USE OF QUADRATIC MODELS WITH MESH ADAPTIVE DIRECT SEARCH FOR CONSTRAINED BLACK BOX OPTIMIZATION

ANDREW R. CONN * AND SÉBASTIEN LE DIGABEL †

Abstract: We consider derivative-free optimization, and in particular black box optimization, where the functions to minimize and the functions representing the constraints are given by black boxes without derivatives. Two fundamental families of methods are available: Model-based methods and directional direct search algorithms. This work exploits the flexibility of the second type of method in order to integrate to a limited extent the models used in the first family. Intensive numerical tests on two sets of forty-eight and one hundred and four test problems illustrate the efficiency of this hybridization and show that the use of models improves significantly the mesh adaptive direct search algorithm.

Keywords: Black box optimization, constrained optimization, quadratic models, mesh adaptive direct search algorithms (MADS).

1. Introduction. Although it is preferable to use derivative-based optimization algorithms, there are many instances where (at least some) derivatives are unavailable or unreliable. Furthermore, optimization's potential for application to complex applied problems, both inside and outside engineering, makes it more and more compelling to consider problems for which accurate derivatives are not always available. This can be because the underlying problem is given by simulation and the problem is sufficiently complex and the model sufficiently crude that any approximation to the derivatives obtained directly from the simulation are just too noisy to be useful or they are too expensive to obtain. Another common reason is legacy code that, though it could have readily provided derivatives, did not. One might ask 'Why not just use automatic differentiation techniques?'. Well, in particular, if the objective function is computed using a black box simulation package, automatic differentiation is typically impossible, and even in the case where the computation is more accessible, often legacy issues may make such an approach unacceptable. Finally, in the case of noisy function evaluations the computed derivatives will usually not be sufficiently accurate and in these circumstances it is frequently the case that a derivative-free method will outperform derivative-based approach.

Finding a way to compute the minimum of a function simply by random sampling is clearly far from ideal. The methods we will consider are designed to do better but they do not rely on derivative information of the objective function or (possibly some of) the constraints, nor are the methods designed explicitly to approximate these derivatives, but rather they approximate these functions using models based upon sampling and/or they exploit knowledge of a sample set of function values.

We consider the following problem

$$\min_{x\in\Omega} f(x) \tag{1.1}$$

where $\Omega = \{x \in X : g_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$ denotes the feasible region, $f, g_j : X \to \mathbb{R} \cup \{\infty\}$ for all $j \in J$, and X is a subset of $\mathbb{R}^n$. Problem (1.1) may be considered as a constrained black box optimization problem since the functions f and $g_j, j \in J$, are often black box functions with no derivative information, and so possibly noisy and costly to evaluate. Such functions are frequently the result of a computer simulation that may fail to evaluate even for feasible points. This simulation may also be non-deterministic so that two evaluations at the same point can give, hopefully slightly, different results.

*IBM Business Analytics and Mathematical Sciences, IBM T J Watson Research Center, P. O. Box 218, Yorktown Heights NY 10598, USA, www.research.ibm.com/people/a/arconn.

†GERAD and Département de mathématiques et de génie industriel, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, Québec H3C 3A7 Canada, www.gerad.ca/Sebastien.Le.Digabel.

The constraint functions g_j, $j \in J$, are those for which a measure of violation is available. They may or may not have derivatives but the algorithm presented in this paper treats them all as general black box constraints. This is clearly not ideal and a later paper will treat the constraints for which derivatives are available in a more appropriate manner. In general we consider that all these constraints need only to be satisfied at the final iterate of an algorithm and that their violation does not prevent successful execution of the simulation. On the other hand, constraints in X are not quantifiable and typically correspond to constraints that must be satisfied at each iterate in order for the simulation to execute. A last type of constraints is called 'hidden' constraints and arises when the simulation fails even for points inside X or Ω. For example, although somewhat artificial, some simulation might be using the log function, so implicitly there is the constraint that its argument is positive. Finally note that equality constraints are not considered since they are not compatible with the methods used in this paper. However, again we expect to be able to explicitly handle in a subsequent paper, at least some, equality constraints.

Methods for tackling such problems include derivative-free algorithms [13]. These methods may be classified into model-based and direct search algorithms. This paper proposes a hybrid method integrating quadratic models in a direct search framework, and its objective is to demonstrate that the addition of models improves the efficiency of the direct search. Such a coupling has been achieved with the SID-PSM algorithm described in [15] with promising results and the present work follows mainly the same lines, but with the addition of constraint handling. Nevertheless, some important differences remain: the mesh adaptive direct search (MADS [6, 7]) algorithm is used instead of the generalized pattern search (GPS [30]), a strategy to maintain well poisedness is implemented, and the quadratic models are used instead of simplex gradients for trial point ordering, although we remark that simplex gradients can be thought of as using linear models. Finally, the procedure developed here has been designed to satisfy the MADS convergence analysis. The implicit filtering method described in [19] may also be viewed as a hybrid method combining direct search and quadratic models although the models are not used the same way as in [15] or in the present work.

The paper is divided as follows: Section 2 gives an overview of the quadratic models and of the MADS algorithm. Section 3 describes the new method coupling MADS and the models and Section 4 presents and comments on the numerical tests on two sets of forty-eight and one hundred and four problems from the literature. These tests demonstrate significant improvement of the MADS method due to the use of models.

2. Derivative-free algorithms and models. As already mentioned, there are two main families of derivative-free methods. The model-based methods, which build (quadratic) approximations to be optimized, either with a line-search or within a trust region, and have to explicitly take care of the geometry when necessary, and then directional direct search methods, which implicitly take care of the geometry. Generally, for smoother problems and when models may be considered accurate, model-based are preferable to direct search methods in that they are able to exploit any structure inherently in the problem. But they are necessarily more complex than direct search methods. Thus if little or no structure is present there is no strong case for model-based methods. In particular, since the geometry of the sample points can be important in model-based methods, one has to periodically deal with that explicitly, whereas direct search methods typically finesse the issue by the choice of their directions. Moreover, direct search methods are much more effectively and easily parallelized than model-based approaches (see [9] for a brief literature review on parallel direct search methods).

This paper proposes the use of quadratic models in a directional direct search framework. The objective is to benefit from the implicit geometry created by the direct search in order to create models that will capture some of the problem properties. We first describe the polynomial interpolation strategies that we have chosen to construct quadratic models. We note that alternative strategies, for example using, radial basis functions as in [4, 31], or linear or higher order models could easily have been chosen instead. The second part of this section focuses on directional direct search methods.

2.1. Quadratic models. We consider the natural basis of the space of polynomials of degree less than or equal to two in $\mathbb{R}^n$, which has $q+1=(n+1)(n+2)/2$ elements,

$$\phi(x)=(\phi_0(x),\phi_1(x),\ldots,\phi_q(x))^\top=\left(1,x_1,x_2,\ldots,x_n,\frac{x_1^2}{2},\frac{x_2^2}{2},\ldots,\frac{x_n^2}{2},x_1x_2,x_1x_3,\ldots,x_{n-1}x_n\right)^\top.$$

Assuming suitable interpolation points, a unique quadratic model of f, m_f, is defined by $q+1$ parameters, $\alpha\in\mathbb{R}^{q+1}$, evaluated at x with $m_f(x)=\alpha^\top\phi(x)$. More generally, in order to obtain an approximation of the objective function, f, an interpolation set $Y=\{y^0,y^1,\ldots,y^p\}$ of $p+1$ elements of $\mathbb{R}^n$ is invoked. Similarly, we would want to determine models for any constraints $g_j, j\in J$ using the same interpolation set.

We first assume that the set of interpolation points Y is given and we do not try to exploit or modify its geometry. The latter will be discussed in Section 2.2. The construction of our model consists of determining the vector α such that $\sum\limits_{y\in Y}(f(y)-m_f(y))^2$ is as small as possible. This implies that we have to solve the system

$$M(\phi,Y)\alpha=f(Y) \tag{2.1}$$

with $f(Y)=\left(f(y^0),f(y^2),\ldots,f(y^p)\right)^\top$ and

$$M(\phi,Y)=\begin{bmatrix}\phi_0(y^0)&\phi_1(y^0)&\ldots&\phi_q(y^0)\\\phi_0(y^1)&\phi_1(y^1)&\ldots&\phi_q(y^1)\\\vdots&\vdots&\vdots&\vdots\\\phi_0(y^p)&\phi_1(y^p)&\ldots&\phi_q(y^p)\end{bmatrix}\in\mathbb{R}^{p+1,q+1}.$$

Generally, system (2.1) may have one, several, or no solutions. If $p\geq q$, i.e. there are more interpolation points than necessary, it is overdetermined and regression is used in order to find a solution in the least square sense, and so we solve

$$\min_{\alpha\in\mathbb{R}^{q+1}}\|M(\phi,Y)\alpha-f(Y)\|^2, \tag{2.2}$$

where here and in what follows $\|\cdot\|$ corresponds to the Euclidean norm. If $M(\phi,Y)$ has full column rank, there is an unique solution given by

$$\alpha=\left[M(\phi,Y)^\top M(\phi,Y)\right]^{-1}M(\phi,Y)^\top f(Y). \tag{2.3}$$

When $p<q$, i.e. there are not enough interpolation points, the system of equations (2.1) is underdetermined and there are an infinite number of solutions. The problem may be regularized (i.e. be made to have a unique solution) by, for example, determining the least squares solution, $\min\limits_{\alpha\in\mathbb{R}^{q+1}}\|M(\phi,Y)\alpha-f(Y)\|^2$, for which $\|\alpha\|$ is minimal. Alternatively one can invoke minimum Frobenius norm (MFN) interpolation, as will now be described, since this is our preference.

