UNIVERSITY OF SOUTHAMPTON

Kriging Methods for Constrained Multi-Objective Electromagnetic Design Optimization

by

Glenn Hawe

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

February 2008

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Glenn Hawe

Design problems in electrical engineering are typically solved using a computationally expensive numerical method, such as the finite element method. As the designs of most interest are those which are optimal in some way, any algorithm used to search for such designs must perform well in a low number of iterations, if they are to be identified within a reasonable time.

Kriging is a method of making predictions (based on a set of observations) which has been used as a surrogate model to reduce computational cost in optimization searches. After reviewing the state-of-the-art in kriging-assisted single and multi-objective optimization, this thesis proposes several novel algorithms for efficiently solving constrained (or unconstrained) multi-objective optimization problems. Using a combination of these novel algorithms and existing methods, a practical optimization tool is integrated into commercial electromagnetic design software (Opera). The tool is demonstrated on four different optimal design problems, which cover the four possibilities of unconstrained and constrained single and multi-objective optimization. Some areas of potential future work are given in the final chapter.

Acknowledgements

Thanks firstly to my supervisor Prof. Jan Sykulski; without his enthusiasm and encouragement, this would not have happened.

Thanks to all the staff at Vector Fields, Kidlington, in particular Dr. Simon Taylor for all his help over the past three years. A mention of thanks to the rest of the KTP team: John Simkin, Kevin Ward and Gill Rysiecki.

Thanks to my parents for providing support in the final stages of writing up.

Thanks to Gerry and Christine McGragh and everyone else at the King's Arms in Kidlington. I look forward to visiting again in the future.

Thanks to everyone who made the past three years fly by, especially Brian, Daphne, Donald, Liam, Mark and Roly.

Finally, many thanks to Agata, especially in the final days of writing up.

Contents

List of Figures

List of Tables

Abbreviations

DACE	Design and Analysis of Computer Experiments
EGO	Efficient Global Optimization
EI	Expected Improvement
GEI	Generalized Expected Improvement
MOEA	Multi Objective Evolutionary Algorithm
MOOP	Multi Objective Optimization Problem
POI	Probability Of Improvement
SOOP	Single Objective Optimization Problem
WEI	Weighted Expected Improvement

Symbols

d	number of design variables
M	number of objective functions
J	number of inequality constraints
K	number of equality constraints
f_i	objective function
g_i	inequality constraint
h_i	equality constraint
$\mathbf{x}$	design vector
θ	kriging hyperparameter
$\mathbf{p}$	kriging hyperparameter
$\mathbf{R}$	correlation matrix
$\mathbf{r}$	correlation vector
$\mathbf{C}$	conditional correlation matrix
$\mathbf{m}$	conditional mean vector
μ	kriging regression constant
$\hat{y}$	kriging prediction
I	improvement
g	generalized expected improvement control parameter
w	weighted expected improvement control parameter
T	probability of improvement control parameter (target objective function)
f^*	credibility of hypothesis control parameter (hypothesized objective function value)

Part I

Introduction

Chapter 1

Introduction

1.1 Background

1.1.1 Electromagnetic Design Problems

Electromagnetic design problems (as with all engineering design problems), may be divided into two broad categories [16]:

Direct Problems, in which a design is specified, and the effects are found.

Inverse Problems, in which a design is sought which produces a specific effect, which may be measured or assumed. If the effect is measured, the problem is an *identification* problem. If the effect is assumed, the problem is a *synthesis* problem, to which a solution may not exist.

This is summarized in Figure 1.1. To solve the direct problem, either analytical, semi-analytical or numerical methods are used. Often analytical methods cannot be used, and so the direct problem is usually solved using a numerical method, such as finite-difference (FD), finite-element analysis (FEA) or boundary-element analysis (BEA). The numerical method can often be computationally expensive (i.e. it is time-consuming), depending on the complexity of the design problem.

The solution of the inverse problem, and in particular the optimal design problem, is an active area of research. The optimal design problem phrases the effects as objectives, which must be minimized or maximized, and so is a synthesis problem. In general, it can only be solved by solving the direct problem multiple times; thus it is inherently more time-consuming than the direct problem.

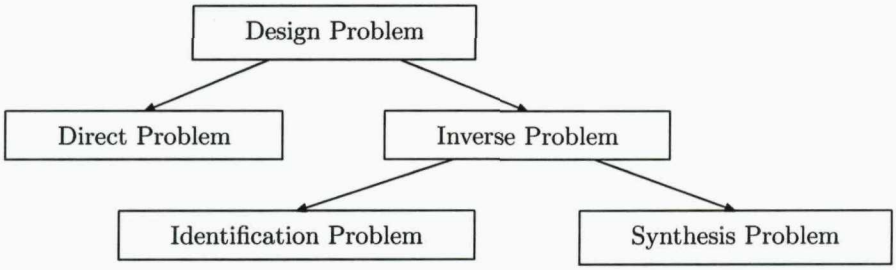


FIGURE 1.1: Types of design problem.

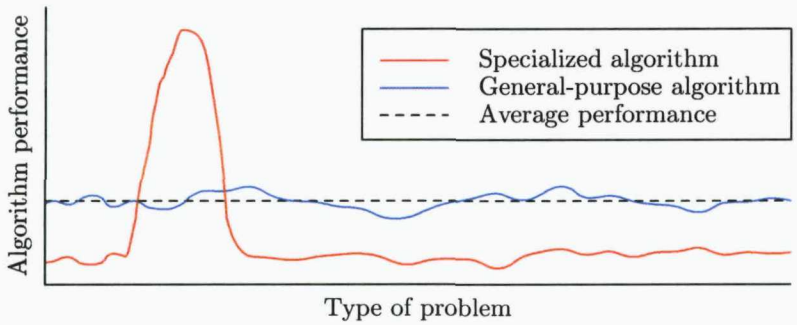


FIGURE 1.2: No Free Lunch theorem of optimization.

1.1.2 Solving Inverse Problems in Electromagnetic Design

The 'No Free Lunch' (NFL) Theorem [17, 18] of optimization denies the existence of any optimization algorithm which outperforms all other optimization algorithms, when averaged over all possible optimization problems. In fact, averaged over all optimization problems, every optimization algorithm performs the same. At first, this seems a dire predicament; however, for design engineers, it is advantageous that this is the case, as their domain of interest is not every possible optimization problem, but rather a subset of them. Thus, because of the NFL Theorem, it *is* possible to identify an algorithm (or more generally, a set of algorithms) which do outperform others *over our domain of interest* [19] (as a consequence of the same algorithms then being outperformed by other algorithms on problems *outside* our domain of interest), as illustrated in Figure 1.2. The domain of interest in this thesis is that of electromagnetic design problems.

The range of algorithms which now exist for solving optimization problems is vast. They may be categorized in many ways: single-objective or multi-objective; deterministic or stochastic; global or local; greedy or cost-effective; and so on .... As may be expected, the wide range of electromagnetic optimal design problems do not fall neatly into one category: they may be single or multi-objective; their objective function landscapes may be simple or relatively complex (although not pathological); the objective

functions may take anything from a few seconds to several days to evaluate; they may be constrained or unconstrained; and so on .... Any attempt to identify a single algorithm to deal with them all would be futile, and so instead ideally a set of algorithms should be available to meet electromagnetic design engineers needs. In particular, the family of problems with multiple computationally expensive objective and constraint functions need to be catered for, and it is algorithms which deal with this family of problems which this thesis focuses on.

Several different methods exist for achieving cost-effectiveness in multi-objective optimization, including small population genetic algorithms, hybrid algorithms, reduction of design variables and fitness inheritance. These methods are not pursued in this thesis, but brief descriptions of these techniques may be found in Appendix A. Instead, the cost-effective method pursued in this thesis is that of surrogate modeling, which is introduced in detail in Chapter 3.

1.2 Motivation

1.2.1 Vector Fields Software

Vector Fields Ltd. is a software company based in Oxford, England, which produces CAD software for electromagnetic design. It produces both low and high frequency simulation software, called Opera and Concerto respectively. Opera is further divided into Opera-2d and Opera-3d. The software is capable of modelling a wide range of applications including antennas, motors, and MRI devices.

Opera calculates electromagnetic field values by solving the governing partial differential equations using the finite element method (FEM), whilst Concerto uses the finite difference time domain (FDTD) method. Both methods are (or rather, *can be*) computationally expensive, and typically a model may take anything from a few seconds (for a simple 2d problem) to several hours or even days (for a complex, long time-domain 3d model) to solve.

Prior to this work, Vector Fields software was analysis driven, that is, it could only be used to solve the direct design problem. It was the purpose of this work to make the software goal oriented, that is, capable of solving the inverse design problem, and in particular the optimal design problem.

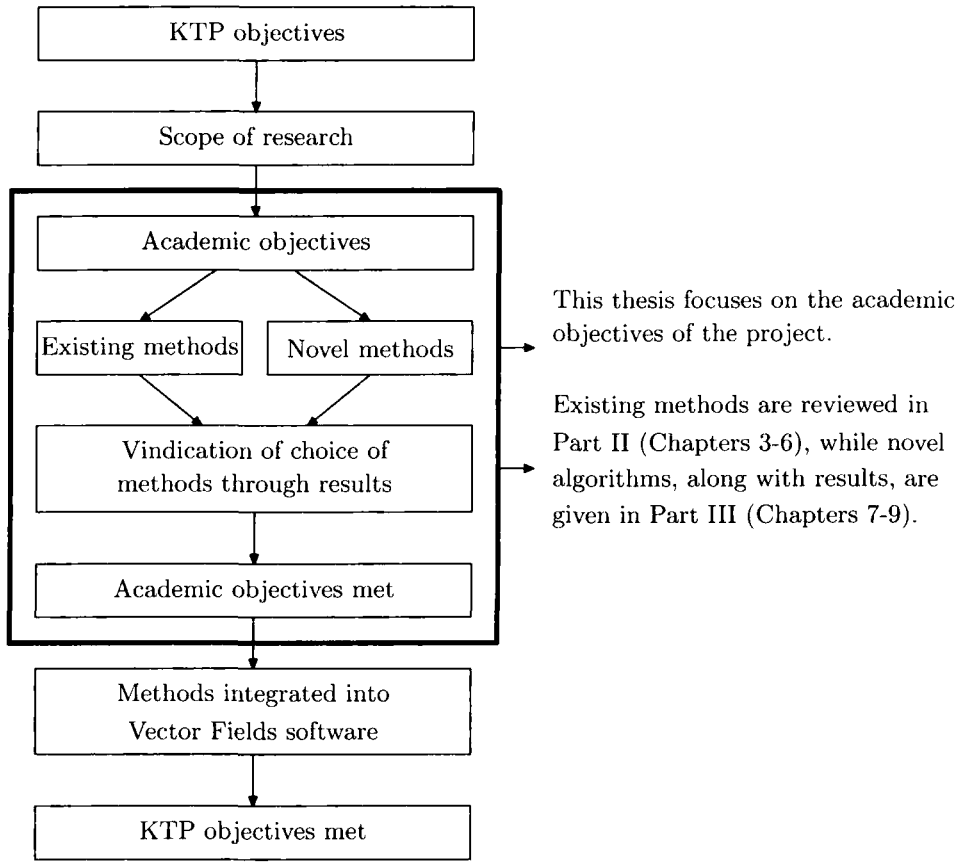


FIGURE 1.3: Objectives and scope of the research.

1.2.2 Objectives and Scope of the Research

The research was undertaken as part of a Knowledge Transfer Partnership (KTP) Scheme between the University of Southampton and Vector Fields Ltd. As such, two sets of objectives existed for the research: the KTP objectives (which focused on the company perspective), and the academic objectives. The two sets were closely related: the KTP objectives defined the overall scope of the research, within which the set of academic objectives were specified. The integration of the methods used in the fulfilment of the academic objectives into Vector Fields software then resulted in the business aims being met, thus satisfying the KTP objectives. This is illustrated in Figure 1.3.

1.2.2.1 KTP Objectives

As stated in the KTP Partnership Proposal and Grant Application Form for the scheme,

'the primary objective is to integrate state-of-the-art optimization tools into Vector Fields' software products for electromagnetic design.'

As other electromagnetic design software companies have already started to develop and release optimization tools in their software [20], the need for a state-of-the-art optimization tool in Vector Fields software is vital in order to remain competitive. In order for the above general objective to be met effectively, the tool must be better than existing rival tools, in terms of capability, versatility, and ease-of-use. Thus the main objective may be met effectively by introducing an optimization tool into the software which satisfies the following more precise objectives:

1. To match features available, and provide features currently unavailable, in rival optimization tools.
2. To provide results which are superior to rival optimization tools in the widest range of design problems possible.
3. To provide a user-friendly interface to the tool.

Optimization facilities available within other commercial electromagnetic design software include optimization of 2d or 3d models, specification of multiple design variables, constraints and objective functions, ability to run in parallel on multiple computers, and continuous progress updates as the optimization proceeds. These features should serve as a minimum specification for the eventual Vector Fields optimization tool (even though no existing tool has all these features themselves).

Furthermore, facilities currently unavailable in all other commercial electromagnetic optimization tools should be included in the software. The most important such facility which will be considered is the approximation of the Pareto-optimal front in multi-objective optimization problems. Offering a commercial multi-objective optimization tool which yields a set of Pareto-optimal solutions, particularly for computationally expensive problems, would give Vector Fields a distinct advantage in the market.

As noted before, the range of problems which exist in electromagnetic design is vast, and vary widely in their characteristics. Some problems may be low-dimensional, and have a single objective which is relatively simple, whilst others may have dozens of design variables and multiple objectives and constraints which require hours to evaluate. Ideally, the tool should be able to cope with the widest range of problems possible. This precludes the use of problem-specific methods, including sensitivity-based approaches to optimization [21], which have been used successfully with Opera in the past [22, 23]. Such approaches have the advantage of having computation times independent of the number of design variables, thus making them useful in topology optimization [24], where the parameterization enables all feasible shapes of electromagnetic devices to be

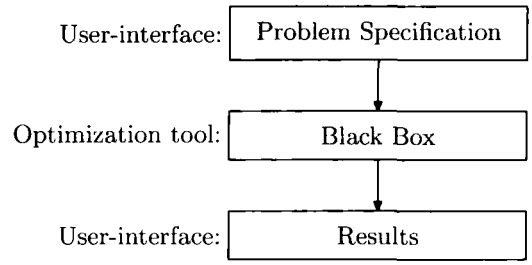


FIGURE 1.4: Interaction between the user and optimization tool.

explored. However they are restrictive in the number of systems they can solve, and so are not deemed versatile enough to be included in the eventual tool.

The generic nature of the tool should not, however, come at a price to the tool's performance. As mentioned before, no single algorithm will be universally applicable to all design problems. Instead, a range of methods should be available, which may be combined to give the most appropriate algorithm, given a particular problem. The existence of a range of different methods within the tool, however, should not (necessarily) be matched by a wide range of options available to the user. Between setting up the problem and receiving the results, ideally the user should not have to interact with the optimization tool. This means that the tool should be a 'black box' from the user's point of view: the user will provide a design, and then wait for results, without caring for how the results are arrived at (this is illustrated schematically in Figure 1.4). Thus, the software should make an automated, intelligent choice of which methods to use to solve a given problem, based on all the initial information which can be extracted from the problem. In addition, the setting of any parameters internal to the methods used should be automatic - or at the very least - sensible default parameters should be provided.

The actual interaction of the user with the optimization tool occurs at two main stages of the optimization process: during the setting up of the problem, and when viewing the results. Regardless of the capability and versatility of the tool, if it is to be accepted and used by design engineers the interface providing this interaction must be user-friendly. Thus, the user should be able to define all the design variables in a parameterized model, along with the required objective functions and constraints, with ease. The results which are presented should be useful, clear, and produced at relevant times.

1.2.2.2 Scope of the Research

As suggested by the black-box model of the optimization tool, there are three main stages in the optimization process: problem specification, solving the problem with an appropriate algorithm, and the presentation of the results. The scope of the work is defined by what has to be achieved within each of these stages:

I Problem Specification

1. Statement of design variables (parameterization of the problem).
2. Statement of constraints and objectives.

II Solving the Problem

1. Choice of initial set of designs to evaluate.
2. Optimization using an appropriate algorithm.
3. Parallelization.
4. Validation of the obtained solutions.
5. Selection of solutions to display to the user.

III Presentation of Results

1. Presentation of the selected solutions to the user.

Some of these areas contain methods which do not require much development *per se*, whilst others have the potential for investigation and improvement. In particular, most of the areas within stage II are active areas of research, and as such have much potential for development. On the other hand, methods used within stages I and III do not allow much scope for improvement, and may be implemented directly into Vector Fields software, without much modification. Work done in these areas will certainly contribute to satisfying the KTP objectives, but it is within stage II that most originality will be sought, and which this thesis concentrates on.

1.2.2.3 Academic Objectives

The general academic aim of the work is the creation of an optimization tool which is sufficient to effectively solve any electromagnetic optimal design problem which may be set up using Vector Fields software. More precisely, the research will attempt to satisfy the following objectives:

A1. To extend well-established cost-effective single-objective optimization techniques to the multi-objective case.

A2. To integrate constraint handling methods into cost-effective multi-objective algorithms.

A3. To use surrogate modelling techniques which are novel to electromagnetic design optimization.

A4. To automate the choice of methods used to solve a given electromagnetic optimal design problem.

A5. To allow sufficient flexibility in parameterization to allow the design variables, constraints and objective functions in an electromagnetic design problem to be easily set up.

A6. To test the developed strategies on a variety of real-life test problems within electromagnetic design.

Implementation of the methods used to achieve these academic objectives into Vector Fields software then leads to the fulfilment of the KTP objectives.

1.3 Methodology

As already mentioned, electromagnetic optimal design problems vary widely in their characteristics. Plenty of algorithms exist for dealing with the simplest: a standard genetic algorithm can deal with single-objective problems with inexpensive objective functions; whilst a standard Multi-Objective Evolutionary Algorithm (MOEA), such as Multi-Objective Genetic Algorithm (MOGA) [25], the Non-dominated Sorting Genetic Algorithm (NSGA, NSGA-II) [26, 27], the Niched-Pareto Genetic Algorithm (NPGA) [28], the Pareto-Archived Evolutionary Strategy (PAES) [29] or the Strength Pareto Evolutionary Algorithm (SPEA, SPEA-II) [30, 31] can deal with multi-objective problems with inexpensive objective problems. This thesis concentrates exclusively on algorithms dealing with the most complex type of problem: those with objective functions so computationally expensive that the use of a cost-effective technique is essential.

The cost-effective technique used within this thesis is that of surrogate modeling, in particular kriging. A wide range of effective kriging-assisted algorithms exist for single-objective optimization problems [14], although their application to electromagnetic design problems has been slow. After providing an introduction to kriging (and more

generally, surrogate modeling) in Chapter 3, these existing single-objective optimization algorithms are reviewed in Chapter 4.

The extension of the ideas in cost-effective single-objective optimization to cost-effective multi-objective optimization has also been slow; indeed all of the algorithms reviewed in Chapter 5 have appeared in the literature since the beginning of this research. The dearth of literature on cost-effective multi-objective optimization is reflected by the absence of a doctoral thesis on the topic (the evolution of thesis topics in the field of cost-effective optimization (shown in Table 1.1) is illustrative of the progress made in the field generally over the past decade). Clearly cost-effective multi-objective optimization is a field in its infancy.

TABLE 1.1: Table of recent PhD theses on cost-effective optimization.

Year	*Title*	*Author*	*University*	*Scope*			
				SOOPs	MOOPs	Constraint-handling	Applications
1997	Computer Experiments and Global Optimization	M. Schonlau	Waterloo, Canada	Y	N	N	Automotive
1997	Aircraft Multidisciplinary Design Optimization Using Design of Experiments Theory and Response Surface Modeling Methods	A. A. Giunta	Virginia, U.S.A.	Y	N	N	Aerospace
2001	Radial Basis Function Methods for Global Optimization	H-M. Gutmann	Cambridge, U.K.	Y	N	N	Test-functions
2001	*Cost-Effective Evolutionary Strategies for Pareto Optimal Front Approximation in Multiobjective Shape Design Optimization of Electromagnetic Devices*	*M. Farina*	*Pavia, Italy*	N	Y	N	*Electromagnetic*
2002	Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations	M. J. Sasena	Michigan, U.S.A.	Y	N	Y	Mechanical Engineering
2004	Efficient Global Aerodynamic Optimisation Using Expensive Computational Fluid Dynamics Simulations	A. J. Forrester	Southampton, U.K.	Y	N	Y	Aerospace

Completing the review Part, Chapter 6 deals with many of the practical issues involved in cost-effective optimization, which are often neglected in the literature. This chapter is intended to be useful to anyone implementing the algorithms in this thesis for themselves.

Chapter 7 then introduces novel algorithms, building mainly on the ideas already discussed. It is the intention that Chapter 4 and Chapter 5 will have presented existing methods in such a way that the inspiration for these novel algorithms is immediately obvious. The emphasis is on constructing algorithms to solve computationally expensive, constrained multi-objective optimization problems, thus providing a natural progression from the work covered in Table 1.1.

Chapter 8 provides details of the implementation of the optimization tool within Opera, and Chapter 9 gives results of this tool on some electromagnetic design problems. Chapter 10 finishes with conclusions and proposals for further work.

Part II

Review

Chapter 2

Optimization

2.1 Single-Objective Optimization

2.1.1 The Single-Objective Optimization Problem

The **Single-Objective Optimization Problem (SOOP)** may be stated as follows [1]:

$$\begin{array}{lll} \text{Minimize} & f(\mathbf{x}) & \\ \text{subject to} & g_j(\mathbf{x}) \geq 0 & j = 1, 2, \ldots, J; \\ & h_k(\mathbf{x}) = 0 & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)} & i = 1, 2, \ldots, d. \end{array} \tag{2.1}$$

The problem is said to be unconstrained if $J = K = 0$; otherwise it is said to be constrained. Each design variable has a lower and an upper bound between which it may vary continuously (however, more generally the design variable may also only be allowed to vary discretely [20]). In order to be feasible, a design vector must satisfy the J inequality and K equality constraints. This gives rise to the *feasible region* S which is a subset of the *decision variable space* $\mathbb{R}^d$,

$$S = \{\mathbf{x} \in \mathbb{R}^d \text{ s.t. } \mathbf{g}(\mathbf{x}) \geq 0 \text{ and } \mathbf{h}(\mathbf{x}) = 0\}, \tag{2.2}$$

[1] Alternatively, it may be formulated as a maximization problem; however this thesis will assume all objective functions are to be minimized.

and its image the *feasible objective space* Z which is a subset of the *objective space* $\mathbb{R}$,

$$Z = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R} \text{ s.t. } \mathbf{x} \in S\}. \tag{2.3}$$

Within the feasible region S, f takes some global minimum value $f_{\min}$: for some particular optimization problems, the value of $f_{\min}$ may be known a-priori, and this can be useful for some algorithms in determining where to evaluate (see Section 4.4). In general however, $f_{\min}$ is unknown. The aim of single-objective optimization is to locate the design vector(s) in S which have the objective function $f_{\min}$.

2.1.2 Difficult Features of Objective Functions

Several features of an objective function may provide particular difficulty to optimization algorithms in locating its global minimum. In [32], five such particularly significant features are identified:

1. The degree of modality.
2. The size of basins of attraction of local minima.
3. The size of improving regions (and the magnitude of oscillations).
4. The degree of randomness in the positions of the minima.
5. The dimension of the search space.

The first four of these features are discussed in detail in [32]. The fifth feature, the so-called 'curse of dimensionality', is a well known problem in optimization, and is best tackled through reduction of the number of design variables, which may be achieved by identifying and eliminating those variables which have least effect on the objective function (see Section A.4).

Performance criteria (and other details) of global optimization algorithms which are often used for comparison in the literature are listed in [33]. These include:

1. Best function value found.
2. CPU time.
3. Number of function evaluations.
4. Accuracy.

5. Average number of iterations required per replication (in multiple replications).
6. Number of replications.
7. Success rate.
8. Tuning parameters.
9. Stopping criteria.
10. Platform.

Many of these performance criteria are trivially obvious: for best function value, closer to optimal is better; for CPU time, faster is better; for number of function evaluations, fewer is better; and so on .... However, some may not be so obvious. For example, the number of tuning parameters is an important detail when recommending a global optimization algorithm: in this case, fewer tuning parameters is (generally) better. Examples of tuning parameters include mutation rate in genetic algorithms, and cooling schemes in simulated annealing algorithms. Because of the importance of this feature (from a user's point of view), this thesis will pay particular attention to distinguishing between algorithms which require tuning parameters, and those which do not.

2.1.3 Single-Objective Test Functions

The most popular suite of test functions for single-objective optimization is the Dixon-Szego test set [34]. Unfortunately, whilst this set still serves as a common test-set for single-objective optimization algorithms in the literature, it has long been recognized that the problems in this set are too simple for most modern algorithms, particularly surrogate model-assisted algorithms.

Using the features identified Section 2.1.2, a new set of test functions for global optimization was proposed in [35]. The severity of the difficult features in these functions can actually be controlled through several parameters, and it is anticipated that these functions, and others like them, shall become a new standard in assessing the performance of single-objective optimization algorithms in the future.

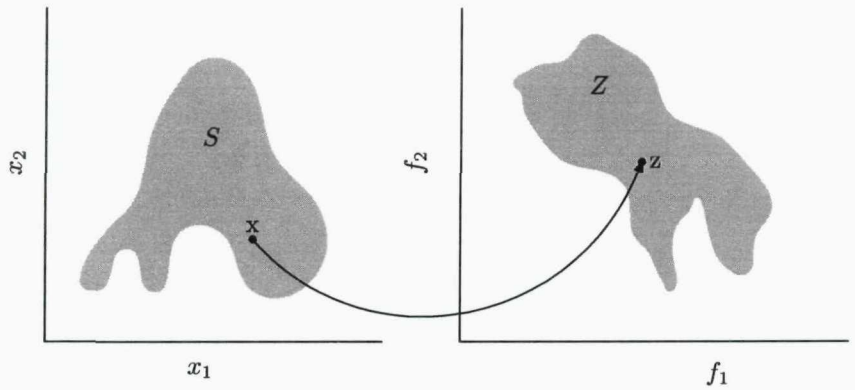


FIGURE 2.1: Example feasible region and feasible objective space.

2.2 Multi-Objective Optimization

2.2.1 The Multi-Objective Optimization Problem

The **Multi-Objective Optimization Problem (MOOP)** may be stated as follows [2]:

$$\begin{array}{lll} \text{Minimize} & f_m(\mathbf{x}) & m = 1,2,\ldots,M; \\ \text{subject to} & g_j(\mathbf{x}) \geq 0 & j = 1,2,\ldots,J; \\ & h_k(\mathbf{x}) = 0 & k = 1,2,\ldots,K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)} & i = 1,2,\ldots,d. \end{array} \tag{2.4}$$

Again, in order to be feasible, a design vector must satisfy the J inequality and K equality constraints. As before, this leads to a *feasible region* S existing within the *decision variable space* $\mathbb{R}^n$; in this case, its image the *feasible objective space* Z is a subset of the *objective space* $\mathbb{R}^M$,

$$Z = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^M \text{ s.t. } \mathbf{x} \in S\}. \tag{2.5}$$

An example feasible region S and feasible objective space Z are illustrated for a two-objective problem in Figure 2.1. Such graphical illustration is useful when the number of objectives or dimension of the design vector is less than or equal to three, but for $M, d > 3$, such a graphical illustration becomes impossible. The aim of the problem is to try to simultaneously minimize the M different objectives f_m, something which in general is not possible, leading to the necessity of defining exactly what constitutes a solution to the MOOP in Equation 2.4.

[2] Again, it may be formulated instead as a maximization problem; however this thesis will assume all objective functions are to be minimized.

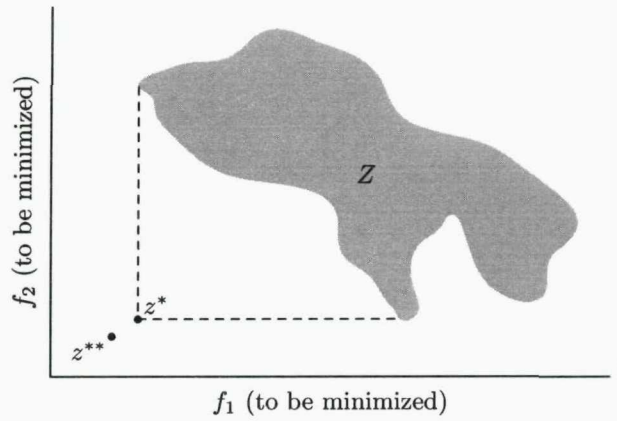


FIGURE 2.2: Ideal and utopian objective vector.

2.2.2 Special Solutions

Definition 2.1 (Ideal Objective Vector). The m-th component of the ideal objective vector $\mathbf{z}^*$ is the constrained minimum solution of the following problem:

$$\begin{aligned} &\text{Minimize} && f_m(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in S \end{aligned}$$

for $m = 1, \ldots, M$.

Definition 2.2 (Utopian Objective Vector). A utopian objective vector $\mathbf{z}^{**}$ has each of its components marginally smaller than that of the ideal objective vector:

$$z_i^{**} = z_i^* - \epsilon_i$$

with $\epsilon_i > 0, i = 1, \ldots, M$.

Both of these special solutions are illustrated in Figure 2.2. It should be noted that the utopian objective vector is not a feasible solution. The ideal objective vector is feasible only when the M objectives of the MOOP are not competing. In this special case, the ideal objective vector $\mathbf{z}^*$ is then the solution to Equation 2.4.

2.2.3 Pareto-Optimality

In general, the M objectives of Equation 2.4 are competing, and so the ideal objective vector is of little use. Another definition of optimal is needed, and the definition usually

adopted is that of Pareto-optimality.[3] Central to the idea of Pareto-optimality is the notion of dominance, and a non-dominated set:

Definition 2.3 (Dominance). For any two solutions $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)} \in S$, $\mathbf{x}^{(1)}$ is said to dominate $\mathbf{x}^{(2)}$ if and only if both the following conditions are true:

1. $f_i(\mathbf{x}^{(1)}) \leq f_i(\mathbf{x}^{(2)})$ for all $i = 1, 2, \ldots, M$
2. $f_i(\mathbf{x}^{(1)}) < f_i(\mathbf{x}^{(2)})$ for at least one $i \in \{1, 2, \ldots, M\}$

Thus, a solution $\mathbf{x}^{(1)}$ is said to dominate another solution $\mathbf{x}^{(2)}$ if and only if it is better in at least one of the objectives, *and* it is no worse in all other objectives. In this sense, $\mathbf{x}^{(1)}$ is a better solution than $\mathbf{x}^{(2)}$. We say that $\mathbf{x}^{(1)}$ is non-dominated by $\mathbf{x}^{(2)}$.

Definition 2.4 (Non-dominated set). Among a set of solutions $P \in S$, the non-dominated set of solutions P' are those that are not dominated by any member of the set P.

Out of a set of feasible solutions P to problem Equation 2.4, the solutions which may be considered optimal from the set are those in the non-dominated set. When the set P of feasible solutions is the entire feasible region S, then the non-dominated set P' is called the (global) *Pareto-optimal set*.

Definition 2.5 ((Global) Pareto-optimal set). The non-dominated set of the entire feasible search space S is the global Pareto-optimal set.

The global Pareto-optimal set is not dominated by any other solutions in S, and so is the solution to Equation 2.4. As well as the global Pareto-optimal set, local Pareto-optimal sets may also exist:

Definition 2.6 ((Local) Pareto-optimal set). A set of solution vectors L is a local Pareto-optimal set if $\exists \delta > 0$ such that $\forall \mathbf{x} \in L$, $\mathbf{x}$ is non-dominated in $L \bigcap B(\mathbf{x}, \delta)$.

The set of values in the objective space which correspond to the solutions in the global (local) Pareto-optimal set is called the global (local) Pareto-optimal front.

Definition 2.7 ((Global) Pareto-optimal Front). The image of the global Pareto-optimal set in the feasible objective space is the global Pareto-optimal front.

Definition 2.8 ((Local) Pareto-optimal Front). The image of a local Pareto-optimal set in the feasible objective space is a local Pareto-optimal front.

The Pareto-optimal front for an objective space Z is highlighted in Figure 2.3. This objective space also has a local Pareto-optimal front, which is also highlighted.

[3] It should be noted that other definitions of optimality exist, such as that of Nash Equilibrium, which is beginning to find applications in electromagnetic design engineering [36, 37].

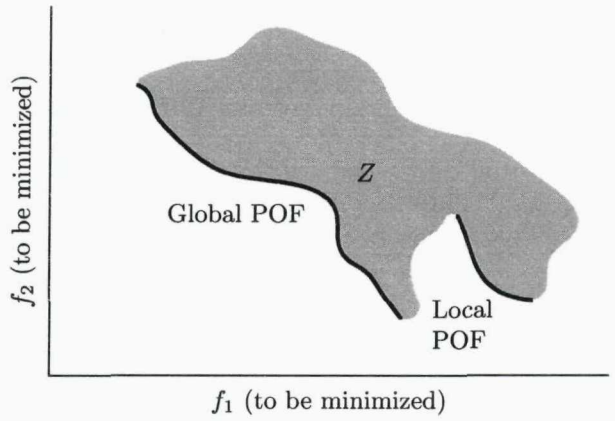


FIGURE 2.3: Global and local Pareto-optimal fronts.

2.2.4 Related Concepts

In addition to Pareto-optimality, two other related concepts are widely used, namely weak and proper Pareto-optimality. The relationship between the concepts is that the properly Pareto-optimal set is a subset of the Pareto-optimal set, which is a subset of the weakly Pareto-optimal set.

Weak Pareto-optimality is based on the notion of strong dominance:

Definition 2.9 (Strong Dominance). A solution $\mathbf{x}^{(1)}$ strongly dominates a solution $\mathbf{x}^{(2)}$ if $f_i(\mathbf{x}^{(1)}) < f_i(\mathbf{x}^{(2)})$ for $i = 1, \ldots, M$.

Thus if a solution $\mathbf{x}^{(1)}$ is better than another solution $\mathbf{x}^{(2)}$ in *all* objectives, then $\mathbf{x}^{(1)}$ is said to strongly dominate $\mathbf{x}^{(2)}$. Using the notion of strong dominance, the *weakly* non-dominated set of solutions can be identified from a general set of solutions.

Definition 2.10 (Weakly non-dominated set). Among a set of solutions $P \in S$, the weakly non-dominated set of solutions P' are those that are not strongly dominated by any other member of the set P.

When the set P of feasible solutions is the entire feasible region, then the non-dominated set is called the *weakly Pareto optimal set*, of which the Pareto optimal set is a subset.

The idea behind proper Pareto optimality is that unbounded trade-offs between objectives are not allowed. Several definitions exist for proper Pareto optimality, see [38] for details.

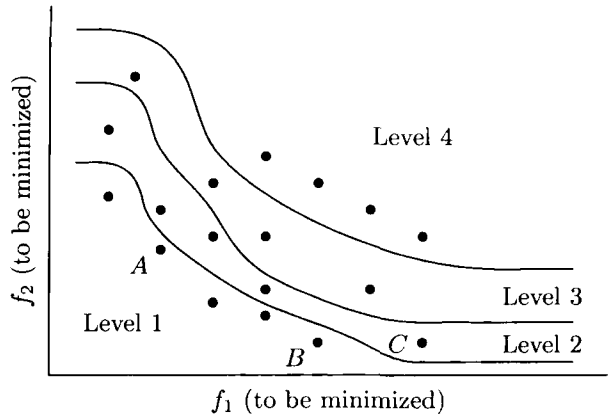


FIGURE 2.4: Non-domination levels.

2.2.5 Identification of Non-Dominated Sets

The identification of the non-dominated set from a set of solutions is a task which is central to any algorithm which tries to yield the Pareto-optimal set. The methods of Kung [39] and Bentley [40], which are based on a divide-and-conquer strategy, have become the standard methods used, however recently Yukish [41] has developed an efficient hybrid method for identification of Pareto-optimal points from multi-dimensional data sets.

2.2.6 Non-Dominated Sorting

Some multi-objective optimization algorithms require solutions in a set to be classified into different non-domination levels. The best non-dominated solutions are called non-dominated solutions of level 1. The next best non-dominated solutions, non-dominated solutions of level 2, are the solutions in the non-dominated set which result by ignoring the non-dominated solutions of level 1 from the population. Non-dominated solutions of level 3 are then found by finding the non-dominated set which result by ignoring the non-dominated solutions of level 1 and 2, and so on. The procedure is continued until all population members are classified into a non-dominated level. Many different algorithms exist for carrying out non-dominated sorting.

The concept of non-dominated sorting can be the source of some confusion. For example, its explanation in [42] finishes with the statement that *'it is important to reiterate that non-dominated solutions of level 1 are better than non-dominated solutions of level 2, and so on.'* This however is rather misleading: one could erroneously assume from this statement that a solution of level 1 must be better than a solution of level 2, which is clearly not the case, as can be seen from Figure 2.4. Shown is a set of solutions which has been

classified into four different non-domination levels. Three solutions are labeled: A, B and C. Solutions A and B are of level 1, whilst solution C is of level 2 (as it is dominated by B). Although of a higher non-domination level than C (where by 'higher' it is meant a low number, with '1' deemed the highest level), solution A does not dominate C: instead it is equivalent to it. The reason A and C are not in the same non-domination level is because of the existence of B, which, while equivalent to A, *does dominate* C, *thus pushing it down a non-domination level.*

So it is important to be clear what can and cannot be said about solutions in different non-domination levels. While *it is not (necessarily) the case* that a solution of level N dominates every solution of level $N+1$, *it is the case* that each solution of level $N+1$ is dominated by at least one solution in level N. This is essentially all that the notion of non-dominating sorting has to offer: when comparing two solutions of level N_1 and N_2 ($N_1 \neq N_2$), we can only say something definite about the relationship between the two groups of solutions, *viz.* for $N_1 < N_2$, a solution in level N_1 either dominates the solution in level N_2 outright, or if not, is equivalent to another solution which does dominate it. Thus, the group of solutions of level N_1 considered as a whole, is better than the group of level N_2 solutions, again considered as a whole, in that for every solution in level N_2, there exists at least one solution in level N_1 which dominates it. It is this property which makes non-dominating sorting a useful concept.

2.2.7 General Objectives in Multi-Objective Optimization

In the absence of higher-level information, all Pareto-optimal solutions are equally important. Therefore it is important that as many Pareto-optimal solutions are found as possible. This leads to there being two main goals in multi-objective optimization:

1. To find a set of solutions as close as possible to the Pareto-optimal front.
2. To find a set of solutions as diverse as possible.

The first goal is simply stating that the solutions found are to be as close to the true optimal solutions as possible. The second goal is specific to multi-objective optimization, and is important because a diverse set of solutions assures us that no single objective is being favoured over any other in the search. It should be noted that the diversity among solutions may be measured in two different spaces, the decision variable space and the objective space. Achieving a good balance between convergence to and diversity along the Pareto-optimal front is important to all multi-objective optimization algorithms [43].

Different performance indicators exist to assess and rank an algorithm's performance in converging to the Pareto-optimal Front, maintaining diversity in solutions, and overall performance considering both convergence and diversity together [44]. These indicators enable easier comparison between multi-objective optimization algorithms. Popular performance indicators (as used in the assessment of the performance of the multi-objective algorithm ParEGO [45], described in Section 5.2.3) include the S measure [46], the additive epsilon indicator [30] and median and worst attainment surface plots [47].

2.2.8 Difficult Features of Pareto-Optimal Fronts

Objective function space may possess certain features which make it particularly difficult for a multi-objective algorithm to locate the Pareto-optimal front. Some of the more common features encountered are described below.

2.2.8.1 Multi-Modality

Definition 2.11 (Multi-modal). A MOOP is multi-modal if and only if more than one local Pareto-optimal Front exists.

Multi-modality is the multi-objective equivalent of multiple solutions in single-objective optimization. The MOOP with objective space as shown in Figure 2.3 is multi-modal (a global Pareto-optimal front is also a local Pareto-optimal front).

2.2.8.2 Isolated Optimum

An algorithm relies on the features of objective space to be able to search for the optimum. If the objective space is relatively featureless, except for the optimum, then any algorithm will have difficulty converging, as it will have no information to guide it.

2.2.8.3 Convexity

Definition 2.12 (Convexity). A MOOP is convex if and only if all objective functions are convex and the feasible region is convex.

Convexity may lead to the failure of some algorithms in maintaining diversity. In the objective space shown in Figure 2.3, the global Pareto-optimal front is non-convex, while the local Pareto-optimal front is convex. In particular, as the feasible region is non-convex, the MOOP is non-convex.

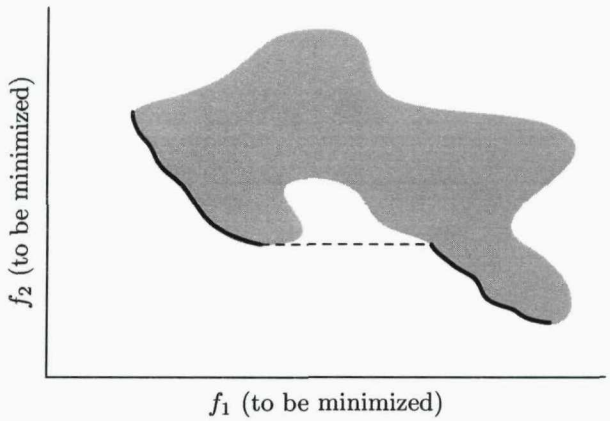


FIGURE 2.5: Discontinuous Pareto-optimal front.

2.2.8.4 Discontinuity

Definition 2.13 (Discontinuous MOOP). A MOOP is discontinuous if and only if the Pareto-optimal front is discontinuous.

An example of a discontinuous Pareto-optimal front is shown in Figure 2.5. Such fronts present difficulties to optimization algorithms as it is then the duty of the algorithm to approximate all branches of the global Pareto-optimal front.

2.2.8.5 Non-Uniformity

When the distribution of points in objective space around the Pareto-optimal front is non-uniform, algorithms will tend to converge where the density of points are higher. Thus diversity of solutions is not well maintained. Figure 2.6 illustrates an objective space which has a non-uniform Pareto-optimal front; in this case it is likely that the algorithm will favour minimizing objective f_2.

2.2.9 Multi-Objective Test Functions

As with single-objective optimization, sets of test functions, which have one or more of these difficult features, exist to allow comparison of multi-objective algorithms (using the different performance indicators mentioned in Section 2.2.7). Some of the most popular sets include the Ziztler-Deb-Thiele (ZDT) problems [48], and Veldhuizen and Lamont's (VLMOP) [49]. However, as with the Dixon-Szego test set in single-objective

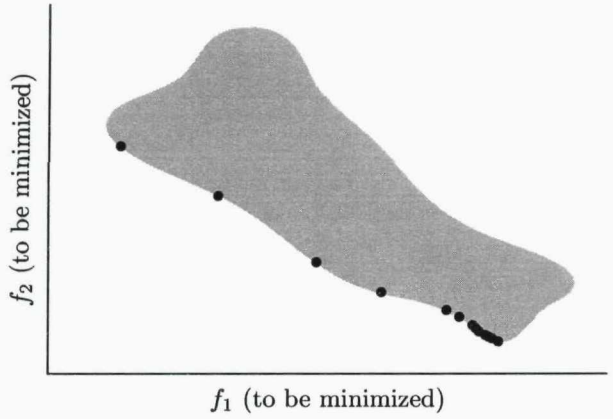


FIGURE 2.6: Non-uniformity in the Pareto-optimal front.

optimization, these have recently received some criticism for being too simple for modern multi-objective algorithms. More difficult test problems have recently been proposed [50, 51], and as with the modern single-objective test functions, they have parameters allowing their difficulty to be fine-tuned.

2.3 Cost-Effective Optimization

Optimization algorithms may be classified by many criteria: single-objective or multi-objective; constrained or unconstrained; cost-effective or 'greedy'; stochastic or deterministic; those requiring derivatives, and those which are derivative-free; and so on .... The possibility of creating hybrid algorithms, combining different classes, only complicates producing a full classification. Furthermore, the number of heuristic algorithms is constantly growing, with inspiration taken from all aspects of nature, including particle swarms [52], ant colonies [53] and artificial immune systems [54] (of particular interest to electrical engineers, an optimization algorithm based on principles from electromagnetics has even appeared [55]).

As the computational expense of objective functions in electromagnetic design is typically high, the family of cost-effective algorithms (those which attempt to minimize the number of objective function evaluations) is of particular importance. Algorithms within this family may be categorized according to how they achieve cost-effectiveness: several of the most popular techniques are illustrated in Figure 2.7. Four of the techniques, namely hybrid algorithms, small population evolutionary algorithms, fitness inheritance and reduction of design variables, are generally not versatile enough for our purposes (e. g. reduction of design variables is irrelevant for design problems with a low number of design variables, and fitness inheritance is only suitable for convex

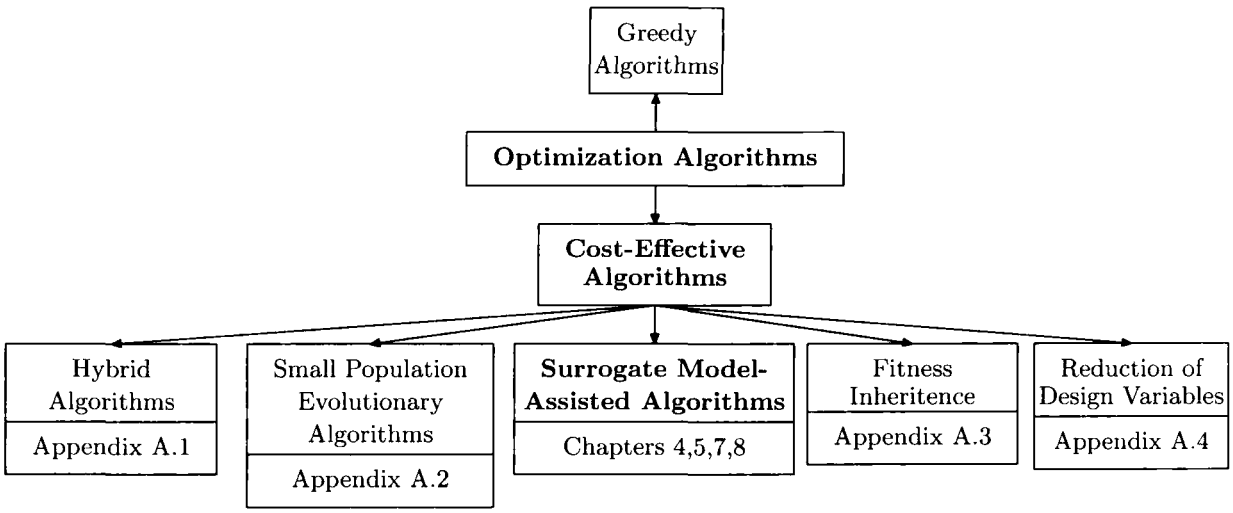


FIGURE 2.7: Types of optimization algorithm.

MOOPs [56]), and so they will not be developed further in this thesis; their discussion is reserved for Appendix A. The cost-effective technique used and developed within this work is surrogate modelling, which is introduced in the next chapter.

Chapter 3

Surrogate Modelling

3.1 Introduction

At the most general level, the problem of estimating dependencies using a set of observations is known as *machine learning* [57, 58]. This is a broad field, which encompasses the problems of *regression*, where the inputs are mapped to a continuous function, and *classification*, where the inputs are mapped to discrete categories. Due to its general nature, methods from machine learning are used in many areas of science and engineering, and so it is perhaps inevitable that along the way, different communities have adopted their own terminologies.

In optimization, the estimation of the relationship between design vector space and objective function space, based on a set of observations, has been used as a tool in lowering the number of iterations used by algorithms in the search for the optimal solutions (thus reducing computational cost). An example of regression, the technique goes by several different names within the engineering community: the three most popular are response surface modelling, meta-modelling, and surrogate modelling, the name adopted in this thesis.

This chapter begins by discussing the theory of surrogate modelling generally, using the example of polynomial fitting as an introduction. The idea of basis functions is introduced through the example of *radial basis functions*, a popular type of Artificial Neural Network used for surrogate modelling. The method known as kriging is then introduced in detail. The use of kriging surrogate models for optimization purposes is then discussed in Chapter 4 and Chapter 5.

