An algorithm using quadratic interpolation for unconstrained derivative free optimization

by A.R. Conn[1] and Ph.L. Toint[2]

Report 95/6 September 12, 1995

[1] IBM T.J. Watson Research Center, P.O.Box 218, Yorktown Heights, NY 10598, USA
Email : arconn@watson.ibm.com

[2] Department of Mathematics, Facultés Universitaires ND de la Paix, 61, rue de Bruxelles, B-5000 Namur, Belgium, EU
Email : pht@math.fundp.ac.be
Current reports available by anonymous ftp from the directory "pub/reports" on thales.math.fundp.ac.be (internet 138.48.4.14)

To appear in "Nonlinear Optimization and Applications",
G. Di Pillo and F. Giannessi, editors, Plenum Publishing, November 1995.

**Keywords:** nonlinear optimization, derivative free algorithms, noisy functions.

# An algorithm using quadratic interpolation for unconstrained derivative free optimization

A. R. Conn*  Ph. L. Toint

September 12, 1995

**Abstract**

This paper explores the use of multivariate interpolation techniques in the context of methods for unconstrained optimization that do not require derivative of the objective function. A new algorithm is proposed that uses quadratic models in a trust region framework. The algorithm is constructed to require few evaluations of the objective function and is designed to be relatively insensitive to noise in the objective function values. Its performance is analyzed on a set of 20 examples, both with and without noise.

## 1 Introduction

We are concerned, in this paper, with the problem of minimizing an objective function whose value is determined by measuring some quantity in the real world. This measure may be of a physical nature (for instance, the depth of a certain layer in geophysical exploration) or be related to other contexts. We will focus on the case where there are no constraints on the problem variables. The generalization to simple bounds, in particular, is quite straightforward, as indicated at the end of the paper. Moreover, since the proposed method is derivative free, one might want to handle constrained problems using an exact penalty function.

Three important features characterize these types of problems. Firstly, the cost of obtaining a function value, that is of performing the measure for particular values of the problem variables, is typically very high. This calls for optimization techniques that make optimal use of all such evaluations, possibly at the expense of more extensive linear algebra calculations within the algorithm itself. The second important feature is that the nature of the function evaluation or some other reasons, prevents the computation of any associated derivatives (gradient or Hessian), a serious drawback for many optimization methods. Finally, the considered measure is usually subject to error itself, introducing some "noise" on the objective evaluation, which puts additional requirements on the minimization's robustness.

*This research of this author was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No F49620-91-C-0079. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Note that these problem features may make the calculation of derivatives by finite differences unattractive. Indeed, the additional function evaluations required in the differencing, by this technique, may be very costly and, most importantly, finite differencing can be unreliable in the presence of noise if no specific action is taken to adapt the differentiation step size to the noise level. Since automatic differentiation (see Griewank (1989), for example) is not applicable to a "physical" measurement procedure, we thus may be forced to consider algorithms that do not proceed to approximate objective function derivatives for a given value of the problem variables.

By extension, we will also consider in this paper unconstrained optimization problems whose objective function is the result of a complex and costly numerical procedure (such as, for example, in the analysis of the in flight vibration of a helicopter rotor), possibly involving some considerable noise (due, for instance, to truncation or approximation in the calculation defining the objective). At variance with the framework described above, automatic differentiation may often be applied to such cases, but the computed derivatives then include differentiation of the noise itself, making the calculated gradients of questionable value to measure the local slope. Furthermore, automatic differentiation is not applicable when the source code for evaluating the objective function is unavailable. Finally, it may not always be straightforward to use, as is for example the case in fluid dynamics calculations where, according to Burns (1995), it may generate unwanted dependence on discretization parameters or on the introduction of artificial viscosity.

Derivative free optimization methods have a long history and we refer the reader to Dixon (1972), Himmelblau (1972) or Polyak (1987) for extensive discussion and references. These methods come in essentially five different classes. The first class contains the algorithms which use finite-difference approximations of the objective function's derivatives in the context of a gradient based method, such as nonlinear conjugate gradients or quasi-Newton methods (see, for instance, Stewart (1967), Dennis and Schnabel (1983), Gill *et al.* (1981) and Gill *et al.* (1983)). The methods in the second class are often referred to as "pattern search" methods, because they are based on the exploration of the variables' space using a well specified geometric pattern, typically a simplex. They were investigated by Spendley *et al.* (1962), Hooke and Jeeves (1961) and Nelder and Mead (1965), the algorithm proposed by the latter still being one of the most popular minimization technique in use today. More recent developments of pattern search methods include proposals by Torczon (1991) and Dennis and Torczon (1991). The approaches of the third type are based instead on random sampling and were developed by Box (1966), Brooks (1958) and Kelly and Wheeling (1962), to cite a few. The methods of the fourth class are based, as for many methods using derivatives, on the use of successive one-dimensional minimizations (line searches) along selected directions. These directions may be chosen amongst the set of coordinate basis vectors, as in Elkin (1968), Ortega and Rheinboldt (1970) or Lucidi and Sciandrone (1995), with possible reorientation of the basis as described in Rosenbrock (1960) and Swann (1964), or on sets of mutually conjugate directions, as proposed by Powell (1964) and later developed by Brent (1973). Finally, the algorithms of the fifth class are based on the progressive building and updating of a model of the objective function, as proposed by Powell (1994a) for linear models and by Powell (1994b) for quadratic ones. There is also a related class of "global modelling" methods, that uses Design of Experiments (DOE) interpolation models. For instance, in a problem with ten variables one may determine 50 suitably chosen function values (perhaps by using optimal designs, see for example Owen (1992)) for determining an initial model that satisfies a maximum likelihood estimator (MLE) criteria and one may use an additional 50 evaluations to refine the model. Details are given in Booker (1994), Mitchell *et al.* (1989), Morris *et al.* (1991) and Sacks *et al.* (1992).

The approach developed below belongs to the "model building and updating" class. Following ideas expressed in Powell (1994b), we will consider a trust region framework where the objective function's model is built by multivariate (quadratic) interpolation. At variance with Powell's proposal, we will however insist on the ability of our algorithm to take long steps and also to progress as early as possible with every available function evaluation.

The purpose of this paper is to present the current state of the authors' ideas in what is likely to be a longer term project. It is organized as follows: Section 2 introduces the problem, notation and the algorithm, while the results of preliminary numerical experience are discussed in Section 3. Some conclusions and perspectives are outlined in Section 4.

## 2 Algorithmic concepts

We consider the problem of finding a vector $x \in \mathbf{R}^n$, a solution of the unconstrained nonlinear program

$$\min_{x \in \mathbf{R}^n} f(x), \tag{2.1}$$

where $f(\cdot)$ is a twice differentiable function from $\mathbf{R}^n$ into $\mathbf{R}$. Although the derivatives of $f(\cdot)$ may exist, we assume that they cannot be calculated. We also assume that, although $f(x)$ can be evaluated at any $x$, the cost of such an evaluation is high compared to that of solving square dense systems of linear equations in $n$ variables. The notation $\langle x, y \rangle$ will be used throughout to denote the Euclidean inner product of $x$ and $y$ and $\|x\|$ will denote the Euclidean norm of $x$.

