BOOSTERS: A Derivative-Free Algorithm Based on Radial Basis Functions

R. OEUVRAY *
Rodrigue.Oeuvay@gmail.com

M. BIERLAIRE †
Michel.Bierlaire@epfl.ch

December 21, 2005

Abstract
Derivative-free optimization involves the methods used to minimize an expensive objective function when its derivatives are not available. We present here a trust-region algorithm based on Radial Basis Functions (RBFs). The main originality of our approach is the use of RBFs to build the trust-region models and our management of the interpolation points based on Newton fundamental polynomials. Moreover the complexity of our method is very attractive. We have tested the algorithm against the best state-of-the-art methods (UOBYQA, NEWUOA, DFO). The tests on the problems from the CUTEr collection show that BOOSTERS is performing very well on medium-size problems. Moreover, it is able to solve problems of dimension 200, which is considered very large in derivative-free optimization.

Key Words: Derivative-Free Optimization, Trust-Region Methods, Radial Basis Functions, CUTEr

1 Introduction

We consider the following problem:

$$\min_{x \in \mathbb{R}^n} f(x),$$

where f is a nonlinear smooth function that is expensive to compute in the sense that it requires significant computational time to evaluate the function. We also assume that the derivatives are not available. This is the case when $f(x)$ is the result of some measurements or of complex computer simulation, for which the source code is not available. We can classify derivative-free methods in two categories: *direct search methods*

*This work was supported by the Swiss National Science Foundation grant 205320-103657
†Operations Research Group ROSO, Ecole Polytechnique Fédérale de Lausanne, Switzerland

and *model-based methods*. Direct search methods explore the domain using systematic rules. Following the authors of [1], we classify them into three categories: pattern search methods (see [2]), simplex search methods (see for example [3]) and methods with adaptive sets of search directions (see [4] and [5]). These methods are not convenient when the objective function is expensive because they require too many function evaluations. The other alternative is the model-based approach. DFO (Derivative-Free Optimization, see [6]) is a trust-region method building quadratic models that interpolate the objective function. UOBYQA (Unconstrained Optimization BY Quadratic Approximation, see [7]) is a derivative-free algorithm that constructs interpolant quadratic models of the objective function as DFO. It is based on the Lagrange functions and the parameters of the model are updated when an interpolation point is moved. Powell [8] has also proposed a method called NEWUOA (NEW Unconstrained Optimization Algorithm) that updates the quadratic model by minimizing the Frobenius norm of the change of the hessian of the model. He considered in particular the case where the quadratic models interpolate the function at $m = 2n + 1$ points. A consequence of this choice is that the complexity of the algorithm is very attractive. An other type of algorithm is a mix between direct-search methods and model-based methods. SMF (Surrogate Management Framework, see [9]) is an algorithm based on a pattern search and gives a general framework for generating and managing a sequence of approximations as surrogates for optimization. But when the objective function is expensive, the number of function evaluations is often too important.
We have decided to develop a new algorithm called BOOSTERS (Bierlaire & Oeuvray Optimization STrategy Exploiting Radial Surrogates). The choice of developing a new algorithm is mainly motivated by the fact that little attention has been paid to non-polynomial models in the context of trust-region methods. To the best knowledge of the authors, this algorithm is the first trust-region method based on RBFs developed in a general context and not for a specific application. Moreover we have shown in [10] that it converges to a first-order critical point. If the function f is globally convex, it is a global minimum. If the function is not convex, which is common in practice, the algorithm seeks a local minimum. This article is organized as follows. We first introduce the concept of trust-region methods in Section 2. Then we define the concept of radial functions in Section 3 and explain why they are interesting in the context of optimization. Section 4 describes our new algorithm. We present in particular in this section a method based on Newton fundamental polynomials of degree 1 to manage the interpolation points set. This original procedure permits to control and improve the geometry of the interpolation points if necessary. We finally present in Section 5 the tests we have performed on standard optimization problems from CUTEr (see [11]).

2 Trust-region methods

In continuous optimization, there are roughly two techniques to force the global convergence of algorithms: *linesearch* and *trust-region methods*. Linesearch algorithms typically consider the iterate computed by the Newton or quasi-Newton method, and check that the objective function has sufficiently decreased. If not, a shorter step in the same direction is considered and tested again, until specific conditions (for example

Armijo's conditions) are verified. Trust-region methods work as follow. At each iteration k, they build a quadratic model m_k around the current iterate x_k. This model is assumed to approximate the objective function sufficiently well in a region called *trust region*. Taylor's theorem guarantees that such a region exists, but does not provide its size for a given level of adequacy. Therefore, the radius Δ_k of the trust region must be updated at each iteration. A minimizer of the quadratic model within the trust region is considered as a candidate for the next iterate. If the function reduction forecasted by the model matches the actual reduction of the objective function sufficiently well, the candidate is accepted and the trust-region radius is possibly increased. If not, the candidate is rejected and the trust-region radius is decreased. The process starts again until a certain convergence criterion is satisfied. More formally, the general framework of the trust-region methods is given below and is taken from [12]. $B(x_k, \Delta_k)$ denotes a ball centered at x_k and of radius Δ_k.

Step 0: Initialization. An initial point x_0 and an initial trust-region radius Δ_0 are given. The constants η_1, η_2, γ_1, and γ_2 are also given and satisfy

$$0 < \eta_1 \leq \eta_2 < 1 \text{ and } 0 < \gamma_1 \leq \gamma_2 < 1.$$

Compute $f(x_0)$ and set $k = 0$.

Step 1: Model definition. Define a model m_k in $B(x_k, \Delta_k)$.

Step 2: Step calculation. Compute a step s_k that "sufficiently reduces the model" m_k and such that $x_k + s_k \in B(x_k, \Delta_k)$.

Step 3: Acceptance of the trial point. Compute $f(x_k + s_k)$ and define

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}.$$

If $\rho_k \geq \eta_1$, then define $x_{k+1} = x_k + s_k$; otherwise define $x_{k+1} = x_k$.

Step 4: Trust-region radius update. Set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, +\infty), & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k], & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k], & \text{if } \rho_k < \eta_1. \end{cases}$$

Increment k by 1 and go to Step 1.

