A trust region method for the optimization of noisy functions

C. Elster
Freiburger Materialforschungszentrum
Universität Freiburg
Stefan-Meier-Str.31
D-7800 Freiburg im Breisgau
Germany
email: clel@ibm.ruf.uni-freiburg.de

A. Neumaier
Institut für Angewandte Mathematik
Universität Freiburg
Hermann-Herder-Str.10
D-7800 Freiburg im Breisgau
Germany
email: neum@indi4.mathematik.uni-freiburg.de

ABSTRACT:

The optimization of noisy or not exactly known functions is a common problem occuring in various applications as for instance in the task of experimental optimization. The traditional tool for the treatment of such problems is the method of Nelder-Mead (NM). In this paper an alternative method based on a trust region approach (TR) is offered and compared to Nelder-Mead. On the standard collection of test functions for unconstrained optimization by Moré et al. [6], TR performs substantially more robust than NM. If performance is measured by the number of function evaluations, TR is on the average twice as fast as NM.

revised, January 1993

KEY WORDS:

unconstrained optimization, noisy functions, Nelder-Mead method, trust region method, quadratic model

1991 MSC Classification: 90C30

1 Introduction

The theory of local optimization provides powerful tools for the task of optimizing a smooth function f. For instance, Newton-type and quasi-Newton methods show superlinear convergence in the vicinity of a nondegenerate optimizer; moreover, global convergence results may be stated for quite wide classes of problems, e.g. for strictly convex functions. Thus these methods combine robustness and the property of being locally fast. A common feature of these methods is the use of a (local) quadratic approximation of f, in the case of Newton-type methods through a (possibly modified) Hessian, and in the case of a quasi-Newton method through an approximation of the Hessian, built up successively. However, these methods assume knowledge of the gradient or the Hessian, respectively, in contrast to other optimization techniques like the Nelder-Mead method (see for instance Gill et al. [2], pp.94-95) or the direction set method of Powell (as explained e.g. by Fletcher [1], pp.87-92). Nevertheless it is believed that a quasi-Newton method using a finite-difference approximation for the gradients seems to be most efficient for the task of optimizing smooth functions when only function values are available.

In some applications, however, the function to be minimized is only known within some (often unknown and low) precision. This might be due to the fact that evaluation of the function means measuring some physical or chemical quantity or performing an (expensive) finite element calculation for solving partial differential equations. The function values gained are corrupted by noise, namely stochastic measurement errors or discretization errors. This means that although the underlying function is smooth, the function values available show a discontinuous behaviour. Moreover, no gradient information is available.

For small variations δx around some x the corresponding function values do not reflect the local behaviour of the function but that of the noise. Therefore a finite-difference approach estimating the gradient (needed for a quasi-Newton method) breaks down.

In the literature (Nocedal [10], p.30, Fletcher [1], p.18, Powell [11], p.230) the recommended method for optimizing noisy functions is the simplex (or polytope) method of Nelder and Mead [9]. (The arguments of Gill et al. [2], p.92, against this method are valid only in the noiseless case; among the optimization algorithms available from IMSL, NAG and netlib, the Nelder-Mead method copes best with large noise.) This method does not use a local model of the function f and works without the assumption of continuity since it is controlled solely by comparing the relative size of the function values: this accounts for its ability to cope with noise. While no convergence

statements may be made (but see Torczon [12] for a convergence proof of a modified version), the method has been proved to be a useful method in many applications.
Although a noisy smooth function, i.e. a smooth function corrupted by noise, is effectively discontinuous, one may use the smoothness of the underlying function and try to separate it from the added noise. A method of this type was suggested by Glad and Goldstein [3] who estimate by the method of least squares repeatedly quadratic models using $O(n^2)$ points in a fixed scaled pattern around the best piont. While this allows them to prove convergence of their algorithm it makes each iteration unduly expensive (see section 3). Instead of using extra function evaluations at points arranged in a fixed pattern, Karidis and Turns [4] suggest a method based on using the previous function evaluations for the least squares fit, thus improving an approach of Winfield [13] who uses interpolation (which is sensitive to noise). In section 2 we present another algorithm based on this idea. Section 3 will then be devoted to an extensive numerical comparison of our algorithm with the method of Nelder-Mead.