The algorithm proposed in this note belongs to the class of "trust-region" methods. Such algorithms are iterative and build, around the current iterate, a model of the true objective function which is cheaper to evaluate and easier to minimize than the objective function itself. This model is assumed to represent the objective function well in a so-called trust region, typically a ball centered at the current iterate, $x_c$ say. The radius of this ball, traditionally denoted by $\Delta$, is called the trust region radius and indicates how far the model is thought to represent the objective function well. A new trial point is then computed, which minimizes or sufficiently reduces the model *within the trust region* and the true objective function is evaluated at this point. If the achieved objective function reduction is sufficient compared to the reduction predicted by the model, the trial point is accepted as the new iterate and the trust region possibly enlarged. On the other hand, if the achieved reduction is poor compared to the predicted one, the current iterate is unchanged and the trust region is reduced. This process is then repeated until convergence (hopefully) occurs.

## 2.1 The quadratic model and how to improve it

One of the main ingredients of a trust region algorithm is thus the choice of an adequate objective function model. We will here follow a well established tradition in choosing a quadratic model of the form

$$m(x_c + s) = f(x_c) + \langle g, s\rangle + \frac{1}{2}\langle s, Hs\rangle, \tag{2.2}$$

where $g$ is a vector of $\mathbf{R}^n$ and where $H$ is a square symmetric matrix of dimension $n$. However, we will depart from many trust-region algorithms in that $g$ and $H$ will not be determined by the (possibly approximate) first and second derivatives of $f(\cdot)$, but rather by imposing that the model (2.2) interpolates function values at past points, that is we will impose that

$$m(x) = f(x) \tag{2.3}$$

for each vector $x$ in a set $I$ such that $f(x)$ is known for all $x \in I$. Note that this interpolation technique is also used by Powell (1994a) and Powell (1994b). Note also that the cardinality of $I$ must be equal to

$$p = \frac{1}{2}(n+1)(n+2) \tag{2.4}$$

to ensure that the quadratic model is entirely determined by the equations (2.3). However, if $n > 1$, this last condition is not sufficient to guarantee the existence of an interpolant. It is indeed well-known (see De Boor and Ron (1992) or Sauer and Xu (1995), for instance) that the points in $I$ must also satisfy some geometric constraints: for instance, six points on a line do not determine a two dimensional quadratic. When the geometry of the points in $I$ is such that the interpolant exists, we follow Sauer and Xu (1995) and say that $I$ is *poised.* If we choose a basis $\{\phi_i(\cdot)\}_{i=1}^p$ of the linear space of $n$-dimensional quadratics, $I = \{x_1, \ldots, x_p\}$ is poised when

$$\delta(I) = \det \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_p) \\ \vdots & & \vdots \\ \phi_p(x_1) & \cdots & \phi_p(x_p) \end{pmatrix} \tag{2.5}$$

is non-zero. Of course, the quality of the model (2.2) as an approximation of the objective function around $x_c$ will be dependent on the geometry of the considered interpolation points, and thus on the value of $|\delta(I)|$. Following Powell (1994b), we will say that this geometry, and hence the model, is *good* (with respect to $x_c$ and the radius $\Delta$) when all the points in $I$ are no further away from $x_c$ than $2\Delta$ and when the value of $|\delta(I)|$ cannot be doubled by adjusting one of the points of $I$ to an alternative value within distance $\Delta$ from $x_c$.

In derivative based trust-region methods, the radius $\Delta$ is decreased whenever sufficient decrease in the objective function is not obtained at the computed trial point (the iteration is then said to be *unsuccessful*). This technique aims at improving the model within the trust region, since Taylor's theorem indicates that the derivative based model (2.2) better fits $f(\cdot)$ in a smaller neighborhood of $x_c$. However, this improvement is not an immediate consequence of reducing $\Delta$ in our case, since our model is based on interpolating function values rather than derivatives. In order to ensure progress of the algorithm away from stationary points, we therefore have to explicitly improve the interpolation model at unsuccessful iterations, either by discarding interpolation points that are too far away from $x_c$ or by improving the geometry of $I$. Note that both these actions imply that $I$ is modified, which usually means that a new point, $x_+$, such that $\|x_+ - x_c\| \leq \Delta$, and its associated objective function value $f(x_+)$ must be computed. If we wish to make the geometry of $I$ as good as possible, we therefore need a measure of the improvement obtained by replacing some past point $x_- \in I$ by $x_+$. We consider two cases. First, if $\|x_- - x_c\| \leq \Delta$, a suitable measure is $|\delta(I)|$, and we therefore wish to compute the factor by which $|\delta(I)|$ is multiplied when $x_-$ is replaced by $x_+$. Remarkably, this factor is independent of the basis $\{\phi_i\}$ and is equal to $|L(x_+, x_-)|$, where $L(\cdot, x_-)$ is the Lagrange interpolation function whose value is one at $x_-$ and zero at all other points of $I$. This very nice result was pointed out by Powell (1994b). Hence, if $\|x_- - x_c\| \leq \Delta$, it makes sense to replace $x_-$ by

$$x_+ = \arg \max_{\|x - x_c\| \leq \Delta} |L(x, x_-)|. \tag{2.6}$$

On the other hand, if $\|x_- - x_c\| > \Delta$, it is important to take this inequality into account when choosing a suitable replacement $x_+$. One possible method, again inspired by Powell (1994b), is to compare $x_+$ not with $x_-$ directly, but rather with the best point on the segment joining $x_c$ to $x_-$ limited to the ball of radius $\Delta$ around $x_c$. This "scaled down" version of $x_-$ is the vector that maximizes $|L(x_c + td_-, x_i)|$ for $t \in [0, \Delta]$, where $d_- = (x_- - x_c)/\|x_- - x_c\|$. Hence, $x_+$ may be chosen in this case as

$$x_+ = \arg \max_{\|x - x_c\| \leq \Delta} S(x, x_-), \tag{2.7}$$

where

$$S(x, x_-) = \frac{|L(x, x_-)|}{\min[1, \max_{t \in [0,\Delta]} |L(x_c + td_-, x_-)|]}. \tag{2.8}$$

The minimum in the denominator of (2.8) guarantees that the scaled down version of $x_-$, namely $\arg\max_{t \in [0,\Delta]} |L(x_c + td_-, x_-)|$, is treated exactly as any other point within distance $\Delta$ from $x_c$ (that is according to (2.6)). This feature of $S(\cdot, \cdot)$ and the definition of the Lagrange interpolation function imply that

$$|L(x, x_-)| = S(x, x_-) \ \text{ whenever } \ \|x_- - x_c\| \leq \Delta, \tag{2.9}$$

and $S(\cdot, \cdot)$ may thus be used instead of $|L(\cdot, \cdot)|$ in (2.6), making the distinction on $\|x_- - x_c\|$ unnecessary. Note that the Lagrange interpolation function $L(\cdot, \cdot)$ is also a quadratic determined by function value interpolation, and therefore only exists, together with $S(\cdot, \cdot)$, if $I$ is poised.