The trust-region model is generally of the form

$$m_k(x) = f(x_k) + \nabla f(x_k)^t (x - x_k) + \frac{1}{2}(x - x_k) H_k (x - x_k),$$

where H_k is a symmetric approximation of the Hessian of f. When the derivatives are not available, one builds an interpolation model. The main difference between interpolation models and gradient-based models is that the former are considered as a suitable approximation of the objective function only under some conditions. These

conditions depend mainly of the geometry of the points. If they are satisfied, we say that the model is *valid* in the trust region. If not, a new point is generated to improve the accuracy of the model. The class of algorithms based on interpolation models are called *conditional* trust-region method. The term "conditional" just means that the model is a convenient approximation of f only if some conditions are satisfied.

The general framework of trust-region methods guarantees the convergence to a first or second-order critical point depending on the assumptions on the model and on the objective function. A full analysis of trust-region methods can be found in [12].

In the context of derivative-free optimization, the most effective algorithms are trust-region methods based on quadratic models that interpolate the objective function f. These methods are based on quadratic models of the type

$$m_k(x) = f(x_k) + g_k^t(x - x_k) + \frac{1}{2}(x - x_k)^t H_k(x - x_k),$$

where H_k is a symmetric matrix to be determined. The vector g_k and the matrix H_k are determined by the interpolation conditions, that is

$$m_k(y_i) = f(y_i), \quad i = 1, \ldots, m$$

where y_i are the interpolation points at iteration k and $y_1 = x_k$.

3 Radial basis functions

Our derivative-free algorithm uses interpolation models based on RBFs. In the next section, we give the most important results about the use of these functions in multivariate interpolation. Finally, we explain in Section 3.2 why we have chosen to build trust-region models based on cubic RBFs.

3.1 Multivariate interpolation

RBFs constitute an efficient tool for solving the multivariate scattered data interpolation problem. Duchon's contributions (see [13]) initiated their development and ever since they have been an active research field (see for example [14]). Numerous researchers have focused on the properties of these functions and they have also been used in many applications (see [15]).

The scattered data interpolation problem consists in finding a function that interpolates another function at some given points. Interpolation functions based on radial functions have some nice properties. The matrix of the system defined by the interpolation constraints is non-singular under some weak conditions. Moreover the interpolation function based on radial basis functions is optimal in the sense that it minimizes a norm defined on a set of interpolation functions (see [16]).

The choice of a radial basis function mainly depends on the application. Hardy [15] proposed a method employing multiquadric radial functions for a large variety of problems. Most of these applications are one, two or three dimensional. RBFs appear also naturally in a number of applications since they are the solutions of variational problems. Cubic splines (see for example [17]) are the smoothest interpolation functions

in dimension one while the thin plate splines (see [13]) have the same property in dimension 2. Powell [18] uses linear radial and multiquadric functions for interpolation of functions with many variables. The main reason of using these RBFs is that an iterative algorithm has been developed for these cases. Gutmann [19] has proposed a RBFs method for global optimization. His framework is general and several RBFs can a priori be used by his method. Käck [20] has developed a new algorithm for global optimization based on RBFs, that can handle linear as well as nonlinear constraints.
Here, we are interested in multivariate interpolation based on RBFs. In order to interpolate a function f whose values on a set $Y = \{y_1, \ldots, y_m\} \subset \mathbb{R}^n$ are known, we consider a function of the form

$$s(x) = \sum_{i=1}^{m} \lambda_i \phi(\|x - y_i\|) + q(x),$$

where q is a low degree polynomial, $\lambda_i, i = 1, \ldots, m$ are the interpolant parameters to be determined and ϕ is an application from $\mathbb{R}_+$ to $\mathbb{R}$. RBFs can be classified using the concept of conditionally positive definite functions. Let Ω be a subset of $\mathbb{R}^n$ and let us denote by $\pi_d(\mathbb{R}^n)$ or π_d the set of n-variate polynomials of degree at most d. ϕ is said to be *conditionally positive definite of order p on Ω*, $p \in \mathbb{N}$, if for all $m \in \mathbb{N}$ and all possible choices of sets

$$X = \{x_1, \ldots, x_m\} \subset \Omega$$

of m distinct points, the quadratic form induced by the $m \times m$ matrix A defined by

$$a_{ij} = \phi(\|y_i - y_j\|), 1 \leq i, j \leq m$$

is positive definite on the subspace

$$V_p = \begin{cases} \left\{\alpha \in \mathbb{R}^m : \sum_{j=1}^{m} \alpha_j q(x_j) = 0, \ \forall\, q \in \pi_{p-1}(\mathbb{R}^n)\right\}, & p \geq 1, \\ \mathbb{R}^m, & p = 0. \end{cases}$$

Conditionally positive definite of order 0 means that the matrix A is positive definite on $\mathbb{R}^m$. The most prominent examples of conditional positive definite RBFs of order p are

$$\phi(r) = \begin{cases} (-1)^{\lceil \beta/2 \rceil} r^\beta, \beta > 0, \beta \notin 2\mathbb{N}, & p \geq \lceil \beta/2 \rceil, \\ (-1)^{k+1} r^{2k} \log(r), k \in \mathbb{N}, & p \geq k+1, \\ (c^2 + r^2)^\beta, \beta < 0, & p \geq 0, \\ (-1)^{\lceil \beta \rceil} (c^2 + r^2)^\beta, \beta > 0, \beta \notin \mathbb{N}, & p \geq \lceil \beta \rceil, \\ e^{-\alpha r^2}, \alpha > 0, & p \geq 0, \\ (1-r)^4(1+4r), & p \geq 0, n \leq 3, \end{cases}$$

where $\lceil x \rceil$ is the smallest integer equal to or larger than x. The main interest of the concept of conditionally positive definite functions is the characterization of the non-singularity of the interpolation matrix. Let ϕ be conditionally positive definite of order p, $\widehat{p}$ be the dimension of $\pi_{p-1}(\mathbb{R}^n)$, $q_1, \ldots, q_{\widehat{p}}$ be a basis of this linear space, and let Q

be the matrix

$$\begin{pmatrix} q_1(y_1) & q_{\widehat{p}}(y_1) \\ \vdots & \vdots \\ q_1(y_m) & q_{\widehat{p}}(y_m) \end{pmatrix}.$$

