Constrained Optimization Involving Expensive Function Evaluations

Brekelmans, R.C.M.; Driessen, L.; Hamers, H.J.M.; den Hertog, D.

No. 2001-87

CONSTRAINED OPTIMIZATION INVOLVING EXPENSIVE FUNCTION EVALUATIONS: A SEQUENTIAL APPROACH

By Ruud Brekelmans, Lonneke Driessen, Herbert Hamers and Dick den Hertog

November 2001

ISSN 0924-7815

Discussion paper

Constrained optimization involving expensive function evaluations: a sequential approach[1]

Ruud Brekelmans[2]

Lonneke Driessen[3]

Herbert Hamers[4]

Dick den Hertog[4]

November 9, 2001

Abstract

This paper presents a new sequential method for constrained non-linear optimization problems. The principal characteristics of these problems are very time consuming function evaluations and the absence of derivative information. Such problems are common in design optimization, where time consuming function evaluations are carried out by simulation tools (e.g., FEM, CFD). Classical optimization methods, based on derivatives, are not applicable because often derivative information is not available and is too expensive to approximate through finite differencing.

The algorithm first creates an experimental design. In the design points the underlying functions are evaluated. Local linear approximations of the real model are obtained with help of weighted regression techniques. The approximating model is then optimized within a trust region to find the best feasible objective improving point. This trust region moves along the most promising direction, which is determined on the basis of the evaluated objective values and constraint violations combined in a filter criterion. If the geometry of the points that determine the local approximations becomes bad, i.e. the points are located in such a way that they result in a bad approximation of the actual model, then we evaluate a geometry improving instead of an objective improving point. In each iteration a new local linear approximation is built, and either a new point is evaluated (objective or geometry improving) or the trust region is decreased. Convergence of the algorithm is guided by the size of this trust region. The focus of the approach is on getting good solutions with a limited number of function evaluations (not necessarily on reaching high accuracy).

Keywords: nonlinear programming, derivative free optimization, trust region, filter.
JEL code: C61.

---

[1]The authors gratefully acknowledge financial support from EIT-io.

[2]Corresponding author. CentER Applied Research, P.O. Box 90153, 5000 LE, Tilburg, The Netherlands. E-mail: R.C.M.Brekelmans@.kub.nl

[3]Centre for Quantitative Methods BV, Eindhoven, The Netherlands.

[4]CentER & Department of Econometrics and Operations Research, Tilburg University, The Netherlands.

1 Introduction

In the past, design merely consisted of experimentation and physical prototyping. In the last decade, physical computer simulations, such as finite element methods, are widely used in engineering design and analysis. It is impressive to see the enormous market for these Computer Aided Engineering (CAE) tools. Examples of CAE tools can be found in Aerodynamics, Computational Fluid Dynamics, Computational Electromagnetics, Mechanical Engineering, and Electronic Circuit Simulations.

CAE tools enable the designers to simulate the performance of a product or process, but still designers are confronted with the problem of finding settings for a, possibly large, number of design parameters. These parameters should be set optimal with respect to several simulated product or process characteristics. These characteristics, called response parameters, may originate from different engineering disciplines. Since there are many possible design parameter settings and computer simulations are time consuming in many cases, the crucial question becomes how to find the best possible setting with a minimum number of simulations.

In this paper we consider such constrained optimization problems that contain nonlinear functions that require expensive function evaluations (i.e. simulations). The characteristic feature of the problems we consider is that in many cases simulations do not yield any derivative information. Moreover, there is no explicit information about the functional relationship between the designs and the responses simulated by the black-box computer model. Finally, we only consider optimization problems arising from deterministic simulations. In Brekelmans et al. (2001) the authors introduced a new optimization method for this type of problems. This paper describes the algorithm in greater detail and presents many sophistications and extensions to the algorithm.

Classical optimization methods, based on derivatives, are not applicable because often derivative information is not available and is too expensive to approximate through finite differencing. Hence, the optimization algorithm has to be derivative free. Recent derivative free trust-region methods are not applicable to the constrained optimization problem that we are interested in. Powell (2000), Conn et al. (1997), Dennis and Torczon (1994) and Torczon (1992) focus on unconstrained optimization. Additionally, the quadratic models which some of these authors advocate require exactly $\frac{1}{2}(n+1)(n+2)$ observations, where n is the dimension of the design space. For many black-box optimization problems, especially with a large number of design variables or significant time/cost of black-box evaluations, this number is too large for practical applications. Algorithms designed for global optimization (e.g. Jones et al., 1998) also require too many function evaluations in the entire feasible region. Therefore, for practical reasons, we have to restrict ourselves to finding local optima of the optimization problem. Moreover, high accuracy is not desired and not attainable due to the inaccuracy of the simulation model. Moreover, a high accuracy also requires too many function evaluations in the neighborhood of the optimum. In Toropov's multi-point approximation method (see Toropov et al., 1993) in each iteration a series of n new simulations is performed to obtain a new local approximation. See also Etman (1997). Since for the problems we consider each simulation is time consuming, also this method requires too many simulations. In our method simulations to improve the local approximation are performed adaptively. This means that a simulation to improve the approximation is performed only when the

current geometry of the design points simulated already is too bad.

Besides iterative algorithms which are often based on local approximations, also non-iterative algorithms based on global approximations are proposed in the literature. See e.g. Schoofs et al. (1994), Cramer et al. (1994), Dennis and Torczon (1996), and den Hertog and Stehouwer (2001). In these methods explicit global approximations for the functional relationships between the designs and the responses are made. By substituting these relationships into the original design optimization problem, an explicit nonlinear optimization remains to be solved. Key issues in these approaches are the choice of the points which should be simulated to get maximal global information and which kinds of global approximation models to use. The advantages of global approximation methods is that global insight is obtained on the behavior of responses. Moreover, when the design optimization problem changes somewhat (e.g. a change in a bound on a response), no new simulations have to be carried out since the global approximations can be used again. The disadvantage of this approach, however, is the high number of simulations required to obtain good global approximations. We therefore focus on an iterative method which uses local approximations.

Other approaches to the design optimization problem, which are used especially in the field of discrete-event simulation, are for example local search, tabu search, simulated annealing (e.g. Glover et al., 1996). These methods, however, assume that one simulation run can be performed quickly and only deal with (simple) constraints on the design parameters. For the problems we are interested in, i.e. design problems with time-consuming simulation runs, these methods require too many runs. Moreover, we will consider the constrained case.

This paper presents an iterative optimization algorithm that fits into the general trust-region framework (see Conn et al., 2000). Moreover, we report on the preliminary results of our toolbox SEQUEM, in which this new algorithm has been implemented. Some of the key elements of the algorithm are inspired by the filter method (see Fletcher and Leyffer, 1998) which jointly considers the objective value and constraints of the optimization problem. The filter method is used for the selection of the current iterate, and the computation and selection of new designs for the next function evaluation.

This paper is organized as follows. In Section 2 the mathematical formulation of the design optimization problem is given. Section 3 describes the steps of the algorithm. Numerical results are presented in Section 4. Section 5 concludes.

2 Problem description

In this paper the problem of black-box optimization is considered. The main characteristic of a black-box optimization problem is that the objective and constraint functions are, at least for an important part, not known explicitly. This situation often occurs when dealing with simulation models. A set of design parameters $x \in X \subseteq \mathbb{R}^n$ is fed to the black-box, for example a simulation tool, and a set of response parameters $r(x) \in \mathbb{R}^m$ is returned according to some unknown relationship between the designs and responses. The set X is the domain on which the black-box machine returns sensible responses. A design vector $x \notin X$ is not guaranteed to produce sensible output, or may produce no output at all. The black-box process is illustrated by Figure 1. In many applications the

black-box process is very time consuming or costly. Furthermore, there is no derivative information available for the responses $r(x)$.

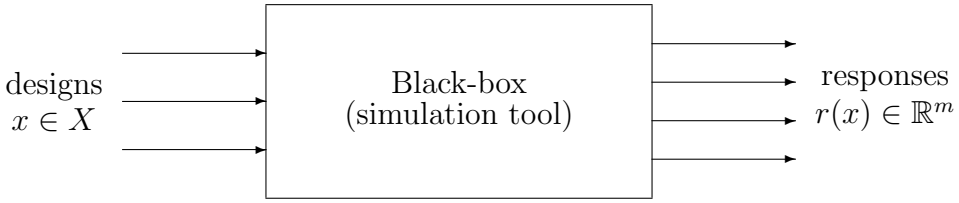


Figure 1: Black-box optimization.