A special situation occurs in this respect in the first iterations of the algorithm. Due to the assumed high cost of a function evaluation, we may wish to define a model of the type (2.2) as soon as a few objective values have been computed. This means that, although we have a set of points poised for quadratic interpolation, the objective function value may not be known for each of them. As above, we denote by $I$ the set of points for which the objective value is known, and we denote by $J$ the set of remaining points, where the objective value is still unknown. Thus $I$ may contain less than $p$ points, although the set $I \cup J$ is poised for quadratic interpolation. The Lagrange interpolation function $L(\cdot, \cdot)$ and $S(\cdot, \cdot)$ are thus well defined, but the model (2.2) is no longer fully specified by the interpolation conditions (2.3). Two solutions are then possible to determine a suitable model.

- The first is to take out the remaining degrees of freedom in the model by imposing a variational criterion. In our proposal, we have considered computing the model of minimal Frobenius norm, that is the model for which $\|g\|^2 + \|H\|_F^2$ is minimal, which still satisfies the interpolation conditions (2.3).
- The second solution is to build an sub-quadratic model, that is a model in which not all degrees of freedom of a full quadratic model are exploited but which does interpolate all the data points available. Such models can be obtained by the multivariate interpolation algorithms proposed by Sauer and Xu (1995) and Sauer (1995), for example. The actual form of the model is determined, in this approach, by the number and geometry of the available interpolation points and by the polynomial basis used to span the linear space of multivariate quadratics. We refer the reader to Sauer and Xu (1995) for further details.

Therefore computing a suitable model when $|I| < p$ is possible. Of course, if such a model needs to be improved, we will bias our procedure to reduce the cardinality of $J$, therefore enriching the information on which the model is based. This also means that, for some iterations, the Lagrangian interpolation function $L(\cdot, \cdot)$, and thus the function $S(\cdot, \cdot)$, now depend on the set of points in $I \cup J$ rather than $I$ alone.

We finally note that, when the model must be improved after an unsuccessful iteration, it may happen that points where the objective function has been previously computed are no further away then $\Delta$ from $x_c$ and yet do not belong anymore to $I$. It is then sometimes possible to improve the geometry of the current set $I \cup J$ by replacing one point in $I \cup J$ by such a previous point, and this procedure can be repeated as long as improvement of the model's geometry is obtained.

We are now ready to describe the algorithm we propose to use to improve the quadratic model after an unsuccessful iteration. In this description and later in the paper, we denote by

$$M = \{x \in \mathbf{R}^n \mid f(x) \text{ is known } \} \tag{2.10}$$

the set of points where the objective function has been evaluated so far.

**A1: Geometry improvement**

The sets $I$, $J$ and $M$, the radius $\Delta$ and the current point $x_c$ are given.

**Step 1: attempt to reuse past points that are close to $x_c$.**

For each point $x_i \in M \setminus I$ such that $\|x_i - x_c\| \le \Delta$,

- determine which of the current interpolation points in $I \cup J$ can be exchanged with $x_i$ to maximally improve the interpolation set geometry, that is compute

$$x_- = \arg \max_{x_j \in (I \cup J)} S(x_i, x_j); \tag{2.11}$$

- perform the exchange if the improvement is sufficient. That is redefine

$$I = (I \setminus \{x_-\}) \cup \{x_i\} \ \text{ if } \ S(x_i, x_-) \geq 2. \tag{2.12}$$

If at least one exchange (2.12) has been performed, successfully terminate Algorithm A1.

**Step 2: attempt to replace a point of $J$ distant from $x_c$.**
Determine the point in $J$ that is furthest from $x_c$, that is

$$x_- = \arg\max_{x_i \in J} \|x_i - x_c\|. \tag{2.13}$$

If $\|x_- - x_c\| > 2\Delta$, find a better point closer to $x_c$ to replace $x_-$, that is compute

$$x_+ = \arg\max_{\|x - x_c\| \leq \Delta} S(x, x_-), \tag{2.14}$$

calculate $f(x_+)$, set $M = M \cup \{x_+\}$, $I = I \cup \{x_+\}$, $J = J \setminus \{x_-\}$ and successfully terminate Algorithm A1.

**Step 3: attempt to replace a point of $I$ distant from $x_c$.**
Determine the point in $I$ that is furthest from $x_c$, that is

$$x_- = \arg\max_{x_i \in I} \|x_i - x_c\|. \tag{2.15}$$

If $\|x_- - x_c\| > 2\Delta$, find a better point close to $x_c$ to replace $x_-$, that is compute

$$x_+ = \arg\max_{\|x - x_c\| \leq \Delta} S(x, x_-), \tag{2.16}$$

calculate $f(x_+)$, set $M = M \cup \{x_+\}$, perform the exchange $I = (I \setminus \{x_-\}) \cup \{x_+\}$ and successfully terminate Algorithm A1.

**Step 4: attempt to replace a point of $J$ close to $x_c$.**
Find the point in $J$ whose replacement maximally improves the interpolation set geometry, that is compute

$$x_- = \arg\max_{x_i \in J} [\arg\max_{\|x - x_c\| \leq \Delta} S(x, x_i)] \tag{2.17}$$

and let $x_+$ be the $x$ that realizes the inner maximum in (2.17). Then, if $S(x_+, x_-) > 1$, calculate $f(x_+)$, set $M = M \cup \{x_+\}$, $I = I \cup \{x_+\}$, $J = J \setminus \{x_-\}$ and successfully terminate Algorithm A1.

**Step 5: attempt to replace a point of $I$ close to $x_c$.**
Find the point in $I$ whose replacement maximally improves the interpolation set geometry, that is compute

$$x_- = \arg\max_{x_i \in I} [\arg\max_{\|x - x_c\| \leq \Delta} S(x, x_i)] \tag{2.18}$$

and let $x_+$ be the $x$ that realizes the inner maximum in (2.18). Then, if $S(x_+, x_-) \geq 2$, calculate $f(x_+)$, set $M = M \cup \{x_+\}$, perform the exchange $I = (I \setminus \{x_-\}) \cup \{x_+\}$ and successfully terminate Algorithm A1.

**Step 6: geometry deemed satisfactory.**

No geometry improvement can be identified with the current $\Delta$: unsuccessfully terminate Algorithm A1.

**end of A1**

In this algorithm, we have consistently used the function $S(\cdot,\cdot)$ to measure the geometrical improvement of the interpolation set. We have also attempted to reduce $J$ by considering the elimination of points of this set first (in Steps 2 and 4). If Step 6 is reached, this means that the model's geometry is good (in the sense defined above), given the current value of the trust-region radius, and thus that further improvement around $x_c$ will only be obtained by reducing this radius, forcing the interpolation points to be closer to $x_c$ if necessary.