It can be shown (see for example [21]) that the matrix

$$\begin{pmatrix} A & Q \\ Q^t & 0 \end{pmatrix}$$

is nonsingular if and only if $y_1, \ldots, y_m$ satisfy

$$q \in \pi_{p-1}(\mathbb{R}^n) \text{ and } q(y_i) = 0 \; i = 1, \ldots, m \implies q = 0.$$

In this case $\{y_1, \ldots, y_m\}$ is said to be $\pi_{p-1}(\mathbb{R}^n)$-*unisolvent* or π_{p-1}-*unisolvent*, and there is a unique function $s(x)$ of the form

$$s(x) = \sum_{j=1}^{m} \lambda_j \phi(\|x - y_j\|) + q(x), \; q \in \pi_{p-1}(\mathbb{R}^n)$$

which interpolates f at the points $y_i, i = 1, \ldots, m$. The coefficients are the solutions of the linear system of equations

$$\begin{pmatrix} A & Q \\ Q^t & 0 \end{pmatrix} \lambda = \begin{pmatrix} f \\ 0 \end{pmatrix},$$

where $f^t = (f(y_1) \ldots f(y_m))$ and λ is the vector of the model parameters and is of dimension $(m + \widehat{p})$.

3.2 Choice of the trust-region model

Most derivative-free algorithms are based on quadratic models. Our idea is to use more general models based on RBFs. In this context, we seek a function $\phi(\|x - x_j\|)$ as smooth as possible and we would like that the degree of the polynomial added to the radial terms does not exceed one. Models with higher degree are interesting but not attractive in our context since they require too many interpolation points. Consequently our trust-region model will be of the form

$$m_k(x) = \sum_{i=1}^{m} \lambda_i \phi(\|x - x_i\|) + c_0 + c^t x,$$

and will ideally belong to $C^2(\mathbb{R}^n)$. The fact that the model is twice differentiable is important for the convergence theory of our method (see [10]). The following lemma characterizes the differentiability of the model.

Lemma 1 *Let* $0 < b \leq +\infty, \Omega = \{x \in \mathbb{R}^n | \; \|x\| < b\}, u : \Omega \to \mathbb{R}$ *and* $\phi : [0, b) \to \mathbb{R}$ *such that* $u(x) = \phi(\|x\|) \; \forall x \in \Omega$. *Then*

$$u \in C^2(\Omega) \iff \phi \in C^2([0, b)) \text{ and } \phi'(0) = 0.$$

This result in well-known in PDE theory. Table 1 shows the C^2-differentiability conditions for different radial functions.

$\phi(r)$	*order*	C^2
$r^\beta, \beta > 0, \beta \notin 2\mathbb{N}$	$\lceil \beta/2 \rceil$	$\beta \geq 1$
$r^{2k}\log(r), k \in \mathbb{N}$	$k+1$	$k > 1$
$(c^2+r^2)^\beta, \beta < 0$	0	always
$(c^2+r^2)^\beta, \beta > 0, \beta \notin 2\mathbb{N}$	$\lceil \beta \rceil$	always
$\phi(r) = e^{-\alpha r^2}, \alpha > 0$	0	always

Table 1: Analysis of the differentiability of the main RBFs

It can be seen that there are several possible choices of ϕ. The radial function of type r^β is the only one without parameters, except the exponent, which belongs to $C^2(\mathbb{R}^n)$. Moreover, if $2 < \beta < 4$, then ϕ is of order 2 meaning that the interpolation system is non-singular if the interpolation set is π_1-unisolvent. This is equivalent to the existence of $(n+1)$ affinely independent points in the interpolation set. Note that cubic splines, corresponding to the choice of $\beta = 3$ in dimension 1, are the smoothest interpolation functions (see for example [17]). As a consequence of the discussion above, we have chosen to use trust-region models of type

$$m_k(x) = \sum_{i=1}^{m} \lambda_i \|x - x_i\|^3 + c_0 + c^t x.$$

If the number of interpolation points is m, then the model has $(m+n+1)$ parameters, m for the radial terms and $(n+1)$ for the linear function. But when the number of points is $(n+1)$, the solution of the interpolation system is just a linear function since all the parameters $\lambda_i, i = 1, \ldots, n+1$ are zero. Consequently the simplest nonlinear model is based on $(n+2)$ interpolation points and has $(2n+3)$ parameters.

4 Algorithm description

The algorithm is a trust-region method using interpolation models based on RBFs. In this context, the main issues are:

- the concept of validity,
- the building of the model,
- the trust-region subproblem.

These points are treated in detail in this section. It is organized as follows. We begin in Section 4.1 by giving a rigorous definition to the concept of validity. In the next section, we present the general framework of our new algorithm. In Section 4.3 we explain how to check this property and to improve the accuracy of the model when it is necessary. The trust-region subproblem consists in minimizing the model within the trust region.

We explain in Section 4.4 how to get a minimizer or a point that sufficiently decreases the model in order to guarantee the convergence to a critical point.

4.1 The concept of validity

The existence and the unicity of our model is guaranteed if there are $(n+1)$ affinely independent points in the interpolation set (see Section 3.2). A useful tool to measure the affine independence of points is based on Newton fundamental polynomials of order 1. This approach is explained in [12] in a more general context involving NFPs of degree $d \geq 1$. NFPs of order 1 are defined as follows. We assume that the interpolation set contains $(n+1)$ points y_i, $i = 1, \ldots, n+1$. The point y_1 is associated with the constant function $N_1^{[0]}(x) = 1$ and each point y_{i+1} is associated with a single polynomial $N_i^{[1]}(x), i = 1, \ldots, n$ of degree 1 satisfying the conditions

$$N_i^{[1]}(y_{j+1}) = \delta_{ij}, \quad j = 0, \ldots, n, \ i = 1, \ldots, n.$$

