Efficient optimization of electronic circuits

Remie, V.

Award date:
2006

TECHNISCHE UNIVERSITEIT EINDHOVEN

Department of Industrial and Applied Mathematics

MASTER'S THESIS

Efficient optimization of electronic circuits

by
V. Remie

Supervisors: prof.dr. A.M.Cohen, dr.E.J.W.ter Maten, dr.ir. C.Lin

Eindhoven, August 17, 2005

Abstract

The group Philips Electronic Design & Tools, Analogue Simulation has developed the optimization tool Adapt, which is used for the design of electronic circuits.

An essential part of Adapt is the algorithm Gridmom, which solves optimization problems for functions defined on a grid. Gridmom solves minimization problems in an iterative process, where in every iteration quadratic models of the function that is to be optimized are constructed. The minima thus found are then coupled back to the original function. During the process the quality of these models is continually improved around the last found minimum point. The algorithm stops when the changes in these minimum points are very small.

Each function evaluation in Gridmom implies a circuit simulation by one of the supported simulators PStar and Spectre. Since circuit simulation is in general expensive with respect to processing power, the time that is needed to optimize a circuit with Adapt will be mainly determined by the cumulative time used by the simulator. Therefore, it is important that the number of evaluations in Gridmom be kept to a minimum. In this Master's Thesis we report on the project "Efficient Optimization of Electronic Circuits" that has been performed to improve the efficiency of Gridmom.

In the actual Gridmom algorithm the building of the quadratic models starts with sampling techniques. The techniques that have currently been used are quite problem independent. By including more knowledge about the actual functions a better optimization model may be constructed, which has a positive influence on the speed of the algorithm and possibly on the quality of the results. The efficient global optimization algorithm EGO satisfies these properties. This algorithm uses Kriging modelling, a statistic and geometric modelling technique used for point selection. Another point of interest is the convergence speed of the Gridmom algorithm when the area around the minimum is non-differentiable. Using piecewise linear approximative models instead of the quadratic approximative models should improve the speed of the algorithm.

These strategies have been tested for different setups. EGO has been implemented in Gridmom and tested for two different termination criteria, automatic termination and termination after a specified number of evaluations. From the results it follows that it is hard to say if EGO gives the desired improvement in the efficiency of Gridmom, further investigation is therefore needed. The use of piecewise linear approximative models have been implemented for one-dimensional functions and tested. From the results it follows that there is improvement in the value of the minimum as well as the number of evaluations needed by Gridmom.

Preface

This MSc Thesis describes the results of my MSc project for the Technische Universiteit Eindhoven. The project has been performed in cooperation with Philips Electronic Design & Tools / Analogue Simulation (ED&T/AS), which is headed by Jan van Gerwen. I have worked on Adapt, the optimization tool for electronic circuits, which is provided by ED&T/AS. My supervisors were Arjeh Cohen[1], Jan ter Maten[2], and Achie Lin[3].

I would like to thank everybody, who supported me to write this MSc Thesis. In particular, I mean Jan ter Maten, Achie Lin and Arjeh Cohen for their kind support, during the whole project. I am also gratefully to Ahmed El Guennouni[4], who spent a lot of time with supporting implementation in Matlab and with reviewing this thesis.

Besides these people, I would like also to mention explicitly Jan van Gerwen and also all other members of the group ED&T/AS.

[1] Professor Discrete Algebra and Geometry at the Department of Industrial and Applied Mathematics of the TU/e.

[2] Employee of ED&T/AS and scientist Scientific Computing at the Department of Industrial and Applied Mathematics of the TU/e

[3] Employee of ED&T/AS

[4] Employee of ED&T/AS

Contents

Chapter 1

Introduction

1.1 Analysis

Adapt is an optimization tool for the design of electronic circuits. It varies design parameters in order to make the design satisfy its specifications. In general, Adapt optimizes a performance quantity like distortion, power, area or delay time. Other design metrics impose constraints on the optimization problem, for example transition times.

An essential part of Adapt is the algorithm for solving the constrained optimization problem. The constrained non-linear optimization algorithm Gridmom is developed for application in Adapt. Gridmom is based on Response Surface Model techniques with trust regions, and solves a minimization problem on quadratic approximative models that are based on function evaluations at grid points. During the process the quality of the approximative model is continually improved around the last found minimum point (in a chosen trust region around this point). This in turn can lead to an update of the minimum point. During this phase the chosen trust region can expand or shrink. The process terminates when all neighboring grid points around a minimum of the current model have been evaluated and no further improvement can be made.

1.2 Goals

Each evaluation in Adapt implies a circuit simulation by one of the supported simulators, currently Pstar[1] and Spectre. Since circuit simulation is in general expensive with respect to processing power, the time that is needed to optimize a circuit with Adapt will be mainly determined by the cumulative time used by the simulator. The efficiency of Adapt is directly related to the efficiency of the applied algorithm Gridmom.

Reducing the number of evaluations in Gridmom, and also the quality of the solutions, is therefore of much interest for further study. In this Master's Thesis we report on the project "Efficient Optimization of Electronic Circuits" that has been performed to improve the efficiency of Gridmom.

1.3 Contribution

In the actual Gridmom algorithm the building of the initial Response Surface Model starts with general techniques, like Hooke-Jeeves and Uniform Design. These techniques are quite problem independent.

[1] See [1] for more information on the Pstar simulator.

By including more knowledge about the actual functions a better optimization model may be constructed, which has a positive influence on the speed of the algorithm and possibly on the quality of the results. There are several techniques that use this strategy. For instance, one can apply special interpolation techniques based on Kriging[2] to get information where best to evaluate new points to get better models quicker. Another point of interest is the convergence speed of the Gridmom algorithm. The convergence speed is small if the area around the minimum is non-differentiable. Using piecewise linear approximative models instead of the quadratic approximative models could drastically improve the convergence speed. Both techniques are studied practically, adaptation of the Matlab interface of Adapt, and theoretically, developing new algorithms and providing mathematical background. Also, some attention is given to other techniques.

For the understanding of using above-mentioned techniques a description of the Gridmom algorithm is given in chapter 2, and the Kriging interpolation techniques are described in chapter 3, the mathematical background of the techniques is discussed in chapter 4 and the results following from implementation in Matlab on selected representative test cases in chapter 5. Finally, some conclusions are drawn in the chapter 6.

[2]Stochastic model based on linear regression techniques.

Chapter 2

State of the Art: Gridmom

2.1 Introduction

This chapter contains a description of the Gridmom algorithm that is developed for application in Adapt. This description reflects the source code of the algorithm and is inspired by [7, 8]. The gridmom algorithm has the following two characteristic properties.

1. A nonlinear optimization method is used. It is not possible to predict how the objective and constraint functions will respond to a change in the variables, since no sensitivities with respect to the parameters are provided by the simulations. A linear programming method is therefore not suitable.

2. A method is used for avoiding preliminary evaluation point clustering, since the effect can reduce the efficieny of Gridmom or even give an incorrect solution, for example resulting in a local optimum which is not a global optimum.

In the next sections we discuss the structure of Gridmom. First, the optimization problem is transformed to an augmented Lagrangian optimization problem. This is discussed in section 2.2. The minimum returned by Gridmom must satisfy the Karush-Kuhn-Tucker conditions, these are discussed in section 2.3. Then, Gridmom uses an iterative process, the method of multipliers, to transform the augmented Lagrangian optimization problem, which is discussed in sections 2.4. Next, in the method of multipliers, a subproblem is solved in every step of the iteration. This subproblem is discussed in 2.5.

2.2 Transformation of minimization problem

Gridmom solves a constrained optimization problem by approximation (see [7]) that is expressed as

$$\begin{array}{lllll} \text{minimize} & f(\mathbf{x}), & \mathbf{x} = (x_1, x_2, ..., x_n), \quad x_i \in \mathbb{R} \quad \forall i \in \{1, 2, ..., n\}, & (2.1) \\ \text{subject to} & a_i \leq x_i \leq b_i, & i = 1, 2, \ldots, n, & \\ & c_i(\mathbf{x}) \leq 0, & i = 1, 2, \ldots, l, & \\ & \hat{c}_i(\mathbf{x}) = 0, & i = l+1, l+2, ..., l+m & \end{array}$$

where f, c_i and $\hat{c}_i$ are non-linear functions. For solving an optimization problem it is usual to have one kind of constraints only. Therefore we transform the inequality constraints in equality constraints by introducing slack variables $s_i \in \mathbb{R}$,

$$c_i(\mathbf{x}) \leq 0 \quad \Leftrightarrow \quad c_i(\mathbf{x}) + s_i = 0 \quad \text{such that} \quad s_i \geq 0 \tag{2.2}$$

Associated with (2.1) is a *Lagrangian* function,

$$\mathcal{L}(\mathbf{x}; \lambda) = f(\mathbf{x}) - \sum_{i=1}^{l} \lambda_i c_i(\mathbf{x}) - \sum_{i=l+1}^{l+m} \lambda_i \hat{c}_i(\mathbf{x}) \tag{2.3}$$

When determining the minimum of $\mathcal{L}$ we only use the derivatives with respect to x_i and λ_i ($i \in \{1, 2, ..., n\}$). Therefore, we include the slack variables in the terms associated with the inequality constraints,

$$\mathcal{L}(\mathbf{x}; \mathbf{s}; \lambda) = f(\mathbf{x}) - \sum_{i=1}^{l} \lambda_i \{c_i(\mathbf{x}) + s_i\} - \sum_{i=l+1}^{l+m} \lambda_i \hat{c}_i(\mathbf{x}) \tag{2.4}$$

The function of (2.4) can be thought of as a replacement for the objective function. This provides us an equivalent optimization problem where (2.4) is the objective function and where the constraints c_i are left out. We can go even further.

Instead of $\mathcal{L}$ we take a function that augments the Lagrangian function $\mathcal{L}$ with a quadratic penalty function,

$$\Phi(\mathbf{x}; \lambda; \mu; \mathbf{s}) = \mathcal{L}(\mathbf{x}; \mathbf{s}; \lambda) + \sum_{i=1}^{l} \mu_i \{c_i(\mathbf{x}) + s_i\}^2 + \sum_{i=l+1}^{l+m} \mu_i \{\hat{c}_i(\mathbf{x})\}^2 \tag{2.5}$$

where a penalty factor μ_j is added to each constraint c_j and $\hat{c}_j$ for all $j \in \{1, 2, \ldots, l\}$ respectively $j \in \{l+1, l+2, \ldots, l+m\}$, and where μ is a vector containing the penalty factors. Suppose that we minimize Φ with respect to x_i and s_i for all i. Then we can rewrite the terms of Φ containing s_i,

$$-\sum_{i=1}^{l} \lambda_i \{c_i(\mathbf{x}) + s_i\} + \sum_{i=1}^{l} \mu_i \{c_i(\mathbf{x}) + s_i\}^2 = \sum_{i=1}^{l} \left[\mu_i \left\{ c_i(\mathbf{x}) + s_i - \frac{\lambda_i}{2\mu_i} \right\}^2 - \frac{\lambda_i^2}{4\mu_i} \right] \tag{2.6}$$

We minimize the first part of the right hand side of (2.6), $\{c_i(\mathbf{x}) + s_i - \frac{\lambda_i}{2\mu_i}\}^2$, and we obtain the following values of s_i with the constraint $s_i \geq 0$ from (2.2),

$$\begin{aligned} c_i(\mathbf{x}) - \tfrac{\lambda_i}{2\mu_i} < 0 &\Rightarrow s_i = \tfrac{\lambda_i}{2\mu_i} - c_i(\mathbf{x}) \\ c_i(\mathbf{x}) - \tfrac{\lambda_i}{2\mu_i} \geq 0 &\Rightarrow s_i = 0 \end{aligned} \tag{2.7}$$

As a result of (2.7) we sum up two statements,

$$\begin{aligned} c_i(\mathbf{x}) \geq \tfrac{\lambda_i}{2\mu_i} \quad &\Leftrightarrow \quad \{c_i(\mathbf{x}) + s_i - \tfrac{\lambda_i}{2\mu_i}\}^2 = \{c_i(\mathbf{x}) - \tfrac{\lambda_i}{2\mu_i}\}^2 \\ c_i(\mathbf{x}) < \tfrac{\lambda_i}{2\mu_i} \quad &\Leftrightarrow \quad \{c_i(\mathbf{x}) + s_i - \tfrac{\lambda_i}{2\mu_i}\}^2 = 0^2 = 0 \end{aligned} \tag{2.8}$$

Combining the results of (2.8) with (2.4) we obtain the following augmented Lagrangian function,

$$\Phi(\mathbf{x}; \lambda; \mu) = f(\mathbf{x}) + \sum_{i=1}^{l} \{\mu_i \max\{c_i(\mathbf{x}) - \frac{\lambda_i}{2\mu_i}, 0\}^2 - \frac{\lambda_i^2}{4\mu_i}\} + \sum_{i=l+1}^{l+m} \{\mu_i \hat{c}_i^2(\mathbf{x}) - \lambda_i \hat{c}_i(\mathbf{x})\} \tag{2.9}$$

The max-function is included to distinguish between inactive inequality constraints on one hand and active and violated inequality constraints on the other hand.

The minimum of (2.1), and therefore also the minima of (2.4) and (2.9), must satisfy the Karush-Kuhn-Tucker conditions. These conditions are discussed in section 2.3.

2.3 Karush-Kuhn-Tucker conditions

2.3.1 First-order conditions

Let $\mathbf{x}^*$ be a vector that satisfies the constraints in (2.1) and let λ^* be the corresponding set of multipliers. Then $\mathbf{x}^*$ and λ^* must satisfy the first-order Karush-Kuhn-Tucker conditions,

1. $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x};\mathbf{s};\lambda)\Big|_{(\mathbf{x}^*;\mathbf{s};\lambda^*)} = \mathbf{0}$
2. $\lambda_i^* c_i(\mathbf{x}^*) = 0, \quad i = 1, 2, \ldots, l$
3. $\lambda_i^* \leq 0, \quad i = 1, 2, \ldots, l$

The condition for $\mathbf{x}^*$ satisfying the constraints in (2.1) can also be expressed as

$$\nabla_{\lambda}\mathcal{L}(\mathbf{x};\mathbf{s};\lambda)\Big|_{(\mathbf{x}^*;\mathbf{s};\lambda^*)} = \mathbf{0} \tag{2.10}$$

This is easy to see since

$$\begin{aligned}
& \nabla_{\lambda}\mathcal{L}(\mathbf{x};\mathbf{s};\lambda)\Big|_{(\mathbf{x}^*;\mathbf{s};\lambda^*)} = \mathbf{0} \\
\Rightarrow \quad & \nabla_{\lambda}\left[f(\mathbf{x}) - \sum_{i=1}^{l} \lambda_i \{c_i(\mathbf{x}) + s_i\} - \sum_{i=l+1}^{l+m} \lambda_i \hat{c}_i(\mathbf{x}) \right] \Big|_{(\mathbf{x}^*;\mathbf{s};\lambda^*)} = \mathbf{0} \\
\Rightarrow \quad & \nabla_{\lambda}\left[\sum_{i=1}^{l} \lambda_i \{c_i(\mathbf{x}) + s_i\} + \sum_{i=l+1}^{l+m} \lambda_i \hat{c}_i(\mathbf{x}) \right] \Big|_{(\mathbf{x}^*;\mathbf{s};\lambda^*)} = \mathbf{0} \\
\Rightarrow \quad & (c_1(\mathbf{x}^*) + s_1, c_2(\mathbf{x}^*) + s_2, \ldots, c_l(\mathbf{x}^*) + s_l, \hat{c}_{l+1}(\mathbf{x}^*), \hat{c}_{l+2}(\mathbf{x}^*), \ldots, \hat{c}_{l+m}(\mathbf{x}^*))^T = \mathbf{0}
\end{aligned}$$

Regarding the first condition of the first-order Karush-Kuhn-Tucker conditions and (2.10), we can say that the the Lagrange multipliers have a big influence on the values of the constraints and therefore on the position of the minimum of (2.4).

Regarding the second condition of the first-order Karush-Kuhn-Tucker conditions, we have the following properties,

- $c_i(\mathbf{x}^*) < 0 \quad \Rightarrow \quad \lambda_i^* = 0,$

  the constraint is called *inactive*, e.g. the constraint doesn't influence the position of the constrained optimum. Note that the minimum is attained at an interior point.

- $c_i(\mathbf{x}^*) = 0 \quad \& \quad \lambda_i^* \neq 0,$

  the constraint is called *active*, e.g. the constraint influences the position of the constrained optimum. Note that the minimum is attained at a boundary point.

For $\mathbf{x}^*$ optimal, some of the inequalities will be tight and some not. Those not tight can be ignored ($\lambda_i^* = 0$). Those that are tight can be treated as equalities.

The third and last condition of the first-order Karush-Kuhn-Tucker conditions follows directly from the construction of the Karush-Kuhn-Tucker conditions, here we will not go further into details.

2.3.2 Second-order conditions

The vector $\mathbf{x}^*$ must also satisfy the second-order Karush-Kuhn-Tucker conditions. This means, each $\mathbf{y} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ for which

$$\mathbf{y}^T \nabla_{\mathbf{x}} c_i(\mathbf{x})\Big|_{(\mathbf{x}^*)} = 0 \tag{2.11}$$

also satisfies

$$\mathbf{y}^T \left\{ \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{s}; \lambda)\Big|_{(\mathbf{x}^*; \mathbf{s}; \lambda^*)} \right\} \mathbf{y} > 0 \tag{2.12}$$

In other words, (2.4) must be positive semi-definite.

2.4 Method of multipliers

2.4.1 Introduction

The method of multipliers applies the Lagrange formalism to solve (2.1) by means of an iterative process. It is essential that the optimization is restricted to points on a grid. Initially, the grid is relatively coarse, but during the optimization process the grid is being refined.

First, a counter k is initialized at $k = 1$. The point $\mathbf{x}$ and penalty factor μ are set to initial values $\mathbf{x}^{(0)}$ and $\mu^{(0)}$, the multipliers λ are set to $\lambda^{(0)} = 0$. Define the augmented Lagrangian of the $k-$th iteration,

$$\Phi^{(k)}(\mathbf{x}) \ := \ \Phi(\mathbf{x}; \lambda = \lambda^{(k-1)}; \mu = \mu^{(k-1)}), \quad k \geq 1 \tag{2.13}$$

In the k-th iteration (2.13) is minimized with respect to $\mathbf{x}$, giving the argument $\mathbf{x}^{(k)}$ for which (2.13) is minimal,

$$\mathbf{x}_{(k)} = \min_{\mathbf{x}} \{\Phi^{(k)}(\mathbf{x})\} \tag{2.14}$$

This is a bound constrained optimization problem, which will be discussed in section 2.5. After this iteration, the multipliers λ_i and the penalty factors μ_i are updated, as to be discussed in the next subsection.

Finally, let $g_{\text{sp,min}}$ and $g_{\text{sp,tol}}$ be variables for the grid spacing, where $g_{\text{sp,min}} < g_{\text{sp,tol}}$. Then the method of multipliers stops after k iterations if and only if one of the following properties hold:

- The grid spacing is smaller than $g_{\text{sp,tol}}$ and the minimum among the evaluated points satisfies the Karush-Kuhn-Tucker conditions of section 2.3.
- The grid spacing is smaller than $g_{\text{sp,min}}$.

2.4.2 Updating parameters

At the end of the k-th iteration the multipliers and penalty factors are updated.

The new multiplier values $\lambda_i^{(k)}$ for the equality constraints in (2.13) in the $k+1$-th iteration are computed according to the first-order update rule,

$$\lambda_i^{(k)} = \lambda_i^{(k-1)} - 2\mu_i^{(k-1)} \hat{c}_i(\mathbf{x}_{(k)}), \ i = l+1, l+2, ..., l+m \tag{2.15}$$

where $\mathbf{x}_{(k)}$ is as defined in (2.14). For inequality constraints a similar rule applies,

$$\lambda_i^{(k)} = \lambda_i^{(k-1)} - 2\max\{\mu_i^{(k-1)} c_i(\mathbf{x}_{(k)}), \lambda_i^{(k-1)}\},\ i = 1, 2, ..., l \tag{2.16}$$

To see that (2.15) and (2.16) are sound iteration schemes we use induction on k. Consider we have a minimization problem with equalities only. Suppose we have obtained $\lambda^{(k-1)}$ in the $k-1$-th iteration. Then consider the corresponding minimum $\mathbf{x}_{(k)}$ of (2.13) in the k-th iteration. We have $\nabla_{\mathbf{x}}\Phi^{(k)}(\mathbf{x}_{(k)}) = \mathbf{0}$. So,

$$\nabla_{\mathbf{x}} f(\mathbf{x}_{(k)}) + \sum_{i=l+1}^{l+m} \{2\mu_i^{(k-1)} \hat{c}_i(\mathbf{x}_{(k)}) - \lambda_i^{(k-1)}\} \cdot \nabla_{\mathbf{x}} \hat{c}_i(\mathbf{x}_{(k)}) = \mathbf{0} \tag{2.17}$$

since $\nabla_{\mathbf{x}}\{\mu_i^{(k-1)}\{\hat{c}_i(\mathbf{x}_{(k)})\}^2 - \lambda_i^{(k-1)}\hat{c}_i(\mathbf{x}_{(k)})\} = \{2\mu_i^{(k-1)}\hat{c}_i(\mathbf{x}_{(k)}) - \lambda_i^{(k-1)}\} \cdot \nabla_{\mathbf{x}}\hat{c}_i(\mathbf{x}_{(k)})$. Let $\mathbf{x}_{(k)}$ also be the minimum of (2.1). Then the augmented Lagrangian in the $k+1$-th iteration must be equal to zero, so

$$\nabla_x \mathcal{L}(\mathbf{x}_{(k)}; \lambda^{(k)}) = \nabla_x f(\mathbf{x}_{(k)}) - \sum_{i=l+1}^{l+m} \lambda_i^{(k)} \nabla_x \hat{c}_i(\mathbf{x}_{(k)}) = \mathbf{0} \tag{2.18}$$

To satisfy (2.18) we choose $\lambda_i^{(k)} = \lambda_i^{(k-1)} - 2\mu_i^{(k-1)}\hat{c}_i(\mathbf{x}_{(k)})$, since then (2.17) implies (2.18). In an equivalent way (2.16) can be shown.