Goal of the optimization problem is to minimize (maximize) a given function of the design and response parameters, subject to a set of constraints regarding (combinations of) the design and response parameters. The black-box minimization problem can be mathematically formulated as follows:

$$\min_{x \in X} z = f_0(x, r(x)) + g_0(x) \tag{1a}$$

$$\text{s.t. } l_i^f \leq f_i(x, r(x)) \leq u_i^f, \qquad 1 \leq i \leq m_f, \tag{1b}$$

$$l_j^g \leq g_j(x) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{1c}$$

$$l^x \leq x \leq u^x. \tag{1d}$$

In this model it is assumed that the functions f_i $(0 \leq i \leq m_f)$ and g_j $(0 \leq j \leq m_g)$ are known. Note that this does not imply that problem (1) is explicitly known, since it still contains the unknown function $r(x)$ coming from the black-box machine. Moreover, it is not required that the lower and upper bounds in (1b) and (1c) are finite. The bounds l^x and u^x, however, are assumed to be finite. The feasible region of the design constraints, (1c) and (1d), is assumed to completely lie within X. For convenience it is assumed that X is defined by the feasible region of constraints (1c) and (1d).

3 Sequential algorithm

In this section we describe the sequential algorithm SEQUEM, which is a variant of the trust-region method. See Conn et al. (2000) for an extensive discussion of trust-region methods. The main steps of the SEQUEM algorithm are shown below.

Initialization (See Section 3.2)
repeat
  Select current iterate (See Section 3.3)
  Approximate (local) model (See Section 3.4)
  Compute filter improving designs (See Section 3.5)
  Compute geometry improving designs (See Section 3.6)
  if trust region can be decreased **then** (See Section 3.7)
    Decrease trust region
  else

Select design to be evaluated (See Section 3.7)
Call black-box
end if
until stop criterion satisfied

A new aspect, compared to the standard trust-region algorithm, is the use of the filter method for both the selection of the current iterate and for the trial step. The filter method introduced by Fletcher and Leyffer (1998) enables the optimization of a constrained problem without the use of a penalty function.

In the remainder of this section the steps of the algorithm are described in greater detail. It should be noted that most steps of the algorithm are in a great extend independent of each other. Therefore, it is possible to use different implementations for these steps. The SEQUEM optimization package contains many different implementations of the steps in the basic trust-region algorithm.

3.1 Preparations

Besides problem definition (1) the optimization method discussed in this paper requires additional information about the problem, which is important for finding an optimum in an efficient manner. The additional information assumed to be available is as follows.

Start design: $x^{(0)} \in X$. In practice, designers often have a good idea about, or already work with a design that yields nice results, but which has not been optimized yet. Therefore, this design provides the starting point for the algorithm.

Initial step size: $\Delta^{(0)} \in \mathbb{R}^n$. Initial step sizes are needed to choose the step sizes taken in each dimension. The initial step sizes should be such that the effect on the responses will be of comparable magnitude. The step sizes can be seen as a way of scaling the design parameters.

Constraint scaling. One of the ingredients of the algorithm is that constraint violations are allowed during the execution of the algorithm. This can shorten the path to the solution of the problem. To be able to exploit constraint violations to a maximum extent it is important that different constraint violations can be compared using a certain measure. For ease of notation it is assumed that the constraints in (1b) are already scaled in such a way that a constraint violation of magnitude 1 for $f_i(x, r(x))$ and for $f_j(x, r(x))$ $(i \neq j)$ are of similar relevance.

3.2 Initialization

Before the main body of the algorithm can be entered several actions have to be performed. Firstly, a collection of design vectors has to be evaluated by the black-box machine to be able to approximate $r(x)$. Hereto, we use the start design $x^{(0)}$ and the initial step sizes $\Delta^{(0)}$. Besides the start design $x^{(0)}$ we evaluate design vectors in the neighborhood of $x^{(0)}$ taking a step size $\pm\Delta_j^{(0)}$ in dimension i $(1 \leq j \leq n)$. Hence, if k design vectors are required for the initialization, then the following vectors are used:

$$x_j^{(i)} := x_j^{(0)} + a_{ij}\Delta_j^{(0)}, \qquad 1 \leq i \leq k-1,\ 1 \leq j \leq n,$$

where $a_{ij} = \pm 1$ which are computed using a Design of Experiments (DOE) of size $k-1$. A D-optimal design will set a_{ij} to maximize $\det(D^T D)$, where

$$D := \begin{pmatrix} \varphi(x^{(1)})^T \\ \vdots \\ \varphi(x^{(k-1)})^T \end{pmatrix},$$

with $\varphi(x)$ the basis of the desired model to be fitted. In this paper we focus on the linear model, hence $\varphi(x)^T = (1 \ x^T)$. The number of design vectors evaluated in the initialization stage has to be sufficiently large to fit the desired model when approximating $r(x)$. Hence, if a linear model is being used we need $k \geq n+1$. On the other hand, it is desirable to choose the number of evaluations as small as possible since evaluations require a considerable amount of time.

Secondly, an initial *trust region* has to be defined. Throughout the algorithm a box-shaped trust region is used, which is completely determined by its center $x^{(c)} \in \mathbb{R}^n$ and step sizes $\Delta \in \mathbb{R}^n$. The box-shaped trust region is defined by

$$TR(x^{(c)}, \Delta) := \left\{ x \in \mathbb{R}^n : |x_j - x_j^{(c)}| \leq \Delta_j, \ 1 \leq j \leq n \right\}.$$

It seems natural to let the size of the initial trust region depend on the size of $\Delta^{(0)}$. Also we would like to use our estimates in a slightly larger region than the region in which we have performed the initial evaluations. Therefore, the initial trust region after k black-box evaluations is chosen as $\Delta^{(k)} = \delta \Delta^{(0)}$ with $\delta \geq 1$. Note that the center of the trust region is determined by the current iterate which is selected in the first step of the main algorithm (see Section 3.3).

3.3 Selection of current iterate (filter method)

The current iterate is a very important element of the algorithm as it determines the center of the trust region in which our next black-box evaluation will take place. At the termination of the algorithm, the sequence of selected current iterates forms a path from the start design to the final approximation of the optimum of problem (1). The idea is to select that design from the set of evaluated designs as current iterate that is closest to the optimum of problem (1). However, in practice the exact location of the optimum is not known, and consequently it is impossible to determine which of the evaluated designs is closest to it. Therefore, the algorithm has to select the most promising design based upon alternative criteria.

If a new black-box evaluation has been carried out, then the corresponding design is not guaranteed to be the current iterate for the next iteration. A new evaluation might result in a worse objective value than the objective value at the current iterate, for example because of a bad approximation of the local model. In such a case it may be wise not to shift the center of the trust region to this new design but stick to the old current iterate. Unfortunately, since we are dealing with a constrained optimization problem it is not sufficient just to compare objective values to decide upon the next current iterate. It is also important to take into account the feasibility of a design. The last evaluated design may have improved the objective value, but on the other hand also moved outside the feasible region. This yields two conflicting sources of information regarding the distance

to the optimum. One way to deal with constraints is to accept only feasible designs as current iterate. However, this approach forces the path towards the optimum to stay in the feasible region and may unnecessarily slow down the convergence of the algorithm. If an infeasible design is encountered it may very well be closer to the optimum than the old current iterate, and thus it should provide a promising step on the path towards the optimum. Hence, it seems sensible that the algorithm has to use multiple criteria based upon the objective value and the feasibility of the designs.

The algorithm selects the current iterate based upon the *filter method* introduced by Fletcher and Leyffer (1998). The filter method uses two criteria: the value of the objective function and the satisfaction of the constraints. Note that we have assumed that X is defined as the feasible region of constraints (1c) and (1d) which have to be satisfied to perform a black-box evaluation. Hence, regarding the constraint violations we can concentrate on constraints (1b). The two filter criteria are formally defined as follows. For $x \in X$ and $r \in \mathbb{R}^m$ define

$$z(x,r) := f_0(x,r) + g_0(x), \tag{2}$$

$$h(x,r) := \sum_{i=1}^{m_f} \max\{-s_i^l(x,r),\ -s_i^u(x,r),\ 0\}, \tag{3}$$

where $s_i^l(x,r)$ and $s_i^u(x,r)$ are the slack variables corresponding to constraints (1b), i.e.,

$$\begin{aligned} s_i^l(x,r) &:= f_i(x,r) - l_i^f, && 1 \le i \le m_f, \\ s_i^u(x,r) &:= u_i^f - f_i(x,r), && 1 \le i \le m_f. \end{aligned}$$

Hence, $z(x,r)$ is the objective value of problem (1), and $h(x,r)$ is the sum of the constraint violations of constraints (1b). Thus, $h(x,r(x)) = 0$ means that design x satisfies (1b).