These polynomials are called *Newton fundamental polynomials of degree 1* and the values $N_i^{[1]}(y_{i+1})$ are called *interpolation pivots*. The procedure that computes these polynomials (see [12], page 325) may be considered as a version of the Gram-Schmidt orthogonalization procedure with respect to the inner product

$$< p, q >= \sum_{y \in Y} p(y)q(y), \quad p, q \in \pi_1(\mathbb{R}^n),$$

where Y is the interpolation set. One can show that Y is *poised* (see for example [12]) if and only if all the interpolation pivots are nonzero. We can now define the concept of validity. We say that the model is valid in $B(x_k, \Delta_k)$ if there are $(n+1)$ points in $\{x \mid \|x - x_k\| \leq C_1 \Delta_k\}$ whose pivots are larger than a certain threshold θ_1. C_1 is a constant larger or equal to 1 that measures the proximity of the points to the current iterate. A typical value of this constant is 2. Moreover, if we do not find $(n+1)$ points in Y with a pivot larger than $\theta_0 (\theta_0 << \theta_1)$, then we consider that Y is not poised and that the geometry of the points is too bad to compute the model. In this situation, we generate as many points as necessary to obtain a better geometry of the points. We also generate a new point when the model is not valid and when an unsuccessful iteration occurs. This procedure is also based on NFPs. We first select the point with the worst interpolation pivot and then we generate a new point by maximizing its corresponding NFP within the trust region. The maximum number of points in the interpolation set is fixed to a user-defined constant L_1 depending on the size of the problems and on the processing power which is available. Typically, for small problems, $L_1 = n(n+1)/2$ may be approriate, while smaller values must be used when n is larger. Concrete examples for the choice of L_1 are discussed in Section 5.1.

4.2 The framework of the algorithm

Conditional algorithms have been developed to deal with models that can be considered as a suitable approximation of the objective function only under some conditions.

These models are typically interpolation models and are said to be valid if the geometry of the interpolation points satisfies some conditions. When an unsuccessful iteration occurs and if the model is not valid, then the geometry of the points is improved instead of reducing the trust-region radius. The general framework of the algorithm is given below.

Step 0: Initialization. A set of points containing $(n+1)$ affinely independent points is given. For instance, a simplex centered at a starting point x_0 can be generated, where x_0 is provided by the user. An initial radius Δ_0 and a maximum radius Δ_{max} are also given. Moreover the real constants $\alpha, \mu, \varepsilon, \eta_1, \eta_2, \gamma_1, \gamma_2$ are also given and satisfy the conditions $\alpha \in (0,1)$, $\mu > 0$, $\varepsilon > 0$, $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$. We also have a constant integer L_1 which determines the maximum number of points of the interpolation set. Set $k = 0, \rho_{-1} = \eta_1$.

Step 1: Validity/Improvement step. Determine if the model is valid. If it is not the case and if $\rho_{k-1} < \eta_1$, generate a new point to improve the model.

Step 2: Computation of the model. Compute the parameters of the interpolation model.

Step 3: Final criticality test. If $\|\nabla m_k(x_k)\| \leq \varepsilon$, test if the model is valid in $B(x_k, \delta_k)$ for some $\delta_k \in (0, \mu\|\nabla m_k(x_k)\|)$. If $\|\nabla m_k(x_k)\| \leq \varepsilon$ but the model is not valid, add as many points as necessary to ensure that the model is valid in $B(x_k, \alpha\mu\|\nabla m_k(x_k)\|)$ and return to Step 3.

Step 4: Selection of the next iterate.

- Generate a minimizer $x_k + s_k$ of the trust-region model.
- Compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}.$$

  If $\rho_k \geq \eta_1$, set $x_{k+1} = x_k + s_k$ and the iteration is said to be successful. Otherwise, set $x_{k+1} = x_k$.

Step 5: Update of the trust-region radius.

- The trust-region for the next iteration is given by

$$\Delta_{k+1} \in \begin{cases} \min[\gamma_2\Delta_k, \Delta_{max}], & \text{if } \rho_k \geq \eta_2, \\ \gamma_1\Delta_k, & \text{if } \rho_k < \eta_1 \text{ and the model is valid,} \\ \Delta_k, & \text{otherwise.} \end{cases}$$

- Increment k by 1 and go to Step 1.

The only difference with the framework described in Section 2 is that we have added two new steps. One is the "Validity/Improvement step" and is explained in detail in the next section and the second one is the "Final criticality test" where the stopping condition is tested. The implicit loop in the Step 3 may be viewed as a model improvement inner iteration, with the aim of ensuring that $\nabla m_k(x_k)$ is not too different from the gradient of the objective function (see [10] for more details).

4.3 Validity and improvement of the model

The validity/improvement procedure is composed of 4 steps. In Step 0, we initialize the basis of the NFPs. Then we check the validity of the model (Step 1). If it is valid, then we stop the procedure. No improvement is required. Otherwise we make sure that the interpolation set is poised (Step 2). If it is not the case, then we generate as many points as necessary to obtain a geometry that makes possible the computation of the model parameters. In the last step (Step 3), we generate a new point if $\rho_k < \eta_1$ and if the model is not valid. In practice, the value of θ_0 is relatively small in comparison to θ_1 and the generation of new points in Step 2 is quite rare. But if it is the case, then the model may become valid. It is the reason why we check the existence of a pivot smaller than θ_1 in Step 3 meaning that the model is not valid.

Initialization step

This step initializes the basis of the polynomials of degree at most 1 with the canonical basis of $\pi_1(\mathbb{R}^n)$. It also initializes the set Y_{temp} containing the points selected by the algorithm and the set I corresponding to the indices of the NFPs which have not yet been associated with a point.

Step 0: Initialization. Set the $N_i^{[l]}, i = 0, 1, l = 1, \ldots, n$, to the canonical basis of $\pi_1(\mathbb{R}^n)$. Set $Y_{temp} = \{\varnothing\}$ and $I = \{1, \ldots, n\}$.

Validity step

The purpose of this procedure is to determine the "good points" in the region defined by $\{x \mid \|x - x_k\| \leq C_1\Delta_k\}$. A good point means that its pivot is larger than θ_1. This procedure is also used to test the validity of the model in $B(x_k, \delta_k)$ for the "Final criticality test" step by replacing the domain $\{x \mid \|x - x_k\| \leq C_1\Delta_k\}$ by $B(x_k, \delta_k)$ in the following step. The procedure works as follows.

Step 1: Validity of the model. Loop over the polynomials.