Finally, some remarks on the update process of the Lagrange multipliers,

1. The lagrange multipliers in (2.16) are equal to $\mathbf{0}$ if and only if the constraint is inactive.
2. At the start-phase of the process there is often no information available on the value of $\lambda_i^{(0)}$. Thus, in most situations one chooses $\lambda_i^{(0)} = 0$ for all i.

Besides $\lambda_i^{(k)}$ we need to choose values for $\mu_i^{(k)}$ too. [7, 8] required to choose $\mu_i^{(k)} = 10^k \cdot \max\{10, 10 \cdot f(\mathbf{x}_{\text{start}})\}$. A practical choice for the update of the penalty factors is to multiply them by a factor of 10. A good reason for choosing $\mu_i^{(k+1)} = 10 \cdot \mu_i^{(k)}$ is that if $\mu_i^{(k)}$ gets too big then Φ will be *ill-conditioned*. This means that the ratio between the largest and the smallest eigenvalues, the condition number, is very large. For example, we consider a quadratic function that has the following contour-plot,

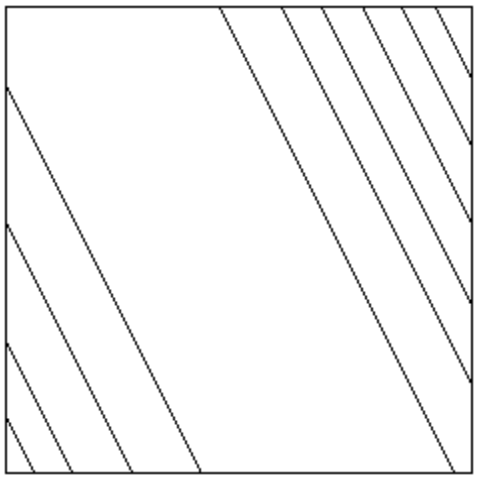

Figure 2.1: Contour-plot of quadratic function that is ill-conditioned.

The corresponding condition number of the Hessian is 2500, and furthermore the Hessian is positive semi-definite. By the figure we see that the change of the quadratic function is strongly dependent on the direction of the pertubation. The contours of the quadratic function are elongated, and the quadratic function is said to be ill-conditioned.

2.5 Solving subproblem

2.5.1 Introduction

The major part of work that has to be performed in Gridmom is solving the bound-constrained subproblem (2.14), in other words, minimize $\Phi^{(k)}(\mathbf{x})$, the augmented Lagrangian of (2.9). The part of Gridmom that solves this optimization problem is named *Gridmin*.

Gridmin consists of three phases, the starting phase, descent phase and the refinement-check phase. In the first iteration, $k = 1$, a starting point $\mathbf{x}_{\text{start}}$ is chosen and evaluated. Then, in the starting phase, $\Phi^{(1)}$ is evaluated at a sufficient number of points in order to construct a continuous approximation function. After some points have been evaluated Gridmin gets into a loop. This loop contains the descent phase and the refinement-check phase, where new points are computed and evaluated.

For $k > 1$ the described iteration steps are equivalent, except for using the starting phase. The starting phase is only invoked in the first iteration. In every iteration k, after every new function evaluation, it is checked if the obtained minimum of $\Phi^{(k)}$ satisfies the termination criteria, as formulated in subsection 2.5.2. If this is the case the loop terminates and the obtained minimum of $\Phi^{(k)}$ is returned.

The starting phase, descent phase and refinement-check phase are discussed in subsection 2.5.3, 2.5.4 respectively 2.5.5.

2.5.2 Termination criteria

Gridmin terminates when a stationary point or a fixed maximum number of function evaluations has been attained. Let $Y = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(N)}\}$ be the set of minima found during the iterations in the descent phase or during the refinement-check phase. Then, the following criteria are applied to test for a stationary point,

$$| f(\mathbf{y}^{(1)}) - f(\mathbf{y}^{(N)}) | < \varepsilon_q(1+ | f(\mathbf{y}^{(i)}) |) \tag{2.19}$$

$$\| \mathbf{y}^{(1)} - \mathbf{y}^{(i)} \| < \varepsilon_y(1+ \| \mathbf{y}^{(1)} \|), \quad i = 2, 3, ..., N \tag{2.20}$$

for certain ε_q and ε_y.

2.5.3 Starting phase

Gridmin includes two different starting phase algorithms, viz,

1. Hooke-Jeeves method
2. Uniform Design technique

The Hooke-Jeeves method [13] combines exploratory moves, in which one variable is changed at a time, with pattern moves, which follow heuristic rules to compute a new base point for the next set of exploratory moves.

The alternative, the Uniform Design technique [5], which is used in Gridmom by default, generates a sample point distribution in the feasible domain that is as uniform as possible. Given s variables (factors) and q discrete values (levels) for each of the variables, the Uniform Design technique selects q out of the possible combinations. The general principals of the design are the following,

1. If neither q nor $q+1$ is equal to a prime number, set the number q_0 equal to q, if q is odd, or to $q+1$, if q is even. Note that q_0 is always an odd number. If s is sufficiently small compared to q, a set of s positive integers a_i can be found that do not have a divider in common with q, $\gcd(a_i, q) = 1, i = 1, \ldots, s$. Given such a set of numbers a_i, the distribution points are given by,

$$P_k = (ka_1, \ldots, ka_s) \qquad \mod q_0, \quad k = 1, \ldots, q_0.$$

2. In the special case where q equals a prime number p or $p-1$, a positive integer a $(0 < a < p)$ can be selected. From a a set of distribution points can be obtained defined by,

$$P_k = (k, ka, \ldots, ka^{s-1}) \qquad \mod p, \quad k = 1, \ldots, p.$$

3. The most uniformly distributed set of points is obtained by determining the optimal set of integers $\mathbf{a} = (a_1, \ldots, a_s)$ or $\mathbf{a} = (1, a, \ldots, a^{s-1})$. This optimal set is found by minimizing the function,

$$\zeta(q, \mathbf{a}) = \frac{1}{q} \sum_{k=1}^{q} \prod_{i=1}^{s} \left\{ 1 - \frac{2}{\pi} \log \left[2 \sin \left(\pi \frac{a_{ik}}{q+1} \right) \right] \right\}$$

   where $a_{ik} = ka_i \mod q$, if $k < q$, and $a_{iq} = q$.

4. If q is equal to $q_0 + 1$ or $p + 1$, the first q points of the optimal set of distribution points are selected.

Note that the Uniform Design technique is function independent. Furthermore, in practice we choose q equal to the number of desired distribution points and s equal to the dimension of the objective function.

2.5.4 Descent phase

The descent phase consists of an iterative process. At the start of each iteration a least-square fit is applied to the evaluated points to build a bound constrained quadratic approximation function. Then, a trust region method [1] calculates the minimum of the bound constrained function and the set of evaluated points, the evaluated optimum and some other parameters are updated. We discuss the iterative process.

Define, at the start of the process, $S^{(0)}$ the set of all evaluated points[2] in Gridmom, a reference point $\mathbf{x}^{(0)}$ equal to the point where a minimum of $\Phi^{(k)}$ is attained among all points in $S^{(0)}$, and a trust region radius $\Delta^{(0)}$. Also, define ratio bounds $0 \leq \rho_1, \rho_2 \leq 1$, trust-region radius bounds $\Delta_{\min}$ and $\Delta_{\max}$, counter variables $c_{\max} > 0$ and $c = 0$, and a termination variable $\varepsilon_T > 0$.

[1] This method is inspired by the work reported by Elster and Neumaier [3].

[2] For $k = 0$ we have $S^{(0)} = \{\mathbf{x}\}$, where $\mathbf{x}$ is the initial starting point given as input to Gridmom.

Consider the i-th iteration, $i \geq 1$. Define $S^{(i)} = S^{(i-1)} \cup \{\mathbf{x}^{(i-1)}\}$ where $\mathbf{x}^{(i-1)}$ is the reference point obtained in the previous iteration. Let $q^{(i)}$ be the quadratic approximation function,

$$q^{(i)}(\mathbf{x}) = \gamma_i + \mathbf{g}_i^T(\mathbf{x} - \mathbf{x}^{(i-1)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(i-1)})^T \mathbf{H}_i (\mathbf{x} - \mathbf{x}^{(i-1)}) \tag{2.21}$$

where $\gamma_i \in \mathbb{R}$, $\mathbf{g}_i$ is a vector and $\mathbf{H}_i$ a symmetric matrix. These are called the gradient and Hessian of $q^{(i)}$ respectively. One formulates a least square fitting problem as the minimization of the sum of squares of the difference between $\Phi^{(k)}$ and $q^{(i)}$,

$$\min_{(\gamma_i, \mathbf{g}_i, \mathbf{H}_i)} \sum_{\mathbf{x} \in T^{(i)}} \parallel q^{(i)}(\mathbf{x}) - \Phi^{(k)}(\mathbf{x}) \parallel_2^2 \tag{2.22}$$

where $T^{(i)} \subseteq S^{(i)}$ and where $\| \cdot \|_2$ denotes the l^2-norm. Define $s^{(i)}$ as the number of elements in $S^{(i)}$. Note that for $s^{(i)} \leq n$ there exists no subset $T^{(i)}$ of $S^{(i)}$ such that (2.22) is feasible. When choosing $s_i > n$, one can distinguish three cases.

Suppose we are in the case where $s^{(i)} \geq \frac{1}{2}(n+1)(n+2)$. Let $\mathbf{H}_i$ be a symmetric $n \times n$-matrix. The gradient $\mathbf{g}_i$ contains n independent parameters and $\mathbf{H}_i$ contains $n + \frac{n^2-n}{2} = \frac{1}{2}n(n+1)$ independent parameters. Thus, $q^{(i)}$ contains $1 + n + \frac{1}{2}n(n+1) = \frac{1}{2}(n+1)(n+2)$ independent parameters, which implies that there are sufficient many evaluated points $T^{(i)} \subseteq S^{(i)}$ for which (2.22) is feasible. Let $\parallel \mathbf{x} - \mathbf{y} \parallel_\Omega$ be the scaled distance between $\mathbf{x}$ and $\mathbf{y}$ defined as a scaled infinite norm,

$$\parallel \mathbf{x} - \mathbf{y} \parallel_\Omega = \max_{i \in \{1,2,\ldots,n\}} \left\{ \frac{|x_i - y_i|}{b_i - a_i} \right\} \tag{2.23}$$

where Ω denotes the feasible region of (2.1) and a_i and b_i the lower respectively upper bound of the i-th variable, see 2.1. Define a rank mapping $R^{(i)} : S^{(i)} \mapsto \mathbb{Z}^+$ such that $R^{(i)}(\mathbf{x}) = r$ implies that there are exactly $r-1$ points $\mathbf{y} \in S^{(i)}$ with the property $\parallel \mathbf{y} - \mathbf{x}^{(i-1)} \parallel_\Omega \leq \parallel \mathbf{x} - \mathbf{x}^{(i-1)} \parallel_\Omega$. Let $\mathbf{x}_a$ be a point in $S^{(i)}$ such that $R^{(i)}(\mathbf{x}_a) = \frac{1}{2}(n+1)(n+2)$. Let $T^{(i)}$ be the set containing all points of $S^{(i)}$ that are at scaled distance smaller than $\parallel \mathbf{x}_a - \mathbf{x}^{(i-1)} \parallel_\Omega \cdot (1 + \varepsilon_T)$ from $\mathbf{x}^{(i-1)}$,

$$T^{(i)} = \{\mathbf{x} \in S^{(i)} \mid \parallel \mathbf{x} - \mathbf{x}^{(i-1)} \parallel_\Omega < \parallel \mathbf{x}_a - \mathbf{x}^{(i-1)} \parallel_\Omega \cdot (1 + \varepsilon_T)\} \tag{2.24}$$

for some ε_T. Note that the choice of ε_T is related to the distribution of the points of $S^{(i)}$ in Ω. To solve (2.22) for given $T^{(i)}$ we need to rewrite (2.22) to a set of equations; suppose that we have $T^{(i)} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p\}$ for some $p \in \mathbb{Z}$. Define $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jn})^T$ for all $j \in \{1, 2, \ldots, p\}$. We need to minimize $\parallel q^{(i)}(\mathbf{x}_j) - \Phi^{(k)}(\mathbf{x}_j) \parallel_2^2$ for all j. Substituting (2.21) in this expression leads to the minimization problem $\min\{\parallel \mathbf{A}\mathbf{y} - \mathbf{b} \parallel_2^2\}$, where,

$$\mathbf{A} = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} & \frac{1}{2}x_{11}x_{11} & \frac{1}{2}x_{11}x_{12} & \ldots & \frac{1}{2}x_{11}x_{1n} & \ldots & \frac{1}{2}x_{1n}x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} & \frac{1}{2}x_{21}x_{21} & \frac{1}{2}x_{21}x_{22} & \ldots & \frac{1}{2}x_{21}x_{2n} & \ldots & \frac{1}{2}x_{2n}x_{2n} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 1 & x_{p1} & x_{p2} & \ldots & x_{pn} & \frac{1}{2}x_{p1}x_{p1} & \frac{1}{2}x_{p1}x_{p2} & \ldots & \frac{1}{2}x_{p1}x_{pn} & \ldots & \frac{1}{2}x_{pn}x_{pn} \end{pmatrix},$$

$$\mathbf{y} = \begin{pmatrix} \gamma_i \\ g_{i1} \\ g_{i2} \\ \vdots \\ g_{in} \\ h_{11} \\ h_{12} \\ \vdots \\ h_{1n} \\ \vdots \\ h_{nn} \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} q^{(i)}(\mathbf{x}_1) \\ q^{(i)}(\mathbf{x}_2) \\ \vdots \\ q^{(i)}(\mathbf{x}_p) \end{pmatrix} \tag{2.25}$$

and where $\mathbf{g}_i = (g_{i1}, g_{i2}, \ldots, g_{in})^T$ and $\mathbf{H}_i = (h_{jk})$. If the left matrix of (2.25) does not have full column rank, then other evaluated points are added to $T^{(i)}$ till (2.25) is feasible or till $T^{(i)} = S^{(i)}$. When $T^{(i)} = S^{(i)}$ and $\mathbf{A}$ still does not have full column rank, then an algorithm[3] using *Singular Value Decomposition* and *QR Decomposition* techniques is used to solve (2.25) approximatively.

Then, an expression for $q^{(i)}$ is constructed, say $q^{(i)}_*$. Let $\mathbf{x}_{a'}$ be a point in $S^{(i)}$ such that $R^{(i)}(\mathbf{x}_{a'}) = n + 2$ and redefine $T^{(i)}$ in (2.24) by taking $\mathbf{x}_a = \mathbf{x}_{a'}$. Also, redefine the approximation function $q^{(i)}$,

$$q^{(i)}(\mathbf{x}) = \gamma_i + \mathbf{g}_i^T(\mathbf{x} - \mathbf{x}^{(i-1)}) + \frac{1}{2}h_i(\mathbf{x} - \mathbf{x}^{(i-1)})^T\mathbf{H}_i(\mathbf{x} - \mathbf{x}^{(i-1)}) \tag{2.26}$$

where $\mathbf{H}_i$ is fixed and equal to the Hessian of $q^{(i)}_*$ and where h_i represents a scaling variable. One can formulate again a least square fitting problem as the minimization of the sum of squares of the difference between $\Phi^{(k)}$ and the new $q^{(i)}$,

$$\min_{(\gamma_i, \mathbf{g}_i, h_i)} \sum_{\mathbf{x} \in T^{(i)}_*} \| \, q^{(i)}(\mathbf{x}) - \Phi^{(k)}(\mathbf{x}) \, \|_2^2 \tag{2.27}$$

where $T^{(i)}_* \subseteq S^{(i)}$. This second least square fit is performed with a set of evaluated points $T^{(i)}_*$ that is strictly contained in $T^{(i)}$ since the number of independent parameters is equal to $1 + n + 1 = n + 2$.

Suppose now the case where $2n+1 \leq s^{(i)} < \frac{1}{2}(n+1)(n+2)$. Let $\mathbf{H}_i$ be a diagonal Hessian matrix. The gradient $\mathbf{g}_i$ contains n independent parameters. Thus, $q^{(i)}$ contains $1+n+n = 2n+1$ independent parameters, which implies that there are sufficiently many evaluated points $T^{(i)} \subseteq S^{(i)}$ for which (2.22) is feasible. Let $\mathbf{x}_b$ be a point in $S^{(i)}$ such that $R^{(i)}(\mathbf{x}_b) = 2n + 1$. Let $T^{(i)}$ be the set containing all points of $S^{(i)}$ that are at scaled distance (see (2.23)) smaller than $\| \, \mathbf{x}_b - \mathbf{x}^{(i-1)} \, \|_\Omega \cdot (1 + \varepsilon_T)$ from $\mathbf{x}^{(i-1)}$,

$$T^{(i)} = \{\mathbf{x} \in S^{(i)} \mid \| \, \mathbf{x} - \mathbf{x}^{(i-1)} \, \|_\Omega < \| \, \mathbf{x}_b - \mathbf{x}^{(i-1)} \, \|_\Omega \cdot (1 + \varepsilon_T)\} \tag{2.28}$$

for some ε_T. To solve (2.22) for given $T^{(i)}$ and $\mathbf{H}_i$ we write (2.22) as a set of equations as in (2.25),

[3]This algorithm is decribed in [10].

$$\begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} & \frac{1}{2}x_{11}^2 & \frac{1}{2}x_{12}^2 & \dots & \frac{1}{2}x_{1n}^2 \\ 1 & x_{21} & x_{22} & \dots & x_{2n} & \frac{1}{2}x_{21}^2 & \frac{1}{2}x_{21}^2 & \dots & \frac{1}{2}x_{2n}^2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{p1} & x_{p2} & \dots & x_{pn} & \frac{1}{2}x_{p1}x_{p1} & \frac{1}{2}x_{p2}^2 & \dots & \frac{1}{2}x_{pn}^2 \end{pmatrix} \times \begin{pmatrix} \gamma_i \\ g_{i1} \\ g_{i2} \\ \vdots \\ g_{in} \\ h_{11} \\ h_{22} \\ \vdots \\ h_{nn} \end{pmatrix} = \begin{pmatrix} q^{(i)}(\mathbf{x}_1) \\ q^{(i)}(\mathbf{x}_2) \\ \vdots \\ q^{(i)}(\mathbf{x}_p) \end{pmatrix} \tag{2.29}$$

where $T^{(i)} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$ for a fixed $p \in \mathbb{Z}$. Then, (2.29) is solved with the same techniques used for solving (2.25) and an expression for $q^{(i)}$ is constructed. After that, redefine $T^{(i)}$ in (2.28) by taking $\mathbf{x}_b = \mathbf{x}_{a'}$. Then, (2.27), a second least square fitting problem for $q^{(i)}$, is solved for a fixed but scaled Hessian.