Assume that at the current stage of the algorithm k design vectors $x^{(1)}, \ldots, x^{(k)}$ have been evaluated. Let $z^{(i)}$ and $h^{(i)}$ denote the values of $z(x^{(i)}, r(x^{(i)}))$ and $h(x^{(i)}, r(x^{(i)}))$ $(1 \le i \le k)$.

The filter method uses the concept of domination for the objectives z and h.

Definition 1 *A pair* $(z^{(i)}, h^{(i)})$ *is said to* dominate *another pair* $(z^{(j)}, h^{(j)})$ *if and only if both* $z^{(i)} \le z^{(j)}$ *and* $h^{(i)} \le h^{(j)}$.

Applying this domination principle on a set of pairs leads to the formation of a filter:

Definition 2 *A* filter *is a list of pairs* $(z^{(i)}, h^{(i)})$ *such that no pair dominates any other. A pair* (z,h) *is said to be acceptable for inclusion in the filter if it is not dominated by any other point in the filter.*

The filter can be graphically represented in the (z,h)-space as illustrated by Figure 2. Each point dominates the points located in the upper-right area from that point.

From here on we shall say that a design dominates another design, or that a design is accepted in the filter, as a shorthand of referring to the (z,h)-pairs of corresponding designs.

The basic principle of the use of a filter for the selection of the current iterate is that the last evaluated design becomes the current iterate if it is accepted in the filter.

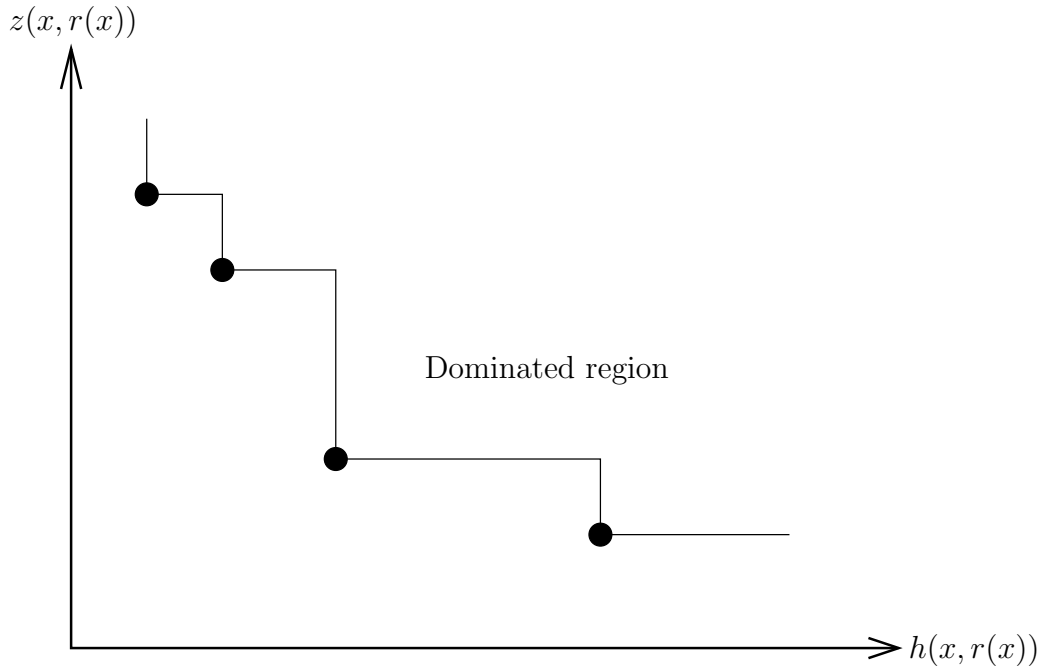


Figure 2: Example of a filter.

If the design is dominated by any of the previously evaluated designs, then the current iterate remains unchanged. Hence, the current iterate is defined as the last accepted entry into the filter. We have one exception to this rule, namely immediately after the initialization phase we select the start design $x^{(0)}$ if its feasible, or the most feasible filter entry otherwise.

As mentioned by Fletcher and Leyffer (1998) several problems can arise with the use of the filter method as outlined above. To overcome these problems Fletcher and Leyffer (1998) present some refinements in the filter's acceptance criteria. These problems and corresponding refinements have proved relevant to the SEQUEM algorithm as well.

A first problem that can be encountered is that while we are ultimately interested in a feasible solution for problem (1) the filter does not prevent the inclusion of an infinite sequence of designs for which $z^{(i+1)} < z^{(i)}$ and $h^{(i+1)} > h^{(i)}$ with $h^{(i)} \to \infty$. This situation can easily be excluded by adding the condition $h^{(i)} < h_{\max}$ for the inclusion in the filter, with $h_{\max} > 0$. An easy way to implement this condition is by adding an artificial pair $(-\infty, h_{\max})$ to the filter. At the start of the algorithm we set $h_{\max} = \gamma m_f$, hence, a design for which all m_f constraints in (1b) are violated by an amount γ lies exactly on the upper bound $h_{\max}$. Note that during the course of the algorithm $h_{\max}$ can be decreased to force the algorithm towards feasible solutions.

A second problem that can occur is that the path from the current iterate to the optimum is blocked by one of the previously evaluated designs, a so-called *blocking entry*. Of course, the basic idea of the filter method is to reject a pair $(z^{(k)}, h^{(k)})$ that is dominated by another pair. However, note that two filter pairs $(z^{(i)}, h^{(i)})$ and $(z^{(j)}, h^{(j)})$ can be close together while corresponding designs $x^{(i)}$ and $x^{(j)}$ are located far apart. Therefore, a new evaluated design inside the trust region around the current iterate, say $x^{(i)}$, can be

blocked from the filter by the design $x^{(j)}$. This can be especially annoying when the entire trust region is located in the infeasible region of problem (1) thereby preventing the algorithm to find a route back to the feasible region. Consequently, if such a situation is recognized, then the blocking entry is removed from the filter. Once a blocking entry is removed, it can never return in the filter in the remainder of the algorithm.

A possible third problem is the convergence to a pair (z, h) with $h > 0$. As explained by Fletcher and Leyffer (1998) this can be prevented by producing an envelope below the filter that prevents points arbitrarily close to the filter from being accepted. In all test problems that we have used to test the algorithm we have not encountered this problem. This is possibly due to the fact that $h_{\max}$ is decreased in certain stages of the algorithm forcing the designs towards the feasible region. Hence, to keep the algorithm as simple as possible the algorithm imposes no additional restrictions to avoid marginal filter improvements.

We are now ready to formulate the exact selection method for the current iterate. It is assumed that the evaluated designs, which are the candidates for the next current iterate, are denoted by $x^{(1)}, \ldots, x^{(k)}$. Furthermore, there is an upper bound $h_{\max}$ for the constraint violation function $h(x, r)$, and the indices of the blocking entries which are banned from the filter are given by the set $\mathcal{B}$. The decisions regarding $h_{\max}$ and $\mathcal{B}$ are discussed in Section 3.7. The set of indices that correspond to filter pairs after k evaluations is given by

$$\mathcal{F}_k := \big\{ i \in \{1, \ldots, k\} \setminus \mathcal{B} \;\big|\; h^{(i)} < h_{\max}, \\ \nexists j \in \{1, \ldots, k\} \setminus \mathcal{B} \text{ such that } (z^{(j)}, h^{(j)}) \text{ dominates } (z^{(i)}, h^{(i)}) \big\}.$$

The current iterate now corresponds with the last point accepted in the filter, i.e., $x^{(c)}$ with $c = \max_{i \in \mathcal{F}_k} i$.

3.4 Approximation of local model

An important part of the algorithm is the approximation of the black-box responses $r(x)$. It is our aim to approximate the actual response function $r(x)$ in the neighborhood of the current iterate $x^{(c)}$, more precisely within the trust region. Two conflicting aspects play a role here: the desire of high quality approximations and the time/cost involved by the black-box evaluations needed to fit the model. Obviously, high quality approximations are desirable. However, accurate approximations usually require a complex model specification which, in turn, requires many black-box evaluations to estimate the model parameters.