- Select the current iterate x_k, and update the set $Y_{\textit{Temp}} = Y_{\textit{Temp}} \cup \{x_k\}$;
- update the Newton polynomials by

$$N_i^{[1]}(x) = N_i^{[1]}(x) - N_i^{[1]}(x_k), \quad i = 1, \cdots, n;$$

- for $i = 1, \ldots, n$
  - select some $y_{k_i} \in Y \setminus Y_{\textit{Temp}}$ such that $y_{k_i} \in \{x \mid \|x - x_k\| \leq C_1\Delta_k\}$ and such that $|N_i^{[1]}(y_{k_i})|$ is maximal;
  - If this point exists and if $|N_i^{[1]}(y_{k_i})|$ is larger than θ_1, then
    - update the set by $Y_{\textit{Temp}} = Y_{\textit{Temp}} \cup \{y_{k_i}\}$ and $I = I \setminus \{i\}$;
    - normalize the current polynomial:

$$N_i^{[1]}(x) = N_i^{[1]}(x)/N_i^{[1]}(y_{k_i}); \tag{1}$$

· update the Newton polynomials:

$$N_j^{[1]}(x) = N_j^{[1]}(x) - N_j^{[1]}(y_{k_i})N_i^{[1]}(x) \tag{2}$$

for $j \neq i, j = 1, \cdots, n$.

– If $|Y_{Temp}| < (n+1)$, then go to Step 2. Else stop.

Poisedness step

This procedure determines if the interpolation set is poised. If not, we generate as many points as necessary to obtain a better geometry. These points are determined by maximizing the NFPs in the trust region. The procedure is described below.

Step 2: Poisedness. Loop over the polynomials.

– While $|I| > 0$

∗ select $i \in I$;

∗ select some $y_{i_1} \in Y \setminus Y_{Temp}$ such that $|N_i^{[1]}(y_{i_1})|$ is maximum;

∗ determine $j \in I$ that maximizes $|N_j^{[1]}(y_{i_1})|$;

∗ If $|N_j^{[1]}(y_{i_1})|$ is strictly smaller than θ_0, then

· generate a new point y_{k_1}:

$$y_{k_1} = \max_{x \in B(x_k, \Delta_k)} |N_i^{[1]}(x)|;$$

· compute $f(y_{k_1})$ and update Y;

else $i = j$ and $k_1 = i_1$.

∗ update $Y_{Temp} = Y_{Temp} \cup \{y_{k_1}\}$ and $I = I \setminus \{i\}$.

∗ normalize the current polynomial: $N_i^{[1]}(x) = N_i^{[1]}(x)/N_i^{[1]}(y_{k_1})$;

∗ update the Newton polynomials:

$$N_j^{[1]}(x) = N_j^{[1]}(x) - N_j^{[1]}(y_{k_1})N_i^{[1]}(x)$$

for $j \neq i, j = 1, \cdots, n$;

– go to Step 3.

The update of the interpolation set works as follows. If $|Y| < L_1$ (the parameter determining the maximum number of interpolation points), then we simply add the new point. However, if $|Y| = L_1$, the addition of the new point is immediately followed by the removal of the oldest point in the set. This step is quite similar to the step that checks the validity of the model. The main difference is that after selecting a point, we re-optimize the process by determining the best NFP still available for that point. A bad point for a certain NFP may be a good one for another choice of polynomial. The absence of this phase of re-optimization may cause troubles in our management of the interpolation point. The risk is that the value of the pivots does not reflect enough

the geometry of the points and that the maximization of the NFP having the worst interpolation point does not necessarily generate the best point for the geometry.Note that steps 1 and 2 should not be merged. Indeed it is important to first select the good points to check the validity of the model before testing the poisedness to avoid the pathological case described before.

Improvement step

After an unsuccessful iteration and when the model is not valid, we generate a new point by maximizing the NFPs corresponding to the smallest pivot. This is explained in the improvement step below.

> **Step 3: Improvement.** Let $N_i^{[1]}(x)$ be the polynomial with the smallest interpolation pivot. If this value is strictly smaller than θ_1, then
>
> – generate a new point y_{k_1}:
>
> $$y_{k_1} = \max_{x \in B(x_k, \Delta_k)} N_i^{[1]}(x);$$
>
> – compute $f(y_{k_1})$ and update Y.

Our management of the interpolation points is independent of the scaling of the objective function. But the concept of validity defined as before shows its limits when the problem is ill-conditioned. This happens namely when a problem involves variables with different order of magnitude. This means that the objective function is highly sensitive to small changes in certain components of the vector x and insensitive to such changes in other components. In this situation, the contours of the objective function near the minimizer tend towards highly eccentric ellipses. Spherical trust regions are not appropriate to the case of poorly scaled functions. We can trust our model m_k to be reasonably accurate only over short distances along highly sensitive function directions, while it is reliable over longer distances along the less sensitive directions. From these remarks it follows that our management of the interpolation points is not very convenient for ill-conditioned problems.

The complexity of the Gram-Schmidt orthogonalization algorithm is $O(n^3)$ and a direct consequence is that the complexity of the "Validity/improvement procedure" is $O(mn^2)$, where $m = |Y|$.

4.4 The trust-region subproblem

The trust-region subproblem is at the core of trust-region methods. For an overview of this topic, see [12]. This subproblem consists in finding the minimizer of the trust-region model. In fact it is not necessary to find the minimizer of the trust-region model in order to apply the convergence theory of trust-region methods. The key point is to make sure that the total decrease is at least a fraction of that obtained at the Cauchy point (see [12], page 131). In practice, one can compute the approximate Cauchy

point using the well-known Armijo linesearch (see [22]) or to compute a minimizer of the trust-region subproblem. The approximate Cauchy point is obtained quite easily but the decrease of the objective function is often not very satisfying. The key point in derivative-free optimization is to know how much time we can spend in the algorithm as compared to the time spent to evaluate the objective function. If this time is negligible, then we can afford to compute a minimizer of the trust-region subproblem but if it is not the case, we should prefer to compute the approximate Cauchy point. In our applications, we have tested both methods depending on the cost of evaluating the objective function but also on the size of the problem. An iteration in the algorithm has a complexity of $O(n^3)$. This is not negligible for problem whose dimension is larger than 50. In this situation the use of the linesearch instead of computing an exact minimizer could be an interesting alternative if the cost of evaluating the objective function is not very high. In the tests we have performed (see Section 5), we have used two different strategies: for large problems of size 200, we have computed the approximate Cauchy point (see Section 4.4.1) while we have determined a minimizer of the trust-region subproblem by using CFSQP (see Section 4.4.2) when the size of the problems was smaller.

