Optim Eng (2010) 11: 501–532
DOI 10.1007/s11081-009-9087-1

# A method for simulation based optimization using radial basis functions

**Stefan Jakobsson · Michael Patriksson · Johan Rudholm · Adam Wojciechowski**

Received: 29 January 2008 / Accepted: 27 May 2009 / Published online: 19 June 2009


**Abstract** We propose an algorithm for the global optimization of expensive and noisy black box functions using a surrogate model based on radial basis functions (RBFs). A method for RBF-based approximation is introduced in order to handle noise. New points are selected to minimize the total model uncertainty weighted against the surrogate function value. The algorithm is extended to multiple objective functions by instead weighting against the distance to the surrogate Pareto front; it therefore constitutes the first algorithm for expensive, noisy and multiobjective problems in the literature. Numerical results on analytical test functions show promise in comparison to other (commercial) algorithms, as well as results from a simulation based optimization problem.

**Keywords** Simulation based optimization · Radial basis functions · Multiobjective · Noise · Response surface · Surrogate model · Black box function

S. Jakobsson
Fraunhofer–Chalmers Research Centre for Industrial Mathematics, Gothenburg, Sweden
e-mail: stefan.jakobsson@fcc.chalmers.se

M. Patriksson · J. Rudholm · A. Wojciechowski (✉)
Department of Mathematical Sciences, Chalmers University of Technology, Gothenburg, Sweden
e-mail: wojcadam@chalmers.se

M. Patriksson
e-mail: mipat@chalmers.se

J. Rudholm
e-mail: chm@etek.chalmers.se

M. Patriksson · J. Rudholm · A. Wojciechowski
Department of Mathematical Sciences, University of Gothenburg, Gothenburg, Sweden

## 1 Introduction

### 1.1 Background

This paper introduces a new algorithm for simulation based optimization. The algorithm was developed as part of a project to optimize combustion engine simulations at Volvo Car Corporation and Volvo Powertrain. The purpose of performing combustion engine simulations is to find an optimal engine design with respect to fuel consumption while at the same time keeping the emissions of soot and nitrogen at an acceptable level. The performance of a combustion engine varies between load cases; therefore, a design should be found that performs well on several load cases. The problem described is a multiobjective simulation based optimization problem viewing the fuel consumption at the different load cases as the objective functions and the emission bounds as constraints. The problem is characterized by the following aspects. There are no analytic derivatives of the objective function available (usually known as a black box function); a simulation takes 42 hours to complete and the resulting output may contain different types of disturbances (such as numerical noise).

As the engine optimization illustrates, an interest in simulation based optimization for industrial applications exists. Research and development departments often model a new product as a computer simulation before constructing the initial (physical) prototype. A common way in the industry to make a good choice of design variable values for a given simulation model is to generate a set of sample points (set of design variable values) in the variable space, run the simulation for this set and create a *surrogate model*[1] of the resulting values. The surrogate model may for instance be an interpolation based on radial basis functions (RBFs), neural network models or moving least squares. This approach is usually called design of experiments, and is followed by letting an optimization algorithm find an approximate optimum of the surrogate model. As the surrogate model contains inaccuracies the surrogate optimum may correspond poorly to the simulation optimum. The surrogate model can be improved by evaluating points in areas far away from previously evaluated points and areas with interesting surrogate model values. The model is updated with the evaluated points and again optimized. The procedure is iterated until an acceptable accuracy is obtained. The algorithm proposed in this paper automates the procedure just outlined.

### 1.2 Simulation based optimization

In this paper, we restrict ourselves to treating simulations as expensive black-box functions. Let $\boldsymbol{x}$ be the decision variables, $f$ the objective function. Assuming a $d$-dimensional parameter domain $\Omega = \{\boldsymbol{x} \in \mathbb{R}^d : x_{L,j} \leq x_j \leq x_{U,j},\ j = 1, \ldots, d\}$, the optimization problem considered is to

$$\text{minimize} \quad f(\boldsymbol{x}), \tag{1a}$$

$$\text{subject to} \quad x_{L,j} \leq x_j \leq x_{U,j} \quad j = 1, \ldots, d, \tag{1b}$$

[1] The term was first used in Booker et al. (1999).

where (1b) are referred to as the box constraints. The objective function $f$ is the black box function evaluated by simulation.

Another important aspect of simulations is that the response obtained often contains noise, i.e. it differs from the phenomenon being simulated. Mainly two types of noise are present in simulations. The first type is ordinary numerical noise which may for example be a product of rounding errors in the simulation process. The second type of noise arises from certain types of simulations, particularly Finite Element Method (FEM) and Computational Fluid Dynamics (CFD) based simulations where geometrical structures are described by unions of simpler objects (such as triangles and tetrahedrons), forming a mesh. When the geometry of the object simulated is modified, some mesh objects are removed and others are introduced. This may give rise to discontinuous changes in the simulation output, whereas the phenomenon being simulated often is a continuous function. When dealing with noisy problems, we will call the function being simulated the *true* function and denote it by $f$. The output of the simulation will be denoted by $\tilde{f}$ and called the *noisy* function. We are interested in minimizing $f$, but we have only access to $\tilde{f}$. When noise is not present, $f = \tilde{f}$.

More specifically, we are considering simulation based problems with the following features:

1. $f$ is continuous;
2. $\tilde{f}$ is expensive to evaluate;
3. no analytical derivatives of $\tilde{f}$ are available ($\tilde{f}$ is a black box function);
4. box constraints (1b);
5. noise may be present ($\tilde{f}$ perturbed from $f$);
6. the parameter space is of low dimension;
7. there are possibly several objective functions.

The continuity assumption is made in order to motivate the use of interpolation/approximation as surrogate models. Of course, an algorithm based on surrogate models can be applied to any function but to be successful the quality of these approximations are crucial and this is hard to obtain if the underlying function is discontinuous. Since we do not make any convexity assumptions, a method for solving the above problem should be globally convergent; the method must also keep function evaluations at a minimum, and be derivative-free.[2] We limit the class of problems to ones having a low number of decision variables (up to about 6). This is necessary to obtain acceptable running times for the algorithm presented in this paper. Furthermore, industrial applications often have more than one objective function (among others, consisting of outputs of several simulations or different outputs from a single simulation). In this scenario, methods for single objective optimization will not work; instead methods for so called multiobjective optimization must be used.

### 1.3 Previous research

Several properties of simulation based optimization make most conventional optimization algorithms unsuitable. The absence of analytic derivatives narrows the field

[2] Also with regard to finite-difference approximations of the gradient, since such approximations cost too many function evaluations.

to algorithms using no first or second order information about the function. The biggest problem, however, is that objective functions obtained by simulations are expensive to evaluate. Therefore the use of algorithms inspired by physics and natural selection (simulated annealing and genetic algorithms for instance), which use many thousand function evaluations, is not an option. Direct search methods (such as pattern search methods for example, see Lewis et al. 2000), also require a lot of function evaluations. Derivative free and global algorithms not falling into any of the above categories include trust-region methods (such as DFO from Conn et al. 1997 or NEWUOA from Powell 2006), Shor's $r$-algorithm (Shor 1985), DIRECT[3] (*DIviding RECTangles*) (Jones et al. 1993) and MCS (*Multilevel Coordinate Search*) (Huyer and Neumaier 1999). These algorithms generally converge to a global minimum in a smaller number of function evaluations than those mentioned previously, according to numerical experiments on reference functions. They could therefore be considered for use directly on expensive problems.

Most algorithms developed for solving simulation based optimization problems fall into the category of *Response Surface Methods* (RSMs). An RSM provides a surrogate model (or a response surface) of the expensive function (simulation); this surrogate model is then used for optimizing the simulation. Examples of surrogate models are linear or quadratic approximations, or some sort of interpolation. A widely used form of interpolation consists of linear combinations of basis functions. There are different choices of these basis functions, the most popular ones being kriging, thin plate splines, cubic splines and multiquadratics. Common to all surrogate-based optimization methods is the concept of using the surrogate for iteratively selecting new points to evaluate. These points are then used to further refine the surrogate model. A common approach among the RSMs, called the *two stage* approach by Jones, can be described by the following steps:

0. create and evaluate an initial set of points
1. create a surrogate model using the evaluated points
2. select a new point using the surrogate model and evaluate it
3. go to step 1, unless a stopping criterion is met.

The initial sample in step 0 has to be created without any à priori knowledge of the function. Usually the set is created using some design of experiments, such as latin hypercubes.

Step 2 is the most critical step of any algorithm of this class, since this is where function evaluations may be saved. The algorithm must balance *local* and *global* searches. A local search entails trusting the surrogate model, and selecting points at the (global) minimum of the model. A global search selects points in areas where few evaluations have been performed and thus improves the accuracy of the model there. If the search strategy is very local, the algorithm will only evaluate points close to the current surrogate minimum, possibly missing minima in regions with low surrogate accuracy. If it, on the other hand, is very global and the current surrogate minimum is already close to a global minimum, function evaluations will be wasted. The stopping

[3]DIRECT was used as an external solver in the implementation of the algorithm presented here.

criterion in step 3 is usually defined by the number of function evaluations, which in turn is bounded by some allotted running time for the entire optimization procedure.

The proposed algorithm in this paper is a two-stage RSM. Below follows a short summary of previous research on RMS-based algorithms; for an in-depth study, see Jones (2001).

A common approach is to combine conventional algorithms such as genetic algorithms or pattern search together with surrogate models in order to solve expensive problems (see for instance Booker et al. 1998). This is usually done by selecting the point for evaluation as the minimizer of the surrogate.

Jones et al. (1998) propose a method based on kriging basis functions, called Efficient Global Optimization (EGO), and uses *expected improvement* of the surrogate to select new points. This algorithm is among the first to use RSMs and is frequently cited in recent publications.

Gutmann (2001) constructs the response surface with RBFs. New points are selected for evaluation by using a *target-value* $f^*$ (a guess of the minimal objective value) which is varied in cycles during the course of iterations to achieve a balance between local and global searches. In short, the point selected for evaluation is that which maximizes the smoothness of the interpolation, using some external global optimization algorithm.

Regis and Shoemaker (2005) proposes the Constrained Optimization using Response Surfaces (CORS) algorithm. CORS evaluates new points by minimizing the surrogate value under a constraint on the distance to the previously evaluated points. The constraint is cycled during the optimization procedure. Regis and Shoemaker (2007) propose the Stochastic Response Surface (SRS) algorithm. The next point to evaluate is chosen by randomly spreading a large number of points over the parameter domain and selecting the point with the minimal weight. The weight of a point is defined by one part decreasing as the distance to the closest evaluated point increases and one part decreasing with the surrogate value at the point. The balance between these weights is cycled. These two methods have some similarities with the strategy proposed in this paper for deterministic single objective problems (see Sect. 2.2). The main difference is that we select the next point to evaluate by considering decrease of weight over the whole domain (by integrating over it) in contrast to the pointwise approach used in SRS and CORS.

Huang et al. (2006) provide the first RSM-based algorithm that considers noisy objective functions, called Sequential Kriging Optimization (SKO). The algorithm is an extension of Jones' EGO algorithm to handling noisy objective functions.

Algorithms for solving multiobjective problems is a vast research area. A common approach is to use genetic algorithms (such as MATLAB's gamultiobj), but alternative methods exist (such as the NBI method presented in Das and Dennis 1998). The literature on algorithms for expensive multiobjective problems however is less extensive. Some of the response surface methods may be extended to multiple objectives. Knowles (2006) for instance developed ParEGO, an adaptation of Jones' EGO algorithm, for multiobjective optimization. In Audet et al. (2008) problems with two objectives are solved by using the mesh adaptive direct search (MADS) method on several single objective problems. An algorithm developed directly for multiobjective problems is presented in Messac and Mullur (2008) where the Normal Constraint

(NC) method is extended to handle expensive functions by creating local RBF approximations of the objectives. However, the article focuses more on creating a good final surrogate model given evaluated points close to the Pareto front.