## 2.2 The trust-region step

After examining how the model is built and how it can be improved if necessary, a suitable step is computed using this model by applying the standard trust-region technology: the step $s$ is calculated that minimizes the model (2.2) within the ball centered at $x_c$ and of radius $\Delta$, that is

$$s = \arg \min_{\|s\| \leq \Delta} m(x_c + s). \tag{2.19}$$

This calculation can be exact (see Hebden (1973), Moré (1978) or Dennis and Schnabel (1983) for instance) or approximate (see Powell (1970), Dennis and Mei (1979), Steihaug (1983), Toint (1981) or Conn *et al.* (1992) for examples). As we take the view that an objective function evaluation is very costly, we opt for the first choice and first compute $s_1 = -H^{-1}g$. If the length of $s_1$ is at most $\Delta$, we set $s = s_1$. Otherwise, we apply a Levenberg-Marquardt type technique to compute the (unique) value of $\lambda \geq 0$ such that

$$\|s_2\| = \| - (H + \lambda I_n)^{-1} g\| = \Delta, \tag{2.20}$$

where $I_n$ denote the $n \times n$ identity matrix, and set $s = s_2$. We then (as is standard) compute the ratio of achieved vs. predicted reduction

$$\rho = \frac{f(x_c) - f(x_c + s)}{m(x_c) - m(x_c + s)}. \tag{2.21}$$

However, instead of immediately proceeding to update the current iterate and trust-region radius, as would be traditional, we first examine if we cannot afford a possibly much longer step in the case where the model fits the true objective well enough. The motivation of these "long steps" or "jumps" is to use the current information to progress as much as possible using the current information (again keeping in mind the high cost of evaluating the objective). More precisely, we first test if $\rho \geq 0.9$, indicating an excellent ratio of achieved to predicted reduction. In this case, we examine in succession all the past points $x_i \in M \setminus I$, determine, for each of these points, the ratio

$$\rho_i = \frac{f(x_i) - f(x_c)}{m(x_i) - m(x_c)}, \tag{2.22}$$

and compute the maximal distance $\|x_i - x_c\|$ for all $x_i$ such that the ratio (2.22) is only slightly worse than (2.21), that is $\rho_i \geq 0.85$. Let $\Theta$ denote this maximum distance. If it is much larger than $\Delta$, the model (2.2) is thus likely to be valid in a much larger region than that in which the step $s$ has been computed. A larger trial step, or jump, $d$ may then be computed as

$$d = \arg \min_{\|s\| \leq \Theta} m(x_c + s), \tag{2.23}$$

with a chance of success very comparable to that of the original $s$. We may then decide to use the step $d$ instead of the shorter $s$ (redefining $s = d$) whenever sufficient progress is made, that is when

$$\rho_d = \frac{f(x_c) - f(x_c + d)}{m(x_c) - m(x_c + d)} \geq 0.05 \text{ and } f(x_c + d) < \min(f(x_c), f(x_c + s)). \tag{2.24}$$

The mechanism of these jumps provides the possibility of very rapid progress when the model is adequate. This is for instance the case when the objective function is itself quadratic in the region of interest, exactly or approximately.

Once a new step $s$ has been determined and the objective function evaluated at $x_c + s$, one then has to decide if this trial point should replace the current iterate $x_c$. As is usual in trust-region methods, the new point is accepted if the ratio of achieved vs. predicted reduction is larger than some small constant (0.05 in our implementation). If this is the case, one naturally includes the new point in the interpolation set $I$, simultaneously dropping the point in $I \cup J$ whose replacement by $x_c + s$ is most beneficial for the interpolation set geometry, whenever dropping is necessary (for example if $|I| = p$). The point $x_c + s$ may also be included in $I$ even if descent is not obtained, but if the interpolation set geometry is improved substantially by dropping another point of $(I \cup J) \setminus \{x_c\}$. However, both these possibilities may fail, in which case the iteration is declared unsuccessful, $x_c$ is kept unchanged and an attempt is made to improve the geometry underlying the model using Algorithm A1. This latter action may also be advisable when the achieved reduction is small, say when $\rho \leq 0.15$, even if the iteration is successful.

Finally, the trust-region radius must be updated. Strictly speaking, one might consider decreasing $\Delta$ only when all other methods for improving the geometry fail (that is when Algorithm A1 terminates unsuccessfully at Step 6) and the proposed trust-region step fails, but this makes the reduction of $\Delta$ much too slow, as it typically takes of the order of $p$ iterations to reach this stage. Instead, we propose to reduce $\Delta$ at all unsuccessful iterations, i.e. geometric and trust-region iterations. Similarly, $\Delta$ may be increased at every iteration that is clearly successful (that is when $\rho \geq 0.75$, say). A new iteration may then begin.

A last practical concern is to define an adequate stopping criterion. Two cases may occur in our algorithm. The first is when the objective function can no longer be significantly improved (as a consequence of the noise, for example), even if the trust-region radius is small, say below a tolerance $\epsilon_\Delta$, say. The second case is when a point $x$ is found such that $f(x) - f_\ell$ is below a small tolerance $\epsilon_f$, where $f_\ell$ is a lower bound on the objective function value. Such a bound is indeed often known by the user: for instance, $f_\ell = 0$ is a trivial choice for least-squares calculations. It can of course be set to minus infinity if the information is not available.

### 2.3 The complete algorithm

After outlining the mechanism of the algorithm, we are now in position to formally state it in full detail. In the description that follows, we denote by $\eta_r$ and $\eta_a$ the relative and absolute noise levels on the evaluation of $f(x)$, given $x$.

**General Algorithm**

The values for $x_0$, $\epsilon_\Delta$, $f_\ell$, $\epsilon_f$, $r > 1$, $\Delta_0$, $\eta_a$ and $\eta_r$ are given.

**Step 0: initialization.**

Define $\Delta = \Delta_0$ and set $M = I = \{x_0, x_1\}$ where $x_1$ is a random point different from $x_0$ such that $\|x_1 - x_0\| \leq \Delta$. Compute $f(x_0)$ and $f(x_1)$ and set $x_c = x_0$. Define $p$ according to (2.4) and $J = \{x_2, \dots x_{p-1}\}$ such that $I \cup J$ is poised.

**Step 1: start a new iteration.**

Select $x_t = \arg\min_{x \in I} f(x)$. If

$$f(x_t) < f(x_c) - 0.1\epsilon_\Delta\Delta, \tag{2.25}$$

set $x_c = x_t$.

**Step 2: convergence test.**

If

$$\Delta < \epsilon_\Delta \quad \text{or} \quad f(x_c) < f_\ell + \epsilon_f, \tag{2.26}$$

stop and return $x_c$ as an approximate solution.

**Step 3: compute a quadratic model, if possible.**

- If $|I| = p$, attempt to construct a quadratic model of the form

$$m(x_c + s) = f(x_c) + g^T s + \frac{1}{2} s^T H s \tag{2.27}$$

by defining the vector $g$ and the symmetric matrix $H$ such that

$$m(x) = f(x) \qquad \text{for each } \ x \in I. \tag{2.28}$$

- Else, that is if $|I| < p$,

**variant 1:** attempt to construct a quadratic model of the form (2.27) by defining the vector $g$ and the symmetric matrix $H$ as the solution of the problem

$$\min_{g,H} \|g\|^2 + \|H\|_F^2 \tag{2.29}$$

such that (2.28) holds. If the model cannot be safely computed, attempt to improve the model using Algorithm A1 and go to Step 7.