4.4.1 The approximate Cauchy point

The idea behind the concept of the approximate Cauchy point is to use a backtracking linesearch along the direction given by the gradient of the model. More precisely, we determine the smallest integer j such that the point z_j defined by

$$z_j = x_k - \alpha_1^j \frac{\Delta_k}{\|\nabla m_k(x_k)\|} \nabla m_k(x_k)$$

satisfies the conditions

$$m_k(z_j) \leq m_k(x_k) + \beta_1 \nabla m_k(x_k)(z_j - x_k),$$

where $\alpha_1 \in (0,1)$ and $\beta_1 \in (0, \frac{1}{2})$ are given constants.

4.4.2 CFSQP

CFSQP [23] is the acronym of C code for Feasible Sequential Quadratic Programming. It is a set of C functions for the minimization of a smooth objective function subject to nonlinear equality and inequality constraints, linear equality and inequality constraints, and simple bounds on the variables. It is based on SQP methods, modified in order to generate feasible iterates if necessary.

5 Numerical results

We have compared the performances of the following derivative-free algorithms: BOOSTERS, DFO, UOBYQA and NEWUOA. We have taken the implementations in Fortran distributed by the authors. All these methods are trust-region methods and the stopping criterion for all of them is based on the size of the trust region. We first begin by briefly

explaining the concept of performance profiles. They were introduced in [24] and provide an effective tool to compare solver performance on a collection of problems. The performance profile of a solver is an empirical (cumulative) distribution function for a performance metric. Dolan and Moré [24] use the ratio of the solver resource time for a given solver versus the best time of all the solvers. But in derivative-free optimization, the performance is generally measured by the number of function evaluations until convergence. This is the reason why the CPU time is generally not provided in the tests we have performed.

The performance profile for a method is the cumulative distribution function for a given performance metric. If $f_{p,a}$ is the performance metric of algorithm a on problem p (the number of function evaluations in our case), then the *performance ratio* is defined by

$$r_{p,a} = \frac{f_{p,a}}{\min_a\{f_{p,a}\}}, \tag{3}$$

if algorithm a has converged for problem p, and $r_{p,a} = r_{\text{fail}}$ otherwise, where r_{fail} must be strictly larger than any performance ratio (3). For any given threshold π, the overall performance of algorithm a is given by

$$\rho_a(\pi) = \frac{1}{n_p}\Phi_a(\pi) \tag{4}$$

where n_p is the number of problems considered, and $\Phi_a(\pi)$ is the number of problems for which $r_{p,a} \leq \pi$.

In particular, the value $\rho_a(1)$ gives the probability that algorithm a wins over all other algorithms. The value $lim_{\pi \rightarrow r_{\text{fail}}}\rho_a(\pi)$ gives the probability that algorithm a solves a problem and, consequently, provides a measure of the robustness of each method.

We have tested the algorithms on the unconstrained problems of CUTEr with the characteristics that the objective function is not a polynomial function of order 2 or less and not a sum of squares. It is clear that our method is not a judicious choice for such objective functions where the use of methods based on quadratic models is recommended. The test set contains problems with a fixed or a variable dimension. We have 18 problems with a fixed dimension smaller than 20 and one can parametrize the size of 39 other functions. This makes a total of 57 problems.

5.1 Tests of different variants

We have tested several variants of our method depending on the maximum number L_1 of interpolation points. Table 2 gives the variants we have tested.

As mentioned in Section 3.2, the minimum number of interpolation points with non null radial terms is $(n + 2)$. The value $(2n + 1)$ corresponds to the number of interpolation points in NEWUOA while $n(n+1)/2$ is the number of parameters in a full quadratic model. The variant V_4 is motivated by the fact that in very small dimension (<5) it is absolutely not prohibitive to put more points in the interpolation set than the limit imposed by the use of quadratic models. For example a full quadratic model

Variant	L_1
V_1	$n+2$
V_2	$2n+1$
V_3	$\frac{n(n+1)}{2}$
V_4	$\max\left\{\frac{n(n+1)}{2}, 15\right\}$

Table 2: List of the variants we have tested. The value L_1 corresponds to the maximum number of interpolation points.

has 6 parameters in dimension 2 but a model based on RBFs can easily integrate 15 interpolation points. We have tested these variants on problems of dimension smaller than 20. It is also important to note that when the number of interpolation points is $O(n^2)$, then the complexity of an iteration in BOOSTERS grows from $O(n^3)$ to $O(n^6)$. The problems are divided in four classes depending on their size. The distribution of the dimension of the problems is given in Table 3.

dimension	$2-5$	$6-10$	$11-15$	$16-20$
nbr of problems	16	14	14	13

Table 3: Distribution of the dimension of the problems.

The starting point is taken from CUTEr. The maximum number of function evaluations is fixed to 5000 and the stopping criterion is the size of the trust region (Δ_{min} =1.0e-04). The performance metric is the number of function evaluations until convergence is reached. We consider that an algorithm has converged if the relative error in x is less than 1% or if the absolute value is less than 10^{-2} if the solution is 0. If the exact solution is not known, then we consider the value provided by the best algorithm as the solution and the same criteria as before apply. The performance profiles of the four variants of BOOSTERS are given in Figure 1.

We see that the more points, the better the result. The variant V_4 is about 1 time out of 2 the best and its performance profile converges the most quickly.

We have compared the best variant (V_4) of BOOSTERS with its competitors on the same test set as in the preceding paragraph. The performance profiles of the algorithms are given in Figure 2. NEWUOA is not only the algorithm with the lowest complexity (with BOOSTERS) but has also the best results for these tests. It is the best algorithm about 3 times out of 10 and its performance profile converges the most quickly. DFO is quite effective but it is also the less robust algorithm in the sense that it has failed to converge in about 35 % of the tests. The performances of UOBYQA and BOOSTERS are quite similar even though the second algorithm seems a little better. BOOSTERS is the best algorithm about one time out of four and is at the second place in terms of robustness.

