 Q_{\rm old})_{ij} \right\}^2 \tag{5.3}$$

is made as small as possible, subject to the constraints that have been mentioned. We are going to consider briefly whether a version of this updating method may be useful for unconstrained minimization without derivatives.

There are usually $m - \widehat{m}$ independent degrees of freedom in the solution of the equations (5.1) by a quadratic polynomial $Q \in \mathcal{M}$. Let $Q_{\rm new}$ be the model function that is being generated, and let $Q_{\rm old}$ be the model function at the beginning of the current iteration. As in the previous paragraph, we take up the freedom in $Q_{\rm new}$ by minimizing the sum of squares (5.3). Further, we ensure that $Q_{\rm new}$ is well defined by imposing the following two conditions on the positions of the interpolation points $\underline{x}_i$, $i = 1, 2, \ldots, \widehat{m}$. Firstly, we require the $(n+1) \times \widehat{m}$ matrix

$$X = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \underline{x}_1 & \underline{x}_2 & \cdots & \underline{x}_{\widehat{m}} \end{pmatrix} \tag{5.4}$$

to have rank $n+1$, because otherwise a nonzero polynomial, p say, of degree at most one, would satisfy $p(\underline{x}_i) = 0$, $i = 1, 2, \ldots, \widehat{m}$. Then $Q_{\rm new}$ could be changed by the addition of any multiple of p, which would preserve the interpolation equations without altering $\nabla^2 Q_{\rm new}$. Secondly, letting b_j, $j = 1, 2, \ldots, m$, be a basis of $\mathcal{M}$, we require the $\widehat{m} \times m$ matrix B with the elements

$$B_{ij} = b_j(\underline{x}_i), \qquad 1 \le i \le \widehat{m}, \quad 1 \le j \le m, \tag{5.5}$$

to have rank $\widehat{m}$. Thus the equations (5.1) have a solution for any right hand sides. It follows from these conditions that $\widehat{m}$ is in the interval $[n+1, m]$, and a technique for satisfying them is given later. The freedom in the initial quadratic interpolant Q is removed by minimizing $\|\nabla^2 Q\|_F$.

It would be inconvenient to include first derivatives in the expression that is minimized to take up the freedom in $Q_{\rm new}$, partly because the value of $\underline{\nabla} Q_{\rm new}(\underline{x})$ requires a particular vector $\underline{x} \in \mathcal{R}^n$ to be chosen. Moreover, the proposed use of Frobenius norms provides a very welcome projection property when F is a quadratic polynomial. Specifically, in this case, the updating of the quadratic model gives the inequality

$$\begin{aligned} \|\nabla^2 F - \nabla^2 Q_{\rm new}\|_F^2 &= \|\nabla^2 F - \nabla^2 Q_{\rm old}\|_F^2 - \|\nabla^2 Q_{\rm new} - \nabla^2 Q_{\rm old}\|_F^2 \\ &\le \|\nabla^2 F - \nabla^2 Q_{\rm old}\|_F^2, \end{aligned} \tag{5.6}$$

which is proved by the following elementary argument. The difference $F - Q_{\rm new}$ is a quadratic polynomial that vanishes at the points $\underline{x}_i$, $i = 1, 2, \ldots, \widehat{m}$. Therefore the construction of $Q_{\rm new}$ implies that the least value of the function

$$\|(\nabla^2 Q_{\rm new} - \nabla^2 Q_{\rm old}) + \theta\,(\nabla^2 F - \nabla^2 Q_{\rm new})\|_F^2, \qquad \theta \in \mathcal{R}, \tag{5.7}$$

occurs when θ is zero, which is the condition

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (\nabla^2 Q_{\rm new} - \nabla^2 Q_{\rm old})_{ij} \, (\nabla^2 F - \nabla^2 Q_{\rm new})_{ij} \;=\; 0. \tag{5.8}$$

Hence the first line of expression (5.6) is true, and the second line is obvious. Thus, in the quadratic case $F \in \mathcal{M}$, we can take the view that the errors of the approximations $Q \approx F$ decrease monotonically as the iterations proceed, except for the effects of computer rounding errors.