Suppose now the case where $n + 1 \leq s^{(i)} < 2n + 1$. Define $\mathbf{H}_i = \emptyset$.[4] The gradient $\mathbf{g}_i$ contains n independent parameters. Thus, $q^{(i)}$ contains $n + 1$ independent parameters, which implies that there are sufficiently many evaluated points $T^{(i)} \subseteq S^{(i)}$ for which (2.22) is feasible. Let $\mathbf{x}_c$ be a point in $S^{(i)}$ such that $R^{(i)}(\mathbf{x}_c) = n + 1$. Let $T^{(i)}$ be the set containing all points of $S^{(i)}$ that are at scaled distance (see (2.23)) smaller than $\| \mathbf{x}_c - \mathbf{x}^{(i-1)} \|_\Omega \cdot (1 + \varepsilon_T)$ from $\mathbf{x}^{(i-1)}$,

$$T^{(i)} = \{\mathbf{x} \in S^{(i)} \mid \| \mathbf{x} - \mathbf{x}^{(i-1)} \|_\Omega < \| \mathbf{x}_c - \mathbf{x}^{(i-1)} \|_\Omega \cdot (1 + \varepsilon_T)\} \tag{2.30}$$

for some ε_T. To solve (2.22) for given $T^{(i)}$ and $\mathbf{H}_i$ we write (2.22) as a set of equations as in (2.25),

$$\begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} & \mathbf{x}^T\mathbf{H}_i\mathbf{x} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} & \mathbf{x}^T\mathbf{H}_i\mathbf{x} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{p1} & x_{p2} & \dots & x_{pn} & \mathbf{x}^T\mathbf{H}_i\mathbf{x} \end{pmatrix} \times \begin{pmatrix} \gamma_i \\ g_{i1} \\ g_{i2} \\ \vdots \\ g_{in} \\ h_i \end{pmatrix} = \begin{pmatrix} q^{(i)}(\mathbf{x}_1) \\ q^{(i)}(\mathbf{x}_2) \\ \vdots \\ q^{(i)}(\mathbf{x}_p) \end{pmatrix} \tag{2.31}$$

where $T^{(i)} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$ for a fixed $p \in \mathbb{Z}$. Then, (2.31) is solved with the same techniques used for solving (2.25) and a solution $q^{(i)}$ is returned.

Suppose that (2.22) is solved for $T^{(i)}$ defined by one of the three described situations. Define a trust region,

$$\mathcal{B}^{(i)} = \{\mathbf{x} \in \mathbb{R}^n \mid \| \mathbf{x} - \mathbf{x}^{(i-1)} \|_\infty \leq \Delta^{(i-1)}\} \tag{2.32}$$

where $\| \cdot \|_\infty$ is the l^∞-norm, defined as $\| \mathbf{x} \|_\infty = \max_i |x_i|$. Since the infinity norm is used in (2.32) the trust region $\mathcal{B}^{(i)}$ has a square or (hyper-) cubic form. With (2.32) one can apply an algorithm[5] to find $\mathbf{x}^{\text{opt}} \in \mathcal{B}^{(i)}$ such that

$$\mathbf{x}^{\text{opt}} = \min\{\mathbf{x} \in \mathcal{B}^{(i)} \mid \forall\, \mathbf{y} \in \mathcal{B}^{(i)} \, [\, q^{(i)}(\mathbf{x}) \leq q^{(i)}(\mathbf{y}) \,]\} \tag{2.33}$$

[4] Thus (2.21) becomes a linear least-square fitting problem.

[5] In the implementation of Gridmin a black box method is called, see [9] for a similar algorithm.

Let $\mathbf{x}^{(i)}$ be the grid point nearest to $\mathbf{x}^{\text{opt}}$. If $\mathbf{x}^{(i)} \in S^{(i)}$, then Gridmin leaves the descent phase.

Otherwise, the values of the new reference point $\mathbf{x}^{(i)}$ and the trust region radius $\Delta^{(i)}$ are updated as follows; In the first place, define a ratio ρ between $\Phi^{(k)}$ and $q^{(i)}$,

$$\rho = \frac{\Phi^{(k)}(\mathbf{x}^{(i-1)}) - \Phi^{(k)}(\mathbf{x}^{(i)})}{q^{(i)}(\mathbf{x}^{(i-1)}) - q^{(i)}(\mathbf{x}^{(i)})} \tag{2.34}$$

The trust region is updated according to the following update rules,

$$\begin{aligned} \rho \geq \rho_2 \quad &\Rightarrow \quad \Delta^{(i)} = \min\{2\Delta^{(i-1)}, \Delta_{\max}\} \\ \rho < \rho_1 \quad &\Rightarrow \quad \Delta^{(i)} = \max\{\tfrac{1}{2}\Delta^{(i-1)}, \Delta_{\min}\} \\ \rho_1 \leq \rho \leq \rho_2 \quad &\Rightarrow \quad \Delta^{(i)} = \Delta^{(i-1)}. \end{aligned} \tag{2.35}$$

where the choice of $\Delta_{\min}$ is related to the grid spacing.

In the second place, define $\mathbf{x}_{\min}^{(i)} = \arg\min_{\mathbf{x} \in S^{(i)}} \{\Phi^{(k)}(\mathbf{x})\}$. Consider a ratio ρ' between $\Phi^{(k)}$ and $q^{(i)}$,

$$\rho' = \frac{\Phi^{(k)}(\mathbf{x}^{(i-1)}) - \Phi^{(k)}(\mathbf{x}_{\min}^{(i)})}{q^{(i)}(\mathbf{x}^{(i-1)}) - q^{(i)}(\mathbf{x}^{(i)})} \tag{2.36}$$

The new reference point is updated according to the following update rules,

$$\begin{aligned} \rho' \geq \rho_1 \quad &\Rightarrow \quad \mathbf{x}^{(i)} \leftarrow \arg\min\{\Phi^{(k)}(\mathbf{x}_{\min}^{(k,i)}), \Phi^{(k)}(\mathbf{x}^{(i)})\} \\ \rho' < \rho_1 \quad &\Rightarrow \quad \text{no changes in value of } \mathbf{x}^{(i)}. \end{aligned} \tag{2.37}$$

Finally, if $\Phi^{(k)}(\mathbf{x}^{(i)}) < \Phi^{(k)}(\mathbf{x}^{(i-1)})$ then $c \leftarrow 0$ else $c \leftarrow c + 1$. If $c \geq c_{\max}$ then Gridmin also leaves the descent phase.

2.5.5 Refinement-check phase

The refinement-check phase checks if the minimum found in the descent phase could be improved. At the start a least square fit is applied to the evaluated points to build a bound-constrained linear approximation function. Then the minimum of the bound-constrained function is determined. If this minimum is not better than the minimum obtained in the descent phase, a quadratic approximation function is constructed and minimized as in the descent phase. At last, some parameters are updated.

Define, at the start of the process, S the set of all evaluated points in Gridmom and define a reference point $\mathbf{x}^*$ equal to the point where a minimum of $\Phi^{(k)}$ is attained among all points in S. Let $\varepsilon_S > 0$ be equivalent to ε_T defined in subsection 2.5.4. Also, define a grid spacing variable $g_{\text{sp}} > 0$, a range variable r and at last a counter $c = 0$.

Let l be a linear approximation function,

$$l(\mathbf{x}) = \gamma + \mathbf{g}^T(\mathbf{x} - \mathbf{x}^*) \tag{2.38}$$

where $\mathbf{g}$ denotes the gradient of l. One can formulate a least square fitting problem as the minimization of the sum of squares of the difference between $\Phi^{(k)}$ and l,

$$\min_{(\gamma, \mathbf{g})} \sum_{\mathbf{x} \in T} \| \, l(\mathbf{x}) - \Phi^{(k)}(\mathbf{x}) \, \|_2^2 \tag{2.39}$$

for some $T \subseteq S$. Define the distance between two points as in (2.23). Let T contain all points of S that are at distance closer than one grid spacing times a small factor,

$$T = \{\mathbf{x} \in S | \{\| \mathbf{x} - \mathbf{x}^* \|_\Omega < g_{\text{sp}}(1 + \varepsilon_S)\} \tag{2.40}$$

Suppose we have $T = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p\}$ for some $p \in \mathbb{Z}$, where $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jn})^T$ for all $j \in \{1, 2, \ldots, p\}$ as performed in the descent phase. We need to minimize $\| \mathbf{Ay} - \mathbf{b} \|_2^2$, where $\mathbf{Ay} = (l(\mathbf{x}_1), l(\mathbf{x}_2), \ldots, l(\mathbf{x}_p))^T$ and where $\mathbf{b} = (\Phi^{(k)}(\mathbf{x}_1), \Phi^{(k)}(\mathbf{x}_2), \ldots, \Phi^{(k)}(\mathbf{x}_p))$. Substituting (2.38) leads to the minimization problem $\min\{\| \mathbf{Ay} - \mathbf{b} \|_2^2\}$, where,

$$\mathbf{A} = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{p1} & x_{p2} & \ldots & x_{pn} \end{pmatrix} \quad , \quad \text{y} = \begin{pmatrix} \gamma_i \\ g_{i1} \\ g_{i2} \\ \vdots \\ g_{in} \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} q^{(i)}(\mathbf{x}_1) \\ q^{(i)}(\mathbf{x}_2) \\ \vdots \\ q^{(i)}(\mathbf{x}_p) \end{pmatrix} \tag{2.41}$$

and where $\mathbf{g} = (g_1, g_2, \ldots, g_n)^T$. Let t be the size of T. In the case where $\mathbf{A}$ does not have full column rank two cases must be distinguished.

Consider the case where $t < n + 1$. Then, new points are added to T till (2.41) is feasible. These points are chosen in such a way that they are close to the reference point $\mathbf{x}^*$; take the gradient of the last fitted quadratic approximation function, say $\mathbf{g} = (g_1, g_2, \ldots, g_n)$. Define a mapping $R : \{g_1, g_2, \ldots, g_n\} \mapsto \{1, 2, \ldots, n\}$ such that $R(g_i)$ implies that $\sum_{j=1}^{n} 1_{g_j < g_i} = r - 1$. Then points are evaluated and added to T according to an iterative process consisting of a maximum of $n + 1 - t$ iterations. Thus, consider the i-th iteration, $i \geq 1$. Let $j \in \mathbb{Z}^+$ be a index such that $r(g_j) = i$. Then, the point $\mathbf{x}^* + d\mathbf{e}_i$, where $d \in \mathbb{Z}^+$ is chosen such that $\mathbf{x}^* + d\mathbf{e}_i \in \Omega$, is evaluated and added to T. Also, update the counter: $c \leftarrow c + 1$.

Also, consider the case where $t \geq n + 1$. Then, other evaluated points are added to T till (2.41) is feasible. Consider an iterative process and consider the i-th iteration, $i \geq 1$. Let $j \in \mathbb{Z}^+$ a index such that $r(g_j) = c + i$. Then, the point $\mathbf{x}^* + d\mathbf{e}_i$, where $d \in \mathbb{Z}^+$ is chosen such that $\mathbf{x}^* + d\mathbf{e}_i \in \Omega$, is evaluated and added to T.

Finally, the set of equations $\mathbf{Ay} = \mathbf{b}$ is updated by adding the equation $l(\mathbf{x}^* + d\mathbf{e}_{c+i}) = \Phi^{(k)}(\mathbf{x}^* + d\mathbf{e}_{c+i})$. The iterative processes of both cases terminate when the updated matrix of $\mathbf{A}$ has full column rank.

When $\mathbf{A}$ has full column rank, (2.39) is solved and an expression for l is returned. Also, S is updated, by means of adding all new evaluated points: $S \leftarrow S \cup T$. For minimizing l two cases must be considered.

Consider the case in which there exists a point $\mathbf{x} \in S$, $\mathbf{x} \neq \mathbf{x}^*$, such that $\Phi^{(k)}(\mathbf{x}) < \Phi^{(k)}(\mathbf{x}^*)$. Then, Gridmin leaves the refinement-check phase.

Also, consider the case in which for all $\mathbf{x} \in S$ we have $\Phi^{(k)}(\mathbf{x}) \geq \Phi^{(k)}(\mathbf{x}^*)$. Define a sign function f_{sgn} such that,

$$f_{\text{sgn}}(\mathbf{x}) = D\mathbf{x} \tag{2.42}$$

where D is the diagonal matrix with entries $|x_i|^{-1}$ for all $i = 1, \ldots, n$. Furthermore if $x_i = 0$ then we define $|x_i|^{-1}$ to be 1. With the found expression for l one tries to minimize l within Ω by means of defining a new point $\mathbf{x}'$ and comparing $\Phi^{(k)}(\mathbf{x}')$ to $\Phi^{(k)}(\mathbf{x}^*)$; let $f_{\min}$ and $f_{\max}$ be functions such that

$$\begin{aligned} f_{\min}(\mathbf{x}, \mathbf{y}) &= (\min\{x_1, y_1\}, \min\{x_2, y_2\}, \ldots, \min\{x_n, y_n\}) \\ f_{\max}(\mathbf{x}, \mathbf{y}) &= (\max\{x_1, y_1\}, \max\{x_2, y_2\}, \ldots, \max\{x_n, y_n\}) \end{aligned} \tag{2.43}$$

Define

$$\mathbf{x}' = f_{\max}(f_{\min}(\mathbf{x}^* - f_{\text{sgn}} \cdot g_{\text{sp}} \cdot r, \mathbf{b}), \mathbf{a}) \tag{2.44}$$

where $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$. The choice of r is related to the size of $\| \mathbf{x}_{\text{start}} \|$ and $|b_i - a_i|$ for all $i \in \{1, 2, \ldots, n\}$.

If $\mathbf{x}' \in S$ then $\mathbf{x}'$ is added to S and evaluated.

If $\mathbf{x}' \notin S$, or if $\mathbf{x}' \in S$ such that $\Phi^{(k)}(\mathbf{x}') < \Phi^{(k)}(\mathbf{x}^*)$, then a least-square fit is applied to a subset $S' \subseteq S$ to build a bound constrained quadratic approximation function q. Subsequently a trust region method is applied to q to calculate the minimum. The constructing and minimizing process of q is equal to the process of $q^{(i)}$ in the i-th iteration of the descent phase, see subsection 2.5.4. Let c_{desc} be the number of iterations Gridmin went through in the descent phase. The trust region Δ belonging to the minimizing process of q are equivalent to $\Delta^{(c_{\text{desc}})}$. Let $\mathbf{x}''$ be the minimum of q. Then, if $\mathbf{x}'' \in S$, the point $\mathbf{x}''$ is added to S and evaluated.

Finally, if $\mathbf{x}'' \in S$ or $\mathbf{x}'' \in S \Rightarrow \Phi^{(k)}(\mathbf{x}'') < \Phi^{(k)}(\mathbf{x}^*)$, then the grid spacing g_{sp} is refined as follows,

$$g_{\text{sp}} \leftarrow \left(\frac{n_l}{n_l + 1} \right) g_{\text{sp}} \tag{2.45}$$

where n_l is equal to the number of times Gridmin has entered the refinement-check phase, including the current one.

Chapter 3

State of the Art: Kriging

3.1 Introduction

In this chapter we discuss the process of Kriging modelling[1]. Kriging modelling is a modelling technique based on linear regression through a set of points. Section 3.2 is about constructing a Kriging model by fitting some variables with help of an experimental design. Section 3.3 is about prediction with respect to the Kriging model. In section 3.4 it is explained how to see if the obtained Kriging model works well for a set of sampled points. Section 3.5 is about an optimization algorithm that exploits the Kriging modelling techniques. The algorithm tries to select certain points such that a Kriging model in these points give the best representation of the behavior of the input function around the global minimum. Finally, we discuss some extra items concerning Kriging modelling and the mentioned optimization algorithm in section 3.6.

3.2 Model construction

Suppose we want to model a deterministic response $y(\mathbf{x})$ as a realization of a stochastic process Y:

$$Y(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\beta + \epsilon(\mathbf{x}) \tag{3.1}$$

where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x})\ f_2(\mathbf{x})\ \cdots\ f_k(\mathbf{x}))^T$ is a vector of functions $f_i : \mathbb{R}^n \mapsto \mathbb{R}$, $i \in \{1, 2, \ldots, k\}$, $\beta = (\beta_1\ \beta_2\ \cdots\ \beta_k)^T$ a vector of unknown coefficients to be estimated and the error $\epsilon(\mathbf{x})$ a normal distributed variable with mean zero and variance σ^2. Thus we are dealing with a regression model where $\mathbf{f}^T(\mathbf{x})\beta$ and $\epsilon(\mathbf{x})$ represent the regression and correlation terms respectively. Furthermore, let $S = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$ be an experimental design such that $\mathbf{x}^{(i)} = (x_1^{(i)}\ x_2^{(i)}\ \cdots\ x_n^{(i)})^T$, $x_j^{(i)} \in \mathbb{R}$ for all i, j and let $\mathbf{y} = (y(\mathbf{x}^{(1)})\ y(\mathbf{x}^{(2)})\ \cdots\ y(\mathbf{x}^{(N)}))^T$ be the corresponding response data.

The conceptual problem in (3.1) is that we cannot assume that the errors are independent when modelling a deterministic system. In a deterministic system any lack of fit will be entirely modelling error in stead of measurement error or noise. This means that the error terms are really collections of left-out terms in $\mathbf{x}^{(i)}$.

Consider $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in S$ for some $i, j \in \{1, 2, \ldots, N\}$, $i \neq j$, such that they are close together. Then the errors $\epsilon(\mathbf{x}^{(i)})$ and $\epsilon(\mathbf{x}^{(j)})$ should also be close. Thus, it makes no sense to assume that $\epsilon(\mathbf{x}^{(i)})$ and $\epsilon(\mathbf{x}^{(j)})$ are independent. It is better to say that they are correlated, where the correlation is high if $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are close and low if $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are far apart. So, there is a certain relation between the

[1] For additional information on Kriging modelling, see [4, 11]

distance of two sampled points and their correlation. We do not use the Euclidian distance, however, since this measurement weights all variables equally. Define a new distance d,

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_{h=1}^{n} \theta_h |x_h^{(i)} - x_h^{(j)}|^{p_h} \quad (\theta_h > 0,\ p_h \in [1, 2])$$

and let the correlation be

$$C(\epsilon(\mathbf{x}^{(i)}), \epsilon(\mathbf{x}^{(j)})) = e^{-d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})} \quad \in [0, 1]$$

If the distance d is small, then $C \to 1$, and if the distance is large, then $C \to 0$. Note that θ_h is a measurement for importance and activity of the h-th coordinate. For example, if θ_h is large, then d will also be large and the correlation will be low. Also, note that p_h is related to the smoothness of the function in coordinate direction h.

Thus, let $\mathbf{R} = (r_{ij})$ the matrix of correlations of the points in S that is, $r_{ij} = C(\epsilon(\mathbf{x}^{(i)}), \epsilon(\mathbf{x}^{(j)}))$, and define the covariance between two points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$, $i, j \in \{1, 2, \ldots, n\}$, as $\mathrm{Cov}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \mathbb{E}[\epsilon(\mathbf{x}^{(i)})\epsilon(\mathbf{x}^{(j)})] - \mathbb{E}[\epsilon(\mathbf{x}^{(i)})]\mathbb{E}[\epsilon(\mathbf{x}^{(j)})] = \mathbb{E}[\epsilon(\mathbf{x}^{(i)})\epsilon(\mathbf{x}^{(j)})] = \sigma^2 r_{ij}$. This extended regression model, where the correlation of the errors is related to the distance of the design sites, is also called a *Kriging* model.