MFN interpolation consists of choosing a solution of (2.1) that minimizes the Frobenius norm of the curvature subject to the interpolation conditions. This is captured in the quadratic terms of α. Thus writing α as $\alpha=\begin{bmatrix}\alpha_L\\\alpha_Q\end{bmatrix}$ with $\alpha_L\in\mathbb{R}^{n+1}$, $\alpha_Q\in\mathbb{R}^{n_Q}$, and $n_Q=n(n+1)/2$, our model at x, $m_f(x)$ is then given by $m_f(x)=\alpha_L^\top\phi_L(x)+\alpha_Q^\top\phi_Q(x)$ with $\phi_L=(1,x_1,x_2,\ldots,x_n)^\top$ and $\phi_Q=\left(\frac{x_1^2}{2},\frac{x_2^2}{2},\ldots,\frac{x_n^2}{2},x_1x_2,x_1x_3,\ldots,x_{n-1}x_n\right)^\top$. The corresponding MFN α vector is then found by solving

$$\min_{\alpha_Q\in\mathbb{R}^{n_Q}}\frac{1}{2}\|\alpha_Q\|^2\quad\text{subject to}\quad M(\phi_L,Y)\alpha_L+M(\phi_Q,Y)\alpha_Q=f(Y). \tag{2.4}$$

This problem has a unique solution if the matrix

$$F(\phi, Y) = \begin{bmatrix} M(\phi_Q, Y)M(\phi_Q, Y)^\top & M(\phi_L, Y) \\ M(\phi_L, Y)^\top & 0 \end{bmatrix} \in \mathbb{R}^{p+n+2,p+n+2}$$

is nonsingular, in which case we say that the sample set Y is poised in the minimum Frobenius norm sense. Note that

$$\min_{\alpha_Q \in \mathbb{R}^{n_Q}} \frac{1}{2}\|\alpha_Q\|^2$$

is equivalent to minimizing the Frobenius norm of the Hessian matrix of the model $m_f(x)$. Solving

$$F(\phi, Y) \begin{bmatrix} \mu \\ \alpha_L \end{bmatrix} = \begin{bmatrix} f(Y) \\ 0 \end{bmatrix} \tag{2.5}$$

gives $\alpha_L \in \mathbb{R}^{n+1}$ and $\mu \in \mathbb{R}^{p+1}$, the Lagrange multipliers of problem (2.4). Then computing

$$\alpha_Q = M(\phi_Q, Y)^\top \mu \in \mathbb{R}^{n_Q} \tag{2.6}$$

completes the model construction.

2.2. Well poisedness. We now discuss the general geometrical properties of the interpolation set Y, but with a particular definition of well poisedness. In the determined case, $p = q = (n+1)(n+2)/2 - 1$, the set Y is said to be poised if $M(\phi, Y)$ is nonsingular but from a computational point of view we more generally need some measure of the quality of the poisedness. The quality of the geometry of Y may be measured using Lagrange polynomials of degree two expressed in the natural basis, $L_i(x) = \beta_i^\top \phi(x)$ with $\beta_i \in \mathbb{R}^{q+1}$ and $i = 0, 1, \ldots, q$, and which satisfy $L_i(y^j) = 1$ if $i = j$ and $L_i(y^j) = 0$ otherwise. We say that Y is Λ-poised if it is shifted and scaled so that the smallest ball containing Y is $B(0; 1)$, the ball of radius 1 (with the Euclidean norm) centered at the origin, and if

$$\max_{0 \le i \le q} \max_{x \in B(0;1)} |L_i(x)| \;\le\; \Lambda.$$

The smaller Λ, the better the geometry of Y for interpolation (Y is well-poised).

Chapter 6 of [13] introduces various techniques and algorithms allowing one to maintain and improve well poisedness within model-based methods. In the context of directional direct search algorithms, using such techniques is likely to be redundant because direct searches tend to implicitly produce trial point sets with good geometric properties. This impression is reinforced by the numerical results in [15], and furthermore since no well poisedness maintenance is carried out during their search step there is some similarity to the experience of G. Fasano, J.L. Morales, and J. Nocedal in [17], who found that the omission of the geometry phase did not seem to harm the efficiency and robustness of their algorithm. In their experience, although high ill-conditioning in the interpolation system was observed, along with models that were frequently poor, the overall progress appeared to be satisfactory. They conjectured that there may be a self-correction mechanism that prevents the conditioning from growing steadily but this explanation is not clear to the present authors. In any case we felt that there was some interest in considering, both maintaining and ignoring, well poisedness in our context, verifying these impressions and exploring whether **any** improvement would be made by considering the former. Consequently we decided to test a simple strategy for well poisedness, in the over-determined case, when there are more interpolation points than needed ($p > q$) [1]. This strategy, when enabled, is applied after a first

[1] It may be more natural to be concerned about well poisedness in the undetermined case but since we did not want to incur the costs of new function evaluations and in addition we assumed the direct method promotes good geometry naturally, we only considered the overdetermined case.

set Y has been constructed with $p > q$, and produces a new set with $p = q$. At the same time it constructs the Lagrange polynomial necessary for interpolation so that the models are simply computed via, for example,

$$m_f(x) = \sum_{i=0}^{q} f(y^i) L_i(x). \tag{2.7}$$

The algorithm for maintaining well poisedness is inspired from Algorithms 6.2 and 6.3, p.95 in [13], and is presented in Figure 2.1. It will enable us to always recognize if System (2.1) is not well-poised. If that is the case instead of choosing new point to control the condition number of $M(\phi, Y)$, we modify the singular value decomposition (SVD) [20] of the appropriate matrix by replacing all values smaller than a threshold by the threshold. Note that only if we are unable to find a subset that is sufficiently well-poised is the SVD modified. The main difference with [13] is that no optimization is made on the whole ball $B(0; 1)$ in order to compute Λ-poisedness, but rather on a finite set of points. The reason for this is that we decided not to incur any new evaluations but insist on selecting the candidates from a finite set of previously evaluated points. Consequently the strategy remains reasonably simple, is not costly, and possibly improves the geometric properties of the interpolation sets, for the overdetermined case.

```
[0] initializations
        given $Y = \{y^0, y^1, \ldots, y^p\}$ a set of points in $\mathbb{R}^n$ such that $p > q$
        $\hat{Y} \leftarrow \emptyset$
        $L_i(x) = \phi_i(x), i = 0, 1, \ldots, q$
[1] for $i = 0$ to $q$
        [1.1] POINT SELECTION
                find $y^j \in \underset{y \in Y}{\operatorname{argmax}} |L_i(y)|$
                if $L_i(y^j) = 0$: stop (unable to construct a poised set)
                $\hat{Y} \leftarrow \hat{Y} \cup \{y^j\}$ , $Y \leftarrow Y \setminus \{y^j\}$
        [1.2] NORMALIZATION
                $L_i(x) \leftarrow L_i(x)/L_i(y^j)$
        [1.3] ORTHOGONALIZATION: for $j = 0$ to $q$ and $j \neq i$:
                $L_j(x) \leftarrow L_j(x) - L_j(y^j)L_i(x)$
[2] poisedness improvement: for $k = 0, 1, \ldots$
        find $\hat{\imath}$ and $y^j \in Y$ such that $|L_{\hat{\imath}}(y^j)| \geq |L_i(y)|, i \in \{0, \ldots, q\}, y \in Y$
        let $\hat{y}$ be the $\hat{\imath}$-th element of $\hat{Y}$
        $\hat{Y} \leftarrow \hat{Y} \cup \{y^j\} \setminus \hat{y}$ , $Y \leftarrow Y \setminus \{y^j\}$
        update all Lagrange polynomial coefficients as in steps [1.2] and [1.3]
        if poisedness is not improved, revert last swap and goto [3]
[3] terminate
        $Y \leftarrow \hat{Y}$
        $p \leftarrow q$
```

FIGURE 2.1. *The algorithm inspired from Algorithms 6.2 and 6.3 p.95 in [13] for the transformation of a set Y of $p + 1 > q + 1$ elements into a well-poised set of $p = q$ elements. The algorithm also constructs Lagrange polynomials allowing interpolation on the new set.*

2.3. Directional direct search methods and MADS. This class of methods is not based on models and uses search directions in order to generate trial points at which to evaluate the functions. These points usually lie on a spatial discretization called *the mesh* in order to ensure convergence, and simple decrease is used (an iterate is accepted as soon as it reduces the objective function). Different algorithms from this class are detailed in Chapter 7 of [13]. As the present work uses MADS, and as MADS generalizes most other directional direct searches, the remainder of this section gives a simplified description of this method.