To the authors' knowledge, the literature does not contain any previous algorithms designed for noisy, expensive multiobjective optimization problems. Furthermore the focus of the response surface algorithms is on pointwise uncertainty which tend to be large at domain boundaries. This may result, as in Sect. 3.1, Fig. 6C, in selecting many points at the boundaries. Another issue that we have observed, also present in Fig. 6C, is the tendency of some algorithms[4] to "cluster" points, that is, to select new points for evaluation close to already evaluated points, by emphasizing local search too much. This can be a good strategy for reaching below the 1% target fast on test functions. In real simulation based problems however, the decision maker is seldom interested in simulating many very similar designs, as the accuracy of the simulation model is limited. Instead it is important to find the region containing a global minimum. Furthermore, when the objective function contains noise, interpolating densely placed points may result in a strongly oscillating surrogate model (see Sect. 2.1.2, Fig. 1A).

### 1.4 Motivation for the proposed algorithm

As discussed in Sect. 1.3, the existing algorithms may display unwanted characteristics in real-life situations. The algorithm proposed is therefore designed to solve the type of problems described in Sect. 1.2, as well as to be able to accommodate the following additional features:

1. avoid clustering of points
2. avoid over-emphasis of border regions
3. reduce overall uncertainty.

In order to achieve these goals the proposed algorithm, henceforth referred to as *qualSolve*, implements a novel way of selecting new points and a way of handling noise in the objective function.

This article is composed by the following parts. Section 2 describes the inner workings of the algorithm proposed: how the interpolation is constructed, how new points for evaluation are chosen and how the algorithm is adapted to multiobjective optimization. Section 3 reports on numerical results on some reference functions, as well as on a simulation based problem. Finally, Sect. 4 contains conclusions and suggests improvements to the proposed algorithm and areas of future research.

## 2 The optimization algorithm

The algorithm *qualSolve* presented in this article can be characterized as a two-stage approach (see Sect. 1.3). This means that sampled data is used to create a surrogate model (an approximation or interpolation of the real objective function) which then is used to find the best point for the next expensive function evaluation.

[4] The design of SRS and CORS however also prevents "clustering".

Section 2.1 shows how surrogate models are constructed with RBF interpolation. However, interpolating data from the noisy function $\tilde{f}$ may result in an oscillating surrogate model corresponding poorly to the true function $f$ (see Sect. 1.2). Therefore an approach for approximation using RBFs is proposed. The approximation allows deviations from the noisy data, enabling the surrogate model to, in a sense, more accurately represent the true function $f$.

Section 2.2 presents the strategy for choosing new points. The idea is to evaluate points such that the surrogate model will be improved in areas close to the optimum. This is measured by the *quality function*, which is an integration over an approximation of the uncertainty decrease in the model multiplied by a weight function depending on the surrogate model's value. The next evaluated point is then selected as the maximizer of the quality function.

In Sect. 2.3 the algorithm is extended to multiobjective optimization. This is done by creating a surrogate model for each of the objective functions, and modifying the weight function so that it uses a measure of distance to the surrogates' Pareto front instead of the surrogate function value. Finally, Sect. 2.4 presents some implementation details.

## 2.1 Creating a surrogate model of the objective function

### *2.1.1 Interpolation as a surrogate model*

If no noise is present in the objective function, an interpolation is a good choice for a surrogate model. The following definitions will prove useful when creating interpolations based on radial basis functions (RBFs). For more details on RBFs and the theory behind them see Wendland (2005).

Let $\pi_n(\mathbb{R}^d)$ denote the space of all polynomials of order less or equal to $n$ in $\mathbb{R}^d$.

**Definition 1** (Conditionally positive definite function) A continuous function $\Phi$: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called conditionally positive definite of order $m$ if, for all $n \in \mathbb{N}$ any $n$ pairwise different points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and all $\boldsymbol{c} \in V_m \backslash \{\boldsymbol{0}^n\}$ it holds that

$$\sum_{j=1}^{n} \sum_{k=1}^{n} c_j c_i \Phi(\boldsymbol{x}_j, \boldsymbol{x}_i) > 0,$$

where

$$V_m = \left\{ \boldsymbol{c} \in \mathbb{R}^n : \sum_{j=1}^{n} c_j p(\boldsymbol{x}_j) = 0, \ \forall p \in \pi_{m-1}(\mathbb{R}^d) \right\}.$$

The radial function $\phi$: $[0, \infty) \to \mathbb{R}$ is called conditionally positive definite of order $m$ in $\mathbb{R}^d$ if $\Phi(\boldsymbol{x}, \boldsymbol{y}) := \phi(\|\boldsymbol{x} - \boldsymbol{y}\|)$ is conditionally positive definite of order $m$.

Note that a conditionally positive definite function of order $l$ is also conditionally positive definite of order $m > l$ as $V_m \subseteq V_l$.

**Table 1** List of common RBFs

| Name | $\phi(r)$ | Order |
|---|---|---|
| Linear | $-r$ | 1 |
| Cubic | $r^3$ | 2 |
| Thin plate spline | $r^2 \log r$ | 2 |
| Multiquadratic | $-\sqrt{r^2+1}$ | 1 |
| Inverse multiquadratic | $1/\sqrt{r^2+1}$ | 0 |
| Gaussian | $\exp(-r^2)$ | 0 |

**Definition 2** The points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ with $n \geq \dim \pi_m(\mathbb{R}^d)$ are called $\pi_m(\mathbb{R}^d)$ unisolvent if the zero polynomial is the only polynomial from $\pi_m(\mathbb{R}^d)$ that vanishes at all of the points in $X$.

The construction of an RBF-based interpolation is then done as follows. Given the $\pi_l(\mathbb{R}^d)$ unisolvent points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and corresponding function values $\boldsymbol{f} = (f_1, \ldots, f_n)^{\mathrm{T}}$, choose a radial conditionally positive definite function $\phi$ (for instance from Table 1) of order $m \leq l+1$. Let $Q = \dim(\pi_{m-1}(\mathbb{R}^d))$ and $\{p_k\}_{k=1}^{Q}$ be the basis of $\pi_{m-1}(\mathbb{R}^d)$. Now define the matrices

$$A_{ij} = \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \quad i, j = 1, \ldots, n,$$
$$P_{ik} = p_k(\boldsymbol{x}_i), \quad i = 1, \ldots, n, \ k = 1, \ldots, Q.$$

Then the linear system

$$\begin{pmatrix} A & P \\ P^{\mathrm{T}} & \mathbf{0}^{Q\times Q} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \mathbf{0}^{Q} \end{pmatrix}, \tag{2}$$

has a unique solution (see below). By solving it the coefficients for the RBF-interpolations are obtained. The interpolation is the function

$$S : \mathbb{R}^d \to \mathbb{R}, \qquad S(\boldsymbol{x}) := \sum_{j=1}^{n} \alpha_j \phi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) + \sum_{k=1}^{Q} \beta_k p_k(\boldsymbol{x}). \tag{3}$$

The following proposition guarantees the unique solution. For a proof see Wendland (2005, Theorem 8.21).

**Proposition 1** *Suppose $\phi$ is conditionally positive definite of order $m$ and $X$ is $\pi_{m-1}(\mathbb{R}^d)$ unisolvent. Then (2) is uniquely solvable.*

In this article we have restricted ourselves to using linear tails ($Q = 2$) in the interpolation (3). Hence conditionally positive functions of order less or equal 2 must be used. Expressing the matrix $P$ rowwise, we then obtain

$$P_{k\cdot} = (1, x_{1,k}, \ldots, x_{j,k}), \quad k = 1, \ldots, d. \tag{4}$$

By demanding that the points $X$ are $\pi_1(\mathbb{R}^d)$ unisolvent (i.e., they do not belong to a common hyperplane) a unique solution to the linear system (2) is guaranteed. Solving the system using the matrix defined in (4) gives the coefficients to the interpolation

in (3). This is the surrogate model to be used in the next step of the algorithm, which is to decide where to perform the next evaluation of the function $f$.

### 2.1.2 Approximation as a surrogate model

When the objective function $f$ is noisy, interpolating the responses of the noisy function $\tilde{f}$ may lead to an oscillating surrogate model that corresponds poorly to the true[5] objective function $f$ (see Fig. 1A). To prevent this type of behavior a surrogate model that deviates from the data points (that is an approximation) will be introduced. In order to present a method for RBF approximation some definitions and theoretical results are necessary. The function space containing all sums of RBFs of the form (3) is called the native space $\mathcal{N}_\phi$. This space is equipped with the following semi norm.[6]

**Definition 3** (Semi norm) Let $S$ be an interpolation (that is functions on the form (3)) with coefficients $\boldsymbol{\alpha}$ and data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. The squared semi norm in the native space is defined as

$$|S|^2_{\mathcal{N}_\phi} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_i \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|).$$

The semi norm will be useful when generalizing the interpolation to an approximation, as it can be seen as a measure of function complexity. The following theorem shows that the interpolation $S$ obtained by solving (2) is the function from $\mathcal{N}_\phi$ that interpolates the data $\{X, \tilde{\boldsymbol{f}}\}$ with the smallest semi norm $|\cdot|_{\mathcal{N}_\phi}$ (for a proof, see Wendland 2005, Theorem 13.2):

**Theorem 1** *Suppose $\Phi \in C(\Omega \times \Omega)$ is a conditionally positive definite function of order $m$. Suppose further that $X$ is $\pi_{m-1}$ unisolvent. Let $S$ be the interpolant obtained by solving (2). Then*

$$|S|_{\mathcal{N}_\Phi} = \text{minimum}\{|\tilde{S}|_{\mathcal{N}_\Phi} : \tilde{S} \in \mathcal{N}_\Phi \text{ with } \tilde{S}(\boldsymbol{x}_j) = \tilde{f}_j,\ j = 1, \ldots, n\}. \tag{5}$$

We will restrict ourselves to utilizing RBFs as our conditionally positive definite functions $\Phi(\boldsymbol{x}, \boldsymbol{y}) := \phi(\|\boldsymbol{x} - \boldsymbol{y}\|)$. A new contribution by this article is the following method for constructing RBF approximations by generalizing the minimization problem (5) to allow deviations from data. The new minimization problem is constructed so that the size of the deviation introduced is balanced against that of the surrogate norm. The problem is to

$$\begin{aligned} \text{minimize} \quad & \eta |S|^2_{\mathcal{N}_\phi} + (1-\eta)\|\boldsymbol{e}\|^2_{l^2}, \\ \text{subject to} \quad & S(\boldsymbol{x}_i) = \tilde{f}_i + e_i, \quad i = 1, \ldots, n, \\ & S \in \mathcal{N}_\phi, \\ & \boldsymbol{e} \in \mathbb{R}^n. \end{aligned} \tag{6}$$

[5] See Sect. 1.2 for a discussion of the true and noisy function.

[6] Wendland (2005, Chap. 10) gives a detailed definition of the native space $\mathcal{N}_\phi$, and establishes the semi norm properties of Definition 3.