The Kriging model has $2n + k + 1$ parameters: $\beta_1, \beta_2, \ldots, \beta_k, \sigma, \theta_1, \theta_2, \ldots, \theta_n, p_1, p_2, \ldots, p_n$. We estimate these parameters using maximum likelihood estimation (MLE) on the design S. Let $\mathbf{F} = (f_{ij})$ be a matrix such that $f_{ij} = f_j(\mathbf{x}^{(i)})$. The likelihood function is formulated as

$$\begin{aligned} L(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}) &= \frac{1}{\sqrt{|\det(\mathbf{R})|}} \prod_{i=1}^{n} \mathbb{P}(Y = \mathbf{x}^{(i)} | \beta, \sigma) \\ &= \frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}\sqrt{|\det(\mathbf{R})|}}\, e^{-\frac{(\mathbf{y}-\mathbf{F}\beta)^T)\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)}{2\sigma^2}} \end{aligned} \tag{3.2}$$

Assuming that $\theta_1, \theta_2, \ldots, \theta_n$ and $p_1, p_2, \ldots, p_n$ are fixed, it is possible to obtain expressions for the remaining parameters β and σ, say $\hat{\beta} = (\hat{\beta}_1 \hat{\beta}_2 \cdots \hat{\beta}_k)^T$ and $\hat{\sigma}$, that maximizes the likelihood function. In doing this, we first define a function l that is equal to the logarithm of L of (3.2),

$$\begin{aligned} l(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}) &= \log(L(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})) \\ &= -\tfrac{n}{2}\log(2\pi) - n\log(\sigma) - \tfrac{1}{2}\log(\det(|\mathbf{R}|)) - \tfrac{(\mathbf{y}-\mathbf{F}\beta)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)}{2\sigma^2} \end{aligned}$$

Now we need to find an expression for $\hat{\beta}$ and $\hat{\sigma}$ by solving $\frac{\partial l}{\partial \beta} = 0$ and $\frac{\partial l}{\partial \sigma} = 0$ respectively. In the first place, we obtain an expression for $\hat{\beta}$. We need to solve the following equation,

$$\frac{\partial l}{\partial \beta} = 0 \quad \Leftrightarrow \quad \frac{\partial}{\partial \beta}\left\{\frac{(\mathbf{y} - \mathbf{F}\beta)^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\beta)}{2\sigma^2}\right\} = 0$$

Since σ is independent of β we solve the following equation,

$$\frac{\partial}{\partial \beta}\left\{(\mathbf{y} - \mathbf{F}\beta)^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\beta)\right\} = 0$$

Using the vector derivative rules of Appendix A.1 one obtains

$$\begin{aligned}
\frac{\partial}{\partial\beta}\left\{(\mathbf{y}-\mathbf{F}\beta)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)\right\} &= \frac{\partial}{\partial\beta}\left\{(\mathbf{y}-\beta^T\mathbf{F}^T)\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)\right\} \\
&= \frac{\partial}{\partial\beta}\left\{\mathbf{y}^T\mathbf{R}^{-1}\mathbf{y}\right\} - \frac{\partial}{\partial\beta}\left\{\beta^T\mathbf{F}^T\mathbf{R}^{-1}\mathbf{y}\right\} - \\
&\quad \frac{\partial}{\partial\beta}\left\{\mathbf{y}^T\mathbf{R}^{-1}\mathbf{F}\beta\right\} + \frac{\partial}{\partial\beta}\left\{\beta^T\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}\beta\right\} \\
&= \mathbf{0} - \mathbf{F}^T\mathbf{R}^{-1}\mathbf{y} - (\mathbf{y}^T\mathbf{R}^{-1}\mathbf{F})^T + (\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^T\beta + \\
&\quad \mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}\beta
\end{aligned} \tag{3.3}$$

Since $\mathbf{R}^{-1}$ is symmetric we have $(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^T = \mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}$. Including this result into (3.3) gives us the following matrix equation:

$$\mathbf{F}^T\mathbf{R}^{-1}\mathbf{y} = \mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}\beta$$

Thus, the maximum likelihood estimator of β is equal to

$$\hat{\beta} = (\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^T\mathbf{R}^{-1}\mathbf{y}$$

Finally, we obtain an expression for $\hat{\sigma}$ by solving the following equation,

$$\frac{\partial l}{\partial\sigma} = 0 \quad \Leftrightarrow \quad \frac{\partial}{\partial\sigma}\left\{-n\log\sigma - \frac{(\mathbf{y}-\mathbf{F}\beta)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)}{2\sigma^2}\right\} = 0$$

After computing the partial derivative with respect to σ one gets

$$-\frac{n}{\sigma} + \frac{(\mathbf{y}-\mathbf{F}\beta)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)}{\sigma^3} = 0 \quad \Leftrightarrow \quad \frac{n}{\sigma} = \frac{(\mathbf{y}-\mathbf{F}\beta)^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)}{\sigma^3}$$

Thus, the maximum likelihood estimator of σ is equal to

$$\hat{\sigma} = \sqrt{\frac{(\mathbf{y}-\mathbf{F}\hat{\beta})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\hat{\beta})}{n}}$$

If the correlation parameters are not fixed, then $\hat{\beta}$ and $\hat{\sigma}$ can be seen as functions in the correlation variables $\theta_1, \theta_2, \ldots, \theta_k$, $p_1, p_2, \ldots, p_{k-1}$ and p_k. After substitution of $\hat{\beta}$ and $\hat{\sigma}$ in (3.2) one obtains a function expressed in the variables $\theta_1, \theta_2, \ldots, \theta_k$, $p_1, p_2, \ldots, p_{k-1}$ and p_k only. From this function one obtains expressions for the estimators of these variables by solving $\frac{\partial l}{\partial\theta_j} = 0$ and $\frac{\partial l}{\partial p_j} = 0$ for all $j \in \{1, 2, \ldots, k\}$.

3.3 Prediction

Now we know how to construct a Kriging model and how to obtain estimators for the variables in the model, it is important to get a good understanding of the impact of correlated errors on prediction in our model. Define $\mathbf{x}^*$ to be the point at which we are predicting y. Let $\mathbf{r}(\mathbf{x}^*) = (r_1(\mathbf{x}^*)\ r_2(\mathbf{x}^*)\ \cdots\ r_n(\mathbf{x}^*))^T$, with $r_i(\mathbf{x}^*) = C(\epsilon(\mathbf{x}^*), \epsilon(\mathbf{x}^{(i)}))$ for all $i \in \{1, 2, \ldots, n\}$, denote the n-vector of correlations between $\epsilon(\mathbf{x}^*)$ and the error terms at the design sites of $\mathcal{S}$. Now consider the linear predictor

$$\hat{y}(\mathbf{x}^*) = \mathbf{c}^T(\mathbf{x}^*)\mathbf{y} \tag{3.4}$$

where $\mathbf{c}(\mathbf{x}^*) \in \mathbb{R}^N$ is deterministic but which still has to be defined. Now we are able to construct an expression for the prediction error. Let $\epsilon = (\epsilon(\mathbf{x}^{(1)})\, \epsilon(\mathbf{x}^{(2)})\, \cdots\, \epsilon(\mathbf{x}^{(N)}))$. Then one gets

$$\begin{aligned} \hat{y}(\mathbf{x}^*) - y(\mathbf{x}^*) &= \mathbf{c}^T(\mathbf{x}^*)\, \mathbf{y} - y(\mathbf{x}^*) \\ &= \mathbf{c}^T(\mathbf{x}^*)(\mathbf{F}\beta + \epsilon) - (f(\mathbf{x}^*)^T\beta + \epsilon(\mathbf{x}^*)) \\ &= \mathbf{c}^T(\mathbf{x}^*)\, \epsilon - \epsilon(\mathbf{x}^*) + (\mathbf{F}^T\mathbf{c}(\mathbf{x}^*) - f(\mathbf{x}^*))^T\beta \end{aligned}$$

To keep the predictor unbiased with respect to β we demand that $\mathbf{F}^T\mathbf{c}(\mathbf{x}^*) = f(\mathbf{x}^*)$. Under this condition the mean squared error s of the predictor in (3.4) is equal to

$$\begin{aligned} s^2(\mathbf{x}^*) &= \mathbb{E}[\{\hat{\mathbf{y}}(\mathbf{x}^*) - \mathbf{y}(\mathbf{x}^*)\}^2] \\ &= \mathbb{E}[\{\mathbf{c}^T(\mathbf{x}^*)\epsilon - \epsilon(\mathbf{x}^*)\}^2] \\ &= \mathbb{E}[\epsilon^2(\mathbf{x}^*) + \mathbf{c}^T(\mathbf{x}^*)\epsilon\, \epsilon^T\mathbf{c}(\mathbf{x}^*) - 2\mathbf{c}^T(\mathbf{x}^*)\epsilon\, \epsilon(\mathbf{x}^*)] \end{aligned}$$

We have $\mathbb{E}[\epsilon^2(\mathbf{x}^*)] = \sigma^2$, since ϵ is normal distributed with mean zero and variance σ^2. Furthermore, $\mathbb{E}[\epsilon\, \epsilon^T] = \sigma^2\mathbf{R}$ since $\mathrm{Cov}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \mathbb{E}[\epsilon(\mathbf{x}^{(i)}), \epsilon(\mathbf{x}^{(j)})] = \sigma^2 r_{ij}$ for all $i, j \in \{1, 2, \ldots, N\}$. These two expectations and the definition of $\mathbf{r}$ lead to another expression:

$$\mathbb{E}[\epsilon\, \epsilon(\mathbf{x}^*)] = \sigma^2\mathbf{r}(\mathbf{x}^*)$$

Combined with the two other expectations we get the following expression for the mean squared error s of the predictor,

$$s^2(\mathbf{x}^*) = \sigma^2(1 + \mathbf{c}^T(\mathbf{x}^*)\mathbf{R}\, \mathbf{c}(\mathbf{x}^*) - 2\mathbf{c}^T(\mathbf{x}^*)\mathbf{r}(\mathbf{x}^*)) \tag{3.5}$$

The idea is to keep (3.5) as low as possible, thus we need to minimize (3.5) with respect to $\mathbf{c}$ and subject to the constraint $\mathbf{F}^T\mathbf{c}(\mathbf{x}^*) = f(\mathbf{x}^*)$. Define a Lagrangian function corresponding to this minimization problem in $\mathbf{x}^*$,

$$\mathcal{L}_{\mathbf{x}^*}(\mathbf{c}; \lambda) = \sigma^2(1 + \mathbf{c}^T(\mathbf{x}^*)\, \mathbf{R}\, \mathbf{c}(\mathbf{x}^*) - 2\mathbf{c}^T(\mathbf{x}^*)\, \mathbf{r}(\mathbf{x}^*)) - \lambda^T(\mathbf{F}^T\mathbf{c}(\mathbf{x}^*) - f(\mathbf{x}^*))$$

where $\lambda \in \mathbb{R}^k$. We need to find $\mathbf{c}$ and λ such that the partial derivatives of this Lagrangian function with respect to $\mathbf{c}$ and λ in the predicting point $\mathbf{x}^*$ are equal to zero:

$$\begin{aligned} \frac{\partial\mathcal{L}_{\mathbf{x}^*}}{\partial\mathbf{c}(\mathbf{x}^*)} &= \sigma^2\{(\mathbf{R} + \mathbf{R}^T)\mathbf{c}(\mathbf{x}^*) - 2\mathbf{r}(\mathbf{x}^*)\} - \mathbf{F}\lambda &&= 0 \\ \frac{\partial\mathcal{L}_{\mathbf{x}^*}}{\partial\lambda} &= \mathbf{F}^T\mathbf{c}(\mathbf{x}^*) - \mathbf{f}(\mathbf{x}^*) &&= 0 \end{aligned} \tag{3.6}$$

Define $\tilde{\lambda} = -\frac{\lambda}{2\sigma^2}$ and substitute $\tilde{\lambda}$ into (3.6). Together with (3.6) and noticing that $\mathbf{R}^T = \mathbf{R}$ we obtain one system of equations,

$$\begin{pmatrix} \mathbf{F}^T & \mathbf{0} \\ \mathbf{R} & \mathbf{F} \end{pmatrix} \begin{pmatrix} \mathbf{c}(\mathbf{x}^*) \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} f(\mathbf{x}^*) \\ \mathbf{r}(\mathbf{x}^*) \end{pmatrix} \tag{3.7}$$

With help of row- and column operations we are able to solve (3.7); multiplying the second row by $\mathbf{F}^T\mathbf{R}^{-1}$, subsequently subtracting the first row from the second row and then multiplying the new second row with $(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^{-1}$ gives the following result:

$$\begin{pmatrix} \mathbf{F}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{c}(\mathbf{x}^*) \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} f(\mathbf{x}^*) \\ (\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^{-1}(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) - f(\mathbf{x}^*)) \end{pmatrix}$$

yielding

$$\tilde{\lambda} = (\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^{-1}(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) - f(\mathbf{x}^*))$$

and substituting this result into (3.7) gives us

$$\mathbf{c}(\mathbf{x}^*) = \mathbf{R}^{-1}(\mathbf{r}(\mathbf{x}^*) - \mathbf{F}\tilde{\lambda})$$

Since $\mathbf{R}$ is symmetric and therefore also $\mathbf{R}^{-1}$ we have $(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^T = \mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}$. Thus,

$$\begin{aligned} \hat{y}(\mathbf{x}^*) \;=\; \mathbf{c}^T(\mathbf{x}^*)\mathbf{y} \quad &= \quad (\mathbf{r}(\mathbf{x}^*)) - \mathbf{F}\tilde{\lambda})^T\mathbf{R}^{-1}\mathbf{y} \\ &= \quad \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{y} - (\mathbf{F}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) - f(\mathbf{x}^*))^T(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^T\mathbf{R}^{-1}\mathbf{y} \end{aligned} \tag{3.8}$$

Now substitute $\hat{\beta}$ into (3.8) and we get the predictor

$$\begin{aligned} \hat{y}(\mathbf{x}^*) \quad &= \quad \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{y} - (\mathbf{F}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) - f(\mathbf{x}^*))^T\hat{\beta} \\ &= \quad f^T(\mathbf{x}^*)\hat{\beta} + \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\hat{\beta}) \end{aligned} \tag{3.9}$$

Also, substituting $\mathbf{c}$ (and $\tilde{\lambda}$) leads to the variance of the predictor

$$s^2(\mathbf{x}^*) = \sigma^2(1 - \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) + \frac{(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) - f(\mathbf{x}^*))^T(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) - f(\mathbf{x}^*))}{\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}}) \tag{3.10}$$

To see that this predictor interpolates the data, we take $\mathbf{x}^* = \mathbf{x}^{(i)}$ for some $i \in \{1, 2, \ldots, N\}$. Then $\mathbf{r}(\mathbf{x}^*) = \mathbf{r}(\mathbf{x}^{(i)})$ is equivalent to the i-th column of $\mathbf{R}$. Since the columns of $\mathbf{R}^{-1}$ are independent (otherwise $\mathbf{R}^{-1}$ would not exist) we have $\mathbf{r}^T\mathbf{R}^{-1} = (\mathbf{R}^{-1}\mathbf{r})^T = \mathbf{e}_i^T$. Substituting this result and $\mathbf{x}^{(i)}$ into (3.9) leads to the result

$$\begin{aligned} \hat{y}(\mathbf{x}^{(i)}) \quad &= \quad f^T(\mathbf{x}^{(i)})\hat{\beta} + \mathbf{e}_i(\mathbf{y} - \mathbf{F}\hat{\beta}) \\ &= \quad f^T(\mathbf{x}^{(i)})\hat{\beta} + y(\mathbf{x}^{(i)}) - f^T(\mathbf{x}^{(i)})\hat{\beta} \\ &= \quad y(\mathbf{x}^{(i)}) \end{aligned}$$

It turns out that the correlation of the errors affect our estimate of prediction accuracy (intuitively, for some $\mathbf{x}'$ and $\mathbf{x}''$ we should be more confident about $y(\mathbf{x}')$ than about $y(\mathbf{x}'')$ if $d(\mathbf{x}^{(i)}, \mathbf{x}')$ is small for at least one i and $d(\mathbf{x}^{(i)}, \mathbf{x}'')$ is large for all possible i). This is easy to verify; substitute $\mathbf{x}^* = \mathbf{x}^{(i)}$ into (3.10),

$$s^2(\mathbf{x}^{(i)}) = \sigma^2\left\{1 - \mathrm{Cov}(\mathbf{x}^{(i)}, \mathbf{x}^{(i)}) + \frac{(f(\mathbf{x}^{(i)}) - f(\mathbf{x}^{(i)}))^T(f(\mathbf{x}^{(i)}) - f(\mathbf{x}^{(i)}))}{\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}}\right\}$$

which implies that $s^2(\mathbf{x}^{(i)}) = \sigma^2(1 - 1 + 0) = 0$, like it should be. Now we are able to make the following statements about the mean squared error,

- the mean squared error at a sampled point is equal to zero;
- the mean squared error at a point that is very far away from the sampled points is equal to σ or $-\sigma$;
- the square root of the mean square error between these extremes is $\sigma(1 - \varepsilon)$ where $0 < \varepsilon < 1$ and where ε is related to the distance to the sampled points.

We can make similar statements about the predictors. At a sampled point $\mathbf{x}^{(i)}$ we have $\hat{y}(\mathbf{x}^{(i)}) = y(\mathbf{x}^{(i)})$, as seen before. At a point $\mathbf{x}''$ very far away from the sampled points (where $\mathbf{r}$ is as small as possible) $\hat{y}(\mathbf{x}'')$ is about $\mathbf{f}^T(\mathbf{x}'')\beta$. For $\mathbf{x}'$ between these extremes, $\hat{y}(\mathbf{x}')$ is based on a smooth interpolation of the sampled points.

Finally, there are two remarks about the Kriging modelling. In the first place, the weak point here is that we do not always know $\beta_1, \beta_2, \ldots, \beta_k, \sigma^2, \theta_1, \theta_2, \ldots, \theta_N, p_1, p_2, \ldots, p_N$; we only know some expressions for their estimators. For notational reasons we use all parameters without hat from now on. In the second place, the big difference between linear regression and Kriging modelling is that linear regression describes what the function is, but Kriging modelling describes the behavior of the function.

3.4 Model validation

For testing if the obtained Kriging model works well one could use cross-validation; leave a point out of the sampled points and construct a new Kriging model and predict the point left out based on this model. However, if there are very few observations or major outliers, then dropping a single observation could have a large effect on the maximum likelihood estimators. Therefore, in practice, only $\mathbf{R}$ and $\mathbf{r}$ are re-calculated. Then, it is tested if the left-out point satisfies the model by means of the following value,

$$\frac{y(\mathbf{x}^{(i)}) - \hat{y}_{-i}(\mathbf{x}^{(i)})}{s_{-i}(\mathbf{x}^{(i)})}$$

where $\hat{y}_{-i}$ is the predictor constructed with the reduced set of sample points $\mathcal{S}\backslash\{\mathbf{x}^{(i)}\}$ and where s_{-i} is the mean square error constructed with this set. This value must be roughly in the interval $[-3, +3]$.

3.5 Efficient global optimization

3.5.1 Introduction

After obtaining a Kriging model that satisfies cross validation we want to find the minimum of the modelled function. However, our model construction methods put too much emphasis on exploiting in well-known areas instead of on points where we are uncertain about the surrounding surface. It is therefore necessary to put some emphasis on sampling where we are uncertain (there is a certain probability that it provides a better minimum). We use a method concerning the so-called expected improvement function. The idea is to maximize this function and add the maximum to the Kriging model as an addition to the set of sampled points. In the first subsection we discuss the expected improvement function and its advantages in application. Combining the optimization of the expected improvement function and the construction of Kriging models in an efficient way could lead to a good global optimization algorithm. An example is the Efficient Global Optimization algorithm, also called EGO[2]. This algorithm is described in subsection 3.5.3.

3.5.2 Expected improvement

Define $f_{\min} = \min\{y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \ldots, y(\mathbf{x}^{(N)})\}$ and let y be the deterministic computer response as defined in section 3.2. If y is sampled at $\mathbf{x}$ to determine $y(\mathbf{x})$, then the improvement $I(\mathbf{x})$ over $f_{\min}$ is

[2]For additional information, see [4, 11]

defined as

$$I(\mathbf{x}) = \begin{cases} f_{\min} - y(\mathbf{x}) & \text{if } y(\mathbf{x}) < f_{\min} \\ 0 & \text{otherwise} \end{cases} = \max\{0, f_{\min} - y\}$$

Intuitively, the improvement function shows us where y has lower function values than $f_{\min}$, see the figure below

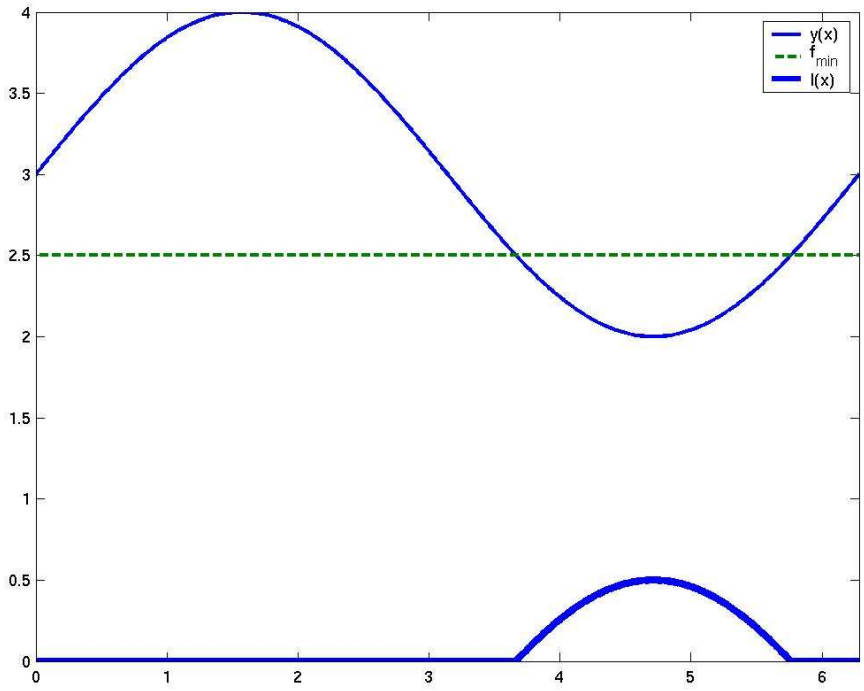

Figure 3.1: A plot of the improvement function

One problem is that we do not know the behavior of y. We will therefore approximate y by a stochastic variable $Y_{\mathbf{x}}$, which is normally distributed with mean $\hat{y}(\mathbf{x})$ and variance $s^2(\mathbf{x})$ for a given point $\mathbf{x}$. We consider here the normal distribution since the Kriging model is a realization of a stochastic process with linear regression terms and normally distributed correlation terms. In fact, $Y_{\mathbf{x}}$ represents the uncertainty of $\hat{y}$ and thus of y.