MADS is designed specifically for difficult black box problems. In particular it is designed especially to deal with very nonsmooth problems and hidden constraints, situations for which model-based methods typically have difficulty with. Each trial point lies on the mesh

$$M_k \; = \; \{x + \Delta_k^m Dz \; : \; x \in V_k, \; z \in \mathbb{N}^{n_D}\} \subset \mathbb{R}^n \; ,$$

where $V_k \subset \mathbb{R}^n$, *the cache*, is the set of all evaluated points by the start of iteration k, $\Delta_k^m \in \mathbb{R}_+$ is the *mesh size parameter* at iteration k, and D is a fixed matrix in $\mathbb{R}^{n \times n_D}$ composed of n_D directions. D must satisfy some conditions but typically corresponds to $[I_n \;\; -I_n]$ with I_n the identity matrix of dimension n, and $n_D = 2n$.

Each iteration is made of three steps: the *search*, the *poll*, and the updates. The search is flexible and allows one to generate a finite number of trial points anywhere on the mesh. Search methods may be specific to a given problem or generic. The latter is proposed by this paper. The poll is more rigidly defined and consists of constructing directions used to generate trial points. The union of all the normalized directions generated in an infinite number of iterations must have the property of forming a dense set in the unit sphere. Different ways of doing so are detailed in the two implementations of [3, 7]. Trial points are generated at a distance from the current iterate bounded above by Δ_k^p, using the infinity norm. This distance is called the *poll size parameter* and is always larger than or equal to Δ_k^m.

Constraints handling is now described. It determines whether a point should be considered as a new incumbent and if the iteration is successful or not. Constraints are handled by the progressive barrier approach (described in [8]) which does not exploit derivative information of the constraints, even if they are available, and which applies only to inequalities. Equalities may be transformed into pairs of inequalities with a certain tolerance, but the present technique does not do so, and it is not evident that this is a desirable way to handle them. The progressive barrier is inspired by the filter method of [18] and makes use of the following *constraint violation function*:

$$h(x) \; = \; \begin{cases} \displaystyle\sum_{j \in J} (\max(g_j(x), 0))^2 & \text{if } x \in X, \\ \infty & \text{otherwise.} \end{cases} \tag{2.8}$$

The function h is nonnegative and $x \in \Omega$ if and only if $h(x) = 0$. Moreover, x is in $X \setminus \Omega$ if $0 < h(x) < \infty$. During the algorithm course, infeasible points are accepted or rejected depending on their f and h values and according to the following *dominance* relation between two points $x, y \in X$: x *dominates* y, denoted $x \prec y$, if and only if $h(x) \le h(y)$, $f(x) \le f(y)$, and $x \neq y$. The negation of the dominance relation is denoted with $x \not\prec y$ which means that x does not dominate y.

We can easily extend this relation in order to compare feasible and infeasible points of X, by stating that any point in Ω dominates any point in $X \setminus \Omega$. In addition, for two points x, y in X such that $x \not\prec y$ and $y \not\prec x$, one can use additional criteria in order to rank x and y. These criteria may include a directional measure with respect to a recent successful direction, the angular deviation from a gradient approximation, and the feasibility, based on the h value. These extensions to the dominance relation allow one to define an ordering property according to which a list of points possessing h and f values, or *approximations* to them, may be sorted. In the remainder of the paper, the notation $x \prec y$ implies, as in filter methods, that these additional criteria have been considered.

The last step of an iteration consists of checking the success of the iteration, choosing the next iterate, and updating the value of the mesh and poll size parameters. The iteration is declared a success when a new dominant point has been found. In this case, the poll and mesh sizes are possibly increased. Otherwise they must be decreased.

Figure 2.2 gives a simplified description of the method. Some additional notation is used in order to illustrate a possible way to update the mesh. The rational parameter τ is called the *mesh update basis* and is usually set to 4, and $\ell_k \in \mathbb{Z}$ is the *mesh index* at iteration k.

We conclude this section by mentioning that MADS possesses convergence properties based on the Clarke calculus [12]. These results ensure that under mild assumptions MADS converges globally to a point satisfying local optimality conditions. The interesting reader may consult references [7, 8] for most of the theoretical results.

```
[0] Initializations
    $x_0 \in \Omega$, $\tau \in \mathbb{Q}$, $\ell_0 \leftarrow 0$, $k \leftarrow 0$
[1] Iteration $k$
    $\Delta_k^m \leftarrow \min\{1, \tau^{-\ell_k}\}$ , $\Delta_k^p \leftarrow \tau^{-\ell_k/2}$
    [1.1] SEARCH (optional)
        evaluate $f$ and $h$ on a finite set $S_k \subset M_k$
    [1.2] POLL (optional if the SEARCH was successful)
        compute directions $D_k = \{d_1, d_2, \ldots, d_{|D_k|}\}$
          with $\|d_k\| \simeq \tau^{|\ell_k|/2} = \Delta_k^p / \Delta_k^m$ for all $d_k \in D_k$
        compute $P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k$
        evaluate $f$ and $h$ on $P_k$
[2] Updates
    if the iteration is successful (there exists $y \in S_k \cup P_k$ such that $y \prec x_k$)
        $x_{k+1} \leftarrow y$
        $\ell_{k+1} \leftarrow \ell_k - 1$
    else (iteration failure)
        $x_{k+1} \leftarrow x_k$
        $\ell_{k+1} \leftarrow \ell_k + 1$
    $k \leftarrow k + 1$
    goto [1] if no stopping condition is met
```

FIGURE 2.2. *A simplified version of the MADS algorithm.*

3. Integrating quadratic models into MADS. This section describes the choices made to integrate quadratic models into MADS. This is done at two different locations in the algorithm. First a search strategy called the *model search* is defined. It takes place within the step [1.1] of the algorithm in Figure 2.2 and generates up to four trial points. Second an ordering strategy called the *model ordering* is executed each time a list of trial points needs to be evaluated, which may occur twice in the algorithm, after the search step [1.1] and after the poll step [1.2]. This new formulation of the MADS algorithm using quadratic models is summarized by the algorithm in Figure 3.1 and detailed by Sections 3.1, 3.2 and 3.3.

3.1. Models construction. Quadratic models are used as *local models*: they are typically trusted only in a certain neighborhood, although in practice that neighborhood could, especially in non-asymptotic iterations, be rather large. This is in contrast to *global models*, used for example in the Surrogate Management Framework of [11] or in [10, 29]. We first describe the types of models that are possibly constructed for the objective function and the constraints.

As we consider problems with quantifiable inequality constraints $g_j(x) \leq 0$ with $j \in J$, every time a model is needed, $|J| + 1 = m + 1$ models are constructed in practice: one for each constraint $g_j \leq 0, j \in J$ and one for the objective f. These models are denoted m_f and $m_{g_j}, j \in J$, and we hope that, eventually at least, in a certain region B we have

$$m_f(x) \simeq f(x) \text{ and } m_{g_j}(x) \simeq g_j(x), j \in J, \text{ for all } x \in B.$$

```
[0] Initializations
[1] Iteration $k$
    [1.1] SEARCH
        $S_k \leftarrow \emptyset$
        MODEL SEARCH (Section 3.2)
            select two model centers (feasible and infeasible incumbents)
            for each center
                construct interpolation set $Y$ and scale it to $Y_s$
                construct $m+1$ models ($m_f$ and $m_{g_j}, j \in J$)
                minimize $m_f(x)$
                   subject to $m_{g_j}(x) \leq 0, j \in J$, and $x \in B_\infty(0;1)$
                select at most two trial points in $M_k$ to add in $S_k$
        if other optional search generated other points in $S_k$, apply
           MODEL ORDERING (Section 3.3):
            construct $m+1$ models
            evaluate the models on $S_k$
            sort points of $S_k$ according to rules defined in 2.3
        evaluate opportunistically $f$ and $h$ on $S_k$
    [1.2] POLL (optional if the SEARCH was successful)
        compute $P_k$ and apply MODEL ORDERING (Section 3.3):
            construct $m+1$ models
            evaluate the models on $P_k$
            sort points of $P_k$
        evaluate opportunistically $f$ and $h$ on $P_k$
[2] Updates
```

FIGURE 3.1. *High level description of the MADS algorithm with models. Only the differences with the algorithm from Figure 2.2 are reported.*

This has been preferred to the construction of two models (one for f and one for h) for obvious reasons, including scaling, and because some constraints may be smoother than others. In all that follows, the infinity norm is used and we define $B_\infty(c;r) = \{x \in \mathbb{R}^n : \|x - c\|_\infty \leq r\}$, the closed box with center c and radius r, and $N = \{1, 2, \ldots, n\}$. The choice of the infinity norm has been made because in the presence of bounds and considering that the poll trial points in MADS are constructed within the ball $B_\infty(x_k; \Delta_k^p)$, defined with that norm, it is the natural choice.