We want to emphasize that the algorithm proposed is the result of extensive computer experiments with many features we have found in the literature, and some new ideas. We performed these tests as preparation for real life applications to the optimization of chemical experiments where a single measurement (function evaluation) takes several man hours (or days), and we needed to find out about the best algorithm which is robust and does not waste function evaluations.
For the present paper we eliminated from our code all features which did not significantly ($< 10\%$) degrade the performance. Thus we aimed for maximal simplicity subject to good performance, and we report here on the definition and behavior of this simplified version, without any fine-tuning. As a result, several (expensive) safeguards which would have been needed for proving convergence, have been dropped. Thus we are not able to *prove* convergence of our algorithm; however, we find it more important to have an algorithm that is *fast and robust in practice* than to have one which *converges in theory* (under necessarily idealized conditions like arbitrarily many function evaluations) but is slow in practice (and not even more robust).

2 A trust region method

We consider the unconstrained minimization of a smooth function $f(x)$ ($x \in \mathbf{R}^n$) in a small number of variables ($n \leq 10$, say), and assume that all information about f is gained by computing or measuring some noise-

corrupted value, $f^\sigma(x)$,

$$f^\sigma(x) \approx f(x). \tag{1}$$

The evaluation of f is assumed to be expensive, so that the efficiency of an optimization procedure can be measured exclusively by the number of function evaluations needed for the optimization.
We propose an algorithm which proceeds in two phases. Phase I is a starting procedure which produces a properly scaled initial trust region, whereas phase II attempts to locate a local minimizer by sequential optimization of quadratic models over ellipsoidal trust regions. Whenever some degeneracy condition is satisfied, the procedure is restarted, i.e. phase I is entered again. This feature helps recovering from false convergence. The two phases alternate until convergence.

Phase I.
Scaling phase: Given a starting point x^0, set $f_0 = f^\sigma(x^0)$. Let σ denote a robust bound of the noise. This bound may be found by repeating the first function evaluation various times, and maybe for slightly different x to account for the variation of systematic errors like discretization errors. We took 3 evaluations at the starting point and used $\sigma = 3\hat{\sigma}$, where $\hat{\sigma}$ denotes the estimated standard deviation.
In Phase I, scaling line searches along the coordinate directions $\pm e^i, i = 1, \ldots, n$, are performed as follows:
Set $f_1 = f^\sigma(x^0 + e^i)$, $f_2 = f^\sigma(x^0 - e^i)$ and define $\overline{\delta f} = \max_{i=1,2} |f_i - f_0|$.
If $\overline{\delta f} > \sigma$ and $\min_{i=1,2} f_i < f_0 + \sigma$, no more evaluations along this direction are taken, else if $\overline{\delta f} > \sigma$ and $\min_{i=1,2} f_i > f_0 + \sigma$, f is evaluated at $x^j = x^0 + \alpha_1^{2-j} e^i \quad (\alpha_1 < -1 \text{ fixed})$ for $j = 3, 4, \ldots$ until $f^\sigma(x^j) < f_0 + \sigma$ or $j = n_{\max}$.
In case that $\overline{\delta f} < \sigma$, f is evaluated at $x^j = x^0 + \alpha_1^{j-2} e^i$ for $j = 3, 4, \ldots$ until $|f^\sigma(x^j) - f_0| > \sigma$ or $j = n_{\max}$. (We used $n_{\max} = 10$ and $\alpha_1 = -5$.)
This ensures proper scales for each direction, e.g. the nearly smallest scales over which significant changes in f occur, and the danger of getting stuck within the noise at the starting point is reduced.
More precisely we defined the scaling factors s_i to be the smallest variation performed in direction e^i which caused a significant change of f (in the case that no significant change ocurred for all steplengths, s_i was set to the largest steplength.).