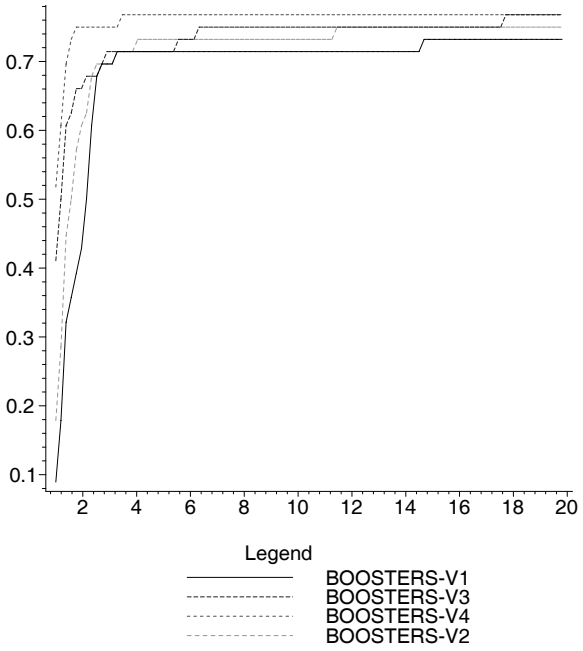

Figure 1: Performance profiles of the four variants of BOOSTERS.

5.2 BOOSTERS versus NEWUOA

At some iterations, DFO has to recompute all the parameters of the trust-region model, meaning that the complexity of an iteration is $O(n^6)$ in the worst case (when all the parameters are recomputed) while it is $O(n^4)$ in UOBYQA. NEWUOA and BOOSTERS are much better with a complexity of only $O(n^3)$ (only in the worst case for NEWUOA, otherwise $O(n^2)$) when the number of interpolation points is linear in n. This makes a strong difference. On a problem of dimension 50, DFO needs more than 9 hours on our PC (Dell computer, 1000 MHz, 256 MB of RAM, Linux) to compute the 500 first iterations while NEWUOA, the fastest algorithm, only needs a few seconds. Indeed, the direct competitor of NEWUOA is the variant of BOOSTERS based on $(2n + 1)$ interpolation points. These two algorithms are quite similar: they build models that interpolate the objective function at $(2n + 1)$ different points and have the same complexity. We have compared both solvers on problems with a medium dimension (between 20 and 81). The distribution of the dimensions is given in Table 4 and the performance profiles are given in Figure 3.

The performance profile of BOOSTERS is significantly better than NEWUOA. This result is quite surprising since the tests performed on problems in small dimension (≤ 20) have shown that NEWUOA was the best algorithm. The larger the size is, the more effective BOOSTERS seems. The problems of CUTEr with a variable dimension seem less hard than the small problems with a fixed dimension. Indeed the scaling of

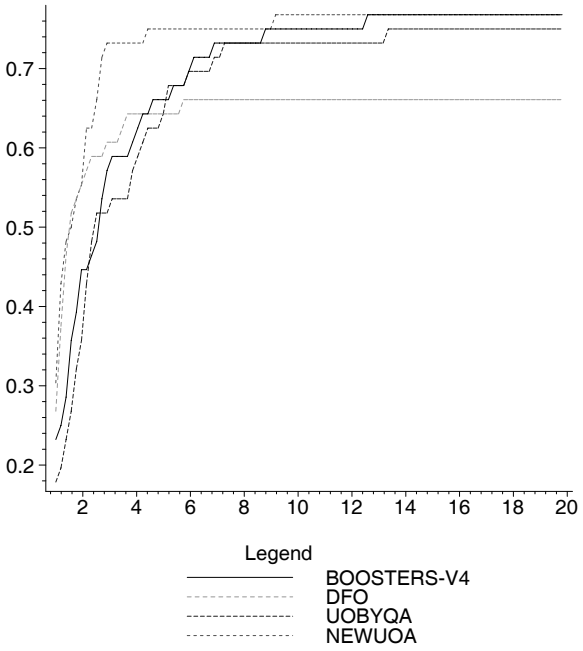

Figure 2: Performance profiles of the algorithms.

dimension	20	40	60	$80-81$
nbr of problems	10	10	12	7

Table 4: Distribution of the dimension of the problems.

small problems is often not good and the management of the interpolation points in BOOSTERS is not very convenient in this situation as explained in Section 4.1.

5.3 Large problems

In derivative-free optimization, problems with more than hundred variables are difficult to handle. But our method based on $(n+2)$ interpolation points can easily tackle these problems. In this situation, the trust-region models have $(2n+3)$ parameters, $(n+2)$ for the radial terms and $(n+1)$ for the linear function. The resolution of the system to determine the parameters of the model only requires to solve two linear systems of dimension $(n+1)$ and $(n+2)$. The computational time is only twice the time necessary to compute a linear model while the number of parameters is much larger. To the best knowledge of the authors, the largest problem ever tested by NEWUOA is in dimension 160. We have tested here our variant on some problems of dimension 200. Our objective was to solve these problems in less than 10000 function evaluations. This

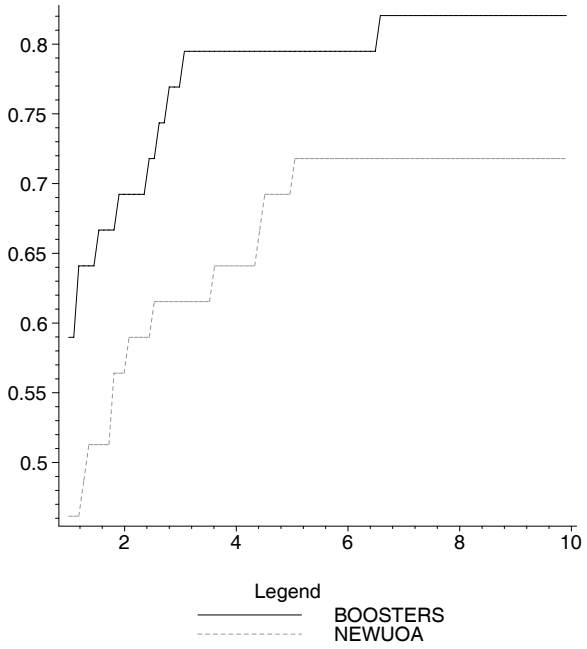

Figure 3: Performance profiles of the algorithms.

number is relatively low in comparison to the size of the problems. The framework of the algorithm imposes to have a valid model before decreasing the trust-region radius and it sometimes requires a lot of function evaluations. Table 5 gives the value of the objective function f after 10000 function evaluations while Figure 4 plots the logarithm of the value of f in function of the number of evaluations for the DIXMAAN[A-D] problems. In view of these results, we can consider that the algorithm has converged in all cases. For information, the resolution takes more than 3 hours per problem on a Dell computer (1000 MHz, 256 MB of RAM, Linux).