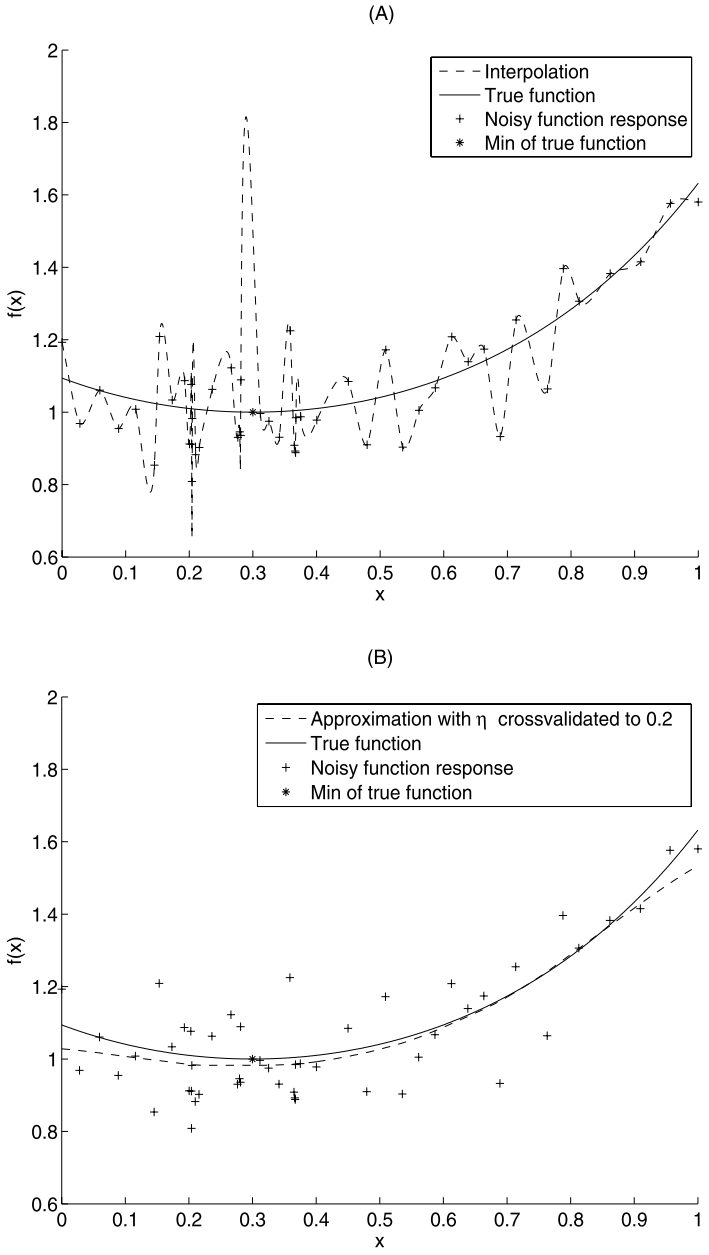


**Fig. 1** (**A**) interpolation of a noisy function response, (**B**) approximation of the same data. The noisy function was obtained by adding normally distributed noise to the smooth true function

The fixed parameter $\eta \in (0, 1)$ controls the balance between reducing the error and reducing the surrogate semi norm. One extreme case is $\eta \to 1$, where minimizing the semi norm is prioritized much higher than minimizing the error, leading to an

approximation with zero semi norm (which is a polynomial in the case of conditionally positive definite RBFs). The other extreme is $\eta \to 0$ which in turn prioritizes the error much higher than the semi norm leading to an interpolation. By choosing $\eta$ somewhere in between a balance is obtained where some error is allowed, but the characteristics of the data points are still present.

Having formulated the approximation as a solution to the problem (6), the question of how to solve this problem and obtain the function $S$ remains. Denote the matrix in (2) by

$$\tilde{A} = \begin{pmatrix} A & P \\ P^{\mathrm{T}} & \mathbf{0}^{Q\times Q} \end{pmatrix}, \quad \tilde{A} \in \mathbb{R}^{(n+Q)\times(n+Q)}.$$

**Proposition 2** *Let $\phi$ be a conditionally positive definite function of order $m$ and $X \subset \mathbb{R}^d$ be $\pi_{m-1}(\mathbb{R}^d)$ unisolvent. Let $B \in \mathbb{R}^{n\times n}$, $B_{ij} = (\widetilde{A}^{-1})_{ij}$ for $i, j = 1, \ldots, n$. The optimal value of $\boldsymbol{e}$ in the problem (6) is obtained by solving the linear system*

$$\left(\frac{\eta - 1}{\eta} I^{n\times n} - B^{\mathrm{T}} A B\right)\boldsymbol{e} = B^{\mathrm{T}} A B \tilde{\boldsymbol{f}}. \tag{7}$$

*Proof* Theorem 1 implies that, given an error vector $\boldsymbol{e}$, the function $S \in \mathcal{N}_\phi$ satisfying the constraints in (6) with least norm is the interpolation. Its coefficients are obtained by solving:

$$\begin{pmatrix} A & P \\ P^{\mathrm{T}} & \mathbf{0}^{Q\times Q} \end{pmatrix}\begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{f}} + \boldsymbol{e} \\ \mathbf{0}^{Q} \end{pmatrix}. \tag{8}$$

According to Proposition 1 there exists a unique solution. Hence, $\boldsymbol{\alpha} = B(\tilde{\boldsymbol{f}} + \boldsymbol{e})$ and $\boldsymbol{\beta} = C(\tilde{\boldsymbol{f}} + \boldsymbol{e})$, where $C \in \mathbb{R}^{d\times n}$, $C_{i-nj} = (\widetilde{A}^{-1})_{ij}$ for $i = n+1, \ldots, n+d$ and $j = 1, \ldots, n$. Using Definition 3, the relation $|S|^2_{\mathcal{N}_\phi} = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) = \boldsymbol{\alpha}^{\mathrm{T}} A \boldsymbol{\alpha}$ is obtained, and the problem (6) reduces to a quadratic programming problem of the following form:

$$\underset{\boldsymbol{e}\in\mathbb{R}^n}{\text{minimize}}\, q(\boldsymbol{e}) = \eta(\tilde{\boldsymbol{f}} + \boldsymbol{e})^{\mathrm{T}} B^{\mathrm{T}} A B(\tilde{\boldsymbol{f}} + \boldsymbol{e}) + (1-\eta)\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}.$$

For functions $g \in C^2$ the convexity of $g$ is equivalent to it having a positive semidefinite Hessian. Since $A$ is a symmetric matrix,

$$\nabla^2 q(\boldsymbol{e}) = 2\eta B^{\mathrm{T}} A B^{\mathrm{T}} + 2(1-\eta) I^{n\times n}. \tag{9}$$

Next the positive definiteness of this Hessian is proven. Letting $\boldsymbol{x} \in \mathbb{R}^n$ be arbitrary, by using the definitions of $B$ and $C$,

$$\begin{pmatrix} A & P \\ P^{\mathrm{T}} & \mathbf{0}^{Q\times Q} \end{pmatrix}^{-1}\begin{pmatrix} \boldsymbol{x} \\ \mathbf{0}^{Q} \end{pmatrix} = \begin{pmatrix} B\boldsymbol{x} \\ C\boldsymbol{x} \end{pmatrix}$$

holds. Multiplying this relation by $\left(\begin{smallmatrix} A & P \\ P^{\mathrm{T}} & \mathbf{0}^{Q\times Q} \end{smallmatrix}\right)$ from the left the following is obtained:

$$\begin{pmatrix} \boldsymbol{x} \\ \mathbf{0}^{Q} \end{pmatrix} = \begin{pmatrix} A & P \\ P^{\mathrm{T}} & \mathbf{0}^{Q\times Q} \end{pmatrix}\begin{pmatrix} B\boldsymbol{x} \\ C\boldsymbol{x} \end{pmatrix}.$$

Considering the lower row of this equation yields:

$$P^{\mathrm{T}} B\boldsymbol{x} = \mathbf{0}^{Q}.$$

Because $\phi$ is conditionally positive and $P^{\mathrm{T}} B\boldsymbol{x} = \mathbf{0}^{Q}$, definition 1 implies:

$$\boldsymbol{x}^{\mathrm{T}} B^{\mathrm{T}} AB\boldsymbol{x} > 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \backslash \{\mathbf{0}^n\}.$$

This proves that $B^{\mathrm{T}} AB$ is a positive definite matrix, and hence the Hessian defined in (9) is positive definite, which in turn implies that $q$ is a (strictly) convex function. For convex functions $q$ in unconstrained domains the minimum is given by the solution to the linear equation

$$\nabla q(\boldsymbol{e}) = \mathbf{0}^n.$$

Inserting, differentiating, and using the symmetry of $A$ gives:

$$\eta(2B^{\mathrm{T}} AB\boldsymbol{e} + 2B^{\mathrm{T}} AB\tilde{\boldsymbol{f}}) + 2(1-\eta)\boldsymbol{e} = \mathbf{0}^n,$$

and the vector $\boldsymbol{e}$ is obtained by solving the linear system (7). As $B^{\mathrm{T}} AB$ is positive definite and $(\eta - 1)/\eta < 0$ the system has a unique solution. □

Having obtained the optimal errors $\boldsymbol{e}$, the surrogate is obtained by solving the system (8), as claimed.

In Fig. 1B a surrogate model is constructed for the same set of points as in Fig. 1A but by using the approximative method with $\eta = 0.2$. We observe that the approximation becomes less oscillating than the interpolation and also resembles the true function more than the interpolation does. Furthermore, because of the less oscillating behavior, locating the minimum of the approximation is often easier than for the interpolation. When we consider a multiobjective optimization problem, reducing oscillations created by noise becomes even more important, as a Pareto front of oscillating functions becomes discontinuous. Such a property would make it difficult for multiobjective solvers to correctly locate the surrogate's Pareto front.

The downside to using approximation as a surrogate model is the introduction of a new parameter that has to be chosen with care: the approximation parameter $\eta$. The next section describes cross-validation, a method for estimating the approximation parameter $\eta$ from data. The method was used when generating the data shown in Fig. 1B.

### *2.1.3 Estimating the approximation parameter η*

A method for estimating $\eta$ is the subject of this section. The method is called *cross-validation* and has its roots in statistics, see e.g. Kohavi (1999). The method is also used in Jones et al. (1998) for another type of parameter estimation. The implementation in this article uses the so-called Leave-one-out cross-validation, the idea of which is the following. Consider a set of points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and corresponding noisy function values $\tilde{\boldsymbol{f}} = (\tilde{f}_1, \ldots, \tilde{f}_n)$. The noisy function values are assumed to be the sum $\tilde{f}_i = f(\boldsymbol{x}_i) + \varepsilon_i$. The first term is the value of the true objective function $f$

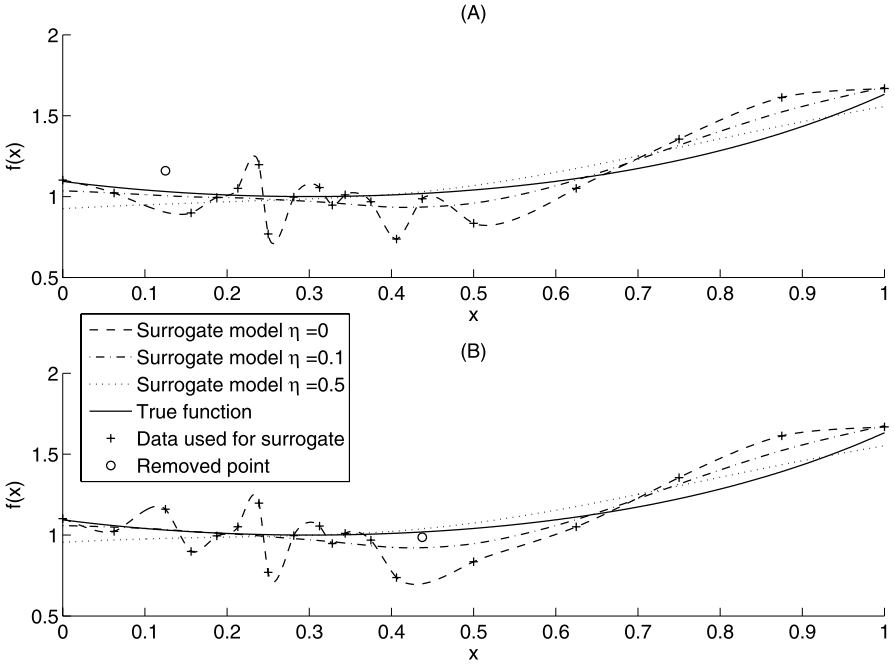


**Fig. 2** Surrogate models obtained by removing one data point; in (**A**) the point at 0.12 was removed whereas in (**B**) it was the point at 0.43. The noisy function used was produced by adding normal distributed noise to a smooth true function

at $\boldsymbol{x}_i$. The second corresponds to the noise and consists of independent and identically distributed random variables $\varepsilon_i$ with zero mean: $E[\varepsilon_i] = 0$. Remove the data point $i$, choose a value of the approximation parameter $\eta$ and construct a surrogate model $S_{\eta\{X\setminus\boldsymbol{x}_i\}}$ of all the remaining points. Evaluate $S_{\eta\{X\setminus\boldsymbol{x}_i\}}$ at the removed point. Figure 2 shows surrogate models obtained by removing one point and using the others to create a surrogate, and doing so for two different points. The best choice of $\eta$ should be that which allows the approximation to predict the value of a missing point with as small an error as possible. Formally, the approximation parameter $\eta$ that solves the following optimization problem is chosen:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{n} (S_{\eta\{X\setminus\boldsymbol{x}_i\}}(\boldsymbol{x}_i) - \tilde{f}_i)^2, \\ &\text{subject to} \quad \eta \in (0, 1). \end{aligned} \tag{10}$$