Phase II.
Descent phase: In order to model the behaviour of f based on the known values of f^σ, f is approximated by means of a surrogate function h. The fact that any smooth function is locally well described by a quadratic model

together with the fact that this is the simplest type of model permitting an unconstrained optimum makes the choice of a quadratic model for the surrogate function most natural. This choice assures correct local modelling and allows fast convergence in the vicinity of a local nondegenerate optimizer, just as quasi-Newton methods. (Of course, this only holds when the model is correctly estimated, and we do not enforce this strictly; see later remarks on convergence.) By optimizing the surrogate function over a trust region, a new approximation for the optimizer of f will be gained. The trust region is built with the help of the points used for determining the surrogate function.

The details are as follows. Let (x^i, f_i^σ) denote the points where f^σ has already been evaluated, where $i < j$ shall indicate that f_i^σ has been evaluated before f_j^σ. Put

$$f^* = \min_i f_i^\sigma \tag{2}$$

and let x^* denote the corresponding argument. We determine the set S consisting of the $\binom{n+2}{2} + k_1$ (we took $k_1 = 3$) nearest neighbours of x^*, where the distance d is defined by the scaled 2−norm,

$$d(x^*, x)^2 = \sum_{i=i}^{n} \left(x_i^* - x_i\right)^2 / s_i^2. \tag{3}$$

A trust region C will be defined in terms of the scaled distance, namely

$$C = \left\{ x \in \mathbf{R}^n | d(x, x^*)^2 \leq \rho \right\}, \tag{4}$$

where

$$\rho = \alpha_3^{1+\frac{N-i^*}{k_2}} \overline{d} \quad \text{with} \quad \overline{d} = \max_{x \in S} d(x, \tilde{x})^2; \tag{5}$$

α_3 is a fixed positive number < 1 (we chose $\alpha_3 = 0.5$) and k_2 some positive integer (we took $k_2 = \binom{n+2}{2}$). N denotes the total number of function evaluations used so far and i^* is the corresponding number of the best recent point. This choice of the trust region as a naturally scaled ellipsoid around the best recent point x^* and slightly smaller than the current cloud of points is fairly natural. It automatically enlarges the trust region in the case of a successful move to the boundary of the ellipsoid and results in slow contraction in the case of unsuccessful evaluations of f. The latter occurs typically because the quadratic approximation is insufficient. Thus, stronger contraction is forced whenever no improvement is made for longer times.
We also experimented with the traditional strategies proposed in the literature on trust regions (e.g. Fletcher [1], Moré [5]) – indeed, we began our investigations with these strategies, but on the average, these had less effect on the final results than the chosen one. We think this has several reasons:

The inacurate gradients obtained from the model fit reduce the effectiveness of a fine-tuned step size adaption. The comparison of predicted versus achieved reduction is corrupted by noise; note that the noise level σ estimated in phase I may be quite inacurrate. Finally, since function evaluations are used only, it takes after a very good step $O(n^2)$ iterations before the quadratic model picks up enough local information to get an accurate quadratic model on the more local level; in the intermediate steps, the model predictions are somewhat erratic independent of the details of the trust region strategy.

The surrogate function h of the form

$$h(x) = \alpha + g^t(x - x^*) + \frac{1}{2}(x - x^*)^t G(x - x^*) \tag{6}$$

(with $\binom{n+2}{2}$ free parameters) is then obtained by means of a least squares fit:

$$\theta := (\alpha, g, G) = \arg \min_{\alpha,g,G} \sum_{x \in S} \left(h(x) - f^\sigma(x)\right)^2; \tag{7}$$

a minimum norm solution is used when fewer than $\binom{n+2}{2}$ points are available or when the matrix $\mathbf{X}$, defined by rewriting (6) in the form

$$h(x) \equiv \sum_{j=1}^{\binom{n+2}{2}} \theta_j \mathbf{X}_{ij}, \quad x \in S, \tag{8}$$

is nearly rank deficient. In order that this is detected correctly, the columns of $\mathbf{X}$ were scaled such that the maximum of the absolute values over each column equals 1.