In our algorithm linear approximations are used to model the responses $r(x)$, i.e., for each response variable $r_j(x)$ $(1 \leq j \leq m)$ we specify the linear model

$$r_j(x) = (1 \; x^T)\beta^{(j)} + e^{(j)}, \qquad x \in \mathbb{R}^n,$$

where $\beta^{(j)} \in \mathbb{R}^{n+1}$ contains the $n + 1$ parameters of the model, and $e^{(j)}$ is the error term.

A disadvantage of the linear model is that it cannot properly approximate the non-linear function $r_j(x)$. However, if the trust region is small enough, then the response $r_j(x)$ can be approximated locally with a linear function. Moreover, the main objective is not the accurate approximation of $r(x)$, but finding the location of the optimum of

problem (1). Since the linear model requires only $n+1$ observations to fit the model the linear model saves black-box evaluations which can be used to explore the design space leading to the solution of problem (1).

Suppose that k design vectors $x^{(1)}, \ldots, x^{(k)}$ and corresponding observations $r(x^{(1)}), \ldots, r(x^{(k)})$ are obtained so far. For notational purposes we use the matrices

$$D := \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(k)} & \cdots & x_n^{(k)} \end{pmatrix} \tag{4}$$

and

$$R := \begin{pmatrix} r_1(x^{(1)}) & \cdots & r_m(x^{(1)}) \\ \vdots & \ddots & \vdots \\ r_1(x^{(k)}) & \cdots & r_m(x^{(k)}) \end{pmatrix}.$$

Let the current trust region be given by $TR(x^{(c)}, \Delta)$, where $x^{(c)}$ is the current iterate which is determined as explained in Section 3.3. Recall that $x^{(c)}$ is one of the design vectors evaluated so far. By the nature of the algorithm we are only interested in approximating $r(x)$ within the trust region. Therefore, the parameters of the linear model are estimated using weighted least squares (WLS). The weight of observation i, w_i, is chosen equal to 1 if observation i has to be used to fit the linear model, and 0 otherwise. The WLS problem for $r_j(x)$ $(1 \leq j \leq m)$ then becomes

$$\min_{\beta^{(j)} \in \mathbb{R}^{n+1}} \sum_{i=1}^{k} w_i \big(r_j(x^{(i)}) - (1 \; x^{(i)T})\beta^{(j)}\big)^2, \tag{5}$$

or in vector notation

$$\min_{\beta^{(j)} \in \mathbb{R}^{n+1}} \big(R_j - D\beta^{(j)}\big)^T W \big(R_j - D\beta^{(j)}\big),$$

where R_j denotes the j-th column of R, and $W \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the weights $w_1, \ldots, w_k$ on its main diagonal.

As explained before, the linear model may have difficulties approximating the responses when these behave nonlinearly inside the trust region. Therefore, it may be desirable to let the approximation be exact in the current iterate. Hence, to yield this property the WLS problem in (5) could be extended with the constraint

$$r_j(x^{(c)}) = (1 \; x^{(c)\,T})\beta^{(j)}. \tag{6}$$

In the selection of the weights we restrict ourselves to 0/1-type weights. Hence, the WLS procedure explained above is equal to applying standard least squares using only a subset of the complete set of observations. Note that WLS is by no means limited to the use of 0/1-type weights.

It remains to decide whether an observation i should be included in the subset used to fit the linear model. Since we want to obtain good approximations inside the trust region

it is obvious that all designs inside the trust region should be included. However, a design that is just outside the trust region could provide useful information for the approximation at the boundary of the trust region. Therefore, in principle, only designs $x^{(i)}$ that are inside the *weight-region*, a blown-up version of the actual trust region $TR(x^{(c)}, \Delta)$, receive weight 1. Thus,

$$w_i = \begin{cases} 1 & \text{if } x^{(i)} \in TR(x^{(c)}, \omega\Delta), \\ 0 & \text{if } x^{(i)} \notin TR(x^{(c)}, \omega\Delta), \end{cases} \tag{7}$$

where $\omega \geq 1$ is a parameter that specifies relative size of the weight-region to the trust region. This weight selection is illustrated by Figure 3. The designs inside the weight-region receive weight 1 are marked with a filled dot, and the designs outside the weight-region receive weight 0 are marked with an open dot.

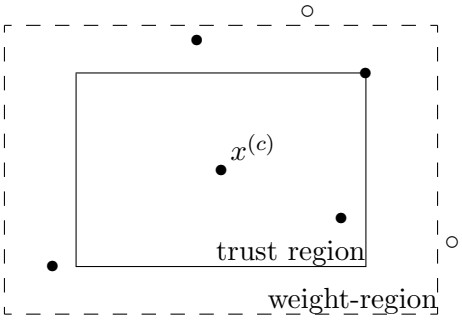


Figure 3: The trust region and the weight-region.

Unfortunately, in some cases using only the weight formula in (7) is not applicable as there is an extra condition that the weights $w_1, \ldots, w_k$ have to satisfy. The weighted least squares solution of (5) is not uniquely determined if the matrix $D^T W D$ is singular. A unique solution exists if the $(\sum_{i=1}^{k} w_i) \times (n+1)$ matrix $\tilde{D}$, whose rows are given by the rows of D that correspond to weights $w_i = 1$, has linearly independent columns. It immediately follows that we need $\sum_{i=1}^{k} w_i \geq n+1$, which, unfortunately, is not sufficient to guarantee non-singularity. This is illustrated by Figure 4 where the three designs inside the weight-region are linearly dependent. In a case like this it is required to add a design outside the weight-region to the subset of observations determined by (7). The selection of the design $x^{(i)} \notin TR(x^{(c)}, \omega\Delta)$ that has to be added is determined by the distance to the current iterate $x^{(c)}$. The design that is closest to $x^{(c)}$ will be given $w_i = 1$. This procedure is then repeated until the design matrix $\tilde{D}$ has linearly independent columns.

3.5 Computing filter improving designs

The approximations of the responses computed in Section 3.4 are used to move towards the optimum of problem (1). We do this by substituting the linear approximations denoted by $\hat{r}_j(x)$ $(1 \leq j \leq m)$ into (1). Since the approximations are assumed only to approximate the actual responses within the trust region we add the restriction $x \in$

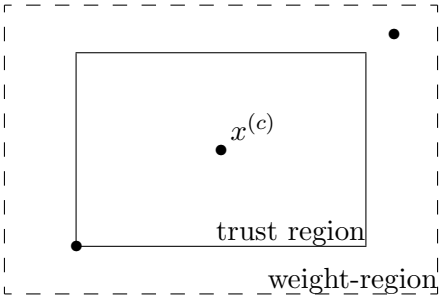


Figure 4: Linearly dependent designs.

$TR(x^{(c)}, \Delta)$. This yields the problem

$$\begin{aligned}
&\min\ z = f_0(x, \hat{r}(x)) + g_0(x) && && (8a)\\
&\text{s.t. } l_i^f \le f_i(x, \hat{r}(x)) \le u_i^f, && 1 \le i \le m_f, && (8b)\\
&\quad l_j^g \le g_j(x) \le u_j^g, && 1 \le j \le m_g, && (8c)\\
&\quad l^x \le x \le u^x, && && (8d)\\
&\quad x \in TR(x^{(c)}, \Delta). && && (8e)
\end{aligned}$$

This problem is now explicitly known and can be solved using an NLP solver.

Firstly, suppose that problem (8) is feasible and let x^* denote its solution. In Section 3.3 we have selected the current iterate which is the design that is expected to be closest to the optimum of problem (1) based upon the filter method. Hence, the filter concept can also be used to compute filter improving designs instead of a feasible design alone. The local solution x^* has an expected location $(z(x^*, \hat{r}(x^*)), 0)$ in the filter space. In general, every design $x \in TR(x^{(c)}, \Delta)$ corresponds to an expected pair $(z(x, \hat{r}(x)), h(x, \hat{r}(x)))$. By allowing a certain constraint violation it may be possible to yield a lower expected objective value than $z(x, \hat{r}(x))$. The most efficient design w.r.t. the filter criterion that yields an expected objective value of at most z is given by the solution of the problem

$$\begin{aligned}
&\min\ h(x, \hat{r}(x)) && && (9a)\\
&\text{s.t. } z(x, \hat{r}(x)) \le z, && && (9b)\\
&\quad l_j^g \le g_j(x) \le u_j^g, && 1 \le j \le m_g, && (9c)\\
&\quad l^x \le x \le u^x, && && (9d)\\
&\quad x \in TR(x^{(c)}, \Delta). && && (9e)
\end{aligned}$$

Solving (9) for several different values of z yields a set of designs that could provide promising candidates with respect to the (expected) filter improvement. Sensible objective targets z should be below $z(x^*, \hat{r}(x^*))$, and above the objective value that results from solving (8) without constraint (8b).