We consider the expectation of the improvement function for given $\mathbf{x}$, the expected improvement function $\mathbb{E}[I(\mathbf{x})]$. This function is equivalent to the following expression,

$$\mathbb{E}[I(\mathbf{x})] = (f_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \tag{3.11}$$

For the proof of (3.11) we need to know something about the normal distribution and about expectations. Information about these subjects can be found in Appendix A.3. Now the proof is as follows,

$$
\begin{aligned}
\mathbb{E}[I(\mathbf{x})] &= \mathbb{E}[\max_{\mathbf{x}}\{f_{\min} - Y_{\mathbf{x}}, 0\}] \\
&= \mathbb{E}[\max_{\mathbf{x}}\{f_{\min}, Y_{\mathbf{x}}, \}] - \mathbb{E}[Y_{\mathbf{x}}] \\
&= \mathbb{P}(Y_{\mathbf{x}} > f_{\min})\mathbb{E}[Y_{\mathbf{x}}] + \mathbb{P}(Y_{\mathbf{x}} \leq f_{\min})\mathbb{E}[f_{\min}] - \mathbb{E}[Y_{\mathbf{x}}] \\
&= (1 - \mathbb{P}(Y_{\mathbf{x}} \leq f_{\min}))\mathbb{E}Y_{\mathbf{x}} + \mathbb{P}(Y_{\mathbf{x}} \leq f_{\min})\mathbb{E}[f_{\min}] - \mathbb{E}[Y_{\mathbf{x}}] \\
&= (\mathbb{E}[f_{\min}] - \mathbb{E}[Y_{\mathbf{x}}])\mathbb{P}(Y_{\mathbf{x}} \leq f_{\min}) \\
&= (\mathbb{E}[f_{\min}] - \mathbb{E}[Y_{\mathbf{x}}])\mathbb{P}(Y_{\mathbf{x}} < f_{\min}) + \mathbb{E}[f_{\min} - Y_{\mathbf{x}}]\mathbb{P}(Y_{\mathbf{x}} = f_{\min}) \\
&= (f_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\tfrac{f_{\min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\tfrac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)
\end{aligned}
$$

Note that $\mathbb{E}[f_{\min} - Y_{\mathbf{x}}] = s(\mathbf{x})$ in the last line in the second expression since the model is sampled in $\mathbf{x} = \arg\min\{y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \ldots, y(\mathbf{x}^{(N)})\}$.

I want to conclude with two remarks about this expected improvement function,

- Since $f_{\min}$, $\hat{y}(\mathbf{x})$ and $s(\mathbf{x})$ are known, we are able to compute $\mathbb{E}[I(\mathbf{x})]$ for values of $\mathbf{x}$.
- The maximum of this expected improvement function could be interpreted as the point where the probability for a better minimum than the current one is the highest compared to all other points.

3.5.3 EGO algorithm

The EGO algorithm is used for solving an optimization problem where the objective function f is fitted by a Kriging model. The algorithm consists of an iterative process.

At the start of the algorithm three variables are initialized. In the first place the set S which is equal to the empty set. With help of Latin Hypercube Sampling[3] points are added to S. Next, a variable $\mathbf{x}_{\min}$ is initialized and defined as $\mathbf{x}_{\min} = \min\{S\}$. Finally, the set T is initialized and contains all evaluations of the elements of S.

Consider the i-th iteration. In the first place, a Kriging model function is constructed over S and T and the unknown parameters of the model are estimated by maximum likelihood estimation. If this model does not satisfy cross validation, then a log- or inverse transformation is applied to the Kriging model and the unknown parameters are re-estimated. If even the transformed model does not satisfy cross validation, EGO terminates and returns $\mathbf{x}_{\min}$ respectively $f(\mathbf{x}_{\min})$. Otherwise, if cross validation is satisfied, the point $\mathbf{x}_{\max}$ is determined in such a way that the expected improvement reaches a maximum in $\mathbf{x}_{\max}$. If this expected improvement in $\mathbf{x}_{\max}$ is smaller than 1% of $f_{\min}$, then EGO also terminates and returns $\mathbf{x}_{\min}$ respectively $f(\mathbf{x}_{\min})$, otherwise $\mathbf{x}_{\max}$ is added to S, $f(\mathbf{x}_{\max})$ to T, and $\mathbf{x}_{\min}$ respectively $f(\mathbf{x}_{\min})$.

For a better understanding of the algorithm a visualization of the process is made up in the flow-chart below,

[3] See Appendix A.2 for more information on Latin Hypercube Sampling.

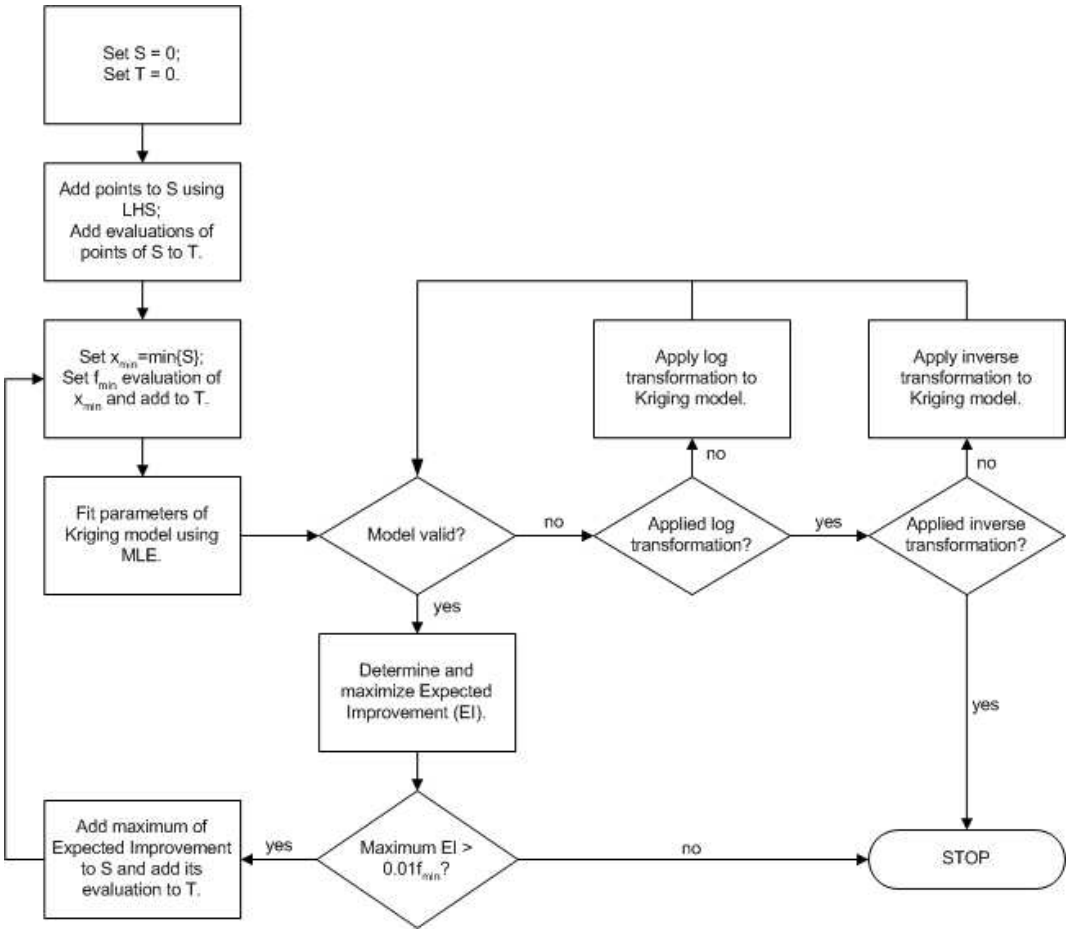

Figure 3.2: Flow-chart of the EGO algorithm

3.6 Miscellaneous

3.6.1 Introduction

This section contains some important items concerning the Kriging model construction and the EGO algorithm. In the first place we discuss for computational reasons the complexity of Kriging. Next section is about the optimization of the expected improvement function. Finally, we discuss an alternative for the expected improvement function, the generalized expected improvement function.

3.6.2 Complexity Kriging

In theory, the construction of a Kriging model is rather expensive for a large number of parameters. The computational difficulties lie in the construction of the inverse of the correlation matrix R. The Matlab toolbox DACE[4] allows us to construct Kriging models for different experimental designs. Below we test if these difficulties could also occur in DACE.

Define the following function,

[4]See [12] for a tutorial on DACE.

$$y(\mathbf{x}) = \sum_{j=1}^{n} x_j + \sum_{j=1}^{n} \sin 2x_j$$

where $0 \leq x_j < 10$ for all j. Let $\mathcal{S}$ be an experimental design consisting of N design sites and furthermore let $\mathbf{y} = (y(\mathbf{x}^{(1)})\ y(\mathbf{x}^{(2)})\ \cdots\ y(\mathbf{x}^{(N)}))$. Assume $\theta_i = 10$ for all i and assume $p_j = 2$ for all j. One can construct a Kriging model for y,

$$Y(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\beta + \epsilon(\mathbf{x}) \tag{3.12}$$

where $f(\mathbf{x}) = (1\ x_1\ x_2 \cdots x_n)$. With this definition of f it is easily seen that we need at least $n+1$ design sites to make the construction of (3.12) possible, so $N \geq n+1$. It is desirable to minimize the number of responses needed, so we take $N = n+1$ from now on.

For different values of n we have constructed a Kriging model with the DACE toolbox and computed the runtime of the implemented algorithm in Matlab and also the runtime of several functions that are called inside the algorithm. For the construction of $\mathcal{S}$ we use Latin Hypercube Sampling, see Appendix A.2 for a detailed description. The table below shows for eight different values of n the runtime for the implemented algorithm, for the functions *Objfunc* and *Corr* and the runtime of the remaining code:

n	runtime algorithm	runtime *Objfunc*	runtime *Corr*	runtime remaining code
200	0.67	0.22	0.17	0.28
210	0.79	0.29	0.24	0.26
220	0.88	0.30	0.24	0.34
230	0.99	0.34	0.27	0.38
240	1.13	0.38	0.31	0.44
250	1.25	0.43	0.35	0.47
350	3.86	1.50	1.32	1.04
500	10.59	3.57	3.14	3.88

The plot below demonstrates a non-linear relation between the dimension n and the different runtimes,

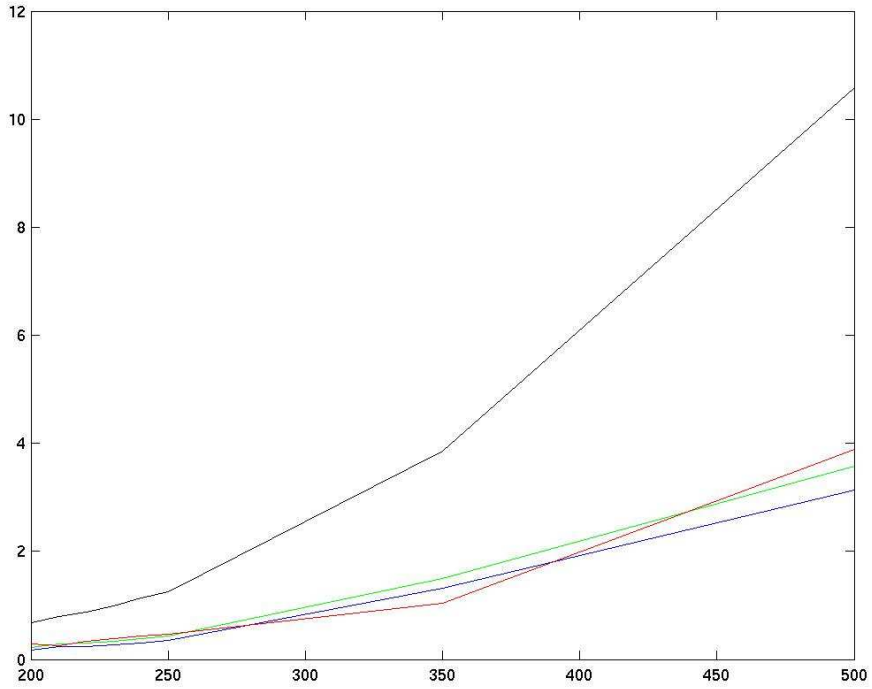

Figure 3.3: Relation between dimension of function and runtime of DACE

It is hard to say if the relation between n and the runtimes is still polynomial. But even if it turns out to be exponential, we do not expect computational difficulties, since even if $n = 500$ the runtime of DACE is small.

3.6.3 Optimization expected improvement function

This subsection is about the optimization of the expected improvement function. We study the behavior of the expected improvement function to determine which optimization algorithm we should use. We start with an example.

Consider an expected improvement function that is related to a Kriging model of the function $y(x) = \sin(\frac{x^2}{4})$ on the interval $[0, 2\pi]$ with set of design sites $\{0, 2, 2\pi\}$. See the figure below,

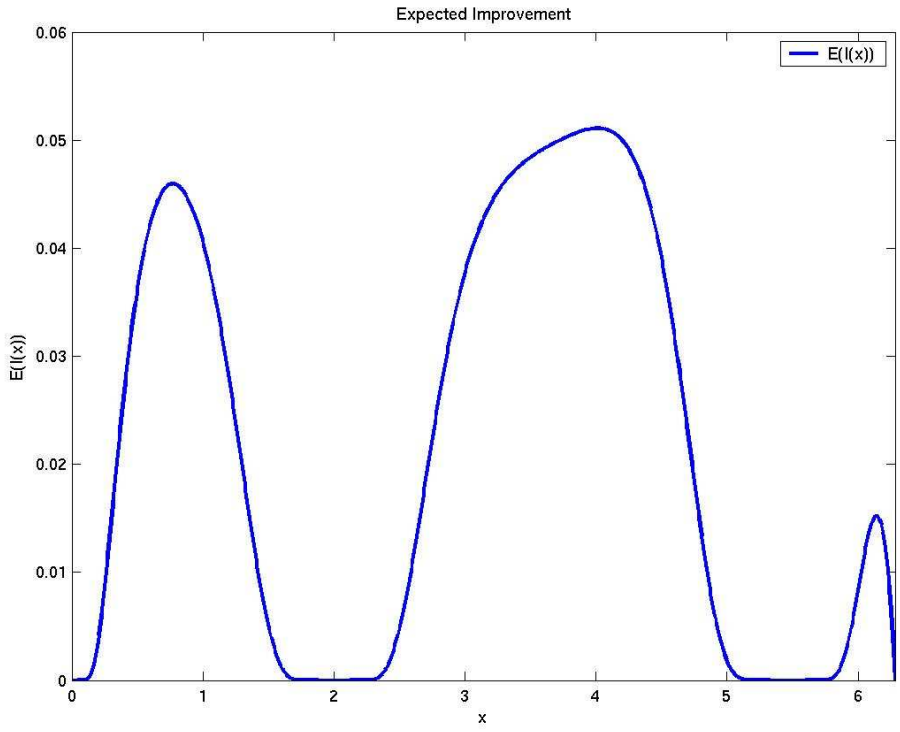

Figure 3.4: Expected improvement function related to the Kriging model of the function $y(x) = \sin(\frac{x^2}{4})$ at $[0, 2\pi]$

We see that the function contains many local maxima, around $x = 0.8$ and $x = 6.2$. We have observed, we will not show here, that for n-dimensional cases the number of local maxima will grow exponentially, which makes it even harder to optimize. However, it is not that expensive evaluating expected improvement functions. So, we need an algorithm that is applicable to multi-modal functions (functions that have more than one minimum) and where the number of evaluations that are needed is acceptable. An optimization algorithm that can handle these properties well is DIRECT[5].

We demonstrate DIRECT with the Rastrigin function,

$$f(x_1, \ldots, x_n) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$

where $n \in \mathbb{N}$. The figure below shows the Rastrigin function for $n = 2$,

[5]See [6] for a tutorial on DIRECT.

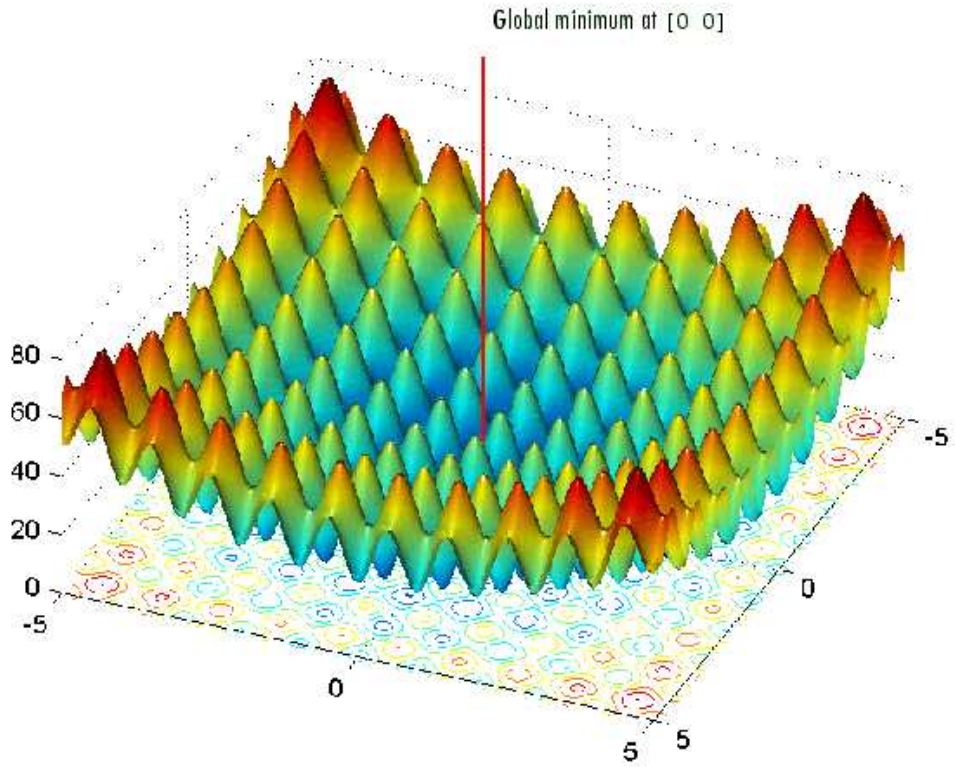

Figure 3.5: A plot of the Rastrigin function for $n = 2$ in the interval $-5 \leq x_1 \leq 5$, $-5 \leq x_2 \leq 5$

By the figure we see that this function has many local minima. We also see that the functions has a global minimum at $(0, 0)$. Applying DIRECT to the Rastrigin function, allowing a maximum of 1000 evaluations respectively iterations, results into the table below for different values of n,

n	number of iterations	number of evaluations	value of the minimum
1	25	1013	0
2	27	1093	1.33937e-12
3	23	1035	1.86342e-06
4	19	1021	0.00277905
5	22	1105	0.00419968

By the table we see that for $n = 1, \ldots, 5$ the global minimum is attained within acceptable values for the number of iterations respectively evaluations.

3.6.4 Generalized Expected Improvement

The EGO algorithm works well especially when the deterministic computer response $y(\mathbf{x})$ is well approximated by the Kriging model. Given the correlation parameters $\mathbf{r}$ and $\mathbf{R}$, the expected improvement criterium optimally chooses where to sample one point according to an average case analysis. However, when $\mathbf{r}$ and $\mathbf{R}$ are poorly estimated, this average case analysis is not sensible, and typically

the search is too local. Therefore we introduce a version of the expected improvement that searches more globally. This goal is achieved by generalizing the expected improvement function with ad additional parameter g, where the larger the value g takes the more globally will the algorithm tend to search.

Suppose the function is sampled at $\mathbf{x}$ to determine $y(\mathbf{x})$. The generalized improvement $I^g(\mathbf{x})$ over $f_{\min}$ is defined as

$$I^g(\mathbf{x}) = \left\{ \begin{array}{cc} (f_{\min} - y)^g & \text{if } y(\mathbf{x}) < f_{\min} \\ 0 & \text{otherwise} \end{array} \right\} = \max\{0, (f_{\min} - y)^g\}$$

where $g \in \mathbb{N}$. For $g = 0$ we obtain the probability of improvement,

$$\begin{array}{rcl} \mathbb{E}(I^0(\mathbf{x})) & = & \mathbb{P}(z < \frac{f_{\min} - \hat{y}}{s(\mathbf{x})}) \\ & = & \Phi(\frac{f_{\min} - \hat{y}(x)}{s(\mathbf{x})}) \end{array}$$

and for $g > 0$ it is possible to show (see [16]) that $\mathbb{E}(I^g(\mathbf{x}))$ is

$$E(I^g(\mathbf{x})) = s^g \sum_{k=0}^{g} (-1)^k \binom{g}{k} \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right)^{g-k} T_k(\mathbf{x}) \tag{3.13}$$

where $T_k(\mathbf{x})$ is given by (see [16])

$$T_k(\mathbf{x}) = \left\{ \begin{array}{ll} -\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \left(\sum_{j=1}^{(k-1)/2} \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right)^{2(j-1)} \prod_{i=j}^{(k-1)/2} 2i + \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right)^{k-1} \right) & \text{if } k \text{ is odd} \\ \Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \prod_{i=1}^{k/2} (2i-1) - \phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \cdot & \\ \quad \left(\sum_{j=2}^{k/2} \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right)^{2j-3} \prod_{i=j}^{k/2} (2i-1) + \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right)^{k-1} \right) & \text{if } k \text{ is even} \end{array} \right.$$

and furthermore satisfies the following recursive relation (see [16])

$$T_k(\mathbf{x}) = -\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right) \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right)^{k-1} + (k-1) T_{k-2}$$

with starting points $T_0(\mathbf{x}) = \Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)$ and $T_1\mathbf{x} = -\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)$. For example, for $g = 1, 2$ we obtain from (3.13)

$$\begin{array}{rcl} \mathbb{E}(I(\mathbf{x})) & = & (f_{\min} - \hat{y}(\mathbf{x})) \Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + \phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \\ \mathbb{E}(I^2(\mathbf{x})) & = & s^2(\mathbf{x}) \left[\left(\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})} \right)^2 + 1 \right) \Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \right] \end{array}$$

Of course, $g = 1$ reproduces the expected improvement derived in (3.11).