**variant 2:** attempt to construct a sub-quadratic model such that (2.28) holds.

**Step 4: compute a short step.**

Solve the trust region problem

$$\min_{\|s\|\leq\Delta} m(x_c + s) \tag{2.30}$$

for the step $s$. If the predicted model reduction is too small, that is if

$$m(x_c) - m(x_c + s) < \max\left[\epsilon_\Delta \Delta, \frac{\eta_a}{2}, \frac{\eta_r}{2}|f(x_c + s)|)\right], \tag{2.31}$$

then attempt to improve the model's geometry using Algorithm A1 and go to Step 7. Otherwise set $M = M \cup \{x_c + s\}$, compute $f(x_c + s)$ and the ratio (2.21).

**Step 5: attempt a long step.**

If $x_c + s$ is very successful, that is if $\rho \geq 0.9$, compute

$$\Theta = \min(1000\Delta, \max \|x_i - x_c\|), \tag{2.32}$$

where the maximum is taken on all $x_i \in M \setminus I$ such that

$$f(x_i) - f(x_c) >= 0.85(m(x_i) - m(x_c)) \text{ and } \|x_i - x_c\| \geq \Delta. \tag{2.33}$$

If $\Theta > (1 + r)\|s\|$, compute a long step $d$ by solving

$$\min_{\|d\|\leq\Theta} m(x_c + d). \tag{2.34}$$

If $\|d\| > (1 + r)\|s\|$, then set $M = M \cup \{x_c + d\}$ and evaluate $f(x_c + d)$. In this latter case and if (2.24) holds, redefine $s = d$ and $\rho = \rho_d$.

**Step 6: possibly add the new point to the interpolation set.**

Compute the best points in $I$ and $J$ that can be replaced by $x_c + s$, that is determine

$$x_I = \arg \max_{x\in I\setminus\{x_c\}} S(x_c + s, x) \text{ and } x_J = \arg \max_{x\in J} S(x_c + s, x) \tag{2.35}$$

and define

$$S_I = S(x_c + s, x_I) \text{ and } S_J = S(x_c + s, x_J). \tag{2.36}$$

Then, if sufficient descent is obtained, that is if $\rho \geq 0.05$, set

$$\begin{array}{ll} I = (I \setminus \{x_I\}) \cup \{x_c + s\} & \text{if } S_I > 2S_J, \\ I = I \cup \{x_c + s\} \text{ and } J = J \setminus \{x_J\} & \text{if } S_I \leq 2S_J. \end{array} \tag{2.37}$$

Else, if the geometry of the interpolation set can be improved, that is if $\max(S_I, S_J) > 1$, set

$$\begin{array}{ll} I = (I \setminus \{x_I\}) \cup \{x_c + s\} & \text{if } S_I > S_J, \\ I = I \cup \{x_c + s\} \text{ and } J = J \setminus \{x_J\} & \text{if } S_I \leq S_J. \end{array} \tag{2.38}$$

If both the preceding conditions fail or if $\rho < 0.15$, attempt to improve the model's geometry using Algorithm A1.

**Step 7: update the trust region radius.**

• If the step $s$ is successful, that is if $\rho \geq 0.75$, then set

$$\Delta = \min\left[r\Delta, \max(\Delta, r\|s\|)\right]. \tag{2.39}$$

• Else, if $\Delta > \epsilon_\Delta r$ and the model has been computed at Step 3 without running into conditioning problems, set

$$\Delta = \max(\epsilon_\Delta r, \frac{\Delta}{r}). \tag{2.40}$$

Then, if Algorithm A1 has failed to improve the geometry with the previous value of $\Delta$, apply it again with the updated value (2.40).

• If both the preceding conditions fail and Algorithm A1 has failed to improve the model's geometry with the current value of $\Delta$, set

$$\Delta = \frac{\Delta}{r}. \tag{2.41}$$

• In all cases, go to Step 1.

**end of General Algorithm**

We now need to comment on some features introduced in the algorithm, but not discussed above.

1. We note that, for the algorithm to be well-defined, one needs the set $I \cup J$ to remain poised throughout the calculation. Indeed, this property is required for the Lagrange interpolation function $L(\cdot,\cdot)$, and hence for the function $S(\cdot,\cdot)$, to exist. We also note that, at every step of the algorithm, *except in (2.37) and in Steps 2 and 3 of Algorithm A1*, care is taken that introducing a new point in the set $I \cup J$ does not deteriorate the interpolation set geometry (since a new point $x_+$ is only introduced if $S(x_+, x_-) > 1$ for some $x_- \in I \cup J$), and hence that the poised nature of $I \cup J$ is maintained. However, no such test is performed when sufficient descent is obtained or when far away points are replaced by points closer to $x_c$, in which case (2.37) forces the introduction of $x_c + s$ or $x_+$ in $I$. For the algorithm to be well-defined, we therefore need to prove that the determinant $\delta(I \cup J)$ defined in (2.5) remains nonzero, when these exchanges are performed.

   Consider the case of (2.37) first. If we define

$$\phi(x) = (\phi_1(x), \ldots, \phi_p(x))^T \tag{2.42}$$

   for any $x \in \mathbf{R}^n$, the fact that $\delta(I \cup J) \neq 0$ before the exchange implies that the columns of this determinant form a basis of $\mathbf{R}^p$, and hence that

$$\phi(x_c + s) = \sum_{x_j \in (I \cup J)} \alpha_j \phi(x_j) \tag{2.43}$$

   for some coefficients $\alpha_j$. Assume now that all $\alpha_j$ are zero for $j = 1, \ldots, p$. Hence $\phi(x_c + s) = 0$. As a consequence, the functions $\{\phi_i(\cdot)\}_{i=1}^p$ cannot form a basis for the

$n$-dimensional quadratics, since they do not span the the particular quadratic that is equal to one everywhere, including at $x_c + s$. This contradicts the definition of $\{\phi_i(\cdot)\}_{i=1}^p$ and there must thus exists at least one $j$ such that $\alpha_j \neq 0$. If $x_j \in I$, we may then perform the first exchange of (2.37) with $x_I = x_j$ and obtain that

$$\delta((I \setminus \{x_j\}) \cup \{x_c + s\} \cup J) = \alpha_j \delta(I \cup J) \neq 0. \tag{2.44}$$

(We obtain the same result if $x_j \in J$, performing then the second exchange of (2.37) with $x_J = x_j$.) This implies that $S_I$ or $S_J$ is nonzero and thus that the interpolation set remains poised after (2.37), as desired. We may also apply the same reasoning, with $x_c + s$ replaced by $x_+$, for verifying that $I$ stays poised after the exchanges of Steps 2 and 3 of Algorithm A1.

In theory, the above analysis guarantees that the model (2.2) and the Lagrange interpolation functions are always well-defined in the course of the computation. In practice however, it may happen that, although theoretically nonzero, $\delta(I \cup J)$ becomes comparable to machine precision. In this case, the model based on all available interpolation points is numerically undefined. The second of our algorithmic variants then automatically reduces the interpolation set to provide a well-defined model based on fewer points. The details of this procedure are once more described by Sauer and Xu (1995). The first variant instead forces improvement of the geometry of the interpolation set until this problem disappears: this is the purpose of the test at the end of the section of Step 3 related to this variant.