3.2 General Theory of Surrogate Modelling

3.2.1 Example: Polynomial Models

The simplest type of surrogate model to visualize (and construct) is a polynomial model; that is, one only involving terms of the form $x_1^{g_1} x_2^{g_2} \dots x_d^{g_d}$, where $x_1, x_2, \dots, x_d$ are the components of the d-dimensional design vector $\mathbf{x}$ and where $\sum_{i=1}^{d} g_i \leq G$, G being the order of the polynomial. For example, the prediction $\hat{y}$ of a first order polynomial model is given by

$$\hat{y}(\mathbf{x}) = \beta_0 + \sum_{i=1}^{d} \beta_i x_i, \tag{3.1}$$

whilst that of a second order polynomial model is given by

$$\hat{y}(\mathbf{x}) = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \sum_{j=1}^{d} \sum_{i=1}^{d} \beta_{d-1+i+j} x_i x_j, \tag{3.2}$$

where the values of the coefficients β_i are to be determined. In general, denoting a basis of the set of all polynomials in $\mathbf{x}$ of degree G by $\{f_k(\mathbf{x}) | k = 1, 2, \dots, m\}$, the prediction made by a polynomial model may be written as

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^{m} \beta_k f_k(\mathbf{x}) \tag{3.3}$$

where the β_ks are determined by fitting the polynomial to the observed data through minimization of an *error function*, which measures the discrepancy between the prediction $\hat{y}$ (for a given β) and the observed data. One popular error function is the sum of the squares between the prediction and the observations

$$E(\beta) = \frac{1}{2} \sum_{k=1}^{N} (\hat{y}(\mathbf{x}_k) - y(\mathbf{x}_k))^2. \tag{3.4}$$

Minimization of this function is known as *least-squares fitting*. In general, interpolation can only be achieved if the number of coefficients being fit, m, is equal to the number of observations, N. This is a drawback when modelling deterministic computer experiments: because running such a computer experiment with the same inputs twice yields the same outputs, it is desirable that surrogate models approximating such experiments always be interpolating. Furthermore, even when polynomial models do interpolate the observations, this can often be at the expense of *over-fitting*: wildly oscillating polynomials are fit to the data as a result of the β_i, which minimize the error function, being very large in magnitude. Ways to alleviate this problem exist: for example, in *regularization*, a penalty term is added to the error function in order to penalize

β_is of large magnitude:

$$\tilde{E}(\beta) = \frac{1}{2}\sum_{k=1}^{N}(\hat{y}(\mathbf{x}_k) - y(\mathbf{x}_k))^2 + \frac{\lambda}{2}\|\beta\|^2. \tag{3.5}$$

This technique is common in the surrogate modelling literature, and not surprisingly, goes by different names depending on which model is being used: with artificial neural networks, it is known as *weight decay*; in statistics it is known as *shrinkage*. It should be noted that while this technique can resolve the problem of over-fitting, the problem arises of finding a suitable value for λ; furthermore, the fact remains that the model will only be interpolating for a particular number of observations.

3.2.2 Basis Functions

In order for a surrogate model to be interpolating in general, it is necessary to use some additional *basis functions*, each centered around one of the n sampled points. Then the prediction of the surrogate model may be written as

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^{m}\beta_k f_k(\mathbf{x}) + \sum_{j=1}^{n} b_j \phi(\mathbf{x} - \mathbf{x}_j). \tag{3.6}$$

Some of the choices for ϕ include:

$$\phi(r) = \exp\left(-\beta r^2\right) \quad \text{(Gaussian)} \tag{3.7}$$

$$\phi(r) = r \quad \text{(linear)} \tag{3.8}$$

$$\phi(r) = r^3 \quad \text{(cubic)} \tag{3.9}$$

$$\phi(r) = \sqrt{r^2 + \gamma^2} \quad \text{(multiquadric)} \tag{3.10}$$

$$\phi(r) = r^2 \log r \quad \text{(thin plate spline)} \tag{3.11}$$

$$\phi(r) = \exp\left(-\textstyle\sum_{l=1}^{d}\theta_l |r_l|^{p_l}\right) \quad \text{(kriging)} \tag{3.12}$$

where $r = \|\mathbf{x} - \mathbf{x}_i\|$, $\gamma > 0$ in the multiquadric case, and $\theta_l \geq 0$, $p \in (0, 2]$ in the kriging case. The first five of these belong to a class of functions known as *radial basis functions*, which are now briefly discussed.

3.2.2.1 Radial Basis Functions

In the radial-basis function approach, the approximation model is usually written as

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{n} \beta_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{3.13}$$

where $\|\cdot\|$ denotes a norm (usually the Euclidean norm). Using the interpolation conditions

$$\hat{y}(\mathbf{x_i}) = y(\mathbf{x_i}), \tag{3.14}$$

and denoting

$$\phi_{ij} = \phi(\|\mathbf{x_i} - \mathbf{x_j}\|), \tag{3.15}$$

$\mathbf{\Phi}$ as the $n \times n$ matrix with elements ϕ_{ij},

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^T, \tag{3.16}$$

and

$$\beta = [\beta_1, \beta_2, \ldots, \beta_n]^T, \tag{3.17}$$

then the coefficients β may be found by

$$\beta = \mathbf{\Phi}^{-1}\mathbf{y}. \tag{3.18}$$

The non-singularity of $\mathbf{\Phi}$ is guaranteed by Micchelli's theorem [59].

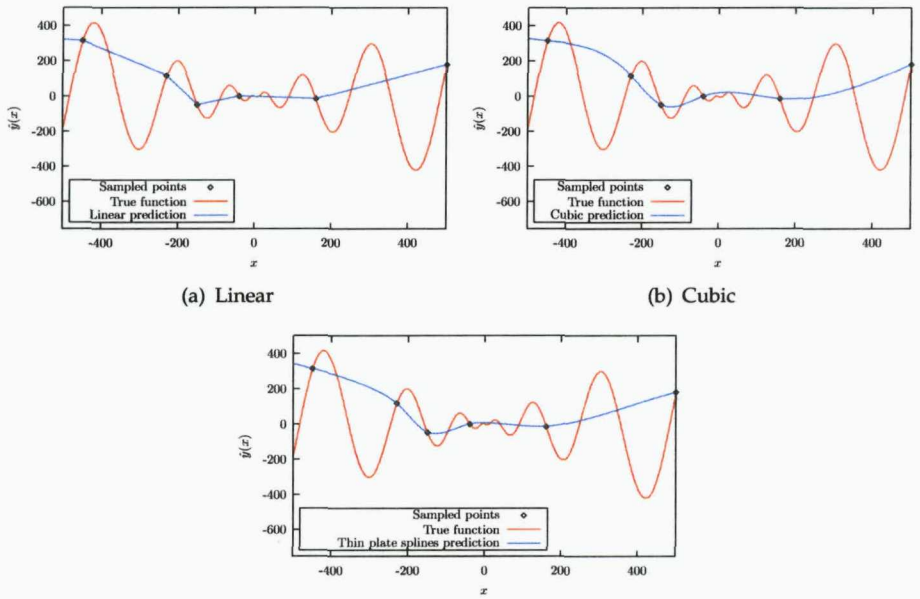
Figure 3.1 shows approximations to the Schwefel function (which will be used in the next chapter to demonstrate optimization methods) using linear, cubic and thin plate spline radial basis functions; Figure 3.2 shows approximations using the multiquadric radial basis function for different values of γ.

Review papers appear regularly comparing the merits of different types of surrogate model, see e. g. [60, 61, 62, 63] for four recent reviews. One surrogate model which stands out, due to its solid statistical foundations, is kriging, to which the remainder of this chapter is devoted.

3.3 Kriging

3.3.1 Background

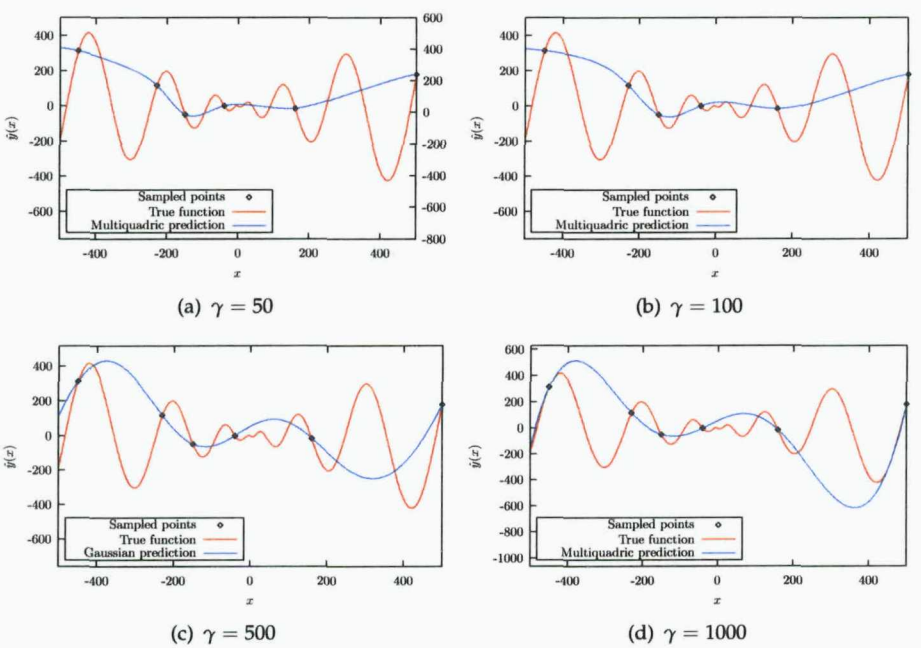
In the 1960s, geologists such as Matheron [64], developed a statistical method for making predictions, and named it kriging, after D. G. Krige who had originally used the technique to analyze mining data [65] in the 1950s. Since then it has formed the basis of the field of geostatistics, and has found applications in fields of research far removed



(a) Linear (b) Cubic

(c) Thin plate spline

FIGURE 3.1: Linear, cubic and thin plate spline radial basis function approximations to the Schwefel function



(a) $\gamma = 50$ (b) $\gamma = 100$

(c) $\gamma = 500$ (d) $\gamma = 1000$

FIGURE 3.2: Multiquadric radial basis function approximations to the Schwefel function

from geology. A version for predicting computer experiments with deterministic output, known as Design and Analysis of Computer Experiments (DACE), was developed in the late 1980s [66]. The following section derives the main DACE equations, necessary for the optimization methods which follow in later chapters.

A comprehensive treatment of kriging may be found in [67], and [68] provides further theory. The original DACE approach to kriging is given in [66], whilst a more detailed and up-to-date discussion may be found in [69]. Alternative approaches to kriging are given in [14, 70]. In the machine learning literature, the method is known as Gaussian Processes: a recent overview of this approach may be found in [71].

3.3.2 Design and Analysis of Computer Experiments

3.3.2.1 Curve Fitting: A Probabilistic Perspective

The standard linear regression model takes the form [57, 71]

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^{m} \beta_k f_k(\mathbf{x}) + \epsilon(\mathbf{x}). \tag{3.19}$$

The sum $\sum_{k=1}^{m} \beta_k f_k(\mathbf{x})$ may be viewed as a global approximation to the true function, and ϵ as an additive Gaussian noise, representing our uncertainty. By considering this model at the sampled points

$$\hat{y}(\mathbf{x}^i) = \sum_{k=1}^{m} \beta_k f_k(\mathbf{x}^i) + \epsilon(\mathbf{x}^i), \tag{3.20}$$

it can be seen that to be interpolating, the Gaussian distribution $\epsilon(\mathbf{x}^i)$ must be $\mathcal{N}(0, \sigma^2)$, where σ^2 is to be determined.

By considering two design vectors $\mathbf{x}^i, \mathbf{x}^j$, close to each other in design variable space, it can be expected that their corresponding objective function values will be similar. This is modeled statistically by saying that the errors $\epsilon(\mathbf{x}^i)$ and $\epsilon(\mathbf{x}^j)$ are *correlated*. In kriging, this correlation is modelled by the function

$$R(\epsilon(\mathbf{x}^i), \epsilon(\mathbf{x}^j)) = \prod_{k=1}^{n} e^{-\theta_k |x_k^i - x_k^j|^{p_k}}. \tag{3.21}$$

Here θ_k determines how fast the correlation between design vectors drops away in the k^{th} coordinate direction, and p_k determines the smoothness of the function in the k^{th} coordinate direction.

The kriging model may be simplified further by replacing the polynomial in Equation 3.19 by a constant term (a technique known as ordinary kriging)

$$\hat{y}(\mathbf{x}) = \mu + \epsilon(\mathbf{x}). \tag{3.22}$$

In doing so, the kriging model is then a probabilistic model which has $2d + 2$ parameters: $\mu, \sigma^2, \theta_1, \ldots, \theta_d, p_1, \ldots, p_d$. These are chosen to maximize the likelihood of the observations (essentially, the probability of the observations, given the model parameters - see [57] for an introduction), which for this model is given by

$$\frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}|\mathbf{R}|^{1/2}} \exp\left[\frac{-(\mathbf{y} - \mathbf{1}\mu)^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}\right], \tag{3.23}$$

where $\mathbf{R}$ is the correlation *matrix*, which represents the correlation between each pair of evaluated design vectors, defined as the $n \times n$ matrix whose $i - j^{\text{th}}$ entry is $R(x^i, x^j)$, $\mathbf{y}$ is the $n \times 1$ vector of observed objective function values,

$$\mathbf{y} = [y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \ldots, y(\mathbf{x}^{(n)})]^{\mathrm{T}}, \tag{3.24}$$

and $\mathbf{1}$ is the $n \times 1$ vector filled with ones. It is more common to maximize the log of Equation 3.23, which, ignoring constant terms, is

$$-\frac{n}{2}\log(\sigma^2) - \frac{1}{2}\log(|\mathbf{R}|) - \frac{(\mathbf{y} - \mathbf{1}\mu)^{T}\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}. \tag{3.25}$$

By setting the derivatives of Equation 3.25 with respect to σ^2 and μ equal to zero, it is found that

$$\hat{\mu} = \frac{\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{1}} \tag{3.26}$$

and

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})}{n}, \tag{3.27}$$

where the hats signify that these are the optimal values of μ and σ. Substituting Equation 3.26 and Equation 3.27 into Equation 3.25, the 'concentrated log-likelihood' function is formed:

$$-\frac{n}{2}\log(\hat{\sigma}^2) - \frac{1}{2}\log(|\mathbf{R}|) \tag{3.28}$$

which only depends on θ and $\mathbf{p}$. By maximizing this to find $\hat{\theta}$ and $\hat{\mathbf{p}}$, Equation 3.26 and Equation 3.27 may be used to compute $\hat{\mu}$ and $\hat{\sigma}^2$.

3.3.2.2 The Kriging Prediction Formula

Consider an unevaluated design vector $\mathbf{x}^*$. Suppose y^* is an estimate of the function value for $\mathbf{x}^*$. Now consider the 'augmented' log-likelihood function, obtained by adding $(\mathbf{x}^*, y^*)$ to the observations, and using the parameter values obtained by maximizing the likelihood (for the observations) as explained in the previous section. With the model parameters fixed, the augmented log-likelihood function is a function of y^*. As y^* varies, its value is a measure of how consistent the estimate $(\mathbf{x}^*, y^*)$ is with the description of variation, as determined by the observed points. Intuitively, a good prediction for the function is the value of y^* which maximizes the augmented log-likelihood: this indeed is the kriging prediction.

To derive the expression for $\hat{y}$, it is necessary to set the derivative of the augmented log-likelihood with respect to y^* equal to zero, and rearrange for y^*. First denote by $\tilde{\mathbf{y}} = (\mathbf{y}^T, y^*)^T$ the vector of function values composed of the observed values, and the estimated value $\hat{y}$, and further denote by $\mathbf{r}$ the correlation vector, which expresses the correlation between an unevaluated design vector $\mathbf{x}$ and the n evaluated design vectors, defined as

$$\mathbf{r}(\mathbf{x}) = [R(\mathbf{x}, \mathbf{x}^{(1)}), R(\mathbf{x}, \mathbf{x}^{(2)}), \ldots, R(\mathbf{x}, \mathbf{x}^{(n)})]^{\mathrm{T}}. \tag{3.29}$$

Then, writing the correlation matrix for the augmented set as $\tilde{\mathbf{R}}$

$$\tilde{\mathbf{R}} = \begin{pmatrix} \mathbf{R} & \mathbf{r} \\ \mathbf{r}^T & 1 \end{pmatrix} \tag{3.30}$$

the part of the augmented likelihood which is dependent on y^* is

$$-\frac{(\tilde{\mathbf{y}} - \mathbf{1}\hat{\mu})^T \tilde{\mathbf{R}}^{-1} (\tilde{\mathbf{y}} - \mathbf{1}\hat{\mu})}{2\hat{\sigma}^2}. \tag{3.31}$$

Substituting in for $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{R}}$, this becomes

$$-\frac{\begin{pmatrix} \mathbf{y} - \mathbf{1}\hat{\mu} \\ y^* - \hat{\mu} \end{pmatrix}^T \begin{pmatrix} \mathbf{R} & \mathbf{r} \\ \mathbf{r}^T & 1 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{y} - \mathbf{1}\hat{\mu} \\ y^* - \hat{\mu} \end{pmatrix}}{2\hat{\sigma}^2}. \tag{3.32}$$

Using the following identity for the inverse of a partitioned matrix

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{M} & -\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{M} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \end{pmatrix} \tag{3.33}$$

where

$$\mathbf{M} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}^{-1})^{-1}, \tag{3.34}$$

the matrix $\tilde{\mathbf{R}}^{-1}$ may be written as

$$\begin{pmatrix} \mathbf{R}^{-1} + \mathbf{R}^{-1}\mathbf{r}(1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})^{-1}\mathbf{r}^T\mathbf{R}^{-1} & -\mathbf{R}^{-1}\mathbf{r}(1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})^{-1} \\ -(1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})^{-1}\mathbf{r}^T\mathbf{R}^{-1} & (1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})^{-1} \end{pmatrix}. \tag{3.35}$$

Substituting into Equation 3.32, the terms of the augmented log-likelihood depending on y^* are

$$\left[\frac{-1}{2\hat{\sigma}^2(1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})}\right](y^* - \hat{\mu})^2 + \left[\frac{\mathbf{r}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})}{\hat{\sigma}^2(1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})}\right](y^* - \hat{\mu}). \tag{3.36}$$

As said at the outset, the expression for y^* is found by setting the derivative of the augmented likelihood with respect to y^* equal to zero, i. e. by solving

$$\left[\frac{-1}{\hat{\sigma}^2(1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})}\right](y^* - \hat{\mu}) + \left[\frac{\mathbf{r}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})}{\hat{\sigma}^2(1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})}\right] = 0. \tag{3.37}$$

This gives

$$\hat{y}(\mathbf{x}^*) = \hat{\mu} + \mathbf{r}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}), \tag{3.38}$$

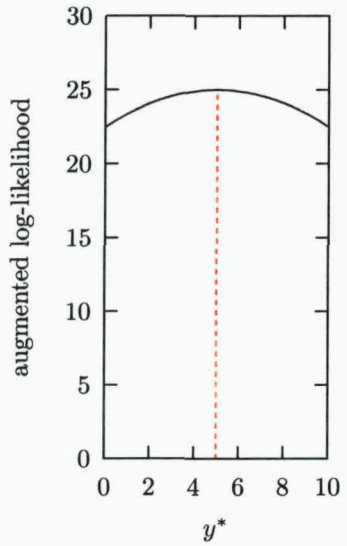
which is the general expression for the kriging predictor.

As a final note, it should be pointed out that Equation 3.38 is consistent with the general surrogate model equation, Equation 3.6. By letting the polynomial terms equal $\hat{\mu}$, and b_i be the i^{th}element of $\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})$, the kriging predictor becomes a linear combination of polynomial terms (a constant), and a set of basis functions.

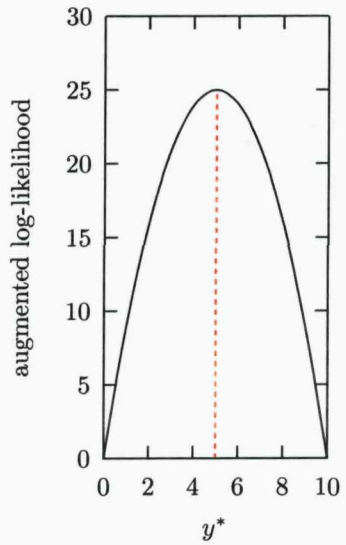
3.3.2.3 The Standard Error Formula

As shall be made clear in the following chapters, an estimate of the potential error in the kriging prediction is extremely useful when using the model to decide where to evaluate next in design variable space.

From the derivation of the kriging prediction above, intuitively the less consistent the prediction is for other values of y^* (around the value predicted by the kriging model), as measured by the value of the augmented log-likelihood, the higher our confidence in the prediction y^* should be. This is illustrated schematically in Figure 3.3. Two possible augmented log-likelihood functions are plotted for the optimal value of y^*. In Figure 3.3(a), values around the optimal value of y^* also give high values for the augmented log-likelihood, and so our confidence in value of y^* is reduced. In Figure 3.3(b) however, the augmented log-likelihood drops off rapidly away from the optimal value of y^*. Thus, non-optimal values of y^* are much less consistent with the observations,



(a) Shallow variation of the augmented log-likelihood function around the (optimal) predicted value corresponds to a low confidence in the prediction.



(b) Rapid variation of the augmented log-likelihood function around the (optimal) predicted value corresponds to a high confidence in the prediction.

FIGURE 3.3: Confidence in a kriging prediction may be related to the variation of the augmented log-likelihood function.

and so our confidence in the kriging prediction (which maximizes the augmented log-likelihood) is relatively high.

Mathematically, the curvature of the augmented log-likelihood function around the optimal value of y^* (i. e. the kriging prediction), is inversely related to our estimate of the potential error in the prediction. The curvature is measured by the absolute value of the second derivative of the augmented log-likelihood function with respect to y^*:

$$\frac{1}{\hat{\sigma}^2(1-\mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})}. \tag{3.39}$$

Therefore, using the intuitive argument above, an estimate of the potential error in the kriging prediction is given by

$$\hat{\sigma}^2(1-\mathbf{r}^T\mathbf{R}^{-1}\mathbf{r}). \tag{3.40}$$

The full derivation of the mean-squared error $s^2(\mathbf{x})$ (which may be found in, e. g. [66]), is given by

$$s^2(\mathbf{x}) = \hat{\sigma}^2\left[1-\mathbf{r}^T\mathbf{R}^{-1}\mathbf{r}+\frac{(1-\mathbf{r}^T\mathbf{R}^{-1}\mathbf{r})}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}\right], \tag{3.41}$$

where the extra term is interpreted as representing an additional uncertainty which results from having to estimate the value of $\hat{\mu}$ (as opposed to knowing it exactly).

The notion of the mean squared error, and its square root, the 'standard error', $s(\mathbf{x})$, is useful in quantifying uncertainty in kriging predictions, as shall be made clear in the following chapter.

Chapter 4

Kriging-Assisted Single-Objective Optimization

4.1 Introduction

In the previous chapter, kriging surrogate models were introduced as a method of estimating an unknown function, based on a set of observations. The purpose of this chapter is to review how such models may be used to select design vectors for evaluating in order to achieve the goal of single-objective optimization, that is, to locate the global minimum of a single unknown function in as few objective function evaluations as possible. Complications such as handling of constraints and choice of initial experimental design, common to both single and multi-objective optimization, are left to Chapter 6.

4.1.1 Jones' Taxonomy

Jones' taxonomy of surrogate model based (single-objective) optimization methods [14], reproduced in Table 4.1, uses two main criteria to classify methods: the kind of surrogate model used, and the method of selecting search points. These two main criteria are then further subdivided: surrogate models are divided into those which interpolate the observed points and those which do not, and selection methods are divided into two-stage and one-stage varieties. Two-stage selection methods are so called because each iteration involves two main stages: fitting the surrogate model to the observed points, and then constructing a utility function based on this surrogate model to determine the next search point. One-stage methods are so called because each iteration involves only one-stage: determining the design vector which, if it had a value f^* (which is an

TABLE 4.1: Jones' taxonomy of surrogate model-assisted optimization methods, taken from [14].

Kind of Surrogate Model			Method for selecting search points					
			Two-stage approach				One-stage approach	
			Minimize the Response Surface	Minimize a Lower Bounding Function	Maximize the Probability of Improvement	Maximize Expected Improvement	Goal seeking: find point that achieves a given target	Optimization: find point that minimizes an objective
Not interpolating (smoothing)	Quadratic polynomials and other regression models		Not discussed					
Interpolating	Fixed basis functions. NO statistics.	Thin-plate splines, Hardy multi-quadrics	Section 4.2.1				Section 4.4.2.1	Section 4.4.2.2
	Tuned basis functions. Statistical Interpretation.	Kriging		Not Discussed	Section 4.4.1.1	Section 4.2.2		

estimate of the minimum) would yield the most credible response surface interpolating it and the already observed design vectors. Almost all surrogate-model assisted optimization algorithms used in the literature are two-stage, however one-stage algorithms have been successfully constructed using both kriging [14] and radial basis function surrogate models [72, 73, 74].

Although extremely useful, the taxonomy in [14] is too broad for the purposes of this thesis. In particular, this thesis only considers kriging models. Furthermore, the taxonomy does not address some practical issues: methods are not classified according to how 'tunable' a selection method is, that is whether they have parameters which can (and must) be set to control the balance between exploration and exploitation; nor are they classified by how capable each method is of selecting multiple points for evaluation each iteration, something which is extremely important if parallelization is to be taken advantage of. For this reason, a new kriging-only taxonomy, based on more practical criteria, is now proposed.

4.1.2 Taxonomy of Kriging-Assisted Single-Objective Optimization Methods

The alternative taxonomy, for kriging-assisted methods only, is shown in Table 4.2. The two main criteria used to classify the methods are the number of design vectors which are to be evaluated each iteration, and how 'tunable' each method is to the balance between exploration and exploitation. Methods are divided at the broadest level between those which are tunable, that is those which do allow control over this balance through a changeable parameter, and those which allow no control, and so are non-tunable; tunable methods are then further divided into those which require an estimate of the minimum, and those which do not.

TABLE 4.2: Taxonomy of kriging model-assisted single-objective optimization methods.

Number of Points to Evaluate per Iteration	**Method for selecting search points**					
	Non-tunable Utility Functions: No parameters to set		**Tunable Utility Functions**: Parameters allowing the balance between exploration and exploitation to be altered			
			Non-target based: does not require an estimate of the optimum		Target-based: requires an estimate f^* of the optimum	
	Surface Minimum	Expected Improvement	Generalized Expected Improvement	Weighted Expected Improvement	Probability (of achieving)	Credibility (of hypothesis)
Single Point	Section 4.2.1	Section 4.2.2.1	Section 4.3.1.1	Section 4.3.2.1	Section 4.4.1.1	Section 4.4.2.1
Multiple Points		Section 4.2.2.2	Section 4.3.1.2	Section 4.3.2.2	Section 4.4.1.2	Section 4.4.2.2

In this chapter, each utility function in Table 4.2 will be demonstrated on the multimodal Schwefel test function [75]:

$$f(\mathbf{x}) = \sum_{i=1}^{d} -x_i \sin(\sqrt{|x_i|}) \tag{4.1}$$

To allow plotting of the kriging surfaces, and utility functions, the one-dimensional case ($d = 1$) is used:

$$f(x) = -x \sin(\sqrt{|x|}). \tag{4.2}$$

The six starting design vectors $x = -450, -230, -150, -40, 160, 500$ are used to initialize each approach.

4.2 Non-tunable Utility Functions

Non-tunable utility functions are characterized by having no parameters which need to be defined in order to select a point to evaluate. Two such methods are now discussed, namely minimizing the constructed response surface, and maximizing the expectation value of the improvement.

4.2.1 Minimum of Response Surface

The most natural (and naïve) approach to selecting where to evaluate, based on a surrogate model, is to simply interpret the surrogate model as an accurate representation of the true function, and to evaluate the minimum of the surface. This approach has been used by practitioners of optimization for many years, however the approach is fraught with potential failure modes, even for kriging surrogate models.

Typically, the point chosen to be evaluated will be a *false minimum* - a point which is a minimum of the surrogate model but not of the true function. Even worse, if the false minimum is actually an observed design vector, that is, one which has been used to

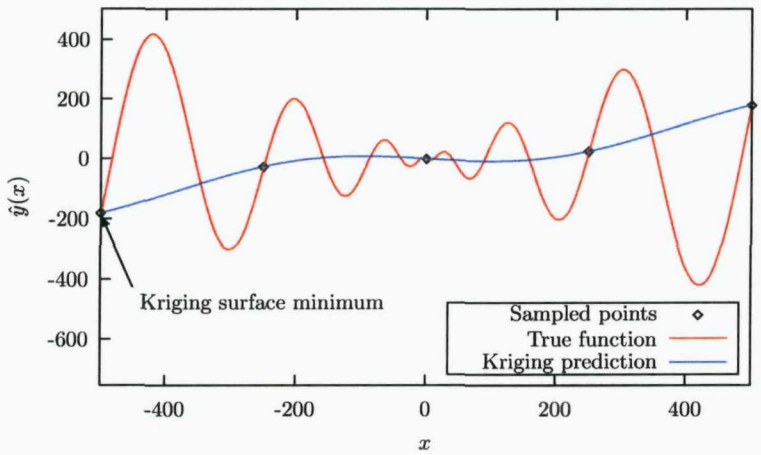
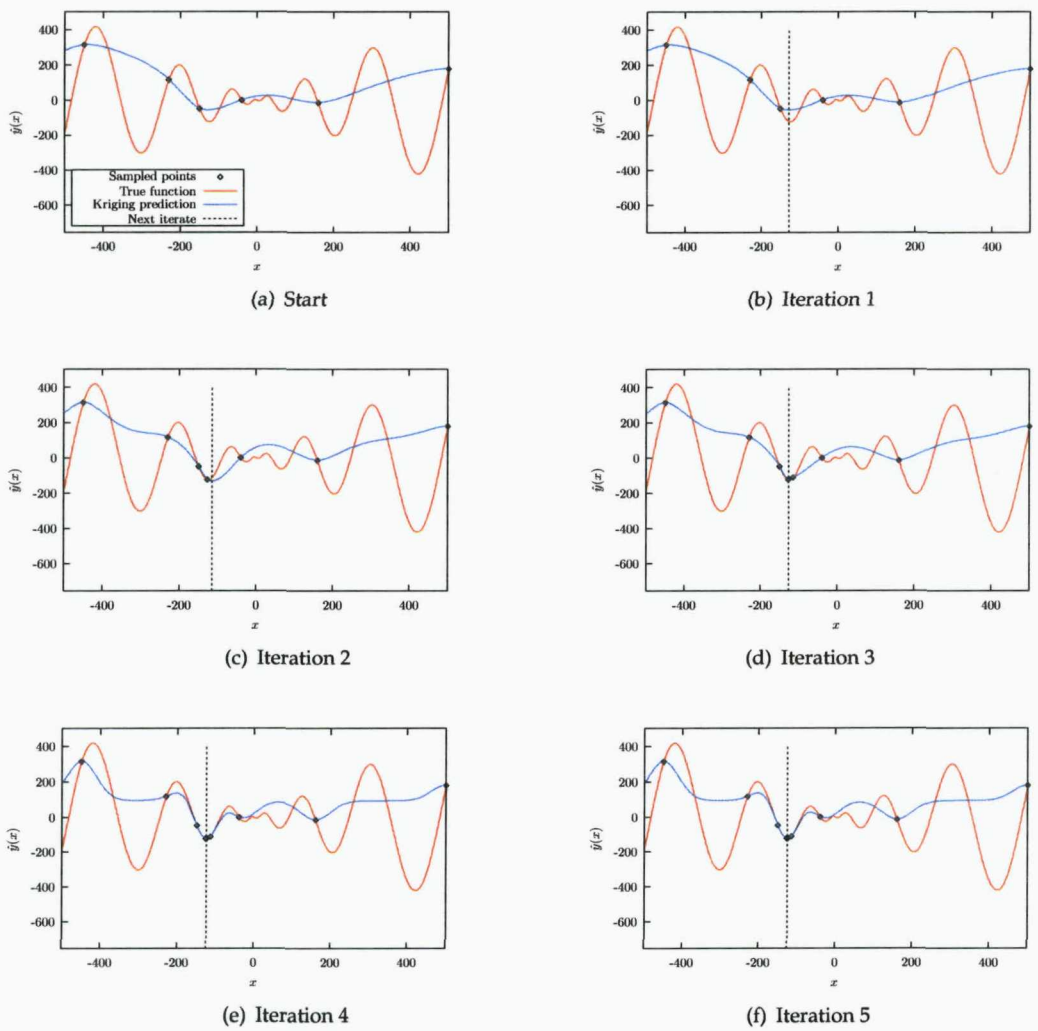


FIGURE 4.1: Minimum of the surrogate model has already been sampled.



(a) Start

(b) Iteration 1

(c) Iteration 2

(d) Iteration 3

(e) Iteration 4

(f) Iteration 5

FIGURE 4.2: Iterations of the 'strawman' approach using a kriging model on the Schwefel test function.

construct the kriging model, as shown in Figure 4.1, then an algorithm using this approach will chose to evaluate a design vector which has already been evaluated. Not only is sampling an already evaluated design vector clearly of no use when using deterministic computer code to evaluate the objective function (as the output will be exactly the same as before), using the same solution twice in the construction of a surrogate model leads to the inversion of a singular matrix. At this point, this approach fails.

Even if the minimum of the kriging model is not an observed point, but instead close (in the Euclidean sense) to one, then the resulting change in the kriging model during reconstruction in the first stage of the subsequent iteration may be insignificant, meaning that many iterations are then wasted sampling around a false optimum. For an example of this occurring with a kriging model on the Schwefel test function, see Figure 4.2. (In this case, the false optimum is actually a local optimum of the true function. However it is still undesirable behaviour for an algorithm to have no other option other than terminate at a given iteration, especially when the global optimum has not even been located.)

Clearly this 'strawman approach' (as dubbed by Jones in [14]) is fraught with potential failure. Its main problem is that it does not take into account how unexplored a region of design variable space is when deciding where to evaluate: it simply exploits the most promising region of design variable space. Furthermore, there is no obvious way to select multiple points using this method: unless several global minima exist in the surrogate model, only one design vector can be chosen.

4.2.2 Expected Improvement: The EGO Algorithm

From Chapter 3 the prediction made by a kriging surrogate model may be viewed as the realization of a Gaussian process Y. This allows for the concept of improvement to be defined: for a *single-objective to be minimized*, the *improvement* may be measured by how far the value realized by the objective function is below the current minimum, and so may be written as [70]:

$$I(\mathbf{x}) = \max(f_{\min} - Y(\mathbf{x}), 0). \tag{4.3}$$

Now the expectation value of the improvement (commonly referred to as the *expected improvement*) may be found by integrating it over the likelihood of achieving it (given by the normal density function):

$$E[I(\mathbf{x})] = \int_0^\infty I \left\{ \frac{1}{\sqrt{2\pi}s(\mathbf{x})} \exp\left[-\frac{(f_{\min} - I - \hat{y}(\mathbf{x}))^2}{2s^2(\mathbf{x})} \right] \right\} \mathrm{d}I \tag{4.4}$$

which turns out to be [70]:

$$E[I(\mathbf{x})] = \begin{cases} (f_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{\min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{f_{\min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) & \text{if } s(\mathbf{x}) > 0 \\ 0 & \text{if } s(\mathbf{x}) = 0 \end{cases} \tag{4.5}$$

where $\hat{y}(\mathbf{x})$ is the objective function value of $\mathbf{x}$ as predicted by the kriging model, given by Equation 3.38, $s(\mathbf{x})$ is the *root mean squared error* in this prediction, given by Equation 3.41, and ϕ and Φ are the standard Gaussian density and distribution functions respectively.

The expected improvement function may be viewed as a *fixed* compromise between exploration and exploitation: the first term in Equation 4.5 is large when the kriging prediction is lower than the current minimum, and so favours searching promising regions of design variable space, whilst the second term is large when the standard error is large, and so favours searching regions with a high uncertainty in their values. In particular, note that it does not have any parameters which need to be set, hence its classification as a 'non-tunable' utility function.

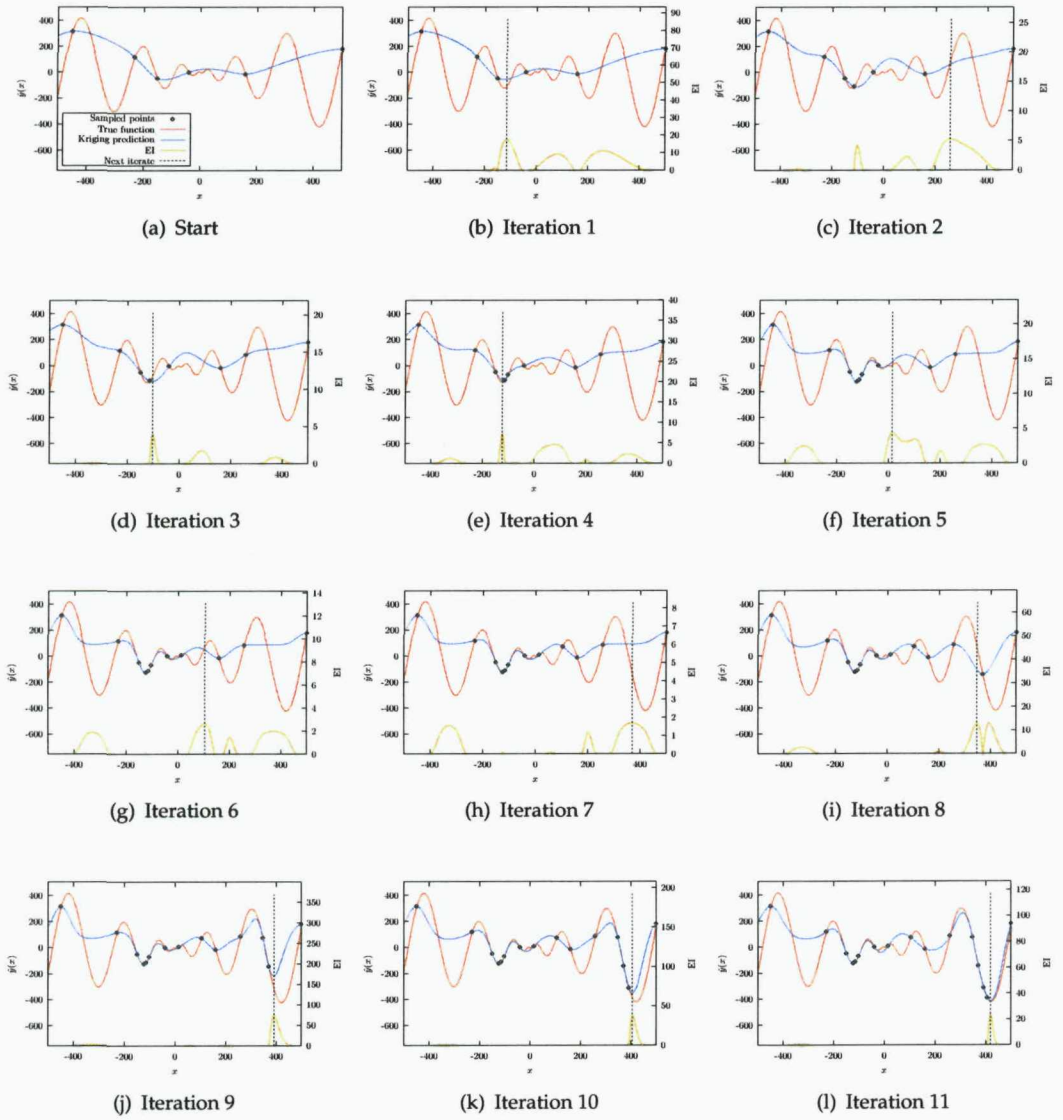
4.2.2.1 Single Point

The Efficient Global Optimization (EGO) algorithm [70] uses the expected improvement in Equation 4.5 as a utility function to determine the next point to evaluate from a kriging model. Results on standard 1d test functions show it to be extremely effective, and convergence to the global minimum is guaranteed [76]. Its impressive performance is recognized not only through significant attention in the literature, but also its inclusion in the TOMLAB optimization environment [1] as one of the methods chosen to optimize SOOPs with a computationally expensive objective function. Recently, it has been proven that, under mild assumptions, the EGO algorithm converges to the global minimum [78]. It is demonstrated on the Schwefel test function in Figure 4.3: the global minimum is located in just 11 iterations.

4.2.2.2 Multiple Points

In [79], the natural multi-modality of the expected improvement function was used to select multiple design vectors for evaluation each iteration, as parallelization was

[1] TOMLAB is one of the most powerful pieces of commercial optimization software available. The other algorithm included for computationally expensive SOOPs is rbfsolve [73]. A third algorithm is to be added in the future, based on extended radial basis functions [77].



(a) Start (b) Iteration 1 (c) Iteration 2

(d) Iteration 3 (e) Iteration 4 (f) Iteration 5

(g) Iteration 6 (h) Iteration 7 (i) Iteration 8

(j) Iteration 9 (k) Iteration 10 (l) Iteration 11

FIGURE 4.3: Iterations of the expected improvement approach on the Schwefel test function.

taken advantage of. Given N_p processors, the best N_p local maxima of Equation 4.5 were selected for evaluation.

4.3 Non-Target Based Tunable Utility Functions

Non-target based tunable utility functions are characterized by allowing the balance between exploration and exploitation to be controlled by a parameter, which is not an estimate of the global minimum. Two simple utility functions exist which are of

TABLE 4.3: Values of g, as used in a cooling scheme in [15].

Iteration	g	Iteration	g
1-4	20	20-24	2
5-9	10	25-34	1
10-19	5	≥ 35	0

this type (both based on the expected improvement utility function): the generalized expected improvement, and the weighted expected improvement.

4.3.1 Generalized Expected Improvement

4.3.1.1 Single Point

By defining the generalized improvement as

$$I^g(\mathbf{x}) = \max\{(f_{\min} - Y(\mathbf{x}))^g, 0\}, \tag{4.6}$$

where g is an integer, the expectation value of this, known as the Generalized Expected Improvement (GEI), may be shown to be [15]

$$\mathrm{GE}(I^g(\mathbf{x})) = s^g \sum_{k=0}^{g} (-1)^k \left(\frac{g!}{k!(g-k)!}\right) \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)^{g-k} T_k \tag{4.7}$$

where

$$T_k = -\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)^{k-1} + (k-1)T_{k-2} \tag{4.8}$$

with

$$T_0 = -\Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \tag{4.9}$$

$$T_1 = -\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right). \tag{4.10}$$

The higher the value of the integer g, the greater the level of improvement is being sought, and consequently the more emphasis is placed on searching regions of high uncertainty. For the case of $g = 1$, Equation 4.7 reduces to the expression for EI.

In practice, a *cooling* scheme, or *cyclic* scheme is used to control the balance between exploration and exploitation during an optimization search. For example, in [15], the cooling scheme in Table 4.3 is used. The GEI criterion (normalized to 1) for several values of g is shown for the first iteration of the Schwefel function in Figure 4.4.

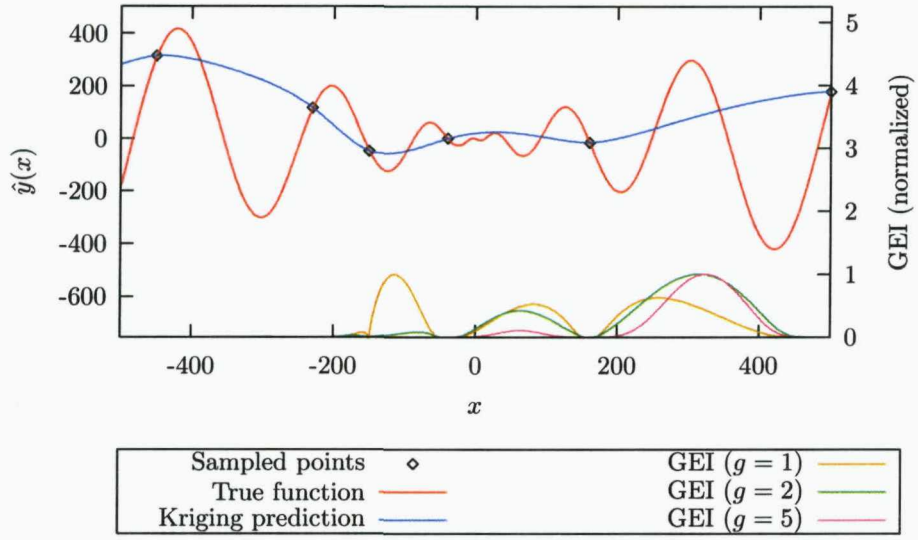


FIGURE 4.4: Generalized Expected Improvement for different values of g

4.3.1.2 Multiple Points

By using several values of g per iteration, several points may be selected for evaluation each iteration. In practice, these points usually group together, and so only one design vector from each group is chosen for evaluation (the design vector associated with the largest g should be chosen, to ensure the search is more global). This has not been pursued in the literature, but is a natural method for selecting multiple points to evaluate each iteration.

4.3.2 Weighted Expected Improvement

4.3.2.1 Single Point

GEI is useful for placing a greater amount of emphasis on searching regions of high uncertainty than is given by EI, but not so useful for placing greater amounts of emphasis on searching around the current minimum, as it has only one setting which allows this ($g = 0$). Recalling the meaning of the two terms in the expression for the EI, it is obvious to see how the Weighted Expected Improvement (WEI) utility function [80]

$$WE[I(\mathbf{x})] = \begin{cases} w(f_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + (1-w)s(\mathbf{x})\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) & \text{if } s(\mathbf{x}) > 0 \\ 0 & \text{if } s(\mathbf{x}) = 0 \end{cases} \tag{4.11}$$

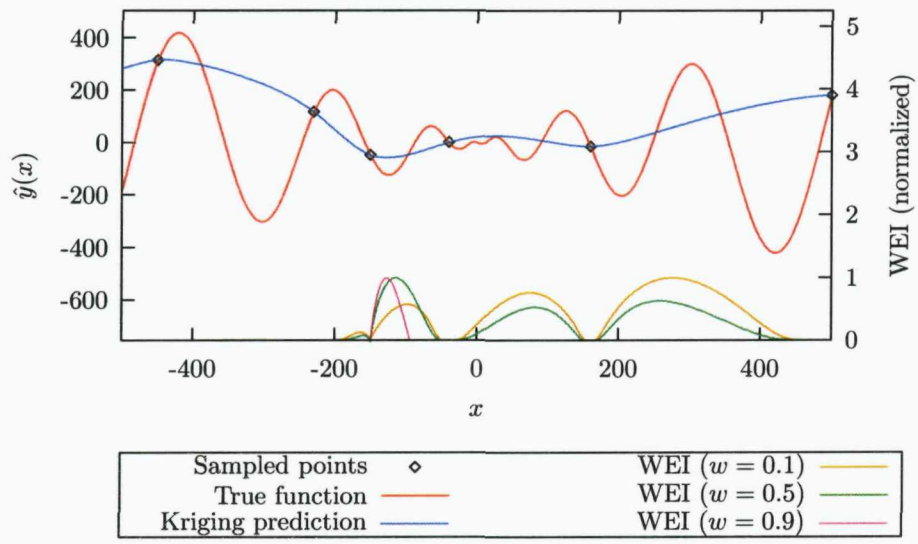


FIGURE 4.5: Weighted Expected Improvement for different values of w

provides this level of control through a weighting parameter w. Notable values of w are $w = 1$, which places all the emphasis on the first term (thus heavily favouring exploitation); $w = 0$, which places all the emphasis on the second term (thus heavily favouring exploration); and $w = 0.5$, which is equivalent to EI (subject to a scale factor of 0.5). Again, in practice, a *cooling* scheme or *cyclic* scheme is used to vary the utility function parameter w. For example, in [80], the value of w is cycled through the pattern $w = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The WEI criterion (normalized to 1) is shown for several values of w for the first iteration of the Schwefel function in Figure 4.5.

4.3.2.2 Multiple Points

As with GEI, WEI may be used to select multiple points by simply grouping maximizers of WEI together for several different values of w, and choosing one design vector from each group to evaluate each iteration. Again, this has not been pursued in the literature, but is a natural method for selecting multiple points to evaluate each iteration.