In practice the question arises which value of g to choose. We see that there is a tradeoff in choosing between small improvement with large probability, local search, versus large improvement with small probability (global search). In practice, as g increases, larger improvements receive more weight and the search is more global. A lot of research has been performed on this question, nevertheless without any good results that provides a good criterium.

Chapter 4

Enhancements

4.1 Introduction

This chapter pays attention to techniques and algorithms that could improve the efficiency of Gridmom. There are several steps performed in the Gridmom algorithm where alternative techniques/algorithms improve the convergence speed of the algorithm and the quality of the solutions. Also Gridmom encounters problems regarding special types of functions, for example functions where the gradient does not exist around the global minimum, which could be avoided by using these alternatives. So far, there are six alternatives. Two of them, efficient optimization in starting phase and an algorithm for solving gradient problems around the minimum of the objective function have been implemented and extensively tested, these two are described first. Next, the other alternatives are discussed.

4.2 Efficient optimization in starting phase

4.2.1 Introduction

In gridmom the starting phase covers the building of the initial response surface model. General sampling techniques like Hooke-Jeeves and Uniform Design are used. Since global search in the starting phase has a big influence on the speed and quality of local search in the descent and refinement-check phase, it is important that the global search be performed accurately. Mutual independency of the sampling strategy and the objective function is excluded. So, we need to find alternatives for Hooke-Jeeves and Uniform Design. In the previous section we described Kriging modelling and an algorithm using this modelling technique, EGO. We could use this algorithm in the starting phase.

The current termination criterium for EGO is that the maximum of the expected improvement is smaller than one percent of the minimum. For testing the EGO algorithm we consider if this termination criterium is suitable, and otherwise alternatives as well. In next subsections we discuss one criterium that seems appropriate, and one criterium where the quality must be determined by a good configuration of the parameters.

4.2.2 Appropriate termination criterium

Let $\mathbb{E}[I(\mathbf{x})]$ be the expected improvement in the EGO algorithm and let $f_{\min}$ be the best value for the minimum that have been obtained so far. Then EGO terminates if the maximum of $\mathbb{E}[I(\mathbf{x})]$ is smaller than one percent of $f_{\min}$, that is

$$\max_{\mathbf{x}}\{\mathbb{E}[I(\mathbf{x})]\} < 0.01 f_{\min}$$

A drawback of this termination criterium is, however, that the convergence speed of EGO is completely depending on the behavior of the objective function. Since we do not have an expression for the objective function, it is hard to say under which conditions the termination criterium works well.

Not let us have a look at the behavior of EGO during the running process of the algorithm. We consider the following function,

$$f(x) = \min\{10 - \sin x - e^{0.02x}, 9 + 0.25\cos x\}$$

on the interval $[0, 10]$. We use the EGO algorithm for minimizing this function on the given interval, where the where the selected initial points used for the construction of the first Kriging model are equal to the interval bounds including a random chosen point on $[0, 10]$, say $x = 9.5$. After ten rounds in EGO we obtain the following situation

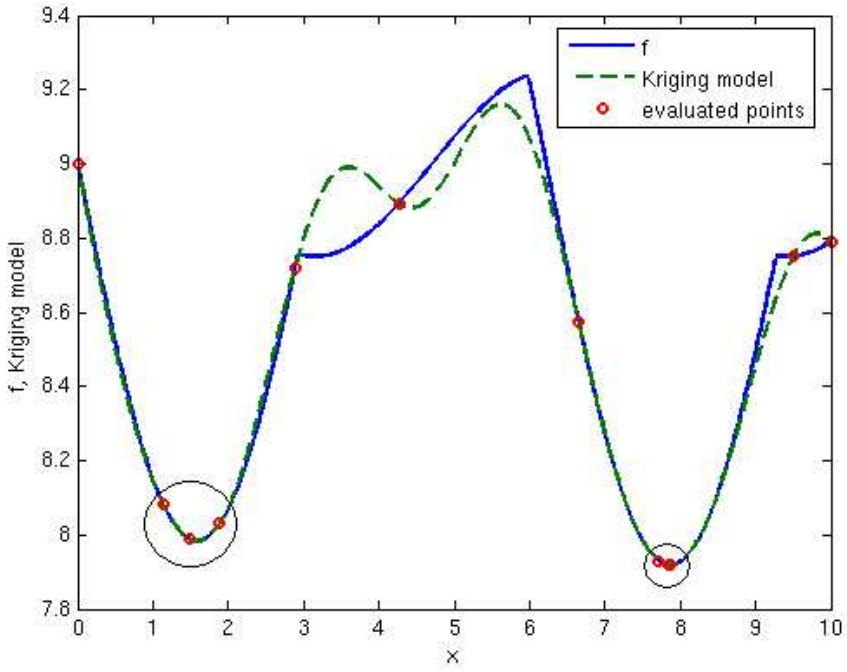

Figure 4.1: Optimization of f in EGO after ten rounds

We see immediately clusters emerging around $[1, 2]$ and $[7.5, 8]$. Continuing the algorithm provides us new points that have a certain probability to appear in these clusters. If we use a grid these new points are rounded to their closest grid points. Regarding the definition of a cluster, there is a non-negligible probability that two points in a cluster are rounded to the same grid point. We could use this characteristic as the groundwork of a new termination criterium.

So, another appropriate termination criterium could be letting EGO terminate when the argument of the maximum expected improvement has already been evaluated before.

4.2.3 Alternative termination criterium

We have observed in chapter 3 that the EGO algorithm consists of two parts, first Latin Hypercube Sampling is applied to obtain an initial set of points, next the initial set is extended by adding points that follow from maximizing the expected improvement of the Kriging model interpolating this set.

Define (l, v) to be a design where l is the number of points sampled in part one and v the number of points obtained in part two. For example, regarding the termination criterium of previous subsection, we choose l as small as possible, such that the number of sampled points is large enough to make the construction of a Kriging model possible; the choice of v depends on the optimization problem. But maybe we could say more about the best choice for v. In general, we consider if there is a relation between a design (l, v) and the behavior, or definition, of an optimization problem. Such relation could lead to a new, alternative, termination criterium.

So, an alternative termination criterium could be letting EGO sample l points with Latin Hypercube Sampling and v points in the remaining part of the algorithm, where the choice for l and v is related to the specifications of the optimization problem.

4.3 Non-differentiable minima

4.3.1 Introduction

In Gridmom the descent and refinement-check phase covers the optimization of the initial response surface model built in the starting phase. Quadratic approximative models are constructed and minimized in the descent phase. Consider an optimization problem where the minimum of the objective function is continu but non-differentiable. It has been observed that in this case the convergence speed of the Gridmom algorithm is often low. This is obvious since we should not expect that a differentiable model works well around a non-differentiable minimum. The process speed could be increased when we apply a piecewise linear approximative fit instead of the quadratic fit. For example, if the function is one-dimensional we use two pieces. But how can we decide whether we are dealing with a non-differentiable minimum? Different tests showed us that the Hessian of the quadratic model is large in such situation. In the first subsection we discuss the situation where the dimension of the objective function is one. Then, in the last subsection, we consider maintaining higher-dimension objective functions.

4.3.2 One-dimensional objective functions

Consider the situation where we have an objective function in one variable. It is easy to check if the Hessian of the quadratic approximative model is large since it contains only one element. A possible implementation for one-dimensional objective functions is visualized in the flowchart below,

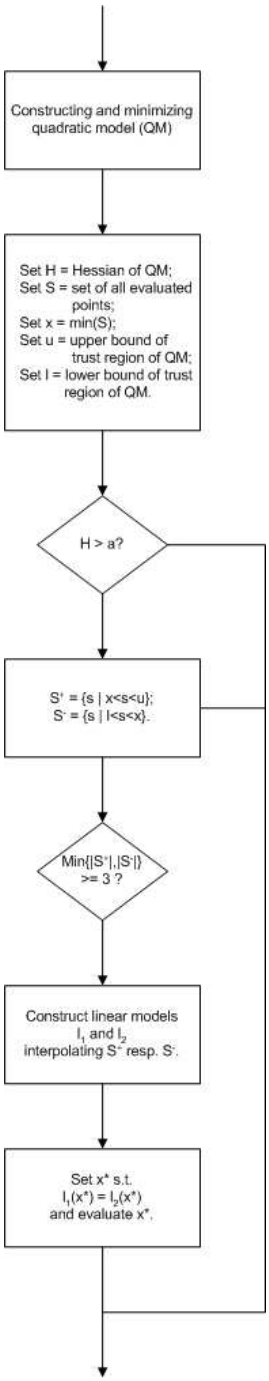

Figure 4.2: Implementation in the descent phase for one-dimensional objective functions with a non-differentiable minimum

So, after a quadratic approximative model has been constructed and minimized, and if the obtained minimum does not satisfy the termination criteria of Gridmom, then we check if the Hessian of the quadratic model is bigger than some fixed $a \in \mathbb{R}$. If this is the case, then let S be the set of all current evaluated points in the Gridmom algorithm, let $x_{\min} = \min\{S\}$, and define two sets S^+ and S^-,

$$\begin{aligned} S^+ &= \{s \in S | x_{\min} < s < u\} \\ S^- &= \{s \in S | l < s < x_{\min}\} \end{aligned}$$

where u and l are respectively the upper and lower bound of the trust region of the quadratic model. If both S^+ and S^- consist of more than some fixed $b \in \mathbb{R}$ elements, then we construct two linear regression models l_1 and l_2 interpolating S^+ respectively S^-. The unique intersection of l_1 and l_2 is the point we are looking for. This point is evaluated and added to S. Finally Gridmom continues the descent phase by checking if the point satisfies the termination criteria of Gridmom.

4.3.3 Higher-dimensional objective functions

In the previous subsection we discussed an implementation scheme for one-dimensional objective functions that have a minimum that is continu but not differentiable. For n-dimensional objective functions it is getting more complicated. Regarding the implementation scheme, we discuss the proposed changes for the Hessian criterium, next those for the linear fitting process. For the last item we also propose an alternative, cheaper, strategy, and afterwards we compare both strategies.

The Hessian criterium does not change that much. In the n-dimensional case the Hessian $H = (h_{ij})$ is a $n \times n$ symmetric matrix, such that $h_{ij} = h_{ji}$ for all $i, j = 1, 2, \ldots, n$. For one-dimensional objective functions the Hessian need to be large, which remains the same for the n-dimensional case. So, all the elements of H must be large.

Now consider the linear fitting process. For one-dimensional objective functions we construct two lines l_1, l_2 by means of linear regression. In the n-dimensional case we use hyperplanes of dimension n. We choose dimension n since we want to maximize the probability that the hyperplanes have an intersection that is non-empty. It may be clear that, for example, two lines have a very small probability to intersect in an $n+1$-dimensional vector space. Consider we have constructed m hyperplanes of dimension n by means of linear regression,

$$l_i(x_1, \ldots, x_n) = a_{i1} + \sum_{j=2}^{n+1} a_{ij} x_{j-1}, \quad i \in \{1, \ldots, m\}$$

The intersection of these hyperplanes can be found by solving the following system of equations,

$$\begin{aligned} l_1 &= l_2 \\ l_1 &= l_3 \\ &\vdots \quad \vdots \\ l_1 &= l_m \\ &\vdots \quad \vdots \\ l_m &= l_{m-2} \\ l_m &= l_{m-1} \end{aligned} \tag{4.1}$$

We simplify the system of equations in (4.1) to a system of m equations,

$$\begin{aligned} l_1 &= l_2 \\ l_2 &= l_3 \\ &\vdots \quad \vdots \\ l_{m-1} &= l_m \\ l_m &= l_1 \end{aligned} \tag{4.2}$$

Since we have n unknown variables in (4.2) we need at least n equations to find these variables, so $m \geq n$. We take $m = n$ from now on since we want as few hyperplanes as possible. Let us rewrite (4.2) in terms of a matrix equation, say $\mathbf{Ax} = \mathbf{PAx}$, where $A = (a_{ij})$, $\mathbf{x} = (1\ x_1\ \cdots\ x_n)^T$, and

$$P = \left(\begin{array}{cccc|c} 0 & 0 & \cdots & 0 & 1 \\ \hline 1 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{array}\right)$$

Define $\mathbf{B} = (\mathbf{I} - \mathbf{P})\mathbf{A}$. Then the matrix equation is equivalent to $\mathbf{Bx} = \mathbf{0}$. This matrix equation is easily solved with for example Gaussian elimination or LU-decomposition techniques. The problem is only, that $\mathbf{B}$ does not have always full rank, which means the matrix equation cannot always be solved. For this reason we also discuss an alternative strategy.

Consider again the linear fitting process. Instead of one piecewise linear fitting over n dimensions, we could also apply n times piecewise linear fitting over 1 dimension. Consider the following lines,

$$\begin{aligned} l_{i1} &: \mathbb{R} \mapsto \mathbb{R} \quad l_{i1}(x_i) = a_{i11} + a_{i12}x_i \\ l_{i2} &: \mathbb{R} \mapsto \mathbb{R} \quad l_{i2}(x_i) = a_{i21} + a_{i22}x_i \end{aligned}$$

where $i = 1, \ldots, n$. The intersection of each pair of lines can be written in terms of a system of equations,

$$\begin{array}{ccc} l_{11} & = & l_{12} \\ \vdots & & \vdots \\ l_{n1} & = & l_{n2} \end{array} \tag{4.3}$$

Let us rewrite (4.3) in terms of a matrix equation, say $\mathbf{A_1x} = \mathbf{A_2x}$, where

$$\mathbf{A_i} = \begin{pmatrix} a_{1i1} & a_{1i2} & 0 & \cdots & 0 \\ a_{2i1} & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ a_{ni1} & 0 & \ddots & 0 & a_{ni2} \end{pmatrix}$$

for $i = 1, 2$, and $\mathbf{x} = (1 x_1 \cdots x_n)^T$. Let us define $\mathbf{B} = \mathbf{A_1} - \mathbf{A_2}$. Then the matrix equation is equivalent to $\mathbf{Bx} = \mathbf{0}$. In $\mathbf{B}$, the elements $a_{i12} - a_{i22}$ are nonzero for all $i = 1, \ldots, n$. This property follows directly from the Hessian criterium. It may be clear that since this property holds, $\mathbf{B}$ has always full rank and so the matrix equation can be solved.

Comparing the two linear fitting strategies we see a lot of differences. For each of the n hyperplanes we need at least n points, so we need at least n^2 points for the whole fitting process. For the linear models, however, we need at least 2 points for one model, and since we have $2n$ models this means we need at least $4n$ points. Furthermore, if we take exactly n^2 points, we have to make a division of n groups containing n points each, and this division has to be performed in such way that the probability that the n hyperplanes intersect in one point is maximal. For the linear models we do not need to make any division.

Finally, both strategies seems to be useful. The first strategy is expensive, both the fitting process and the division of the points, but the results are expected to be accurate. The second strategy is

cheaper, but the results are expected to be less accurate, since this strategy expects to objective function to be simple. Accurate results are expected if the objective function is build up componentwise, which will almost never be the case.

4.4 Alternatives

Besides the two alternatives discussed before, there are also four ideas that have not been tested yet, but which are surely worth to consider. These ideas are described in the next subsections.

4.4.1 Kriging modelling in descent phase

We have seen in chapter 2 that in the descent phase a quadratic approximative model is constructed and minimized inside the trust region. Instead of constructing a quadratic model with the set of points that lie in the trust region, we could also fit a Kriging model to these points and minimize the model with DIRECT.

4.4.2 Efficient optimization in refinement-check phase

We have seen in chapter 2 that in the refinement-check phase it is checked if the current value of the minimum could be improved. This is achieved by the construction and minimization of a linear model. If needed, extra points are evaluated and added to the linear model. Instead of the linear model, we could also use the EGO algorithm.

4.4.3 Degeneration in descent phase

It could occur that the points that are found and evaluated in the descent phase are heading in a certain direction, we can say that these points lie in some hyperplane and we could expect that new points will also lie in this hyperplane, or at least very close. This event is also called degeneration, see the figure below for the 2-dimensional case,

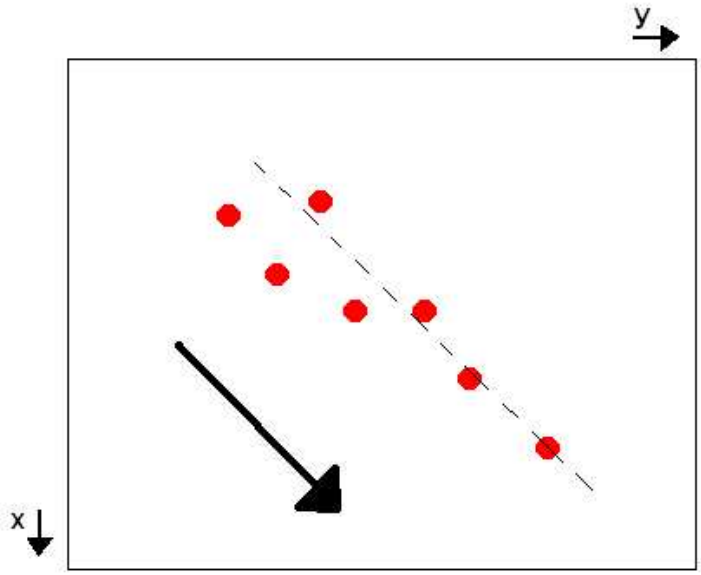

Figure 4.3: Degeneration in the descent phase

It is desired not to look in one certain direction but to have a larger view. A solution could be in case degeneration is detected to use Uniform Design to sample some extra points around the current best point.

Chapter 5

Test Problems

5.1 Introduction

In this section we discuss several tests[1] that we have performed with respect to the enhancements in chapter 4. The next sections are about tests with respect to the use of the EGO algorithm in the starting phase and treating non-differentiable minima for one-dimensional objective functions.

5.2 EGO in starting phase: appropriate criterium

We have tested several functions[2]. Since a majority of the functions gave nearly equivalent results, we only discuss a selection of these functions.

5.2.1 Rastrigin function

Consider the following function,

$$f(x_1, \ldots, x_n) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$

for $n = 2$, see figure 3.6.3. We have seen before that this function contains a lot of local minima, and therefore is hard to minimize in Gridmom. We compare the number of samples needed in the old Gridmom code to the number of samples EGO needs before it terminates. The function is minimized on $[-7.5, 2.5]$. The reason we do not prefer minimizing on $[-5, 5]$ is that the function is symmetric in the origin and the results of DIRECT could be influenced by this symmetric aspect. We have performed tests for 20 different initial points determined by latin hypercube sampling and we compare the mean numbers of evaluations, $\mu_{\text{old,eval}}$ and $\mu_{\text{new,eval}}$, and the mean numbers of the objective function value, $\mu_{\text{old,fval}}$ and $\mu_{\text{new,fval}}$. We obtain the following results,

[1] To perform these tests, some changes have been made in the source code of Gridmom. For further usage the changes are documented in the updated source code.

[2] The functions we are using have been used before to perform tests in Gridmom.

dimension	$\mu_{\text{old,eval}}$	$\mu_{\text{old,fval}}$	$\mu_{\text{new,eval}}$	$\mu_{\text{new,fval}}$
$n = 2$	34.6	6.71593	54.75	3.38286
$n = 3$	51	8.10888	65.75	7.06419
$n = 4$	73.9	22.4859	122	5.77075
$n = 5$	103.35	27.411	114.45	18.5062

The table shows us that for different dimensions of the Rastrigin function we get a better value for the minimum evaluation in the new code, however more evaluations are needed to reach this minimum.