Secondly, if the local problem (8) is infeasible, then a first alternative is to minimize the constraint violations. Focusing on constraints (8b) this comes down to solving (9)

without constraint (9b). Just like the case where the local problem is feasible this yields an upper bound for the objective value that can be used to solve (9). Hence, the same approach as above can be used if the local problem is infeasible.

For numerical reasons the objective in (9a) is very nasty as it holds all the approximated constraints (1b) into a single value. By introducing m_f extra decision variables we can overcome this problem by reformulating problem (9) to

$$\min \sum_{i=1}^{m_f} h_i \tag{10a}$$

$$\text{s.t. } z(x, \hat{r}(x)) \leq z, \tag{10b}$$

$$h_i \geq -s_i^l(x, \hat{r}(x)), \qquad 1 \leq i \leq m_f, \tag{10c}$$

$$h_i \geq -s_i^u(x, \hat{r}(x)), \qquad 1 \leq i \leq m_f, \tag{10d}$$

$$h_i \geq 0, \qquad 1 \leq i \leq m_f, \tag{10e}$$

$$l_j^g \leq g_j(x) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{10f}$$

$$l^x \leq x \leq u^x, \tag{10g}$$

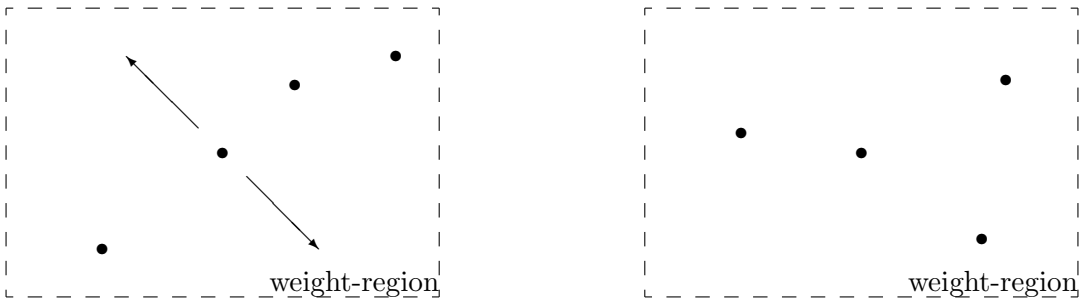
$$x \in TR(x^{(c)}, \Delta). \tag{10h}$$

It should be mentioned that solving several instances of problem (10) in addition to problem (8) is computationally intensive. However, these computations can be useful as they might save some much more costly black-box evaluations. Note that the computation of the set of filter improving designs does not involve any black-box evaluation. The actual selection of the design to be evaluated by the black-box machine is discussed in Section 3.7.

3.6 Improving the geometry

In Section 3.4 it is explained that the designs used to approximate the responses have to satisfy a certain condition to be able to estimate a linear model using least squares. Unfortunately, this condition is not sufficient to guarantee accurate approximations. There are two important reasons why approximations may be lacking accuracy. Firstly, the actual behavior of the responses within the trust region does not match a linear model well enough. This does not necessarily have to be problematic if the approximations do result in improvements with respect to the filter criterion. However, if it is not possible to find improvements, then it is desired to decrease the trust region such that the linear model can approximate the responses more accurately.

Unfortunately, there is a second reason why the quality of the approximations can be insufficient to find improving designs. This situation occurs when the designs used to fit the model are located in such a way that they do not contain enough information to estimate the parameters of the linear model. The designs are then said to have a bad *geometry*. Usually, this happens when there is little variance in one or more of the directions of the design space. This is illustrated by Figure 5(a) where the designs in the weight-region are nearly located on a straight line. Unlike the designs in Figure 4 they are not linearly dependent, and can be used to fit a linear model using least squares. However, approximations in the directions given by the arrows in Figure 5(a) are prone to result in significant errors. A set of designs has a good geometry if it fills the design



(a) Bad geometry (b) Good geometry

Figure 5: Examples of bad and good geometry.

space properly in all directions of the design space. An example of a good geometry is shown in Figure 5(b).

As a measure of the geometry we use a concept from the field of the Design of Experiments (DOE). In Driessen et al. (2001) it is shown that this is an attractive measure for solving black-box optimization problems. In Section 3.2 we already used a D-optimal design of experiment for the creation of the initial set of designs. A D-optimal design maximizes the determinant of $D^T D$, with D, the design matrix of the model, defined by (4). Since we do not use all available designs we concentrate on the weighted version: $D^T W D$.

For $x \in \mathbb{R}^n$ let $D(x) \in \mathbb{R}^{(k+1)\times(n+1)}$ be the design matrix for a linear model with all evaluated designs $x^{(1)}, \ldots, x^{(k)}$ extended with x, i.e.,

$$D(x) := \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(k)} & \cdots & x_n^{(k)} \\ 1 & x_1 & \cdots & x_n \end{pmatrix}.$$

Augmentation of an existing set of designs with a new design x inside the trust region using the D-optimality criterion yields the objective $\det(D(x)^T D(x))$. The procedure used to approximate the local model explained in Section 3.4 prevents the straightforward application of optimizing the objective mentioned above. This objective is based upon the assumption that all available designs will be used for the next regression. In our case this is not true, and, even worse, only until a new design has been evaluated it is possible to compute the weights that determine which subset of the evaluated designs is used to fit the model. If another current iterate is selected, then automatically the weight-region is moved accordingly. Even if the current iterate remains the same after evaluating a new design, then this might have consequences for some of the other designs' weights. Naturally, since the new design is chosen inside the trust region it is given weight 1. However, if the previous computation of the weights required adding some extra designs outside the weight-region to guarantee non-singularity of $D^T W D$, then this may not longer be necessary in the new situation.

For the computation of geometry improving designs it is assumed that the current iterate does not change. If this assumption turns out to be invalid, then this implies

that an improvement has been made in the filter sense. This can be seen as a stroke of good fortune, since the only reason to perform a geometry improving step is the lack of confidence in the ability of the current model to yield a filter improving step. The advantage of knowing that the current iterate does not change is that the new weight matrix $W(x)$ after adding design x can be computed beforehand. The best geometry improving design then follows from the problem

$$\max_{x \in X} \det\big(D(x)^T W(x) D(x)\big) \tag{11a}$$

$$\text{s.t. } l_j^g \leq g_j(x) \leq l_j^u, \qquad 1 \leq j \leq m_g, \tag{11b}$$

$$l^x \leq x \leq u^x \tag{11c}$$

$$x \in TR(x^{(c)}, \Delta). \tag{11d}$$

Dykstra (1971) has shown that the determinant in (11a) can be replaced by a numerically more efficient quadratic function. This function is given by $x^T (D^T W(x)_{-(k+1)} D)^{-1} x$, where $W(x)_{-(k+1)}$ is the weight matrix after adding design x with the last row and column deleted. Note that this method requires that $D^T W(x)_{-(k+1)} D$ is non-singular.

Another special case of the geometry objective function in (11a) is when exactly $n+1$ weights on the diagonal of $W(x)$ are equal to 1. Let $\tilde{D}(x)$ denote the matrix whose rows correspond the designs that have weight equal to 1. Since $\tilde{D}(x)$ is a square $(n+1) \times (n+1)$ matrix, (11a) simplifies to $\det(\tilde{D}(x))^2$. Hence, the objective is simply to maximize the determinant of $\tilde{D}(x)$ which is a linear function of x.

In any other case, it is not possible to use a simplification of the geometry objective (11a). Hence, the rather more difficult optimization problem (11) has to be solved to obtain a a geometry improving design.

In general, problem (11) has a convex objective function which makes it difficult to find a global maximum to the optimization problem. A standard NLP method can be used to solve (11), however this does not guarantee a global optimum. By using several starting points one can increase the probability of finding the global optimum. Moreover, the local optima of (11) which result from this method can be useful as well. With respect to the geometry measure they might be only slightly worse than the global optimum, whereas for the objective and constraints of problem (1) they can be much better. Therefore, all local optima of (11) are saved as possible candidates for the next black-box evaluation.