4.4 Target Based Tunable Utility Functions

Target based tunable utility functions are characterized by allowing the balance between exploration and exploitation to be controlled by a parameter f^*, which is an estimate of the global minimum. Again, the concept of improvement is used: first, simply evaluating the design vector which maximizes the probability of achieving at least a

particular level of improvement $f_{\min} - f^*(> 0)$; secondly, evaluating the design vector which, if it had an objective function value of $f^* < f_{\min}$, would yield the most credible surface. These utility functions may easily be used to select multiple design vectors to evaluate, by using different estimates of the global minimum f^*. In particular, values of f^* close to the current minimum $f_{\min}$ yield local searches, whilst values of $f^* \ll f_{\min}$ yield more exploratory searches.

4.4.1 Probability of Improvement

4.4.1.1 Single Target

Recall that, in the case of minimization, an improvement is said to occur if $\mathbf{x}$ has an objective function value less than the current minimum observation $f_{\min}$. It has already been seen how the expectation value of the improvement for unevaluated design vectors was used as a utility function to determine which design vector should be evaluated next. This section introduces instead the probability of achieving a certain level of improvement.

Setting $f^* < f_{\min}$, the probability of $\mathbf{x}$ having an objective function less than f^* may be calculated by integrating the area under the curve given by the random process Y. In the case of kriging, where Y is a normal distribution, this area is [81]:

$$\Phi\left(\frac{f^* - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \tag{4.12}$$

where Φ is the normal cumulative distribution function. This is illustrated in Figure 4.6, where the shaded area represents the probability of x^* having an objective function value $T = f* < f_{\min}$.

Maximizing this *probability of improvement* proves to be very effective in balancing searching around the current optimum, and searching in unexplored areas of objective space. However it has one drawback, in that it is very sensitive to the value of f^*: for f^* which is too close in value to $f_{\min}$, the search is excessively local; for f^* which is set too small, the search is excessively global, as can be seen in Figure 4.7 for three levels of improvement set using the equation

$$f^* = s_{\min} - \alpha(f_{\max} - f_{\min}) \tag{4.13}$$

with three values of α (lower values of α correspond to less ambitious targets).

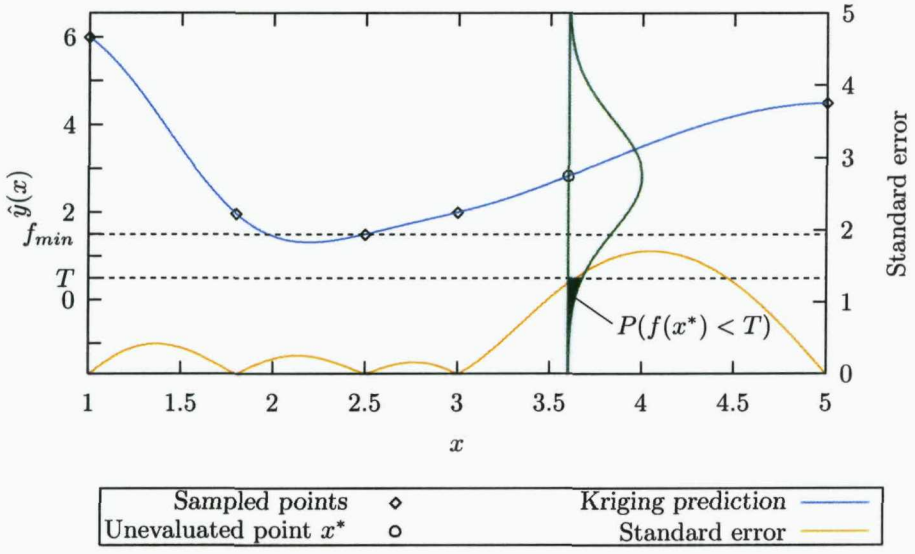


FIGURE 4.6: Modeling the prediction made by a kriging model as the realization of a Gaussian distribution, with mean and standard error as predicted by the kriging model.

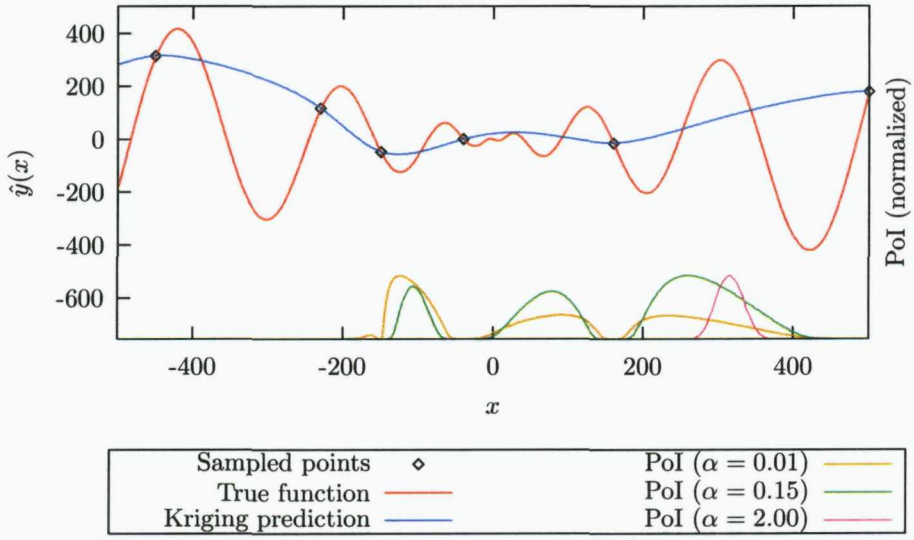


FIGURE 4.7: Probability of Improvement for different values of α

TABLE 4.4: Values of α used in the enhanced probability of improvement approach, as recommended in [14].

Target Number	α	Target Number	α	Target Number	α
1	0.0	10	0.07	19	0.25
2	0.0001	11	0.08	20	0.30
3	0.001	12	0.09	21	0.40
4	0.01	13	0.10	22	0.50
5	0.02	14	0.11	23	0.75
6	0.03	15	0.12	24	1.00
7	0.04	16	0.13	25	1.50
8	0.05	17	0.15	26	2.00
9	0.06	18	0.20	27	3.00

A way of overcoming this sensitivity problem has already been seen: maximizing the *expected improvement*, rather than the probability of improvement. However, a more natural way, and the way recommended in [14], is to simply evaluate multiple design vectors each iteration, by using different levels of improvement.

4.4.1.2 Multiple Targets

A way of varying f^* is suggested in [14]. At each iteration, 27 values of f^* were evaluated, using formula Equation 4.13 with the 27 values of α given in Table 4.4. The maximizers of the 27 different utility functions tend to group together, as illustrated in Figure 4.8 for the first iteration on the Schwefel function. Obviously evaluating so many design vectors so close to one another is not sensible practice, and so only one design vector is chosen from each group for evaluation (the one corresponding to the highest target number in Table 4.4, as this encourages more global searching). Figure 4.9 shows the iterations of this method on the Schwefel function. This method tends to be very robust, and is one of the approaches identified in [14] for much potential.

4.4.2 Credibility of Hypothesis

One-stage algorithms were first proposed by Jones [82], and introduced to the literature in [14]. As their name suggests, each iteration of a one-stage algorithm involves only one-stage: locating the design vector which, if it had a given objective function value f^* (set to be an estimate of the true minimum), would yield the most credible response surface, given the design vectors already observed. Intuitively, the 'credibility' of a surface may be seen as related to its smoothness, with smoother surfaces being deemed

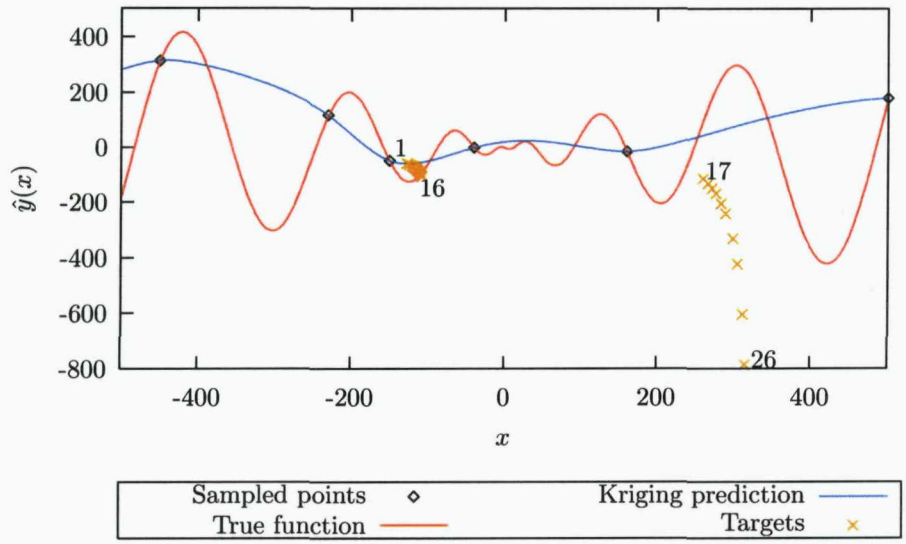


FIGURE 4.8: Location of targets which maximize their respective probability of improvement.

more credible. The method has been implemented successfully using a radial basis function surrogate model, in an algorithm known as *rbfsolve* [72, 73, 74]. This section discusses its implementation using kriging surrogate models. Two variations exist: goal-seeking, and optimization.

4.4.2.1 Single Target

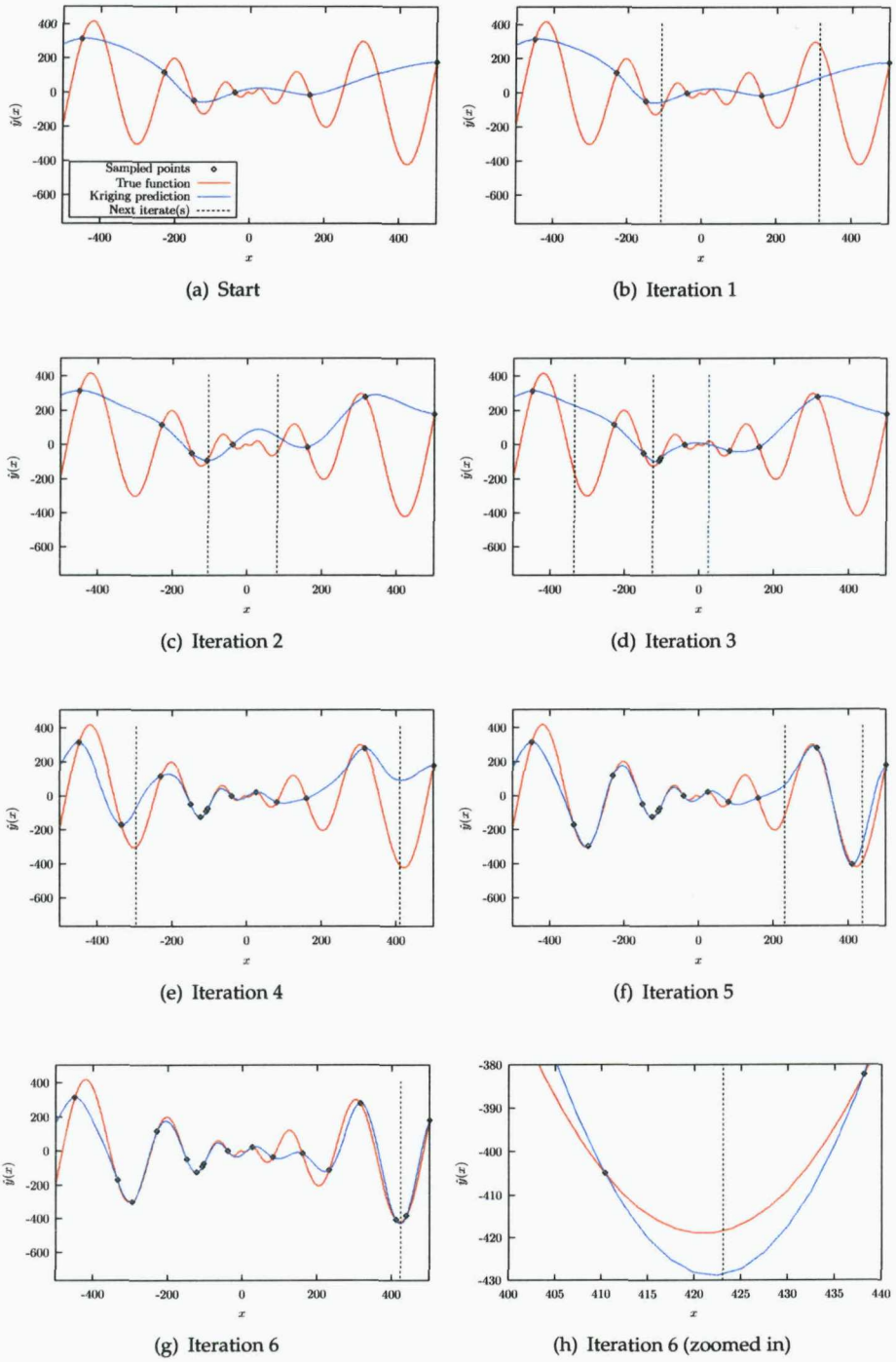
In the goal-seeking one-stage method, a target f^* is set (if the true value of the objective function global minimum were known, f^* would be set to this). The purpose of the algorithm is then to determine the design vector $\mathbf{x}^*$, which, if it had objective function value f^*, would yield the most credible response surface interpolating it and the examples already observed. In kriging, the credibility is given by the conditional likelihood, that is, the likelihood of the examples already observed conditional on the surface passing through the hypothesized point $(\mathbf{x}^*, f^*)$, which is given by [14]:

$$\frac{1}{(2\pi)^{N/2}(\sigma^2)^{N/2}|\mathbf{C}|^{1/2}} \exp\left(\frac{-(\mathbf{y}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{y}-\mathbf{m})}{2\sigma^2}\right) \tag{4.14}$$

where

$$\mathbf{m} = \mathbf{1}\mu + \mathbf{r}(y^* - \mu) \tag{4.15}$$

$$\mathbf{C} = \mathbf{R} - \mathbf{r}\mathbf{r}^T \tag{4.16}$$



(a) Start (b) Iteration 1

(c) Iteration 2 (d) Iteration 3

(e) Iteration 4 (f) Iteration 5

(g) Iteration 6 (h) Iteration 6 (zoomed in)

FIGURE 4.9: Iterations of the enhanced probability of improvement approach on the Schwefel test function.

are the conditional mean and conditional correlation matrix respectively. The next design vector to be evaluated is the one which maximizes the conditional likelihood in Equation 4.14 (which itself is maximized over θ, $\mathbf{p}$, μ and σ^2 for each $\mathbf{x}^*$). Note that setting the derivatives of Equation 4.14 with respect to σ^2 and μ equal to zero and rearranging, it is found that:

$$\sigma^2 = \frac{(\mathbf{y}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{y}-\mathbf{m})}{n} \tag{4.17}$$

$$\mu = \frac{\mathbf{1}^{\mathrm{T}}\mathbf{C}\mathbf{y} + f^*\mathbf{r}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{r} - \mathbf{y}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{r} - f^*\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{r}}{\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{1} - 2\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{r} + \mathbf{r}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{r}} \tag{4.18}$$

Substituting these two expressions into Equation 4.14 and taking logarithms, the conditional log-likelihood (ignoring constant terms) is

$$-\frac{n\log\hat{\sigma}^2 + \log|\mathbf{C}|}{2} \tag{4.19}$$

In practice, it is this expression which is maximized over θ and $\mathbf{p}$ for each $\mathbf{x}^*$; the $\mathbf{x}^*$ with the maximum conditional log-likelihood is then chosen to be evaluated.

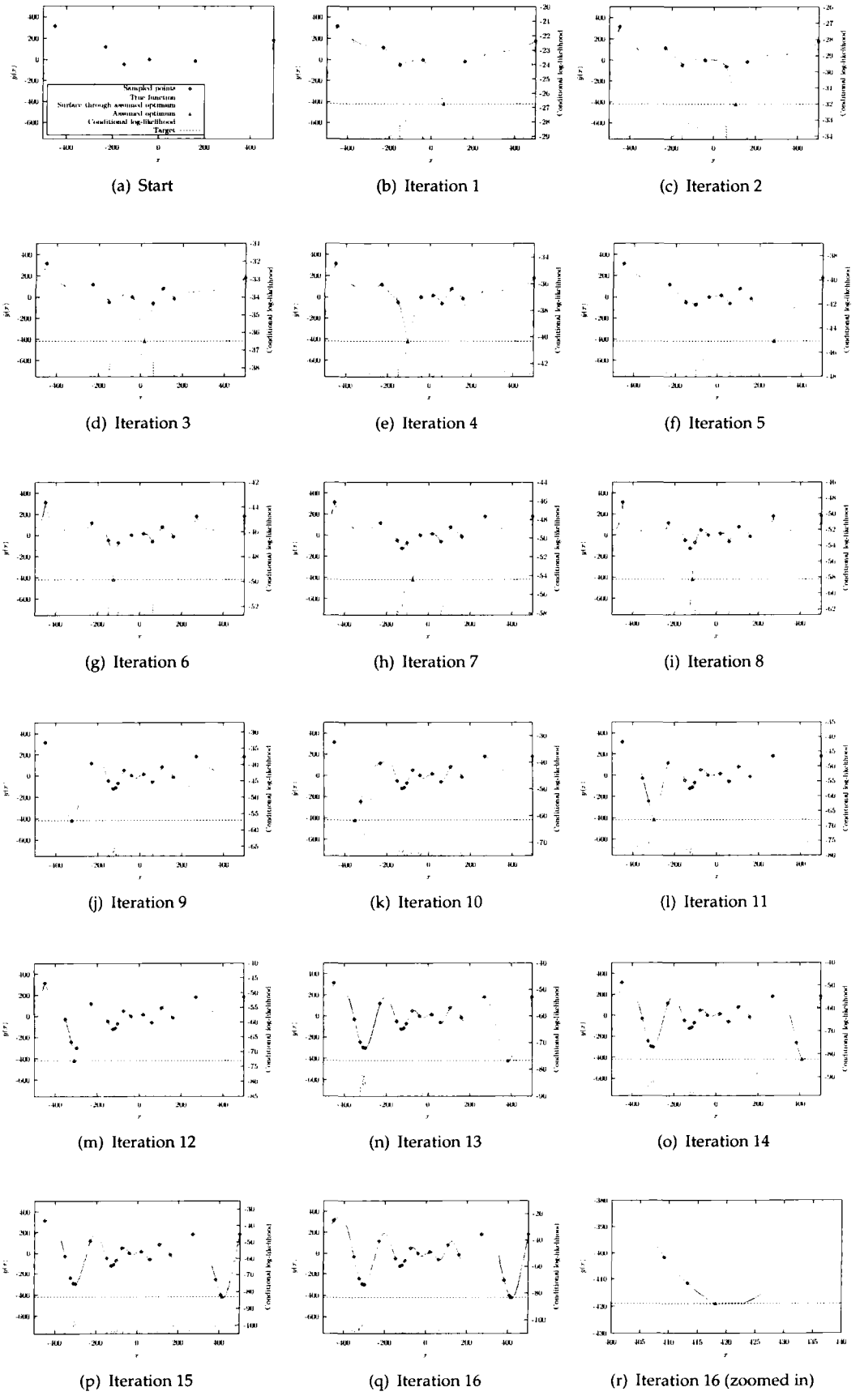
The iterations of this method on the Schwefel test function are shown in Figure 4.10. In each iteration the most credible kriging surface interpolating the observed points and the assumed optimum is shown; the value of f^* is set to be the value of the true minimum.

4.4.2.2 Multiple Targets

As with the previous method, multiple design vectors may be chosen for evaluation by using multiple values of f^* at each iteration. Figure 4.11 shows the iterations of this method on the Schwefel test function using values of f^* as set before.

4.5 Discussion

This chapter, like [14], attempted to present an overview of the different methods available for single-objective optimization using kriging surrogate models. Unlike [14], however, algorithms were categorized by whether or not they have tunable parameters; those with tunable parameters were then categorized further by whether an estimate of the global minimum was needed or not. In [14] two methods were selected as most promising: the probability of improvement with multiple targets, and the one-stage credibility of hypotheses method (with either single or multiple targets each iteration).



(a) Start (b) Iteration 1 (c) Iteration 2

(d) Iteration 3 (e) Iteration 4 (f) Iteration 5

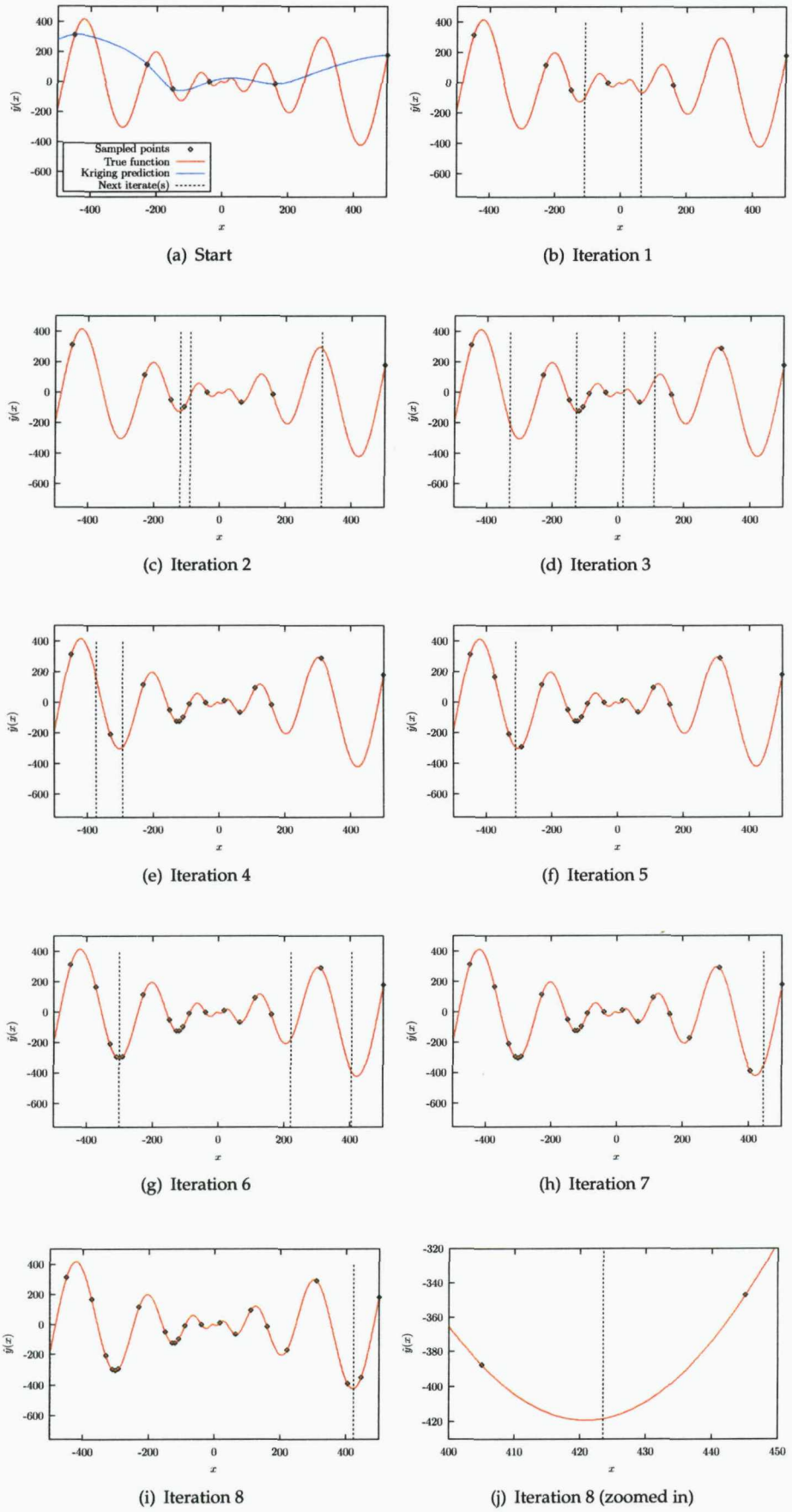
(g) Iteration 6 (h) Iteration 7 (i) Iteration 8

(j) Iteration 9 (k) Iteration 10 (l) Iteration 11

(m) Iteration 12 (n) Iteration 13 (o) Iteration 14

(p) Iteration 15 (q) Iteration 16 (r) Iteration 16 (zoomed in)

FIGURE 4.10: Iterations of the one-stage kriging approach on the Schwefel test function.



(a) Start (b) Iteration 1

(c) Iteration 2 (d) Iteration 3

(e) Iteration 4 (f) Iteration 5

(g) Iteration 6 (h) Iteration 7

(i) Iteration 8 (j) Iteration 8 (zoomed in)

FIGURE 4.11: Iterations of the enhanced one-stage kriging approach on the Schwefel test function.

Since their introduction in the publication of [14], one-stage methods have not received much attention in the literature. However, very recent developments (using radial basis functions, rather than kriging surrogate models) [74, 83] may revive interest in this promising approach.

Even more recently, a new approach based on information theory has been proposed [84]. It outperforms the expected improvement criterion over a range of test functions and appears to be a promising approach to take.

Chapter 5

Kriging-Assisted Multi-Objective Optimization

5.1 Introduction

The previous chapter reviewed methods of selecting design vectors to evaluate, using kriging surrogate models, to achieve the goal of single-objective optimization. This chapter reviews methods of selecting design vectors to evaluate, using kriging surrogate models, to achieve the goal of multi-objective optimization.

5.1.1 Taxonomy of Kriging Assisted Multi-Objective Optimization Methods

At the most general level, multi-objective methods using surrogate models may be divided into scalarizing and non-scalarizing methods. A proposed taxonomy of kriging model-assisted multi-objective methods, based on this division, is given in Table 5.1. Unlike the single-objective taxonomy, methods for selecting multiple points and single points at each iteration have not been distinguished between [1]. Instead distinction is made between the scalarizing function used in the scalarized approach (note the weighting method is not included due to its inferiority to the weighted metric approach).

Scalarizing methods are quite simple: the multiple objectives of the MOOP are combined using some function which maps from $\mathbb{R}^M \rightarrow \mathbb{R}$. Then one of the methods for

[1] all scalarizing methods have natural methods of selecting multiple points (using different weighting vectors for weighted metric approaches, or different values of ϵ or different objective functions in the ϵ-constraint approach); methods for selecting multiple points from the single-objective taxonomy could also be used for the scalarizing methods.

TABLE 5.1: Taxonomy of kriging model-assisted multi-objective optimization methods.

Scalarizing Method	**Method for selecting search points**								
	Scalarized Problem: Single kriging model						**Non-Scalarized Problem**: Multiple kriging models		
	Non-tunable Utility Functions: No parameters to set		**Tunable Utility Functions**: Parameters allowing the balance between exploration and exploitation to be altered				**Non-tunable Utility Functions**: No parameters to set		
			Non-target based: does not require an estimate of the optimum		Target-based: requires an estimate f^* of the optimum				
	Surface Minimum	Expected Improvement	Generalized Expected Improvement	Weighted Expected Improvement	Probability (of achieving)	Credibility (of hypothesis)	Pareto Points of Surfaces	Probability of Improvement	Expected Improvement
ϵ-Constraint Method		Section 5.2.2					Section 5.3.1	Section 5.3.2	Section 5.3.3
Weighted Metric		Section 5.2.3							

TABLE 5.2: Table of scalarizing methods for transforming MOOPs to SOOPs.

Function Name	**Function Expression**	**Constraints**
ϵ-constraint Method	$f_l(\mathbf{x})$	$f_j(\mathbf{x}) \leq \epsilon_j \quad j = 1,2,\ldots,M, j \neq l.$ $x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1,2,\ldots,d.$
Weighting Method	$\sum_{i=1}^{M} w_i f_i(\mathbf{x})$	$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1,2,\ldots,d.$
Weighted L_p Method	$\left(\sum_{i=1}^{M} w_i \lvert f_i(\mathbf{x}) - z_i^* \rvert^p\right)^{\frac{1}{p}}$	$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1,2,\ldots,d.$
Weighted Tchebycheff Method	$\max_{i=1,2,\ldots,M} [w_i \lvert f_i(\mathbf{x}) - z_i^* \rvert]$	$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1,2,\ldots,d.$
Augmented Weighted Tchebycheff Method	$\max_{i=1,2,\ldots,M} [w_i \lvert f_i(\mathbf{x}) - z_i^{**} \rvert] + \rho \sum_{i=1}^{M} \lvert f_i(\mathbf{x}) - z_i^{**} \rvert$	$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1,2,\ldots,d.$
Modified Weighted Tchebycheff Method	$\max_{i=1,2,\ldots,M} \left[w_i \left(\lvert f_i(\mathbf{x}) - z_i^{**} \rvert + \rho \sum_{i=1}^{M} \lvert f_i(\mathbf{x}) - z_i^{**} \rvert \right) \right]$	$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1,2,\ldots,d.$

single-objective optimization from the previous chapter is used to minimize the scalarized function. By varying the parameters which control how the multiple objectives are combined, an approximation to the Pareto-optimal front can be built up. Scalarizing methods are discussed in Section 5.2.

Non-scalarizing methods consider each objective function individually. The simplest of these is just to evaluate the Pareto points predicted by the multiple surrogate models (equivalent to the strawman approach in single-objective optimization); however more sophisticated methods are beginning to appear in the literature. Non-scalarizing methods are discussed in Section 5.3.

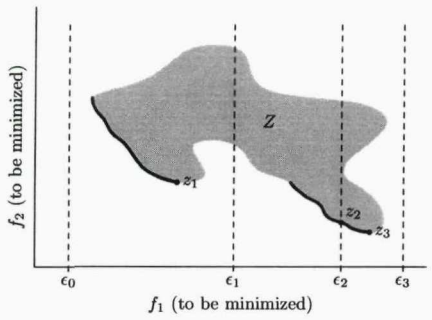


FIGURE 5.1: The ϵ-constraint method, for different values of ϵ.

5.2 Scalarizing Methods

5.2.1 Scalarizing Functions

The main purpose of a scalarizing function is to combine the multiple objectives of a MOOP in such a way that the contours of the resulting function are able to capture every point on the Pareto-optimal front. Recent reviews of scalarizing methods for multi-objective optimization in engineering may be found in [85, 86]. Some of the most popular scalarizing functions are shown in Table 5.2. With the exception of the ϵ-constraint method, it is necessary to normalize each objective f_i first so that it lies in the range [0,1]. The following sections discuss these methods in more detail: in all cases, each f_i is assumed to have been normalized.

5.2.1.1 ϵ-constraint Method

The ϵ-constraint method considers only one of the objectives for minimization, whilst treating the other objectives as constraints to be satisfied, thus transforming the MOOP to a constrained SOOP. The problem to be solved becomes:

$$\begin{aligned} &\text{Minimize} && f_l(\mathbf{x}) && \\ &\text{subject to} && f_j(\mathbf{x}) \leq \epsilon_j, && j = 1,2,\ldots,M, j \neq l \\ &\text{and} && x_i^{(L)} \leq x_i \leq x_i^{(U)} && i = 1,2,\ldots,d. \end{aligned} \tag{5.1}$$

The way the ϵ-constraint method works is illustrated for a two-objective problem in Figure 5.1, where the Pareto-optimal front is displayed as a bold line. In this example, f_2 is chosen as the objective to be minimized, whereas different upper bounds $\epsilon_i, i = 0,1,2,3$ are shown for objective f_1. In this case:

- ϵ_0 is too low; no feasible solution has a value of $f_1 < \epsilon_0$, and so no solution is found.
- ϵ_1 yields solution z_1; it is the solution which has the lowest f_2 value, for $f_1 < \epsilon_1$.
- ϵ_2 yields solution z_2; it is the solution which has the lowest f_2 value, for $f_1 < \epsilon_2$.
- ϵ_3 does not restrict the feasible set at all, and yields solution z_3; it is the solution which has the lowest f_2 value in the entire feasible region.

Several important properties of these methods are discussed in [38], including:

1. The solution of Equation 5.1 is weakly Pareto-optimal.
2. $\mathbf{x}^*$ is Pareto-optimal if and only if it is a solution of Equation 5.1 $\forall l = 1, \ldots, M$ where $\epsilon_j = f_j(\mathbf{x}^*), j = 1, \ldots, M, j \neq l$.
3. If $\mathbf{x}^*$ is the unique solution of Equation 5.1 for some l, with $\epsilon_j = f_j(\mathbf{x}^*), j = 1, \ldots, k, j \neq l$, then $\mathbf{x}^*$ is Pareto-optimal.

The second of these properties implies that all Pareto-optimal solutions may be found to a MOOP using this method. Despite this appealing feature, it does require solving a SOOP with a (potentially) high number of constraints, which is undesirable.

The remainder of the methods in this section combine the multiple objectives into a single expression to be minimized.

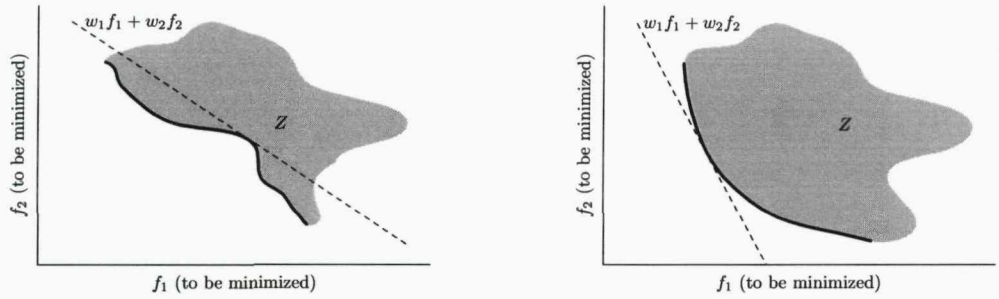
5.2.1.2 Weighting Method

The weighting method is the simplest way of combining multiple objectives into one single objective: each objective is simply associated with a weighting coefficient, and the weighted sum is then the single objective to be minimized. Formally, the problem becomes:

$$\begin{aligned} &\text{Minimize} \quad \sum_{i=1}^{M} w_i f_m(\mathbf{x}) \\ &\text{subject to} \quad x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1, 2, \ldots, d. \end{aligned} \tag{5.2}$$

where $w_i \geq 0 \forall i$ and $\sum_{i=1}^{M} w_i = 1$.

Several results for the weighting method are given in [38], the most significant of which are:



(a) Weighting method for concave MOOP.

(b) Weighting method for convex MOOP.

FIGURE 5.2: Weighting method on concave and convex MOOPs.

1. The solution of Equation 5.2 is weakly Pareto-optimal.
2. The solution of Equation 5.2 is Pareto-optimal if $w_i > 0, \forall i$.
3. If the solution to Equation 5.2 is unique, then it is Pareto-optimal.
4. If the MOOP is convex, then $\forall$ Pareto-optimal solutions $\mathbf{x}^*$, $\exists \mathbf{w} \in \mathbb{R}^M$ such that $\mathbf{x}^*$ is a solution of Equation 5.2.

This final theorem is illustrated in Figure 5.2. For the convex problem in Figure 5.2(b), each solution on the Pareto-optimal front has a weighting vector associated with it such that a contour exists which is tangential to it, and which intersects no other solution inside the feasible region. However, for the concave problem in Figure 5.2(a), solutions exist on the Pareto-optimal front for which no such weighting vector exists (such as that indicated).

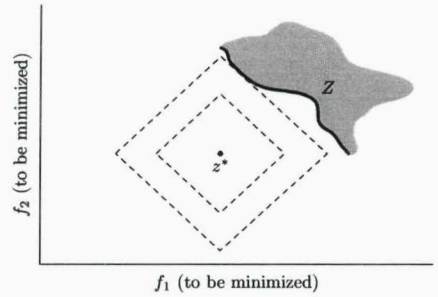
So clearly the simplicity of the weighting method comes at a price: only convex MOOPs can have their Pareto-optimal fronts approximated with any degree of reliability.

5.2.1.3 Weighted L_p Metric and Weighted Tchebycheff Metric

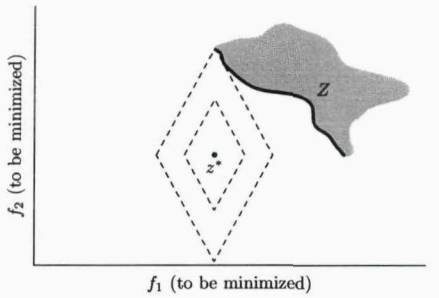
Some of the inadequacies of the previous method may be overcome by using different metrics to define the distance of a solution from the Utopian point. A common type of metric is the L_p metric, defined as $||\mathbf{z}^1 - \mathbf{z}^2||_p = \left[\sum_{i=1}^{d}(|z_i^1 - z_i^2|^p)\right]^{1/p}$. Contours of the L_1 and L_2 metrics are shown in Figure 5.3 and Figure 5.4 respectively.

Using the weighted L_p metric method, the problem to be solved becomes

$$\begin{aligned} &\text{Minimize} \quad \left(\sum_{i=1}^{M} w_i |f_i(\mathbf{x}) - z_i^*|^p\right)^{\frac{1}{p}} \\ &\text{subject to} \quad x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1, 2, \ldots, d. \end{aligned} \tag{5.3}$$

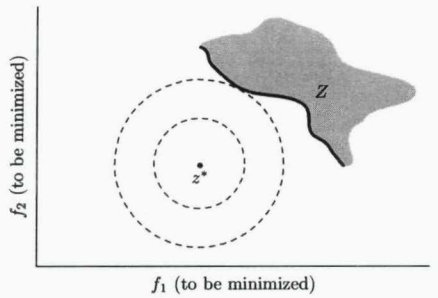


(a) Contours for weighted L_1 metric ($w_1 = w_2$).

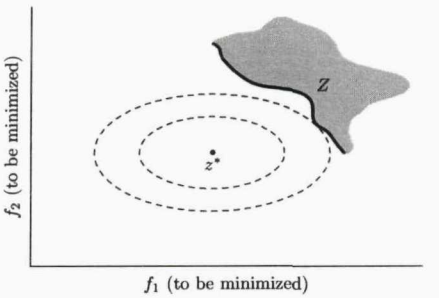


(b) Contours for weighted L_1 metric ($w_1 \neq w_2$).

FIGURE 5.3: Contours for the weighted L_1 metric.



(a) Contours for weighted L_2 metric ($w_1 = w_2$).



(b) Contours for weighted L_2 metric ($w_1 \neq w_2$).

FIGURE 5.4: Contours for the weighted L_2 metric.

where $0 < p < \infty$. For the L_p metric, one of the most important results is that the solution to Equation 5.3 is Pareto-optimal if either the solution is unique, or $w_i > 0, \forall i$ [38]. Whilst appearing promising, the main drawback is in what the theorem does not say: although solving Equation 5.3 produces Pareto-optimal solutions, it does not necessarily produce all of them.

For $p = \infty$, the metric is known as the Weighted Tchebycheff Metric, and the problem becomes:

$$\begin{aligned} &\text{Minimize} \quad \max_{i=1,2,\ldots,M} [w_i | f_i(\mathbf{x}) - z_i^* |] \\ &\text{subject to} \quad x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1, 2, \ldots, d. \end{aligned} \tag{5.4}$$

The most significant result for the weighted Tchebycheff method is that for all MOOPs, and for all Pareto-optimal solutions $\mathbf{x}^*$, $\exists \mathbf{w} \in \mathbb{R}^M (\mathbf{w} > \mathbf{0})$ such that $\mathbf{x}^*$ is a solution of Equation 5.4, where the reference point used is the Utopian vector $\mathbf{z}^{**}$.

Contour lines for the Weighted Tchebycheff Metric are shown in Figure 5.5. As can be seen, solutions on the concave part of the Pareto-optimal front can indeed be located

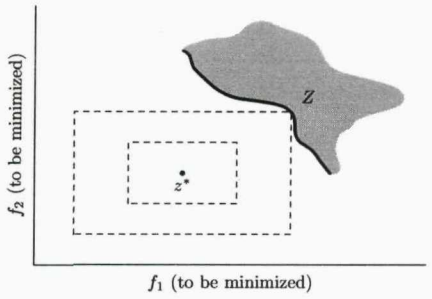


FIGURE 5.5: Contour lines for the weighted Tchebycheff metric.

with this metric. The main drawback with the weighted Tchebycheff method is that, whilst all Pareto-optimal solutions can be found using Equation 5.4, some of them may be weakly Pareto-optimal.

5.2.1.4 Augmented Weighted and Modified Weighted Tchebycheff Metric

The inadequacy of finding weakly Pareto-optimal solutions using the weighted Tchebycheff method may be overcome by varying the metric slightly. In particular, by adding a slope to the contours of the Tchebycheff metric, weakly Pareto-optimal solutions may be avoided. Two methods which may be used to achieve this are the augmented weighted Tchebycheff method:

$$\begin{aligned} &\text{Minimize} \quad \max_{i=1,2,\ldots,M} [w_i |f_i(\mathbf{x}) - z_i^{**}|] + \rho \sum_{i=1}^{M} |f_i(\mathbf{x}) - z_i^{**}| \\ &\text{subject to} \quad x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1, 2, \ldots, d. \end{aligned} \tag{5.5}$$

and the modified weighted Tchebycheff method:

$$\begin{aligned} &\text{Minimize} \quad \max_{i=1,2,\ldots,M} \left[w_i \left(|f_i(\mathbf{x}) - z_i^{**}| + \rho \sum_{i=1}^{M} |f_i(\mathbf{x}) - z_i^{**}| \right) \right] \\ &\text{subject to} \quad x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1, 2, \ldots, d. \end{aligned} \tag{5.6}$$

where ρ is a small positive scalar. The contours for the augmented weighted Tchebycheff metrics are shown in Figure 5.6. The difference in the additional slopes added to the contours of the weighted Tchebycheff metric for the augmented and modified methods is illustrated in Figure 5.7.

Although each of these methods has the disadvantage that some Pareto-optimal solutions may become impossible to find, it may be shown [87] that for every properly

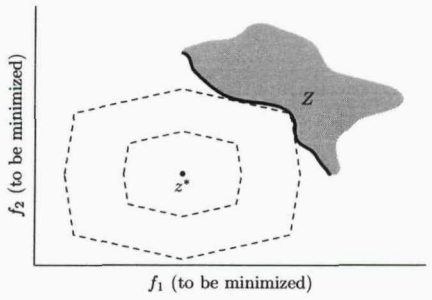


FIGURE 5.6: Contour lines for the augmented weighted Tchebycheff metric.

(a) Contours for augmented weighted Tchebycheff metric.

(b) Contours for modified weighted Tchebycheff metric.

FIGURE 5.7: Difference in contours for the augmented and modified weighted Tchebycheff metrics.

Pareto-optimal solution $\mathbf{x}^*$ of a MOOP, $\exists \mathbf{w} \in \mathbb{R}^M(w_i > 0, \forall i), \rho > 0$ such that $\mathbf{x}^*$ is a solution to Equation 5.5 and Equation 5.6.

5.2.1.5 Discussion

Through use of increasingly sophisticated metrics, it has been seen how scalarizing methods can be used to find:

- All Pareto-optimal solutions to convex MOOPs, using the weighted method,
- All Pareto-optimal solutions to all MOOPs, using the weighted Tchebycheff method,
- All properly Pareto-optimal solutions to all MOOPs, using variants of the weighted Tchebycheff method.

After using any of these methods to transform a MOOP to a SOOP, any of the methods in the previous chapter may then be used to solve the resulting SOOP. By varying the weights used in the scalarizing method, approximations of the Pareto-optimal front can then be built up. This gives rise to a huge number of possible cost-effective algorithms for multi-objective optimization. Surprisingly few (other than the simplest, such as the

weighting method combined with the 'strawman' approach) have been pursued in the literature. Two notable exceptions are discussed in the remainder of this section.

5.2.2 ϵ-constrained EGO Algorithm

In [70], the EGO algorithm is used to solve a practical MOOP, using the ϵ-constraint method. The viscosity of a material was chosen as the objective (which was to be minimized), whilst the yield stress (which was to be maximized) was treated as a constraint, constrained to be greater than a value of C. The method performed well, and resulted in new materials being identified which outperformed existing materials simultaneously in both objectives.

5.2.3 ParEGO

In [45], the EGO algorithm was used to solve MOOPs using the augmented weighted Tchebycheff method. The algorithm, named ParEGO, was tested on a wide range of MOOPs, and the results were compared (using several performance indicators) to the popular NSGA-II algorithm. Unsurprisingly, ParEGO outperformed NSGA-II on almost every problem (NSGA-II is 'greedy', as opposed to cost-effective), and so perhaps the comparison is slightly unfair. As more and more cost-effective multi-objective algorithms appear, more useful comparisons should be possible in the future.

5.3 Non-Scalarizing Methods

As their name suggests, non-scalarizing methods do not combine the multiple objectives of a MOOP into a single objective for a single-objective optimization algorithm to solve. Instead, each objective function is considered individually when determining where to evaluate next in design variable space.

Many greedy multi-objective optimization algorithms exist which are non-scalarizing, in particular Multi-Objective Evolutionary Algorithms (MOEAs). However, non-scalarizing methods for cost-effective multi-objective optimization have only appeared fairly recently. This section discusses such methods.

5.3.1 Strawman

In [88], a special switched reluctance motor was optimized so as to maximize the average torque, and minimize the ripple torque. Each objective was computationally expensive to evaluate, and so kriging surfaces were constructed to approximate each separately. The Pareto-optimal points of the two surfaces were then located (using three different optimization algorithms). The approach did not employ on-line learning: that is, once the Pareto-optimal designs (located by the optimization algorithms) were evaluated, they were not used to create an updated kriging model, for searching again with the optimization algorithms. Instead, a large initial experimental design of 125 points (requiring 2375 finite element field calculations in total) was used, thus ensuring the kriging models constructed - and therefore the Pareto-optimal set identified - were fairly accurate. However, by using a smaller initial experimental design, along with a process of on-line learning (i. e. an iterative procedure of building kriging models, identifying Pareto designs using an optimization algorithm, evaluating these designs and then building updated kriging models for searching again), the approach could have been made more efficient, as evaluations in design variable space could have been targeted earlier towards the Pareto-optimal set.

5.3.2 Probability of Improvement

In Section 4.2.2, the concept of improvement was introduced for single-objective optimization. By using the notion of dominance, this concept can easily be extended to multi-objective optimization. Suppose at a given stage of an optimization search, a set S of Pareto-optimal solutions exist. Then a new solution $\mathbf{z}^*$ is said to constitute an improvement if and only if $\mathbf{z}^*$ is not dominated by the members of S. Thus $\mathbf{z}^*$ may:

1. be equivalent to all members of S, or
2. dominate at least one member of S.

This is illustrated for a two-objective problem in Figure 5.8. Both shaded regions represent the area in objective function space which a solution which constitutes an improvement would map into. Solutions which are equivalent to the 5 existing Pareto-optimal solutions map to the region labeled 'Equivalent Designs'; solutions which dominate at least one of the 5 Pareto-optimal solutions map to the region 'Dominating Designs'. Depending on where exactly the solution lies in this region, it replaces 1 or more solutions in the Pareto-optimal set.

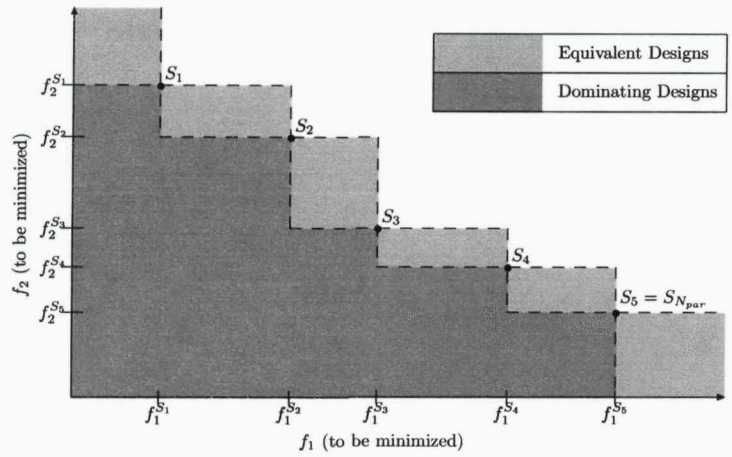


FIGURE 5.8: Regions of equivalence and improvement for $N_{\text{par}} = 5$ Pareto solutions.