5.2.2 Vardim function

Consider the following function,

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} (x_i - 1)^2 + \sum_{i=1}^{n} i\,(x_i - 1)^2 + \sum_{i=1}^{n} i\,(x_i - 1)^4$$

In the figure below we see the Vardim function for $n = 2$ in the interval $-10 \leq x_1, x_2 \leq 10$,

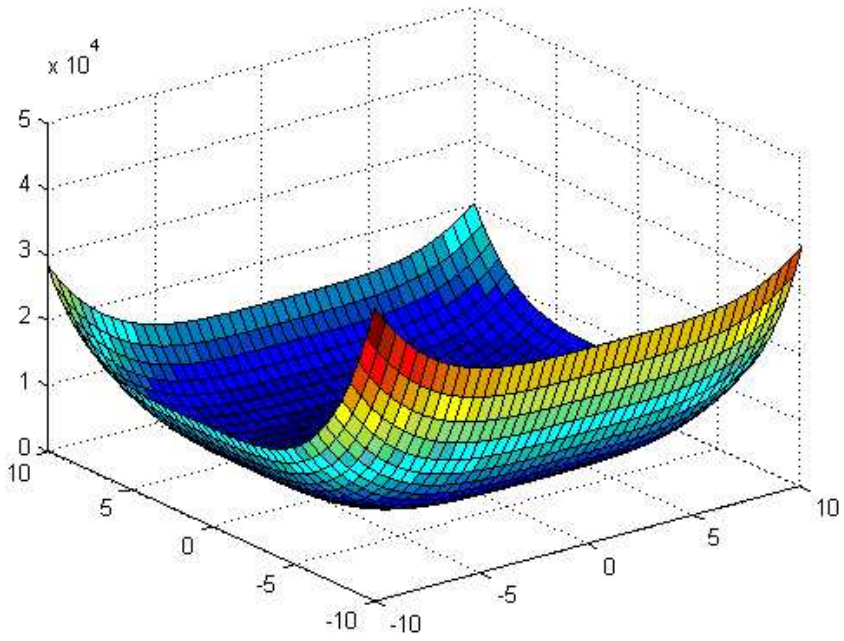

Figure 5.1: A plot of the Vardim function in the interval $-10 \leq x_1, x_2 \leq 10$

Since the gradient in a relatively large area around the minimum is nearly zero Gridmom tends to return a random point in this area as the minimum of the objective function. The table below shows us that for the Vardim function the number of evaluations needed is increasing and that the value of the minimum remains nearly the same,

dimension	$\mu_{\text{old,eval}}$	$\mu_{\text{old,fval}}$	$\mu_{\text{new,eval}}$	$\mu_{\text{new,fval}}$
$n = 2$	30.7	4.03956e-09	39.6	4.03956e-09
$n = 3$	49.7	1.52671e-08	57.95	1.5266e-08
$n = 4$	81.6	3.67637e-08	79.45	3.16942e-08
$n = 5$	118.25	3.87223e-08	66.7	4.62058e-08

5.2.3 Bumpyfun function

Finally, consider the function below,

$$f(x_1, x_2) = 5 + (x_1 + x_2)\sin(-8x_1) - x_1x_2 + x_2 - x_1^2 - 10\cos(1.8x_2)$$

In the figure below we see the bumpyfun function in the interval $-3 \leq x_1 \leq 2,\ 2 \leq x_2 \leq 5$,

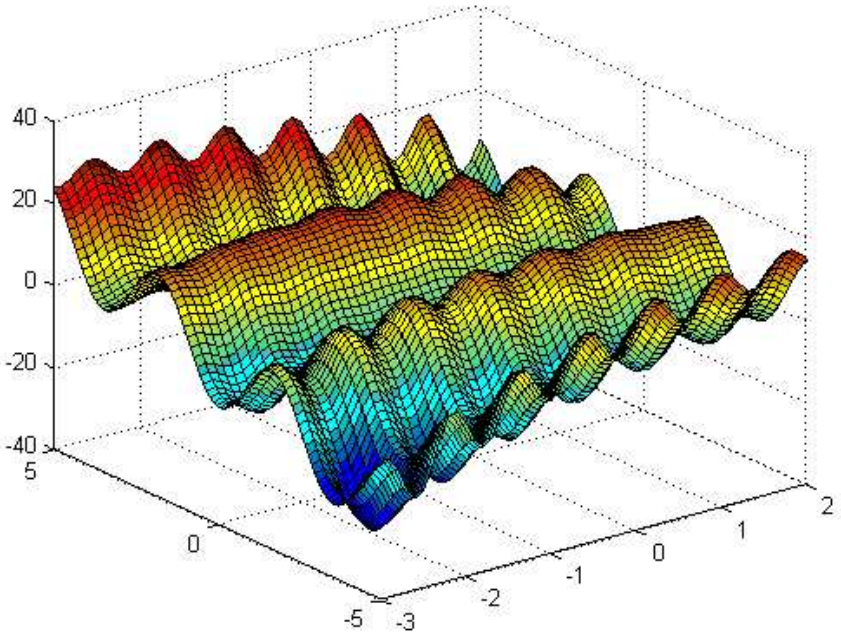

Figure 5.2: A plot of the Bumpyfun function in the interval $-3 \leq x_1 \leq 2, 2 \leq x_2 \leq 5$

We see that the behavior of the function is very random (it is hard to see a trend), which makes it hard for Gridmom to find the minimum. The table below shows us that for the Bumpyfun function we get better results for the value of the minimum, but at cost of the number of evaluations,

$\mu_{\text{old,eval}}$	$\mu_{\text{old,fval}}$	$\mu_{\text{new,eval}}$	$\mu_{\text{new,fval}}$
32.55	-21.6262	35.4	-30.049

5.3 EGO in starting phase: alternative criterium

Like in previous section, we have tested the Rastrigin, Vardim and Bumpyfun functions for different configurations. Also we introduce and test the Worstcase-corner function. Let l be the number of evaluations determined by Latin Hypercube Sampling. The bounds are also evaluated and included in this number. Let v be the number of evaluations performed by EGO.

5.3.1 Rastrigin function

Consider the Rastrigin function for $n = 3$ in the interval $-5 \leq x_1, x_2, x_3 \leq 5$. In the original Gridmom algorithm six points are sampled by Uniform Design techniques by default in the starting phase. In the first place we determine the mean number of evaluations $\mu_{\text{old,eval}}$ and the mean minimum value for the objective function $\mu_{\text{old,fval}}$ for the original Gridmom algorithm by doing tests for 20 different initial points determined by latin hypercube sampling. We obtain the following results,

$\mu_{\text{old,eval}}$	$\mu_{\text{old,fval}}$
36.4	32.2364

The following table is about the number of evaluations the new Gridmom algorithm needs for optimizing for different settings of l and v,

	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$	$l=11$	$l=12$
$v=2$	43	37	37	53	47	59	46	45	46
$v=3$	40	47	39	47	46	50	51	46	47
$v=4$	40	43	53	51	46	51	46	47	48
$v=5$	42	55	50	51	48	50	50	48	49
$v=6$	50	48	45	50	50	51	47	50	50
$v=7$	52	42	55	51	51	52	48	51	51
$v=8$	48	43	56	53	57	52	49	52	51
$v=9$	51	44	47	54	50	53	50	53	54
$v=10$	54	51	48	56	54	56	51	54	55
$v=11$	54	52	52	55	52	57	54	55	57
$v=12$	56	53	53	57	59	60	55	56	58
$v=13$	56	54	54	59	57	57	56	57	56

and the table below is about the value of the minimum, in 4 decimals, of the new Gridmom algorithm returns for different settings of l and v.

	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$	$l=11$	$l=12$
$v=2$	8.954	33.828	33.828	0.994	9.949	4.974	16.914	16.914	16.914
$v=3$	33.828	10.944	33.828	10.944	17.909	10.944	16.914	16.914	16.914
$v=4$	33.828	16.914	9.949	10.944	13.929	13.929	16.914	16.914	16.914
$v=5$	26.863	4.974	4.974	10.944	13.929	13.929	16.914	16.914	16.914
$v=6$	3.979	13.929	8.954	10.944	13.929	13.929	16.914	16.914	16.914
$v=7$	0.994	20.894	16.914	10.944	13.929	13.929	16.914	16.914	16.914
$v=8$	0.0	20.894	16.914	10.944	10.944	8.954	16.914	16.914	8.954
$v=9$	0.994	20.894	7.959	10.944	18.904	11.939	16.914	16.914	8.954
$v=10$	0.994	20.894	25.868	10.944	11.939	21.889	16.914	16.914	8.954
$v=11$	0.994	20.894	25.868	10.944	11.939	21.889	16.914	16.914	8.954
$v=12$	2.984	20.894	25.868	10.944	8.954	3.979	16.914	16.914	8.954
$v=13$	0.994	20.894	25.868	10.944	11.939	3.979	16.914	16.914	8.954

With the previous tables we construct a new table. The table below contains the mean number of evaluations $\mu_{\text{new,eval},p}$, and the corresponding standard deviation $\sigma_{\text{new,eval},p}$, performed in the starting phase when the number of points sampled in the starting phase is equal to p. We choose here $p = l+v$, where l and v attain the same values as in previous tables. Note that only the observations in table 2 are used for computing these means. The number of observations $n_{\text{evals},p}$, all combinations of l and v such that $p = l + v$, is also contained in this table.

Also consider a measurement IMPmin, that measures the percentage of improvements of the minimum compared to the minima obtained with the original Gridmom algorithm, where we speak about an improvement if the concerned minimum is better than the mean of the minima obtained with the original Gridmom algorithm. Equivalently, consider a measurement IMPeval, that is equal to IMPmin, except for the fact that the number of evaluations is observed in stead of the value of the minimum.

In mathematical terminology, these measurements are defined as follows. Let $\text{min}_{l,v}$ and $\text{eval}_{l,v}$ be the minimum value respectively the number of evaluations needed in Gridmom for fixed l and v, see table 1 and 2.

Define IMPmin_p and IMPeval_p,

$$\text{IMPmin}_p = \frac{\sum_{p=l+v} 1_{\text{min}_{l,v} < \mu_{\text{old,min}}}}{n_{\text{evals},p}}$$

$$\text{IMPeval}_p = \frac{\sum_{p=l+v} 1_{\text{eval}_{l,v} < \mu_{\text{old,eval}}}}{n_{\text{evals},p}}$$

where 1 is an indicator function. Thus these measurements have a certainty of 97.5%. The table below also contains values of IMPmin_p, IMPeval_p, and the mean of these two values, the total improvement IMPtotal_p,

$p = l + v$	$n_{\text{evals},p}$	$\mu_{\text{new,eval},p}$	$\sigma_{\text{new,eval},p}$	IMPmin$_p$	IMPeval$_p$	IMPtotal$_p$
$p = 6$	1	43	0	1	0	0.5
$p = 7$	2	38.5	2.12132	0	0	0
$p = 8$	3	41.3333	5.1316	0.333333	0	0.166667
$p = 9$	4	44.25	6.07591	0.75	0	0.375
$p = 10$	5	50.4	3.57771	1	0	0.5
$p = 11$	6	51	4.47214	1	0	0.5
$p = 12$	7	46.8571	3.07834	1	0	0.5
$p = 13$	8	49.25	3.80789	1	0	0.5
$p = 14$	9	49.2222	4.05518	1	0	0.5
$p = 15$	9	50.1111	2.61937	1	0	0.5
$p = 16$	9	51.3333	3.77492	1	0	0.5
$p = 17$	9	51.7778	2.86259	1	0	0.5
$p = 18$	8	52.375	2.13391	1	0	0.5
$p = 19$	7	53.1429	2.60951	1	0	0.5
$p = 20$	6	55	3.79473	1	0	0.5
$p = 21$	5	55.8	2.68328	1	0	0.5
$p = 22$	4	55.5	1	1	0	0.5
$p = 23$	3	56.3333	0.57735	1	0	0.5
$p = 24$	2	57.5	0.707107	1	0	0.5
$p = 25$	1	56	0	1	0	0.5

(5.1)

From the table we see that for a higher number of points sampled in the starting phase by EGO we have a higher number of total evaluations needed in the Gridmom algorithm. Furthermore, we see that the value of the minimum is improving, for almost every configuration. Too bad this is at the cost of the total number of evaluations performed in Gridmom. The best configuration we should use is choosing l and v such that $p = 7$.

5.3.2 Vardim function

Now consider the Vardim function for $n = 3$. We determine the mean number of evaluations $\mu_{\text{old,eval}}$ and the mean minimum value for the objective function $\mu_{\text{old,fval}}$ for the original Gridmom algorithm by doing tests for 20 different initial points determined by latin hypercube sampling. We obtain the following results,

$\mu_{\text{old,eval}}$	$\mu_{\text{old,min}}$
69.25	6.14555e-08

The following table is about the number of evaluations Gridmom needs for optimizing for different settings of l and v,

	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$	$l=11$	$l=12$
$v=2$	51	50	57	49	46	50	48	63	57
$v=3$	47	53	54	53	47	47	54	64	58
$v=4$	66	60	59	49	48	51	55	52	58
$v=5$	65	62	62	47	55	50	69	63	63
$v=6$	63	66	66	68	60	51	58	50	64
$v=7$	64	70	58	68	61	55	57	59	53
$v=8$	58	67	63	52	57	56	58	60	56
$v=9$	57	67	69	67	58	58	67	66	58
$v=10$	61	70	61	57	57	65	62	57	59
$v=11$	62	57	67	74	61	62	63	58	60
$v=12$	67	77	65	78	62	57	64	66	61
$v=13$	56	66	65	67	67	58	75	65	62

The minima for all the configurations in the table above are nearly zero (zero in 4 decimals). At last, the table below contains the same information as table 5.1 for the Vardim function,

$p=l+v$	$n_{\text{evals},p}$	$\mu_{\text{new,eval},p}$	$\sigma_{\text{new,eval},p}$	IMPmin_p	IMPeval_p	IMPtotal_p
$p=6$	1	51	0	1	1	1
$p=7$	2	48.5	2.12132	1	1	1
$p=8$	3	58.6667	6.65833	1	1	1
$p=9$	4	57	6.97615	1	1	1
$p=10$	5	56.6	7.09225	1	1	1
$p=11$	6	56.3333	8.5479	1	1	1
$p=12$	7	54.8571	9.83918	1	0.857143	0.928571
$p=13$	8	59.125	6.22065	1	1	1
$p=14$	9	60.5556	5.81187	1	1	1
$p=15$	9	60.4444	7.73161	1	0.888889	0.944444
$p=16$	9	60.3333	4.4441	1	1	1
$p=17$	9	60.1111	7.91272	1	0.888889	0.944444
$p=18$	8	62.625	5.80486	1	0.875	0.9375
$p=19$	7	64.1429	7.66874	1	0.857143	0.928571
$p=20$	6	62.5	3.88587	1	1	1
$p=21$	5	60.4	4.44972	1	1	1
$p=22$	4	59.75	2.87228	1	1	1
$p=23$	3	67	7.54983	1	0.666667	0.833333
$p=24$	2	63	2.82843	1	1	1
$p=25$	1	62	0	1	1	1

From the table we see that for a higher number of points sampled in the starting phase by EGO we have a higher number of total evaluations needed in the Gridmom algorithm. Furthermore, we see that the value of the minimum is improving in many configurations, as well as the total number of evaluations performed in Gridmom. The best configuration here would be choosing l and v such that $p=8$, since the least evaluations are needed there to compute the minimum.

5.3.3 Bumpyfun function

Let us have a look at the Bumpyfun function. We determine the mean number of evaluations $\mu_{\text{old,eval}}$ and the mean minimum value for the objective function $\mu_{\text{old,fval}}$ for the original Gridmom algorithm by doing tests for 20 different initial points determined by latin hypercube sampling. We obtain the following results,

$\mu_{\text{old,eval}}$	$\mu_{\text{old,fval}}$
32.15	-21.8534

The table below is about the number of evaluations Gridmom needs for optimizing for different settings of l and v,

	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$	$l=11$	$l=12$
$v=2$	29	38	33	40	30	33	33	34	35
$v=3$	36	43	35	42	33	34	37	33	34
$v=4$	34	42	36	33	36	37	35	36	36
$v=5$	37	33	44	39	41	34	36	33	35
$v=6$	36	32	45	34	40	35	37	34	36
$v=7$	35	36	39	35	41	36	38	35	37
$v=8$	38	34	38	35	39	37	39	36	38
$v=9$	36	36	39	37	39	38	40	37	39
$v=10$	37	37	39	39	40	39	41	38	40
$v=11$	38	38	40	38	41	40	42	39	41
$v=12$	39	39	41	39	42	41	43	40	42
$v=13$	40	40	42	40	43	42	44	41	43

and the table below is about the value of the minima, in 4 decimals, corresponding to the configurations in the table above,

	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$	$l=11$	$l=12$
$v=2$	-23.767	-31.598	-23.767	-31.598	-23.767	-31.598	-31.598	-31.598	-31.598
$v=3$	-31.598	-24.643	-23.767	-31.598	-23.767	-31.598	-31.598	-31.598	-31.598
$v=4$	-31.598	-31.598	-23.767	-23.767	-31.598	-31.598	-31.598	-31.598	-31.598
$v=5$	-31.598	-23.767	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=6$	-31.598	-23.767	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=7$	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=8$	-31.598	-24.643	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=9$	-31.598	-24.643	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=10$	-31.598	-24.643	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=11$	-31.598	-24.643	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=12$	-31.598	-24.643	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598
$v=13$	-31.598	-24.643	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598	-31.598

Then, the table below contains the same information as table 5.1 applied to the Bumpyfun function,

$p = l + v$	$n_{\text{evals},p}$	$\mu_{\text{new,eval},p}$	$\sigma_{\text{new,eval},p}$	IMPmin_p	IMPeval_p	IMPtotal_p
$p = 6$	1	29	0	1	1	1
$p = 7$	2	37	1.41421	1	0	0.5
$p = 8$	3	36.6667	5.50757	1	0	0.5
$p = 9$	4	38.5	3.10913	1	0	0.5
$p = 10$	5	35.4	4.44972	1	0.2	0.6
$p = 11$	6	35	4.51664	1	0.166667	0.583333
$p = 12$	7	37.2857	3.98808	1	0	0.5
$p = 13$	8	36.5	2.56348	1	0	0.5
$p = 14$	9	35.8889	2.14735	1	0	0.5
$p = 15$	9	36.7778	2.22361	1	0	0.5
$p = 16$	9	37.1111	1.96497	1	0	0.5
$p = 17$	9	37.8889	2.14735	1	0	0.5
$p = 18$	8	38.375	2.06588	1	0	0.5
$p = 19$	7	39.1429	2.1157	1	0	0.5
$p = 20$	6	39.6667	1.8619	1	0	0.5
$p = 21$	5	40.6	2.07364	1	0	0.5
$p = 22$	4	41	1.82574	1	0	0.5
$p = 23$	3	41.6667	2.08167	1	0	0.5
$p = 24$	2	41.5	0.707107	1	0	0.5
$p = 25$	1	43	0	1	0	0.5

From the table we see that for a higher number of points sampled in the starting phase by EGO we have a higher number of total evaluations needed in the Gridmom algorithm. Furthermore, we see that the value of the minimum is improving in all tested configurations, however the total number of evaluations performed in Gridmom is increasing. There is only one configuration where the value of the minimum is lower and the total number of evaluations is less, namely choosing l and v such that $p = 6$.