The calculation of $Q_{\rm new}$ in the way that has been suggested is a quadratic programming problem with only equality constraints. Therefore it can be expressed as the solution of a linear system of equations. Further, the KKT conditions of this problem imply that the second derivative matrix of $Q_{\rm new}$ has the form

$$\nabla^2 Q_{\rm new} \;=\; \nabla^2 Q_{\rm old} + \sum_{j=1}^{\widehat{m}} \lambda_j \, \underline{x}_j \underline{x}_j^T, \tag{5.9}$$

where the Lagrange multipliers λ_j, $j=1,2,\ldots,\widehat{m}$, satisfy $X\underline{\lambda}=0$, X being the matrix (5.4). Therefore, if we write $Q_{\rm new}$ as the quadratic polynomial

$$Q_{\rm new}(\underline{x}) \;=\; g_0 + \underline{g}^T \underline{x} + \tfrac{1}{2} \, \underline{x}^T (\nabla^2 Q_{\rm new}) \, \underline{x}, \qquad \underline{x} \in \mathcal{R}^n, \tag{5.10}$$

then, because of the interpolation conditions (5.1) and $X\underline{\lambda}=0$, the coefficients $\underline{\lambda} \in \mathcal{R}^{\widehat{m}}$, $g_0 \in \mathcal{R}$ and $\underline{g} \in \mathcal{R}^n$ are defined by the $(\widehat{m}+n+1) \times (\widehat{m}+n+1)$ system of equations

$$\left(\begin{array}{c|c} A & X^T \\ \hline X & 0 \end{array} \right) \left(\begin{array}{c} \underline{\lambda} \\ \hline \underline{\widehat{g}} \end{array} \right) \;=\; \left(\begin{array}{c} \underline{F} \\ \hline 0 \end{array} \right), \tag{5.11}$$

where A is the $\widehat{m} \times \widehat{m}$ matrix that has the elements

$$A_{ij} = \tfrac{1}{2} \, (\underline{x}_i^T \underline{x}_j)^2, \qquad 1 \le i,j \le \widehat{m}, \tag{5.12}$$

where $\underline{\widehat{g}}$ is the vector in $\mathcal{R}^{n+1}$ whose components are g_0 followed by the components of $\underline{g}$, and where the components of $\underline{F}$ are the known function values $F(\underline{x}_i)$, $i = 1,2,\ldots,\widehat{m}$. Thus the amount of work to calculate $Q_{\rm new}$ by a direct method is of magnitude $(\widehat{m}+n)^3$, which is a major improvement over UOBYQA for sufficiently large n, due to our restriction $\widehat{m} = \mathcal{O}(n)$. Further, the author expects to develop an updating technique for generating the sequence of quadratic models that is faster than the use of direct methods. Another improvement over UOBYQA is that the method of this section requires less computer storage. Moreover, equation (5.12) implies that A is not only symmetric but also has no negative eigenvalues.

The author started his research on the material of this section in January, 2002, so it is not complete. The reason for jumping the gun in the publication of results is that it is easy to include the least Frobenius norm updating method in a

	Test Problem		
	Table 2 ($\nu = n$)	Table 2 ($\nu = 2n$)	Table 3 ($\nu = 2n$)
$n=3$	37–141	32–60	46–66
$n=5$	95–232	56–87	114–186
$n=10$	303–1067	156–230	326–395
$n=20$	1370–3316	428–474	785–941

Table 5: Numbers of values of F for the new version of UOBDQA

version of UOBDQA. Specifically, we retain the number of interpolation equations $\widehat{m} = 2n+1$, but, instead of applying formula (2.11), the new algorithm generates Q_{new} by the method of the previous paragraph. Hence $\nabla^2 Q_{\text{new}}$ is usually a full matrix, although the choice of the initial interpolation points by UOBDQA causes $\nabla^2 Q$ to be diagonal at the beginning of the first iteration. Therefore no sparsity is assumed when solving the trust region subproblem of Section 2 by the method of Moré and Sorensen (1983). All other features of UOBDQA are present in the new algorithm, however, including the use of Lagrange functions with diagonal second derivative matrices to control the changes that are made to the positions of the interpolation points. Further, these Lagrange functions are still updated by formula (2.12). Thus the ranks of the matrices X and B are always $n+1$ and $\widehat{m}$, respectively, as required. An objection to this way of maintaining nonsingularity of the system (5.11), however, is that it is not invariant under orthogonal rotations of the space of the variables. Therefore the new version of UOBDQA is intended only for some preliminary investigations of least Frobenius norm updating.