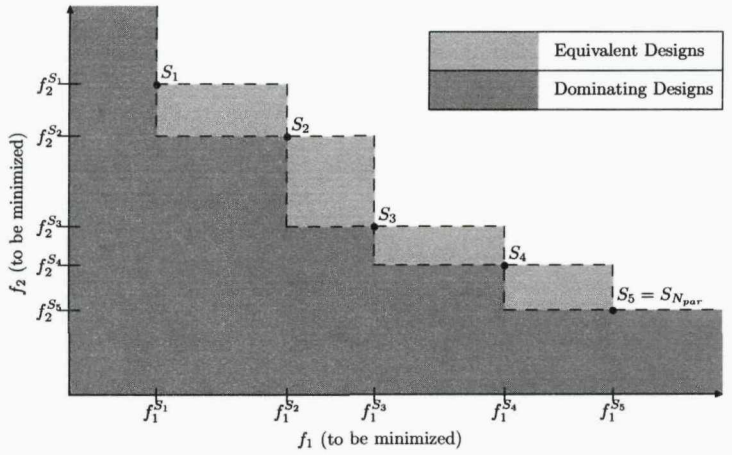


FIGURE 5.9: Regions of equivalence and improvement for $N_{\text{par}} = 5$ Pareto solutions, as used by Keane [13].

In [13, 89, 90] this idea was used to select points for evaluation. In [13, 90], a slightly different convention is used, and solutions which expand on the frontier of the Pareto-optimal front are considered as equivalent to the Dominating Designs, as shown in Figure 5.9. Using this convention, the probability of a design vector constituting an improvement (either a 'Dominating Solution' or an 'Equivalent Solution'), is shown, for a two-objective function problem, to be

$$\begin{aligned} P[I]_{\text{aug}} &= \Phi\left[\frac{f_1^{S_1} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right] + \left\{1 - \Phi\left[\frac{f_1^{S_{Npar}} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right]\right\}\Phi\left[\frac{f_2^{S_{Npar}} - \hat{y}_2(\mathbf{x})}{s_2(\mathbf{x})}\right] \\ &+ \sum_{i=1}^{N_{par}-1}\left\{\Phi\left[\frac{f_1^{S_{i+1}} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right] - \Phi\left[\frac{f_1^{S_i} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right]\right\}\Phi\left[\frac{f_2^{S_i} - \hat{y}_2(\mathbf{x})}{s_2(\mathbf{x})}\right] \end{aligned} \quad (5.7)$$

whilst the probability of it actually dominating at least one existing Pareto-optimal solution is shown to be

$$P[I]_{\text{dom}} = \Phi\left[\frac{f_1^{S_1} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right] + \left\{1 - \Phi\left[\frac{f_1^{S_{N_{par}}} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right]\right\}\Phi\left[\frac{f_2^{S_{N_{par}}} - \hat{y}_2(\mathbf{x})}{s_2(\mathbf{x})}\right] + \sum_{i=1}^{N_{par}-1}\left\{\Phi\left[\frac{f_1^{S_{i+1}} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right] - \Phi\left[\frac{f_1^{S_i} - \hat{y}_1(\mathbf{x})}{s_1(\mathbf{x})}\right]\right\}\Phi\left[\frac{f_2^{S_{i+1}} - \hat{y}_2(\mathbf{x})}{s_2(\mathbf{x})}\right] \quad (5.8)$$

The equations for a design vector constituting an improvement for an arbitrary number of objectives are given in [89].

Both [13] and [89] report a significant weakness in this probability of improvement utility function for MOOPs: it does not favour large improvements. Both however suggest overcoming this weakness by instead evaluating the multi-objective equivalent of the expected improvement.

5.3.3 Expected Improvement

In single-objective optimization, the expected improvement is the integral of the improvement over the likelihood of achieving it. Defining expected improvement in multi-objective optimization is similar:

$$E[I(\mathbf{x})] = \int_{\mathbf{y}\in V_{nd}} I(\mathbf{y}) PDF_{\mathbf{x}}(\mathbf{y}) d\mathbf{y} \quad (5.9)$$

where V_{nd} is the region of objective function space where solutions are non-dominated.

Calculating this integral is non-trivial, and different approaches exist. Both [13] and [89] report reasonable results using the expected improvement criterion for selecting design vectors in multi-objective optimization, and it is certainly more aggressive in its search than the probability of improvement approach.

5.4 Discussion

The extension of cost-effective methods to multi-objective optimization has undoubtedly be slow. Scalarizing methods (due to them being more obvious) have been the most popular, but even then usually only the simplest methods from single-objective optimization are used.

More promising has been the emergence of non-scalarizing cost-effective methods for MOOPs. These have only appeared very recently in the literature, and are the true multi-objective counterparts to the utility functions for single-objective optimization reviewed in the previous chapter.

Both scalarizing and non-scalarizing methods will be developed further in Chapter 7.

Chapter 6

Practical Issues

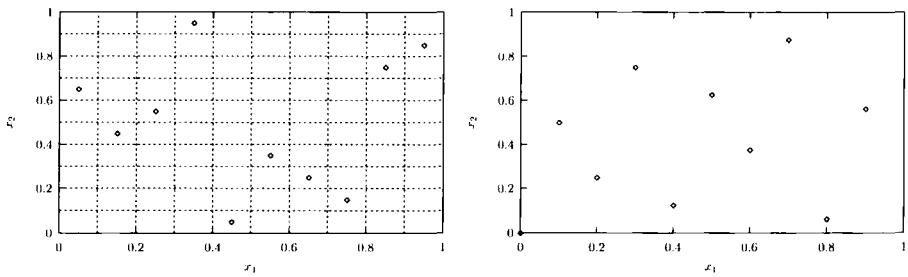
6.1 Introduction

So far, methods for selecting design vectors, using kriging surrogate models, have been reviewed, for both single-objective and multi-objective optimization. This is the main aspect of optimization discussed in the literature, but in practice, much more needs to be considered. This chapter attempts to go some way in discussing those aspects which are sometimes easily overlooked in the literature.

6.2 Choosing the Initial Set of Examples

Obviously, kriging surrogate models cannot be used to select every design vector to evaluate during an optimization search: a certain minimum number of design vectors need to be sampled before a kriging model can even be constructed. This initial set is called an experimental design, and the theory behind selecting suitable points is known as Design of Experiments [91].

Classical experimental designs were for real experiments involving measurements, and as such had to account for features such as randomness and non-repeatability. Experimental designs for computer experiments do not have to take into account such features, and so differ from their classical counterparts. Experimental designs for computer experiments are commonly referred to as *modern* experimental designs [92, 93, 94]. In this section, two common modern experimental designs are discussed, namely the Latin Hypercube, and the Hammersley sequence.



(a) Latin hypercube, $n = 10$ (b) Hammersley sequence, $n = 10$

FIGURE 6.1: Two popular experimental designs on the unit square.

6.2.1 Latin Hypercube

A Latin square of size n may be constructed by partitioning each side into n intervals to create an $n \times n$ grid, and then placing a point in the centre of n squares in the grid, so that each row and each column only contains one point. The generalization of this concept to higher dimensions is known as a Latin Hypercube. It provides a simple way of covering a design space with relatively few points; however, some Latin Hypercubes can be quite poor at providing a uniform spread. Latin hypercubes are very popular in the literature for initializing surrogate-model assisted algorithms, e. g. in [70], a Latin hypercube of size $10d$ was used to initialize the EGO algorithm.

Figure 6.1(a) shows a Latin Hypercube of size 10 on the unit square.

6.2.2 Hammersley Sequence

A Hammersley sequence [95, 96] provides a relatively uniform distribution of points over a space. In particular, it provides a low discrepancy distribution of points, discrepancy being a measure of the deviation of a sequence from the uniform distribution.

The formal construction of a Hammersley sequence may be found in [96]; a Hammersely sequence of size 10 on the unit square is shown in Figure 6.1(b).

6.2.3 Discussion

Unlike most of the other topics in this chapter, the amount of literature on experimental designs is huge. Papers appear regularly comparing different designs, making recommendations on size and so on, however the findings of some contradict the findings of others. No clear conclusions can be drawn easily, other than obviously the more examples included in the design, the more accurate the resulting surrogate model is likely to

be. It also seems to be quite appealing to include the 2^d corner points of design variable space if possible (this is not feasible when the number d of design variables is large); indeed, apart from the central point of design variable space, these are the only points used in the radial basis function of Gutmann [72]. It also seems quite beneficial to include several points close together in design variable space, as this forces the gradient of the resulting surrogate model to be very accurate in this region, and this seems to increase the accuracy of the surrogate model considerably. However, all findings are empirical, and so making general recommendations is not easy. More often than not, as the emphasis in the literature is on selection criteria (as discussed in the previous chapters), not much worry is put into the process, and a standard experimental design (usually a Latin Hypercube) of a size proportional to the number of design variables (e.g. $10d$ in [70]) is used.

6.3 Determining the Parameters of the Kriging Model

To construct the kriging surrogate model, the concentrated log-likelihood function (given in Equation 3.28) needs to be maximized over the kriging parameters θ and $\mathbf{p}$. Typical plots of the log-likelihood against these kriging parameters suggest that a deterministic algorithm, such as the Nelder-Mead simplex algorithm [97], is suitable for this purpose. This is used by some in the literature, e.g. [45], whilst [98] hybridizes it with a genetic algorithm.

6.4 Choosing a Utility Function

Chapter 4 and Chapter 5 reviewed a wide range of utility functions which could be used for selecting design vectors to evaluate during optimization. In practice however, a choice must be made of which one to actually use. Each have their own particular merits and drawbacks: EI is simple to use as it has no parameters to set, and it provides a natural balance between exploration and exploitation, however it can sometimes be slow to converge on the global optimum, even when it has found its basin of attraction; GEI, WEI and POI on the other hand, allow the balance between exploration and exploitation to be controlled, the drawback being that such control is inevitably arbitrary. We have already seen how cooling schemes and cyclic schemes can be used to control the parameters of these algorithms, and how using multiple values of the parameters at each iteration can be used to select multiple points. The 'one-stage' method described in Section 4.4.2 also allows this control, and has the additional advantage of being less

vulnerable to deceptive experimental designs, however it has the drawback that it itself becomes computationally expensive as the number of sampled points grows.

Ultimately, the choice of which utility function to use is determined by the problem, available computing resources, and whether we wish to ever intervene to force iterations to be more exploratory or exploitative. If we simply wish to start an algorithm and then wait for the results, then EI, or GEI/WEI with a suitable cooling or cyclic scheme would seem the most appropriate choice. If multiple computers are at our disposal, allowing several evaluations each iteration, then using multiple targets with POI would seem the most appropriate choice. If iterations are extremely expensive, and/or we have a good estimate of the objective function value at the optimum, then the credibility utility function seems a good option, especially if multiple computers are available for use at each iteration.

6.5 Locating the Optimum of the Utility Function

Whilst the objective functions of practical optimization problems may not display any typical behaviour, the utility functions used in kriging-assisted single-objective optimization do exhibit typical behaviour, as can be seen by examining the figures in Chapter 4. Typically, the EI, GEI, WEI, POI and credibility utility functions are all highly multimodal. The EI, GEI, WEI and POI functions all take values of zero at the sampled points, and can be quite flat (taking values very close to zero) over large regions of design variable space, making them difficult for deterministic algorithms to optimize; the credibility utility function takes values of minus infinity at the sampled points, and can also be quite flat over large regions of design variable space, again making them potentially difficult to optimize.

Very often, the algorithm used to maximize the chosen utility function is not reported in the literature, however it is clear (due to the existence of large numbers of local minima, and existence of large featureless flat regions), that a good global optimization algorithm is needed. In [70], a branch-and-bound algorithm was used to maximize the expected improvement function in the EGO algorithm. Due to the difficulty of implementing this algorithm, a standard genetic algorithm was instead used in [45] to maximize the expected improvement in the ParEGO algorithm.

6.6 Updating the Kriging Model

After each iteration of a kriging-assisted optimization algorithm, a new example (or several new examples) exists, which should be considered in subsequent iterations. In particular, the correlation matrix $\mathbf{R}$, which depends on all the observed points, needs to be updated. This matrix is then to be inverted, which can be computationally expensive. Preferably some technique can be used to reduce this cost.

In [99], a method of updating the correlation matrix is suggested using its Cholesky factorization. Such a technique could be advantageous if the number of sampled points grows very large, however it is only useful when the kriging parameters θ and $\mathbf{p}$ are not being updated. This technique may be particularly useful for reducing the computational cost of the one-stage approach, as it could be used to update the conditional correlation matrix $\mathbf{C}$ at each iteration. In the algorithms used in this thesis however, full updates occur at every iteration: that is, the kriging parameters are optimized using the updated log-likelihood function at each iteration, and the updated correlation matrix $\mathbf{R}$ (or $\mathbf{C}$ in the case of the one-stage approach) is inverted.

6.7 Dealing with Failed Iterations

When running algorithms on mathematical test functions, iterations are always successful, in the sense that whatever the design vector the algorithm chooses to evaluate next, it can actually be evaluated. This however, is not always the case in engineering design problems. Evaluating objective functions and constraints using CAD software involves building a model, generating a mesh, creating a database, running a solver and then post-processing to evaluate the outputs of interest. Any one of these stages is prone to failure; however, if an iteration does fail, this should not subsequently mean that the algorithm then fails. Instead mechanisms should be built into the algorithm to deal with failed iterations.

This issue is crucial for every algorithm included in a practical optimization toolkit. However, this topic is almost entirely overlooked in the literature. Recently however, methods were proposed in [100] for dealing with missing data, which involved imputing values for failed iterations which consequently penalized that region of design variable space.

6.8 Inequality Constraint Handling

In Chapter 4 and Chapter 5, attention was only given to achieving the aim of minimizing the objective function(s) of an optimization problem. In particular, satisfaction of constraints was ignored. However, the constraints are a crucial part of the problem, as they dictate whether a solution is feasible or not: and in many cases, finding a solution which is infeasible (no matter by how little), is as good as finding no solution at all. Clearly significant attention needs to be paid to constraint handling during an optimization search. This section deals with the handling of inequality constraints, whilst the next section deals with the handling of equality constraints.

6.8.1 Probability Methods

Probability methods work with the EI, GEI and POI utility functions [1] in single-objective optimization. They simply involve calculating the probability of each constraint being satisfied (for an unknown design vector), and then multiplying the utility function by this. Thus design vectors which are most probable to violate a constraint have lower utility function values, and so are less likely to be selected for evaluation.

In [101], this method was used successfully with the EGO algorithm to optimize an engine piston subject to certain friction constraints, and in [102] it was used with the EGO algorithm to optimize an airfoil design subject to drag constraints. A drawback of the method is that it seems to influence the value of the utility function too severely, thus preventing the algorithm from exploring points along the constraint boundary where the true optimum lies. This is because the utility function is penalized gradually outwards from the feasible region, this impacting on the feasible region.

6.8.2 Penalty Methods

Penalty methods are perhaps even simpler than probability methods, as they work by reformulating the constrained optimization problem as an unconstrained optimization problem:

$$\begin{aligned} &\text{Minimize} \quad w_f f(\mathbf{x}) + \textstyle\sum_{i=0}^{J} w_i \max(0, g_i(\mathbf{x})) \\ &\text{subject to} \quad x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1, 2, \ldots, d. \end{aligned} \tag{6.1}$$

[1] The reason probability methods cannot be used (directly) with the WEI or credibility utility functions is because they can take positive and negative values.

where the non-negative weighting factors w_f and w_i are included to place relative importance on the objective and constraints, which are all assumed to be normalized. As each inequality constraint g_i is supposed to be less than or equal to zero, the penalizing term is zero when the constraints are satisfied, and strictly positive when a constraint is violated. All the utility functions for single-objective optimization may then be used in the usual manner on Equation 6.1 to be minimized. This approach has been applied successfully to the radial basis function method of Gutmann [72] in the rbfsolve algorithm [73, 103], and also to the EGO algorithm. It has the advantage that the utility function is more sharply penalized at the constraint boundary, thus allowing the optimization algorithm to locate the optimum more quickly. It is the constraint handling method used with the rbfsolve and EGO algorithms in the powerful TOMLAB optimization software.

6.8.3 Expected Violation Method

The expected violation method, proposed in [104], may be used to naturally select multiple points per iteration for evaluation. Suppose N_p points are to be evaluated. Then, a large number of potential candidates for evaluation are selected using a Latin hypercube (which covers the entire search space). The expected violation of each candidate is then a J dimensional vector, whose i^{th} component is defined as:

$$EV_i[(\mathbf{x})] = \begin{cases} (\hat{g}_i(\mathbf{x}) - 0)\Phi\left(\frac{\hat{g}_i(\mathbf{x})-0}{s_{g_i}(\mathbf{x})}\right) + s_{g_i}(\mathbf{x})\phi\left(\frac{\hat{g}_i(\mathbf{x})-0(\mathbf{x})}{s_{g_i}(\mathbf{x})}\right) & \text{if } s_{g_i}(\mathbf{x}) > 0 \\ 0 & \text{if } s_{g_i}(\mathbf{x}) = 0 \end{cases} \tag{6.2}$$

where $\hat{g}_i(\mathbf{x})$ is the kriging prediction of the i^{th} constraint, and $s_{g_i}(\mathbf{x})$ is the mean squared error in this prediction. The value of Equation 6.2 is high when the constraint is likely to be violated, and when there is large uncertainty in the value of the constraint. The method proceeds by rejecting all candidate design vectors for which the maximum component of its expected violation vector is outside a user-defined limit. The N_p design vectors with the highest value of the EI utility function are then selected for evaluation.

This method of inequality constraint handling suffers from the fact that the initial Latin hypercube has to be very large in size in order for suitable design vectors to be found (e.g. of the order of 10^5 candidates were used in [104].)

6.8.4 Constrained Utility Function Method

The constrained utility function method simply transforms the unconstrained problem of maximizing the utility function, to a constrained problem. For example, in [105, 106], the problem of maximizing the expected improvement becomes:

$$\begin{aligned} \text{Maximize} \quad & E[I(\mathbf{x})] = \begin{cases} (f_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{\min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{f_{\min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) & \text{if } s(\mathbf{x}) > 0 \\ 0 & \text{if } s(\mathbf{x}) = 0 \end{cases} \\ \text{subject to} \quad & \hat{g}_j(\mathbf{x}) \geq 0 \qquad\qquad j = 1, 2, \ldots, J; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1, 2, \ldots, d. \end{aligned} \tag{6.3}$$

where $\hat{g}_i(\mathbf{x})$ is the kriging prediction of the i^{th} constraint. This method was also used in [107] to carry out constrained optimization on two electromagnetic devices.

Note that this method is equivalent to that proposed in [80], which modifies the WEI utility function using

$$WE[I(\mathbf{x})] = \begin{cases} w(f_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{\min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + (1-w)s(\mathbf{x})\phi\left(\frac{f_{\min}-\hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \\ \text{if } s(\mathbf{x}) > 0 \text{ and } \hat{g}_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, J \\ \\ 0 \\ \text{if } s(\mathbf{x}) = 0 \text{ or } \exists i \text{ such that } \hat{g}_i(\mathbf{x}) > 0. \end{cases} \tag{6.4}$$

6.9 Equality Constraint Handling

In practice, strict satisfaction of computationally expensive constraints is impossible: it is only realistic to force each equality constraint (which we cannot set ourselves) to be within a certain tolerance of its desired value. Two ways have appeared in the literature for achieving this: transforming the equality constraint into two inequality constraints, and transforming the equality constraint into an objective.

6.9.1 Transformation to Two Inequality Constraints

This approach rewrites each equality constraint $h_i(\mathbf{x}) = 0$ instead as

$$-\epsilon_i^L \leq h_i(\mathbf{x}) \leq \epsilon_i^U. \tag{6.5}$$

where $|\frac{\epsilon_i^L}{h_i}|, |\frac{\epsilon_i^U}{h_i}| \ll 1$. Thus some tolerance is acceptable in achieving the constraint. This is the approach suggested in [106], although no results are given using it.

6.9.2 Transformation to an Objective

This approach rewrites each equality constraint $h_i(\mathbf{x}) = 0$ as

$$\text{Minimize} \quad |h_i(\mathbf{x})|. \tag{6.6}$$

By minimizing this expression, each equality constraint is driven towards zero, which is what is required.

In [102], an equality constraint (on the cross-sectional area of an airfoil) was dealt with using this approach. The original constraint was that $A(\mathbf{x}) \approx A_1$ where $A(\mathbf{x})$ was the area of the airfoil, and A_1 was the desired area. This was reformulated as

$$\text{Maximize} \quad 1 - \frac{|A(\mathbf{x}) - A_1|}{A_1}. \tag{6.7}$$

It should be noted that this is equivalent to the objective in Equation 6.6, as maximization of $-|A(\mathbf{x}) - A_1|$ (the only variable part of Equation 6.7) is equivalent to the minimization of $h(\mathbf{x}) = |A(\mathbf{x}) - A_1|$. Using this approach obviously requires dealing with multiple objectives: in [102] this was done using a MOEA, however a scalarizing approach could also have been used.

6.10 Termination Criteria

Deciding when to terminate an optimization algorithm is an important factor to be considered. When test functions are used, a natural criterion is when the solution found by the algorithm is within a certain tolerance of the true solution; this is impractical for engineering problems however, as the true solution is not known. However, if goals are set for each objective function, then these could provide a natural stopping criterion; in effect this transforms each objective into a constraint to be satisfied.

If goals are not set, natural stopping criteria may be defined either from the point of view of the algorithm (such as solutions converging together, no significant improvements being made in recent iterations, or expected improvement/probability of improvement falling below a certain threshold), or from the user's point of view (such as the algorithm exceeding a certain number of iterations, or exceeding a certain time limit).

Part III

Novel Algorithms and Implementation

Chapter 7

Novel Kriging-Assisted Optimization Algorithms

7.1 Introduction

The purpose of this chapter is to propose novel algorithms which are effective in solving computationally expensive, constrained, multi-objective optimization problems, such as those which arise in electromagnetic design, building on the ideas of previous chapters. Of course, not all electromagnetic design problems are multi-objective: the first section proposes a novel algorithm for single-objective problems.

Most of the algorithms in this chapter are available through use of an 'advanced options' dialog, which allows full control over the selection criterion used at each iteration. This is discussed in more detail in the next chapter.

7.2 Hybrid One-then-Two Stage Single-Objective Algorithm

In Chapter 4 a range of different utility functions was presented for use in single-objective optimization. Each had its own particular merits and drawbacks: this motivates the algorithm proposed in this section, which attempts to draw on the strengths of each.

7.2.1 Motivation

In [14], the credibility of hypothesis utility function is recognized as a promising utility function for single-objective optimization. Perhaps its most attractive feature is that

it can perform well with deceiving experimental designs, something that is certainly not true with other utility functions. However this comes at a price: evaluating the credibility of a hypothesis itself becomes computationally expensive when the number of sampled points becomes large.

Expected improvement and its generalizations on the other hand, remain computationally cheap to evaluate throughout the optimization process. Thus it makes sense to switch to using the expected improvement utility function (and its generalizations) when the computational cost of using the credibility of hypothesis utility function becomes large. This is the main idea behind the following algorithm.

7.2.2 Overview of Algorithm

The proposed algorithm for locating the global minimum in d-dimensional design variable space consists of three steps: initialization, one-stage experimental design, and two-stage optimization search.

7.2.3 Initialization

The only purpose of the initialization step is to sample enough points to allow a non-trivial kriging model to be constructed (i.e. a model which is not a hyper-plane in $\mathbb{R}^d$). The space-filling Hammersley Sequence experimental design, of size $4d$, is used to select the points. The experimental design size of $4d$ is much smaller than is normally used ($10d$ is suggested in [70] for example), as the philosophy of this algorithm is to use information about objective function space to search for the minimum at the earliest possible opportunity [1].

7.2.4 One-Stage Experimental Design

Information about objective function space has now been obtained through sampling $4d$ points, and the aim in this second step is to use this information to strategically choose where to sample next. Let the minimum objective function value of the $4d$ sampled points be $f_{\min}$, and let the maximum be $f_{\max}$. A design vector $\mathbf{x}^*$ is then hypothesized to exist in design variable space which has an objective function value f^* given by Equation 4.13. The value of α determining f^* is varied using a cyclic scheme, and

[1] The authors of the TOMLAB optimization toolbox also seem to appreciate that very small experimental designs can be used with one-stage algorithms [77]. In any case, at least $d + 1$ points are needed for a non-trivial surrogate model to be constructed.

the design vector chosen for evaluation at each iteration is the one which maximizes the credibility of the hypothesis of it having objective function value f^*. This step is repeated for $6d$ iterations until $10d$ points in total have been evaluated.

7.2.5 Two-Stage Optimization Search

In this final step of the algorithm, a kriging model is constructed using the $10d$ sampled points. The GEI utility function is then used (with a cooling scheme) to select points for evaluation, until $g = 1$ is reached. Then the WEI utility function is used to select points for evaluation: a cyclic scheme is used for varying the weighting parameter w. Using WEI to finish the search allows more exploitative iterations to be used than is possible with the GEI utility function.

Full details of the algorithm (including its performance in electromagnetic design) may be found in [3, 4]. Constraint handling techniques may be included by penalizing the utility function being used at each iteration using any of the methods discussed in the previous chapter.

7.3 Kriging-Assisted Scalarizing Algorithms for Constrained Multi-Objective Optimization

In Chapter 5, a selection of both scalarizing and non-scalarizing methods were introduced for kriging assisted multi-objective optimization. From the scalarizing methods section, it is obvious that, due to the large number of utility functions which exist in single-objective optimization, and the large number of scalarizing functions available for transforming a MOOP into a SOOP, a plethora of algorithms are available for cost-effective multi-objective optimization. Despite this, other than the 'strawman' approach, only the expected improvement utility function has been used in scalarizing methods for MOOPs. The goal of this section is to propose new scalarizing algorithms, making better use of the methods available from single-objective optimization. Section 7.4 then deals with non-scalarizing algorithms.

Table 7.1 replicates the taxonomy of kriging model-assisted multi-objective optimization methods from Chapter 5, highlighting how the novel algorithms (scalarizing and non-scalarizing) in this chapter fit into the overall picture. Note that the ϵ-constraint method could alternatively have been used to develop the novel scalarizing algorithms in this section (instead of a weighted metric approach); such scalarizing algorithms remain to be explored.

TABLE 7.1: Taxonomy of kriging model-assisted multi-objective optimization methods.

	Method for selecting search points									
	Scalarized Problem: Single kriging model						**Non-Scalarized Problem**: Multiple kriging models			
zing d	**Non-tunable Utility Functions**: No parameters to set		**Tunable Utility Functions**: Parameters allowing the balance between exploration and exploitation to be altered				**Non-tunable Utility Functions**: No parameters to set			**Tunable Utility Functions**
			Non-target based: does not require an estimate of the optimum		Target-based: requires an estimate f^* of the optimum					
	Surface Minimum	Expected Improvement	Generalized Expected Improvement	Weighted Expected Improvement	Probability (of achieving)	Credibility (of hypothesis)	Pareto Points of Surfaces	Probability of Improvement	Expected Improvement	Enhanced Probability of Improvement
traint i										Section 7.4.1
ed			Section 7.3.4.1			Section 7.3.4.2				

The general procedure of the scalarizing algorithm for constrained multi-objective optimization is shown in Figure 7.1. The following sections describe the main stages of the algorithm in detail.

7.3.1 Experimental Design

As discussed in Section 6.2, several modern space-filling experimental designs now exist, and the choice of which to use is inevitably arbitrary. However, empirical studies (e. g. [108]) suggest that uniform designs tend to give better results. For this reason, a Hammersley sequence [96] is used for the initial selection of design vectors.

Following advice in the literature, the size of the initial experimental design, N_{exp}, is scaled according to the number of design variables d. In constrained optimization however, it turns out to be quite beneficial to have at least one example in the experimental design which is feasible (i. e. satisfies the constraints), and for this reason the experimental design size is also scaled according to the number of constraints,

$$N_{\text{exp}} = 6(d + J + 2K), \tag{7.1}$$

where J and K are the number of inequality and equality constraints respectively. Note, that in the rare case of insufficient successful designs being found from the experimental design to allow construction of an initial kriging model, further experimental design points are selected using higher order Hammersley sequences.

7.3.2 Dealing with Failed Designs

As discussed in Section 6.7, in optimal electromagnetic design the process of evaluating objective functions and constraints typically involves building a model, generating a finite element mesh, creating a database, running a solver and then performing some post-processing using CAD software. Each of these stages is prone to failure, meaning

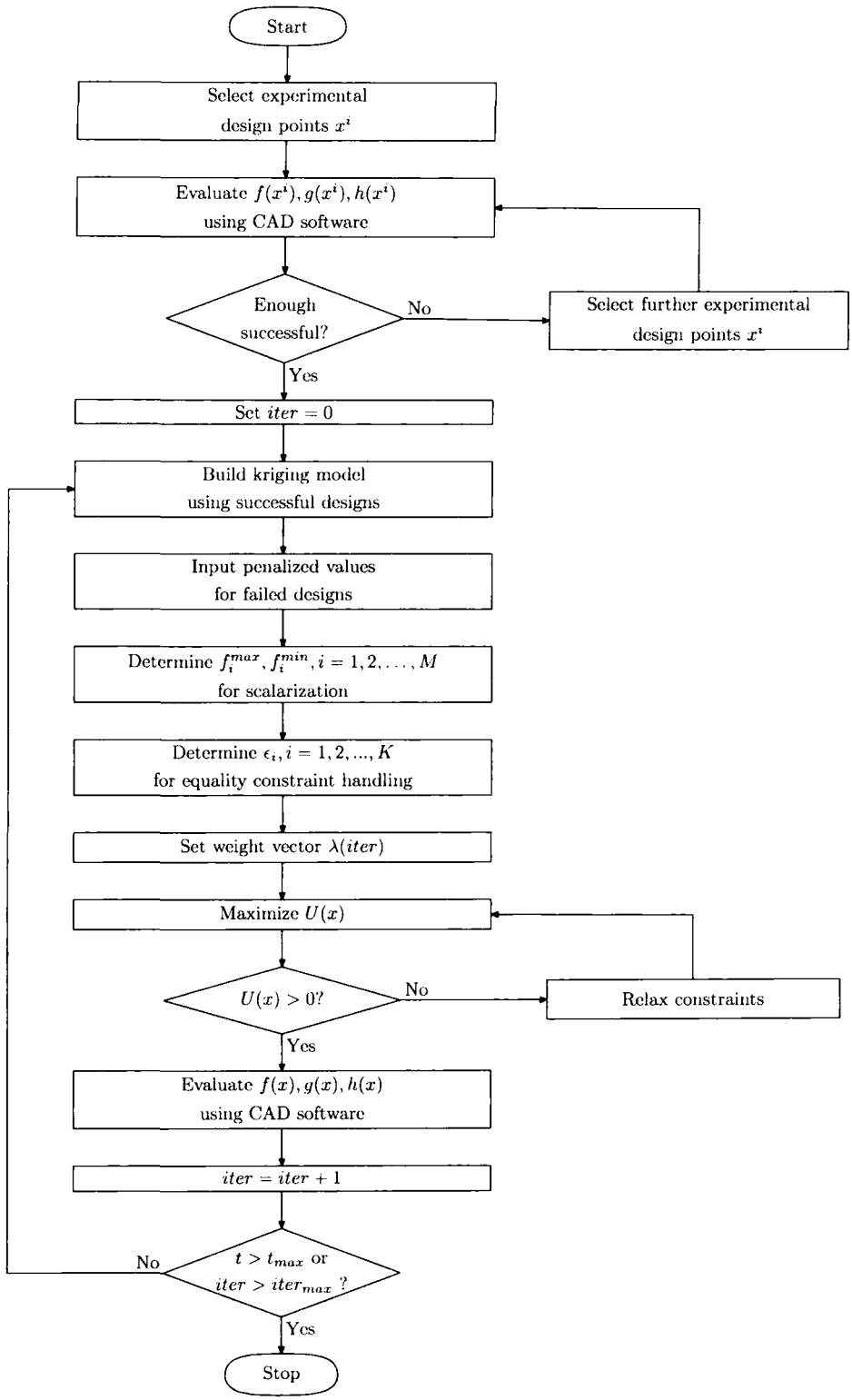


FIGURE 7.1: General procedure for the scalarizing multi-objective optimization algorithm.

that it is not always the case that information will be obtainable from a design selected for evaluation.

Following advice in [100], for each failed design $\mathbf{x}^*$, each objective function value f_i is penalized using a value of

$$f_i(\mathbf{x}^*) = \begin{cases} \hat{f}_i(\mathbf{x}^*) + s(\mathbf{x}^*) \text{ if } f_i \text{ is to be minimized} \\ \hat{f}_i(\mathbf{x}^*) - s(\mathbf{x}^*) \text{ if } f_i \text{ is to be maximized} \end{cases} \tag{7.2}$$

where $\hat{f}_i(\mathbf{x}^*)$ is the kriging prediction of $f_i(\mathbf{x}^*)$ based on the successful designs, and $s(\mathbf{x}^*)$ is the standard error in this prediction. The kriging predictions $\hat{g}_i(\mathbf{x}^*)$, $\hat{h}_i(\mathbf{x}^*)$ (again based on successful designs) are input for the constraint vectors of each failed design; penalization of the objective function values alone is deemed sufficient to guide the algorithm away from the regions with high probability of failure.

7.3.3 Scalarization

Once the experimental design stage is finished, and each of the failed designs penalized, the M objectives of the MOOP are normalized so that each objective function lies within the range [0,1], using

$$\bar{f}_i(\mathbf{x}) = \frac{f_i(\mathbf{x}) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \tag{7.3}$$

where $f_i^{\min}$, $f_i^{\max}$ are the true lower and upper limits of objective f_i respectively. If the true limits $f_i^{\min}$, $f_i^{\max}$ are unknown (as will often be the case), then the maximum and minimum values obtained so far are used. This leads to an ideal objective function value of $\mathbf{z}^* = (0, 0, \ldots, 0)^T$.

Once normalized, the objectives are combined using the augmented weighted Tchebycheff function [38]:

$$f_\lambda(\mathbf{x}) = \max_{j=1}^{M}(\lambda_j|\bar{f}_j(\mathbf{x}) - z_j^*|) + \rho \sum_{j=1}^{M} \lambda_j(|\bar{f}_j(\mathbf{x}) - z_j^*|) \tag{7.4}$$

with ρ set (arbitrarily) to 0.05. The normalized weighting vector $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_M]$ is randomly set at each iteration.

7.3.4 Selection of Design Vectors and Constraint Handling

With the original MOOP now transformed to a SOOP, any of the selection criteria from Chapter 4 may be used to select design vectors for evaluation [7]. In particular, two

possibilities are:

- $U(\mathbf{x})$ is the generalized expected improvement utility function. In this case, a cooling scheme may be used to vary the value of g as iterations proceed. This is discussed in Section 7.3.4.1.
- $U(\mathbf{x})$ is the credibility of hypothesis utility function [6, 9]. In this case, as each objective has been normalized, an ideal target value of $f^* = 0$ may be used at each iteration. This is discussed in Section 7.3.4.2.

To make the algorithm as versatile as possible, constraint handling techniques are included. Each equality constraint $h_i(\mathbf{x}) = 0$ is transformed to two separate inequality constraints, $|h_i(\mathbf{x})| \leq \epsilon_i$, where ϵ_i is taken to be a small percentage (x%) of the range of the values of h_i of the designs so far sampled,

$$\epsilon_i = \frac{x}{100}(h_i^{\max} - h_i^{\min}). \tag{7.5}$$

In this implementation, x is set to be 5. This results in a MOOP with $J + 2K$ inequality constraints. These are dealt with using the constrained utility function method discussed in Section 6.8.4.

7.3.4.1 Generalized ParEGO Algorithm

In [45], the expected improvement criterion was used to select design vectors, after the scalarization process described in Section 7.3.3. As discussed above, a natural generalization of this algorithm is to use instead the generalized expected improvement criterion, along with a constraint handling method such as the constrained utility function method:

$$U(\mathbf{x}) = \begin{cases} s^g \sum_{k=0}^{g} (-1)^k \left(\frac{g!}{k!(g-k)!}\right) \left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)^{g-k} T_k \\ \text{if } \hat{g}_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, J \text{ and} \\ |\hat{h}_i(\mathbf{x})| \leq \epsilon_i, i = 1, 2, \ldots, K \\ \\ 0 \\ \text{if } \exists i \text{ such that } \hat{g}_i(\mathbf{x}) > 0 \text{ or } |\hat{h}_i(\mathbf{x})| > \epsilon_i. \end{cases} \tag{7.6}$$

where the terms are as defined in Section 4.3.1. This algorithm is investigated in [7], and is the default algorithm for constrained (and unconstrained) multi-objective optimization in the Vector Fields software, as discussed in Section 8.3.4.1.

7.3.4.2 Scalarizing One-Stage Algorithm

One attractive utility function for selecting design vectors in single-objective optimization was the credibility of hypotheses function. Surprisingly, it has received little attention in the literature; in particular, it has not been used for multi-objective optimization. However, by combining it with the constrained utility function method, it can easily be used for constrained multi-objective optimization. The utility function $U(\mathbf{x})$ becomes:

$$U(\mathbf{x}) = \begin{cases} \frac{1}{(2\pi)^{N/2}(\sigma^2)^{N/2}|\mathbf{C}|^{1/2}} \exp\left(\frac{-(\mathbf{y}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{y}-\mathbf{m})}{2\sigma^2}\right) \\ \text{if } \hat{g}_i(\mathbf{x}) \leq 0, i = 1,2,\ldots,J \text{ and} \\ |\hat{h}_i(\mathbf{x})| \leq \epsilon_i, i = 1,2,\ldots,K \\ \\ 0 \\ \text{if } \exists i \text{ such that } \hat{g}_i(\mathbf{x}) > 0 \text{ or } |\hat{h}_i(\mathbf{x})| > \epsilon_i. \end{cases} \tag{7.7}$$

where the terms are as defined in Section 4.4.2.1. Note that the conditional likelihood (rather than the conditional log-likelihood) has been used, as this allows a penalty value of 0 (rather than $-\infty$) to be used when a constraint is predicted to be violated. This algorithm is investigated in [9]. Whilst the hybrid one-then-two stage algorithm in the previous section represented the first time the one-stage methodology had been used for selection of design vectors in electromagnetic design optimization, this algorithm represents the first time it has been used for multi-objective optimization (in any discipline).

7.3.4.3 Constraint Relaxation

In each of the two scalarizing algorithms proposed, the design vector which maximizes U is selected for evaluation. The method chosen for dealing with constraints suffers in that it is possible for $U(\mathbf{x}) = 0\ \forall \mathbf{x}$ (as a consequence of at least one constraint being predicted to be violated for every $\mathbf{x}$); when this happens, the constraints are gradually relaxed, until a design vector which yields a non-zero value of U is found, and this is then selected for evaluation.

7.3.5 Termination Criteria

After evaluation of the design vector which maximizes the utility function U, it is added to the set of examples. The algorithm then repenalizes the failed designs, using a kriging model based on the updated set of examples; objectives are then renormalized and

rescalarized (using a different weighting vector λ), and another design vector is selected using the updated utility function U. This procedure continues until one of two possible termination criteria is met: either a maximum time limit $t_{\max}$ has been exceeded, or a maximum number of iterations $iter_{\max}$ have been carried out.

7.4 Kriging-Assisted Non-Scalarizing Algorithms for Constrained Multi-Objective Optimization

In Section 5.3, a drawback of the probability of improvement utility function (itself proposed independently as a result of this research [109]) was identified, namely that it does not favour large improvements. This was overcome by extending the notion of expected improvement to multi-objective optimization, however this approach was rather cumbersome. Furthermore, no control was available over the utility functions in non-scalarizing methods: no parameters were available to fine-tune the search. The goal of this section is to propose a non-scalarizing method which allows this control.

7.4.1 Motivation

The multi-objective algorithm proposed in this section differs from those in Section 7.3 as it is non-scalarizing. The idea behind the algorithm is to provide a means of controlling where exactly we wish the design vectors selected for evaluation to map to in objective function space. The theory developed here is for the two-objective case, but is extensible to an arbitrary number of objectives.

7.4.2 Algorithm (Two-Objective Case)

For each solution $\mathbf{z}$ to a MOOP, three different types of region can be identified in objective function space: regions in which other solutions dominate $\mathbf{z}$, regions in which other solutions are equivalent to $\mathbf{z}$, and regions in which other solutions are dominated by (i.e. worse than) $\mathbf{z}$. Such regions are shown for two solutions A and B in an objective function space which has two objectives f_1 and f_2 (both to be minimized) in Figure 7.2. Naturally these regions overlap: some regions yield improvements on both A and B; other regions only dominate one of A or B; other regions are equivalent to one solution, but better than the other; and so on .... This leads to the notion of there being different levels of improvement possible in MOOPs, depending on the number of solutions dominated by a new solution.

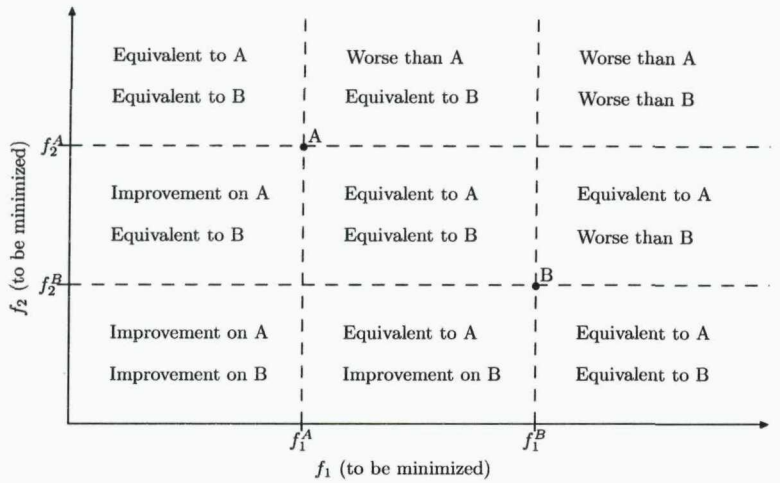


FIGURE 7.2: Regions of improvement, equivalence and detriment for two solutions in a two-objective problem.

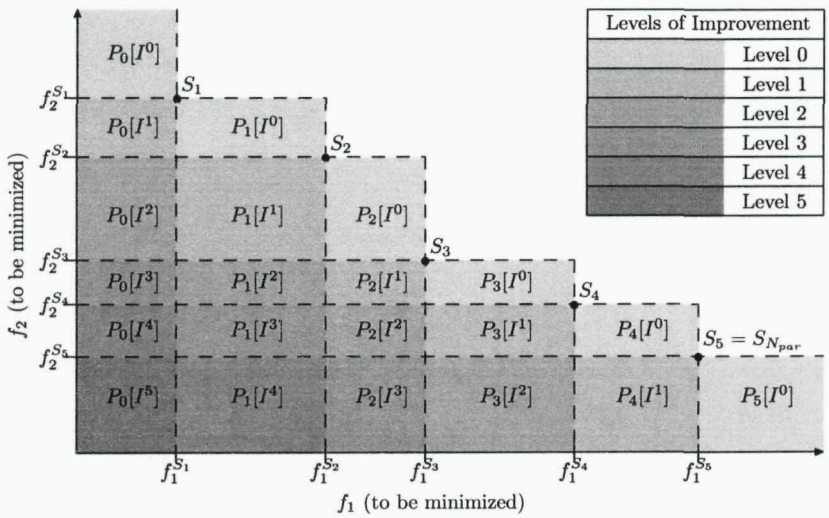


FIGURE 7.3: Probability of improvement levels for $N_{\text{par}} = 5$ Pareto solutions for a two-objective problem.

In general, when N_{par} Pareto-optimal solutions have been identified, $k = N_{\text{par}}$ natural levels of improvement may be defined, where the k^{th} level of improvement yields a solution which dominates *exactly* k of the existing Pareto-optimal solutions. In addition, a level of equivalence may also be defined ($k = 0$), which yields an additional Pareto-optimal solution which does not dominate any of the existing Pareto-optimal solutions (i.e a design vector which maps to the region labeled 'Equivalent Designs' in Figure 5.8). These levels of improvement are shown in Figure 7.3 for the $N_{\text{par}} = 5$ Pareto solutions considered earlier in Chapter 5.

As can be seen, in the two-objective case, for an improvement level k, a design vector may map to $N_{\text{par}} - k + 1$ regions of objective function space. Denoting $P(I^k(\mathbf{x}))$ as the probability that an unknown design vector $\mathbf{x}$ yields a level of improvement k (i.e. it dominates exactly k existing Pareto-optimal solutions), $P_i(I^k(\mathbf{x}))$ as the probability that design vector $\mathbf{x}$ will dominate the k Pareto solutions $S_{i+1}, S_{i+2}, \ldots, S_{i+k}$ (these sub-regions are labeled in Figure 7.3), and defining

$$\Phi_1^i(\mathbf{x}) := \Phi\left(\frac{f_1^{S_i} - \hat{f}_1(\mathbf{x})}{s_1(\mathbf{x})}\right) \tag{7.8}$$

$$\Phi_2^i(\mathbf{x}) := \Phi\left(\frac{f_2^{S_i} - \hat{f}_2(\mathbf{x})}{s_2(\mathbf{x})}\right) \tag{7.9}$$

with

$$\Phi_1^0(\mathbf{x}) := 0 \tag{7.10}$$

$$\Phi_1^{N_{\text{par}}+1}(\mathbf{x}) := 1 \tag{7.11}$$

$$\Phi_2^0(\mathbf{x}) := 1 \tag{7.12}$$

$$\Phi_2^{N_{\text{par}}+1}(\mathbf{x}) := 0 \tag{7.13}$$

where $\hat{f}_1(\cdot)$ and $s_1(\cdot)$ are the kriging predictions and standard errors for the first objective function respectively (similarly for the second objective function), and $f_1^{S_i}$ is the first objective function value of the i^{th} Pareto solution (similarly for the second objective function), then:

$$P(I^k(\mathbf{x})) = \sum_{i=0}^{N_{\text{par}}-k} P_i(I^k(\mathbf{x})) \tag{7.14}$$

$$= \sum_{i=0}^{N_{\text{par}}-k} \left(\Phi_1^{i+1}(\mathbf{x}) - \Phi_1^i(\mathbf{x})\right)\left(\Phi_2^{k+i}(\mathbf{x}) - \Phi_2^{k+i+1}(\mathbf{x})\right). \tag{7.15}$$

Furthermore, denoting by $P^*(I^k(\mathbf{x}))$ the probability that $\mathbf{x}$ will dominate *at least* k existing Pareto-optimal solutions, then

$$P^*(I^k(\mathbf{x})) = \sum_{j=k}^{N_{\text{par}}} \sum_{i=0}^{N_{\text{par}}-j} \left(\Phi_1^{i+1}(\mathbf{x}) - \Phi_1^i(\mathbf{x})\right)\left(\Phi_2^{j+i}(\mathbf{x}) - \Phi_2^{j+i+1}(\mathbf{x})\right). \tag{7.16}$$

By grouping together the maximizers of P (or P^*) over the different levels of improvement $k = 1, 2, \ldots, N_{\text{par}}$, and selecting one representative design vector from each group, it can be seen that Equation 7.15 and Equation 7.16 are two multi-objective equivalents of the 'enhanced probability of improvement' in single-objective optimization, for the case of two objectives. The method is extensible to higher numbers of objectives.

This method has many attractive features. Firstly, as with other non-scalarizing methods, it does not require objectives to be normalized, so upper and lower limits for the objectives do not need to be estimated. Secondly, unique to this method, effort can be concentrated on searching for design vectors in specific regions of objective function space, leading to a more diverse Pareto-optimal front. This simply involves locating the design vector which maximizes the integral corresponding to the desired region of objective function space. Thirdly, the method is naturally parallelizable, as multiple design vectors, each most probable for mapping into different promising regions of objective function space, can be identified and evaluated each iteration.