5.3.4 Worstcase-corner function

At last, consider the following function,

$$f(x_1, x_2) = \max\{(x_1 - 2)^2 + 2, -0.5(x_1 - 3)^2 + 4.5\}$$

In the figure below we see the Worstcase-corner function in the interval $0 \leq x_1, x_2 \leq 4$,

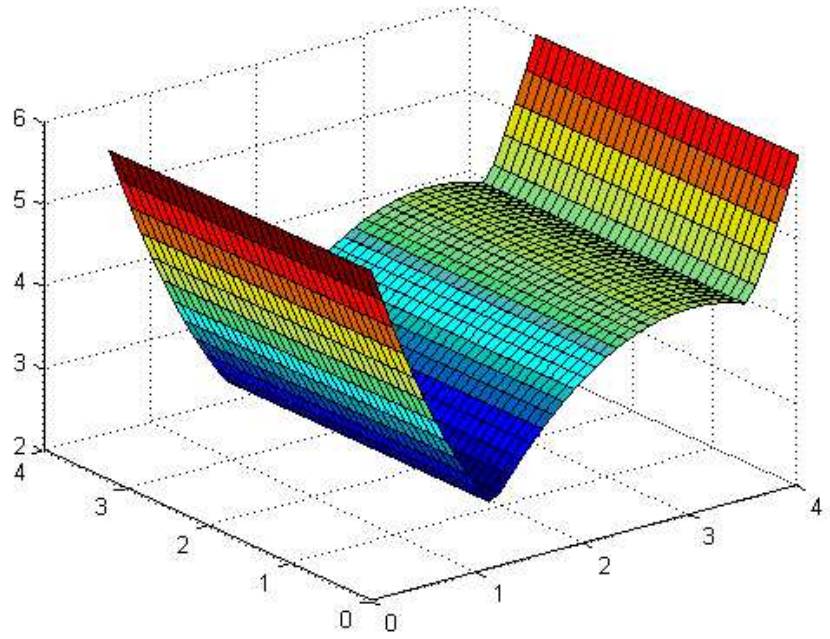

Figure 5.3: A plot of the Worstcase-corner function in the interval $0 \leq x_1, x_2 \leq 4$

We see that this function has two minima for $0 \leq x_1 \leq 4$. In the function prescript we see an extra variable x_2, which is not used in the function formula itself. This dummy variable makes it harder for the Gridmom algorithm to minimize, since for every choice the variable x_2 attains the function value remains the same whereas x_1 is fixed. Also the fact that the gradient do not exist in the minimum makes it harder for Gridmom to minimize. Now let us determine the mean number of evaluations $\mu_{\text{old,eval}}$ and the mean minimum value for the objective function $\mu_{\text{old,fval}}$ for the original Gridmom algorithm by doing tests for 20 different initial points determined by latin hypercube sampling. We obtain the following results,

$\mu_{\text{old,eval}}$	$\mu_{\text{old,fval}}$
47.16	2.75468

The following table is about the number of evaluations Gridmom needs for optimizing for different settings of l and v,

	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$	$l=11$	$l=12$
$v=2$	51	52	55	49	57	50	42	54	55
$v=3$	56	49	56	60	52	54	43	42	43
$v=4$	49	54	56	61	60	50	62	47	48
$v=5$	57	56	60	53	38	58	63	48	46
$v=6$	63	55	66	59	56	59	46	52	47
$v=7$	57	56	67	55	45	60	56	53	48
$v=8$	56	60	68	56	46	55	57	54	49
$v=9$	68	61	68	57	40	60	58	58	50
$v=10$	69	45	53	58	41	56	59	59	51
$v=11$	70	38	57	59	39	57	60	60	52
$v=12$	68	63	56	55	39	58	61	58	53
$v=13$	69	61	53	59	47	59	62	59	54

and next table is about the corresponding values of the minimum the Gridmom algorithm returns,

	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$	$l=11$	$l=12$
$v=2$	4.356	4.356	4.356	4.356	4.356	2.754	2.754	2.754	2.754
$v=3$	2.754	4.356	4.356	4.356	4.356	2.754	2.754	2.754	2.754
$v=4$	4.356	4.356	4.356	4.356	4.356	2.754	2.754	2.754	2.754
$v=5$	4.356	4.356	4.356	4.356	2.754	2.754	2.754	2.754	2.754
$v=6$	4.356	4.356	4.356	4.356	2.754	2.754	2.754	2.754	2.754
$v=7$	4.356	4.356	4.356	4.356	2.754	2.754	2.754	2.754	2.754
$v=8$	4.356	4.356	4.356	4.356	2.754	2.754	2.754	2.754	2.754
$v=9$	4.356	4.356	2.754	4.356	2.754	2.754	2.754	2.754	2.754
$v=10$	4.356	2.754	2.754	4.356	2.754	2.754	2.754	2.754	2.754
$v=11$	4.356	2.754	2.754	4.356	2.754	2.754	2.754	2.754	2.754
$v=12$	4.356	2.754	2.754	2.754	2.754	2.754	2.754	2.754	2.754
$v=13$	4.356	2.754	2.754	2.754	2.754	2.754	2.754	2.754	2.754

At last, the table below contains the same information as table 5.1 for the Worstcase-corner function,

$p = l + v$	$n_{\text{evals},p}$	$\mu_{\text{new,eval},p}$	$\sigma_{\text{new,eval},p}$	IMPmin_p	IMPeval_p	IMPtotal_p
$p = 6$	1	51	0	0	1	0.5
$p = 7$	2	54	2.82843	0.5	1	0.75
$p = 8$	3	51	3.4641	0	1	0.5
$p = 9$	4	54	3.55903	0	1	0.5
$p = 10$	5	58.4	3.04959	0	1	0.5
$p = 11$	6	55.8333	4.35507	0.166667	1	0.583333
$p = 12$	7	55.2857	7.31925	0.285714	1	0.642857
$p = 13$	8	54.875	10.7761	0.5	1	0.75
$p = 14$	9	58.4444	8.07947	0.555556	1	0.777778
$p = 15$	9	55.1111	10.5053	0.777778	1	0.888889
$p = 16$	9	51.5556	8.97373	0.777778	1	0.888889
$p = 17$	9	55.1111	8.57969	0.777778	1	0.888889
$p = 18$	8	54.25	6.98468	0.875	1	0.9375
$p = 19$	7	51.8571	6.46603	1	1	1
$p = 20$	6	53.5	8.04363	1	1	1
$p = 21$	5	54.8	5.89067	1	1	1
$p = 22$	4	57.75	4.57347	1	1	1
$p = 23$	3	57.3333	5.03322	1	1	1
$p = 24$	2	56	4.24264	1	1	1
$p = 25$	1	54	0	1	1	1

From the table we see that the value of the minimum has improved, again at the cost of the number of evaluations needed. However, it is hard to say here what is the best configuration. We see that $p = 19, \ldots, p = 25$ are all appropriate candidates.

5.4 Non-differentiable minimum for one-dimensional objective function

The implementation has been tested for the one-dimensional Worstcase-corner function,

$$f(x) = \max\{(x-2)^2 + 2, -0.5(x-3)^2 + 4.5\}$$

on $[0, 6]$, see the figure below,

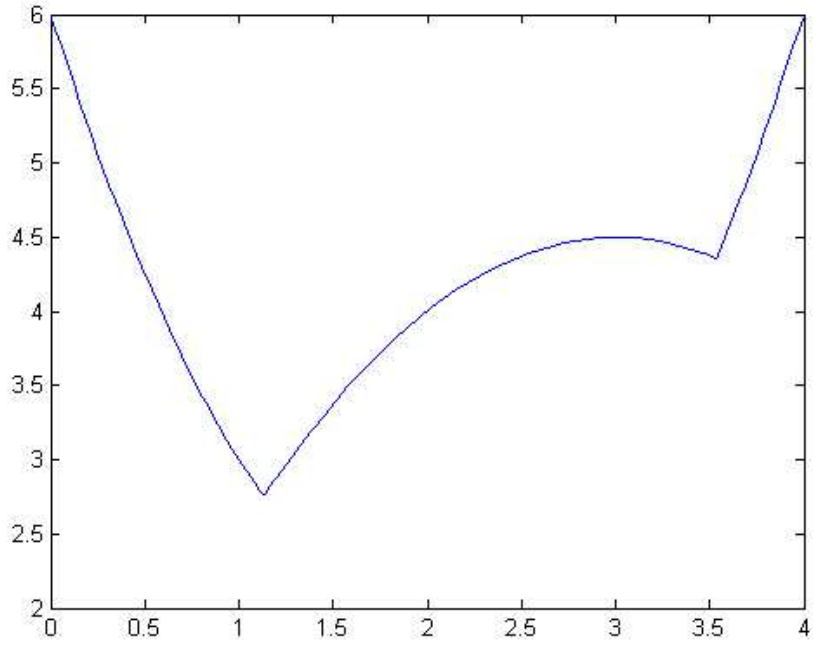

Figure 5.4: Worstcase-corner function for $0 < x < 6$

We have performed tests for 25 different initial points determined by latin hypercube sampling. We obtain the following results,

x_{start}	n_{old}	n_{new}	f_{old}	f_{new}	$\lvert x_{old}\text{-}x_{new}\rvert$	$f_{old} - f_{new}$
0	33	33	2.75432	2.75432	0	0
0.632171	35	35	2.75432	2.75432	1.88402e-08	1.60953e-08
2.83524	33	37	2.75432	2.75432	2.74615e-07	5.13122e-07
5.70147	43	40	2.75432	2.75432	2.23517e-08	3.77553e-08
4.60414	42	40	2.75432	2.75432	0	0
4.90528	39	35	2.75432	2.75432	7.82311e-08	-1.1697e-08
0.377592	38	35	2.75432	2.75432	1.49012e-08	3.74803e-09
2.24067	35	30	2.75432	2.75432	0	0
0.00396507	35	33	2.75432	2.75432	2.98023e-08	2.49474e-08
5.31887	37	35	2.75432	2.75432	0	0
3.34616	43	37	2.75432	2.75432	7.82311e-08	1.3589e-07
1.6697	37	37	2.75432	2.75432	0	0
1.01533	27	32	2.75433	2.75432	1.84581e-06	3.44893e-06
1.32349	42	36	2.75432	2.75432	0	0
3.72979	37	35	2.75432	2.75432	1.45286e-07	2.62962e-07
2.12903	33	33	2.75432	2.75432	0	0
5.55362	37	38	2.75432	2.75432	2.23517e-08	3.88257e-08
2.44506	34	32	2.75432	2.75432	0	0
3.1915	40	39	2.75432	2.75432	1.11759e-08	3.8248e-09
4.94098	39	37	2.75432	2.75432	1.67638e-07	3.13235e-07
0.64497	38	36	2.75432	2.75432	5.76648e-08	1.07748e-07
2.61488	52	43	2.75432	2.75432	1.55859e-07	2.91225e-07
1.75687	38	36	2.75432	2.75432	9.16279e-08	1.59161e-07
1.20081	38	34	2.75432	2.75432	8.94675e-09	1.20806e-08
6	40	42	2.75432	2.75432	7.82311e-08	1.46176e-07

where n_{old} and n_{new} represents the number of evaluations the original respectively current Gridmom algorithm needs for optimizing, and where n_{old}, f_{new}, x_{old} and x_{new} are the corresponding values for the minimum respectively arguments of the minimum. When we look at the results we see that there are 11 situations where both number of evaluations and value of the minimum have improved, 6 situations where one has improved and one remained the same, and at last 8 situations where both number of evaluations and value of the minimum remained the same, or where one of them improves and the other draws back. So, in 17 of the 25 situations we have progress and in the other 8 situations there is not really progress but it is also not worse than before. Other test functions gave more or less the same results, so we may conclude that the check on non-differentiable minima works well.

Chapter 6

Conclusion

6.1 Conclusions

In chapter 4 we have discussed the following techniques and algorithms that could improve the efficiency of Gridmom.

- Using the EGO algorithm in the starting phase as an alternative sampling method. This algorithm is used instead of the Hooke-Jeeves and the Uniform Design algorithms respectively. Hereby we consider different stop criteria for the EGO algorithm instead of the original stop criterium,

  - Termination of EGO when the argument of the maximum expected improvement, rounded to the closest grid point, has already been evaluated before;
  - Termination of EGO when l points are sampled with Latin Hypercube Sampling and v points in the remaining part of the algorithm, for fixed l, $v \in \mathbb{N}$.

  In theory the use of EGO should provide additional information on the behavior of the objective function in an earlier stage of the Gridmom algorithm, and therefore improve the efficiency of Gridmom.

- Applying a piecewise linear approximative fit in the descent phase instead of the quadratic approximative fit, in case that the Hessian of the previous constructed quadratic models is large, and in case the objective function is one-dimensional.

  The use of an piecewise linear model in the descent phase is to improve the convergence speed of Gridmom when the gradient of the minimum of the objective function does not exist.

These techniques and algorithms have been tested in 5. We obtained the following results,

- Using EGO in the starting phase, where we apply the first stop criterium as discussed before, results into an improvement of the value of the minimum, but at the cost of the number of evaluations done by the EGO algorithm. So, using EGO with the mentioned stop criterium in the Gridmom algorithm is not necessarily useful.

- When we use EGO in the starting phase, where we apply the second stop criterium as discussed before, the improvements are completely dependent of the behavior of the objective function.

The Rastrigin, Vardim and Bumpyfun function give the best results for a small number for the sum of the parameters l and v, $l + v$, but the Worstcase-corner function gives the best results if $l + v$ is large. So, it is hard to say for which l, v the EGO algorithm is useful.

- Piecewise linear approximative fit in the descent phase works fine for one-dimensional functions with a non-differentiable minimum. For the Worstcase-corner function we get in $\frac{17}{25} = 68\%$ of the tests better results for the value of the minimum as well as the number of evaluations needed by Gridmom.

6.2 Further research

The next subjects could be investigated more profoundly.

- Up to now, the use of the EGO algorithm in the starting phase applying the first stop criterium discussed in section 6.1 does not give necessarily better results, but it is possible that with extra study of the results and with testing new function it becomes clear whether the algorithm should be used.

- We have discussed and tested piecewise linear approximative fitting in the descent phase for one-dimensional functions with a non-differentiable minimum. In chapter 4 we have also considered higher-dimensional functions and proposed two strategies. It remains the question if these strategies work well in practice.

- Besides using EGO in the starting phase and using piecewise linear approximative fit in the descent phase, some other strategies and observations have been proposed but have not been studied yet. These items, Kriging modelling in the descent phase, efficient optimization in the refinement-check phase, and degeneration in the descent phase, are valuable for further investigation.

Appendix A

Kriging

A.1 Vector Derivatives

Let $\mathbf{x} = (x_1\ x_2\ \cdots\ x_n)^T$ and $\mathbf{y} = (y_1\ y_2\ \cdots\ y_m)^T$ be vectors for some $m, n \in \mathbb{N}$. Also, let $c \in \mathbb{R}$ be a scalar. One can define various derivatives with respect to $\mathbf{x}$, $\mathbf{y}$ and c. Define the derivative of the vector $\mathbf{y}$ with respect to the vector $\mathbf{x}$,

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \tag{A.1}$$

and define the derivative of the scalar c with respect to the vector $\mathbf{x}$ to be the $n \times m$ matrix,

$$\frac{\partial c}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial c}{\partial x_1} & \frac{\partial c}{\partial x_2} & \cdots & \frac{\partial c}{\partial x_n} \end{pmatrix}^T \tag{A.2}$$

consistent with (A.1). With these two definitions we can derive some equations. Let $\mathbf{A} = (a_{ij})$ and $\mathbf{B} = (b_{ij})$ be matrices. Then we obtain:

1. $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}) = \mathbf{I}$;
2. $\frac{\partial}{\partial \mathbf{x}}(\mathbf{y}^T\mathbf{x}) = \mathbf{y}$, since $\frac{\partial}{\partial x_i}(\mathbf{y}^T\mathbf{x}) = y_i$ for all i;
3. $\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x}) = \mathbf{A}^T$, since $\frac{\partial}{\partial \mathbf{x}}(\sum_{k=1}^{n} a_{jk}x_k) = a_{ji}$ for all i, j;
4. $\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x})^T = \left(\frac{\partial}{\partial \mathbf{x}}\mathbf{A}\mathbf{x}\right)^T$;
5. $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}) = \mathbf{A}$, since $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}) = \frac{\partial}{\partial \mathbf{x}}(\mathbf{A}^T\mathbf{x})^T = \frac{\partial}{\partial \mathbf{x}}(\mathbf{A}^T\mathbf{x}) = (\mathbf{A}^T)^T = \mathbf{A}$;

6. $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = (\mathbf{A}+\mathbf{A}^T)\mathbf{x}$, since $\frac{\partial}{\partial x_k}(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i a_{ij} x_j) = \sum_{j=1}^{n} a_{kj}x_j + \sum_{i=1}^{n} x_i a_{ik}$;

7. $\frac{\partial}{\partial \mathbf{x}}((\mathbf{B}\mathbf{x})^T\mathbf{A}(\mathbf{B}\mathbf{x})) = \mathbf{B}^T(\mathbf{A}+\mathbf{A}^T)\mathbf{B}\mathbf{x}$, since $\frac{\partial}{\partial \mathbf{x}}((\mathbf{B}\mathbf{x})^T\mathbf{A}(\mathbf{B}\mathbf{x})) = \frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{B}^T\mathbf{A}\mathbf{B}\mathbf{x})$.

A.2 Latin Hypercube Sampling

Latin Hypercube Sampling is a strategy for generating random sample points in a vector space ensuring that all portions of this vector space are represented. Let $\mathcal{V} \in \mathbb{R}^n$ be an n dimensional vector space where we wish to sample m points for some $n, m \in \mathbb{N}$. Then the Latin Hypercube Sampling strategy consists of three steps:

1. Divide the interval of each dimension into m non-overlapping intervals of the same size by using a uniform distribution;
2. Sample randomly from a uniform distribution in each interval and dimension a point;
3. Combine randomly the point from each dimension.

A.3 Basics of Statistics

A.3.1 Normal Distribution

Let a variate be the set of all random variables that obey a given probabilistic law. Let X be a variate and let $Y \in X$ be normal distributed with mean μ and variance σ^2. Then Y is a statistical distribution with probability function

$$\mathbb{P}(Y = y) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(y-\mu)^2}{2\sigma^2}} \tag{A.3}$$

for $y \in \mathbb{R}$. Now let $Z \in X$ be normal distributed with mean zero and variance 1. This distribution is called a standard normal distribution and is often used for computational reasons. It is easy to convert an arbitrary normal distribution Y to a standard normal distribution Z by changing variables to $Z \equiv (Y-\mu)/\sigma$ and $z := (y-\mu)/\sigma$. We obtain the probability function

$$\begin{aligned} \mathbb{P}(Y = y) &= \mathbb{P}\left(\frac{Y-\mu}{\sigma} = \frac{y-\mu}{\sigma}\right) \\ &= \mathbb{P}(Z = z) \\ &= \frac{1}{\sqrt{2\pi}}\, e^{-\frac{z^2}{2}} \end{aligned} \tag{A.4}$$

with $z \in \mathbb{R}$. This function is often defined by $\phi(z)$. Equivalently we can define the cumulative probability function $\mathbb{P}(Y < y)$ as the integral over the probability function of Y,

$$\mathbb{P}(Y < y) = \int_{-\infty}^{y} \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx \tag{A.5}$$

applying the conversion from Y to Z results into

$$\mathbb{P}(Y < y) = \mathbb{P}(Z < z) \tag{A.6}$$

which is often defined by $\Phi(z)$. Now let $\mathbb{E}(Y)$ be the expected value of Y, also called the expectation of Y. Then $\mathbb{E}(Y)$ is defined as

$$\mathbb{E}(Y) = \int_{y \in \mathbb{R}} y\phi\left(\frac{y-\mu}{\sigma}\right) dy \tag{A.7}$$

In can be proven that $\mathbb{E}(Y) = \mu$. Furthermore, let $\text{Var}(Y)$ be the variance of Y. Then $\text{Var}(Y) = \sigma^2$.

A.3.2 Expectations

Let X_1 and X_2 be two random variables that obey a given probabilistic law and let X_1, X_2 be individually independently distributed. Then we have the following properties for X_1 and X_2 with respect to the expectations (see Appendix A.3.1 for a definition of expectation) of these variables,

1. $\mathbb{E}[X_1] + \mathbb{E}[X_2] = \mathbb{E}[X_1 + X_2]$
2. $\mathbb{E}[X_1 X_2] = \mathbb{E}[X_1]\mathbb{E}[X_1]$
3. $\mathbb{E}[\max\{X_1, X_2\}] = \mathbb{P}(X_1 > X_2)\mathbb{E}[X_1] + \mathbb{P}(X_1 < X_2)\mathbb{E}[X_2]$

Bibliography

[1] *Pstar user manual*. Philips ED&T / Analogue Simulation, 2005.

[2] Kleijnen J.P.C. Siem A.Y.D. den Hertog, D. *The correct Kriging variance estimated by bootstrapping*. 2004.

[3] C. Elster and A. Neumaier. *A grid algorithm for bound constrained optimization of noisy functions*. IMA J.Numer.Anal.15. 1995.

[4] L.F.P. Etman. *Design and analysis of computer experiments: The method of Sacks et al.* Department of Mechanical Engineering, Eindhoven University of Technology, 1994.

[5] K.-T. Fang. and Y. Wang. *Number-theoretic methods in statistics*. Chapman and Hall, London, 1994.

[6] D.E. Finkel. *DIRECT optimization algorithm user guide*. Center for Research in Scientific Computation, North Carolina State University, 2003.

[7] T.G.A. Heijmen. *Nonlinear constrained optimization applied to circuit tuning*. Philips Research, Eindhoven, 2000.

[8] T.G.A. Heijmen and J. van Staalduinen. *Gridmom: a routine for nonlineair constrained optimization*. Philips Research, Eindhoven, 2000.

[9] http://www.nag.co.uk. *NAG Fortran library routine document, E04NFF/E04NFA*.

[10] http://www.nag.co.uk. *NAG Fortran library routine document, F04JGF*.

[11] D.R. Jones, M. Schonlau, and W.J. Welch. *Efficient global optimization of expensive black-box functions*. Kluwer Academic Publishers, Netherlands, 1998.

[12] Nielsen H.B. Søndergaard J. Lophaven, N.S. *DACE: A Matlab Kriging toolbox*. 2002.

[13] G.V. Reklaitis, A. Ravindran, and K.M. Ragsdell. *Engineering optimization: methods and applications*. John Wiley and Sons, New York, 1983.

[14] Welch W.J. Mitchell T.J. Wynn H.P. Sacks, J. *Design and analysis of computer experiments*. 1989.

[15] M.J. Sasena. *Flexibility and efficiency enhancements for constrained global design optimization with Kriging approximations*. University of Michigan, 2002.

[16] M. Schonlau. *Computer experiments and global optimization*. University of Waterloo, Canada, 1997.

[17] Kleijnen J.P.C. van Beers, W.C.M. *Kriging for interpolation in random simulation*. 2001.