The first step necessary to obtain the models is to construct the set $Y = \{y_0, y_1, \ldots, y_p\}$ of $p+1$ interpolation points in $\mathbb{R}^n$. The minimal and maximal size of Y are left as parameters. Model search and model ordering have each a different way of defining a region where the interpolation points must lie, and this will be justified in the next sections. The construction of Y requires the inspection of the cache, the set of already evaluated points, and consider the points inside the region of interest as the only entry candidates.

After the set Y has been constructed, a scaling is performed that transforms the set Y into Y_s. The points are contained in the box $B_\infty(0;\rho)$, for some $\rho > 0$. Scaling details are given in Figure 3.2.

Now, if the number of points inside Y is strictly greater than $q+1 = (n+1)(n+2)/2$, the algorithm presented in Figure 2.1 may be used in order to reduce Y to $q+1$ points with a better geometry. Whether we use or not this strategy is left as a parameter. Notice that in the case where a better geometry is sought, no new function evaluations are required, and after the algorithm for well poisedness terminates there is no need to solve System 2.1 as the associated Lagrange polynomials are available. The desired models are simply determined by computing equation (2.7) and their analogues $m+1$ times.

If the strategy to maintain well poisedness is not enabled, or if this strategy determined that Y is not poised, or simply if $p+1 \leq (n+1)(n+2)/2$, some system resolution will be performed. In particular, Minimum Frobenius Norm (MFN) models are chosen if $p+1 < (n+1)(n+2)/2$, and regression models are used otherwise. The exact interpolation case with $p+1 = (n+1)(n+2)/2$ is also treated by the regression technique and if the set is poised, the optimal value of Problem (2.2) will be zero. Each time a system needs to be solved, a singular value decomposition (SVD) is found for $M(\phi, Y)$ in order to decompose the matrix $F(\phi, Y)$ for MFN interpolation, or $M(\phi, Y)$ for regression or exact interpolation. This decomposition has to be done only once for the $m+1$ models. Systems (2.5) and (2.6) (MFN), or System (2.3), are then solved $m+1$ times in order to obtain the sets of α coefficients for the $m+1$ required models. The advantage of using the SVD decomposition is its numerical stability. Moreover, the problems are sufficiently small that the computational overhead is completely acceptable.

```
Scaling definition, given a set $Y = \{y^0, y^2, \dots, y^p\}$ of points in $\mathbb{R}^n$ and $\rho > 0$,
this algorithm produces $c^Y$ and $s$ in $\mathbb{R}^n$:
    [1] Compute $B_\infty(c^Y; r^Y)$ the smallest right-oriented[a] box around $Y$
        with center $c^Y$ and radius $r^Y$. Each coordinate $i$ of $r^Y$ is equal to
        $r_i^Y = \max_{j=0,\dots,p} |y_i^j - c_i^Y|$.
    [2] Define the scaling vector $s = r^Y/\rho \in \mathbb{R}^n$.

Scaling of a point $x \in \mathbb{R}^n$:
    [1] Set $x \leftarrow x - c^Y$.
    [2] For all coordinates $i \in N$, set $x_i \leftarrow x_i/s_i$ or 0 if $s_i = 0$.

Unscaling of a point $x \in \mathbb{R}^n$:
    [1] For all coordinates $i \in N$, set $x_i \leftarrow x_i s_i$.
    [2] Set $x \leftarrow x + c^Y$.
```

[a]i.e. oriented with the coordinate axes.

FIGURE 3.2. *Steps used to define a scaling from $B_\infty(c^Y; r^Y)$ into $B_\infty(0; \rho)$. The scaling and unscaling operations for a point are also described.*

3.2. Search step based on models (model search). We now explain the model search which is performed at the step [1.1] of the algorithm of Figure 2.2, prior to the poll step. If this search is successful, the poll is not performed. Additional searches may also be considered. For example a speculative search, or optimistic strategy, which is performed after a successful poll and which consists of generating one further trial point along the previously successful direction.

At each iteration k, up to two model centers at which the functions defining the problem have already been evaluated are selected. The first center x^f corresponds to a feasible point with the best objective value while the second x^i is an infeasible point which is not dominated by any other infeasible point and which has the smallest objective value. For each of these centers, if they exist, $m+1$ models are going to be constructed.

Consider one of these two centers and denote it by c. The objective is to construct models, the most accurately possible, within the current poll size Δ_k^p. For that, an *interpolation radius*, $\rho\Delta_k^p$, is determined, where ρ is the *radius factor*. The interpolation set Y is constructed and scaled to Y_s as described in Section 3.1 and by considering all the points from the cache inside $B_\infty(c; \rho\Delta_k^p)$. Note that the smallest box containing points of Y does not necessarily have a radius equal to $\rho\Delta_k^p$. This implies that the box $B_\infty(0; 1)$, once unscaled, does not necessarily contain $B_\infty(c; \Delta_k^p)$.

Having computed the $m+1$ Lagrange polynomials, or MFN, or regression models, on the scaled points, we consider that we can trust these models in $B_\infty(0; 1)$. Then the optimization of the model of the objective function, subject to the m model constraints, is performed within this box (this is the trust region sub-problem). An exact resolution, or even a local minimizer, is not necessary as the convergence of the whole method in not related to this step. A fast resolution is not crucial either. Because of the black box and costly function evaluation context, the time taken by the model optimization is negligible. For our tests, MADS was chosen for convenience, although this is clearly far from an ideal choice.

As we already mentioned, after the model optimization, up to two model ‘solutions’ are considered: the feasible solution with the lowest objective, and the undominated infeasible point with the lowest objective. When they exist, these points are first unscaled, and then projected (i.e. the closest, in the l_∞ sense, mesh point is taken) to the current mesh of size, Δ_k^m, in order to respect the convergence requirements of MADS. These two points are denoted x_m^f and x_m^i in the algorithm of Figure 3.1. The direction between the center c and each candidate is retained, allowing an extended speculative search after model search successes.

These different steps lead to the set S_k^m containing up to four trial points lying on the current mesh: four because each optimization gives up to two points and we recall that two optimizations are performed (one for each center). These points are going to be ordered according to a preference scheme described in Section 3.3 that follows. Finally, the trial points are (optionally) evaluated using the opportunistic strategy and the success of the model search step is checked.

3.3. Models used to sort trial points before evaluations (model ordering). We now present the second strategy to integrate models within MADS, namely the model ordering strategy, performed any time a list of at least two trial points has to be evaluated. This list may have been generated during the poll step or during the search step. If the search step consists only of the model search (i.e no Latin hypercube [28] or Variable Neighborhood Search (VNS) [5, 25], then the ordering is disabled, because the model search candidates have already been sorted according to model values. For the model ordering, no new function evaluations are required as the method consists of sorting the list of trial points prior to any evaluations. This exploits the fact that the evaluations are opportunistic, meaning that they are interrupted as soon as a new success is obtained. If there is no success, all evaluations are performed and the model ordering has no impact. This improves the chances of obtaining a success earlier since the most promising points are evaluated first. Such an approach has been successfully used with simplex gradients in [14, 15, 16]. The model ordering strategy uses the same quadratic models as for the model search.

An interpolation set Y is created in the same way as for the search step in the previous section except that Δ_k^p is not considered. Instead, the smallest box $B_\infty(c; r)$ of radius r containing the trial points to order is used. The point c is the center of this box, where the true functions have not necessarily been evaluated. The interpolation points are taken in $B_\infty(c; \rho r)$ with $\rho > 0$ the same radius factor parameter used for the model search. It is possible that less than $n + 1$ points are gathered in Y, as the trial points have not yet been evaluated. In this case, no model is computed and no ordering is made. Otherwise scaling is performed as before but with ρr instead of $\rho\Delta_k^p$. The $m + 1$ models are constructed and evaluated at all the trial points, and the h function (see equation 2.8) is computed from the m constraint models. Then the trial points are sorted following the order defined in Section 2.3 based on the f and h approximations.

We conclude this section by highlighting the fact that the model search and the model ordering satisfy the convergence requirements of the MADS algorithm: the order in which a list of trial points is evaluated has no impact on the theory and the model search generates a finite number of mesh trial points. Hence the convergence properties of MADS described in [7, 8] are kept. In the numerical experiments of the next section, the impact of keeping points in the mesh will be studied along with some other considerations.

4. Numerical results.

The integration of quadratic models into the MADS framework is tested in this section. Our objective is to demonstrate that the addition of models greatly improves the MADS efficiency. The NOMAD [2, 23] software which implements MADS with the addition of models is used.

4.1. A special case. We first highlight the particular case of the (badly scaled and nonsmooth) Problem DIFF2 which consists of minimizing

$$f(x,y) = |x-y| - 10^{-6}(x+y) \tag{4.1}$$

with $(x,y) \in [-100;100]^2$ and optimal solution $(x^*,y^*) = (100,100)$ with $f(x^*,y^*) = -2 \cdot 10^{-4}$. The function f, drawn in Figure 4.1, is Lipschitz and convex and the MADS theory predicts global convergence to (x^*,y^*). However, in practice, MADS is unable to improve **any starting point**. This is due to the fact that numerically finding a descent direction is difficult (because of the poor scaling, exacerbated by the nonsmoothness) even with the dense sets of directions used by MADS. Thus in practice MADS is unable to find a descent direction. By contrast, our models allow one to easily find a descent direction and typically determine the solution within approximatively 90 function evaluations.