7.4.3 Constraint Handling

The non-scalarizing method may be extended to constrained optimization, by multiplying the probability of improvement by the probability of satisfying the constraints [10], so e. g. Equation 7.15 becomes

$$\begin{aligned} P(I^k(\mathbf{x})) &= \prod_{i=1}^{J} \Phi\Big(-\frac{\hat{g}_i(\mathbf{x})}{s_{g_i}(\mathbf{x})}\Big) \prod_{j=1}^{K} \Big[\Phi\Big(\frac{\epsilon_j - \hat{h}_j(\mathbf{x})}{s_{h_j}(\mathbf{x})}\Big) + \Phi\Big(\frac{\hat{h}_j(\mathbf{x}) - \epsilon_j}{s_{h_j}(\mathbf{x})}\Big)\Big] \\ &\quad \times \sum_{i=0}^{N_{\text{par}}-k} \big(\Phi_1^{i+1}(\mathbf{x}) - \Phi_1^{i}(\mathbf{x})\big)\big(\Phi_2^{k+i}(\mathbf{x}) - \Phi_2^{k+i+1}(\mathbf{x})\big). \end{aligned} \tag{7.17}$$

The design vector which maximizes this is that which is most probable to dominate k existing non-dominated solutions and be feasible. Alternatively, the constrained utility function method may be used to set the value of the utility function in Equation 7.15 or Equation 7.16 equal to zero when the constraints are predicted to be violated. These methods remain to be fully investigated in the literature.

Chapter 8

Implementation

8.1 Introduction

As discussed in Chapter 1, the ultimate goal of this research is a practical optimization tool for use in Vector Fields' electromagnetic design software, Opera. This Chapter discusses the main details of how this tool was implemented, including how it communicates with the rest of Opera. User guides for the new software introduced are given in Appendices B, C and D.

8.2 The Opera Manager

8.2.1 Introduction

A new front-end piece of software, the 'Opera Manager', was written, allowing access to all of the main Opera programs (the Opera-2d Pre and Post-Processor, the Opera-3d Modeller, the Opera-3d Pre-Processor and the Opera-3d Post-Processor), as well as the documentation. It presents the user with a file browser display of the current project folder (which contains the Opera files currently in use) so that known file types can be opened using a double mouse click or the right mouse button context menu, and a 'Batch Processor' window, as shown in Figure 8.1. Full details on using the Opera Manager may be found in Appendix B.

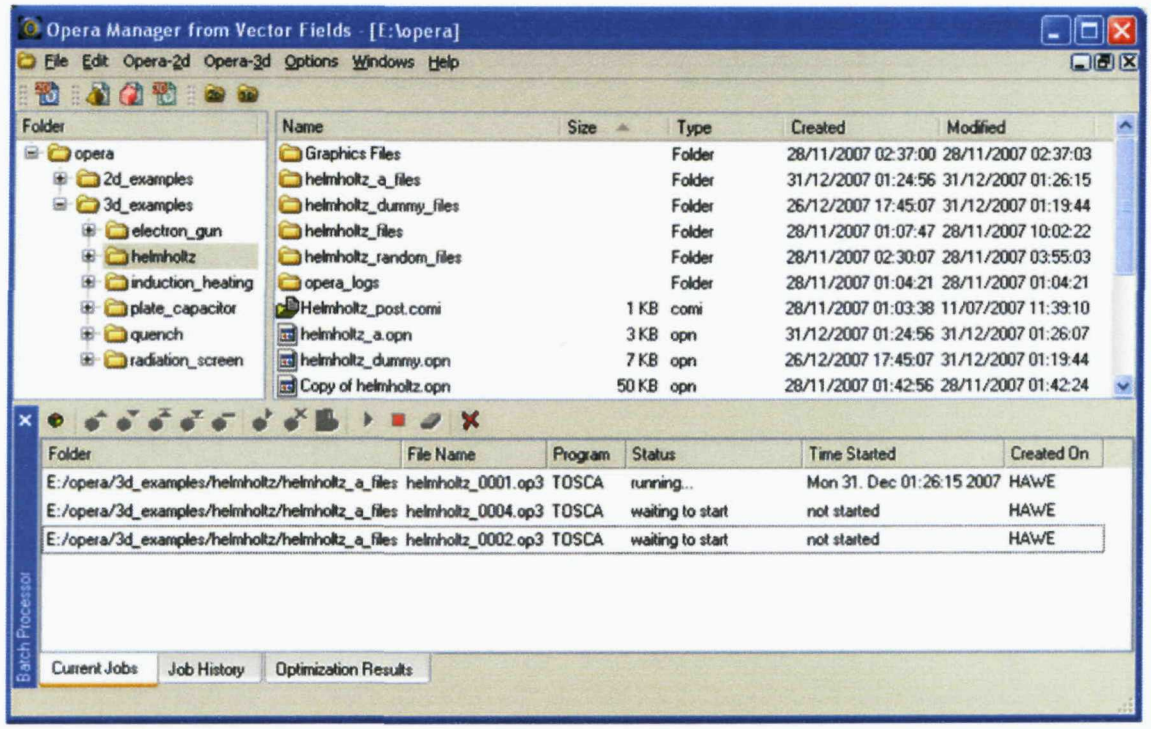


FIGURE 8.1: The Opera Manager, with the Batch Processor docked at the bottom.

8.2.2 The Batch Processor

Using the Opera Manager, unsolved Opera-3d databases (*.op3 files) and Opera-2d data files (*.op2 files) may be sent to a 'Batch Processor' for solving. Handling jobs using a Batch Processor is beneficial, as it allows them to be built into a queue (which may be reordered) before starting. Options may then be imposed on the queue as a whole: for example, the maximum number of jobs to run simultaneously may be specified, or the default CPU priority for each job in the queue may be altered. Full details on using the Batch Processor may be found in Appendix C.

8.2.2.1 Batch Folders and Parallelization

Each Opera Manager has one (and only one) Batch Processor associated with it. Associated with each Batch Processor however are two 'batch folders': a local batch folder and a common batch folder, each of which are regularly monitored. Each job sent to the batch queue is represented by a batch file (written in xml) in one of these two folders. As a common batch folder may be shared between multiple Batch Processors (running on different machines), creating several jobs in a common batch folder allows them to be processed in parallel on multiple machines.

8.2.2.2 Optimization Jobs

Jobs sent to the Batch Processor from within the Opera Manager involve only one stage: solving the Opera *.op2 data file or *.op3 database. During optimization however, each job generally requires three stages:

- Rebuilding and remeshing a parameterized model (the reset parameters being the design variables of the problem).
- Creating a database from the model, and solving with an appropriate solver.
- Post-processing the solved database, to evaluate values of interest.

The batch file representing an optimization job therefore contains information about how to carry out each of these stages. The results of the post-processing are then written back to the optimization batch file by the Batch Processor, to be used by the optimization tool, described in the following section.

8.3 The Optimization Tool

8.3.1 Introduction

The optimization tool consists of the following main components:

- An optimization dialog, allowing optimization problems to be defined from Opera *.op2 data files or *.op3 databases.
- An optimization file, which stores the definition of an optimization problem set up using the optimization dialog, along with results as they are obtained.
- An optimizer executable, which reads the definition of an optimization problem from the optimization file, and writes design vectors to evaluate back to it.
- An 'optimizer controller' object, which handles communication between the optimization dialog, the optimizer executable, the Opera Manager and the Batch Processor.

The optimizer controller object is 'owned' by the Opera Manager (in the sense that it is an object in C++, and parented to the main Opera Manager object). Communication between these four components and the Batch Processor occurs at twelve stages during the entire optimization process, as shown in Figure 8.2.

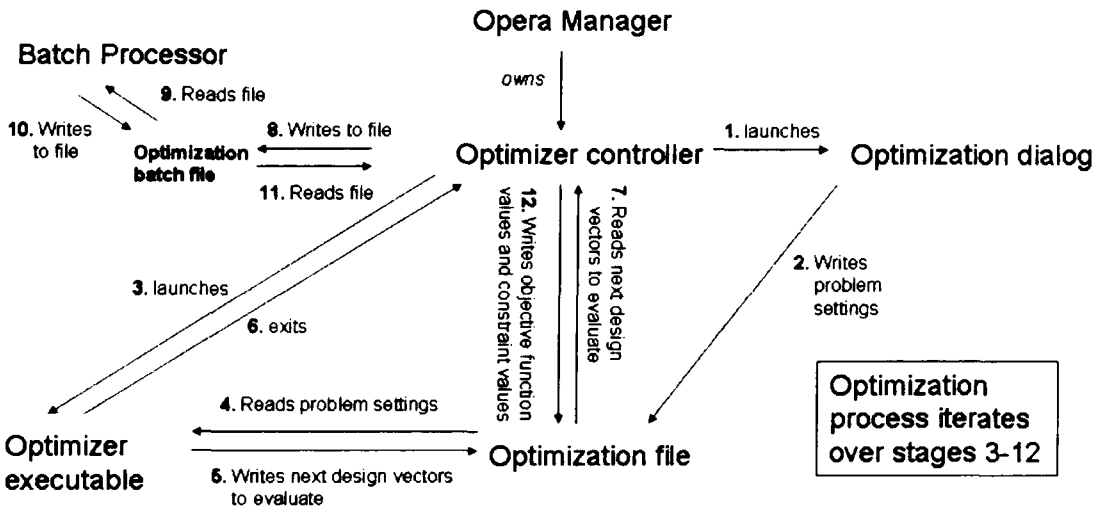


FIGURE 8.2: Communications between the Opera Manager, the optimizer controller, the optimizer executable, and the Batch Processor.

1. The optimization process begins by the user invoking an optimization dialog from an *.op2 data file or an *.op3 database in the Opera Manager. The optimizer controller launches the dialog for the user to define the problem.

2. After defining the problem in the optimization dialog, the settings are written to the optimization file.

3. The optimizer controller then launches the optimizer executable, supplying the optimization file as an argument.

4. The optimizer reads the settings from the optimization file, along with all previous results, and runs an appropriate algorithm to determine where to evaluate next.

5. The optimizer then writes the design vectors to evaluate next back to the optimization file.

6. The optimizer executable exits, letting the optimizer controller know it has done so.

7. The optimizer controller reads the design vectors to evaluate from the optimization file.

8. The optimizer controller creates an optimization batch file in the appropriate batch folder for the Batch Processor.

9. The Batch Processor reads the optimization batch file, and processes the optimization job.

10. When the optimization job has finished, the values of interest (the optimization outputs - see Section 8.3.2.2 below) are written back to the optimization batch file.

11. The optimizer controller reads the optimization outputs from the optimization batch file.

12. The optimization output values, along with the design variable values, are then written back - as a result - to the optimization file.

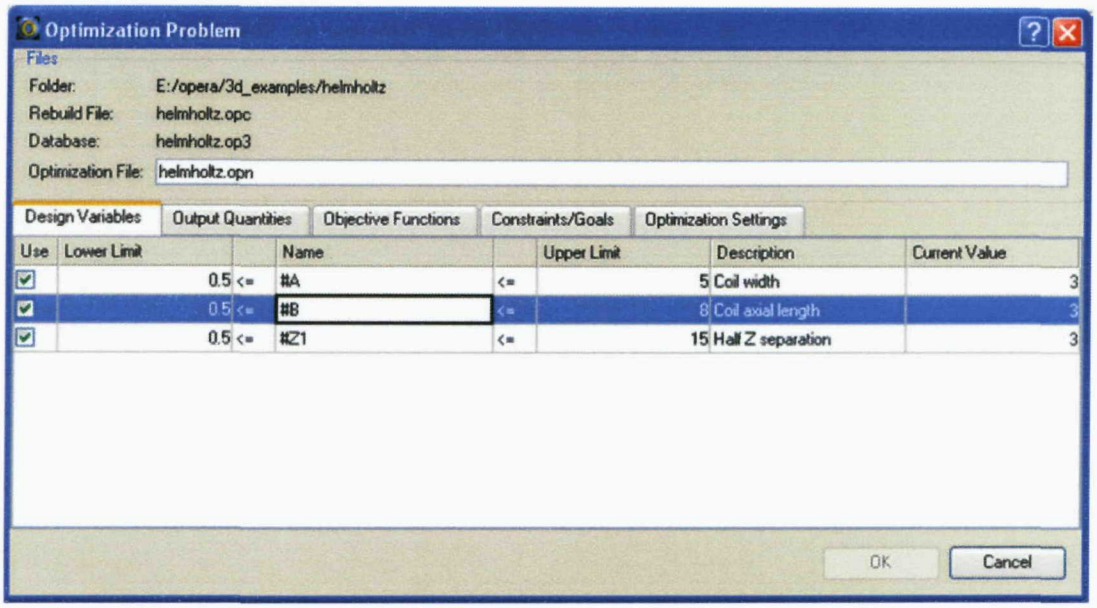
The process then loops back to step 3, with the optimizer controller launching the optimizer executable. Each time an iteration occurs, the optimizer is reading an extra example from the optimization file. The process continues until some user specified termination criterion is met. The rest of this section describes the different components of the optimization tool in more detail.

8.3.2 Optimization Dialog

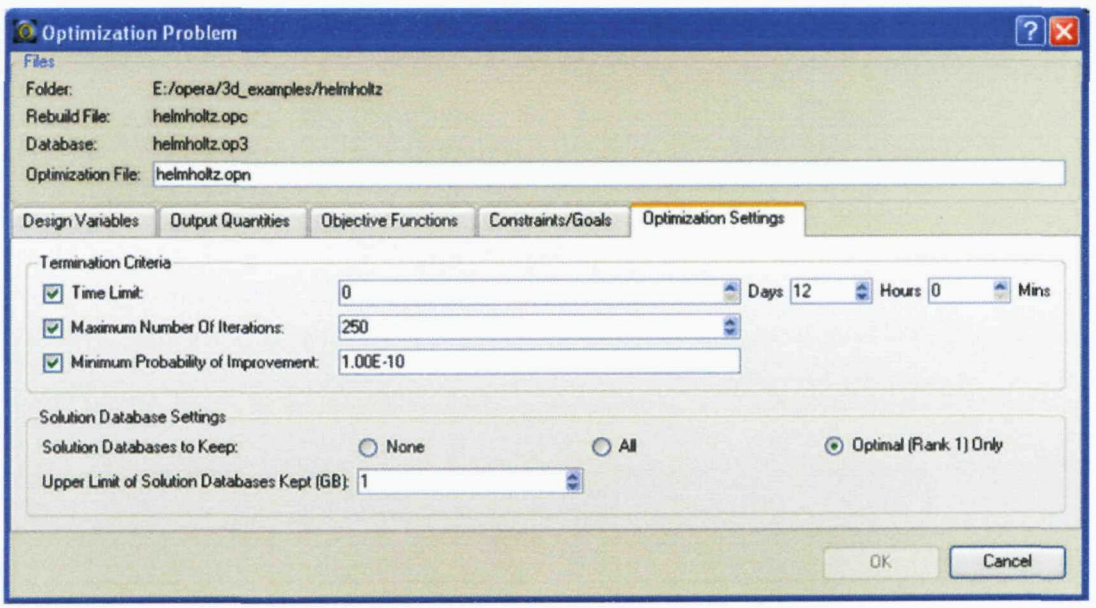
To define an optimization problem in Opera, the following are needed:

- A way to specify the design variables in an Opera model, and their allowed ranges.
- A way to specify methods of calculating the objective functions (i. e. ways of computing each f_i in Equation 2.4).
- A way to specify methods of calculating the constraint functions (i. e. ways of computing each g_i, h_i in Equation 2.4).

A dialog was designed to allow each of these to be defined from an Opera-3d database or an Opera-2d data file. It consists of several different tabs (two of which are shown in Figure 8.3) allowing design variables, objectives, constraints, and some general user settings to be defined. The remainder of this subsection describes briefly how this was implemented; full details of using the dialog to set up an optimization problem (in both Opera-2d and Opera-3d) may be found in Appendix D.



(a) Setting up design variables



(b) Specifying user settings

FIGURE 8.3: Examples of the optimization dialog in use.

8.3.2.1 Specifying Design Variables: Model Dimensions

A new type of user variable - known as a 'model dimension' - was introduced into Opera. When invoking the optimization dialog from an Opera database, the model dimensions present in the corresponding model are identified, and made available for setting as design variables. Model dimensions selected for use as design variables require a lower and upper limit to be specified, whilst those not selected require a constant value to be assigned.

8.3.2.2 Specifying Objectives and Constraints: Optimization Outputs

To specify the objectives and constraints in a problem, a *.comi file [1] must be supplied, containing all the commands needed to evaluate the objective and constraint function values and store them in user variables, which are flagged as 'optimization outputs'. Once specified, each optimization output is then available through two separate tabs for use as an objective (to be minimized or maximized), and as a term in a constraint expressions.

8.3.2.3 User Settings

User settings were deliberately kept to a minimum. Two main criteria are available to be set: the termination criteria for the optimization algorithm, and the preferences regarding what to do with the Opera files created during the optimization process. Further details of these settings may be found in Appendix D.

8.3.3 Optimization File

After defining an optimization problem through the optimization dialog, the settings of the problem are written to an 'optimization file' (written in xml). This contains all of the details needed to state the optimization problem in the form of Equation 2.4. When the optimization process begins, the results are also written to this file. As can be seen in Figure 8.2, this file is central to the communication between the different parts of the optimization tool.

[1] Each action carried out in the Opera-3d Post-Processor (and Modeller) and Opera-2d Pre and Post-Processor is represented by a specific command. *.comi files contain sequences of these commands, thus allowing sequences of actions to be carried out in a solved database, enabling values of interest to be evaluated automatically.

8.3.4 Optimizer Executable

The optimization problem and results (if any) contained in the optimization file are read by an optimizer executable, which runs one iteration of an appropriate algorithm to determine which design vector(s) to evaluate next [2]. These are then written back to the optimization file, and used by the optimizer controller to create optimization batch jobs for the Batch Processor to process.

8.3.4.1 Default Methods

In previous chapters, a range of methods has been described which can be used for efficient constrained (and unconstrained) single and multi-objective optimization. Many of these were included in the optimizer program, however, as mentioned in Section 1.2.2.1, access to all available methods is detrimental in terms of ease-of-use of the software. Unless the end user of the software knows and understands each of the methods available, they will not wish to be left with the decision of which method to optimize their designs with. Therefore some choices need to be made as regards which algorithms to use by default. The default methods chosen are summarized in Table 8.1, and explanations behind their choice are given below.

The experimental design chosen was a Hammersley Sequence, as it gives the most uniform distribution of design vectors. The size of the Hammersley Sequence depends on the number of design variables d, the number of inequality constraints J, and the number of equality constraints K.

The utility function chosen was the Generalized Expected Improvement. It provides a level of control over the balance between exploration and exploitation (which the Expected Improvement does not), and is non-target based, and so does not require an estimate of the optimum from the user. One-stage methods were not chosen as default due to their inherently high computational cost, and because they require a target to be set by the user.

The inequality constraint handling technique chosen was the constrained utility function method. It is simpler to implement than the cumbersome expected violation method, does not prevent exploration of the constraint boundary like the probability method, and does not require the setting of the relative importance between constraints, as the penalty method does.

[2] If no previous results are provided, the designs suggested are an experimental design.

TABLE 8.1: Choice of default methods in the optimizer.

Method	Default Setting
Experimental Design	Hammersley Sequence
Experimental Design Size	$6(d + J + 2K)$
Utility Function	Generalized Expected Improvement
Utility Function Parameters	Cooling scheme
Inequality Constraint Handling Technique	Constrained utility function method
Equality Constraint Handling Technique	Transformation to two inequalities
Scalarizing Method	Augmented weighted Tchebycheff function
Maximum Number of Iterations	$18d$

The equality constraint handling technique chosen was the method which transforms each equality constraint to two inequality constraints. This approach places more emphasis on each equality constraint being satisfied than the method which transforms each equality constraint to an objective.

The scalarizing method chosen was the augmented weighted Tchebycheff function, as it allowed solutions on concave parts of Pareto-optimal fronts to be located, something which is not true of other methods.

8.3.4.2 Advanced Control

Access to the other (non-default) methods was made available through use of an advanced control setting. This was essentially a hidden feature; its purpose was to allow users - already familiar with research into kriging-assisted optimization methods - to use the optimizer as a research tool. The choice of methods comes at two stages: the experimental design stage, and then subsequently at each iteration.

At the experimental design stage, the dialog shown in Figure 8.4 is invoked. It allows choice between two experimental designs (a Hammersley Sequence, and a Random Design), and allows the size of the design to be explicitly set.

Following the experimental design, the optimizer proceeds by selecting design vectors using utility functions. If the problem is single-objective, the dialog shown in Figure 8.5(a) is invoked at each iteration. It allows the choice of the following utility functions (described in Chapter 4), including setting of any parameters:

- Optimum of the kriging surface ('Strawman' approach).
- Expected Improvement (EI).

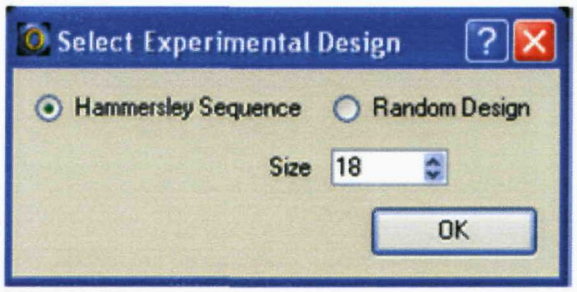


FIGURE 8.4: Experimental design dialog.

- Generalized Expected Improvement (GEI), including the value of g.
- Weighted Expected Improvement (WEI), including the value of w.
- Probability of Improvement (POI), including either the value of α or T.
- Credibility of Hypothesis (CH), including either the value of α or T.

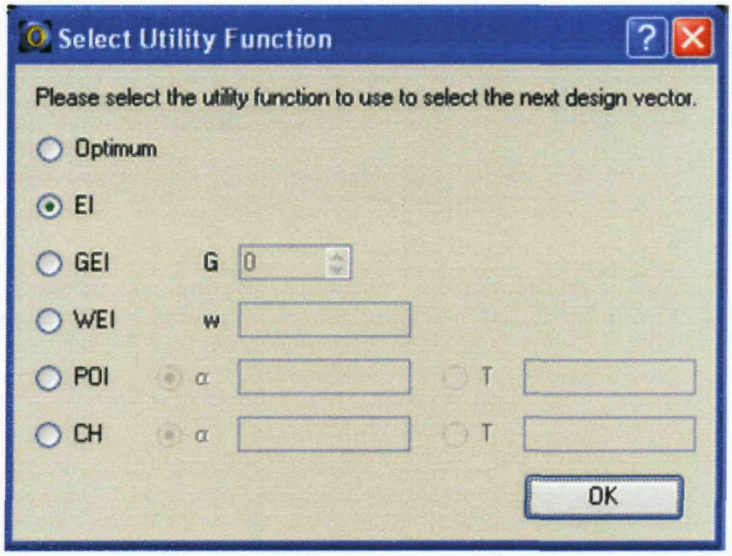
If the problem is multi-objective, the dialog shown in Figure 8.5(b) is invoked at each iteration. It provides extra information regarding the scalarizing function used to transform the MOOP to a SOOP, including values of each of the weights used.

8.3.5 Presentation of Results

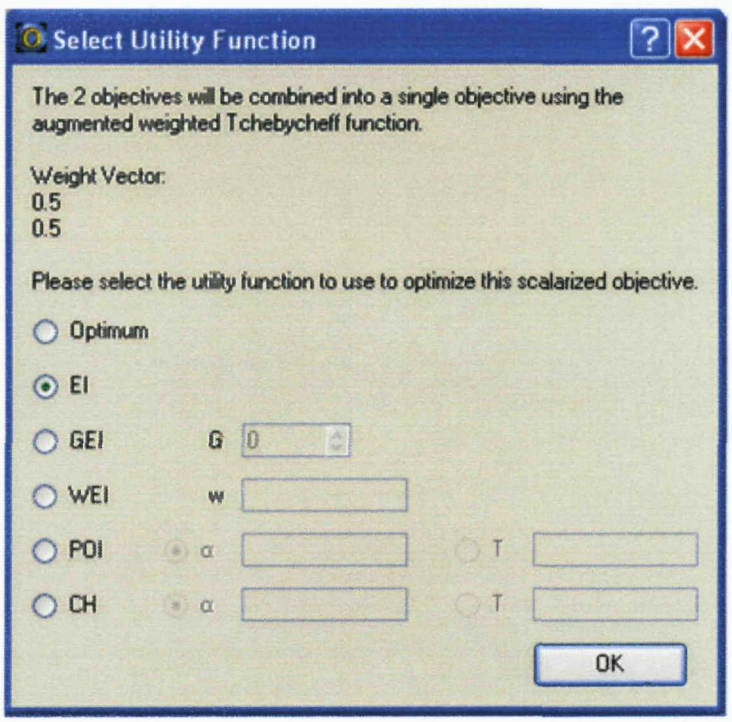
Results are presented in a tab-pane (as shown in Figure 8.6) which forms part of the Batch Processor window. As each result is obtained, it is displayed as a new row in this tab-pane. It is possible that many details exist for each result, so only the most important are displayed, namely:

- The rank of the example (as determined by a non-dominated sorting algorithm).
- The objective function values.
- A summary of constraint satisfaction.
- Iteration number.
- Name of the corresponding Opera database.

A context menu for the tab-pane, invoked by right-clicking on a result, allows other details, such as the values of the design variables, and the values of each of the constraint expressions, to be displayed in a dialog as shown in Figure 8.7.



(a) Advanced utility function dialog (single-objective optimization).



(b) Advanced utility function dialog (multi-objective optimization).

FIGURE 8.5: The advanced utility function dialog.

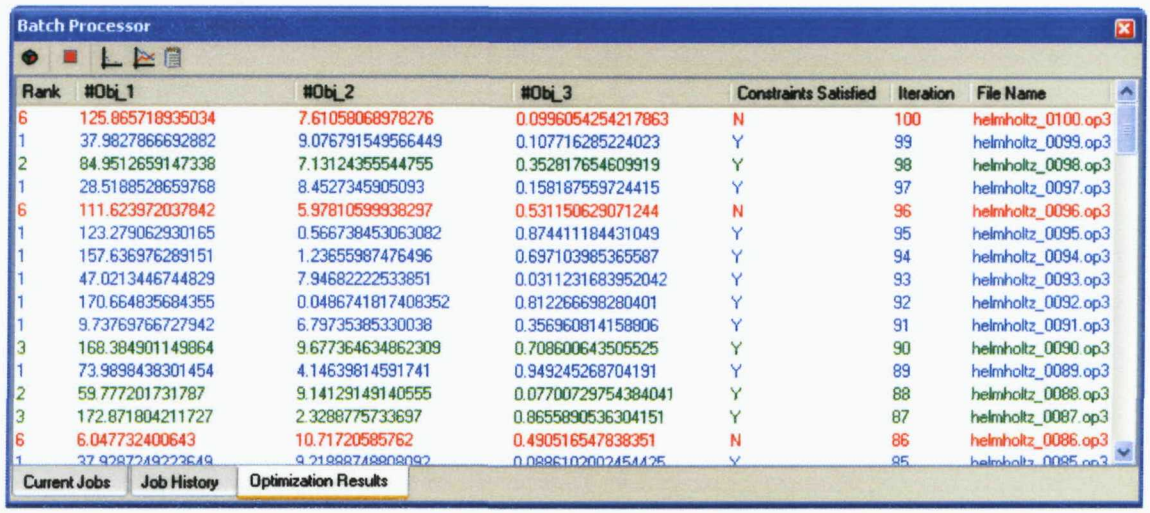


FIGURE 8.6: Display of optimization results in the Batch Processor.

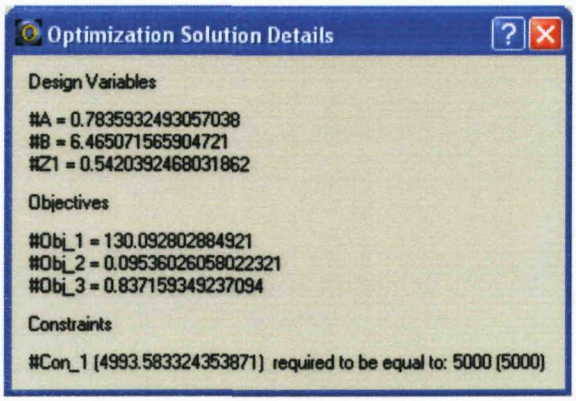


FIGURE 8.7: Dialog showing full details of an optimization result.

Results may be ordered by any of the column headings. To make the tab-pane clearer, results are colour coded, with Pareto-optimal results displayed in blue, other feasible results displayed in green, infeasible results displayed in red, and failed iterations displayed in grey.

Graphical output was also made possible, with graphs being plotted for:

- Normalized objective function values versus iteration number,
- Constraint values versus iteration number,
- Design variables in design variable space (2d and 3d only), and
- Objective function values in objective function space (2d and 3d only).

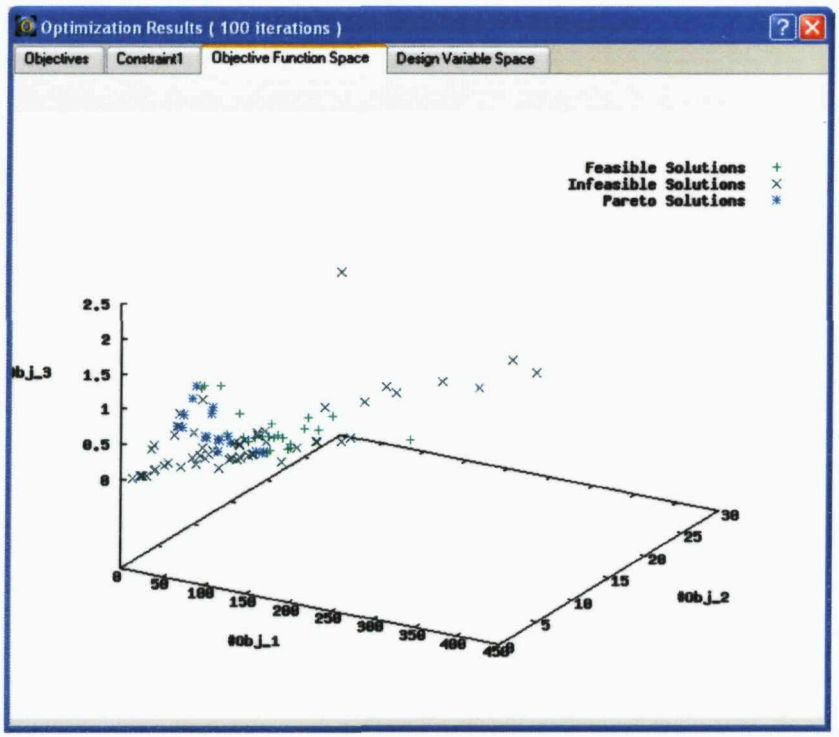


FIGURE 8.8: Example graphical output of the optimization tool.

An example of the graphical output screen is shown in Figure 8.8.

Part IV

Results and Conclusion

Chapter 9

Results

9.1 Introduction

In this chapter, the optimization tool described in the previous chapter is demonstrated on four different electromagnetic design problems in Opera. The problems are chosen to span the four possibilities of unconstrained SOOPs, constrained SOOPs, unconstrained MOOPs, and most difficult of all - and fulfilling one of the main aims of this thesis - constrained MOOPs.

The results given are for the default algorithms in the optimizer. Comparisons with a random search algorithm show the default choices to be effective. A comparison of all the algorithms possible with the optimization tool (through use of the advanced algorithms), is not feasible, due to the huge range of algorithms available (especially when different cooling schemes are considered with the tunable utility functions). Results for other algorithms proposed in this thesis (but not used in this chapter) may be found in e. g. [4, 11].

9.2 Single-Objective Unconstrained Optimization of a Plate Capacitor

9.2.1 Problem Definition

The voltage V (Volts) on the two outer plates of a plate capacitor, as shown in Figure 9.1, along with their perpendicular distances d_1, d_2 (mm) from the inner earthed plate, are allowed to vary in order to achieve a target capacitance. The model is set up as described in the Vector Fields Opera-3d User Guide [110], and solved using the TOSCA

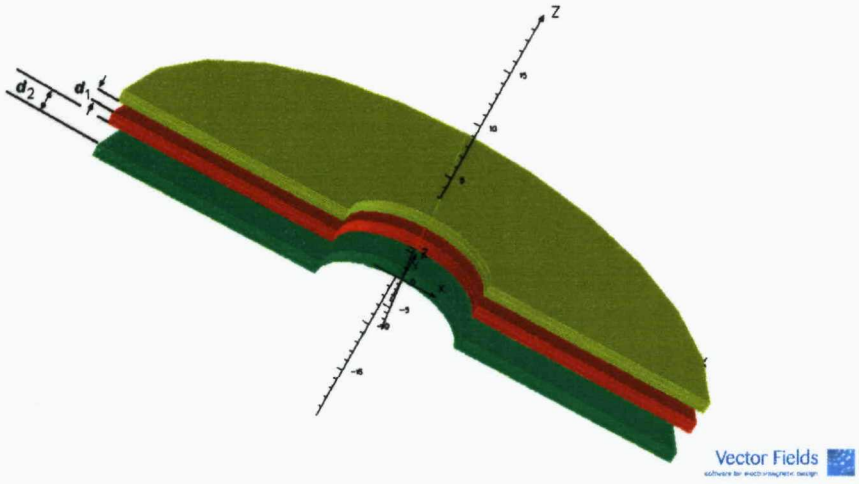


FIGURE 9.1: Plate capacitor arrangement in Opera-3d.

electrostatic solver. By calculating the stored energy E in the Opera-3d Post-Processor, the capacitance C of each design may be found using the relation

$$E = \frac{1}{2}CV^2. \tag{9.1}$$

A target capacitance of 22μF is chosen, and so the unconstrained SOOP to be solved is:

$$\begin{aligned} \text{Minimize} \quad & |C(d_1, d_2, V)(\mu\text{F}) - 22| \\ \text{with} \quad & 1.5 \leq d_1 \leq 4 \text{ mm} \\ & 1.5 \leq d_2 \leq 4 \text{ mm} \\ & 5 \leq V \leq 25 \text{ V} \end{aligned} \tag{9.2}$$

9.2.2 Results

A termination criteria of 35 iterations was chosen. The variation of the objective function is shown in Figure 9.2, and the positions in design variable space of the 35 iterations are shown in Figure 9.3.

The best solution found (after 33 iterations) had a capacitance of 22.0468 μF, thus giving an objective function value of 0.0468. By comparison, the best solution found by a random search algorithm of 100 iterations had a capacitance of 21.4906 μF giving an objective function value of 0.5094 (found after 77 iterations). This is shown as a target in Figure 9.2, which the optimizer outperforms after 31 iterations.

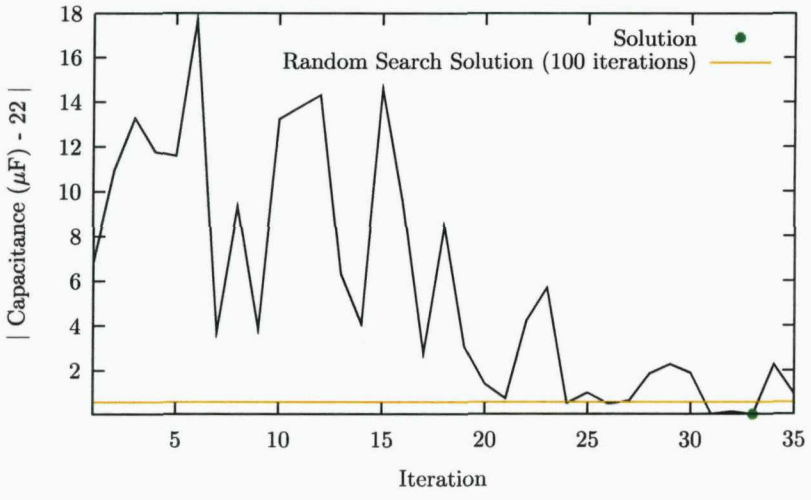


FIGURE 9.2: Variation of the residual capacitance during the optimization search.

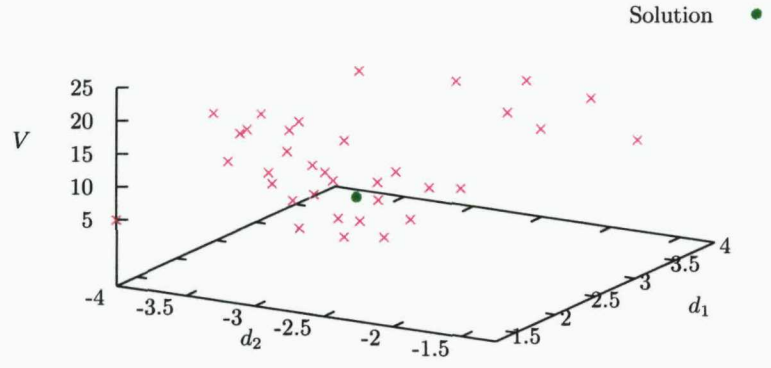


FIGURE 9.3: Positions in design variable space of iterations of the Optimizer for the plate capacitor problem.

9.3 Single-Objective Constrained Optimization of a Coil, Former and Disc

9.3.1 Problem Definition

Figure 9.4 shows a three dimensional cut view of a coil wound around an 'E' shaped former made from iron, positioned close to a circular metallic disc. Using the axial symmetry of the coil, former and disc, this may be transformed into a two-dimensional model, as shown in Figure 9.5, to be solved using Opera-2d. Full details of how to build this model may be found in the Opera-2d User Guide [111].

The current density J_{coil}(A m^{-2}) and width w (cm) of the coil and the conductivity k (Sm) of the disc are treated as variables in order to maximize the force on the disc,

FIGURE 9.4: Coil, former and disc arrangement in Opera-3d.

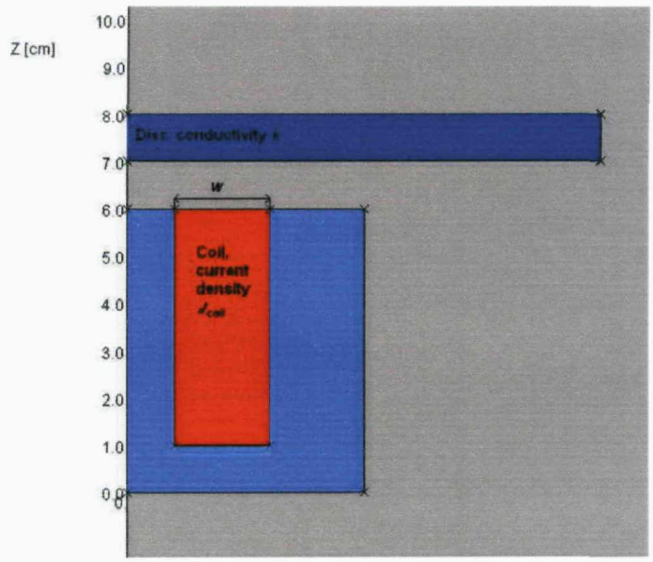


FIGURE 9.5: Coil, former and disc arrangement in Opera-2d.

F_{disc}, subject to a constraint on the maximum current density J_{eddy} induced in the disc. Formally, the constrained SOOP to be solved is:

$$\begin{aligned} \text{Maximize} \quad & F_{\text{disc}}(J_{\text{coil}}, w, k) \\ \text{Subject to} \quad & J_{\text{eddy}} \leq 3.5 \times 10^6 \text{ A m}^{-2} \\ \text{with} \quad & 0.5 \times 10^6 \leq J_{\text{coil}} \leq 4 \times 10^6 \text{ A m}^{-2} \\ & 0.5 \leq w \leq 3.5 \text{ cm} \\ & 4 \times 10^6 \leq k \leq 6 \times 10^6 \text{ Sm} \end{aligned} \tag{9.3}$$

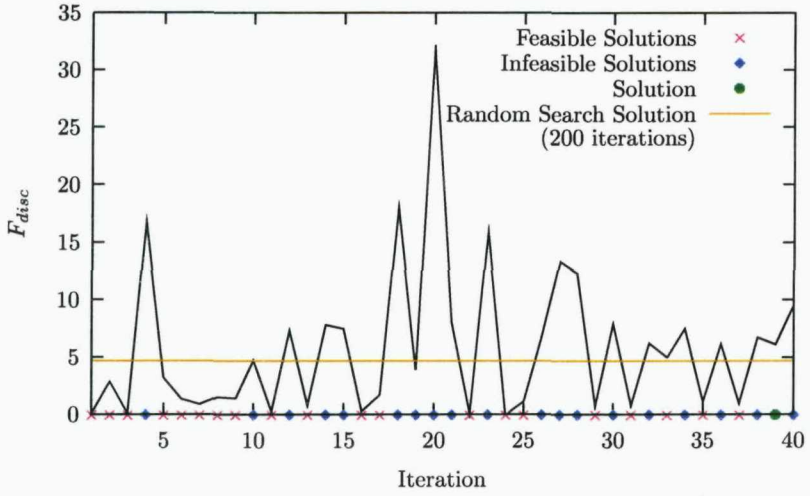


FIGURE 9.6: Variation of the force on the disc during the optimization search.

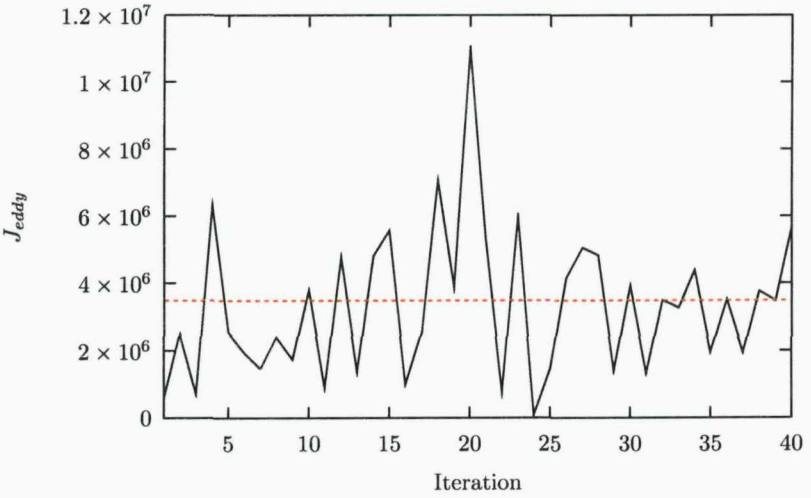


FIGURE 9.7: Variation of the maximum eddy current density during the optimization search. The eddy current density must be no greater than 3.5×10^6 A m^{-2} in order for a design to be feasible.

As the problem involves calculating eddy currents, it requires the Opera-2d transient (TR) time varying analysis.

9.3.2 Results

A termination criteria of 40 iterations was chosen. The variation of the force on the disc is shown in Figure 9.6, and the variation of the eddy current constraint density is shown in Figure 9.7. The positions in design variable space of the 40 iterations are shown in Figure 9.8.

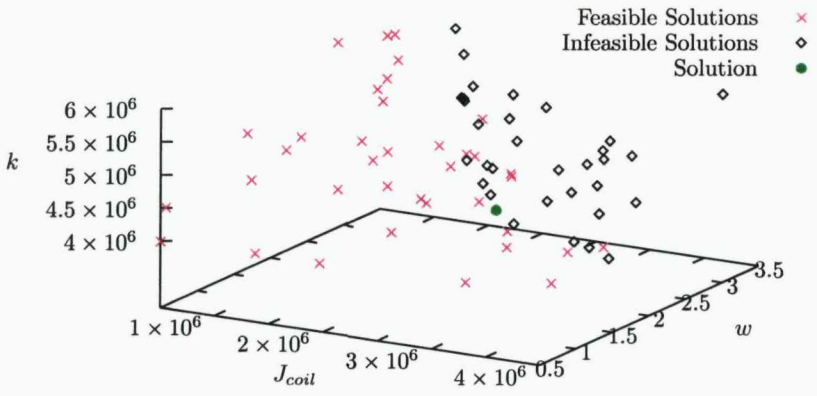


FIGURE 9.8: Positions in design variable space of the designs chosen for evaluation by the Optimizer in the coil, former and disc problem.

The best solution found (after 39 iterations) had a force $F_{\text{disc}} = 6.1059$ N, with the constraint having a value of $J_{\text{eddy}} = 3.489 \times 10^6$ A m^{-2}, just within the limit. By comparison, the best solution found by a random search algorithm of 200 iterations had a maximum of $F_{\text{disc}} = 4.691$ N, with the constraint having a value of $J_{\text{eddy}} = 3.028 \times 10^6$ A m^{-2} (found after 172 iterations). This is shown as a target in Figure 9.6, which the optimizer outperforms after 33 iterations.

9.4 Multi-Objective Unconstrained Optimization of an Electron Gun

9.4.1 Problem Definition

The voltage on, and position of, the focus electrode of an electron gun, as shown in Figure 9.9, are varied so as to achieve two objectives: to focus the beam of electrons on the centre of the anode as much as possible, and to make the electrons hit the anode face as perpendicular as possible.

Formally, denoting the voltage on the focus electrode by V Volts, and its perpendicular distance from the emitting surface by d cm, the unconstrained MOOP to be solved is:

$$\begin{aligned} \text{Minimize} \quad & f_1(V,d) &&= \int_{\text{anode}} J(r) r^2 \mathrm{d}S \\ \text{and} \quad & f_2(V,d) &&= \int_{\text{anode}} \frac{(v_x^2 + v_y^2)}{(v_x^2 + v_y^2 + v_z^2)} \mathrm{d}S \\ \text{with} \quad & 0 \leq V \leq 1000 \text{ Volts} \\ & 4 \leq d \leq 10 \text{ cm} \end{aligned} \tag{9.4}$$

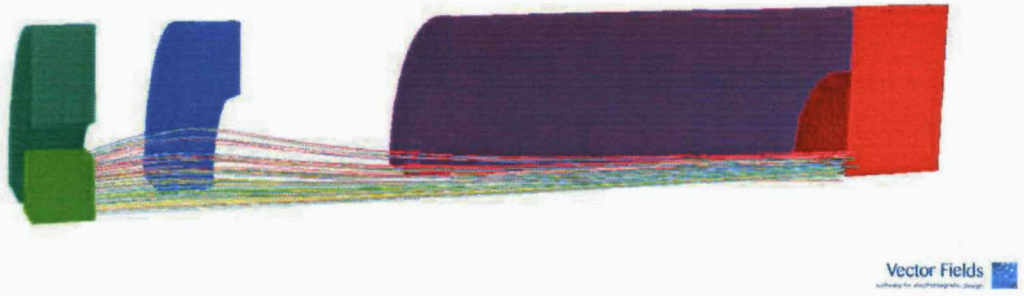


FIGURE 9.9: Arrangement of the electron gun in Opera-3d.

where r is the radial distance from the center of the anode surface, $J(r)$ is the current density at r, and the integrals are taken over the surface of the anode. v_x, v_y, v_z are the components of the electron velocities as they hit the surface of the anode, which lies in the xy plane. Each analysis is carried out using the space charge solver SCALA.

9.4.2 Results

A termination criteria of 140 iterations was chosen. Six Pareto optimal points were obtained, and are shown in Figure 9.10. The current densities on the anode surfaces of the two extreme Pareto solutions (labelled 'A' and 'B' in Figure 9.10) are shown in Figure 9.11, whilst the electron beams in the two extreme Pareto optimal solutions are shown in Figure 9.12. The difference in the solutions can clearly be seen: the beam in Pareto solution A is much more focused on the centre of the anode than the beam in Pareto solution B, whilst the electron paths in the beam in Pareto solution B are 'more parallel' than the paths in the beam of solution A, and thus hit the anode surface at less of an angle.