If removing the point $\boldsymbol{x}_i$ yields a set of points that are not $\pi(\mathbb{R}^d)$ unisolvent, then the term is left out from the sum. In order to decrease computational time, a list of $\eta$ values was evaluated, and the one with the lowest objective value was chosen. This procedure is repeated after every newly evaluated point.

The success of this estimation method relies on the assumption that a surrogate with a value of $\eta$ that can enable good predictions for previously chosen points will also provide good predictions for yet unsampled points. Since the noise is assumed

to be a random and independent quantity, it can not be predicted. Hence the surrogate $S_{\eta\{X\setminus \boldsymbol{x}_i\}}$, where $\eta$ has been obtained by cross-validation, can at best predict the true function.

### 2.2 A strategy for choosing the next evaluation point

All algorithms that handle expensive functions have to keep the number of function evaluations at a minimum. The key issue for algorithms based on surrogate models is to find a balance between local and global searches (see Sect. 1.3, step 2 of a two-stage approach). We will obtain this balance by maximizing an auxiliary function that measures the improvement in certainty gained weighted against surrogate function value.

As uncertainty increases with distance to evaluated points $X$, the measure of uncertainty[7] at a point $\boldsymbol{y}$ is defined as:

$$U_X(\boldsymbol{y}) = \underset{\boldsymbol{x}\in X}{\text{minimum}} \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{11}$$

The goal of selecting new points is not to minimize the overall uncertainty, since the accuracy of the surrogate in areas with high function values is not important. Therefore a measure of the form $\int_\Omega U_X(\boldsymbol{x})\omega(S(\boldsymbol{x}))dV(\boldsymbol{x})$ of the total uncertainty weighted against function value is introduced. The weight function $\omega(S(\boldsymbol{x}))$ (the detailed construction is described below) gives points with high surrogate function value $S$ a low weight and points with low surrogate value a high weight. When choosing a new point to evaluate, the point is chosen such that the *overall decrease* in weighted uncertainty is maximized.

**Definition 4** (Quality function) Let $\Omega \subseteq \mathbb{R}^d$ be the parameter space, $\omega : \mathbb{R} \to \mathbb{R}^+$ be the weight function that prioritizes regions of interest for the optimization, and $U_X : \mathbb{R}^d \to \mathbb{R}^+$ be a measure of uncertainty given by (11). The quality function is then given by:

$$Q : \mathbb{R}^d \to \mathbb{R}, \qquad Q(\boldsymbol{y}) = \int_\Omega (U_X(\boldsymbol{x}) - U_{X\cup \boldsymbol{y}}(\boldsymbol{x}))\omega(S(\boldsymbol{x}))dV(\boldsymbol{x}). \tag{12}$$

An important property of the function $Q$ in (12), which partly motivates its construction, is that the integrand vanishes outside of the domain

$$\Omega_{\boldsymbol{y}} = \left\{\boldsymbol{x} \in \Omega : \|\boldsymbol{x} - \boldsymbol{y}\| \leq \min_{\boldsymbol{w}\in X} \|\boldsymbol{x} - \boldsymbol{w}\|\right\}.$$

It is therefore sufficient to perform an integration over $\Omega_{\boldsymbol{y}}$, which saves computational time. Finally, the point to evaluate next is obtained by solving the following

[7] Regis and Shoemaker (2005) and Regis and Shoemaker (2007) use the distance to closest evaluated point in a similar way.

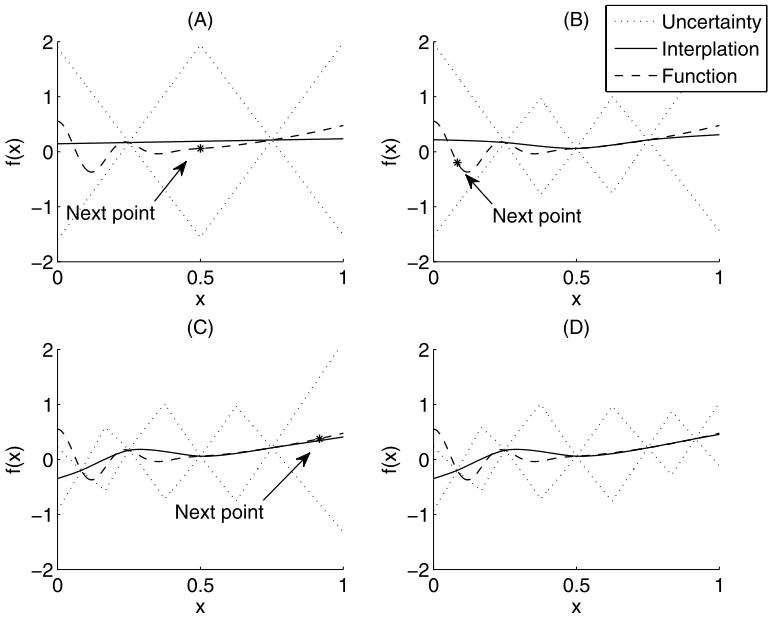


**Fig. 3** A sequence of points chosen by maximizing the quality function, treating the whole domain as equally important

optimization problem[8] using conventional optimization algorithms:

$$\underset{\boldsymbol{y}\in\Omega}{\text{maximize}}\, Q(\boldsymbol{y}). \tag{13}$$

Figure 3 illustrates the process of choosing the next point to evaluate. For illustration purposes the weight function $\omega \equiv 1$ was used, hence treating all parts of the domain as equally important.

Returning to the construction of the weight function, recall that the weight function should be constructed to give small weights to high surrogate function values and large weights to low surrogate function values. Furthermore, considering that we are optimizing a black box function, all parts of the parameter domain have to be regarded as potentially containing a minimum. This is the motivation for placing the following demands on the weight function $\omega$:

- no areas are completely disregarded: $\omega(z) > 0$, $\forall z \in \mathbb{R}$;
- areas with low surrogate function values are weighted higher than areas with high surrogate function values: $\omega$ is strictly decreasing.

A choice of weight function that satisfies these conditions is

$$\omega(z) = \exp\left(-\gamma \frac{z - \min_{\boldsymbol{x}\in\Omega} S_X(\boldsymbol{x})}{\max_{\boldsymbol{x}\in\Omega} S_X(\boldsymbol{x}) - \min_{\boldsymbol{x}\in\Omega} S_X(\boldsymbol{x})}\right), \tag{14}$$

[8]This is in itself a challenging problem in general; a simplification compared to the original problem, however, is that it is not expensive or noisy.

where $\gamma \geq 0$. The function is constructed so that $\omega(\min S_X(\boldsymbol{x})) = 1$ and $\omega(\max S_X(\boldsymbol{x})) = \exp(-\gamma)$. The parameter $\gamma \geq 0$ controls the balance between local and global search. It has a purpose similar to that of the parameter $\beta$ in the algorithm of Regis and Shoemaker (2005). If $\gamma = 0$, we completely ignore the surrogate model and aim to minimize the total uncertainty (i.e. space filling), whereas $\gamma \to \infty$ corresponds to simply choosing $\arg\min S_X(x)$ as the new point to evaluate, hence trusting the model completely.

Although one choice of the value of the parameter $\gamma$ is sufficient to induce both local and global searches, in practice we have noted that cycling[9] between the evaluations generally provides the best results. Cycling a parameter controlling the global–local search is common practice among response surface algorithms.[10]

### 2.3 Multiobjective optimization

The purpose of this section is to extend the algorithm previously described to multiobjective optimization. First a surrogate model $S_i$ is created for each objective function as described in Sect. 2.1. The second and major modification is to adapt the weight function $\omega$ (see Definition 4) so that it becomes relevant for multiobjective optimization. In order to do so, we first introduce some theory.

The general multiobjective optimization problem, to

$$\begin{aligned} &\text{minimize} && \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^{\mathrm{T}}, \\ &\text{subject to} && \boldsymbol{x} \in \Omega, \end{aligned}$$

can be solved using different approaches, depending on information known about the preferences of the decision maker (see Miettinen 1999 for a thorough treatment). Let first $Z_f = \{\boldsymbol{f}(\boldsymbol{x}) : \boldsymbol{x} \in \Omega\}$ denote the feasible objective set. The focus of this article is to find the Pareto front $P(Z_f) \subset \mathbb{R}^m$.

**Definition 5** (Pareto optimality) An objective vector $\boldsymbol{z}^* \in Z_f$ is Pareto optimal if there does not exist another vector $\boldsymbol{z} \in Z_f$ such that $z_i \leq z_i^*$ for all $i = 1, \ldots, m$ and $z_j < z_j^*$ for at least one index $j \in \{1, \ldots, m\}$.

The objective vector $\boldsymbol{z}^*$ is *weakly* Pareto optimal if there does not exist another vector $\boldsymbol{z} \in Z_f$ such that $z_i < z_i^*$ for all $i = 1, \ldots, m$.

**Definition 6** (Pareto optimal set) The Pareto optimal objective set (or, Pareto front) $P(Z_f) \subseteq Z_f$ is defined as the set consisting of all Pareto optimal objective vectors in $Z_f$.

The *weakly* Pareto optimal objective set $P_w(Z_f) \subseteq Z_f$ is defined as the set consisting of all weakly Pareto optimal objective vectors in $Z_f$.

When the objective is to find the Pareto front, an objective vector closer to or at the surrogate models' Pareto front is naturally considered more important than a

[9] Cycling refers to changing the parameter according to a given finite sequence and restarting the sequence when the last element is reached.

[10] Cycling is used in Gutmann (2001), Regis and Shoemaker (2005) and Regis and Shoemaker (2007).

vector further away. Further, as the objectives are black box functions, all points not evaluated may prove to be Pareto optimal. The demands on the weight function $\omega$ hence are the following:

- no areas are completely disregarded: $\omega(\boldsymbol{z}) > 0; \forall \boldsymbol{z} \in Z_f$;
- if $\boldsymbol{z}^1 \in Z_f$ is closer to the Pareto front $P(Z_f)$ than $\boldsymbol{z}^2 \in Z_f$, then $\omega(\boldsymbol{z}^1) > \omega(\boldsymbol{z}^2)$ should hold.