Note that, unless *all* $\binom{n+2}{2} + k_1$ points used for the fit are in a neighborhood of x^* where the objective function is nearly quadratic, the fit will produce large residuals $h(x) - f^\sigma(x)$, and there will be *no* region where the model is particularly accurate. This contrasts with the method of Glad and Goldstein [3] and with trust region methods based on exact gradients, and implies that the term "trust region" is a slight misnomer. On the other hand, these non-local fits are often very useful since they capture some gross global features of the objective function. Now an approximate local minimizer $\hat{x}$ over the trust region of the surrogate function is computed (see the details in section 3). In most iterations, $\hat{x}$ is used for the next evaluation of f^σ. However, the following safeguards are taken against clustering of the evaluation points in the ellipsoidal trust region in case that $\hat{x}$ lies too close to one of the evaluation points.

A set M of 2^n points $x = x^* + \alpha p$ within the trust region is constructed by choosing for each of the 2^n directions $p \in \mathbf{R}^n$ with $p_i = \pm s_i$ (i.e. p

shows in all corners of a scaled cube) a uniformly distributed random length $\alpha \in \left[0, (\rho/n)^{\frac{1}{2}}\right]$ with ρ from (5). Then the quantities

$$\overline{\delta} = \max_{x \in M} \min_{y \in S} d(x, y) \tag{9}$$

and

$$\underline{\delta} = \min_{x \in S} d(x, \hat{x}) \tag{10}$$

are calculated. Now f is evaluated at the model minimizer $\hat{x}$ when $\underline{\delta} \geq \alpha_4 \overline{\delta}$ (we used $\alpha_4 = 0.01$) or when at the last iteration f has not been evaluated at the (previous) $\hat{x}$. Otherwise, f is evaluated at the point where the maximum in (9) is attained.
However, unjustified contraction may occur because of noise. When the algorithm gets stuck at some point – so that for quite a large number of function evaluations no decrease occurs – the algorithm is forced to enter again phase I, i.e. a restart is performed around $(x^*, f^\sigma(x^*))$; all other old information is discarded. In our implementation this is done whenever no improvement occured in the last $k_3 = 3\binom{n+2}{2}$ evaluations of f during phase II. A restart is also performed when the Hessian was negligible ($\sum_{i,j} G_{ij}^2 < 10^{-12} n^2$); this indicates that the trust region has become so small that the second order terms in the model are dominated by noise.

Stopping rules : The algorithm is terminated whenever a prescribed number n_{max} of function evaluations has been performed (in our tests $n_{max} = 400$) or whenever the surrogate function indicates that a local minimizer is reached; to test this we used $d(x^*, \hat{x}) \leq 10^{-12}$. Furthermore, the algorithm is stopped if during both the scaling phase and the descent phase no significant improvement was achieved. With this termination rule, the behaviour of the algorithm is discussed in the numerical comparisons of the next section.
However, stopping rules which stop earlier may be based for example on comparisons of current improvement versus initial improvement. This is of interest for practical applications whenever the evaluation of f is expensive.
Convergence. The theoretical convergence properties of the method are unexplored. We conjecture that the restart feature and the scaling phase (or a slight modification of it) provide global convergence to a stationary point in the absense of noise, independent of the details of phase II.
For the descent phase, the least squares fit can be analyzed as in Glad and Goldstein [3], but their convergence theory applies only when the matrix $\mathbf{X}$ defined in (8) has bounded condition number. We had experimented with versions of our code which enforced a good condition of $\mathbf{X}$, but the number of extra function evaluations needed to do this significantly exceeded the gain due to the improved model. So we dropped this feature, and substituted it

by the far more effective return to the scaling phase when progress stalled. Ignoring the condition of $\mathbf{X}$, however, implies that at no stage we can guarantee that the gradient approximation obtained from the model is close to the exact gradient; therefore, it was impossible to imitate the convergence proofs for trust region methods which assume exact gradient information. For the same reason we cannot ensure superlinear $\binom{n+2}{2}$-step convergence although this is often observed.
Though this state of affairs is somewhat unsatisfactory, we feel that the importance of "noisy" optimization and the significant improvements observed on the test functions with our method (see section 3) are of sufficient interest to users of optimization techniques to justify publication.