9.5 Multi-Objective Constrained Optimization of Helmholtz Coils

9.5.1 Problem Definition

The arrangement of a pair of Helmholtz coils is allowed to vary so as to achieve a uniform central field of 5000 Gauss. Specifically, the width A, coaxial length B, and half z separation Z_1 of the coils (with current density 10^4 A cm^{-2}), as shown in Fig. 9.13,

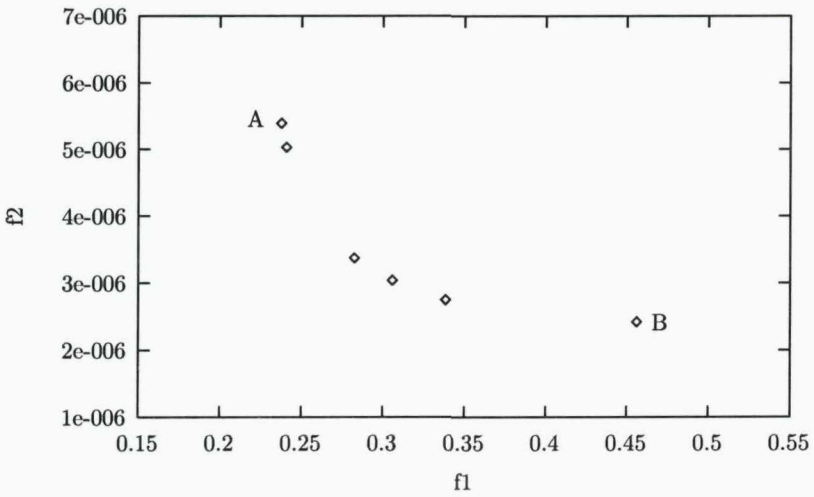
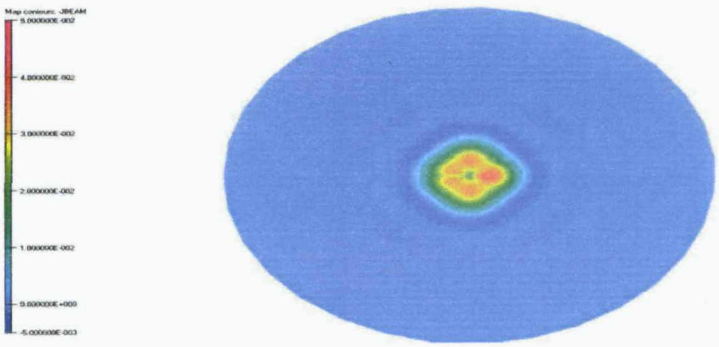
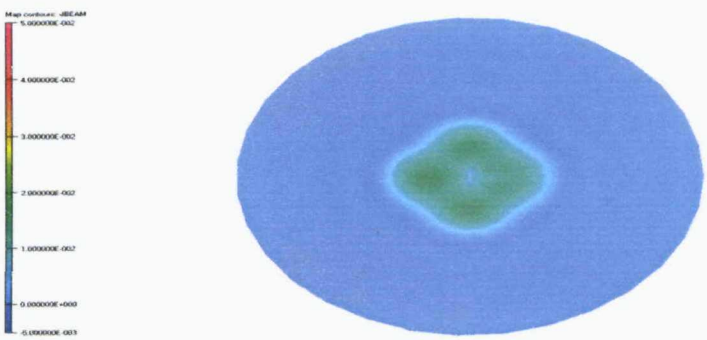


FIGURE 9.10: Pareto-optimal front for the electron gun problem.

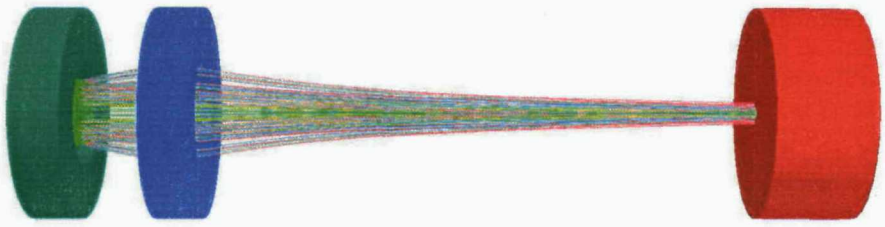


(a) Current density on anode surface of Pareto solution A.

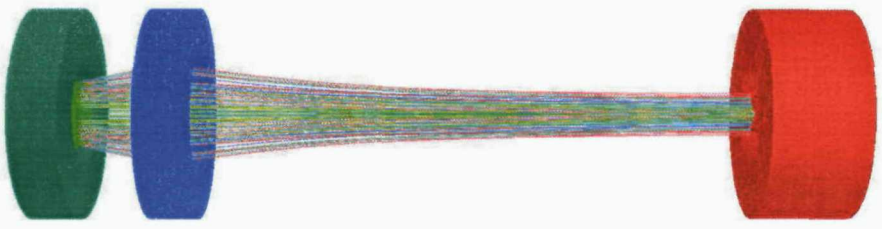


(b) Current density on anode surface of Pareto solution B.

FIGURE 9.11: Anode surface current densities in the two extreme Pareto solutions.



(a) Electron beam in Pareto solution A.



(b) Electron beam in Pareto solution B.

FIGURE 9.12: Electron beams in the two extreme Pareto solutions.

are each allowed to vary within specific ranges. To achieve the goal of a uniform central field of 5000 Gauss, the Legendre polynomial coefficients $A_{00}, A_{20}, A_{40}, A_{60}$ of the z-component of the magnetic flux density are calculated over a sphere (of radius 3) centred around the origin, after solving using the TOSCA magnetostatic solver. To ensure field uniformity, the magnitude of the error harmonics are chosen to be minimized, whilst to achieve the desired central field value, the magnitude of the A_{00} coefficient is constrained to be equal to 5000. The constrained MOOP to be solved is:

$$\begin{aligned} \text{Minimize} \quad & |A_{20}|, |A_{40}|, |A_{60}| \quad \text{(Error harmonics)} \\ \text{Subject to} \quad & |A_{00}| = 5000 \\ \text{with} \quad & 0.5 \leq A \leq 5 \text{ cm} \\ & 0.5 \leq B \leq 8 \text{ cm} \\ & 0.5 \leq Z_1 \leq 15 \text{ cm} \end{aligned} \tag{9.5}$$

9.5.2 Results

A termination criteria of 100 iterations was chosen. The variation of the three objective functions over the final iterations is shown in Figure 9.14: of these 26 designs, only 6 are infeasible, whilst the other 20 are feasible. The variation of the equality constraint over the full 100 iterations is shown in Figure 9.15: it should be noted that the first

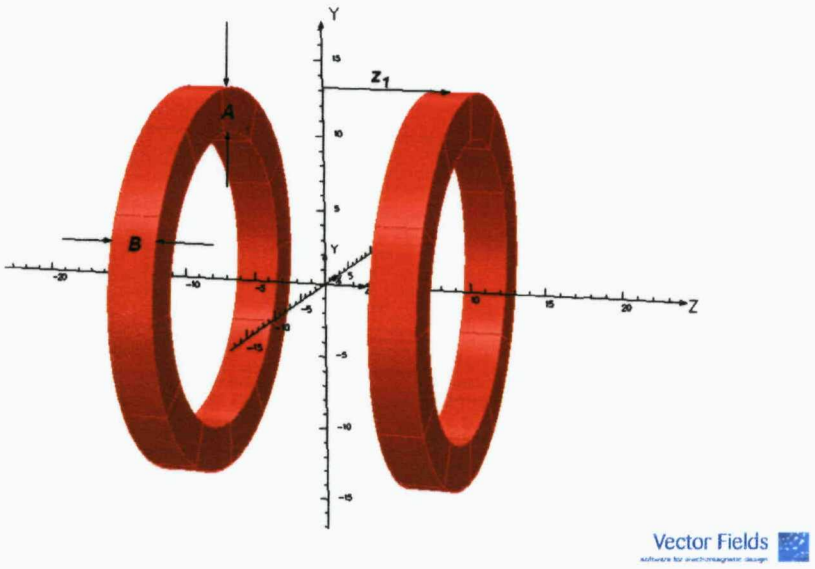


FIGURE 9.13: Arrangement of Helmholtz coils in Opera-3d.

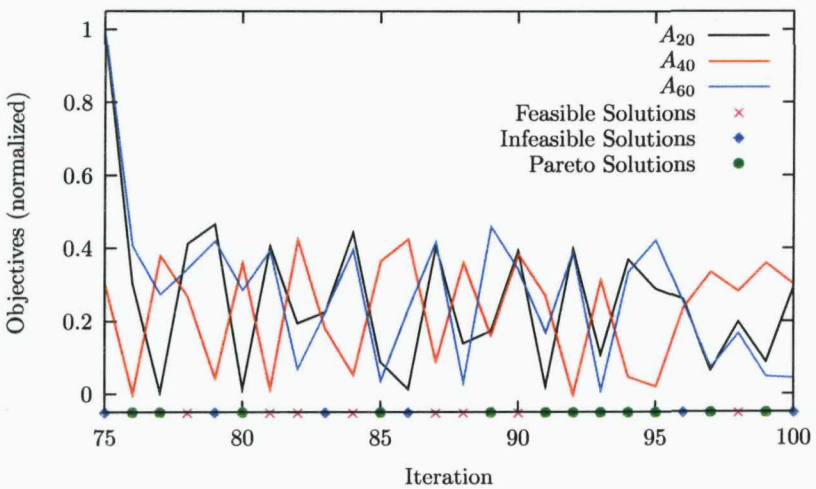


FIGURE 9.14: Variation of the objectives at the end of the optimization search.

30 iterations are experimental design points, hence the lack of constraint satisfaction during these iterations.

The positions of the 100 designs evaluated by the optimizer in design variable space are shown in Figure 9.16. In total, 19 of these were Pareto-optimal; their positions in objective function space are shown in Figure 9.17. For comparison, the 5 Pareto-optimal designs found by a random search algorithm of 150 iterations are also shown.

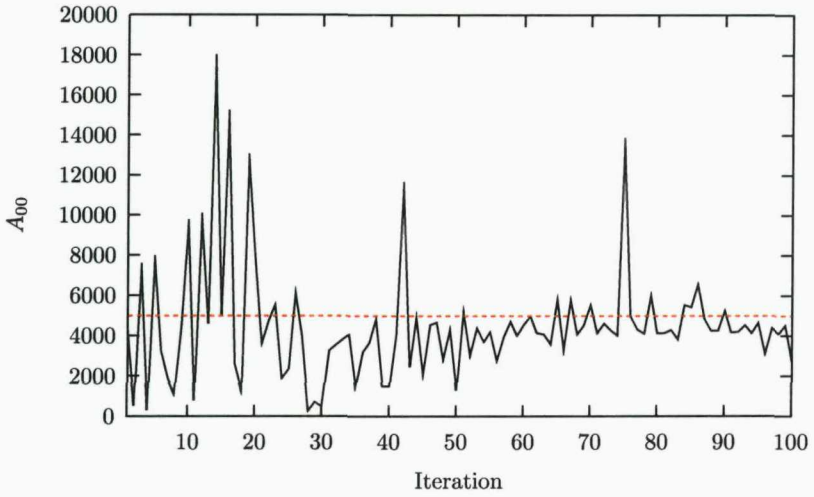


FIGURE 9.15: Variation of the equality constraint during the optimization search.

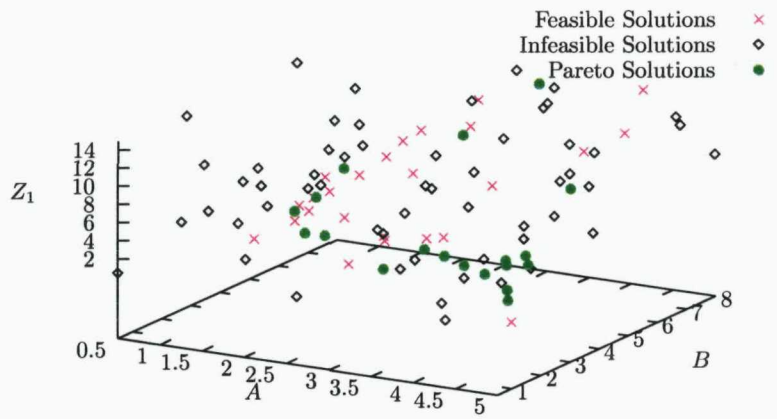


FIGURE 9.16: Positions in design variable space of the designs evaluated by the optimizer in the Helmholtz problem.

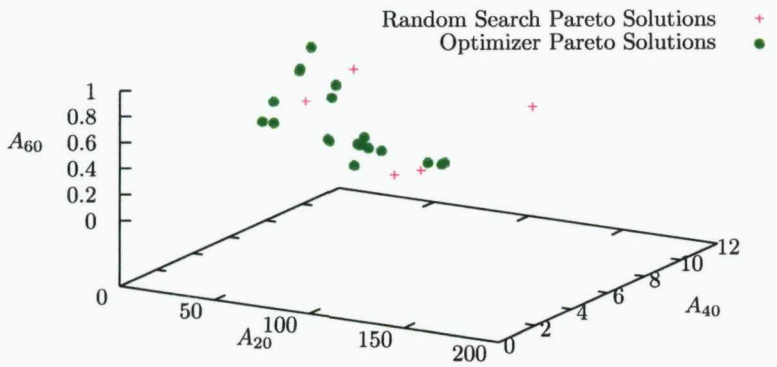


FIGURE 9.17: Pareto solutions located by the Optimizer, and a random search algorithm.

9.6 Discussion

Through four examples, it has been demonstrated that the optimization tool implemented into Opera has been capable of solving a range of optimization problems: constrained and unconstrained, single-objective and multi-objective. Furthermore, it has been applied successfully to both 2d and 3d problems.

The results compare favourably to solutions obtained from random searches of considerably more iterations. Hence the optimization tool is successful in being efficient during searches.

Chapter 10

Conclusions and Further Work

10.1 Conclusions

10.1.1 Cost-Effective Optimization

The inherently high computational cost of optimization in electromagnetic design is an important issue, and many different methods have been proposed to dealing with it. In Chapter 3, one cost-effective technique - surrogate modelling - was reviewed, with particular emphasis on kriging. At the start of this research, kriging was only beginning to be identified as being useful in electromagnetic design optimization [112]; since then, its popularity has grown, with more and more papers using kriging for electromagnetic design optimization appearing both at conferences, and in the literature.

In Chapter 4 and Chapter 5, a review was carried out of the state-of-the-art in kriging-assisted methods for single and multi-objective optimization. Whilst kriging-assisted multi-objective methods are relatively new, some single-objective methods are now quite mature, e. g. [70]. Despite this, the application of many of these methods in electromagnetic design (an ideal application, due to its high computational cost) has been slow. Further techniques in surrogate model-assisted optimization were then reviewed in Chapter 6, with particular emphasis on kriging.

10.1.2 Contribution of this Thesis

The main contributions of this thesis have been:

1. The proposal [109] (independently of [13] and [89]) of non-scalarizing methods for cost-effective multi-objective optimization [5, 10].

2. The development of a general scalarizing algorithm for cost-effective constrained multi-objective optimization [9, 11].

3. The introduction of methods (from outside electromagnetic design optimization) to electromagnetic design optimization, such as the one-stage approach [3, 4, 6, 11].

In single-objective optimization, many utility functions exist which use information relating to the uncertainty in the prediction made by a single kriging surface (modelling the objective function). This thesis extended the idea of using a utility function which considers uncertainty to multi-objective optimization, by extracting and combining information from multiple kriging surfaces (each modelling a different objective). The only way in which multiple kriging models had been used prior to this work was selecting the Pareto-optimal points predicted by them [112], analogous to the 'Strawman' approach in single-objective optimization. By considering the uncertainties in multiple kriging models simultaneously, true multi-objective equivalents of the probability of improvement method, and enhanced probability of improvement method, have been proposed.

Indeed, prior to the work in this thesis, only the ParEGO algorithm [45] made use of a utility function from kriging-assisted single-objective optimization for multi-objective optimization (through use of a scalarizing function). This thesis has generalized this idea, proposing an algorithm which can make use of any arbitrary utility function from single-objective optimization. Constraint handling techniques were included, as was a method for dealing with failed designs, an issue often overlooked in practical design optimization.

Finally, a range of methods from elsewhere have been discussed and used for electromagnetic design optimization for the first time. For example, the use of the one-stage approach in the hybrid one-then-two stage algorithm [3, 4] was the first time it had been used in electromagnetic design optimization; its use in [6] was the first time it had been used for multi-objective optimization anywhere in the literature.

10.1.2.1 KTP Objectives Revisited

The KTP objectives (given in Section 1.2.2.1) were achieved through the release of two new pieces of software: the Opera Manager, released in Opera version 11, and the Optimizer, released in Opera version 12. The Optimizer was designed with ease-of-use in mind, requiring a minimum number of settings from the user once a problem was set up (a set of termination criteria, and some basic options regarding the storage

TABLE 10.1: Work done towards satisfying the academic objectives.

Academic Objective	Work Completed
A1	Non-scalarizing methods were proposed independently through this research in [109]. The scalarizing ParEGO [45] algorithm has been generalized to make use of other utility functions.
A2	Constraint-handling methods have been used in the general scalarizing multi-objective algorithm. A method has been proposed for including constraint-handling in non-scalarizing multi-objective algorithms.
A3	The increasingly popular kriging surrogate modelling method was used; note that at the start of this research, kriging was only beginning to be identified as being useful to electromagnetic design optimization [112].
A4	Sensible default methods were chosen for the optimization tool, resulting in it behaving like a black-box (recall Figure 1.4) to the user.
A5	A user-friendly interface was constructed to allow setting up of an optimization problem in Opera from just one dialog box.
A6	The methods have been demonstrated on a range of test problems: constrained and unconstrained, single and multi-objective.

of the Opera databases created during the optimization search). The setting up of an optimization problem in Opera was also kept as simple as possible, with all options confined to one dialog box. The methods included in the Optimizer mean that even constrained multi-objective optimization problems may be solved, and results on test problems have demonstrated it to be successful in locating optimal solutions efficiently. Further details on the software produced to satisfy the KTP objectives may be found in Appendices B, C and D.

10.1.2.2 Academic Objectives Revisited

The academic objectives A1-A6 given in Section 1.2.2.3 are relisted in Table 10.1, along with evidence of how they were met. In addition to these, a novel algorithm for single-objective optimization was proposed as well [4].

10.2 Further Work

10.2.1 Extensions to the Opera Optimizer

The optimization tool introduced to Opera could be extended and improved in several ways:

- Allow discrete design variables.
- Integrate the optimization dialog into the Post-Processor, rather than have it accessed only from the Opera Manager.
- Introduce non-kriging methods into the optimization tool, for greater flexibility when using the advanced utility function dialog (e. g. offer radial basis function selection methods, such as *rbfsolve* [72]).
- Reduce the computational cost of building kriging models by using cost-effective methods to invert the correlation matrix **R** when the number of examples gets large.
- Adapt algorithms so that all available computing power is being used at all times (e. g. taking full advantage of multi-core machines).

A comparison of the Opera Optimizer with optimization tools in other electromagnetic design software on a range of benchmark problems would also be useful.

10.2.2 Kriging Methods

New kriging-based selection criteria are continually being proposed. One interesting new utility function, proposed very recently [84], makes use of information theory to select design vectors for evaluation. Results show it to outperform EGO over a range of test functions; it appears to be very well suited to electromagnetic design optimization, and a promising area of research.

Non-scalarizing methods have only recently appeared in the literature. Combining constraint handling techniques with such methods was discussed in Section 7.4.3; these approaches remain to be explored, and form an interesting area of future research.

10.2.3 Non-Kriging Methods

Radial basis functions, discussed in Chapter 3, are one of the most popular types of surrogate model. The recently proposed 'adaptive radial basis algorithm' [83], which uses a one-stage methodology similar to *rbfsolve* [72], appears to be a very promising method using this surrogate model. Methods based around the one-stage methodology are in general a very attractive area of future research for at least two main reasons: first, results show them to work extremely well; second, very few such methods have so far been explored.

Another method (from the field of machine learning) which is growing in popularity for the purposes of surrogate modelling is support vector machines (SVMs). Because of their high accuracy, if the uncertainty in predictions made using SVMs can be quantified as effectively as it has been in kriging, it could lead to a wide range of highly efficient algorithms for both single and multi-objective optimization.

Part V

Appendices

Appendix A

Other Cost-Effective Optimization Techniques

A.1 Hybrid Algorithms

Whilst Multi-Objective Evolutionary Algorithms (MOEAs) are highly regarded for their ability to search globally in objective space (thus avoiding getting trapped by local minima), this ability derives mainly from the large number of objective function calls it must evaluate. On the other hand, deterministic search algorithms are highly regarded for their local search ability, and furthermore, possess this ability without the need for a large number of objective function evaluations.

As most optimization problems require both global and local searches, it seems natural to combine MOEAs with deterministic search methods, to produce *hybrid* algorithms, with good global and local search ability. As well as improving convergence to the Pareto-optimal front, hybrid algorithms often require less objective function evaluations to achieve a certain performance than either a MOEA or deterministic algorithm alone.

Deterministic search methods may be used with MOEAs in variety of different ways. Two of the most obvious are [113]:

1. A Posteriori Approach, in which the MOEA runs until a stopping criterion is met (usually either a fixed number of generations has elapsed, or sufficient convergence has occurred). The deterministic search method then runs from each of the resulting non-dominated solutions.

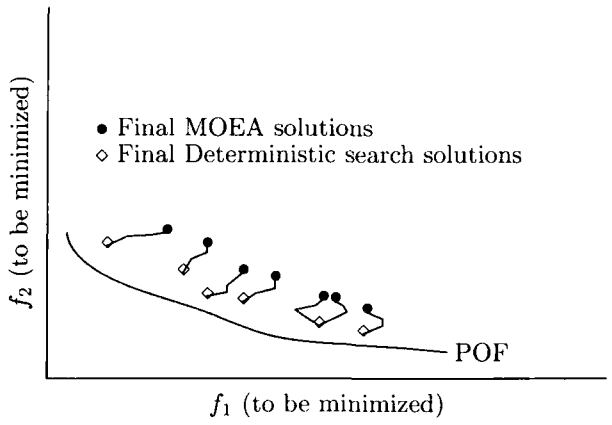


FIGURE A.1: A posteriori hybrid method

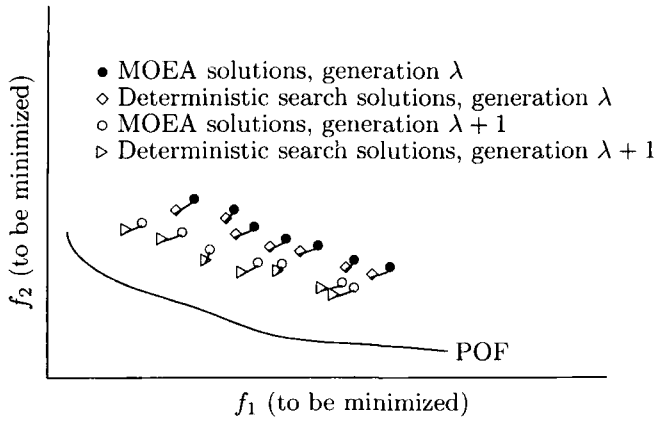


FIGURE A.2: Online hybrid method

2. Online Approach, in which the local search method modifies each solution in a population before the MOEA proceeds to the next generation.

The a posteriori approach and online approach are shown schematically in Figure A.1 and Figure A.2 respectively.

As deterministic search methods can only optimize a single objective, the MOOP must be transformed to a SOOP [38] prior to the deterministic method being used. Hybrid algorithms differ in the MOEA and deterministic search methods they use, in how the deterministic method is used with the MOEA, in the amount of local search performed, and in which classical method is used to transform the MOOP to a SOOP.

A.2 Small Population MOEAs

From an MOEA point of view, a restrictive number of objective function calls means low population sizes, and so perhaps the most obvious way to deal with this is to design the MOEA specifically for small populations [114]. Implementations based on this approach differ in the exact way the algorithm is suited to low populations. The algorithm may need few objective function evaluations to decide upon the next point, or the algorithm may make most use of objective function calculations when choosing the next point.

A.3 Fitness Inheritance

Fitness inheritance is a simple technique which can be used in any EA to reduce cost. Unlike the small population MOEAs and hybrid algorithm techniques, fitness inheritance does not work by reducing the total number n_f of objective function evaluations. Instead, rather than all fitness values in an EA being calculated by performing the costly evaluation of the objective function, some individuals have their fitness calculated by estimation, the estimation being based on the fitness values of their parents. Thus, the number of *costly* evaluations decreases, whilst the total number of evaluations is kept constant; in particular, the population size of the MOEA is not reduced as it is in the previous two techniques.

A.4 Reduction of Design Variables

The value of M in Equation 2.4 is the main determining factor in choosing an appropriate algorithm to perform the optimization; the value of n on the other hand, is the main factor determining the number of objective function evaluations n_f which the chosen algorithm will need to use to locate the optima, as it sets the number of directions in which the algorithm may search in objective space. The relationship between n and n_f is not fixed, and also depends on the size of the intervals within which each design variable may vary, but in general, the larger n is, the lager n_f will be. Thus, computational cost may be reduced by reducing n.

Reducing the number of design variables is only justified however, when there are design variables which have little or no consequence to the objective functions. For example, if there are five design variables set in a problem, and all are heavily influential

to the objective functions, then design variable reduction may not be performed. However, if only two were found to be of significant consequence to the objective functions, then the other three may be held constant, thereby reducing the number of search directions available to the optimization algorithm from five to two. As a consequence, the algorithm requires less objective function evaluations to locate the optima.

Appendix B

Opera Manager User Manual

The Opera Manager

Starting the Opera Manager

Microsoft Windows

On Microsoft Windows systems the Opera Manager is started from the menu bar as follows:

```
Start -> Programs -> Vector Fields Opera -> Opera 12.0
```

Alternatively the Opera Manager can be started from the taskbar notification area by clicking on the Opera icon as shown here:

Linux and Solaris

On Linux and Solaris systems, the Opera Manager is started from the command line by typing:

```
$vfdir/bin/opera_manager
```

where **`$vfdir`** is a shell variable set to the directory where the software is installed.

The Opera Manager Window

The Opera Manager is the "navigation centre" for the complete suite of Opera software. Its main window consists of two areas:

- a workspace allowing multiple Folder Windows to be opened.
- a dockable Batch Processor, which controls the running of analyses.

In addition there is a menu bar and a toolbar located under the title bar. The menu bar gives access to all functions of the Opera Manager.

The toolbar can be used to start the interactive programs:

- Opera-2d/Pre and Post-Processor  ;

  N.B. The Opera-2d/Design Environment can only be started via the menus (see Opera-2d/Design Environment [page 22]).

- Opera-3d/Modeller  , Pre-Processor  and Post-Processor  .;

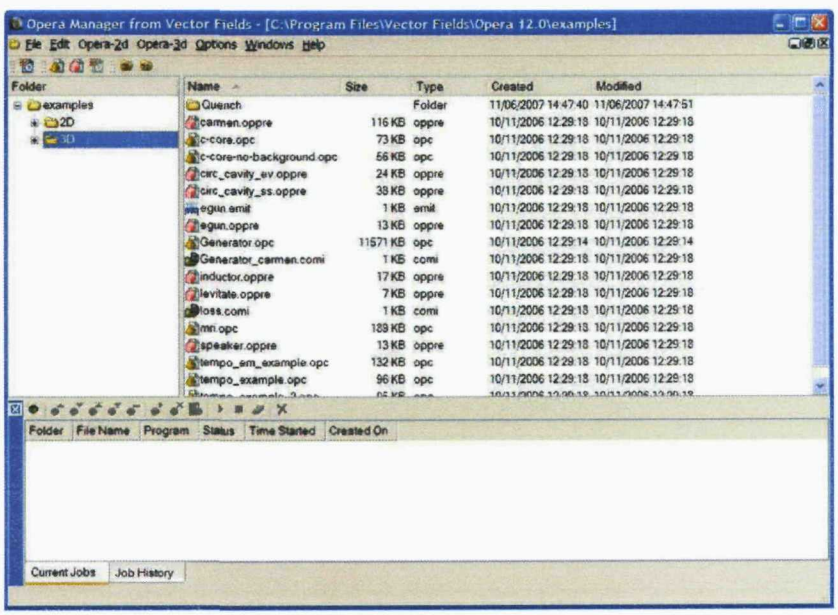


Figure 1 The Opera Manager

or to open the Opera-2d  and Opera-3d  project folders.

Folder Windows and the Workspace

Starting the Manager for the first time

When the Manager is started for the first time, one Folder Window will be created in the workspace. The top level folder of this window will be `C:\` on Windows, or `/` on Linux and Solaris.

Layout of a Folder Window

A Folder Window consists of two parts: on the left hand side is a Folder Pane, and on the right hand side is a Contents Pane. The Folder Pane shows a tree-view of all the folders contained within the top level folder. The Contents Pane shows the contents of whichever folder is selected in the Folder Pane. The Contents Pane consists of several columns which display details about files and folders. Columns may be hidden or displayed by right-clicking on a column header and checking a column to be displayed, as shown in Figure 2.

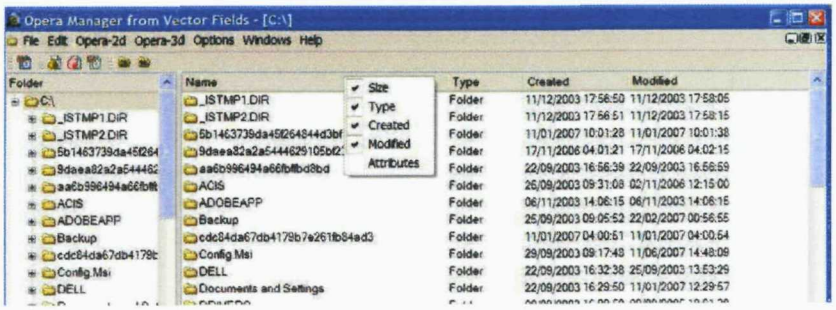


Figure 2 Selecting File Details

Folder Windows may be maximized (as shown above), minimized, restored down, or closed, by clicking on the appropriate button on the top right of the Folder Window. For example, clicking on the restore down button will result in the view shown in Figure 3.

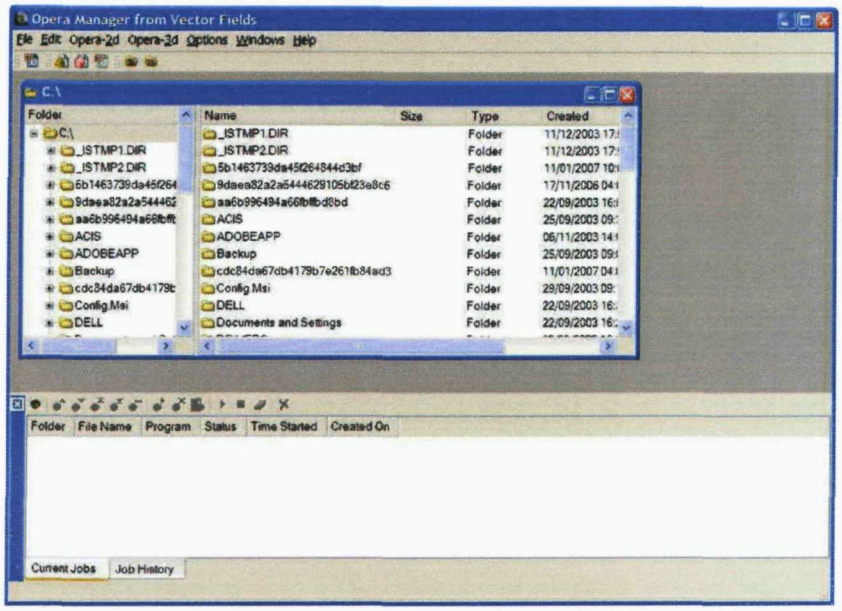


Figure 3 Folder Window Restored Down

Context Menus

Context menus may be invoked by right-clicking on an item in either the Folder Pane or the Contents Pane. Only options relevant to the item which was clicked will be available on the context menu.

Folder Pane Context Menu

In the Folder Pane the context menu appears as shown in Figure 4.

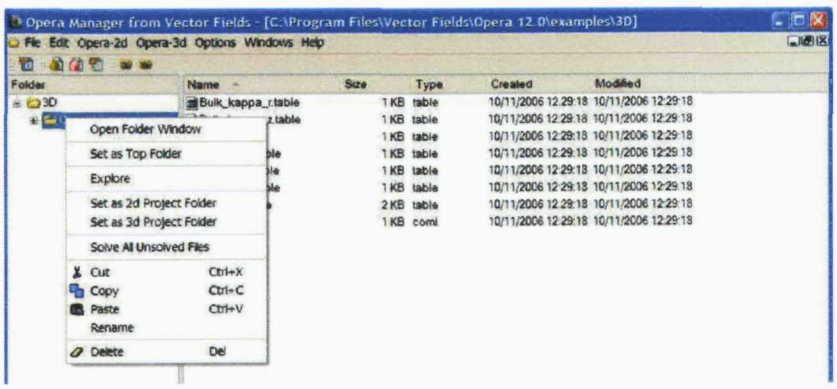


Figure 4 Folder Pane Context Menu

The menu items perform the following actions:

- **`New Folder Window`**: Creates a new Folder Window, with the top level folder set to be the selected folder.
- **`Set as Top Folder`**: Resets the top-level folder of the current Folder Window to be the selected folder.
- **`Explore`** (Microsoft Windows only): Opens up a Windows Explorer window with the address set to the selected folder.
- **`Create New Folder`**: Creates a new folder in the selected folder.
- **`Set as 2d Project Folder`**
  `Set as 3d Project Folder`: Sets the selected folder to be the 2d or 3d project folder.
- **`Solve All Unsolved Files`**: Solves all Opera-2d data files and all Opera-3d databases which have unsolved simulations in the selected folder (N.B. it does not solve files in any sub-folders).
- **`Cut/Copy/Rename/Delete`**: Cut/Copy/Rename/Delete the selected folder.
- **`Paste`** (only available if a file/folder has been cut or copied previously): Pastes the cut or copied file(s)/folder(s) to the folder which is selected in the Folder Pane.

The top-level folder has an additional menu item on its context menu:

- **`Up One Level:`** Makes the Folder Window show the parent folder.

Contents Pane Context Menu

In the Contents Pane a typical context menu (for a file) will be as shown in Figure 5.

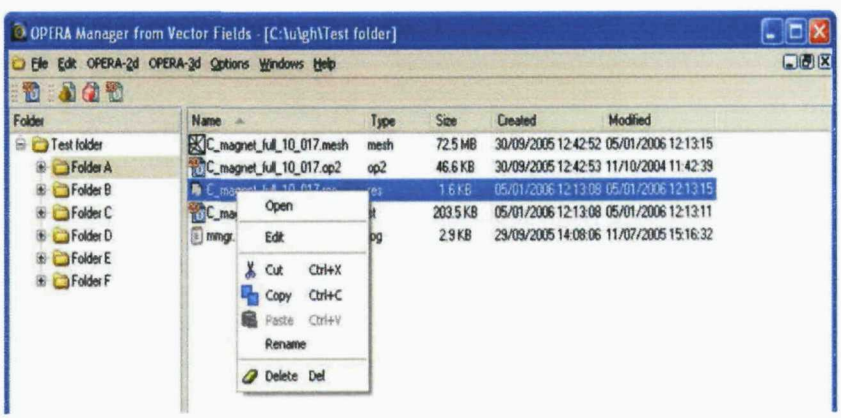


Figure 5 Contents Pane Context Menu

The menu items perform the following actions:

- **Open** data files.

  Starts the default program for the file and opens the file.

  - **.op2*: Opera-2d/Pre and Post-Processor
  - **.opc*, **.opcb*: Opera-3d/Modeller
  - **.oppre*: Opera-3d/Pre-Processor
  - **.op3*: Opera-3d/Post-Processor

- **Open** command input files

  If the file is a command input file (**.comi*), the first line of the file will be read to see if it is a comment identifying the program it was written for. The following text is used to identify which program to start:

  Pre & Post-Processor: Opera-2d/Pre and Post-Processor

  Modeller: Opera-3d/Modeller

  Pre-Processor: Opera-3d/Pre-Processor

  Post-Processor: Opera-3d/Post-Processor

  Command input files without such a comment will have multiple **Open** options in the context menu. Comments are automatically added to **.log* files so that *.comi* files created from *.log* files will refer to the right program. For example, the comment in a *.log* file from the Opera-3d/ Modeller is

  / Opera-3d Modeller Version *nn.nnn*

- **Edit**: Opens the file in an editor. The editor which is used can be specified via the **Options** menu. Whether a file is editable or not is determined from the file type. The software contains a list of editable file types.

- **Cut/Copy/Rename/Delete**: Cut/Copy/Rename/Delete the file.

If the file is an Opera-2d data file (**.op2*) or an Opera-3d database (**.op3*) containing an unsolved simulation the following option is also available:

- Solve with *solver_name*: Adds the Opera file to the batch queue.

If a folder is selected in the Contents Pane, the context menu will have options similar to those available through the context menu in the Folder Pane.

If the context menu is invoked by clicking on an empty area of the Contents Pane, then a context menu appears with options referring to the folder

whose contents are displayed in the Contents Pane (i.e. the folder which is selected in the Folder Pane).

Main Menu Items

Many of the operations available for an item through a context menu are also available via the main menu bar. The option on the menu bar will apply to the item which is currently high-lighted, whether it is in the Folder Pane or the Contents Pane. If no item is high-lighted, then the options will not be available through the main menu bar.

Creating a New Folder Window

Additional Folder Windows may be created in the workspace by the menu route (see Figure 6).

```
Windows -> New Folder Window
```

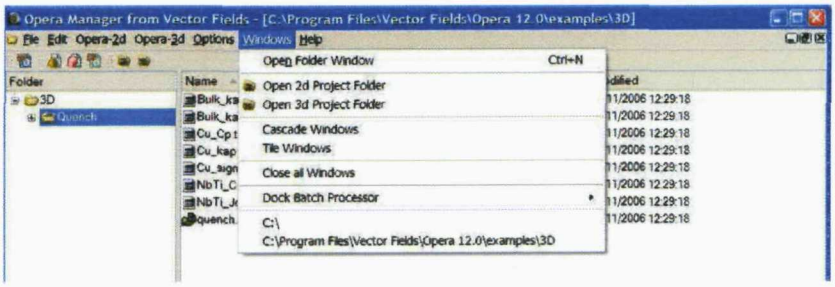


Figure 6 New Folder Window

The dialog can be used to browse to an existing folder or create a new one. Multiple new Folder Windows may be created.

Control of Windows

The Folder Windows in the workspace may be cascaded or tiled via the menu routes:

```
Windows -> Cascade windows
Windows -> Tile windows
```

All the Folder Windows may be closed via the menu route:

```
Windows -> Close all windows
```

A list of all existing Folder Windows within the workspace is listed at the bottom of the Windows menu. Selecting a particular window from this menu will make that Folder Window active.

A list of recent Folder Windows is available through the menu route:

```
File -> Recent Windows
```

This lists the paths of the most recent Folder Windows which have been created, regardless of whether they are still open in the workspace or have since been closed.

Recognised File Types

The following file types are recognised by the Opera Manager. The table shows the icons used and the program, if any, which will open the file in response to a double-click. Those marked "editable" can be opened in an editor from the context menu.

Icon	Type	Usage	Program	Editable
	*ac*	Opera-2d/AC results file	Opera-2d/PP	
	backup	Opera-3d/Pre data file		
	bat	Windows command file		yes
	*bh*	Opera-2d and Opera-3d BH data file		yes
	cas	Opera-2d/DE case file		
	cir	SPICE type circuit data for Opera-2d/PP		
	cmd			yes
	*comi*	Command input file	Opera-2d/PP, Opera-3d/Modeller, Opera-3d/Pre or Opera-3d/Post (see Open command input files [page 14])	yes
	*cond*	Opera-3d conductor data file		yes
	csr	Opera-2d/DE case results		yes
	dem	Opera-2d/DE data file		
	*dm*	Opera-2d/DM results file	Opera-2d/PP	
	dxf	AutoCAD DXF data for Opera-2d/PP		
	*emit*	Opera-2d and Opera-3d emitter data file		yes
	grid	Binary tabulated data from Opera-3d/Post		
	igs	IGES format data for Opera-3d/Modeller		

Icon	Type	Usage	Program	Editable
	*lm*	Opera-2d/LM results file	Opera-2d/PP	
	log	Commands issued to an interactive program Table of system variables from a transient analysis		yes
	lop	Opera-2d circuit data		
	lp	Dialogue file of input and output of interactive programs		yes
	mate	Opera-2d stress and thermal material data		
	*mesh*	Opera-2d mesh file		
	*op2*	Opera-2d data file	Opera-2d/PP	
	*op3*	Opera-3d database	Opera-3d/Post	
	*opc*	Opera-3d/Modeller data file	Opera-3d/Modeller	
	*opcb*	Opera-3d/Modeller binary data file	Opera-3d/Modeller	
	*oppre*	Opera-3d/Pre data file	Opera-3d/Pre	
	res	Text file showing progress of analysis		yes
	*rm*	Opera-2d/RM results file	Opera-2d/PP	
	*sa*	Opera-2d/SA results file	Opera-2d/PP	
	sat	SAT format data for Opera-3d/Modeller		
	script	Opera-2d/DE command script		
	*sp*	Opera-2d/SP results file	Opera-2d/PP	
	*st*	Opera-2d/ST results file	Opera-2d/PP	
	table	Opera-2d or Opera-3d tabulated data file		yes

Icon	Type	Usage	Program	Editable
	*th*	Opera-2d/TH or THTR results file	Opera-2d/PP	
	*tracks*	Opera-2d and Opera-3d trajectory data file		
	tt	Time table data file		yes
	txt			yes
	unv	Universal file read by Opera-3d/Pre and written by Opera-3d/ Post		
	var	Opera-2d/DE variables		yes
	*vl*	Opera-2d/VL results file	Opera-2d/PP	

Appendix C

Batch Processor User Manual

The Batch Processor

The Batch Processor Window

The Batch Processor is a window which may be docked at the bottom, top, left or right of the Opera Manager, or undocked as a separate window. By default it is docked at the bottom of the Opera Manager.

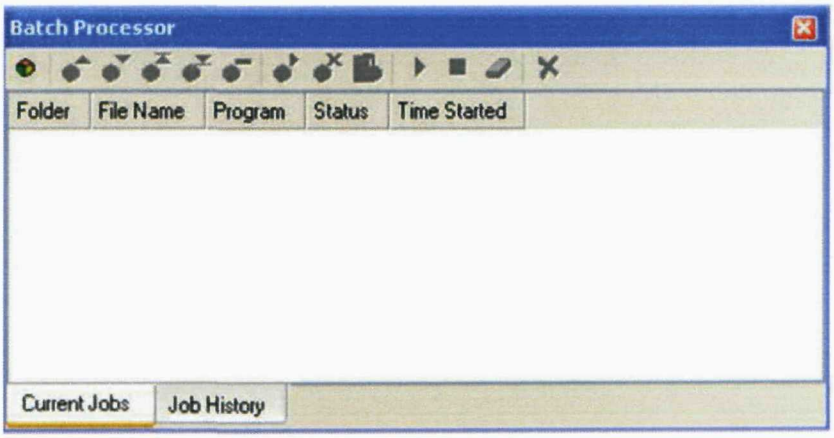


Figure 7 The Batch Processor

The Batch Processor consists of a Current Jobs Pane, a Job History Pane, and a toolbar (see Figure 7).

Current Jobs Pane

The Current Jobs pane shows details of analyses in the batch queue, which are either running, or waiting to be started. The columns of the pane show the details which are self-explanatory. Opera files are added to the batch queue by selecting the solve option from their context menu in the Folder Window, or by dragging and dropping them into the Current Jobs Pane (Microsoft Windows only).

Job History Pane

The Job History pane shows details of jobs which have finished running.

Batch Toolbar

In the Current Jobs Pane, the batch toolbar allows the following operations:

-  Start the queue (if it is not already running).
-  Stop the queue (if it is already running) – this option allows jobs which have started running to finish.
-  Abort the queue (if it is already running) – this option aborts any jobs which are currently running.
-  Clear all jobs from the batch queue (only available if the queue is not running).

The following operations are also available via the toolbar when individual jobs are selected in the Current Jobs Pane:

-  Move the job up the queue.
-  Move the job to the top of the queue.
-  Move the job down the queue.
-  Move the job to the bottom of the queue.
-  Remove the job from the queue.

-  Run the job (if it is not already running).
-  Abort the job (if it is already running).
-  Show the solver window for the analysis (if it is not already visible).

Batch Processor Options

The batch queue has several options which can be set via a dialog box (Figure 8) which may be invoked either via the menu route:

```
Options -> Batch Options
```

or by clicking on the Batch Options tool-button in the Batch Toolbar.

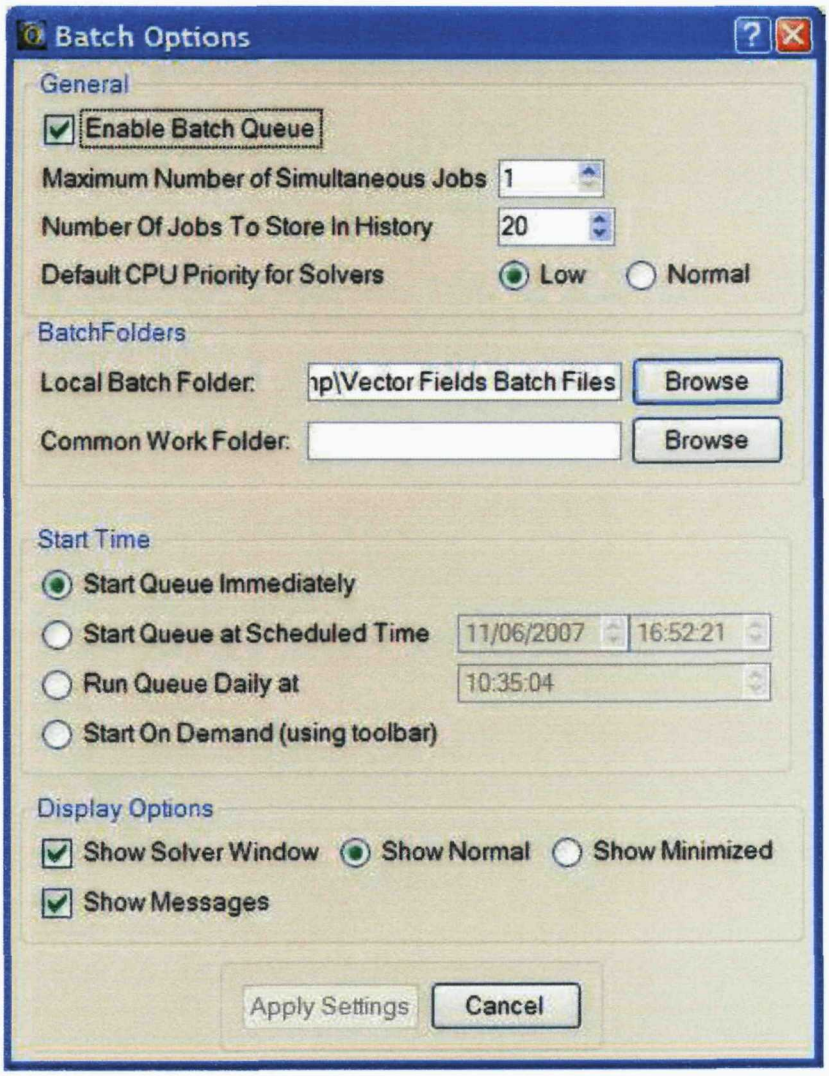


Figure 8 Batch Options Dialog

Enable

The batch queue may be disabled by unchecking the "Enable Batch Queue" box. This will result in all jobs starting immediately when they are selected to be solved.

Numbers of Jobs

The maximum number of jobs which can be run simultaneously.

The number of completed jobs to store in the Jobs History Pane.

CPU Priority

The default CPU priority level which batch jobs have when they start can be set to low or normal.

Local Batch Folder

The local batch folder holds the temporary XML files which contain details of each pending batch job. The default locations for the folder are:

- Windows XP:
  `C:\Documents and Settings\`*`username`*`\Local Settings\Temp\Vector Fields Batch Files`
- Windows Vista:
  `C:\Users\`*`username`*`\Appdata\Local\Temp\Vector Fields Batch Files`
- Linux and Solaris:
  `~/.VectorFields/`*`hostname`*`/Batch_Files`

Common Work Folder

The common work folder is an area of disk space which is shared by several computers. If a data file (**.op2* or **.op3*) exists in the common work folder or one of its sub-folders, when it is added into the batch queue for analysis, the temporary XML file containing the details of the batch job will be stored in the common work folder rather than the local batch folder. Any pending batch jobs in the common work folder can be started by the batch queue on any of the computers which share the folder. Jobs will always be taken from the local batch folder before the common work folder. This allows work to be distributed around computers in a local area network.

Start Time

The start time of the batch queue may also be set. The options available are:

- **`Start immediately`** (the queue is always running, with the maximum number of jobs running at any one time being set as above).
- **`Start at a scheduled time`**

- **`Run the queue daily at`** a specified time
- **`Start on demand:`** only start the queue through use of the "Start Queue" toolbar button.

Solver Window

The solver window for jobs in the queue may be displayed, minimized or not displayed. Similarly, messages for the job (such as "job has finished") may either be displayed or not.