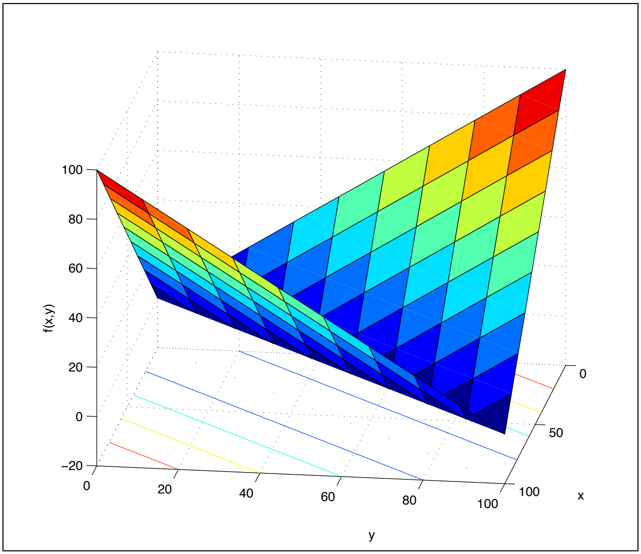

FIGURE 4.1. *Representation of the function $f(x,y) = |x-y| - 10^{-6}(x+y)$ corresponding to Problem DIFF2.*

4.2. Two sets of test problems. After this first particular case, two sets of test problems are considered. The first set contains a variety of problems, namely unconstrained and smooth problems from the CUTEr set [21], unconstrained and nonsmooth problems from [24], smooth and nonsmooth constrained problems from [8, 9, 24], and smooth and two nonsmooth real engineering applications described in [5]. This set corresponds to the one used in [3]. Additional tests from [22] are also considered, as well as the DIFF2 instance described by (4.1), for a total of 48 problems, whose characteristics are summarized by Table 4.1. For this set, only MADS with and without models is tested for this is the only one which handles general constraints (both theoretically and practically with the progressive barrier).

In order to compare with other methods from the literature, a second set of unconstrained problems is considered, for which some results are already available with the NEWUOA [27] and SID-PSM [15] algorithms. NEWUOA is a trust-region interpolation-based algorithm while SID-PSM has already been described at the end of the introduction. These two methods have been benchmarked as state-of-the-art in [15] and [26] for this type of problems but we note that such problems are not the primary target problems for MADS. In these papers, a basis of 22 smooth unconstrained problems from the CUTEr set is used with different starting points in order to produce 53 instances. These instances are modified to give new nonsmooth and noisy problems. Here we consider only the smooth and nonsmooth instances without noise: Problems with stochastic noise have been discarded as they cannot be reproduced with C++ to match

#	Name	Source	n	m	Bnds	Smth	f^*
1	ARWHEAD	[21]	10	0	no	yes	0.0
2	ARWHEAD	[21]	20	0	no	yes	0.0
3	BDQRTIC	[21]	10	0	no	yes	18.2812
4	BDQRTIC	[21]	20	0	no	yes	58.3204
5	BIGGS6	[21]	6	0	no	yes	$6.97074 \cdot 10^{-5}$
6	BRANIN	[22]	2	0	yes	yes	0.397887
7	BROWNAL	[21]	10	0	no	yes	0.0
8	BROWNAL	[21]	20	0	no	yes	0.0
9	CRESCENT10	[8]	10	2	no	yes	-9.0
10	DIFF2	(4.1)	2	0	yes	no	$2 \cdot 10^{-4}$
11	DISK10	[8]	10	1	no	yes	-17.3205
12	ELATTAR	[24]	6	0	no	no	0.561139
13	EVD61	[24]	6	0	no	no	$3.51212 \cdot 10^{-2}$
14	FILTER	[24]	9	0	no	no	$8.40648 \cdot 10^{-3}$
15	G2	[9]	10	2	yes	no	-0.740466
16	G2	[9]	20	2	yes	no	-0.803619
17	GOFFIN	[24]	50	0	no	no	0.0
18	GRIEWANK	[22]	10	0	yes	yes	0.0
19	HS78	[24]	5	0	no	no	-2.49111
20	HS114	[24]	9	6	yes	no	-1429.34
21	L1HILB	[24]	50	0	no	no	0.392475
22	MAD6	[24]	5	7	no	no	0.101831
23	MDO	[5]	10	10	yes	no	-3964.2
24	MXHILB	[24]	50	0	no	no	$4.27416 \cdot 10^{-2}$
25	OSBORNE2	[24]	11	0	no	no	$9.43876 \cdot 10^{-2}$
26	PBC1	[24]	5	0	no	no	$8.90604 \cdot 10^{-2}$
27	PENALTY1	[21]	10	0	no	yes	$7.08765 \cdot 10^{-5}$
28	PENALTY1	[21]	20	0	no	yes	$1.57784 \cdot 10^{-4}$
29	PENALTY2	[21]	10	0	no	yes	$2.95665 \cdot 10^{-4}$
30	PENALTY2	[21]	20	0	no	yes	$6.38969 \cdot 10^{-3}$
31	PENTAGON	[24]	6	15	no	no	-1.85962
32	POLAK2	[24]	10	0	no	no	54.5982
33	POWELLSG	[21]	12	0	no	yes	0.0
34	POWELLSG	[21]	20	0	no	yes	0.0
35	RASTRIGIN	[22]	2	0	yes	yes	0.0
36	SHOR	[24]	5	0	no	no	22.6023
37	SNAKE	[8]	2	2	no	yes	0.0
38	SROSENBR	[21]	10	0	no	yes	0.0
39	SROSENBR	[21]	20	0	no	yes	0.0
40	STYRENE	[5]	8	11	yes	no	-33539100
41	TRIDIA	[21]	10	0	no	yes	0.0
42	TRIDIA	[21]	20	0	no	yes	0.0
43	VARDIM	[21]	10	0	no	yes	0.0
44	VARDIM	[21]	20	0	no	yes	0.0
45	WONG1	[24]	7	0	no	no	680.707
46	WONG2	[24]	10	0	no	no	24.9458
47	WOODS	[21]	12	0	no	yes	0.0
48	WOODS	[21]	20	0	no	yes	0.0

TABLE 4.1

Overview of the first set of test problems. Those for which $m > 0$ have constraints other than bounds. The column 'Bnd' indicates if a problem has bound constraints, the column 'Smth' if the problem is smooth, and the column 'f^*' indicates the best known solution.

the original MATLAB data, as have been the problems with deterministic noise. The reason for the exclusion of the remaining noisy problems is that we noticed that the results for all the methods were too similar to the smooth versions of the problems. We also excluded one problem in particular (BROWN) as it appeared that this was not the same version that was used in [15] and in [26]. The second set of (unconstrained) problems on which we compare NOMAD with NEWUOA and SID-PSM is then made of 104 problems (52 smooth, 52 nonsmooth).

4.3. Results. We first show in Figure 4.2 some data on the size of the interpolation set Y. These statistics have been gathered using the new method on the 48 problems from the first set, with all parameters set to the default values and without the strategy to maintain well poisedness. These default values will be given later on. The aim of this figure is to illustrate that more and more MFN interpolations are used, compared to regressions, when the sizes of the problems become larger. For the problems with $n = 50$ variables, almost only MFN models are used. In the two subfigures, the x-axis corresponds to n the dimension of the problems and one mark corresponds to one execution. For the left subfigure, the size $p+1$ of the interpolation set Y is represented together with the curve $q+1 = (n+1)(n+2)/2$ which represents the dimension of the exact interpolation case and whose value decides if MFN interpolation is used instead of regression. In the right subfigure, the ratio of MFN models over the total number of models is showed.

For all the following results and all the remaining figures of this document, the limit on the number of combined function and constraint evaluations has been set to 1500 as in [15]. Figures correspond to data profiles [26] and they may be read as follows: the x-axis corresponds to the number of evaluations and the y-axis to the proportion of the number of problems "solved" in the sense of the next sentence. Each figure shows one plot for one method and each plot indicates the proportion of problems for which the following measure is satisfied for a given number of evaluations: $f_0 - f_{cur} \geq (1-\tau)(f_0 - f^*)$, where f_0 is the objective value at the starting point, f_{cur} is the objective value after this number of evaluations, and f^* is the best known value indicated by the last column of Table 4.1. The reasonable level of accuracy of $\tau = 10^{-3}$ has been chosen.