3 Numerical results

In order to investigate the reliability and efficiency of the algorithm, extensive numerical tests were performed on the standard collection of test functions of Moré et al. [6] for solving unconstrained problems. We compare the Nelder-Mead method (referred to as NM) as implemented in the NAG library [8] (Mark 14) with the trust region method of section 2, referred to as TR. The least squares problem (7) was solved (after scaling) by the NAG routine G02DAF, and the surrogate optimization to find $\hat{x}$ was performed by computing a spectral factorization of the scaled Hessian $\mathbf{SGS} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ ($\mathbf{\Lambda}$ diagonal, $\mathbf{Q}$ orthogonal, $\mathbf{S} = \mathrm{diag}(s_i)$) performed by the NAG routine F02ABF and approximately solving the inequality

$$d^2(x^*, x^\lambda) \leq \rho \tag{11}$$

with

$$x^\lambda = -\mathbf{SQ}\left(\mathbf{\Lambda} + \lambda\mathbf{I}\right)^{-1}\mathbf{Q}^T\mathbf{S}g. \tag{12}$$

for the minimal nonnegative value of λ. This gives an approximate minimizer of the surrogate function within the trust region. (For a proper treatment of the exceptional "hard" case when $\mathbf{\Lambda} + \lambda\mathbf{I}$ is singular see Moré and Sorensen [7]; we did not implement this.)

We first compared our method with various other methods on the test functions used by Glad and Goldstein [3] in order to show that the Nelder-Mead method is the right benchmark for systematic tests on noisy functions. The test functions are the 2-dimensional Rosenbrock function and the helical valley function, with standard starting points (no. xx and no. xx in [6]). Each function value f^σ was obtained as

$$f^\sigma(x) = f(x) + \epsilon), \quad \epsilon \sim N(0, \sigma), \tag{13}$$

where $N(0, \sigma^2)$ denotes the Gaussian distribution with zero mean and variance σ^2, i.e. stochastic noise was used for the test problems.
...
...
For the systematic tests we used a relative error noise simulation in order to maintain the ability to reach the optimum in the many cases where the minimum function value is zero. Thus each function value f^σ was obtained as

$$f^\sigma(x) = f(x)(1 + \epsilon), \quad \epsilon \sim N(0, \sigma). \tag{14}$$

To model realistic noise conditions, a fairly large value of $\sigma = 0.1$ was taken. Note that the surrogate function is obtained by means of minimizing a model corresponding to absolute errors, hence no specific adaption to the simulation (14) was made. Note also that for the examples where the minimum function value is zero, the relative error noise is larger when one is not yet close to the minimum, which makes it harder for the least squares fit to identify the correct location of the minimizer, which is very well defined. In contrast, absolute error noise results in a large region surrounding the minimizer which, due to the noise, is indistinguishable from the optimum. Thus we can expect similar behaviour in real world problems, where the form of the error often remains unknown. However, a priori knowledge about the noise may of course be included by introducing appropriate weights into the least squares expression.
Table I displays the corresponding results. Column 1 shows the problem type; the first number indicates the test problem, the second number, *st*, determines the starting point x^0 of the optimization, with

$$x^0_\gamma = \tilde{x}^{st}_\gamma, \quad \gamma = 1, \ldots, n, \tag{15}$$

and $\tilde{x}$ denoting the standard starting point for the corresponding test problem [6]. This means, that for all but one problem (problem 20, with $\tilde{x}^0 = 0$), the performance was tested for starting points near to, at intermediate distance, and far away from the solution. With 18 test functions, the total number of test cases was $3 \cdot 17 + 1 = 52$. The third number indicates the dimension of the problem.
The second column contains the algorithm used. The next four columns titled q_i display the quotients

$$q_i = \frac{\underline{f}_i - \underline{f}}{f_0 - \underline{f}}, \tag{16}$$

with f_0 the value of f (and *not* of f^σ) at x^0, $\underline{f}_i = \min_{j \le i} f_j$ (and *not* of f^σ_j) and $\underline{f}$ the global minimum of f. The numbers q_i (which are of course not

available in real applications) describe the speed of approaching the global minimum of the smooth uncorrupted function. In most cases, $\underline{f} = 0$, except for

$$(\text{problem number} : \underline{f}) = \begin{cases} (9 : 1.12793 \cdot 10^{-8}), \\ (20 : 2.28767 \cdot 10^{-3}), \\ (23 : 2.24997 \cdot 10^{-5}), \\ (24 : 9.37629 \cdot 10^{-6}), \\ (16 : 85822.2). \end{cases}$$