2. Another difficulty hidden in our formulation is the choice of the set $J$ at Step 0. We currently randomly generate points in a ball centered at $x_0$ and of radius $\Delta_0$ until the set $I \cup J$ is poised. We then repeatedly apply Algorithm A1 until the geometry (as measured by $S(\cdot,\cdot)$) cannot be improved by a factor higher than 1.1. Other techniques are obviously possible and may be preferable.

3. As for Algorithm A1, we have biased the formulation of Step 6 to encourage $I$ to contain $p$ points (thus yielding $J = \emptyset$) as soon as possible.

4. The parameter $r$ in Step 7 gives the amount by which the trust-region radius is decreased on unsuccessful iterations, or increased on successful ones. We currently use $r = 1.5$. Observe that Step 7 does not automatically reduce $\Delta$ when this reduction would produce a value below the threshold $\epsilon_\Delta$, which would in turn cause immediate termination at the next iteration. Instead, attempts are made to improve the quadratic model (2.2) as much as possible with the current $\Delta$. As a result, the radius is decreased (and the algorithm terminated) only when no improvement in function value appears possible within a region of radius $\epsilon_\Delta$.

   On the other hand, this termination mechanism may be judged too slow, as it typically requires of the order of $p$ iterations with

$$\Delta \in [\epsilon_\Delta, r\epsilon_\Delta]. \tag{2.45}$$

When the dimension increases, the algorithm may thus take a significant number of iterations in order to realize that a minimum has been found. One possible remedy is to also fit, for values of $\Delta$ satisfying (2.45), a linear model (whose geometry is good) around the current iterate. Building such a linear model typically requires of the order of $n$ iterations only, which is much lower than $p$ when $n$ grows. The calculation may then be terminated whenever this linear model does not yield any further significant descent in a region of radius $\epsilon_\Delta$ around the current iterate $x_c$.

5. The mechanism of Step 7 follows the traditional framework for trust-region methods, and may be refined in several ways. For instance, one might increase the value of $r$ if several iterations of the same nature (successful or unsuccessful) occur in sequence. One may also replace (2.39) by

$$\Delta = \max\left(\Delta_{\max}, \min\left[r\Delta, \max(\Delta, r\|s\|)\right]\right) \tag{2.46}$$

   in order to impose an upper bound on the step length, a sometimes wise modification in the context of finite precision arithmetic.

6. Observe finally that the predicted reduction is, in (2.31), compared to the noise on the objective function value: a predicted reduction that is comparable to this noise is not considered as significant. This natural comparison is in accordance with the recommendations of Conn *et al.* (1993).

# 3 Preliminary numerical experience

We now report on some very preliminary numerical experience using the proposed algorithm. The two variants (using minimum norm and sub-quadratic models, respectively) have been programmed in MATLAB 4.1.1 (The Mathworks Inc., 1992) and tested on a set of 20 small dimensional unconstrained problems from the CUTE collection (Bongartz *et al.*, 1995). All calculations were performed on an IBM RS6000 workstation, with the algorithms parameters given by

$$\epsilon_\Delta = 10^{-3}, \quad \epsilon_f = 10^{-5}, \quad f_\ell = -\infty \ \text{ and } \ \Delta_0 = 1. \tag{3.1}$$

The model computation of Step 3 in the algorithm was judged ill-conditioned when the condition number of the associated Vandermonde system exceeds $10^4/\epsilon_M$, where $\epsilon_M$ is the machine precision (for variant 1 when $|I| < p$) or when the smallest pivot in the interpolation scheme (see Sauer (1995) for a precise definition) was smaller than $10^{-8}$ (for all other cases). Successful termination was assumed for both algorithms when the stopping criterion (2.26) were satisfied, while failure was declared after 1000 unsuccessful iterations.

For comparison, we also ran LANCELOT (see Conn *et al.*, 1992) using finite differences and the symmetric rank-one quasi-Newton update, all other algorithmic options being set to their default values. Because we have assumed that the cost of computing the objective function value dominates all other algorithmic costs, we will only discuss the results obtained in terms of number of objective function evaluations. Furthermore, since both variants of the new method choose the initial points in $I \cup J$ randomly, we only report, for these variants, the average number of function evaluations taken over 10 runs, this average being then rounded to the nearest integer.

We first examine the case where the objective function can be evaluated without noise, in which case we may set

$$\eta_a = \eta_r = 0. \tag{3.2}$$

Table 1 reports the number of objective function evaluations required for convergence for each of the 20 problems. The first column of this table gives the name of the considered problem, as specified in the CUTE collection, the second column indicates its number of variables, and columns three to five give the number of functions evaluations for the two variants of our algorithm (the column heading "min.-norm" referring to the variant using minimum Frobenius norm models, and "sub-Q" referring to the variant using sub-quadratic ones), and for LANCELOT[1], respectively.

| problem's name | $n$ | min.-norm | sub-Q | LANCELOT |
|---|---|---|---|---|
| AKIVA | 2 | 63 | 63 | 41 |
| BARD | 2 | 58 | 61 | 43 |
| BEALE | 2 | 41 | 42 | 39 |
| BOX3 | 3 | 44 | 33 | 35 |
| BRKMCC | 2 | 18 | 18 | 14 |
| BROWNDEN | 4 | 101 | 99 | failed |
| CLIFF | 2 | 31 | 34 | 83 |
| CRAGGLVY | 4 | 131 | 142 | 80 |
| CUBE | 2 | 154 | 153 | 89 |
| DENSCHNA | 2 | 32 | 29 | 20 |
| GULF | 3 | 217 | 255 | 143 |
| HAIRY | 2 | 54 | 60 | 363 |
| HATFLDE | 3 | 66 | 67 | 60 |
| HELIX | 3 | 99 | 97 | 76 |
| KOWOSB | 4 | 75 | 68 | 114 |
| PFIT1LS | 3 | 193 | 189 | 219 |
| ROSENBR | 2 | 106 | 112 | 90 |
| SCHMVETT | 3 | 46 | 51 | 35 |
| SISSER | 2 | 26 | 25 | 35 |
| SNAIL | 2 | 329 | 327 | 778 |

Table 1: Number of function evaluations in the absence of noise

[1] Because LANCELOT reports the number of function iterations (each of which evaluates the objective function once) and the number of gradient evaluations separately, the total number of function evaluations was estimated according to the formula $1 +$ number of iterations $+ n($number of gradient evaluations $+1)$, where the first 1 takes the initial objective function evaluation into account, and where number of gradient evaluations has been increased by one to reflect that the package uses a mix of forward and central differences to estimate the gradients.

A few tentative conclusions may be drawn from the results presented in this table.

1. The number of function evaluations required to minimize the test functions remains moderate.

2. The variant using sub-quadratic models and that using the minimum Frobenius norm models do not appear to behave very differently on our examples.

3. In the absence of noise, both variants have an efficiency which is comparable to that of the finite-difference version of LANCELOT. They even seem to outperform this package for the more ill-conditioned or more nonlinear cases (CLIFF, HAIRY, PFIT1LS, SNAIL).