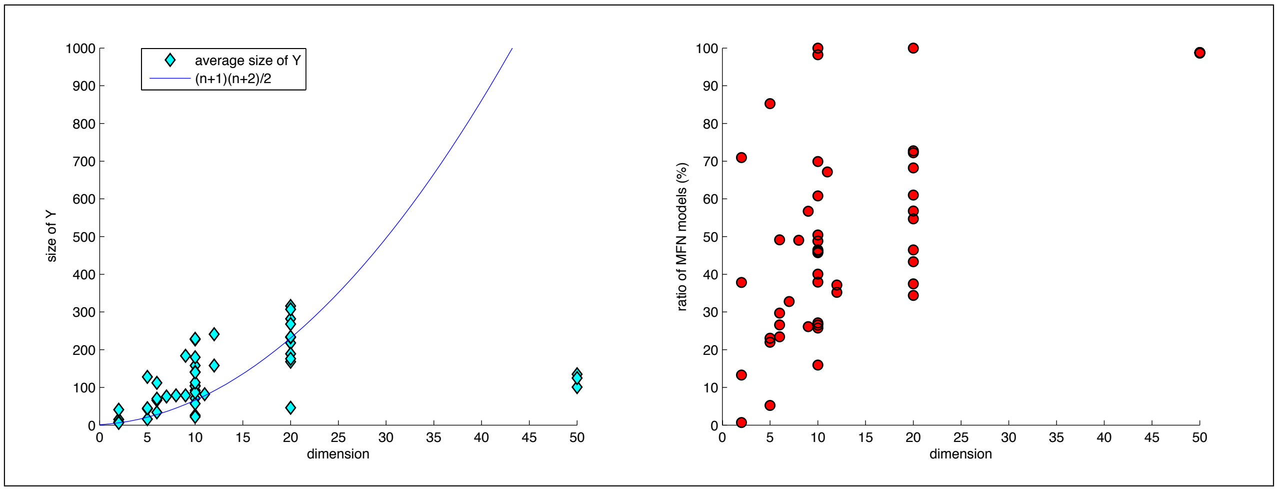

FIGURE 4.2. *Model statistics for the first set of problems. On the two subfigures, each mark corresponds to one of 48 executions with MADS with models and the default parameter values. Note that these are dependent on the dimension, n. The left subfigure shows the average size of the interpolation set Y and the right one indicates the proportion of MFN models over the total number of models.*

For the first set of problems, we focus on the impact of the main parameters of the MADS algorithm augmented with models. The first data profiles are given by Figure 4.3 and measure the effect of maintaining or not well poisedness (WP) for the regression models, using the algorithm of Figure 2.1. For this test, the set of problems has been split into small and big problems, according to the conclusion drawn by Figure 4.2. For small problems (left subfigure), the size of the interpolation set Y is often considerably larger than $q+1 = (n+1)(n+2)/2$, meaning that the strategy to maintain WP is also often used, reducing drastically these sets to $q+1$ points. This situation appears to deteriorate efficiency after $\simeq$800 evaluations and have almost no effect otherwise, meaning that in this context, it is better to use as many points as possible without any special geometry considerations. This is suggested by the theory in [13], since the poisedness cannot be worse than the best poised linear (or quadratic) subset. This important observation seems to confirm that in a direct search context, the geometry of interpolation sets is implicitly good. For larger problems, however, having large interpolation sets become rarer, as is the use of the WP strategy, and the sets are not pruned too severely. Thus using or not WP, at least in the sense that we use it here, seems to have no impact at all for less than 1500 evaluations. In the remainder of the paper, WP is not used.

The impact of the radius factor (ρ) is indicated in Figure 4.4, which suggests that having an interpolation radius larger than the model (trust-region) radius ($\rho > 1.0$) is better. Our default is $\rho = 2.0$.

The next series of tests displays the influence of the maximum number of trial points that we allow to be evaluated in the search step. This number may be 0 (no model search), 1, 2, 3, or 4. For these tests, given in Figure 4.5, the model ordering strategy has been disabled for the poll points but the search points are always ordered according to their model values. Results are very similar for all the model search settings. The default of 4 points has been chosen because more intensive tests (that are not reported here) with larger budgets of evaluations suggested it was slightly better.

We now show the importance of projecting to the mesh the candidates that the model optimizations produced within the model search. We recall that the convergence analysis requires this projection. This impact is given in Figure 4.6 and the results show that, in practice, not projecting to the mesh may be slightly faster for the first 1000 evaluations, but after that, projecting to the mesh is more efficient. We conclude that there is no loss due to the projection, and that the slight observed improvement is most likely because of improving poisedness.

The next results are given by Figure 4.7 which shows the influence of using only one out of the two new strategies

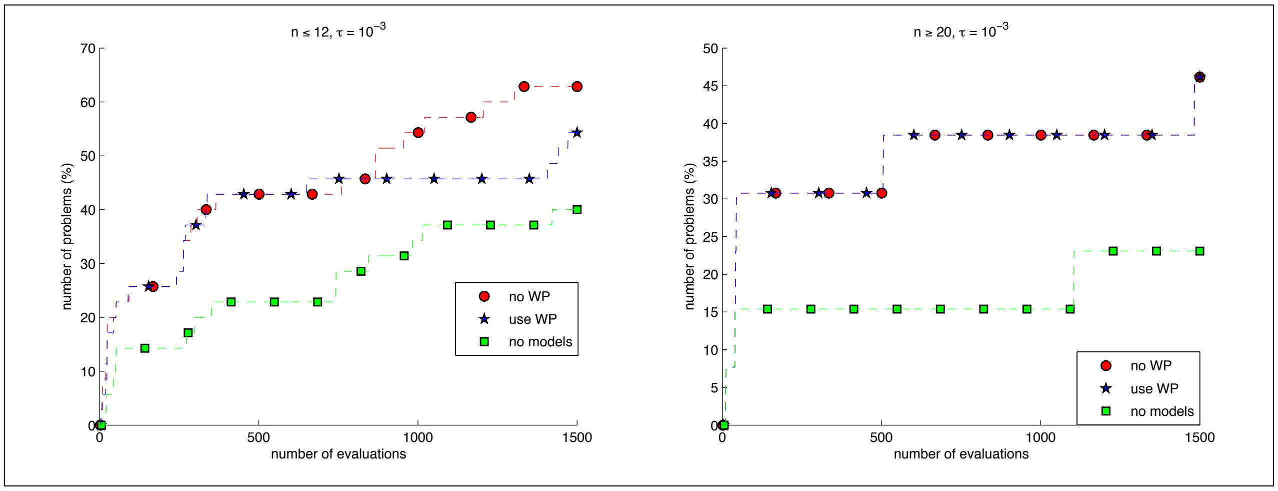

FIGURE 4.3. *Impact of maintaining or not well poisedness (WP) for the first set of problems. The left subfigure is for the 34 'small' problems with $n \leq 12$ and the right subfigure is for the 14 'big' problems with $n \geq 20$.*

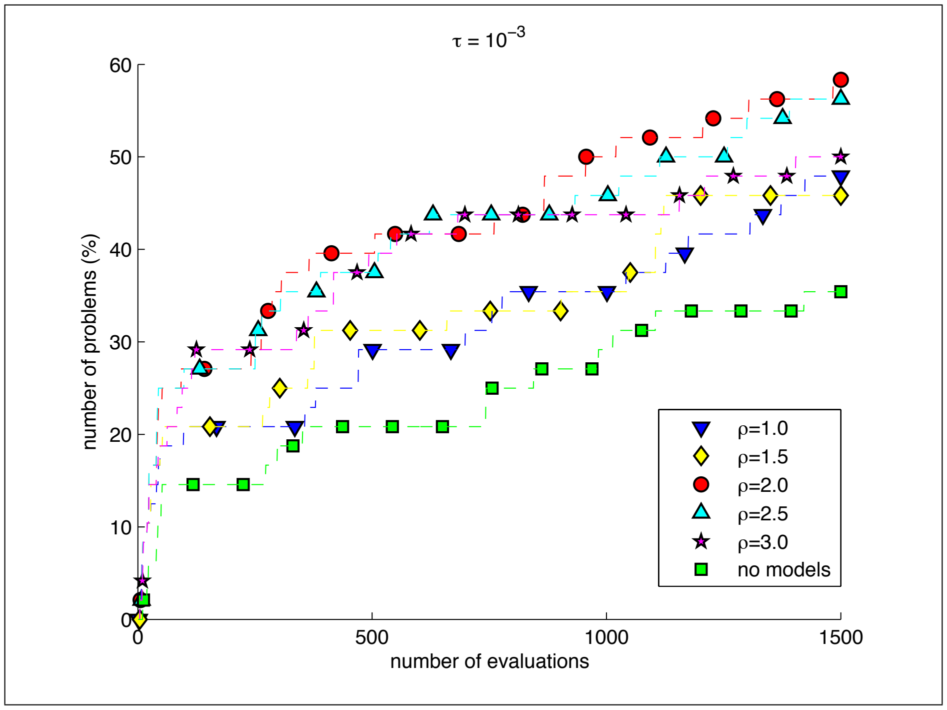

FIGURE 4.4. *Radius factor (ρ) impact for the first set of problems. Model search and model ordering are enabled when models are used.*

(model search and model ordering). In both cases the use of models improves MADS, but model search alone is more efficient than model ordering alone, and the latter is only slightly better than neither strategy. Model ordering in addition to model search is better than only one strategy.