Unfortunately we do not have access to the set $Z_f$, so instead we have to utilize the surrogate feasible set $Z_S$. Define the extended Pareto front as

$$P_e(Z_S) = P_w(Z_S + \mathbb{R}^m_+). \tag{15}$$

If a point is close to $P_e(Z_S)$ it is good candidate being close to the $P(Z_f)$ and the model uncertainty at that point should be low. The weight function is therefore constructed so that it decreases with the distance to the extended front $P_e(Z_S)$ and is greater than zero for all areas:

$$\omega(\boldsymbol{z}) = \exp\left(-\gamma \frac{\text{dist}_S(\boldsymbol{z})}{\text{dist}_S(\tilde{\boldsymbol{z}})}\right). \tag{16}$$

The vector $\tilde{\boldsymbol{z}}$ is used for scaling and is defined by $\tilde{\boldsymbol{z}} = \max_{\boldsymbol{z} \in Z_S} \text{dist}_S(\boldsymbol{z})$. The distance function $\text{dist}_S : \mathbb{R}^m \to \mathbb{R}_+$ is defined as the smallest Euclidean distance between the point $\boldsymbol{z}$ and a point at the extended Pareto front $P_e(Z_S)$ of the surrogate functions in a domain scaled to unity. Figure 4 gives a graphical representation of the distance function and the extended Pareto front for two objective functions. The distance function is defined as

$$\text{dist}_S(\boldsymbol{z}) = \min_{\boldsymbol{z}^* \in P_e(Z_S)} \sqrt{\sum_{j=1}^{m} \left(\frac{z_j^* - z_j}{z_{j,\max} - z_{j,\min}}\right)^2}, \tag{17}$$

where $z_{j,\max} = \max_{\boldsymbol{x} \in \Omega} S_j(\boldsymbol{x})$ and $z_{j,\min} = \min_{\boldsymbol{x} \in \Omega} S_j(\boldsymbol{x})$. Analogous to the case of a single objective optimization the weight $\omega$ is constructed such that it varies between 1 and $e^{-\gamma}$. Indeed, objective vectors belonging to the Pareto front of the surrogate $\boldsymbol{z} \in P(Z_S)$ (which implies that they belong to the extended front $\boldsymbol{z} \in P_e(Z_S)$) obtain

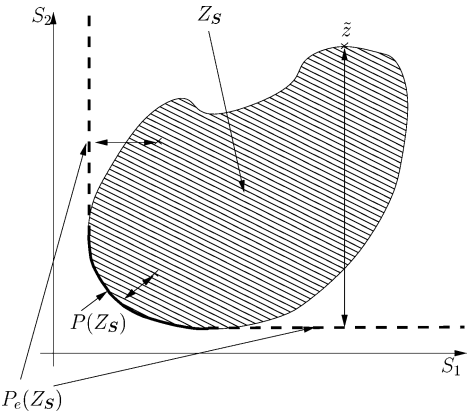


**Fig. 4** Illustration of the distance function to the extended Pareto front. The *dashed line* symbolizes the part that the Pareto front is extended with to obtain the extended Pareto front. The value of the distance function at three different points (including $\tilde{\boldsymbol{z}}$) is illustrated by the double *arrows*

**Algorithm 1** Solver for single objective optimization

```
generate initial points X                                   ▷ initial points
evaluate objective function f at X to obtain f
while maximal number of evaluations not exceeded do
    if interpolation then                                   ▷ create surrogate model
        f̂ = f
    else if approximation then
        cross-validate to obtain η
        obtain e by solving linear system (7)
        f̂ = f + e
    end if
    obtain α, β by solving linear system (2)
    use α, β to create surrogate S by (3)
    minimize S
    choose next γ from cycle
    obtain y by solving (13)                                ▷ find point to evaluate
    evaluate function f at y
    add f(y) to f and y to X
end while
k = arg min_{i=1,...,n}{f_i}
f_min = f_k
x_min = x_k
return x_min, f_min, S
```

the weight $\omega(z) = 1$, while the feasible point farthest away, $\tilde{z}$, fulfills $\omega(\tilde{z}) = e^{-\gamma}$. As in single objective optimization, the parameter $\gamma \geq 0$ balances global and local searches. The extremal choice $\gamma = 0$ corresponds to a global search and $\gamma \to \infty$ corresponds to minimizing the uncertainty at the Pareto front (hence a local search).

The Pareto front of the surrogate models can be found by using a multiobjective solver that is adapted to solving the original problem without considering noise and expensiveness. This is often in itself a challenging problem as the surrogates may be highly nonlinear and multi modal. To compute the vectors $z_{\max}$ and $z_{\min}$ an external solver for single objective optimization must be used. The main implementation difficulty lies in constructing the distance function $\text{dist}_S$ together with the extended Pareto front $P_e(Z_S)$; see Sect. 4 for a further discussion on this.

### 2.4 Implementation

Below follows a short overview of the implementation; for more details, see Rudholm and Wojciechowski (2007).

The complete[11] single objective algorithm is presented in Algorithm 1, using notation from previous parts of this section. The goal of the single objective algorithm is to solve the problem defined in (1). The task is to find the minimum of $f$, that is

[11] Transformations and other implementation details are left for presentation transparency.

**Algorithm 2** Solver for multiobjective optimization

```
generate initial points X in Ω                                  ▷ initial points
for j = 1 : m do
    evaluate objective function j at X to obtain f_j
    create surrogate model S_j
end for
solve multiobjective problem for surrogate functions S = (S_1, ..., S_m)
while maximal number of evaluations not exceeded do
    chose next γ from cycle
    obtain y by solving (13)                                    ▷ find point to evaluate
    add y to X
    for j = 1 : m do
        evaluate function f_j at y
        update surrogate model S_j
    end for
    solve multiobjective problem for surrogate functions S = (S_1, ..., S_m)
end while
return P(Z_S), P_Ω(Z_S), S
```

$f_{\min}$, and the point where it is obtained, that is $\boldsymbol{x}_{\min}$. The different parts of the two-stage approach (see Sect. 1.3) can be identified according to the comments. As external solver TOMLAB's[12] implementation of DIRECT followed by SNOPT[13] was used. During all test runs the RBFs used were thin plate splines. It should also be mentioned that the implementation performs a unit transformation of the parameter domain $\Omega \to [0, 1]^d$. Furthermore, similar to the median cut strategy performed in Gutmann (2001) and Regis and Shoemaker (2005), a logarithmic transformation of the objective functions is performed. The transformation is adapted to have a small impact close to the current surrogate minima and reducing larger function values.

In Algorithm 2, the algorithm for multiobjective optimization is described. The method for creating the surrogate models is the same as in the single objective case, and is hence not described in detail. The output of the multiobjective algorithm (the genetic solver MultiOb was used, see Hanne 2006) is the Pareto optimal objective set $P(Z_{\boldsymbol{S}})$ of the final surrogate models and the corresponding Pareto optimal parameter set $P_\Omega(Z_{\boldsymbol{S}}) := \{\boldsymbol{x} \in \Omega : \boldsymbol{S}(\boldsymbol{x}) \in P(Z_{\boldsymbol{S}})\}$. As external single objective solver TOMLAB's implementation of DIRECT followed by SNOPT was used.

## 3 Results and comparisons

The purpose with this section is to evaluate the performance of the algorithm on analytical test functions found in the literature. The algorithm is evaluated on deterministic functions, functions disturbed by noise and deterministic functions with multiple

[12] Commercial optimization software developed by Tomlab Optimization Inc.

[13] Commercial software package for nonlinear problems included in TOMLAB.

**Table 2** The Dixon–Szegö test bed. Each function has one global minimum except for Branin, which has three

| Function | $d$ | # local min | Domain |
|---|---|---|---|
| Branin | 2 | 3 | $[-5, 10] \times [0, 15]$ |
| Goldstein–Price | 2 | 4 | $[-2, 2]^2$ |
| Hartman 3 | 3 | 4 | $[0, 1]^3$ |
| Shekel 5 | 4 | 5 | $[0, 10]^4$ |
| Shekel 7 | 4 | 7 | $[0, 10]^4$ |
| Shekel 10 | 4 | 10 | $[0, 10]^4$ |
| Hartman 6 | 6 | 4 | $[0, 1]^6$ |

objectives. The impact on the final result of the initial point strategy is also examined. To test the algorithm on a real simulation problem, a multiobjective optimization of the simulation of a heater element was also performed.

### 3.1 Results for deterministic functions

The suite of deterministic functions is part of the well-known test bed of Dixon and Szegö (see Dixon and Szegö 1978), summarized in Table 2. The test functions are subject to box constraints only.

The algorithm qualSolve was compared with EGO (originally presented in Jones et al. 1998, here using TOMLAB's[14] implementation by Björkman and Holmström 2000), Gutmann's algorithm (originally presented in Gutmann 2001), CORS (originally presented in Regis and Shoemaker 2005) and rbfSolve (TOMLAB's improvement of Gutmann's algorithm by Björkman and Holmström 2000) on the seven test functions. All algorithms are designed to handle expensive objective functions and hence are suited for comparison. During the run of qualSolve the parameter $\gamma$ was cycled over the values $(0, 10, 20, 20, 50, 50, \infty)$, where '$\infty$' simply means evaluating the surrogate minimum without taking the uncertainty under consideration. Similarly as for the CORS algorithm the performance and behavior of the algorithm is dependent on the cycle used. As the functions are deterministic the surrogate model chosen was an interpolation. As measure of progress, following Gutmann (2001), the relative error

$$e_i = \frac{|f_i - f^*|}{|f^*|},$$

was used. Here, $f_i$ denotes the current evaluated function value and $f^*$ denotes the global minimum objective value. The functions belonging to the Dixon–Szegö test bed all have non-zero optimal values, making this termination criterion feasible. The algorithms were terminated when the relative error $e_i$ reached below 1%, or when the number of function evaluations exceeded 150, whichever came first.

Two choices of initial points were used. The first was to use all corner points of the domain, resulting in $2^d$ initial points (following Gutmann 2001). To see how much of an impact the initial points have on the outcome, a second run was performed using

[14] Commercial optimization software developed by Tomlab Optimization Inc.

**Table 3** Dixon–Szegö test functions evaluated using the $2^d$ corners of the domain as initial points. Shown is the number of function evaluations until the relative error reached below 1%. The best values are shown in bold. '–' denotes that the relative error never reached below 1% during the 150 functions evaluations

| Function | qualSolve | Gutmann | rbfSolve | EGO | CORS |
|---|---|---|---|---|---|
| Branin | 32 | 44 | 59 | **21** | 34 |
| Goldstein–Price | 60 | 63 | 84 | 125 | **49** |
| Hartman 3 | 46 | 43 | 18 | **17** | 25 |
| Shekel 5 | 70 | 76 | – | – | **41** |
| Shekel 7 | 85 | 76 | – | – | **46** |
| Shekel 10 | 71 | **51** | – | – | **51** |
| Hartman 6 | 99 | 112 | 109 | **92** | 108 |

**Table 4** Dixon–Szegö test functions evaluated using LHD of size $d + 1$ as initial points. Shown for each algorithm are the percentage of runs that reached 1% relative error in less than 150 function evaluations, the mean number of evaluations to reach the target and the standard deviation. Runs that did not reach the target within 150 function evaluations are not included in the mean. The lowest mean is indicated in bold. The total number of runs for each function and algorithm was 20

| Function | qualSolve | rbfSolve | EGO |
|---|---|---|---|
| Branin | 100% 26.9 (4.5) | 100% 62.7 (1.3) | 100% **23.0** (0.0) |
| Goldstein–Price | 100% **30.4** (11.0) | 100% 63.0 (0.0) | 0% |
| Hartman 3 | 100% 38.8 (16.9) | 100% **28.0** (0.0) | 100% 31.0 (0.0) |
| Shekel 5 | 30% **61.0** (10.9) | 0% | 0% |
| Shekel 7 | 60% **66.0** (7.6) | 0% | 0% |
| Shekel 10 | 60% **78.0** (39.8) | 0% | 0% |
| Hartman 6 | 95% **50.7** (14.7) | 100% 122.0 (0.0) | 0% |

$d + 1$ initial points by randomly generating Latin Hypercube Designs (LHDs, see Ye et al. 2000 for further information) and choosing the design with the largest minimal distance between two neighboring points from the designs satisfying $\pi_1$ unisolvence. 20 runs were performed for each test function and algorithm. The results are shown in Tables 3 and 4; it is apparent that initial points do play a big role in the speed of convergence. Note further that some runs did not finish in less than 150 iterations.