The 7th column displays q_f, i.e. the quotient (16) reached with n_f (8th column) function evaluations. Here n_f denotes the total number of function evaluations, limited by $n_{max} = 400$, i.e. n_f equals 400 unless the algorithm has terminated earlier.
The next three columns display the number of function evaluations n_i necessary for reducing q_i to less than $10^{-1}, 10^{-2}$ and 10^{-6} respectively. A dash indicates that the corresponding reduction was not achieved.
The last column contains the number of times the scaling phase was entered by TR.

Table I (appended at the end of the paper).

Let us focus on some aspects:
Recognition of optimizer: For the test problem $(11, 1, 3)$, the starting point is already optimal. Thus the only nontrivial entry in the corresponding row of table I is the number n_f of function values needed to recognize optimality. We get $n_f = 68$ for NM and $n_f = 53$ for TR.

Robustness : For many applications, the ability to obtain a significant reduction of the initial function value is the most relevant aspect. As indicators we use the number fail_1 and fail_2 of failures of reducing the function by a factor of 0.1 or 0.01 (relatively as described above). One recognizes that NM fails to achieve a 0.1 reduction in 27 cases (=52%) and a 0.01 reduction in 29 cases (=56%), whereas TR fails in only 4 (= 8 %) and 5 (=10%) cases, respectively.
Sometimes it is of importance to achieve a large reduction. This is much harder to achieve. Indeed, the number fail_6 of failures of making (again relative in the above sense) a 10^{-6} reduction with 400 function evaluations was 41 (=79%) for NM and 16 (=31%) for TR.

Speed : In order to compare the speed of the methods we give, for $i = 1, 2, 6$, the mean number nf_i of function evaluation for achieving a 10^{-1} reduction

and so on. Cases where a prescribed reduction was not achieved were counted with the value $n_{max} = 400$. The results for NM are $\mathrm{nf}_1 = 220$, $\mathrm{nf}_2 = 238$, $\mathrm{nf}_6 = 329$ in contrast to $\mathrm{nf}_1 = 72$, $\mathrm{nf}_2 = 94$ and $\mathrm{nf}_6 = 229$ for TR. We see that for achieving a moderate decrease (0.1 or 0.01), TR is on the average about twice as fast as NM.

Superiority : Finally we count how often each method performed better in all respects, i.e., when the values for $q_{50}, q_{100}, q_{150}, q_{200}, n_1, n_2$, and n_6 were all better or equal (with at least one value better) for one method. In only 5 cases (=10%), NM was uniformly better than TR, whereas TR performed in 25 cases (=48%) uniformly better than NM.

Thus, the proposed trust region method TR is, by each standard, a significant improvement over the Nelder-Mead method NM. However, since, in contrast to NM, each step of TR requires extensive calculations, the TR method can be recommended only for problems of moderately low dimension where each function evaluation is expensive (as in the examples mentioned in the introduction). Since the test functions used are all very cheap to evaluate, we refrained from giving timing results. In order to limit storage, one may hold only a fixed number of old points within the algorithm, but performance is degraded if this number is taken too small.
Although random numbers were used to obtained these results, the relations stated above seem to be quite independent from the particular realizations used. In order to illustrate the variation of the above given results, a couple of successive runs through the test set for both methods were performed. Summaries of the corresponding results are displayed in table II. The first column contents the batch number, columns $2-4$ show the number of failures for reducing the objective function by a factor of 10^{-1}, 10^{-2} and 10^{-6} for both methods. Columns $5-7$ display the values for $\overline{n}_1$, $\overline{n}_2$ and $\overline{n}_6$ for both methods. The last column contains the number of cases where one of the methods performed better in all aspects.