Finally Figure 4.8 summarizes all the results on the first set of problems by simply comparing the MADS algorithm with and without the models. The version with the models uses both the model search and the model ordering strategies, a radius factor ρ set to 2, no maintenance of well poisedness, and at most 4 trial points which are projected to the mesh for the model search. These parameters are defined as the default values for the current NOMAD release 3.5.0. The figure shows very clearly the improvement due to the introduction of models.

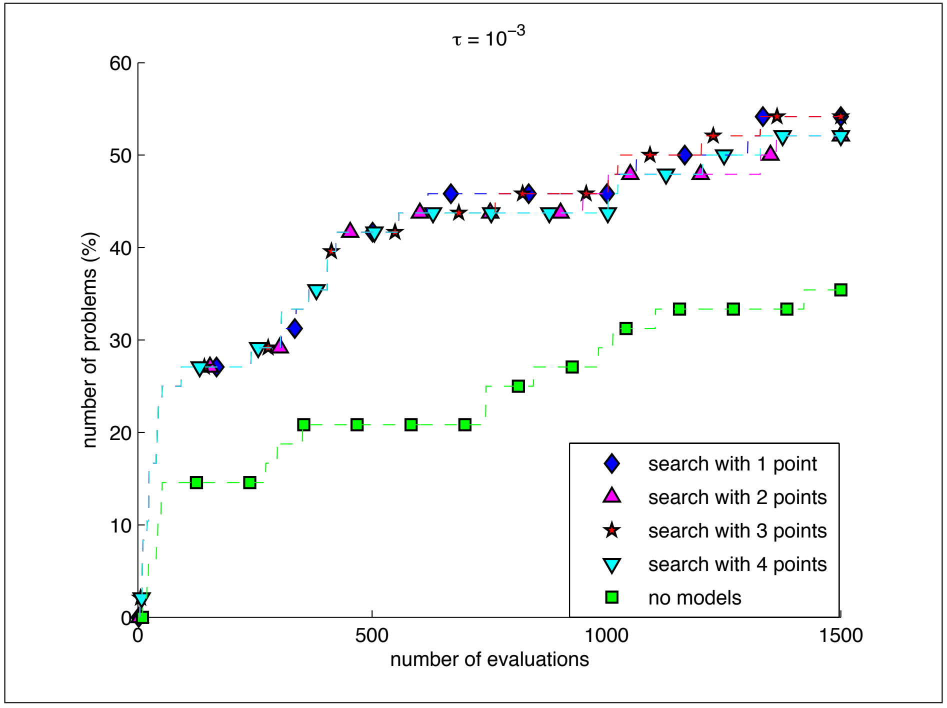

FIGURE 4.5. *Impact of the maximum number of trial points allowed in the model search (model ordering is off), for the first set of problems.*

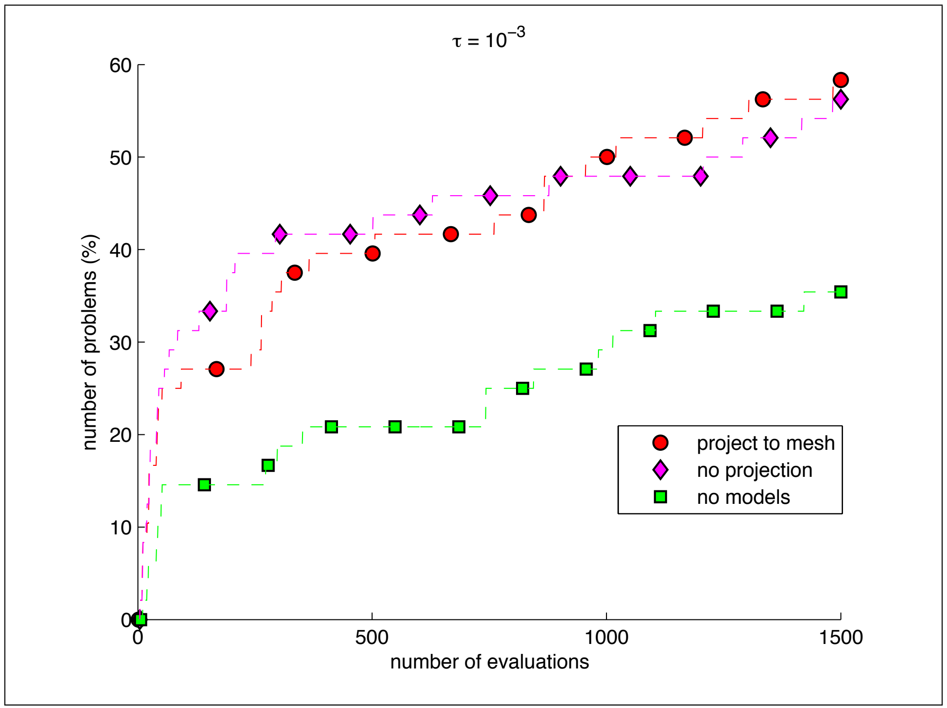

FIGURE 4.6. *Impact of projecting model search points to the mesh or not, for the first set of problems.*

We now focus on the second set of 104 problems for the comparison with the NEWUOA and SID-PSM methods. We emphasize to the reader that this set of problems is not the type of problems that NOMAD and MADS are primarily targeted for since these problems are academic, unconstrained, and without hidden constraints. These results are however shown for comparison purpose, and unconstrained problems are the only ones that can be used since NEWUOA and

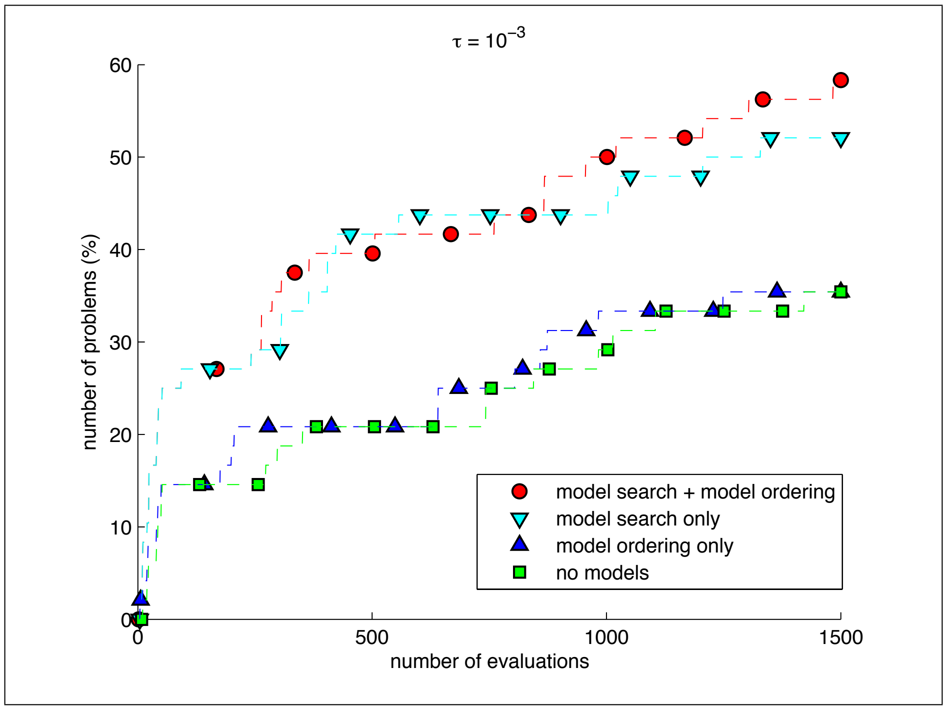

FIGURE 4.7. *Influence of using only model search or only model ordering, for the first set of problems.*