To give a graphical illustration of the differences between the algorithms we ran qualSolve, EGO, Gutmann's algorithm (as implemented in TOMLAB) and rbfSolve on the 'Tilted' Branin[15] test function (see Fig. 5). Each algorithm was allowed to perform 30 function evaluations. Figure 6 shows the points evaluated by the algorithms and interpolations created from them using a thin plate spline as RBF. It should be pointed out that EGO does not use RBFs to construct the response surface (it uses kriging) and rbfSolve uses a cubic RBF. Still, interpolating using thin plate splines gives a hint on how response surfaces based on these points will appear.

[15]The Branin test function was tilted to obtain only one global minimum instead of three; see Huang et al. (2006).

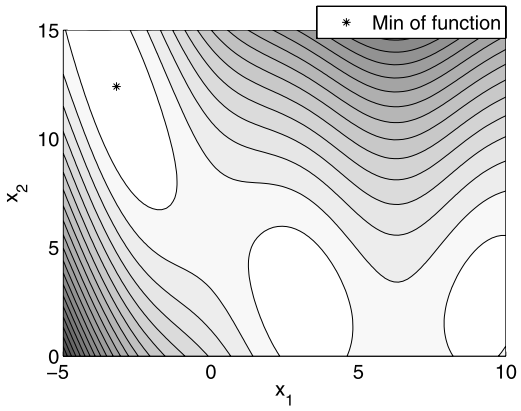


**Fig. 5** 'Tilted' Branin test function

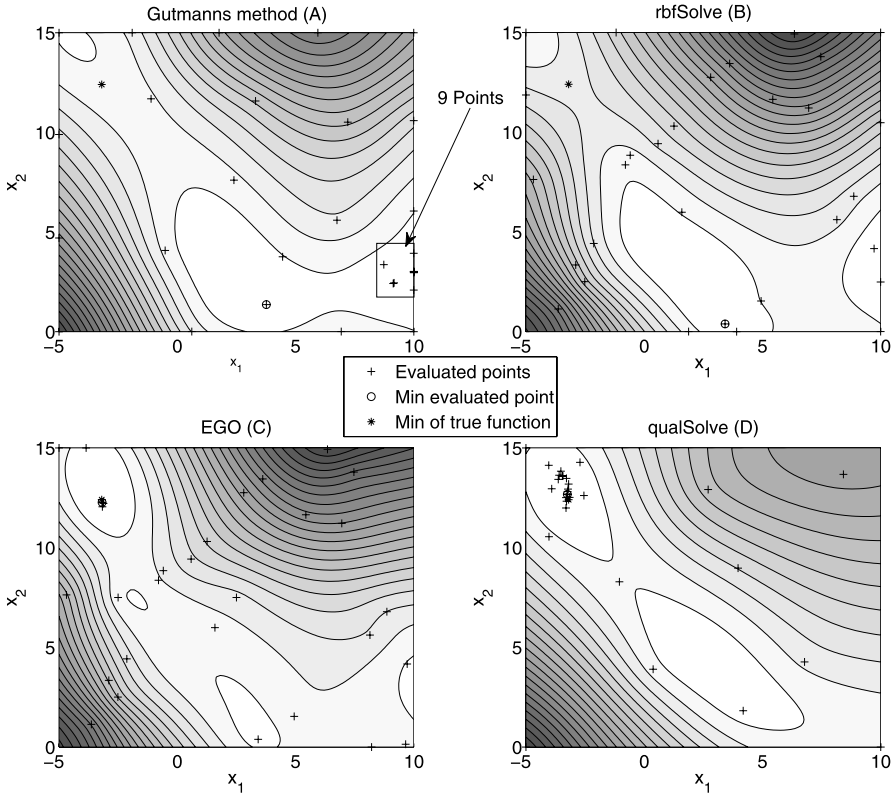


**Fig. 6** Interpolations created from points evaluated by different methods, when solving the 'Tilted' Branin test function

Comparing Figs. 6A and D shows that qualSolve increases the density of evaluated points in a continuous manner when approaching the minimum, whereas Gutmann's algorithm tends to mix very global searches with a very local clustering of evalu-

ated points (consider the points inside the box, where it is impossible to distinguish individual points). This behavior is not shared by rbfSolve. Instead, many evaluated points seem to be placed in areas with high function values without any focus on local search. For both algorithms the result is a very low number of points in the area around the global minimum of the function. EGO, on the other hand, succeeds in locating the global minimum. Still, it could be argued that the transition between the globally spread points and the locally spread points around the minimum is worse than that of qualSolve.

### 3.2 Results for non-deterministic functions

In order to investigate the performance of qualSolve on non-deterministic test functions (test functions containing noise), the test bed presented in Table 5 was used as the true[16] functions. The noisy functions were created by adding a normally distributed random value. For details on the test functions see Huang et al. (2006).

The qualSolve algorithm was compared with SKO (see Sect. 1.3), which is an extension of EGO designed to handle expensive and noisy objective functions. To stress the importance of handling noise, comparisons with the TOMLAB solver rbfSolve were also performed, although this algorithm was not developed to handle noisy functions.

Following Huang et al. (2006), a measure of convergence is defined:

$$G_i = \frac{f(\boldsymbol{x}_{\text{in}}) - f(\boldsymbol{x}_i)}{f(\boldsymbol{x}_{\text{in}}) - f^*}, \tag{18}$$

where $\boldsymbol{x}_i$ is the minimum proposed by the algorithm at the $i$:th iteration and $f$ is the true function. The point $\boldsymbol{x}_{\text{in}}$ is the median initial point $\boldsymbol{x}_{\text{in}} = \arg\text{median}_{\boldsymbol{x}\in X_{\text{in}}} f(\boldsymbol{x})$, where $X_{\text{in}}$ is the set of initial points. The measure is constructed such that $G_i \leq 1$ for all $i$. Reaching $G_i = 1$ means that the global minimum has been found.

The parameter $\gamma$ was cycled between $(0, 10, 20, 20, 50, 50)$. The reason for omitting the '$\infty$' term in contrast to the deterministic cycle from Sect. 3.1 is that we wish to find the minimum of the true function $f$ and evaluating the noisy function $\tilde{f}$ at a suspected minimum does not (in contrast to the deterministic case) provide us with

**Table 5** Test functions used as true functions for non-deterministic algorithm evaluation. The Branin function has been tilted so that only one local minimum is global. The test function Ackley 5 contains an oscillating term, and is therefore littered with local minima

| Function | $d$ | # local minima | Domain |
|---|---|---|---|
| Six-hump camel back | 2 | 6 | $[-1.6, 2.4] \times [-0.8, 1.2]$ |
| 'Tilted' Branin | 2 | 3 | $[-5, 10] \times [0, 15]$ |
| Hartman 3 | 3 | 4 | $[0, 1]^3$ |
| Ackley 5 | 5 | – | $[-1.6, 2.4]^5$ |

[16] See Sect. 1.2 for a discussion of true and noisy functions.

**Table 6** Various test functions with LHD of size $11d$ as initial points. Noise distributed with $N(0, \sigma^2)$ was added. Shown are the means and standard deviations (in parentheses) of the number of function evaluations necessary for reaching $G_i \geq 0.99$. The lowest mean is indicated in bold. The percentage is the number of runs that reached this target. The maximal amount of evaluations was 200 for all functions, except for the two Six-hump camel back ones which were only allotted 100 evaluations. The total number of runs for each function and algorithm was 50. Data for SKO is taken from Huang et al. (2006), whereas data for rbfSolve was obtained by performing runs with TOMLAB's implementation

| Function | $\sigma$ | qualSolve | SKO | rbfSolve |
|---|---|---|---|---|
| S.-h. c. b. | 0.12 | 100% 31.4 (11.2) | 100% **29.2** (5.7) | 94% 30.1 (6.4) |
| S.-h. c. b. | 0.24 | 92% 34.7 (12.2) | 94% **29.4** (6.6) | 86% 37.4 (15.4) |
| 'Tilted' B. | 2.0 | 100% 58.4 (25.1) | 98% **28.4** (5.3) | 80% 60.7 (19.2) |
| Hartman 3 | 0.08 | 100% 61.2 (22.9) | 96% **45.4** (7.9) | 100% 71.3 (12.0) |
| Ackley 5 | 0.06 | 100% 100.5 (16.9) | 94% **98.9** (5.6) | 12% 148.3 (31.1) |

**Table 7** Various test functions with Latin Hypercube Design of size $d + 1$ as initial points. See caption of Table 6

| Function | $\sigma$ | qualSolve | rbfSolve |
|---|---|---|---|
| Six-hump camel back | 0.12 | 100% **17.2** (10.9) | 96% 33.0 (13.1) |
| Six-hump camel back | 0.24 | 98% **21.9** (16.0) | 84% 35.2 (18.2) |
| 'Tilted' Branin | 2.0 | 98% **47.3** (22.5) | 42% 71.8 (26.3) |
| Hartman 3 | 0.08 | 100% 37.1 (25.7) | 100% **32.9** (15.5) |
| Ackley 5 | 0.06 | 100% **59.6** (18.4) | 10% 103.0 (23.9) |

a certain value of $f$ at that point. The approximation parameter $\eta$ was chosen by cross-validation among the values ($10^{-8}$, $10^{-7}$, $10^{-6}$, $10^{-5}$, $10^{-4}$, $10^{-3}$, $10^{-2}$, 0.1, 0.2).

Two choices of initial points were used. The first was $11d$ LHD initial points, as in Huang et al. (2006). This strategy of course yields many more than the $d + 1$ required by qualSolve. Hence, the second choice was to use $d + 1$ LHD initial points. The algorithms were run until $G_i \geq 0.99$ or a maximal number of function evaluations was reached. If the value of $G_i$ did not reach 0.99 the run was considered as not converged. Table 6 shows the results obtained when $11d$ initial points were used. SKO attained the lowest mean value, but qualSolve had a higher convergence percentage on all test functions except on the Six-hump camel back. The poor performance of rbfSolve illustrates the necessity of handling noise explicitly. In Table 7 the number of initial points was decreased to $d + 1$. The performance of qualSolve was considerably improved yielding the best results overall.

### 3.3 Results for vector-valued functions

The success of a multiobjective optimization algorithm for expensive function evaluations is somewhat harder to quantify than that of a single objective algorithm. One of the few algorithms designed for these problems is *ParEGO* Knowles (2006). Knowles

also compares the performance of ParEGO to that of NSGA-II, a genetic algorithm, showing promising results. Unfortunately only relative figures of merit were used.