We now turn to the case where noise is present in the objective function and give, in Table 2, the number of function evaluations required for algorithm termination for two different noise levels. We have chosen the absolute and relative noise levels to be equal to 0.0001 and 0.01. We note that, as expected, noise typically prevents the algorithm finding the true minimum of the test functions. Moreover, there is no guarantee that the algorithm will stop at a point whose function value differs from the true noise-free minimum value by an amount comparable to the noise level, as it may indeed stop at any point at which the slope is of the order of the noise. As before, the number shown are averages over 10 runs, rounded to the nearest integer. The first number in each entry of columns three and four corresponds to a noise level of 0.0001 and the second to a level of 0.01. In all cases, $\eta_a$ and $\eta_r$ were chosen equal and identical to the considered noise level.

We also indicate in Table 2, the number of function evaluations required by a trust-region code in which the finite-difference step size was chosen as a function of the noise level, according to the recommendations of Dennis and Schnabel (1983), pages 97–99. This code is based on LANCELOT subroutines, but uses the machinery of this package in a context for which it was not originally designed. The resulting program however represents, in the authors' view, a reasonable yard-stick for measuring the performance of the new methods versus a more traditional technique. When results for this code are shown within square brackets, this indicates that the final objective function value was substantially above the optimal noise-free value, even taking the noise into account. For instance, the code did not succeed to reduce the objective function beyond its value at the starting point for problem SCHMVETT and noise level 0.01. These cases may be considered as practical failures, although they cannot formally be interpreted as such, because one of the specified stopping criteria[2] was met.

These results allow us to draw three more conclusions.

1. The comparable performance of both variants is confirmed in the presence of noise. It also seems to be significantly higher than that of the finite-differences based code.

[2]In order to provide meaningful comparison with the other algorithms, successful termination was assumed when either the approximate gradient was smaller than $10^{-5}$ or the trust region smaller than $10^{-3}$. The initial trust-region radius was set to one and the total number of function evaluations was estimated as for LANCELOT without noise.

| problem's name | $n$ | min.-norm | sub-Q | finite-differences trust-region code |
|---|---|---|---|---|
| AKIVA | 2 | 47/29 | 44/29 | [23]/[15] |
| BARD | 2 | 44/29 | 44/29 | 44/40 |
| BEALE | 2 | 38/28 | 41/28 | 49/68 |
| BOX3 | 3 | 35/26 | 27/23 | 47/120 |
| BRKMCC | 2 | 18/16 | 18/16 | 13/18 |
| BROWNDEN | 4 | 93/81 | 98/86 | 54/54 |
| CLIFF | 2 | 28/22 | 27/23 | 86/86 |
| CRAGGLVY | 4 | 105/46 | 99/60 | 85/90 |
| CUBE | 2 | 131/43 | 138/43 | 77/158 |
| DENSCHNA | 2 | 30/24 | 26/21 | 20/23 |
| GULF | 3 | 123/26 | 133/25 | 1791/198 |
| HAIRY | 2 | 49/50 | 72/40 | [200]/[18] |
| HATFLDE | 3 | 50/38 | 58/34 | 86/110 |
| HELIX | 3 | 95/87 | 69/50 | 80/92 |
| KOWOSB | 4 | 43/24 | 53/26 | 81/138 |
| PFIT1LS | 3 | failed/failed | 127/51 | 203/126 |
| ROSENBR | 2 | 100/59 | 118/73 | 95/85 |
| SCHMVETT | 3 | 38/25 | 37/24 | 24/[7] |
| SISSER | 2 | 25/20 | 23/19 | 35/53 |
| SNAIL | 2 | 333/15 | 377/25 | [207]/[36] |

Table 2: Number of function evaluations for absolute and relative noise levels 0.0001/0.01

2. The minimum-norm variant seems a little less robust than the sub-quadratic model one, which is itself substantially more robust that the finite-differences based code.

3. The effort (expressed in number of function evaluations) decreases when the noise increases when one of the two new algorithms is used. This is to be expected because more severe noise allows the minimization to stop sooner. However, we do not always observe this phenomenon for the finite-differences based code.

Furthermore, a closer look at the detailed results show that, although not guaranteed (as already mentioned), the best function value reported by the new algorithm is very often within the range $[f_*, f_*(1+\eta_r)+\eta_a]$, where $f_*$ is the minimum objective value for the noiseless problem.

These preliminary conclusions should of course be taken with caution, given the very limited amount of testing reported here. They are however encouraging.

## 4 Conclusions and perspectives

We have presented the current stage of development of a derivative free algorithm for unconstrained optimization. This algorithm is based on the trust-region paradigm and allows large steps to be taken when appropriate. Furthermore, the choice of models in the first iterations of the algorithm permits a substantial reduction of the objective function at the outset of the calculation. The algorithm is acceptably efficient and its performance is comparable, in the absence of noise, to that of good finite-difference quasi-Newton methods. Moreover, it behaviour also seems satisfactory in the presence of noise.

The authors realize that much additional development is necessary before the ideas presented here can result in a final algorithm and corresponding software. In particular, the following directions of investigation appear to be of interest and are the subject of ongoing work.

- If multivariable interpolation allows for sub-quadratic models, it also allows for models that are polynomials of degree higher than two. The use of these models suggests a number of interesting questions concerning their solution and the handling of the associated geometry.

- The amount of linear algebra involved at each iteration of the current version of the algorithm is relatively high, in accordance with the view that this cost is dwarfed by that of evaluating the objective function. However, there are cases where this workload may become excessive when the problem's dimension increases and the cost of evaluating the objective function is not very high. Variants of the algorithm requiring less linear algebra work per iteration are thus of interest.

  Furthermore, our current implementation of the variant using sub-quadratic models uses the Newton-type interpolation method of Sauer (1995), but yet bases its choice of a new interpolation point on (2.7)–(2.8), which uses the Lagrange interpolation functions. This is somewhat inefficient and should be modified to use the Newton interpolation functions throughout.

- Applying the ideas developed in this paper in the context of large-scale problems is also possible, in particular by the using the partially separable structure (see Griewank and Toint (1982), Conn *et al.* (1990) or Conn *et al.* (1992), for instance) which is very often associated with large problems. One can indeed use multivariate interpolation techniques to build a model for each of the element functions in a partially separable decomposition, provided such a decomposition is available. But this obviously raises a number of questions on how to maintain a suitable overall model and how to handle the geometry at the level of each element.

- The inclusion of constraints in our framework is also a very interesting development. A first stage is of course to adapt our technique in order to handle bound constrained problems, which can be done, for instance, by suitably modifying the trust region definition and using the $\ell_\infty$-norm for determining its shape (see Conn *et al.* (1988) or Conn *et al.* (1992)). But we also wish to handle more general constraints. As mentioned in the introduction, this can be done in several ways, including augmented Lagrangian techniques or exact penalty functions.

- The development of a proper convergence theory covering the final version of our algorithm is highly desirable.