3.7 Evaluate candidate or decrease trust region

In Sections 3.5 and 3.6 it is explained how to compute candidates for the next black-box evaluation. The final step of the iterative algorithm is either to select one of the candidates to be evaluated by the black-box machine or to decrease the trust region. In the remainder of this section we do not make any difference between candidates that are a result of the filter improving step, or the geometry improving step. Instead, each candidate is judged on the basis of a number of criteria, and the decision is based thereupon. Moreover, in addition to the individual criteria for each of the candidates the decision depends on a number of general criteria, which are equal for all candidates.

Before we proceed with the detailed description of the selection procedure we introduce some notation. Let k denote the total number of black-box evaluations carried out by the algorithm so far. The set of designs that have been evaluated so far is denoted by

$\mathcal{H} = \{x^{(1)}, \ldots, x^{(k)}\}$. The collection of candidates for the next black-box evaluation is denoted by $\mathcal{C}$.

Next, the individual criteria for the candidate set are discussed.

Relative geometry measure

A *relative geometry measure* is computed for all candidates. An absolute geometry measure for design $y \in \mathcal{C}$, which is also used in the geometry improving step, is given by $\det(D(y)^T W(y) D(y))$ as in (11a). The value of this determinant is heavily influenced by the dimension n and the choice of weights $W(y)$, and, consequently, it is not immediately recognized whether it indicates a good or a bad geometry. Let $y^{(g)}$ denote the candidate that has the highest value for this absolute measure. The relative geometry measure of a candidate $y \in \mathcal{C}$ now results from the quotient

$$\gamma(y) = \frac{\det(D(y)^T W(y) D(y))}{\det(D(y^{(g)})^T W(y^{(g)}) D(y^{(g)}))}.$$

Obviously $\gamma(y^{(g)}) = 1$. Furthermore, a small value of $\gamma(y)$ denotes a geometry that is much worse than the geometry corresponding to $y^{(g)}$. Since we are not interested in immediately rejecting all candidates other than $y^{(g)}$ a candidate $y \in \mathcal{C}$ is said to be acceptable w.r.t. the geometry if $\gamma(y) \geq \gamma^*$ for a certain γ^*.

Distance to previous designs

It is desirable that each design that is evaluated by the black-box provides additional information about the behavior of the responses in the neighborhood of the trust region. A design that is very close to a design that has already been evaluated is unlikely to yield more information than a design that has a larger distance to each of the already evaluated designs. Therefore, the candidates are judged by the *smallest distance* to any of the already evaluated designs. Because we have to relate the additional information to the size of the trust region, the distances are computed after scaling with the current trust-region size $\Delta^{(k)}$, i.e.,

$$\delta(y) = \min_{x \in \mathcal{H}} \| \operatorname{diag}(\Delta^{(k)})^{-1}(y - x) \|, \qquad y \in \mathbb{R}^n.$$

Any candidate $y \in \mathcal{C}$ with $\delta(y) < \delta^*$, for a fixed value of δ^*, is said to be too close to an old design and is rejected for evaluation by the black-box machine during this iteration. Note that this does not exclude y for black-box evaluation in any of the next iterations of the algorithm after the trust region has been decreased.

In general, designs with a small value for $\delta(y)$ are not particularly good w.r.t. the geometry measure $\gamma(y)$ as well. However, if the distribution of the designs in the neighborhood of the trust region is reasonably well, then adding a design y close to one of the old designs will not necessarily result in a failure of the geometry criterion. Therefore, a distance related measure is needed in addition to the geometry measure $\gamma(y)$.

Expected filter improvement

The previous two criteria only depend on the positions of the designs and the trust region in the design space X. There is need for another criterion that evaluates the (expected) value of a design w.r.t. problem (1). In line with the approach taken previously, this is done by combining the local model and the filter method. As explained in Section 3.5 a design $y \in \mathcal{C}$ corresponds to an expected location $(z(y, \hat{r}(y)), h(y, \hat{r}(y)))$ in the (z, h)-space. If this expected filter pair is not dominated by the current filter, then this pair adds a certain area to the dominated region of the filter as illustrated by Figure 6. The

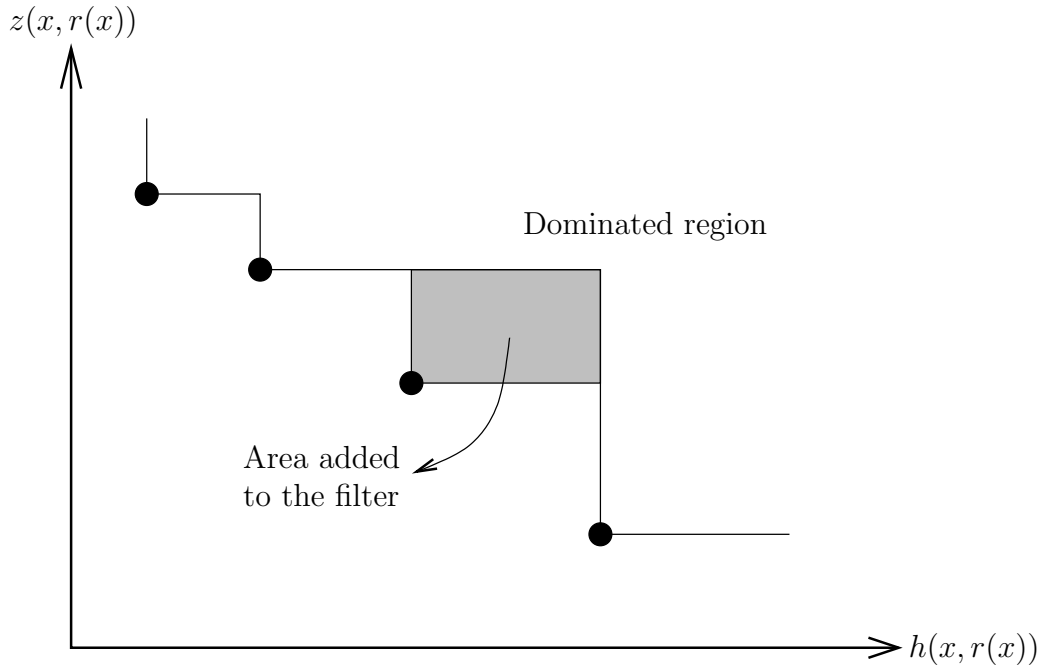


Figure 6: Extension of the dominated region by adding a filter pair.

improvement in filter sense of a design $y \in \mathcal{C}$ is measured by the area $\omega(y)$ that this design adds to the dominated region of the current filter. If $\omega(y)$ is large, then y is expected to yield a big filter improvement which makes it an attractive design for the next black-box evaluation.

Note that unfair comparison due to the existence of pairs beyond the extreme points of the current filter can be prevented by adding the artificial filter pairs $(-\infty, h_{\max})$ and $(z_{\max}, 0)$ to the current filter, with $z_{\max}$ a practical maximum value for the objective function, and $h_{\max}$ as explained in Section 3.3.

If we summarize the criteria above, then it can be said that we require a minimum standard w.r.t. the geometry and distance criteria measured by $\gamma(y)$ and $\delta(y)$, respectively, and the best possible expected filter improvement measured by $\omega(y)$. Hence, the *best candidate* according to these three criteria is given by

$$y^* = \underset{y \in \mathcal{C}}{\operatorname{argmax}} \big\{ \omega(y) : \ \gamma(y) \geq \gamma^*, \ \delta(y) \geq \delta^* \big\}.$$

It can be concluded that y^* is a very promising one among the candidates in $\mathcal{C}$. However, whether y^* will actually be evaluated also depends on other factors. In addition to the three individual measures above, which allow comparison between the candidates $y \in \mathcal{C}$, the decision taken in this step of the algorithm also depends on the following.

Current geometry

The *current geometry* plays an important role in deciding whether the trust region can be decreased or not. If the current geometry is not good, then it is not desirable to decrease the trust region since the current predictions of the local model might become more accurate if the geometry is improved. The geometry measure $\gamma(y)$ measures the effect of adding design y on the determinant relative to the best geometry design $y^{(g)}$; therefore, it does not give any information about the absolute effect on the determinant. The two determinants $\det(D^T W D)$ and $\det(D(y)^T W(y) D(y))$ together indicate the effect on the D-optimality criterion after adding a design y. Hence, if the quotient

$$\phi(y) = \frac{\det(D(y)^T W(y) D(y))}{\det(D^T W D)}$$

is large, then this indicates that the geometry measure can be significantly improved by adding the design y. For a good current geometry we require that the determinant does not increase too much after adding the best geometry improving design $y^{(g)}$, i.e., $\phi(y^{(g)}) \leq \phi^*$, for a certain ϕ^*.