In light of this, what follows is a simple comparison of the performance of qualSolve and MATLABs genetic algorithm for multiobjective problems *gamultiobj*. The algorithms were run twice on each test function, the runs were terminated after 100 and 250 function evaluations respectively. For qualSolve the $\gamma$ parameter was cycled between the values (0, 10, 10, 10, 20, 20, 20, 20, 20, 20, 50, 50, 50, 50, 50, 50, 50, $\infty$, $\infty$, $\infty$). The $\infty$ symbol here indicates choosing a point on the surrogate's Pareto front with the largest uncertainty. The cycle may be recognized as a prolonged version of the cycle used on single objectives in Sect. 3.1, motivated by the fact that finding a whole set (the Pareto front) generally requires more function evaluations than finding a point (the minimum). Most multiobjective test problems, including the widely used DTLZ test suite in Deb et al. (2001), have been developed for the evaluation of algorithms for inexpensive functions and are hence extremely challenging when only a limited number of function evaluations is allowed. To the authors' knowledge, there is no test suite developed for expensive problem algorithms; hence we have chosen to use only three test functions of mixed difficulty; this is therefore by no means an extensive study. OKA1 was used as an example of a challenging test function, Kursawe was of medium difficulty whereas the modified Deb bimodal was chosen as an example of a smooth test function. The parametric domain for OKA1 and Modified Deb bimodal is two dimensional while Kursawe is three dimensional. The test functions selected all supply two objective functions, and have not been subjected to any noise. Here follows the analytical expressions of the test functions used; 3D plots of the two-dimensional functions can be seen in Figs. 7–8.

Kursawe taken from Kursawe (1991):

$$f_1(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right),$$

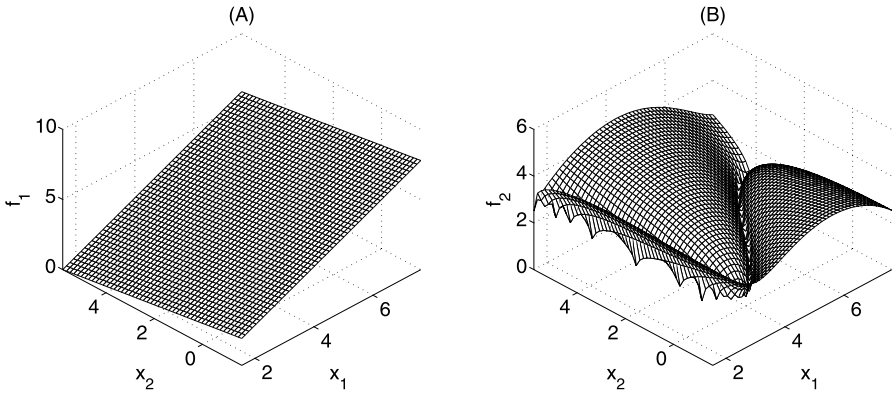


**Fig. 7** The objective functions of the OKA1 test function. The bottom of the valley in (**B**) is strongly oscillating

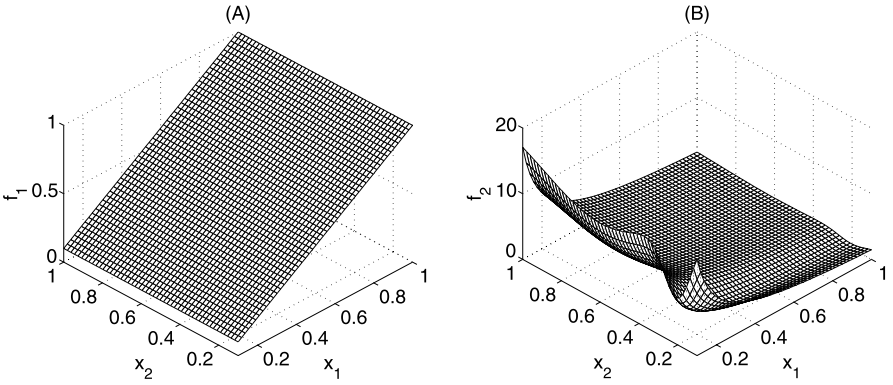


**Fig. 8** The objective functions of the modified Deb bimodal test function

$$f_2(\boldsymbol{x}) = \sum_{i=1}^{n}(|x_i|^{0.8} + 5\sin(x_i)^3),$$
$$\boldsymbol{x} \in [-5, 5]^3.$$

OKA1 taken from Knowles (2006) (Fig. 7):

$$f_1(\boldsymbol{x}) = x_1',$$
$$f_2(\boldsymbol{x}) = \sqrt{2\pi} - \sqrt{|x_1'|} + 2|x_2' - 3\cos(x_1') - 3|^{\frac{1}{3}},$$
$$x_1' = \cos\left(\frac{\pi}{12}\right)x_1 - \sin\left(\frac{\pi}{12}\right)x_2,$$
$$x_2' = \sin\left(\frac{\pi}{12}\right)x_1 + \cos\left(\frac{\pi}{12}\right)x_2,$$
$$x_1 \in \left[6\sin\left(\frac{\pi}{12}\right), 6\sin\left(\frac{\pi}{12}\right) + 2\pi\cos\left(\frac{\pi}{12}\right)\right],$$
$$x_2 \in \left[-2\pi\sin\left(\frac{\pi}{12}\right), 6\cos\left(\frac{\pi}{12}\right)\right].$$

Deb bimodal taken from Deb (1999) (Fig. 8):

$$f_1(\boldsymbol{x}) = x_1,$$
$$f_2(\boldsymbol{x}) = \frac{g(\boldsymbol{x})}{x_1},$$
$$g(\boldsymbol{x}) = 2 - \exp\left(-\left(\frac{x_2 - 0.2}{a}\right)^2\right) - 0.8\exp\left(-\left(\frac{x_2 - 0.6}{0.4}\right)^2\right),$$
$$\boldsymbol{x} \in [0.1, 1]^2;$$

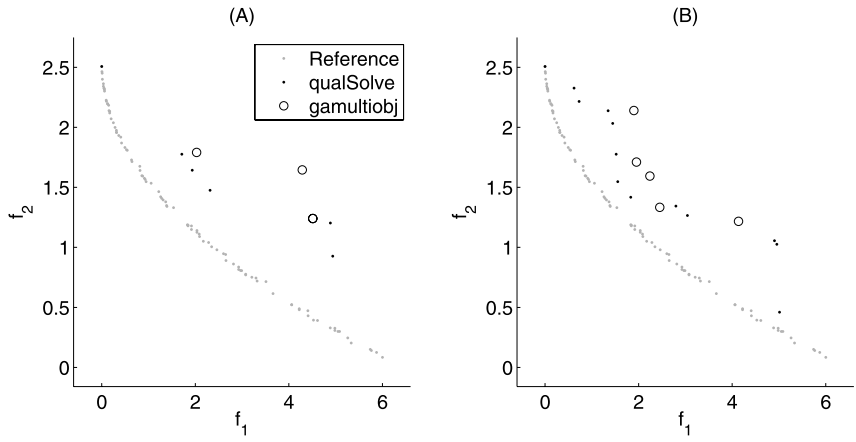


**Fig. 9** The Pareto fronts of the test function OKA1. The *gray dots* represent the reference Pareto front and the *black dots* the non-dominated points returned by qualSolve. The *rings* represent points returned by gamultiobj. Figures (**A**) and (**B**) represent the results after 100 and 250 function evaluations respectively, with 6 respectively 13 non-dominated points returned by qualSolve

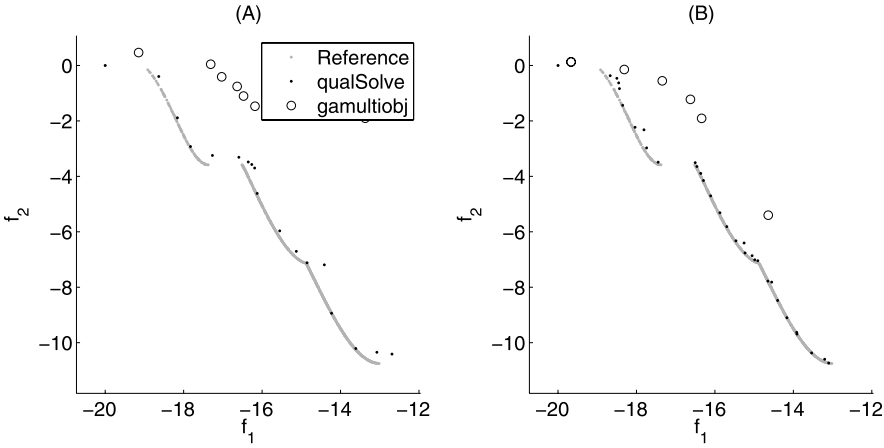


**Fig. 10** The Pareto fronts of the test function Kursawe. The number of non-dominated points produced by qualSolve was 18 after 100 evaluations and 32 after 250 evaluations. See caption of Fig. 9

where the original value $a = 0.004$ has been modified to $a = 0.1$ in order to widen the "valley" created by the exponential function and obtain a very smooth test function. Figures 9–11 depict the resulting non-dominated points returned by the algorithms after 100 and 250 function evaluations, respectively. MultiOb (see Hanne 2006) was used to produce a reference Pareto front; multiple runs were performed on each test function and the results merged into a single Pareto front. This procedure amounted to a total of 250,000 function evaluations for each test function. In each figure the number of non-dominated points returned by qualSolve is reported. QualSolve performs well on Kursawe and modified Deb bimodal, whereas it fails to provide the

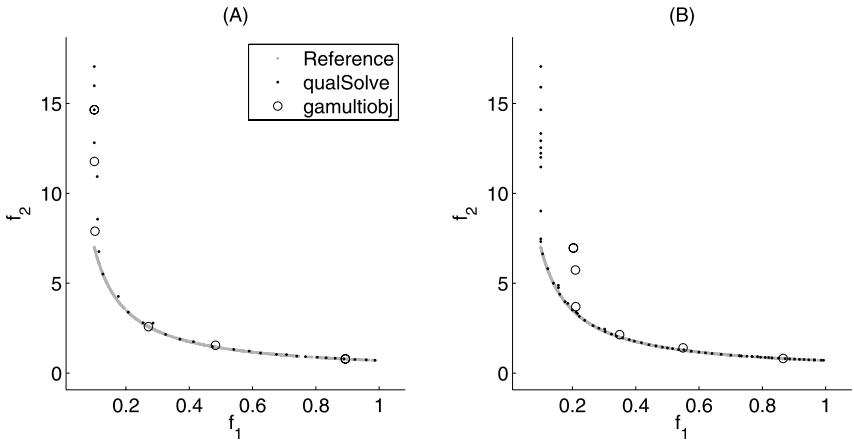


**Fig. 11** The Pareto fronts of the test function modified Deb bimodal. The number of non-dominated points produced by qualSolve was 30 after 100 evaluations and 70 after 250 evaluations. See caption of Fig. 9

correct front for OKA1. The reason is that OKA1 is a test function that has a very strong nonlinear behavior close to the minima (see Fig. 7); the algorithm locates the valley but fails to find the bottom of the oscillations. MATLAB's gamultiobj performs generally worse on all test functions. Note that the results of gamultiobj in Fig. 11A are better than in Fig. 11B. The reason is that genetic algorithms have a stochastic component and may very well yield different results in consecutive runs.

### 3.4 Optimization of a heater element

In order to evaluate the algorithm on a real simulation, a comparably fast and simple simulation was constructed: The simulation models a straight pipe through which a fluid is forced, and in the pipe a heater element is mounted, see Fig. 12. The heater element has the form of an ellipsoid whose equatorial and polar radii are variable. When a fluid is forced through the pipe, two things occur: (a) the pressure will be lower at the end of the pipe than at the beginning; and (b) the temperature will be higher at the end of the pipe than at the beginning. The direction of the flow determines the meaning of the words "end" and "beginning" as used in this context.