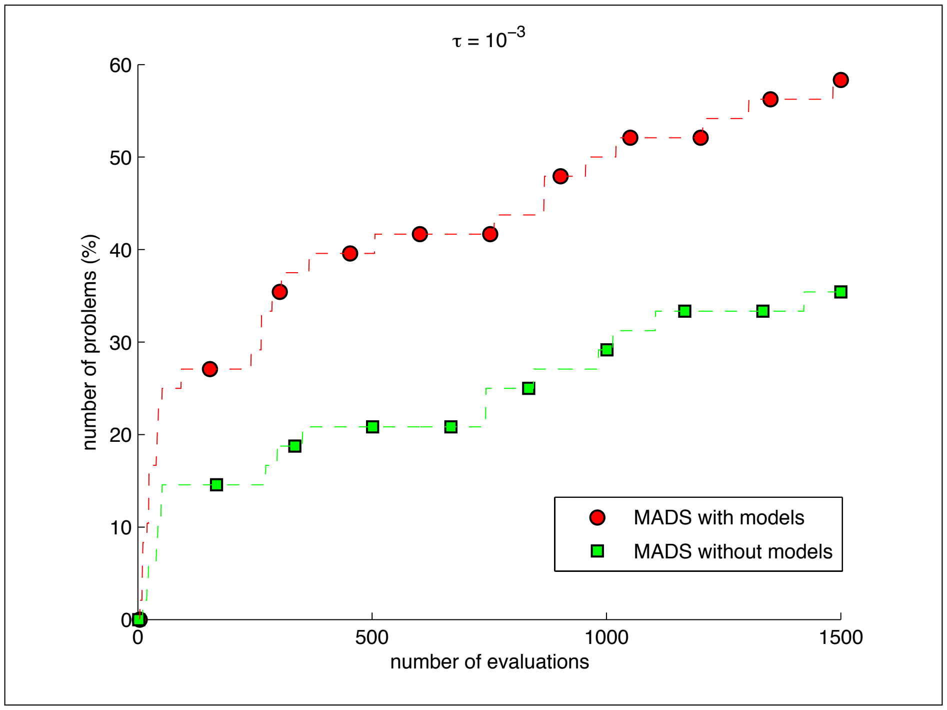

FIGURE 4.8. *MADS without and with models for the first set of problems.*

SID-PSM do not handle general constraints (However SID-PSM handles constraints for which derivatives are available, which is not yet the case for NOMAD). Figure 4.9 shows the data profiles in the smooth and nonsmooth cases. As for the first set of problems, the impact of MADS plus models versus MADS alone is obvious. In the smooth case, NEWUOA and SID-PSM are dominant and faster, although NOMAD gives more or less the same quality of results as SID-PSM after around 1200 evaluations. In the nonsmooth case, NEWUOA and SID-PSM are still faster at the beginning, but NOMAD dominates NEWUOA after a while and eventually reaches the same quality as SID-PSM.

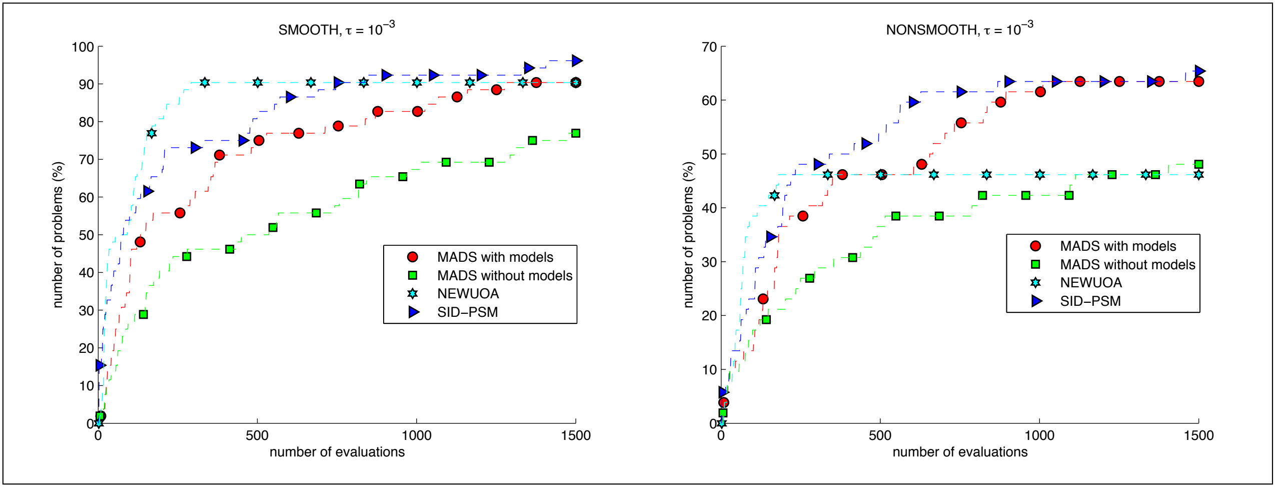

FIGURE 4.9. *Comparison with the NEWUOA and SID-PSM methods on the second set of unconstrained problems.*

Discussion. In this work, black box constrained optimization is considered. The quadratic models usually employed by model-based methods have been integrated into a directional direct search framework. This framework, via its search and poll setting, is flexible enough to allow such coupling. In particular the MADS direct search algorithm has been chosen for its ability to generate dense sets of directions, its convergence theory and its ability to practically and theoretically handle constraints. This method still belongs to the MADS class of algorithms and inherits its convergence results based on the Clarke calculus. In addition it remains a convenient algorithm to parallelize, contrary to pure model-based methods, and it is also relatively easily adapted to integer and categorical variables (see [1]).

The main contribution of this work is to show that clear improvements due to the introduction of models into MADS have been demonstrated by numerical tests: First on a set of 48 problems including difficult constrained problems, and then on a set of 104 unconstrained and easier problems for which the method has not been originally developed, but for which results with other algorithms were available. This new version is publicly available on the NOMAD website [2] and test data and results are available upon request.

Future work includes the use of models within the parallel framework of [9] and with integer and categorical variables. Improvement of the present implementation is also possible, for example by using a more adapted method for model optimization, which is currently performed entirely by MADS, and taking advantage of derivatives when they are available. We plan to investigate further well poisedness in the underdetermined case.

Acknowledgments. We would like to thank Ana Custódio and Stefan Wild for having made their test problems and results available. We also thank Charles Audet, Ana again, John Dennis, and Luís Vicente, for their useful comments.

REFERENCES

[1] M.A. Abramson, C. Audet, J.W. Chrissis, and J.G. Walston. Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3(1):35–47, 2009.

[2] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, Jr., and S. Le Digabel. The NOMAD project. Software available at `http://www.gerad.ca/nomad`.

[3] M.A. Abramson, C. Audet, J.E. Dennis, Jr., and S. Le Digabel. OrthoMADS: A deterministic MADS instance with orthogonal directions. *SIAM Journal on Optimization*, 20(2):948–966, 2009.

[4] L.T.H. An, A.I.F. Vaz, and L.N. Vicente. Optimizing radial basis functions by D.C. programming and its use in direct search for global derivative-free optimization. Technical Report Preprint 09-37, Department of Mathematics, University of Coimbra, 2009. Paper available at http://www.mat.uc.pt/~lnv/papers/rbf-dc.pdf.

[5] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2):299–318, 2008.

[6] C. Audet, A.L. Custódio, and J.E. Dennis, Jr. Erratum: Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 18(4):1501–1503, 2008.

[7] C. Audet and J.E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.

[8] C. Audet and J.E. Dennis, Jr. A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization*, 20(4):445–472, 2009.

[9] C. Audet, J.E. Dennis, Jr., and S. Le Digabel. Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM Journal on Optimization*, 19(3):1150–1170, 2008.

[10] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, and V. Torczon. Optimization using surrogate objectives on a helicopter test example. In J. Borggaard, J. Burns, E. Cliff, and S. Schreck, editors, *Optimal Design and Control*, Progress in Systems and Control Theory, pages 49–58, Cambridge, Massachusetts, 1998. Birkhäuser.

[11] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17(1):1–13, 1999.

[12] F.H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[13] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. MPS/SIAM Book Series on Optimization. SIAM, Philadelphia, 2009.

[14] A.L. Custódio, J.E. Dennis, Jr., and L.N. Vicente. Using simplex gradients of nonsmooth functions in direct search methods. *IMA Journal of Numerical Analysis*, 28(4):770–784, 2008.

[15] A.L. Custódio, H. Rocha, and L.N. Vicente. Incorporating minimum Frobenius norm models in direct search. *Computational Optimization and Applications*, 46(2):265–278, 2010.

[16] A.L. Custódio and L.N. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18(2):537–555, 2007.

[17] G. Fasano, J.L. Morales, and J. Nocedal. On the geometry phase in model-based algorithms for derivative-free optimization. *Optimization Methods and Software*, 24(1):145–154, 2009.

[18] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, Series A, 91:239–269, 2002.

[19] P. Gilmore and C.T. Kelley. An implicit filtering algorithm for optimization of functions with many local minima. *SIAM Journal on Optimization*, 5(2):269–285, 1995.

[20] G.H. Golub and C.F. Van Loan. *Matrix Computations*, chapter 2.5.3 The Singular Value Decomposition, pages 70–71. The John Hopkins University Press, Baltimore and London, third edition, 1996.

[21] N.I.M. Gould, D. Orban, and Ph.L. Toint. CUTEr (and SifDec): a constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[22] A. Hedar. Global optimization test problems. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

[23] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm, 2010. To appear in ACM Transactions on Mathematical Software.

[24] L. Lukšan and J. Vlček. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report V-798, ICS AS CR, 2000.

[25] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[26] J.J. Moré and S.M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009. Test problems and results available at http://www.mcs.anl.gov/~more/dfo/.

[27] M.J.D. Powell. The NEWUOA software for unconstrained optimization without derivatives. In P. Pardalos, G. Pillo, and M. Roma, editors, *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, pages 255–297. Springer US, 2006.

[28] T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments*, chapter 5.2.2, Designs Generated by Latin Hypercube Sampling, pages 127–132. Springer Verlag, 2003.

[29] D.B. Serafini. *A Framework for Managing Models in Nonlinear Optimization of Computationally Expensive Functions*. PhD thesis, Department of Mathematical Sciences, Rice University, Houston, Texas, 1998.

[30] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[31] S.M. Wild, R.G. Regis, and C.A. Shoemaker. ORBIT: optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219, 2008.