Appendix D

Optimizer User Manual

The Optimizer

Introduction to Optimization

The general aim in optimization is to minimize a set of objective functions

$$f_1, f_2, \ldots, f_M,$$

subject to a set of inequality constraints

$$g_1 \leq 0, g_2 \leq 0, \ldots, g_J \leq 0,$$

and equality constraints

$$h_1 = 0, h_2 = 0, \ldots, h_K = 0.$$

Each objective and constraint is a function of a set of design variables

$$x_1, x_2, \ldots, x_d,$$

each of which has a numerical lower and upper limit. A specific combination of design variables, e.g.

$$x_1^0, x_2^0, \ldots, x_d^0$$

is referred to as a design. Formally this may be stated as

$$\begin{aligned} &\text{Minimize } f_m(x), m = 1, 2, \ldots, M \\ &\text{subject to } g_j(x) \le 0, j = 1, 2, \ldots, J \\ &\text{and } h_k(x) = 0, k = 1, 2, \ldots, K \\ &\text{with } x_i^{(L)} \le x_i \le x_i^{(U)} \end{aligned}$$

This is the standard formulation of an optimization problem. In general however, each objective function may be either maximized or minimized, and each constraint may be rewritten in many ways, e.g.

$$p(x) \le q(x)$$

$$p(x) - q(x) = g(x) \le 0$$

The Vector Fields Optimizer allows this more general statement of the problem via the optimization dialog.

If J=K=0, the problem is unconstrained. Otherwise, the problem is constrained. In constrained problems, a design may be feasible (if it satisfies all the constraints), or infeasible (if one or more constraint is not satisfied). If M=1, the problem is a single-objective optimization problem (SOOP). The solution to a SOOP is simply the feasible design which minimizes the single objective f. If M>1, the problem is a multi-objective optimization problem (MOOP). In general, no "ideal" single solution exists for MOOPs, which simultaneously minimizes each objective function. Instead, a set of (feasible) designs which represent the best possible compromise between each of the objectives are identified as solutions: these designs are called the Pareto-optimal solutions. A design is Pareto-optimal if and only if no other design exists which is strictly better than it in at least one objective, and no worse in all the other objectives.

Defining Optimization Problems in Opera

The Optimization Dialog

All optimization problems in Opera are defined through the optimization dialog. This dialog allows:

1. design variables to be defined, along with their numerical limits,
2. objective functions to be defined, including whether they are to be minimized or maximized, and
3. inequality and equality constraints to be defined.

In addition, some options may be defined for the optimization process itself. These are:

1. termination criteria for the optimization algorithm, and
2. which Opera databases to keep. (Each iteration of the optimization process involves the creation of an Opera database (representing a particular design); the user can state whether each of these databases should be kept or deleted. Alternatively, only the databases representing the best designs may be kept.)

Starting the Dialog

The optimization dialog is invoked from the Opera Manager in one of two ways. Either

1. select the *op3* file (3d) or *op2* file (2d) to be optimized in the Opera Manager and then use the menu route **`File->Optimize`**, or
2. right click on the *op3* file (3d) or *op2* file (2d) to be optimized from within the Opera Manager and select **`Optimize`** from the context menu.

Prerequisites for Opera-2d

The prerequisites for an *op2* file being eligible for optimization are:

1. the model must be parameterized using model dimensions (for quantities which are to be set as design variables), and
2. a command file (with the same name as the *op2* file) must exist (in the same folder as the *op2* file), which builds the parameterized model.

If a *comi* file is used for the rebuilding of models (necessary for 2d optimization), the model dimension user variables will be available for using as design variables from the optimization dialog. It is important that the comi file DOES NOT reset the value of the model dimension variables. To achieve this, it is advisable to state all user variables as constants (or parameters) at the start of the *comi* file, and use them as such in the construction of the model. Then at the end of the *comi* file, change the type of all user variables which are to be used as design variables to model dimension.

For example, here is a the start and end of a comi file:

```
-----------------------------------
$cons name=#a value=1
$cons name=#b value=3
$cons name=#c value=5

... model building commands ...

//#a and #b are to be used as design variables
$modeldimension name=#a
$modeldimension name=#b
-----------------------------------
```

Prerequisites for Opera-3d

The prerequisites for an *op3* file being eligible for optimization are:

1. the model used to create the database must have been parameterized using model dimensions (for quantities which are to be set as design variables), and
2. either
   i) a parameterized *opc* model file (built in Opera 12 or later, with the same name as the *op3* database) must exist (in the same folder as the *op3* file), or
   ii) a command file (with the same name as the *op3* database) must exist (again, in the same folder as the *op3* file), which builds the parameterized model (including preparing and meshing the model body, but not including preparation of the database).

Defining Design Variables

The design variables in a problem are a subset of the model dimensions used in creating the Opera model. After an *op3* or *op2* file has been used to invoke the optimization dialog, the model dimensions in the file are displayed in the **Design Variables** tab of the Optimizer dialog. Each model

dimension may be selected for use as a design variable by selecting it in the **Use** column, and entering lower and upper numerical limits to define its range; alternatively, a model dimension may be given either a constant (numerical) value, or a value based on other model dimensions by deselecting the **Use** column and entering a value in the **Current Value** column. Note: if you wish to use one design variable as the limit of another design variable (e.g. an inner radius must be less than an outer radius), then this must be done through the **Constraints/Goals** tab. Only numerical limits may be entered for each design variable in the **Design Variables** tab.

Optimization Outputs

To facilitate the evaluation of objective functions and constraints, a set of "optimization outputs" are defined. These are user variables which are evaluated in the Post-Processor (either 2d or 3d) using *comi* files. Optimization outputs may be viewed as the superset of objective functions and "expensive" terms in constraint expressions ("expensive" terms are those which must be evaluated in the Post-Processor). To define optimization outputs, a *comi* file must be written. This *comi* file must:

1. carry out necessary calculations on the results of analysis,
2. store each optimization output in a separate user variable (either a constant, parameter or model dimension), and
3. contain a comment line containing the word "out" followed by a space separated list of the names of the optimization output variables.

The command files setting up these optimization output quantities are loaded in the **Optimization Outputs** tab of the Optimizer Dialog. (To load a *comi* file, select it from the drop down list. Alternatively, press the browse button and browse to the location of the command file you wish to use.) Once a *comi* file is loaded, the output variables defined in it are listed in the adjacent **Output Variables** list. Note that more than one *comi* file may be used to define optimization outputs: to create additional rows in the tab, place the cursor in the *comi* file list and press return.

Defining Objective Functions

When a set of optimization outputs has been loaded, each output is listed in the **Objective Functions** tab. Each output may be selected as an objective, to be either minimized or maximized. At least one output must be selected as an objective function before a problem may be defined; however not every output necessarily needs to be used as an objective function.

Defining Constraints

The constraints of the optimization problem may be defined in the **Constraints/Goals** tab. Each constraint requires two expressions and a required relationship (**<=**, **~=** or **>=**) between them. Each expression may be composed of terms which are either design variables, optimization outputs, or numbers, combined using the standard mathematical operators **+**, **-**, **/**, *****, and ****** (for powers). Each expression may also use any of the arithmetic, trigonometric or exponential functions valid in the Post-Processor. Note that additional rows (for additional constraints) may be created by placing the cursor in the list in the **Expression 2** column, and pressing return.

Note that it is good practice to keep the number of equality constraints to a minimum; this may be achieved by rewriting each such constraint as two separate inequality constraints instead. In the case of inexpensive equality constraints (i.e. equality constraints on the design variables), it may be possible to change the problem definition to avoid certain equality constraints. For example, rather than having **#x** and **#y** both selected as design variables, with the constraint that **#x+#y=1**, it is better to just have **#x** as a design variable (say), with the value of **#y** "fixed" at **1-#x**.

Defining Optimizer Settings

By clicking on the **Optimization Settings** tab, some settings for the optimization process may be specified: termination criteria for the algorithm, and solution database options.

Three different termination criteria may be imposed on the optimization algorithm:

1. a maximum number of iterations,
2. a maximum elapsed time, or
3. a minimum convergence tolerance.

Each iteration of the optimization process involves the creation of an Opera database or results file (corresponding to a particular design, chosen by the optimization algorithm); the user can state whether each of these databases should be kept or deleted. Alternatively, only databases representing the best designs may be selected to be kept. If databases are kept (whether all or optimal), the user can specify an upper limit (in GB) which the stored database files must not exceed.

The Optimization Process

Overview

Once an optimization problem has been defined through the optimization dialog, the problem settings are saved to an "optimization file" by pressing the **OK** button. The name of this file may be changed through the optimization dialog, however its extension must be *opn*. Once the optimization file has been written, the optimization process begins. The optimizer iteratively determines which designs are to be evaluated, and sends them to the Batch Processor for processing (ensure that the **Start Queue Immediately** option is selected for the starting of jobs in the Batch Processor Options [page 38]). The databases and results files corresponding to these designs are located in a sub-folder (of the same name as the *opn* file) of the original working folder (i.e. the folder which the original *op3* or *op2* file is in).

The optimization process may be summarized as follows:

1. Problem defined through the optimization dialog, and written to *opn* file.
2. Optimizer determines initial designs to evaluate.
3. Initial designs sent to Batch Processor.
4. Initial designs processed.
5. Optimizer reads in processed jobs.
6. Optimizer determines next jobs.
7. Next jobs sent to Batch Processor.
8. Next jobs processed.

The optimization process iterates over stages 5-8 until the user-specified termination criteria is met.

Viewing Opera Optimization Results

The results of the Opera optimization are displayed in an **Optimization Results** tab in the Batch Processor. Each row in this tab corresponds to a different design.

The first column of the results tab shows the rank of each design. This is recalculated after each iteration. Results are ranked as follows:

Single-Objective Optimization: Designs are simply ranked according to their objective function value. If the problem is constrained, all feasible designs are ranked above infeasible designs, which all have the same rank (one below that of the last feasible design). Designs which fail to complete (for whatever reason) are given the bottom rank.

Multi-Objective Optimization: All designs which are Pareto-optimal are given rank 1. All remaining designs are then ranked using a recursive method: a design is of rank m if it is Pareto-optimal in the absence of all designs of rank $n<m$. So, for example, rank 2 designs are those which are Pareto-optimal in the absence of rank 1 designs; rank 3 designs are those which are Pareto-optimal in the absence of rank 1 and rank 2 designs, etc. All feasible designs are ranked in this way. All infeasible designs then take the same rank (one below that of the last feasible design). Designs which fail to complete (for whatever reason) are given the bottom rank.

The next M columns of the results tab show the objective function values of the design. If the problem is constrained, the next column indicates whether each constraint has been satisfied or not by displaying a sequence `Y`s or `N`s: if a constraint is satisfied, a `Y` is displayed; if it is not satisfied an `N` is displayed. The order of the sequence of `Y`s and `N`s corresponds to the order the constraints were defined in the optimization dialog.

The penultimate column shows the iteration number of the design, and the final column displays the name of the database corresponding to the design.

To enable easy identification of designs of interest, the rows are colour-coded:

1. Rank 1 designs are coloured blue.
2. Feasible designs which are not of rank 1 are coloured green.
3. Infeasible designs are coloured red.
4. Failed designs (which correspond to jobs which fail to complete) are coloured grey.

It is possible to order the designs by any column, by clicking on the column heading; it is also possible to display only the rank 1 designs by clicking on the `Display Optimal Only` tool-button  .

The results may be written to a text file by clicking on the `Write Results to File` tool-button , and entering a filename in the displayed dialog. The results are written in the order they are displayed on the results tab. The

first line of the file written is a comment line, with self-explanatory headers for each of the columns.

The problem settings may be altered after the optimization process has started by clicking on the **Change Optimization Problem Settings** tool-button ●. This invokes the optimization dialog for the problem. From this dialog, the following may be altered:

1. the design variables used,
2. the ranges of the design variables,
3. the objective functions used (including whether they are to be minimized or maximized),
4. the constraints used.

The set of optimization outputs cannot be changed because changing these has the potential to render all previous iterations useless. (If it is necessary to change the set of optimization outputs, start a new optimization problem.)

The optimization process may be stopped prematurely by clicking on the red **Stop Optimization** tool-button ■. Clicking this button still allows any pending optimization jobs in the Batch Processor to finish (to stop the jobs from completing, do this as normal from the Batch Processor). Once stopped, the optimization process may be started again by clicking on the **Restart Optimization** tool-button ●. This invokes the optimization dialog again, allowing alterations to the problem settings to be made before restarting.

The **Optimization Results** tab may be hidden from the Batch Processor at any time by right-clicking on an empty area of the Batch Processor toolbar and deselecting **Optimization Results Pane**.

Graphical Output

If gnuplot is installed, graphical output may be displayed for the optimization results. The path to the gnuplot executable must first be supplied to the Opera Manager by using the dialog invoked through the menu route **Options->Set External Program Paths** (see Set External Program Paths [page 30]). Then a graphics window is displayed and updated during each iteration of the optimization process. This graphics window displays:

1. a plot of (normalized) objective function values vs. iteration number,
2. plots of each constraint vs. iteration number,

3. a plot of the positions of the designs in design variable space (for problems with two or three design variables), and
4. a plot of the positions of the designs in objective function space (for problems with two or three objective functions).

The graphics window may be hidden in the usual way by clicking the windows **Close** button, and redisplayed by clicking on the **Show Graphics Window** tool-button  on the **Optimization Results** tab toolbar.

Use of the Optimization (*OPN*) File

The problem definition, along with details of each of the iterations, are written to the optimization (*opn*) file. One database may be used to define different optimization problems, each of which may be written to different optimization files. An optimization problem may be restarted using *opn* files in two ways:

1. opening the *opn* file in the Opera Manager (either by double-clicking on it, or using the Menu route **File->Open**), or
2. by right-clicking on the *opn* file from within the Opera Manager and selecting **Restart Optimization**. Either method brings up the optimization dialog. The problem settings may be changed before restarting, however to ensure previous results can be used, the optimization outputs cannot be changed.

The results contained in an optimization file may be viewed by right-clicking on the optimization (*opn*) file in the Opera Manager, and selecting **Show Results Pane**. This brings up the Optimization Results pane, populated with the results. The associated problem settings are as they are defined in the *opn* file.

Opera-2d Example - a Coil Former and Disc

The Optimization Problem

The example starts from the eddy current example in chapter 5 of the Opera-2d User Guide (see Figure 9). The current density and width of the

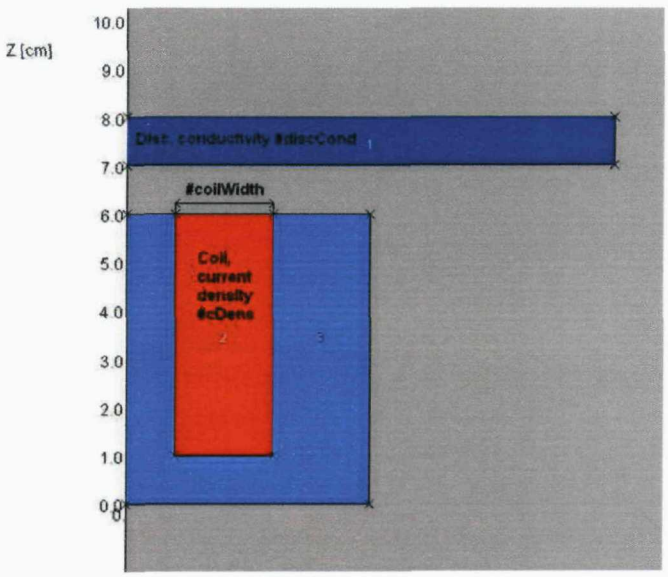


Figure 9 Opera-2d Optimization Example

coil and the conductivity of the disc are allowed to vary within certain limits in order to maximize the force on the disc, subject to a constraint on the maximum current density induced in the disc.

The following parameters are defined:

- the current density in the coil: **`#cDens`**
- the width of the coil **`#coilWidth`**
- the conductivity of the disc: **`#discCond`**
- the force on the disc: **`#forceOnDisc`**
- the peak current density induced in the disc: **`#maxEddyCDens`**

and the problem becomes:

Maximize
#forceOnDisc(#cDens,#coilWidth,#discCond)
subject to
#maxEddyCDens(#cDens,#coilWidth,#discCond)<3.5E6
with constraints
0.5E6<=#cDens<=4E6
0.5<=#coilWidth<=3.5
4E6<=#discCond<=6E6

Preparation of the *OP2* File

The command file *disc.comi* builds the coil former and disc, used in chapters 3 and 5 of the "Opera-2d User Guide". It is parameterized so that the current density is **#cDens**, the width of the coil is **#coilWidth**, and the conductivity of the disc is **#discCond**. At the end of the command file the user variables which are to be used as design variables are changed from constants to model dimensions commands like

```
$modeldimension #cDens
```

1. Run the command file *disc.comi* in Opera-2d/PP.
2. Follow the menu route **File->Save**, enter *disc.op2* as the file name, and choose **Save and analyse now**.

Preparation of Optimization Outputs

We wish to evaluate the values of the optimization outputs (i.e. the objective functions and terms in the (expensive) constraints) in the initial solution file, and then save the commands needed to do this to a comi file. The optimization outputs in this case are the values of the force on the disc, and the peak current density induced in the disc. Once calculated, the values of these outputs need to be stored in user variables, **#forceOnDisc** and **#maxEddyCDens**.

The commands need to do this are described in chapter 5 of the "Opera-2d User Guide". They are given in the command file *disc_post.comi*. This command file also contains an initial comment line with the word "out" followed by a space separated list of the optimization outputs. Before defining the optimization problem, this command file may be run in Opera-2d/PP after reading *disc.ac*, in order to gain understanding of how the outputs are calculated and stored.

Defining the Optimization Problem

From the Opera Manager, right click on the file *disc.op2*, and select **Optimize**. This invokes the optimization dialog.

Defining Design Variables

In this case, all model dimensions are to be set as design variables, so check the **Use** column for each. Enter the desired lower and upper limits for each design variable as shown in Figure 10.

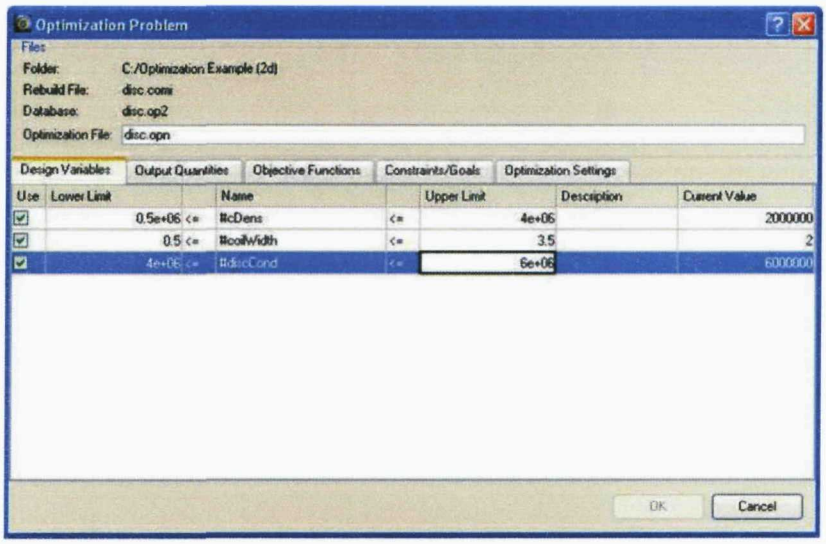


Figure 10 Design Variables Tab

Defining Optimization Outputs

Click on the **Output Quantities** tab, and select the file *disc_post.comi* from the list in the **Command File** column. The outputs defined in the *comi* file are then listed in the **Output Variables** column, as shown in Figure 11.

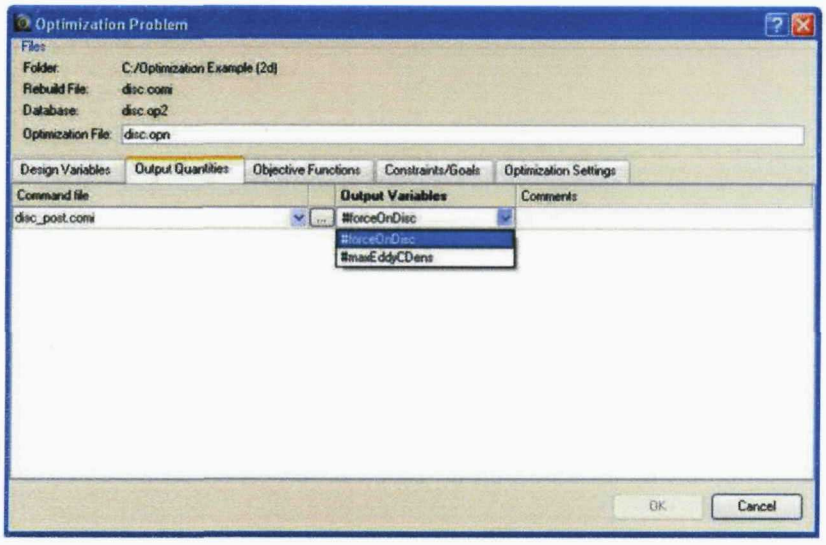


Figure 11 Output Quantities Tab

Defining Objective Functions

Click on the **`Objective Functions`** tab. This displays a tab with each of the optimization outputs listed. We wish to maximize the force on the disc so choose to maximize **`#forceOnDisc`**, as shown in Figure 12. Leave the

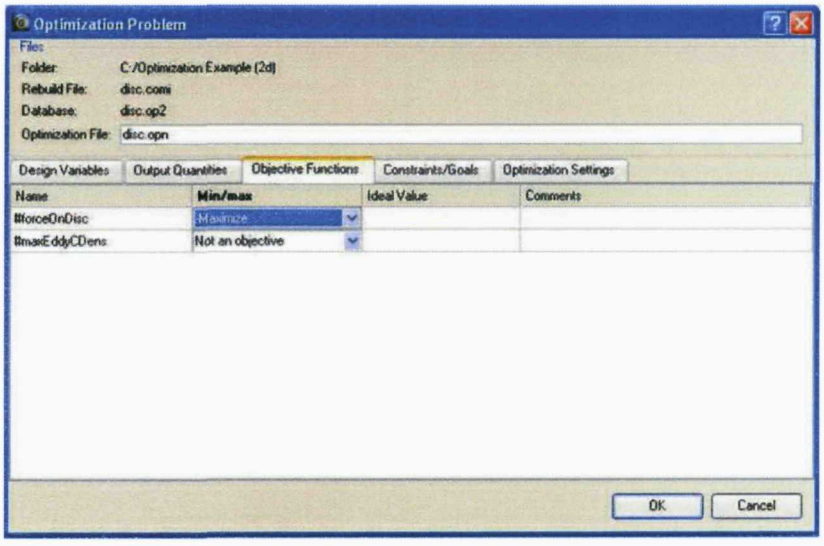


Figure 12 Objective Functions Tab

`Ideal` value blank. Leave **`#maxEddyCDens`** as **`Not an objective`**.

Defining Constraints

Click on the **`Constraints/Goals`** tab. We require the maximum eddy current density induced in the disc, stored in **`#maxEddyCDens`**, to be less than 3.5E6. Select **`#maxEddyCDens`** in the **`Expression 1`** column, the less than or equal sign in the **`Condition`** column, and type 3.5E6 in the **`Expression 2`** column, as shown in Figure 13.

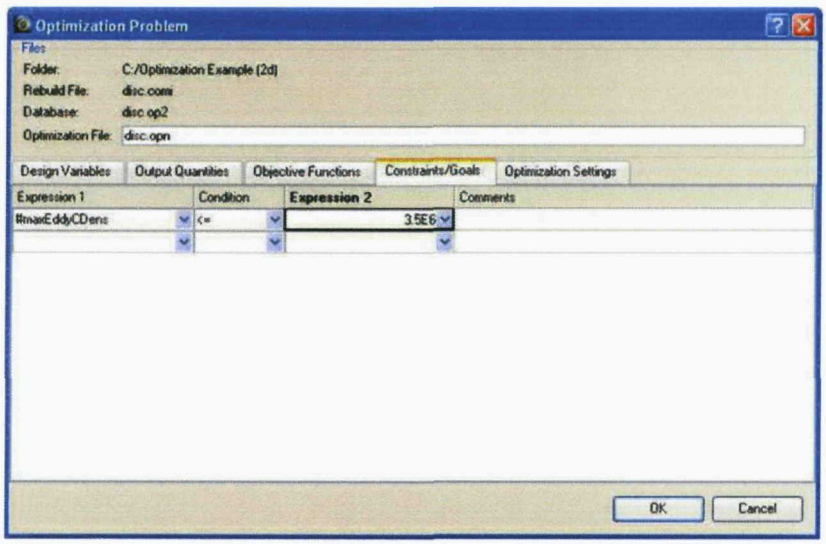


Figure 13 Constraints/Goals Tab

I.4.5) Defining Optimization Settings

Click on the **`Optimization Settings`** tab. Adjust the termination criteria of the optimization algorithm, and the **`Solution Database Settings`** to your own preference (e.g. see Figure 14).

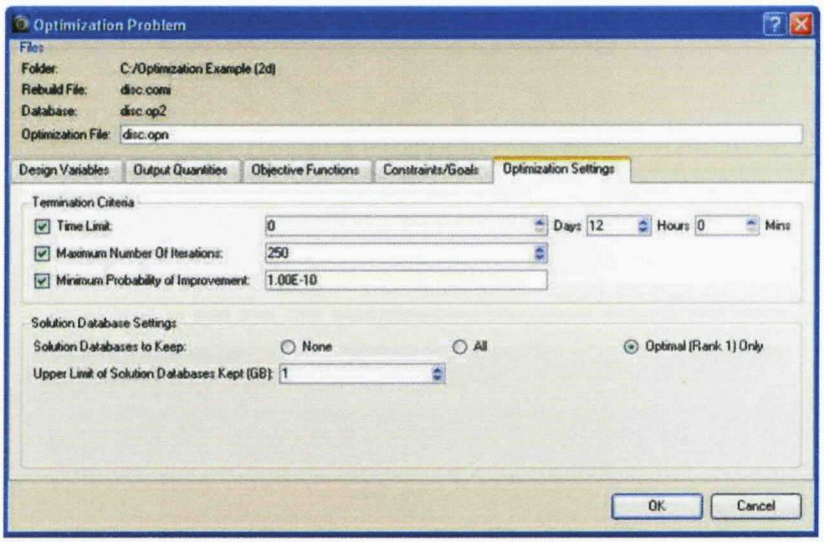


Figure 14 Optimization Settings

Starting the Optimization Process

All parts of the optimization problem have now been defined. When the **OK** button is pressed, the settings defined in the optimization dialog box will be written to an optimization (*opn*) file (of a name of our choosing), and the optimization process will begin.

Enter *disc.opn* as the name of the optimization file, and press **OK**. The optimization process begins.

Viewing Results

Results are displayed in the **Optimization Results** tab in the Batch Processor. They may be reordered by clicking on the appropriate column header. If gnuplot is installed, the results are also displayed graphically, as shown in Figure 15.

Updating Settings

If, after the optimization process has started, we wish to change the problem definition, so that the maximum induced current density allowed is now only 2.0E6, instead of 3.5E6, the problem settings need to be updated.

1. Click on the **Change Optimization Problem Settings** toolbutton . This invokes the optimization dialog for the problem.

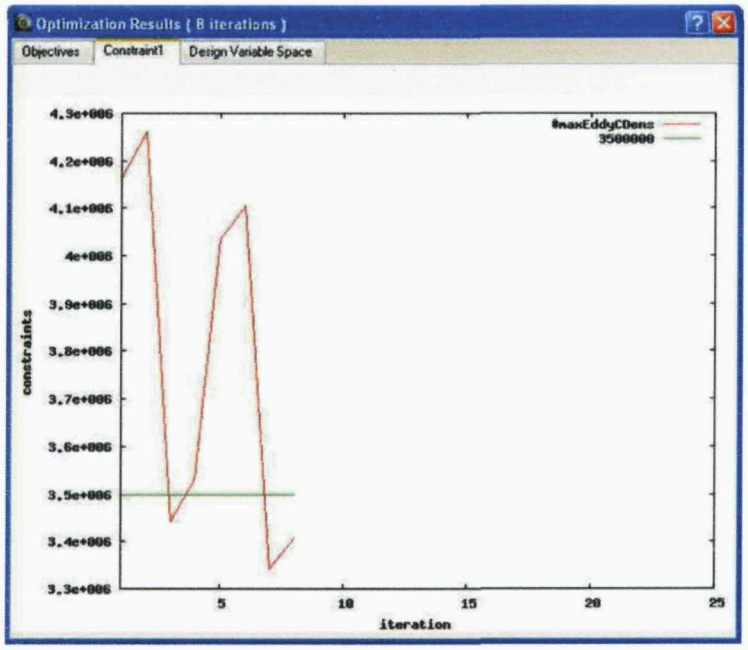


Figure 15 Graph of Constraints

2. Click on the **Constraints/Goals** tab. Change the value of 3.5E6 to 2.0E6. 3.
3. Click **Update** to update the settings.

The updated settings are saved to the *opn* file. These updated settings are used by the optimizer in subsequent iterations. As can be seen from the **Optimization Results** tab, all previous results are reranked according to the new settings.

Restarting from the *OPN* file.

If the optimization process has finished (either through the termination criteria being reached, or by pressing the **Stop Optimization** button on the toolbar of the **Optimization Results** tab), and we wish to restart it, do the following.

1. Right click on *disc.opn* in the Opera Manager and select **Restart Optimization**. This invokes the optimization dialog.
2. Amend the settings if required and click **Restart**.

Opera-3d Example: Helmholtz Coils

The Optimization Problem

The width **#A**, coaxial length **#B**, and half separation **#Z** of a pair of Helmholtz coils, as shown in Figure 16, are allowed to vary within certain limits

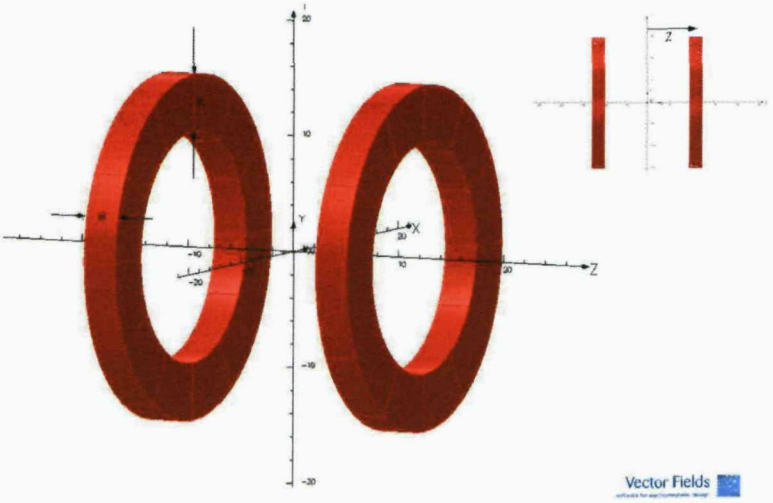


Figure 16 The Helmholtz Pair

in order to achieve a homogenous flux density of 5000 Gauss, inside a sphere of radius 3 about the origin. The homogeneity of the field is characterized by the coefficients of the Legendre polynomials (on a sphere of radius 3); in this case we wish to achieve an A_{00} coefficient of magnitude 5000, and to minimize the magnitude of the error harmonics A_{20}, A_{40}, A_{60}.

Formally this is stated as:

Minimize
abs(A_2_0(#A,#B,#Z)), abs(A_4_0(#A,#B,#Z)),
abs(A_6_0(#A,#B,#Z)
subject to
abs(A_0_0(#A,#B,#Z)=5000
with constraints
0.5<#Z<15
0.5<#A<5
0.5<#B<8

Preparation of the Database

1. Load the example file *helmoholtz.opc* in the modeller, by double-clicking on it in the Opera Manager. This file contains a pair of Helmholtz coils, parameterized such that the half separation between the coils, the width of the coils and the coaxial length of the coils are **#Z**, **#A** and **#B** respectively.
2. Generate the surface mesh and volume mesh for the model. (**Model->Generate Surface Mesh** and **Model->Generate Volume Mesh**).
3. Prepare and solve the database *helmholtz.op3* (see Figure 17).

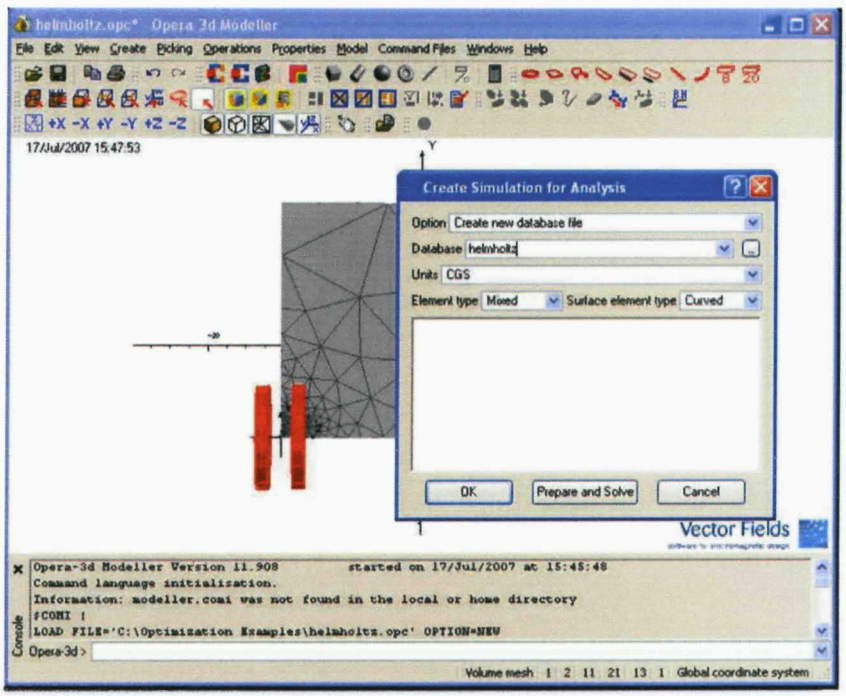


Figure 17 Creating the Database

(**Model->Create Analysis Database** and enter the name *helmholtz*. Click **Prepare and Solve**.).

Preparation of Optimization Outputs

We wish to evaluate the values of the optimization outputs (i.e. the objective functions and terms in the (expensive) constraints) in the initial solved database, and then save the commands needed to do this to a *comi* file. The

optimization outputs in this case are the absolute values of the Lenegendre polynomial coefficients A_0_0, A_2_0, A_4_0 and A_6_0, calculated over the surface of a sphere of radius 3, centred about the origin. Once calculated, the values of these outputs need to be stored in user variables, which we shall call **#Con_1** (for the absolute value of A_0_0), **#Obj_1**, **#Obj_2** and **#Obj_3** (for the absolute values of A_2_0, A_4_0 and A_6_0 respectively).

1. Open the solved database *helmholtz.op3* in the Opera-3d Post-Processor.
2. Use **Fields->Fit Legendre Polynomials to Values...**
3. Click on **Set field point local coordinate system** and set **Local XYZ = Global YZX** to ensure that the spherical polar coordinate system is correctly orientated.
4. Set **Field component** to **Bz**, **Radius of sphere** to 3, **Maximum order** to 10, and select to print the values to screen, as shown in Figure 18.

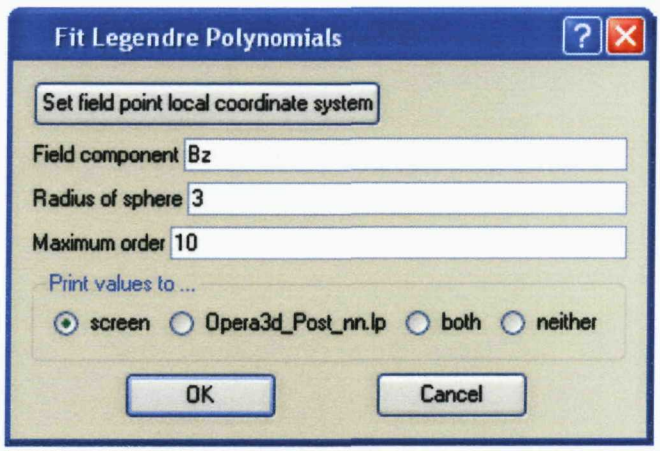


Figure 18 Fit Legendre Polynomials Dialog

Press **OK** to calculate to Legendre coefficients.

The Lengendre polynomial values are printed to screen. We wish to store the absolute values of the components of interest (A_0_0, A_2_0, A_4_0 and A_6_0) as user variables.

5. From the command prompt in the console, issue the following $CONSTANT commands:

```
$cons #Con_1 ABS(a_0_0)
$cons #Obj_1 ABS(a_2_0)
$cons #Obj_2 ABS(a_4_0)
$cons #Obj_3 ABS(a_6_0)
```

The optimization outputs have now been evaluated, and stored in user variables. We wish to save the commands carried out to achieve this to a *comi* file.

6. Open the command file editor (**`Command Files->Command File Editor`**).
7. Use **`File->Open Current Log File...`** to load the commands which have been run. Copy them (Figure 19) and paste them into a

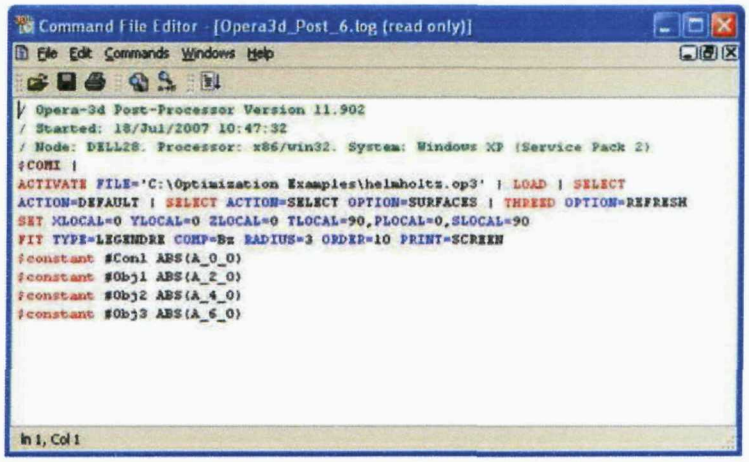


Figure 19 The Commands in the Current Log File

new file.

8. Prepend an extra comment line at the start of the new *comi* file with the word **`out`**, followed by a space-separated list of the optimization outputs, to end up with a file as shown in Figure 20.

   To summarize: Line 1 is the comment line stating explicitly the names of the user variables defined in this comi file which are to be used as optimization outputs. Lines 2-3 evaluate the Lengendre polynomial coefficients of the Bz field over the surface of a sphere of radius 3 (around the origin). Lines 4-7 store the (absolute) values of the A_0_0, A_2_0, A_4_0, A_6_0 coefficients in the user variables which are to be used as optimization outputs.
9. Save the file as *helmholtz_post.comi* in the same folder as *helmholtz.op3*.
10. Close the command file editor, and exit the Post-Processor.

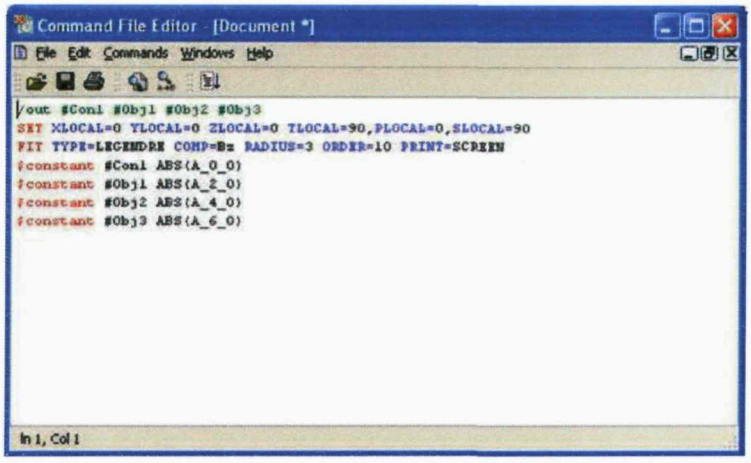


Figure 20 The Post-Processor Command File

Defining the Optimization Problem

From the Opera Manager, right click on the database *helmholtz.op3*, and select **`Optimize`** (Figure 21). This invokes the optimization dialog.

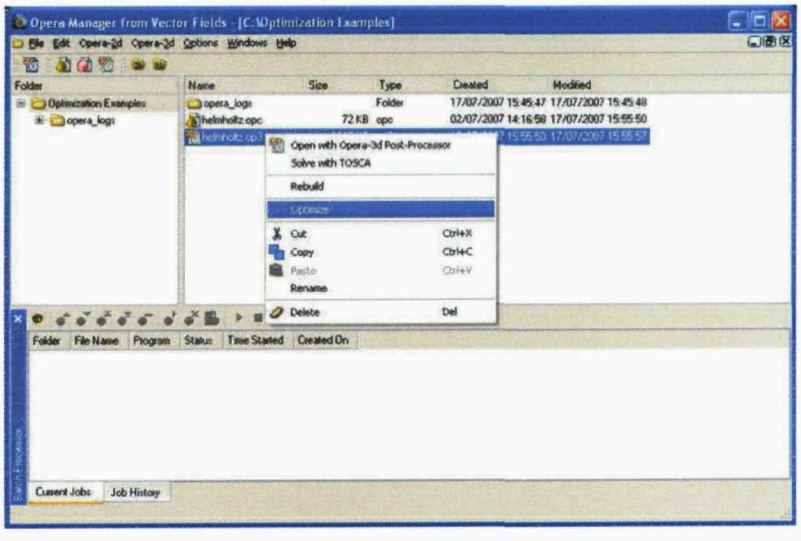


Figure 21 Selecting Optimize from the Manager

Defining Design Variables

The optimization dialog is composed of five tabs, each of which is used to define the settings of the optimization problem. The first tab displays all the

model dimension variables defined in the model. Each model dimension may be set:

1. to vary between numerical limits (i.e. act as a design variable), or
2. to be a constant, or
3. to take a value based on the value of other Model Dimensions (i.e. act as a parameter).

In this case, all model dimensions are to be set as design variables, so check the **Use** column for each. Enter the desired lower and upper limits for each design variable as shown in Figure 22.

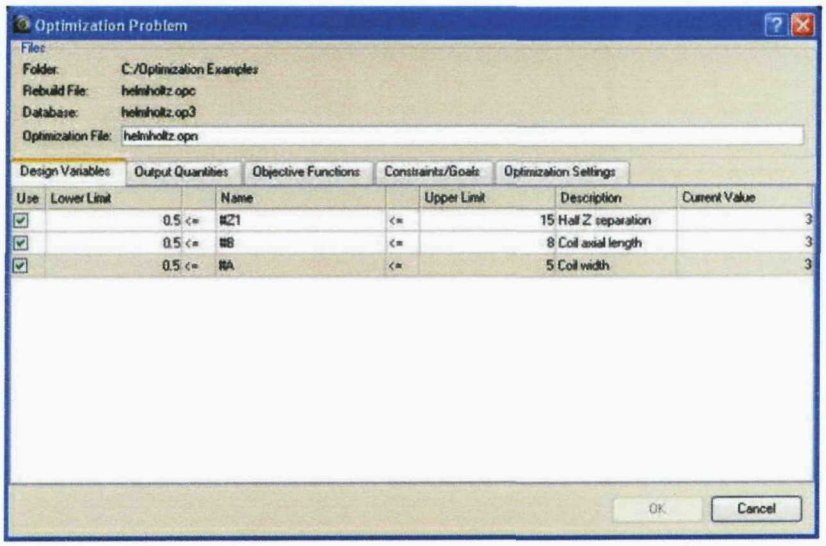


Figure 22 Design Variables Tab

Defining Optimization Outputs

Click on the **Output Quantities** tab, and select the file *helmholtz_post.comi* from the list in the **Command File** column, as shown in Figure 23. The outputs defined in the *comi* file are then listed in the **Output Variables** column.

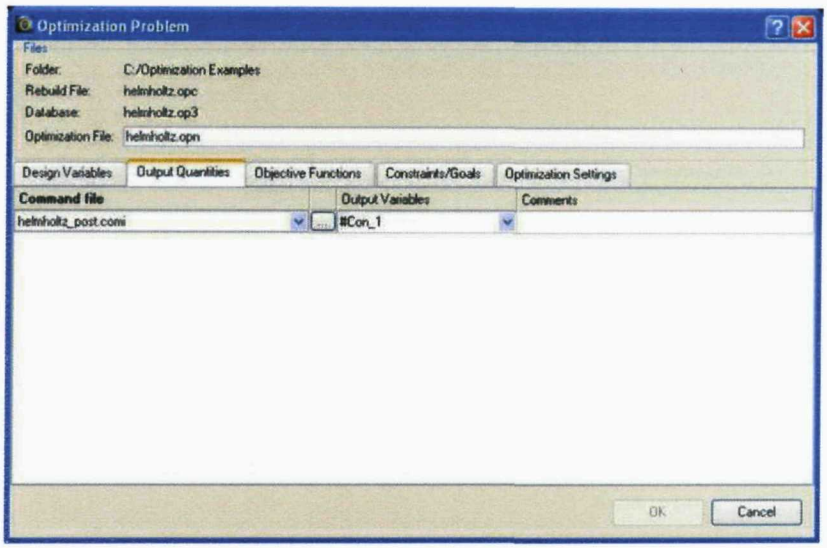


Figure 23 Output Quantities Tab

Defining Objective Functions

1. Click on the **`Objective Functions`** tab. This displays a tab with each of the optimization outputs listed.
2. We wish to minimize the error harmonics A_2_0, A_4_0, A_6_0 (stored in **`#Obj_1`**, **`#Obj_2`** and **`#Obj_3`** respectively). Choose these user variables as shown in Figure 24. The **`Ideal`** values may be

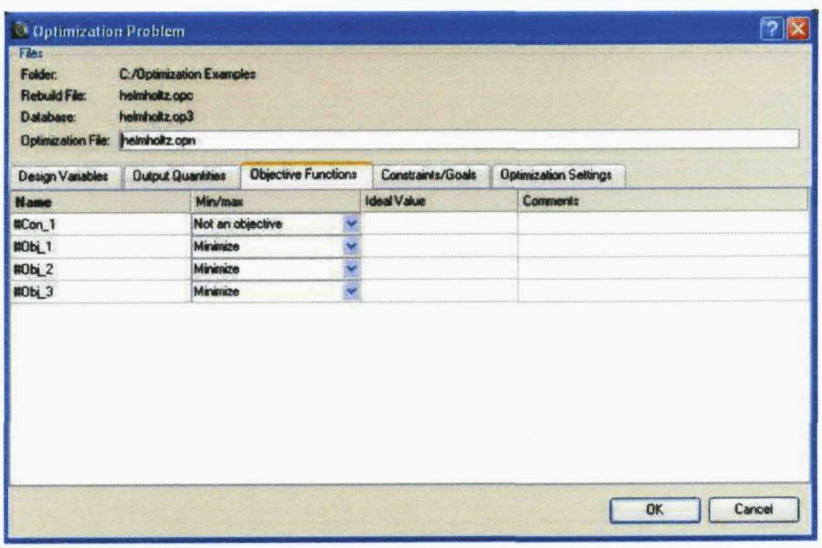


Figure 24 Objective Functions Tab

set for each objective (in this case 0), although this is not necessary. Leave **#Con_1** as **Not an objective**.

Defining Constraints

1. Click on the **Constraints/Goals** tab.
2. We require the magnitude of the A_0_0 Legendre polynomial coefficient, stored in **#Con_1**, to be equal to 5000. Select **#Con1** in the **Expression 1** column, the equality sign in the **Condition** column, and type **5000** in the **Expression 2** column, as shown in Figure 25.