Even though the determinant is a good measure for the dispersion of the designs in the design space X, it does not take into account the position of the trust region. When comparing the possible candidates $y \in \mathcal{C}$ this is not much of a problem, because all candidates are located inside the trust region. However, if the trust region is decreased, then besides an evenly dispersion of the designs it is also desired that the designs are located near the trust region. Therefore, we impose two additional conditions on the current geometry. Firstly, the total number of designs inside the weight region has to be larger than a certain number: $|\{i : \ x^{(i)} \in TR(x^{(c)}, \omega\Delta)\}| > B$. Secondly, the weighted average of the designs has to be located reasonably close to the current iterate, i.e.,

$$\left\| \frac{\sum_{i=1}^k w_i x^{(i)}}{\sum_{i=1}^k w_i} - x^{(c)} \right\| \leq \eta,$$

for a certain value of η.

Decision scheme

Figure 7 shows the decision flow chart about the next action of the SEQUEM algorithm.

The first item we are concerned with is the feasibility of the local model. If the local model is infeasible, i.e., problem (8) has no solution, then the SEQUEM algorithm initiates a *restoration step* which is intended to move the trust region back towards the feasible region of the original problem. It has been argued previously that it is not necessary to search for feasible designs all of the time. The filter concept provides an excellent method for finding a good balance between objective and constraints. However, if the local model is infeasible, then this could mean that the trust region is entirely located

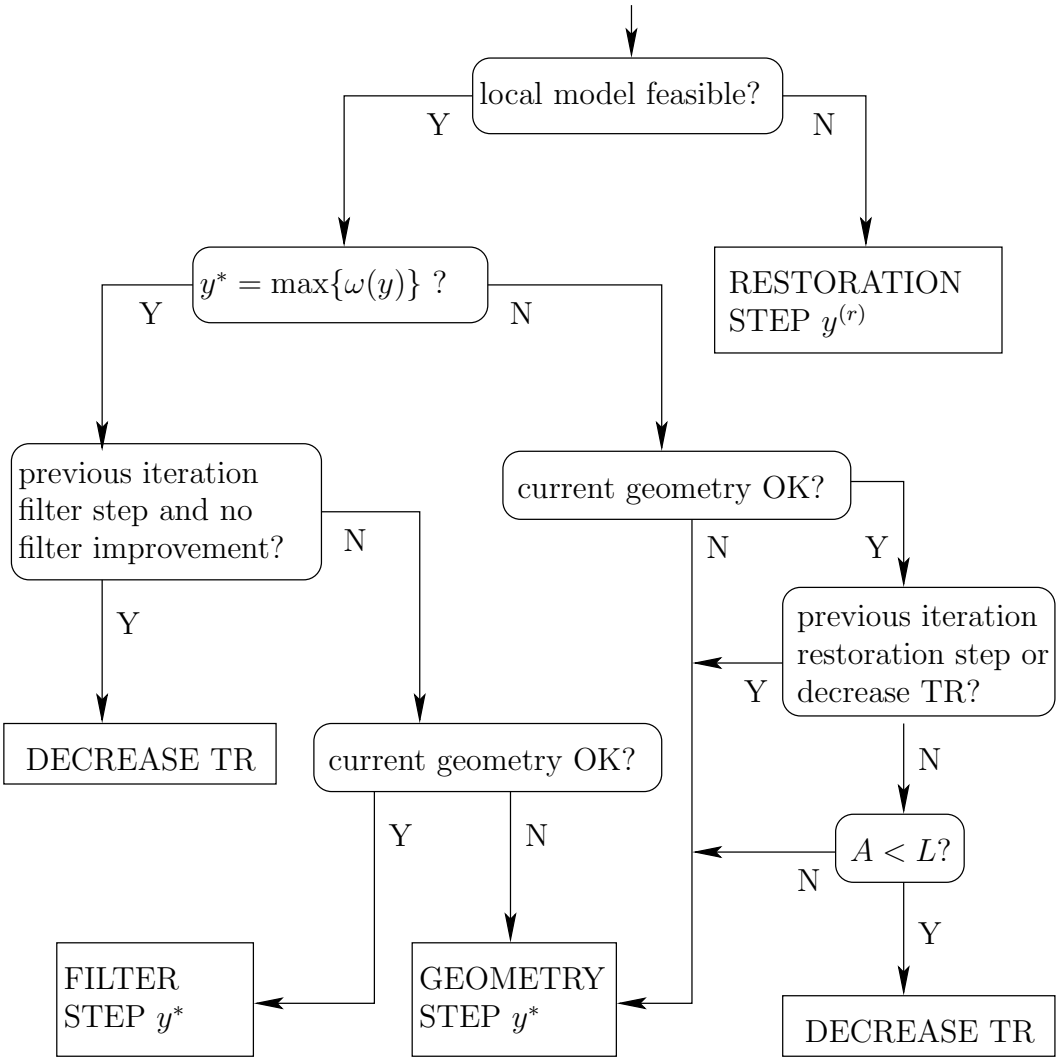


Figure 7: Decision flow chart.

outside the feasible region of problem (1). In this case, we do not want to rely on the filter measure $\omega(y)$ for the selection of a design $y \in \mathcal{C}$, because aiming for a filter improvement possibly with a positive expected constraint violation could lead even further away from the feasible region. Therefore, the restoration step selects the most feasible solution that satisfies the geometry and distance criteria to be evaluated by the simulation tool, i.e.,

$$y^{(r)} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \{h(y, \hat{r}(y)) : \gamma(y) \geq \gamma^*,\ \delta(y) \geq \delta^*\}.$$

If the local model is feasible, then the algorithm selects either y^* to be evaluated by the simulation tool, or the trust region will be decreased. If y^* is evaluated, then it will be labeled as either a *filter step* if we strongly expect y^* to be a filter improvement, or a *geometry step* otherwise.

This decision needs some further examination of the best candidate y^*. It is interesting to know whether there is a candidate that has a larger expected filter improvement, but which does not satisfy the geometry or distance criteria. If there is no such candidate,

then this makes y^* an even more attractive candidate, possibly even a *filter step*. However, if in the previous iteration a filter step did not result in a filter improvement, then this probably means that we are searching in a trust region that is too large, since all other clues are pointing in the direction of filter improvement. If this is not the case, i.e., the previous iteration was not a filter step or the previous iteration was a filter step that resulted in a filter improvement, then there is one final check before we decide that y^* can be labeled a *filter step*. If the current geometry is not good, then the local model might not be accurate enough and it would not be right to initiate a filter step. In this situation a *geometry step* is executed. Alternatively, if the current geometry is good, then y^* is evaluated and labeled a *filter step*.

If there is a candidate that has a larger expected filter improvement than the best candidate, i.e., $y^* \neq \text{argmax}_{y \in \mathcal{C}}\{\omega(y)\}$, then we come to another branch in the decision scheme. The general question in this branch is whether we perform a *geometry step* by evaluating y^*, or whether we *decrease* the trust region.

If the current geometry is not good, then it is likely that we cannot find filter improvements because the local approximations are not accurate enough. Therefore, we perform a *geometry step* if the current geometry is not good.

We also perform a *geometry step* if, in the previous iteration, we either performed a *restoration step* or decreased the trust region. In the former case there is the danger that only a small part of the trust region is in the feasible region of the local model, hence decreasing the trust region may again lead to an infeasible local model in the next iteration. In the latter case the alternative would be to decrease the trust region in two consecutive iterations of the algorithm, which is undesirable because we still need to build good approximations in the new trust region.

Finally, we arrive at the decision point where the current geometry is good and the previous iteration was a filter or a geometry step. In this situation it is important whether a candidate $y \in \mathcal{C}$ with $\omega(y) > \omega(y^*)$ has been rejected because of the geometry criterion or the distance criterion. If one or more of these candidates only fail the geometry criterion, then this merely indicates that the geometry needs to be improved. In this case y^* is a good candidate since it satisfies the geometry criterion and also might yield nice filter results. If a number of candidates with a high expected filter improvement are rejected due to the distance criterion, then this indicates that the local model expects filter improvements for designs that very are close to already evaluated designs, which is not very probable. Therefore, the number of elements in the set $\{y \in \mathcal{C} : \omega(y) > \omega(y^*),\ \delta(y) < \delta^*\}$ is computed. If this number, say A, is smaller than a certain R, then we perform a *geometry step*. Otherwise, we *decrease the trust region* since the local model seems to be inaccurate even though the current geometry is good which indicates that we are near the optimum.