- Besides the case where the evaluation of the objective function involves some noise whose level cannot be controlled, another interesting case is that of objective functions for which the accuracy of evaluation may be specified in advance, with the understanding that a more accurate evaluation may be (possibly substantially) more costly. It is thus of interest to adapt the algorithm presented here in such a way that it specifies, for each evaluation, a degree of accuracy that is sufficient for the algorithm to proceed efficiently while keeping the evaluation cost as low as possible.

- Finally, it is clear that considerable further numerical experience is needed to really assess the techniques discussed above, both from the reliability and efficiency point of view.

As implied by these comments and perspectives, the ideas presented here are thus but a preliminary first step in the development of a robust and efficient algorithm for optimization problems in which derivatives are unavailable. Research in this domain remains challenging and, in the authors' experience, clearly meets a strong and explicit need in several application areas.

## Acknowledgements

The authors are grateful to K. Mints and M. Ferris for their interest in this research and their useful comments.

## References

[Bongartz *et al.*, 1995] I. Bongartz, A. R. Conn, N.I.M. Gould, and Ph. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995.

[Booker, 1994] A. J. Booker. DOE for computer output. Technical Report BCSTECH-94-052, Boeing Computer Services, 1994.

[Box, 1966] M. J. Box. A comparison of several current optimization methods, and the use of transformations in constrained problems. *Computer Journal*, 9, 1966.

[Brent, 1973] R. P. Brent. *Algorithms for Minimization Without Derivatives.* Prentice-Hall, Engelwood Cliffs, USA, 1973.

[Brooks, 1958] S. H. Brooks. A discussion of random methods for seeking maxima. *Journal of Operations Research*, 6, 1958.

[Burns, 1995] J. Burns. The sensitivity equation approach to optimal control. Presentation at the IMA Workshop on Large-Scale Optimization, Minneapolis, 1995.

[Conn *et al.*, 1988] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25:433–460, 1988. See also same journal 26:764–767, 1989.

[Conn *et al.*, 1990] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project. In R. Glowinski and A. Lichnewsky, editors, *Computing Methods in Applied Sciences and Engineering*, pages 42–51, Philadelphia, USA, 1990. SIAM.

[Conn *et al.*, 1992] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*. Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[Conn *et al.*, 1993] A. R. Conn, Nick Gould, A. Sartenaer, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints. *SIAM Journal on Optimization*, 3(1):164–221, 1993.

[De Boor and Ron, 1992] C. De Boor and A. Ron. Computational aspects of polynomial interpolation in several variables. *Mathematics of Computation*, 58(198):705–727, 1992.

[Dennis and Mei, 1979] J. E. Dennis and H. H. W. Mei. Two new unconstrained optimization algorithms which use function and gradient values. *Journal of Optimization Theory and Applications*, 28(4):453–482, 1979.

[Dennis and Schnabel, 1983] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Englewood Cliffs, USA, 1983.

[Dennis and Torczon, 1991] J. E. Dennis and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474, 1991.

[Dixon, 1972] L. C. W. Dixon. *Nonlinear Optimisation*. The English Universities Press Ltd, London, 1972.

[Elkin, 1968] R. Elkin. *Convergence Theorems for Gauss-Seidel and Other Minimization Algorithms*. PhD thesis, University of Maryland, College Park, 1968.

[Gill *et al.*, 1981] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London and New York, 1981.

[Gill *et al.*, 1983] P. E. Gill, W. Murray, M. A. Saunders, and M. Wright. Computing forward-difference intervals for numerical optimization. *SIAM Journal on Scientific and Statistical Computing*, 4:310–321, 1983.

[Griewank and Toint, 1982] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312, London and New York, 1982. Academic Press.

[Griewank, 1989] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: recent developments and applications*, pages 83–108, Dordrecht, NL, 1989. Kluwer Academic Publishers.

[Hebden, 1973] M. D. Hebden. An algorithm for minimization using exact second derivatives. Technical Report T.P. 515, AERE Harwell Laboratory, Harwell, UK, 1973.

[Himmelblau, 1972] D. M. Himmelblau. *Applied Nonlinear Programming.* McGraw-Hill, New-York, 1972.

[Hooke and Jeeves, 1961] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8:212–229, 1961.

[Kelly and Wheeling, 1962] R. J. Kelly and R. F. Wheeling. A digital computer program for optimizing nonlinear functions. Technical report, Mobil Oil Corp., Research Dept., Princeton, New Jersey, 1962.

[Lucidi and Sciandrone, 1995] S. Lucidi and M. Sciandrone. A coordinate descent method without derivatives. Technical Report 10-95 (in preparation), University of Rome "La Sapienza", Rome, 1995.

[Mitchell *et al.*, 1989] T. J. Mitchell, J. Sacks, W. J. Welch, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[Moré, 1978] J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In G. A. Watson, editor, *Proceedings Dundee 1977*, Berlin, 1978. Springer Verlag. Lecture Notes in Mathematics.

[Morris *et al.*, 1991] M. Morris, C. Currin, T. J. Mitchell, and D. Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

[Nelder and Mead, 1965] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[Ortega and Rheinboldt, 1970] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables.* Academic Press, New York, 1970.

[Owen, 1992] A. B. Owen. Orthogonal arrays for computer experiments, integration and visualisation. *Statistica Sinica*, 2:439–452, 1992.

[Polyak, 1987] B. Polyak. *Introduction to Optimization.* Optimization Software Inc., New York, 1987.

[Powell, 1964] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 17:155–162, 1964.

[Powell, 1970] M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, New York, 1970. Academic Press.

[Powell, 1994a] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis, Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, volume 275, pages 51–67, Dordrecht, NL, 1994. Kluwer Academic Publishers.

[Powell, 1994b] M. J. D. Powell. A direct search optimization method that models the objective by quadratic interpolation. Presentation at the 5th Stockholm Optimization Days, 1994.

[Rosenbrock, 1960] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3:175–184, 1960.

[Sacks *et al.*, 1992] J. Sacks, H. P. Wynn, T. J. Mitchell, W. J. Welch, R. J. Buck, and M. Morris. Screening, predicting and computer experiments. *Technometrics*, 34(1):15–25, 1992.

[Sauer and Xu, 1995] Th. Sauer and Yuan Xu. On multivariate Lagrange interpolation. *Mathematics of Computation*, 64:1147–1170, 1995.

[Sauer, 1995] Th. Sauer. Computational aspects of multivariate polynomial interpolation. *Advances in Computational Mathematics*, 3:219–238, 1995.

[Spendley *et al.*, 1962] W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4, 1962.

[Steihaug, 1983] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[Stewart, 1967] G. W. Stewart. A modification of Davidon's minimization method to accept difference approximations of derivatives. *Journal of the ACM*, 14, 1967.

[Swann, 1964] W. H. Swann. Report on the development of a new direct search method of optimisation. Technical Report Research Note 64/3, I.C.I., Central Instruments Laboratory, 1964.

[The Mathworks Inc., 1992] The Mathworks Inc. Matlab reference guide. The Mathworks Inc., 1992.

[Toint, 1981] Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–88, London, 1981. Academic Press.

[Torczon, 1991] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM Journal on Optimization*, 1(1):123–145, 1991.