Table II

Run	fail_1	fail_2	fail_6	nf_1	nf_2	nf_6	better
	NM/TR	NM/TR	NM/TR	NM/TR	NM/TR	NM/TR	NM/TR
1	26/6	28/10	41/20	214/101	231/133	331/257	3/25
2	25/4	27/6	41/21	206/79	226/118	328/254	6/24
3	22/4	25/8	42/20	183/82	209/123	333/255	4/27
4	25/7	28/9	42/22	206/89	229/126	335/253	3/27
5	23/6	28/7	39/19	191/91	232/122	317/249	2/34

Finally we mention that modest changes of the parameters k_l and α_l in TR did not significantly alter the overall performance of TR, although it can make a big difference for particular test functions.

4 Conclusions

The problem of unconstrained optimization of noisy functions was studied and an alternative to the standard method of Nelder-Mead was proposed. The algorithm is based on the assumption that the underlying function is smooth, but evaluations are corrupted by noise. By fitting a quadratic model to a properly chosen subset of already evaluated points, the behaviour of the underlying function is approximated and the next evaluation point is determined by minimizing the model under an ellipsoidal constraint. Restarts add to the robustness of the method.

Experiments with various modifications of our methods allow the conclusion that several features established for gradient-related methods are less relevant in the absence of exact gradient information and the presence of noise. Instead, space exploration strategies seem to be most effective in this situation to enhance progress and robustness. The discrepancy between practical performance and theoretical analysis for such problems is still very large compared to the state of affairs in optimization with exact gradients documented e.g. in [10].

An extensive numerical comparison with the Nelder-Mead method based on the test problems of Moré et. al. shows that the new method is much more reliable and is on the average twice as fast in terms of total number of function evaluations. Since the time for intermediate calculations significantly exceeds that used by the Nelder-Mead algorithm, the conclusion is that TR should be applied in situations where function evaluation costs are the dominating factor.

References

[1] Fletcher, R., *Practical Methods of Optimization*, Wiley, New York, 1987.

[2] Gill, E., Murray, W. and Wright, M.H., *Practical Optimization*, Academic Press, London, 1981.

[3] Glad, T. Goldstein, A., Optimization of functions whose values are subject to small errors, *BIT* **17** (1977), 160-169.

[4] Karidis, J.P., Turns, S.R., Efficient optimization of computationally expensive objective functions, *IBM Research Report* RC10698 (#47972), Yorktown Heights, NY (1984).

[5] Moré, J.J., Recent developments in algorithms and software for trust region methods, pp. 256-287 in: *Mathematical Programming: The State of the Art* (A. Bachem et al, eds.), Springer, Berlin 1983.

[6] Moré, J.J., Garbow, B.S. and Hillstrom, K.E., Testing Unconstrained Optimization Software. *ACM Trans. Math. Software* **7** (1981), 17-41.

[7] Moré, J.J., Sorensen, D.C., Computing a trust region step, *SIAM J. Sci. Stat. Comput.* **4** (1983), 553-572.

[8] Numerical Algorithm Group (NAG), *Fortran Library Mark* 14, Oxford USA, 1990.

[9] Nelder, J.A. and Mead, R., A simplex method for function minimization, *Computer Journal*, **7** (1965), 308.

[10] Nocedal, J., Theory of algorithms for unconstrained optimization, pp. 199-242 in: *Acta Numerica 1992* (A. Iserles, ed.), Cambridge University Press, Cambridge 1992.

[11] Powell, M.S.D., A review of algorithms for nonlinear equations and unconstrained optimization, p.220-232 in: McKenna J. & Temam R. (eds.), *ICIAM 1987 Proceedings*, SIAM, Philadelphia 1988.

[12] Torczon, V., On the convergence of the multidimensional search algorithm, *SIAM J. Optimization* **1** (1991), 123-145.

[13] Winfield, D., Function minimization by interpolation in a data table, *J. Inst. Math. Appl.* **12** (1973), 339-348.