The new algorithm was applied to all the test problems of Section 4. It may be misleading to draw conclusions from the objective functions of Table 1, because of the effects of sparsity. We note, however, that the numbers of calculations of F by the new algorithm when $n=20$, for example, are 341, 2779 and 825 for ARWHEAD, BDQRTIC and CHROSEN, respectively, and that comparisons with the entries in Table 1 for other values of n are similar to comparisons in the case $n = 20$. Moreover, the results for the objective functions of Tables 2 and 3 are given in Table 5. We welcome the fact that the change to UOBDQA corrects the severe inefficiencies for $\nu = n$ that are shown in the last column of Table 2, which suggests that the least Frobenius norm updating method is useful when $\nabla^2 F$ is ill conditioned. The other entries in Table 5 also compare favourably with the corresponding results of UOBDQA in Tables 2 and 3. Experiments with the new algorithm are also promising for larger numbers of variables. Indeed, the change to the updating procedure of UOBDQA decreases the $n=40$ value of $\#F$ in Table 4 from 6831 to 2179. Further, in similar tests with $n=80$ and $n=160$, the value of $\#F$ is reduced from 15147 to 4623 and from 34854 to 9688, respectively. Only two sets of random numbers were tried for the new algorithm when $n=160$, however, because the time of each experiment was about 25 hours. Nevertheless, good accuracy is achieved in both cases, the ratio of the initial to the final calculated value of F being of magnitude 10^{14}.

These results for $n=160$ are reminiscent of an important property of gradient methods for unconstrained optimization that was discovered about 30 years ago. It is that the sequence of calculated vectors of variables can converge at a superlinear rate to a solution of the optimization problem, $\underline{x}_*$ say, without $\nabla^2 Q$ converging to $\nabla^2 F(\underline{x}_*)$, where Q is still the current quadratic model. Further, fewer than n iterations often provide enough accuracy in practice, although n iterations are usually necessary if $\nabla^2 Q$ is required to be a good approximation to $\nabla^2 F(\underline{x}_*)$. In the present situation without derivatives, we recall that a quadratic model has $\frac{1}{2}(n+1)(n+2)$ independent parameters. This number is 13122 when $n=160$, but we have found in this case that the new version of UOBDQA employs fewer than 10000 values of F, when it is applied to the scaled trigonometric function of Tables 3 and 4. In other words, assuming that the amount of work is proportional to the number of function values, the new method solves the minimization problem while UOBYQA is constructing the quadratic model for the first iteration. It follows that trying to achieve good accuracy in all the parameters of the model may be inefficient. Therefore we expect the updating technique of this section to be highly useful to the development of new algorithms for unconstrained optimization calculations.

Acknowledgements

The author is very grateful to the Mathematics Department of the City University of Hong Kong for hospitality and facilities that supported the research that is reported in Section 5. Moreover, Benoit Colson provided helpful advice on the sparse test problems of Table 1, and Beresford Parlett kindly confirmed that the matrix A with the elements (5.12) has no negative eigenvalues.

References

A.R. Conn, N.J.M. Gould, M. Lescrenier and Ph.L. Toint (1994), "Performance of a multifrontal scheme for partially separable optimization", in *Advances in Optimization and Numerical Analysis*, eds. S. Gomez & J-P. Hennart, Kluwer Academic (Dordrecht), pp. 79–96.

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

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

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.

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 & J-P. Hennart, Kluwer Academic (Dordrecht), pp. 51–67.

M.J.D. Powell (2001), “On the Lagrange functions of quadratic models that are defined by interpolation”, *Optim. Meth. Software*, Vol. 16, pp. 289–309.

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

Ph.L. Toint (1978), “Some numerical results using a sparse matrix updating formula in unconstrained optimization”, *Math. Comp.*, Vol. 32, pp. 839–851.

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