5.4 Global analysis of the results

We have tested the algorithms on the problems of CUTEr. On small problems, the performance of BOOSTERS is similar to these of the other algorithms. The solver with the best complexity is NEWUOA and its direct competitor is the variant of BOOSTERS based on $(2n + 1)$ interpolation points. The tests we have performed show that our method surpasses NEWUOA for medium-size problems. Finally, we have solved large problems of dimension 200. The tested variant of BOOSTERS is based on the minimum number of interpolation points necessary to calibrate our model based on RBFs, namely $(n + 2)$ points.

Problem	*Size*	$f(x_*)$	*Solution*
ARWHEAD	200	+1.232293e-05	0.0
DIXMAANA	201	+1.000054e+00	1.0
DIXMAANB	201	+1.000326e+00	1.0
DIXMAANC	201	+1.002541e+00	1.0
DIXMAAND	201	+1.013138e+00	1.0

Table 5: Tests performed on large problems with BOOSTERS. The column entitled $f(x_*)$ corresponds to the value of the objective function after 10000 function evaluations.

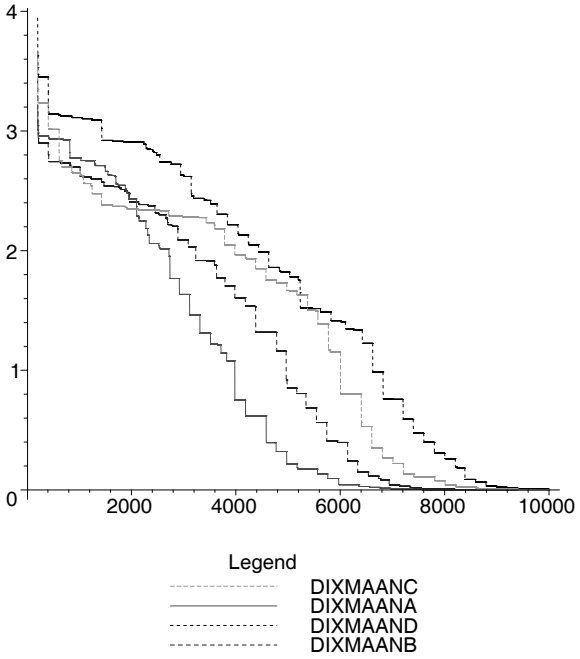

Figure 4: Plot of the logarithm of f in function of the number of evaluations for the DIXMAAN[A-D] problems.

6 Conclusion

We have developed a new derivative-free algorithm based on RBFs. The main originality of our approach is the use of RBFs and the management of the interpolation points based on Newton fundamental polynomials. The complexity of the algorithm is very attractive in comparison with its best competitors, namely UOBYQA, NEWUOA and DFO. Moreover the results of the tests we have performed on CUTEr are excellent.

Our method is a powerful tool when the objective function is a non-linear non-polynomial function. In addition, we have shown in [25] and [10] that the method is also well designed for problems where the objective function is noisy or only an approximation of an unknown function.

References

[1] Robert Michael Lewis, Virginia Torczon, and Michael W. Trosset. Direct search methods: Then and now. *Journal of Computational and Applied Mathematics*, 124:191–207, 2000.

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

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

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

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

[6] A. R. Conn and Ph. L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In G. Di Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 27–47. Plenum Publishing, 1996. Also available as Report 95/6, Dept of Mathematics, FUNDP, Namur, Belgium.

[7] M. J. D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. Technical Report DAMTP NA14, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, UK, 2000.

[8] M. J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives. Technical Report DAMTP NA2004/08, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge CB3 9EW, UK, 2004.

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

[10] R. Oeuvray. *Trust-Region Methods Based on Radial Basis Functions with Application to Biomedical Imaging*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2005.

[11] N. I. M. Gould, D. Orban, and Ph. L. Toint. General CUTEr documentation. Technical Report TR/PA/02/13, CERFACS, 2002.

[12] A. R. Conn, N. I. M. Gould, and Ph. Toint. *Trust region methods*. MPS–SIAM Series on Optimization. SIAM, 2000.

[13] J. Duchon. Splines minimizing rotation-invariate semi-norms in sobolev spaces. *Constructive Theory of Functions of Several Variables*, pages 85–100., 1979.

[14] M.D. Buhmann. *Multivariate Approximation and Apllications*, chapter Approximation and interpolation with radial functions, pages 25–43. Cambridge, 2001.

[15] R.L. Hardy. *Computers and Mathematics with Applications*, volume 19, chapter Theory and applications of the multiquadric-biharmonic method, pages 163–208. Pergamon Press plc, 1990.

[16] R. Schaback. *Approximations: From CAGD to Wavelets*, chapter Comparison of radial basis function interpolants, pages 293–305. World Scientific, 1993.

[17] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press, Cambridge, UK, 1981.

[18] M. J. D. Powell. Radial basis function for interpolation to function of many variables. Technical Report DAMTP NA11, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, UK, 2001.

[19] H.-M. Gutmann. A radial basis function method for global optimization. Technical Report DAMTP NA22, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, UK, 1999.

[20] J.-E. Käck. Constrained global optimization with radial basis functions. Technical report, Department of Mathematics and Physics, Mälardalen University, 2004.

[21] R. Schaback and H. Wendland. Characterization and construction of radial basis functions. In N. Dyn, D. Leviatan, D. Levin, and A. Pinkus, editors, *Multivariate Approximation and Applications*, pages 1–24. Cambridge University Press, 2001.

[22] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa., 1996.

[23] C.T. Lawrence, J.L. Zhou, and A. Tits. User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland, College Park, MD 20742, 1997, 1997.

[24] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming, Serie A*, 91:201–213, 2002.

[25] R. Oeuvray and M. Bierlaire. A new derivative-free algorithm for the medical image registration problem. *International Journal of Modelling and Simulation*, to appear.