4 Preliminary numerical results

The SEQUEM algorithm has been implemented for the case where the functions f_i $(0 \leq i \leq m_f)$ and g_j $(0 \leq j \leq m_g)$ are assumed to be linear. Note that this assumption does not prohibit nonlinear behavior of the black-box process. Moreover, nonlinear constraints on the design parameters can be handled by the introduction of artificial response parameters

representing the nonlinear terms of the constraints. The implementation of SEQUEM has been tested on a number of academical test problems. Since the test problems are stated in analytical form we have to make an assumption about the representation of the problem into the design and response parameters of a black-box process.

Taking into account that the SEQUEM algorithm is specifically designed for optimization problems involving time-consuming simulations the relevant range of the number of design parameters that can be handled by the SEQUEM algorithm is approximately 10–50. This range matches with the size of relevant problems that we have experienced in practice. Moreover, the total simulation time needed to optimize problems with more than 50 design parameters is beyond practical application using any algorithm.

Brekelmans et al. (2001) report results of the SEQUEM algorithm on two small test problems. A larger test problem is represented by problem no. 284 (TP 284) from Schittkowski (1987) which has 15 variables and 10 quadratic inequality constraints and a linear objective function. One response parameter has been introduced to capture the objective function, and one response parameters is introduced for each constraint as well. Hence, the black-box optimization problem has 15 design parameters and 11 response parameters. Problem TP 284 has been solved by the SEQUEM algorithm as well as by the FMINCON procedure from the MATLAB Optimization Toolbox (2000). The algorithm used by FMINCON is a Sequential Quadratic Programming (SQP) method using finite differencing to estimate the gradients.

Figure 8 shows the progress of the best feasible solution as a function of the number of black-box evaluations for both SEQUEM and FMINCON. It can be seen that the objective function decreases much more rapidly for the SEQUEM algorithm than for FMINCON, which makes one big step towards the optimum after about 650 evaluations. SEQUEM, on the other hand, reaches the theoretical optimum function value -1840 within 3% measured from the initial function value -522 after only 200 black-box evaluations.

SEQUEM has difficulty reaching the optimum with high accuracy. Considering the fact that most practical problems involving time-consuming simulations possess a certain degree of noise, or simply require too many simulations before optimality can be guaranteed, this is not a serious drawback.

5 Conclusions

Due to the developments in the area of CAE, computer simulations are used more frequently to evaluate and optimize products and processes. This leads to special nonlinear optimization problems in which the functions are not explicitly known and only time-consuming simulations can be carried out to calculate the response function values. Most of the existing optimization methods require too many iterations, i.e. simulations.

The algorithm proposed in this paper tries to save simulation runs as much as possible by using local approximations which are also based on previous simulation runs. The quality of the approximations is safeguarded by a geometry measure for the locations of the simulation points. This algorithm is implemented and preliminary computational results are promising. The algorithm approaches the optimal region very fast, but has difficulties to approach the optimum closely. However, since the simulation itself possesses some inaccuracy and in practice the number of simulation runs is often restricted, it is

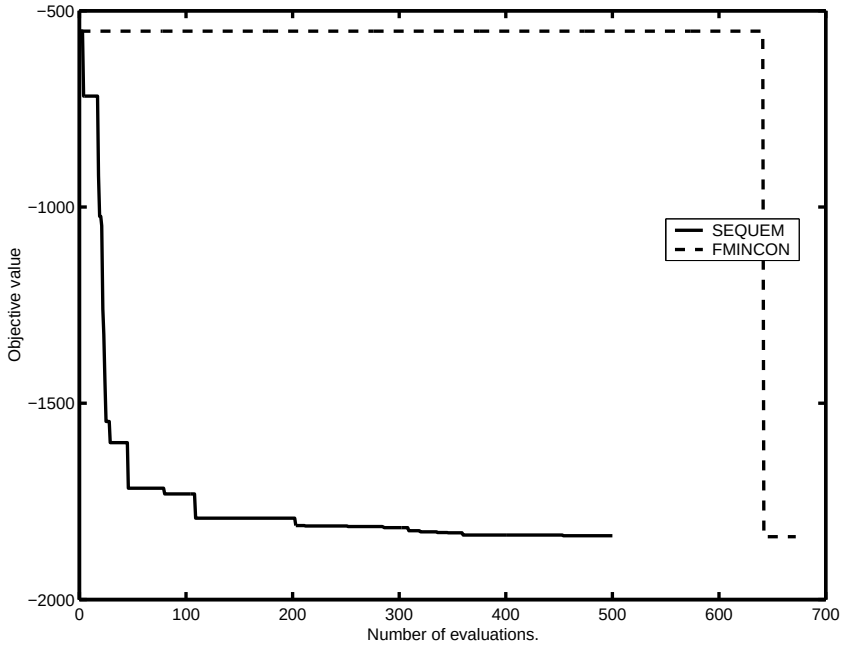


Figure 8: Best feasible solution for SEQUEM and FMINCON as a function of the number of evaluations for test problem TP 284.

normally speaking not necessary to obtain a solution with high accuracy.

The use of a filter in the optimization algorithm provides the designer with a set of possible attractive designs to choose from instead of just a single optimal design. This set, given by the designs in the filter at the end of the algorithm, indicates the gain that can be reached w.r.t. the objective value at the cost of the constraint violations. This can be especially useful when it is possible to twiddle with the bounds of the constraints, and conventional sensitivity analysis is too costly due to the expensive function evaluations required.

References

Brekelmans, R., Driessen, L., Hamers, H., and den Hertog, D. (2001). A new sequential optimization approach to product and process design involving expensive simulations. In Querin, O. M., editor, *Proceedings of the third ASMO UK/ISSMO conference on Engineering Design Optimization: Product and Process Improvement*, pages 49–52.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (2000). *Trust-Region Methods*. MPS/SIAM Series on Optimization. SIAM, Philadelphia.

Conn, A. R., Scheinberg, K., and Toint, P. L. (1997). Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79(1–3):397–414.

Cramer, E. J., Dennis, Jr, J. E., Frank, P. D., Lewis, R. M., and Shubin, G. R.

(1994). Problem formulation for multidisciplinary optimization. *SIAM J. Optimization*, 4(4):754–776.

den Hertog, D. and Stehouwer, H. P. (2001). Optimizing color picture tubes by high-cost non-linear programming. *Accepted for publication in European Journal of Operational Research*.

Dennis, J. E. and Torczon, V. (1994). Derivative-free pattern search methods for multidisciplinary design problems. In *Proceedings of the AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, Florida.

Dennis, J. E. and Torczon, V. (1996). Managing approximation models in optimization. In *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, Washington.

Driessen, L., Brekelmans, R., Hamers, H., and den Hertog, D. (2001). On D-optimality based trust regions for black-box optimization problems. CentER Discussion Paper 2001-69, Tilburg University, The Netherlands.

Dykstra, Jr, O. (1971). The augmentation of experimental data to maximize $|X'X|$. *Technometrics*, 13(3):682–688.

Etman, L. F. P. (1997). *Optimization of Multibody Systems Using Approximation Concepts*. PhD thesis, Technische Universiteit Eindhoven.

Fletcher, R. and Leyffer, S. (1998). Nonlinear programming without a penalty function. Numerical Analysis Report NA/171, University of Dundee, UK.

Glover, F., Kelly, J. P., and Laguna, M. (1996). New advances and applications of combining simulation and optimization. In Charnes, J. M., Morrice, D. J., Brunner, D. T., and Swain, J. J., editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 144–152.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492.

Powell, M. J. D. (2000). UOBYQA: unconstrained optimization by quadratic approximation. Numerical Analysis Report DAMTP 2000/NA14, University of Cambridge.

Schittkowski, K. (1987). *More Test Examples for Nonlinear Programming Codes*, volume 282 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag.

Schoofs, A. J. G., Roozen-Kroon, P. J. M., and Campen, D. H. V. (1994). Optimization of structural and acoustical parameters of bells. In *Proceedings of the AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, Florida.

The Mathworks, Inc. (2000). *Optimization Toolbox For Use with MATLAB: User's Guide, Version 2*.

Torczon, V. (1992). PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines. Technical Report 92-09, Rice University, Deparment of Computational and Applied Mathematics, Houston, Texas.

Toropov, V. V., Filatov, A. A., and Polynkin, A. A. (1993). Multiparameter structural optimization using FEM and multipoint explicit approximations. *Structural Optimization*, 6:7–14.