A good design of a heater is one that gives a large rise in temperature and at the same time has a low pressure increase (higher pressure increase requires a stronger pump, when pumping water through the heater). The heater element design problem therefore has the following form:

$$\begin{array}{ll} \text{minimize} & (\Delta p, -\Delta T)^{\mathrm{T}}, \\ \text{subject to} & 0.003 \leq r_a \leq 0.020, \\ & 0.003 \leq r_b \leq 0.020, \end{array}$$

where $\Delta p$ is the pressure drop and $\Delta T$ is the increase in temperature, both are dependent on the design variable $r_a$ and $r_b$ through simulation. The simulation was con-

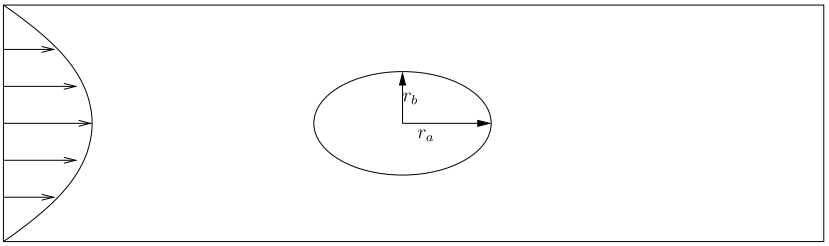


**Fig. 12** The pipe with the heater element. Fluid is forced from the left to the right, the temperature of the fluid rises and the pressure drops as it passes the heater

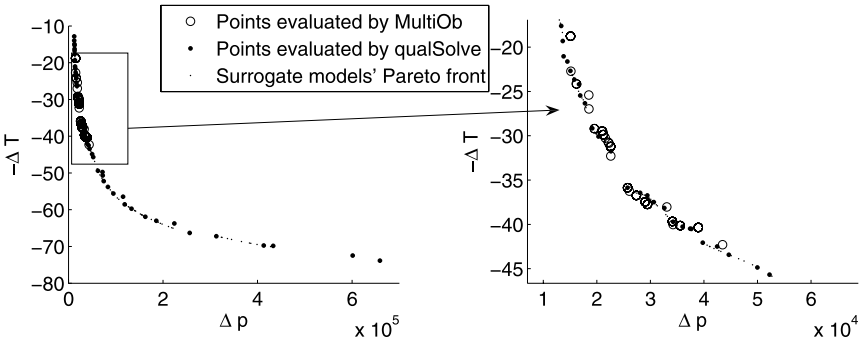


**Fig. 13** Result of the heater simulation. 200 evaluations performed by MultiOb and qualSolve each

structed in COMSOL multiphysics[17] and one simulation run took about two minutes. In this setting, the performance of qualSolve was compared to that of MultiOb. Parameter settings for qualSolve were identical to those used in Sect. 3.3. Both algorithms were run for a maximum of 200 function evaluations. The results can be seen in Fig. 13. What is notable is that qualSolve covers a larger part of the Pareto front than MultiOb, while they perform more or less equally well in the enlarged part (to the right in Fig. 13). In all fairness it should also be stated that 200 function evaluations is far too small a number for MultiOb (it normally requires around 20000 evaluations). For qualSolve, 200 evaluations is often sufficient when optimizing smooth lower dimensional functions (see the results in Sect. 3.3).

## 4 Conclusions and future research

We have presented an algorithm that is designed to handle both single and multiple objective functions defined by simulations. It addresses problems in existing single objective algorithms, such as the clustering of points and the overemphasizing of border regions (see Sect. 1.4), and handles single objective functions with or without

[17] COMSOL multiphysics is a commercial software for FEM simulations, see www.comsol.se.

noise. On the deterministic single objective test functions considered, the proposed algorithm has a comparable performance to existing algorithms. On noisy deterministic single objective test functions, it performs best among the algorithms compared. The algorithm qualSolve is one of few algorithms that also handles expensive multiobjective optimization; to the authors' knowledge it is the first developed specifically for noisy expensive multiobjective problems. This is relevant since there is no lack of noisy multiobjective problems arising from simulations in industry. The results demonstrate that the multiobjective version of the algorithm performs well on deterministic smooth test functions, although more rigorous tests and comparisons of the performance with other algorithms as well as tests on noisy functions should be carried through. The optimization of the heater simulation shows that the algorithm can perform well on a real life simulation based optimization.

There are many aspects of the proposed algorithm that should be subjected to more numerical tests in order to improve the performance. These include the cycle of the $\gamma$ parameter and the choice of $\eta$ in presence of different types of noise. The dependence of different choices of RBFs, initial points and external solvers should be investigated. The method should also be evaluated on more complex simulation based problems.

The main downside to the algorithm presented is the current execution time. As the quality function (Definition 4) contains an integral, the evaluation time of this auxiliary function increases exponentially with the dimension of the parameter domain $\Omega$. If the simulations are assumed to be very expensive (about 40 hours, as combustion engine simulations are) the current implementation becomes unsuitable for problems of parameter domain larger than about six. In order to decrease the execution time and hence increase the competitiveness of the algorithm in problems with more design parameters, the implementation of the numerical integration would have to be improved. This could be done in various ways such as exchanging the trapezoidal rule currently implemented for Monte Carlo integration or Gaussian integration.

Another major difficulty concerns the construction of the extended Pareto front, defined in (15), and the distance function, defined in (17), when the number of objectives increases. The construction of the extended Pareto front is easily accomplished when only two objectives are present, but becomes harder when the objective feasible region has a higher dimension. After constructing the extended front the distance function must be constructed in a way that enables fast evaluations, since the integrand in (12) is evaluated thousands of times. We have implemented it by measuring the distance to evenly distributed points on a grid in the objective function space, and creating an RBF interpolation of these points. This interpolation was then used in place of the distance function. Unfortunately, the number of grid points increases exponentially with the number of objectives making the interpolation computationally slow. Ideas for solving these problems include to better exploit the appearance of a Pareto front when constructing the distance function and to use the fact that the distance increases linearly in most of the domain.

We have not addressed nonlinear constraints. A straightforward way of handling them would be to pass them on to the external solvers used for solving the non expensive optimization problems. However, as the integration in the quality function is done over the domain inside the box constraints, it would lead to measuring the

uncertainty decrease outside of the domain. This in turn would make it attractive to choose points on the border of the domain defined by the non-linear constraints. The problem could possibly be overcome by adding a penalty function inside the weight function $\omega$.

The method presented in this article was developed in a project on simulation based multiobjective optimization of the Volvo D5 diesel engine (see Jakobsson et al. 2008; Saif-Ul-Hasnain 2008). The number of design parameters was five and the number of objectives three, including emissions of soot, $NO_x$, and *Indicated Mean Effective Pressure* (IMEP), which is a measure of the work output from the engine. Some minor modifications of the method were made. For example, to reduce overall computational time by allowing parallel simulations, it was necessary to generate several evaluation points in each iteration. Moreover, since only reasonable efficient engines were of interest, it was necessary to handle constraints depending on the surrogate models in the computation of the Pareto front. To test the reliability of the result, the Pareto fronts of the surrogate models after different number of evaluations were computed and compared both with interpolation and the approximation technique.

**Acknowledgements** This algorithm was developed as part of the GMMC *Combustion Engine Optimization* project which started in the spring of 2006. Participants in this project are the Department of Mathematical Sciences at Chalmers University of Technology, Fraunhofer-Chalmers Research Center for Industrial Mathematics (FCC), Volvo Car Corporation and Volvo Powertrain. The financial support from the Swedish Foundation for Strategic Research is gratefully acknowledged.

## References

Audet C, Savard G, Zghal W (2008) Multiobjective optimization through a series of single-objective formulations. SIAM J Optim 19(1):188–210

Björkman M, Holmström K (2000) Global optimization of costly nonconvex functions. Optim Eng 1:373–397

Booker AJ, Dennis JE Jr, Frank PD, Serafini DB, Torczon V (1998) Optimization using surrogate objectives on a helicopter test example. In: Computational methods for optimal design and control, Arlington, VA, 1997. Progr systems control theory, vol 24. Birkhäuser Boston, Boston, pp 49–58

Booker AJ, Dennis J Jr, Frank PD, Serafin DB, Torczon V, Torsset MW (1999) A rigorous framework for optimization of expensive functions by surrogates. Struct Optim 17:1–13

Conn AR, Scheinberg K, Toint PL (1997) Recent progress in unconstrained nonlinear optimization without derivatives. Math Program 79(1–3):397–414. Lectures on mathematical programming (97) (Lausanne, 1997)

Das I, Dennis JE (1998) Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 8(3):631–657. (Electronic)

Deb K (1999) Multi-objective genetic algorithms: problem difficulties and construction of test problems. Evol Comput 7(3):205–230

Deb K, Thiele L, Laumanns M, Zitzler E (2001) Scalable test problems for evolutionary multi-objective optimization. Technical report, Computer engineering and networks laboratory (TIK), Swiss federal institute of technology (ETH)

Dixon LCW, Szegö G (1978) The global optimization problem: an introduction. North-Holland, Amsterdam

Gutmann HM (2001) A radial basis function method for global optimization. J Glob Optim 19:201–227

Hanne T (2006) Applying multiobjective evolutionary algorithms in industrial projects. In: Küfer KH, Rommelfanger H, Tammer C, Winkler K (eds) Multicriteria decision making and fuzzy systems. Theory, methods and applications. Shaker Verlag, Aachen, pp 125–142

Huang D, Allen TT, Notz WI, Zeng N (2006) Global optimization of stochastic black-box systems via sequential kriging meta-models. J Glob Optim 34:441–466

Huyer W, Neumaier A (1999) Global optimization by multilevel coordinate search. J Glob Optim 14:331–355

Jakobsson S, Saif-Ul-Hasnain M, Rundqvist R, Edelvik F, Andersson B, Patriksson M, Ljungqvist M, Lortet D, Wallesten J (2008) Combustion engine optimization: a multiobjective approach. Optim Eng (in press)

Jones DR (2001) A taxonomy of global optimization methods based on response surfaces. J Glob Optim 21:345–383

Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 79:157–181

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13:455–492

Knowles J (2006) ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans Evol Comput 10:50–66

Kohavi R (1999) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14:th international joint conference on artificial intelligence (IJCAI). Morgan Kaufmann, San Mateo, pp 1137–1145

Kursawe F (1991) A variant of evolution strategies for vector optimization. In: Schwefel HP, Männer R (eds) Parallel problem solving from nature. 1st workshop, PPSN I, vol 496. Springer, Berlin, pp 193–197

Lewis RM, Torczon V, Trosset MW (2000) Direct search methods: then and now. J Comput Appl Math 124:191–207

Messac A, Mullur AA (2008) A computationally efficient metamodeling approach for expensive multiobjective optimization. Optim Eng 9(1):37–67

Miettinen K (1999) Nonlinear multiobjective optimization. International series in operations research & management science, vol 12. Kluwer Academic, Dordrecht

Powell MJD (2006) The NEWUOA software for unconstrained optimization without derivatives. In: Large-scale nonlinear optimization. Nonconvex optim appl, vol 83. Springer, New York, pp 255–297

Regis RG, Shoemaker CA (2005) Constrained global optimization of expensive black box functions using radial basis functions. J Glob Optim 31:153–171

Regis RG, Shoemaker CA (2007) A stochastic radial basis function method for the global optimization of expensive functions. INFORMS J Comput 19(4):497–509

Rudholm J, Wojciechowski A (2007) A method for simulation based optimization using radial basis functions. Master's thesis, Chalmers university of technology, Göteborg, www.chalmers.se/math/EN/research/research-groups/optimization/master-thesis-projects

Saif-Ul-Hasnain M (2008) Simulation based multiobjective optimization of diesel combustion engines. Master's thesis, Chalmers university of technology

Shor NZ (1985) Minimization methods for nondifferentiable functions. Springer series in computational mathematics, vol 3. Springer, Berlin

Wendland H (2005) Scattered data approximation. Cambridge monographs on applied and computational mathematics, vol 17. Cambridge University Press, Cambridge

Ye KQ, Li W, Sudjianto A (2000) Algorithmic construction of optimal symmetric Latin hypercube designs. J Stat Plan Inference 90:149–159