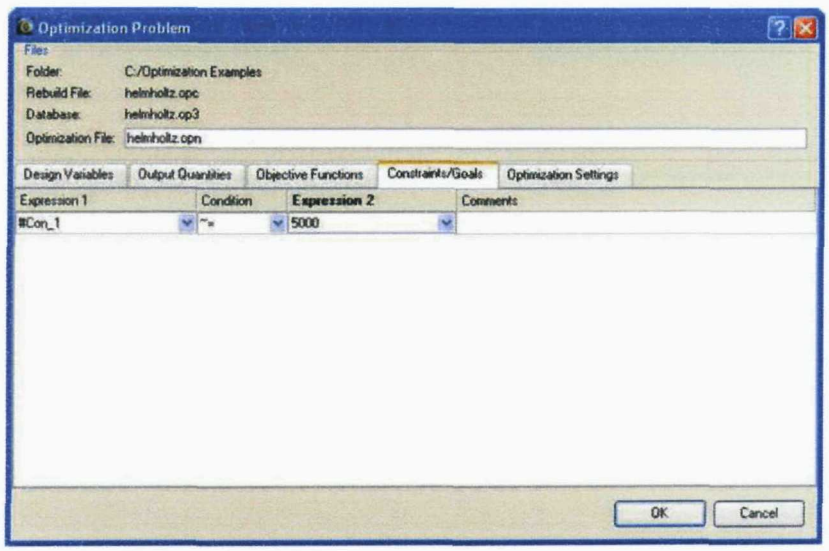


Figure 25 Constraints/Goals Tab

Defining Optimization Settings

1. Click on the **Optimization Settings** tab.
2. Adjust the termination criteria of the optimization algorithm, and the **Solution Database Settings** to your own preference.

Starting the Optimization Process

All parts of the optimization problem have now been defined. When the **OK** button is pressed, the settings defined in the optimization dialog box will be written to an optimization (*on*) file (of a name of our choosing), and the optimization process will begin.

Enter *helmholtz.opn* as the name of the optimization file, and press **OK**. The optimization process begins (see Figure 26).

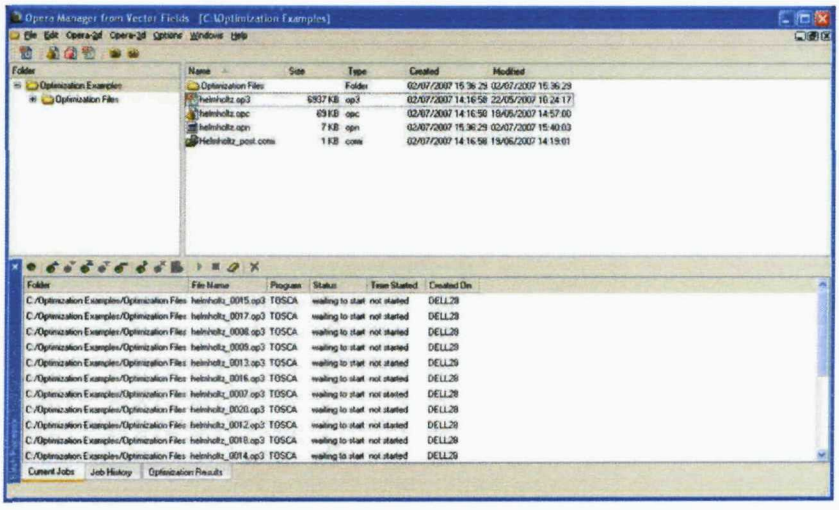


Figure 26 The Optimization about to Start

Appendix E

Conference Papers

The consideration of surrogate model accuracy in single-objective electromagnetic design optimization

G. I. Hawe, School of ECS, University of Southampton, U.K. and Vector Fields Ltd., Oxford, U.K. (glenn.hawe@vectorfields.co.uk)
J. K. Sykulski, School of ECS, University of Southampton, U.K. (jks@soton.ac.uk)

1 Introduction

Optimization problems in electromagnetic design are typified by features which present difficulties to most deterministic search algorithms, e.g. the existence of multiple local minima. Genetic Algorithms (GAs), on the other hand, with their ability to search more globally, are better suited for exploring complicated objective function landscapes. The high computational cost of evaluating the objective function in such problems, however, means that direct use of a GA is often not feasible or impractical, due to the general requirement for a large number of objective function evaluations. Additional cost-effective techniques must be used, with the aim to make the GA require fewer evaluations of the objective function. Techniques used include hybrid algorithms, GAs specially adapted for small population sizes, and simplifying the problem by removing irrelevant design variables. One technique, called *surrogate modelling*, is the focus of this paper.

A surrogate model is a functional relationship between the design variable space of an optimization problem, and the objective function space, which is constructed based on a set of design vectors which have their objective function values known. Having constructed a surrogate model, a GA can then use it to predict fitness values for unevaluated design vectors, rather than call the true expensive objective function, thus reducing computational costs. However, ideally the reliability of the model should be taken into account as well, when choosing points to evaluate; this is discussed further in Section 2. Different methods exist to construct surrogate models, including polynomial approximation, artificial neural networks (ANNs) and kriging; the use of these three types of surrogate model in electromagnetic design optimization is discussed in Section 3. Developments in this area outside the field of electromagnetic design optimization are discussed in Section 4.

2 Model Accuracy

Care should be taken when using a surrogate model to select design vectors to evaluate for optimization purposes. In particular, the existence of false optima (points which are optima of the surrogate model, but which are not optima of the true objective function space, see **Fig 1**) means that selecting points to evaluate based entirely on their predicted objective function value is not desirable. Instead, ideally some measure of the reliability of the predicted objective function value should also be considered, and so the choice of the next point to evaluate becomes a balance between attempting to locate the best points and aiming at improving the accuracy of the surrogate.

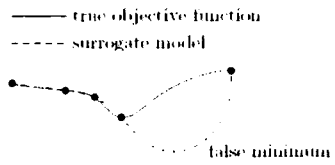


Fig. 1 False minimum in a surrogate model

3 Surrogate-assisted single-objective electromagnetic design optimization

3.1 Polynomial approximations

Polynomial approximation suffers in that inclusion of additional points into the model does not necessarily lead to increased model accuracy. In particular, if only the optimum of the surface is added, the model can converge very quickly to a false optimum [1].

In [2], model accuracy was considered in several ways. The initial set of examples was chosen so as to minimize the condition number of the matrix [M] which was to be inverted in order to determine the polynomial coefficients. A dynamic weighting factor was then used as the optimization process proceeded to place more emphasis on the region around the predicted optimum. Then, in order to ensure that [M] did not become ill-conditioned as the optimization process continued, additional *learning points* were evaluated, chosen specifically so as to minimize the condition number of [M]. The method was successfully used to optimize a brushless permanent magnet motor; an analysis of the errors on predicted optima and learning points indicated that the inclusion of learning points was effective in improving the accuracy of the polynomial surrogate model.

3.2 Artificial Neural Networks

A wide range of different types of ANNs exist which may be used to construct surrogate models. One popular type used is a radial basis function ANN. A method in [3] uses multiquadric radial basis functions to successfully optimize a C-core magnet and a magnetizer. In addition to evaluating the predicted optimum during on-line learning, design vectors in the most *unexplored* regions of design space were also evaluated, with the aim of avoiding local minima; however it is likely this has improved the model accuracy globally as well.

3.3 Kriging

Kriging has recently been recognized as a useful method for surrogate model construction for electromagnetic optimal design [4]. Due to its statistical nature, useful information may be extracted giving an indication of model accuracy and reliability.

The EGO algorithm [5] uses such information to build up an auxiliary function, known as the expected improvement, which automatically balances the objective function values predicted by the kriging model, with the uncertainty in this prediction. By optimizing this auxiliary function, model accuracy increases as the optimum is being searched for. A variation of EGO, known as superEGO, has been used to solve two electromagnetic design problems with expensive objective functions [6], and convergence was found to occur within tens of iterations.

4 Developments Elsewhere

Other algorithms have been developed outside the electromagnetic design community which also consider model reliability when searching for new points. One such approach, based on a radial basis function ANN surrogate model, known as *rbfsolve* [7], predicts the location of a potential new optimum (whose objective function value f* is lower than the current minimum f_{min}) and evaluates a measure of the credibility of the response surface which would interpolate it and the existing data. A measure of the "bumpiness" of the resulting response surface serves as a measure of its credibility, with smoother surfaces being deemed more acceptable.

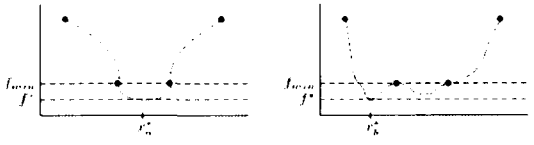


Fig. 2 Two response surfaces which pass through an existing set of examples and a predicted optimum

For example, in **Fig 2**, the proposed optimum (x_a*,f*) is preferred to the proposed optimum (x_b*,f*), as the surface which interpolates it and the existing set of points (shown as black dots) is less "bumpy" than the surface which interpolates (x_b*,f*). The algorithm has performed well on test functions, but has yet to be applied to electromagnetic optimal design problems.

5 Conclusion

Surrogate models have proven to be effective in reducing the cost of electromagnetic optimal design problems. Model reliability has been recognised as an important factor and attempts have been made to ensure model accuracy improves as the optimization search proceeds. However, suitable algorithms exist which are yet to be implemented in electromagnetic design optimization. The full paper will critically assess various surrogate modelling techniques.

6 Literature

[1] Jones, D. R.: A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization, Vol 21, pp. 345-374, 2001

[2] Sykulski, J. K.; Al-Khoury A. H.; Goddard K. F.: Minimal Function Calls Approach With On-Line Learning and Dynamic Weighting for Computationally Intensive Design Optimization. IEEE Transactions on Magnetics, Vol 37, No. 5, pp. 3423-3426, September 2001

[3] Farina, M.; Sykulski, J. K.: Comparitive Study of Evolution Strategies combined with Approximation Techniques for Practical Electromagnetic Optimization Problems. IEEE Transactions on Magnetics, Vol 37, No. 5, pp. 3216-3220, September 2001

[4] Lebensztajn, L.; Marretto, C. A. R.; Costa M. C.; Coulomb J-L.: Kriging: A Useful tool for Electromagnetic Device Optimization. IEEE Transactions on Magnetics, Vol. 40, No. 2, pp. 1196-1199, March 2004

[5] Jones, D. R.; Schonlau M.; Welch W. J.: Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization, Vol. 13, pp. 455-492, 1998

[6] Siah, E. S.; Sasena, M.; Volakis J. L.; Papalambros, P. Y.; Wiese, R. W.: Fast Parameter Optimization of Large-Scale Electromagnetic Objects Using DIRECT with Kriging Metamodeling. IEEE Transactions on Microwave Theory and Techniques, Vol. 52, No. 1, January 2004

[7] Björkman, M.; Holmström, K.: Global optimization of costly nonconvex functions using radialbasis functions. Optimization and Engineering, Vol.1, pp.373-397, 2000

Balancing Exploration and Exploitation using Kriging Surrogate Models in Electromagnetic Design Optimization

G. I. Hawe[1,2] and J. K. Sykulski[1]
[1]School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, U.K.
[2]Vector Fields Ltd., 24 Bankside, Kidlington, Oxford, OX5 1JE, U.K.
E-mail: glenn.hawe@vectorfields.co.uk, jks@soton.ac.uk

Abstract— The balance between exploration and exploitation is an important issue when attempting to find the global minimum of an objective function. This paper describes how this balance may be carefully controlled when using Kriging surrogate models to approximate the objective function.

I. INTRODUCTION

A common technique for reducing computational cost in electromagnetic optimal design problems is to use surrogate models to approximate the relationship between the design variable space and the objective function space. A range of different methods are available for constructing the surrogate model, one of which is Kriging [1]. Kriging surrogate models have the advantage that useful information regarding their accuracy can be obtained, which may then be exploited when choosing the next points to evaluate during optimization, as 'utility functions' may be constructed which seamlessly balance the predicted value of a point's objective function value with the uncertainty in this prediction, thus providing a useful way to achieve the balance between exploration of unknown regions of objective space and exploitation of attractive areas of objective space. This paper discusses the range of such utility functions now available.

II. UTILITY FUNCTIONS

Denote by $y_{\min}$ the minimum objective function value attained in the set of examples used to construct the Kriging surrogate model. At an unevaluated design vector x, denote its objective function value as predicted by the Kriging surrogate model by $\hat{y}$, and the root mean squared error in this prediction by s. Then by writing

$$u = \frac{y_{\min} - \hat{y}}{s}, \tag{1}$$

the "expected improvement" utility function may be defined as [2]:

$$EIF\left[I(x)\right] = (y_{\min} - \hat{y})\Psi(u) + s\psi(u) \tag{2}$$

where Ψ is the standard normal distribution function and ψ is the standard normal density function. This utility function is composed of two terms, the first favoring design vectors with a small predicted objective function value, the second favoring design vectors with large uncertainty in their predicted objective function value. Thus the function is a fixed compromise between exploration and exploitation.

The "generalized expected improvement" utility function is defined as [3]:

$$GEIF\left[I^g(x)\right] = s^g \sum_{k=0}^{g} (-1)^k \left(\frac{g!}{k!(g-k)!}\right) u^{g-k} T_k, \tag{3}$$

where

$$T_k = -\phi(u)u^{k-1} + (k-1)T_{k-2}, \tag{4}$$

with

$$T_0 = \Phi(u) \tag{5}$$

$$T_1 = -\phi(u). \tag{6}$$

The integer parameter g in (3) controls the balance between local and global search, with larger values of g resulting in more emphasis being placed on searching globally. By varying the integer g during an optimization search the emphasis between searching globally and locally can be controlled.

Recently, the "weighted expected improvement" utility function has been proposed in [4]:

$$WEIF\left[I(x)\right] = w(y_{\min} - \hat{y})\Psi(u) + (1-w)s\psi(u) \tag{7}$$

where the real valued parameter w (which is set between 0 and 1) controls the balance between the first and second terms, and thus between searching locally or globally. By varying the value of w during an optimization search, the balance of searching globally and locally can again be controlled.

III. RESULTS

Each of the proposed utility functions are naturally suited for use in electromagnetic design optimization, as they allow the balance of exploration and exploitation to be carefully controlled, whilst computational cost is kept to a minimum. Results on mathematical test functions have shown them to be very efficient in locating global minima. Details of their performance in the optimization of electromagnetic devices will be given in the full paper.

IV. REFERENCES

[1] L. Lebensztajn, C. A. R. Marretto, M. C. Costa and J-L. Coulomb "Kriging: a useful tool for electromagnetic device optimization", *IEEE Transactions on Magnetics*, Vol. 40, No. 2, pp. 1196-1199, March 2004.

[2] D. R. Jones, M. Schonlau and W. J. Welch "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, Vol. 13, pp. 455-492, 1998.

[3] M. J. Sasena, P. Papalambros and P. Goovaerts "Exploration of metamodeling sampling criteria for constrained global optimization", *Engineering Optimization*, Vol. 34, pp. 263-278, 2002.

[4] A. Sobester, S. J. Leary and A. J. Keane "On the design of optimization strategies based on global response surface approximation models", *Journal of Global Optimization*, Vol. 33, pp. 31-59, 2005.

A Hybrid One-Then-Two Stage Algorithm for Computationally Expensive Electromagnetic Design Optimization

G. I. Hawe (*,**), J. K. Sykulski (**)

(*)Vector Fields Ltd., Oxford, U. K. OX5 1JE

(**)School of Electronics and Computer Science,
University of Southampton,
Southampton, SO17 1BJ

Abstract. A novel kriging-assisted algorithm is proposed for computationally expensive single-objective optimization. The principle behind the algorithm is to use information about objective function space at the earliest possible opportunity. After constructing a very small experimental design, a one-stage optimization algorithm is used to select further points to evaluate in design variable space. These points are then used in lieu of a traditional space-filling experimental design to construct the initial kriging model for a normal two-stage optimization algorithm.

Key words: Optimization, Kriging, Design and Analysis of Computer Experiments (DACE).

I. INTRODUCTION

In [1], surrogate-model assisted optimization algorithms are categorized into "two-stage" and "one-stage" varieties. Two-stage algorithms use an experimental design to construct an initial surrogate model (the first stage), and then use this model to determine where to sample next (the second stage), e.g. [2]. This approach has several drawbacks: experimental designs ignore information about objective function space, as they are constructed only to be space-filling in the design variable space, and this may be viewed as being wasteful; also, the surrogate model constructed from the points sampled may not be very accurate. One-stage algorithms do not fit a surrogate model to the observed data. Instead, they choose where to sample next by making a hypothesis about the location of the global minimum, and determining the credibility of surrogate models which would pass through it and the sampled points, e.g. [3]. The point chosen to be sampled is the hypothesised point which has the most credible surrogate model passing through it. This approach has the drawback that the computational cost of locating which point to evaluate next increases dramatically as the number of sampled points increases. In this paper, a hybrid one-then-two stage algorithm, which uses a one-stage algorithm to initialize a two-stage algorithm, is proposed for use with kriging surrogate models [4].

II. ONE-THEN-TWO STAGE ALGORITHM

The proposed algorithm for locating the global minimum in n-dimensional design variable space $D \subset \Re^n$ consists of three steps: initialization, one-stage experimental design, and two-stage optimization search.

A. Initialization

The only purpose of the initialization step is to sample enough points to allow a non-trivial kriging model to be constructed (i.e. a model which is not a hyper-plane in $\Re^n$). The space-filling Hammersley Sequence experimental design [5], of size $2n$, is used to select the points. The experimental design size of $2n$ is much smaller than is normally used ($10n$ is suggested in [2] for example), as the philosophy of this algorithm is to use information about objective function space to search for the minimum at the earliest possible opportunity.

B. One-stage experimental design

Information about objective function space has now been obtained through sampling $2n$ points, and the aim in this second step is to use this information to strategically choose where to sample next. Let the minimum objective function value of the $2n$ sampled points be $f_{\min}$, and let the maximum be $f_{\max}$. A design vector x^* is then hypothesized to exist in design variable space which has an objective function value $f^* = f_{\min} - q(f_{\max} - f_{\min}) < f_{\min}$ where $0 < q \leq 1$. Initially q is set to $q = 1$. Then, the likelihood of the $2n$ sampled points conditional upon a kriging surface passing through (x^*, f^*) is [1]:

$$\frac{1}{(2\pi)^{\frac{n}{2}}(\sigma^2)^{\frac{n}{2}}|\mathbf{C}|^{\frac{1}{2}}} \exp\left[\frac{-(\mathbf{y}-\mathbf{m})\mathbf{C}^{-1}(\mathbf{y}-\mathbf{m})}{2\sigma^2}\right] \tag{1}$$

where:

$$\mathbf{m} = \mathbf{1}\mu + \mathbf{r}(f^* - \mu) \tag{2}$$

$$\mathbf{C} = \mathbf{R} - \mathbf{r}\mathbf{r}^{\mathrm{T}} \tag{3}$$

are the conditional mean and correlation matrices. Here, **R** is the correlation matrix, μ is the mean and σ^2 is the variance predicted by the kriging model. The next point to be evaluated is the x^* which maximizes (1). This value of x^* is the one which yields the most credible kriging surface which interpolates it (with objective function value f^*) and the sampled points. This process is repeated for $8n$ iterations until $10n$ points in total have been evaluated. The value of q is cycled through using $q = |\sin((i\pi/2n)+0.01)|$ at the i^{th} iteration of this stage.

C. Two-stage optimization search

The $10n$ points sampled so far are now used to construct a kriging model. (In this respect, the previous stage may be viewed as the experimental design stage). Utility functions, which balance the values predicted by a kriging model with the uncertainty in the model, are now used to select which points to evaluate next. The generalized expected improvement function [6]:

$$GE\left[I^g(x)\right] = s^g \sum_{k=0}^{g} (-1)^k \left(\frac{g!}{k!(g-k)!}\right) u^{g-k} T_k \tag{4}$$

where

$$T_k = -\phi(u)u^{k-1} + (k-1)T_{k-2} \tag{5}$$

with

$$T_0 = \Phi(u) \tag{6}$$

$$T_1 = -\phi(u) \tag{7}$$

where Φ is the standard normal distribution function and ϕ is the standard normal density function, is used first, as it places most emphasis on regions of high uncertainty, which is what is desired at the beginning of an optimization search. The level of emphasis placed on searching uncertain regions is determined by the value of the integer g, with higher values of g corresponding to more emphasis. A cooling strategy [7] is employed, using high values of g to begin with, and then progressively smaller values. When $g = 1$, the utility function is equivalent to the expected improvement utility function, as used in the EGO algorithm [2]. Upon reaching $g = 1$, we wish to place more emphasis on searching around the current minimum, and to do this the weighted expected improvement utility function [8] is used:

$$WE(I(x)) = w(f_{\min} - \hat{f}(x))\Phi\left(\frac{f_{\min} - \hat{f}(x)}{s(x)}\right) + (1-w)s(x)\phi\left(\frac{f_{\min} - \hat{f}(x)}{s(x)}\right) \tag{8}$$

where $\hat{f}(x)$ is the objective function value as predicted by the kriging surrogate model, and $s(x)$ is the root mean squared error in this prediction. The first term in this function places emphasis on searching around the current minimum, whilst the second term places emphasis on searching regions of design variable space with high uncertainty in their values. The terms are balanced using the weighting parameter w. Using $w = 0.5$ in (8) is equivalent to using $g = 1$ in (4), so we begin using (8) with a value of $w = 0.6$ to select points. Then different values of w are cycled through, using $w = 0.5 + 0.5|\cos(i\pi/2n)|$ at the i^{th} use of (8). The algorithm then proceeds either for a fixed number of iterations, or until an adequate solution has been found.

III. RESULTS AND CONCLUSIONS

Results on test functions have been encouraging. A full analysis of the performance of the proposed algorithm and its application to electromagnetic device optimization will be discussed in the full paper.

REFERENCES

[1] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces", *Journal of Global Optimization*, vol. 21, 2001, pp. 345-383.

[2] D. R. Jones, M. Schonlau, W. J. Welch "Efficient Global Optimization of Expensive Black-Box Functions", *Journal of Global Optimization*, vol. 21, 2001, pp. 455-492.

[3] H. M. Gutmann, "A Radial Basis Function Method for Global Optimization", *Journal of Global Optimization*, vol. 19, no. 3, 2001, pp. 201-227.

[4] L. Lebensztajn, C. A. R. Marretto, M. C. Costa, J-L. Coulomb, "Kriging: A Useful Tool for Electromagnetic Devices Optimization", *IEEE Transactions on Magnetics*, vol. 40, no. 2, 2004, pp 1196-1199.

[5] J. R. Kalagnanam, U. M. Diwekar "An Efficient Sampling Technique for Off-line Quality Control", *Technometrics*, vol. 39, no. 3, 1997, pp. 308-319.

[6] M. Schonlau, W. J. Welch, D. R. Jones, "Global versus Local Search in Constrained Optimization of Computer Models", *IMS Lecture Notes*, vol. 34, 1998 pp. 11-25.

[7] M. J. Sasena, P. Papalambros, P. Goovaerts, "Exploration of Metamodelling Sampling Criteria for Constrained Global Optimization", *Engineering Optimization*, vol. 34, 2002, pp.263-278.

[8] A. Sobester, S. J. Leary, A. J. Keane, "On the Design of Optimization Strategies Based on Global Response Surface Approximation Models", *Journal of Global Optimization*, vol. 33, 2005, pp. 31-59.

An Enhanced Probability of Improvement Utility Function for Locating Pareto Optimal Solutions

Glenn I. Hawe*,† and Jan K. Sykulski†
*Vector Fields, Ltd., 24 Bankside, Kidlington, Oxford, U.K. OX5 1JE
†School of Electronics and Computer Science, University of Southampton, Southampton, U.K. SO17 1BJ
Email: glenn.hawe@vectorfields.co.uk, jks@soton.ac.uk

***Abstract*— This paper describes a novel utility function for choosing design vectors to evaluate in multi-objective optimization problems which are statistically most probable to be Pareto-optimal, given the points already evaluated. The method is tunable to the number of existing Pareto-optimal solutions that an unevaluated design vector is sought to dominate, is naturally parallelized, and removes any need for combining the multiple objectives into a single objective with a scalarizing function.**

I. Introduction

In [1], the probability of improvement criteria was identified as an effective utility function to use with kriging surrogate models [2] for locating design vectors to evaluate in single-objective optimization. By modeling the prediction of a kriging model as the realization of a Gaussian distribution, with mean $\hat{f}(\mathbf{x})$ and standard error $s(\mathbf{x})$ as given by the kriging model, the probability of an unevaluated design vector $\mathbf{x}$ having an objective function value less than T is

$$P(f(\mathbf{x}) < T) = \Phi\left(\frac{T - \hat{f}(\mathbf{x})}{s(\mathbf{x})}\right) \tag{1}$$

where Φ is the normal cumulative distribution function. This is illustrated below in Fig. 1; the probability of the unevaluated design vector x^* having an objective function value less than T is represented by the shaded region.

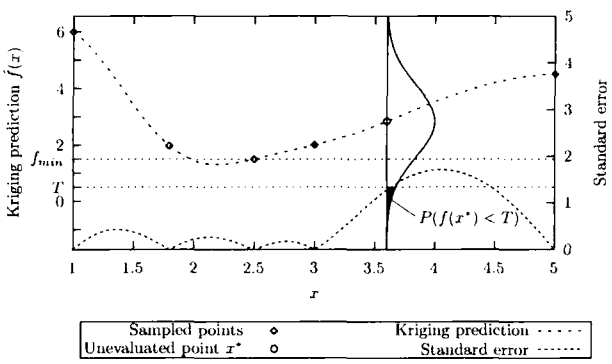


Fig. 1. Uncertainty in objective function value for an unevaluated design vector as predicted by a kriging model.

This method of selecting design vectors to evaluate suffered one drawback however, in that it was sensitive to the level of improvement sought (i.e. the value of T). Two methods were proposed in [1] for overcoming this sensitivity:

1) Evaluate the design vector which maximizes the expectation of the improvement, or
2) Evaluate several design vectors per iteration, each corresponding to a different level of improvement (i.e. a different value of T).

The first method led to the development of the EGO algorithm [3], which has subsequently received significant attention in the literature. However, the second method, referred to as 'enhanced probability of improvement' is the approach recommended in [1]. It has the advantage that it is very robust and easily parallelized, although it suffers in that the different levels of improvement sought in each iteration (i.e. the different values of T used) are arbitrary. This paper extends this enhanced method to the multi-objective case, where it shall be seen that natural levels of improvement exist.

II. Probability of Improvement and Expected Improvement in Multi-Objective Optimization

The concept of improvement in multi-objective optimization has only appeared recently in the literature. First, recall that a solution S_a (with design vector $\mathbf{x}_a$) is said to dominate another solution S_b (with design vector $\mathbf{x}_b$) if and only if S_a is strictly better than S_b in at least one objective, and no worse in all other objectives. The set of Pareto-optimal solutions are then those solutions which are not dominated by any other existing solution. Suppose then, that a set S of N_{par} Pareto-optimal solutions exists, $S = \{S_1, S_2, \ldots, S_{N_{\text{par}}}\}$, after performing an experimental design. Then a design vector $\mathbf{x}$ is said to yield an improvement if it is non-dominated by the solutions in S [4]. This may happen in one of two ways:

1) $\mathbf{x}$ dominates at least one of the solutions in S, or
2) $\mathbf{x}$ does not dominate any solution in S, nor does any solution in S dominate $\mathbf{x}$.

This is shown in Fig. 2 for the two-objective case, when $N_{\text{par}} = 5$ Pareto solutions exist. Design vectors which yield an improvement map to either of the shaded regions; design vectors which dominate at least one solution in S map to the region labeled 'Dominating Designs', whilst design vectors which do not dominate any solution in S (but which still constitute an improvement) map to the region labeled 'Equivalent Designs'. Equations for the probability of an unevaluated design vector yielding an improvement are given in [4] and [5], and the equation for the probability of an unevaluated design vector dominating at least one solution in S for the two-objective case is given in [5]. Both [4] and [5] report problems

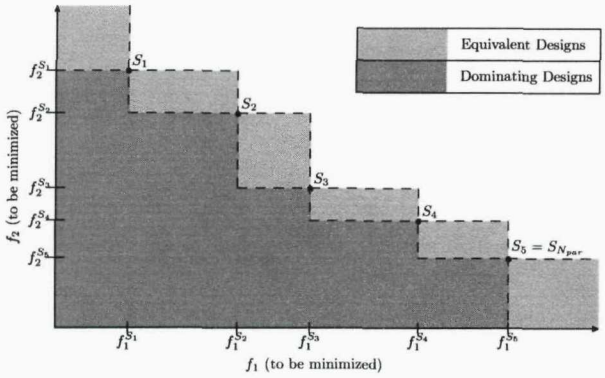


Fig. 2. Regions of dominance and equivalence for $N_{\text{par}} = 5$ Pareto solutions.

with the probability of improvement criteria yielding searches which are not very global, and so only small improvements are made. Both overcome this by instead maximizing the first moment of the probability of improvement around the Pareto-optimal front, which is the multi-objective equivalent of the expected improvement utility function, i.e. they use the first method suggested in Section 1 for overcoming the equivalent sensitivity problem in the single-objective case. The following Section shows how to overcome this sensitivity problem in the multi-objective case using the second method proposed in Section 1, that is, it proposes an enhanced probability of improvement criteria for multi-objective optimization.

III. Enhanced Probability of Improvement in Multi-Objective Optimization

In the multi-objective case, $k = N_{\text{par}}$ natural levels of improvement may be defined, where the k^{th} level of improvement yields a solution which dominates *exactly* k of the existing Pareto-optimal solutions. In addition, a level of equivalence may also be defined ($k = 0$), which yields an additional Pareto-optimal solution which does not dominate any of the existing Pareto-optimal solutions (i.e a design vector which maps to the region labeled 'Equivalent Designs' in Fig. 2). These levels of improvement are shown in Fig. 3 for the $N_{\text{par}} = 5$ Pareto solutions considered earlier.

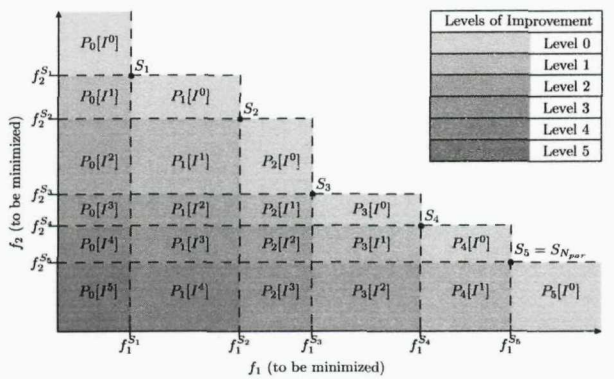


Fig. 3. Probability of improvement levels for $N_{\text{par}} = 5$ Pareto solutions.

As can be seen, in the two-objective case, for an improvement level k, a design vector may map to $N_{\text{par}} - k + 1$ regions of objective function space. Denoting $P(I^k(\mathbf{x}))$ as the probability that an unknown design vector $\mathbf{x}$ yields a level of improvement k (i.e. it dominates exactly k existing Pareto-optimal solutions), $P_i(I^k(\mathbf{x}))$ as the probability that design vector $\mathbf{x}$ will dominate the k Pareto solutions $S_{i+1}, S_{i+2}, \ldots, S_{i+k}$ (these sub-regions are labeled in Fig. 3), and defining

$$\Phi_1^i(\mathbf{x}) := \Phi\left(\frac{f_1^{S_i} - \hat{f}_1(\mathbf{x})}{s_1(\mathbf{x})}\right) \quad (2)$$

$$\Phi_2^i(\mathbf{x}) := \Phi\left(\frac{f_2^{S_i} - \hat{f}_2(\mathbf{x})}{s_2(\mathbf{x})}\right) \quad (3)$$

with

$$\Phi_1^0(\mathbf{x}) := 0 \quad (4)$$

$$\Phi_1^{N_{\text{par}}+1}(\mathbf{x}) := 1 \quad (5)$$

$$\Phi_2^0(\mathbf{x}) := 1 \quad (6)$$

$$\Phi_2^{N_{\text{par}}+1}(\mathbf{x}) := 0 \quad (7)$$

where $\hat{f}_1(\cdot)$ and $s_1(\cdot)$ are the kriging predictions and standard errors for the first objective function respectively (similarly for the second objective function), and $f_1^{S_i}$ is the first objective function value of the i^{th} Pareto solution (similarly for the second objective function), then:

$$P(I^k(\mathbf{x})) = \sum_{i=0}^{N_{\text{par}}-k} P_i(I^k(\mathbf{x})) \quad (8)$$

$$= \sum_{i=0}^{N_{\text{par}}-k} (\Phi_1^{i+1}(\mathbf{x}) - \Phi_1^i(\mathbf{x}))(\Phi_2^{k+i}(\mathbf{x}) - \Phi_2^{k+i+1}(\mathbf{x})). \quad (9)$$

Furthermore, denoting by $P^*(I^k(\mathbf{x}))$ the probability that $\mathbf{x}$ will dominate *at least* k existing Pareto-optimal solutions, then

$$P^*(I^k(\mathbf{x})) = \sum_{j=k}^{N_{\text{par}}} \sum_{i=0}^{N_{\text{par}}-j} (\Phi_1^{i+1}(\mathbf{x}) - \Phi_1^i(\mathbf{x}))(\Phi_2^{j+i}(\mathbf{x}) - \Phi_2^{j+i+1}(\mathbf{x})). \quad (10)$$

Equations (9) and (10) are two multi-objective equivalents of the 'enhanced probability of improvement' in single-objective optimization, for the case of two objectives. The method is extensible to higher numbers of objectives.

IV. Conclusions

A novel utility function, which is easily parallelized and which does not require normalization of objective functions, has been proposed for use in computationally expensive multi-objective optimization. Its performance in optimal electromagnetic design will be discussed in the full paper.

References

[1] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces", *Journal of Global Optimization*, vol. 21, pp. 345–383, 2001.

[2] L. Lebensztajn, C. A. R. Marretto, M. C. Costa and J-L. Coulomb, "Kriging: a useful tool for electromagnetic devices optimization", *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 1196–1199, 2004.

[3] D. R. Jones, M. Schonlau and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions", *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[4] M. T. M. Emmerich, K. C. Giannakoglou and B. Naujoks, "Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels", *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 421-439, 2006.

[5] A. J. Keane, "Statistical Improvement Criteria for Use in Multiobjective Design Optimization", *AIAA Journal*, vol. 44, no. 4, pp. 879–891, 2006.

A Scalarizing One-Stage Algorithm for Efficient Multi-Objective Optimization

Glenn I. Hawe*,† and Jan K. Sykulski†
*Vector Fields, Ltd., 24 Bankside, Kidlington, Oxford, U.K. OX5 1JE
†School of Electronics and Computer Science, University of Southampton, Southampton, U.K. SO17 1BJ
Email: glenn.hawe@vectorfields.co.uk, jks@soton.ac.uk

***Abstract*— A novel kriging-assisted algorithm is proposed for computationally expensive multi-objective optimization problems, such as those which arise in electromagnetic design. The algorithm combines the multiple objectives into a single objective, which it then optimizes using a one-stage method from single-objective optimization. Its efficiency is demonstrated by comparison to a random search on a difficult test function.**

I. Introduction

One popular method of reducing the high computational cost of evaluating objective functions in electromagnetic optimal design problems is the use of surrogate models, such as kriging [1].

Surrogate-model assisted single-objective optimization algorithms may be categorized into 'two-stage' and 'one-stage' varieties [2]. At each iteration of a two-stage algorithm, a surrogate model is constructed from the sampled points (the first stage), and then this model is used to determine where to sample next (the second stage), e.g. [3]. On the other hand, one-stage algorithms choose where to sample next by making hypotheses about the location of the global minimum, and determining the credibility of surrogate models which would pass through each hypothesized optimum and the sampled points, e.g. [4]. The point chosen to be sampled is the hypothesized point which has the most credible surrogate model passing through it. Results on test functions show one-stage methods to be extremely efficient.

One popular technique for solving multi-objective optimization problems (MOOPs) is to combine the multiple objectives into a single objective [5] and then optimize this, e. g. [6]. This paper proposes a novel multi-objective algorithm, which uses a one-stage kriging algorithm to optimize a MOOP, which is scalarized using an augmented Tchbeycheff function [5]. It is believed this is the first time one-stage methodology has been used for multi-objective optimization.

II. One-Stage Kriging Methodology

A brief overview of the one-stage kriging methodology is first given. Suppose that objective function f is a function of an n-dimensional design vector, and suppose N design vectors, $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$ have been evaluated. Given the objective function values of these N design vectors, a hypothesis is made about the value of the objective function at the global minimum of f. Specifically, the global minimum is hypothesized to have an objective function value f^*. Then defining the Gaussian correlation function R (which expresses how two design vectors $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are correlated) as

$$R(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \prod_{k=1}^{n} e^{-\theta_k |x_k^i - x_k^j|^{p_k}} \tag{1}$$

(where θ_k determines how rapid the correlation is lost in the k^{th} design variable, and p_k determines the 'smoothness' of the function in the k^{th} design variable), the $N \times 1$ correlation vector $\mathbf{r}$ as

$$\mathbf{r}(\mathbf{x}) = [R(\mathbf{x}, \mathbf{x}^{(1)}), R(\mathbf{x}, \mathbf{x}^{(2)}), \ldots, R(\mathbf{x}, \mathbf{x}^{(N)})]^{\mathrm{T}}, \tag{2}$$

the $N \times N$ correlation matrix $\mathbf{R}$ as the matrix whose $i-j^{\text{th}}$ entry is $R(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, the $N \times 1$ vector $\mathbf{y}$ as the vector filled with the objective function values of the sampled points,

$$\mathbf{y} = [f(\mathbf{x}^{(1)}), f(\mathbf{x}^{(2)}), \ldots, f(\mathbf{x}^{(N)})]^{\mathrm{T}} \tag{3}$$

and $\mathbf{1}$ as the $N \times 1$ vector filled with ones, then for any design vector $\mathbf{x}^*$, the likelihood of the N examples conditional upon the surface passing through $(\mathbf{x}^*, f^*)$ is [2]:

$$\frac{1}{(2\pi)^{N/2}(\sigma^2)^{N/2}|\mathbf{C}|^{1/2}} \exp\left(\frac{-(\mathbf{y}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{y}-\mathbf{m})}{2\sigma^2}\right) \tag{4}$$

where

$$\mathbf{m} = \mathbf{1}\beta + \mathbf{r}(\mathbf{x}^*)(f^* - \beta) \tag{5}$$

$$\mathbf{C} = \mathbf{R} - \mathbf{r}(\mathbf{x}^*)\mathbf{r}^T(\mathbf{x}^*) \tag{6}$$

are the conditional mean and conditional correlation matrix respectively. The next design vector to be evaluated is the one which maximizes the conditional likelihood in Eq. (4) (which itself is maximized over $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$, $\mathbf{p} = [p_1, p_2, \ldots, p_n]$, β and σ^2 for each $\mathbf{x}^*$). This design vector is the one which, if it had objective function value f^*, would yield the most credible kriging model interpolating it and the N design vectors already observed. Note that setting the derivatives of Eq. (4) with respect to σ^2 and β equal to zero and rearranging, it is found that:

$$\sigma^2 = \frac{(\mathbf{y}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{y}-\mathbf{m})}{N} \tag{7}$$

$$\beta = \frac{\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{y} + f^*\mathbf{r}^T\mathbf{C}^{-1}\mathbf{r} - \mathbf{y}\mathbf{C}^{-1}\mathbf{r} - f^*\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{r}}{\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{1} - 2\mathbf{1}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{r} + \mathbf{r}^T\mathbf{C}^{-1}\mathbf{r}}. \tag{8}$$

and so Eq. (4) only needs to be maximized over θ and $\mathbf{p}$. The following Section proposes a simple way of extending the one-stage method for use in multi-objective optimization.

III. Scalarizing One-Stage Algorithm for Multi-Objective Optimization

After normalizing the n_{obj} objectives of the MOOP using either known or estimated limits of the objective function space, so that each objective function lies within the range [0,1], they are combined using the augmented Tchebycheff function [5]:

$$f_\lambda(\mathbf{x}) = \max_{j=1}^{n_{\text{obj}}}(\lambda_j f_j(\mathbf{x})) + \rho \sum_{j=1}^{n_{\text{obj}}} \lambda_j f_j(\mathbf{x}) \qquad (9)$$

where ρ is a small positive value set (arbitrarily) to 0.05, and $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_{n_{\text{obj}}}]$ is a normalized weight vector.

A Latin Hypercube experimental design [7] of size $5n$ is initially carried out. This is used to initialize $n_{\text{w}} (\geq n_{\text{obj}} + 1)$ *independent* optimization searches, where each search:

- uses a different weighting vector λ, so as to converge towards a different region of the Pareto-optimal front,
- is (arbitrarily) $i_{\max} = 5n$ iterations in length, and
- uses the one-stage kriging method described in Section II with a target $f^* = \kappa(i) f_{\min}$ at iteration i, where $f_{\min}$ is the current minimum value of f_λ and $\kappa(i) = 0.95(0.5 + \frac{i}{2i_{\max}})$ is a scaling factor used to make the search less exploratory (more exploitative) as the iterations proceed.

In the initial $n_{\text{obj}} + 1$ searches, n_{obj} extreme weighting vectors $(1 - \epsilon, \frac{\epsilon}{n_{\text{obj}}-1}, \frac{\epsilon}{n_{\text{obj}}-1}, \ldots, \frac{\epsilon}{n_{\text{obj}}-1}), (\frac{\epsilon}{n_{\text{obj}}-1}, 1 - \epsilon, \frac{\epsilon}{n_{\text{obj}}-1}, \ldots, \frac{\epsilon}{n_{\text{obj}}-1}), \ldots, (\frac{\epsilon}{n_{\text{obj}}-1}, \frac{\epsilon}{n_{\text{obj}}-1}, \frac{\epsilon}{n_{\text{obj}}-1}, \ldots, 1 - \epsilon)$ (where $|\epsilon| \ll 1$) which each heavily favor only one objective, are used, as well as the weighting vector which places equal emphasis on each objective, $(\frac{1}{n_{\text{obj}}}, \frac{1}{n_{\text{obj}}}, \ldots, \frac{1}{n_{\text{obj}}})$. After these first $n_{\text{obj}} + 1$ searches, a further $n_{\text{w}} - n_{\text{obj}} - 1$ searches are then performed, with the weight vector set each time so that the value of its components is the average of the corresponding components of the two weight vectors which yielded points which bound the emptiest region of objective function space. Note that this procedure means the algorithm has a fixed number of iterations, $n_{\text{iter}} = 5(n_{\text{w}} + 1)n$.

It should be emphasized that each optimization search is completely independent of the others: the results from one search are not used in another. This has several benefits:

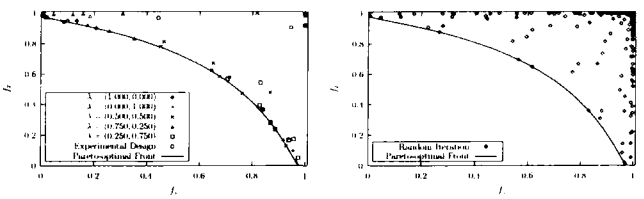
1) The algorithm may be easily parallelized.
2) The conditional correlation matrix $\mathbf{C}$ which is to be inverted to calculate the credibility never exceeds a size of $10n \times 10n$. This is extremely important in keeping the computational cost of the one-stage approach as low as possible.
3) For any given weight vector, the iterations tend to concentrate more and more around one particular region of design variable space. By ignoring the iterations of other searches (using different weight vectors), the degree to which $\mathbf{C}$ becomes ill-conditioned is severely reduced.

IV. Results

The algorithm was tested on a range of test functions; due to lack of space, results are given for one only, VLMOP2 [8]:

$$\begin{aligned} \text{Minimize} \quad f_1(\mathbf{x}) &= 1 - \exp\Big(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2\Big) \\ \text{and} \quad f_2(\mathbf{x}) &= 1 - \exp\Big(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2\Big) \\ \text{with} \quad x_i &\in [-4, 4] \end{aligned}$$

where $n = 2$. $n_{\text{w}} = 5$ different weighting vectors were used in the scalarizing one-stage algorithm, giving $n_{\text{iter}} = 60$ iterations in total. These iterations are shown in Fig. 1 (a) below (each iteration being identified by the weight vector used). For comparison, a random search of 500 iterations was also performed, and these iterations are shown in Fig. 1 (b). As can be seen, the scalarizing one-stage algorithm is much more efficient at locating solutions close to the Pareto-optimal front. Similar results were observed in other test functions.



(a) 60 iterations of scalarizing one-stage algorithm. (b) 500 iterations of random search algorithm.

Fig. 1. Results on VLMOP2 test function.

V. Conclusion

A novel algorithm has been proposed which uses, for the first time, a one-stage methodology for multi-objective optimization. It performed efficiently on a difficult test problem; its performance in multi-objective electromagnetic design optimization will be discussed in the full paper.

References

[1] L. Lebensztajn, C. A. R. Marretto, M. C. Costa and J-L. Coulomb, "Kriging: a useful tool for electromagnetic devices optimization", *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 1196–1199, 2004.

[2] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces", *Journal of Global Optimization*, vol. 21, pp. 345–383, 2001.

[3] D. R. Jones, M. Schonlau and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions", *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[4] H. M. Gutmann, "A Radial Basis Function Method for Global Optimization", *Journal of Global Optimization*, vol. 19, pp. 201–227, 2001.

[5] K. M. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, 1999.

[6] J. Knowles, "ParEGO: A Hybrid Algorithm With On-Line Landscape Approximation for Expensive Multiobjective Optimization Problems", *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[7] M. D. McKay, W. J. Conover and R. J. Beckmantitle, "A comprison of three methods for selecting values of input variables in the analysis of output from a computer code" *Technometrics*, vol. 21, pp. 239-245, 1979.

[8] A. Van. Veldhuizen, "Multiobjective Evolutionary Algorithms: Classifications, Analyzes, and New Innovations", PhD. Dissertation, Air Force Institute of Technology, Wright-Patterson AFB, Jun. 1999.

ISEF 2007 - XIII International Symposium on Electromagnetic Fields
in Mechatronics, Electrical and Electronic Engineering
Prague, Czech Republic, September 13-15, 2007

SCALARIZING COST-EFFECTIVE MULTIOBJECTIVE OPTIMIZATION ALGORITHMS MADE POSSIBLE WITH KRIGING

G. I. Hawe[1,2] and J. K. Sykulski[2]

[1]Vector Fields Ltd., 24 Bankside, Kidlington, Oxford, U. K.
[2]School of ECS, University of Southampton, U. K.

Abstract *– **The use of kriging in cost-effective single-objective optimization is well established, and a wide variety of different criteria now exist for selecting design vectors to evaluate in the search for the global minimum. Additionly, a large number of methods exist for transforming a multi-objective optimization problem to a single-objective problem. With these two facts in mind, this paper discusses the range of kriging assisted algorithms which are possible (and which remain to be explored) for cost-effective multi-objective optimization.***

Selection Criteria with Kriging Models for Single-Objective Optimization

The use of kriging surrogate models in single-objective optimization is now well established (including within the electromagnetic design community [1]) and a wide variety of methods now exist for exploiting statistical information from them for the purpose of selecting a design vector to evaluate in the search for the global minimum [2]. These include (but are not limited to) evaluating the design vectors which maximize: the probability of improvement (POI) [2]; the expectation value of the improvement (EI) [3]; the generalized expected improvement (GEI) [4]; the weighted expected improvement (WEI) [5]; the credibility of a hypothesis (CH) about the location of the minimum [2] (also known as the one-stage approach); and most recently the '*minimizer entropy*' (ME) criterion [6] . Due to lack of space, their exact descriptions are omitted; suffice to say several allow the delicate balance between exploration and exploitation to be controlled through 'cooling' schemes; furthermore many offer the facility for selecting multiple design vectors for evaluation each iteration.

Scalarization of Multi-Objective Optimization Problems

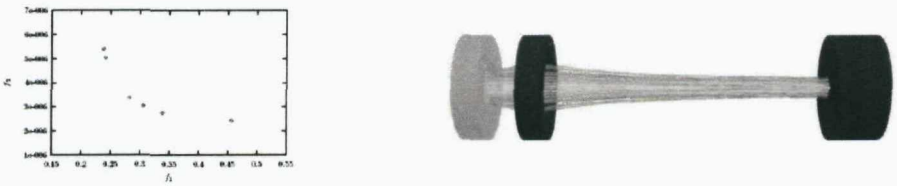
A popular method for solving a multi-objective optimization problem (MOOP) is to transform it to a single-objective optimization problem (SOOP), and then solve the SOOP using a single-objective optimization algorithm. A large number of methods exist for transforming a MOOP to a SOOP including (but again not limited to): ε-constraint (ε-C); weighting method (W); weighted metrics (WM) (including the Tchbycheff metric) method; achievement scalarizing function approach (AF); lexicographic ordering approach (LO); and the value function method (VF), descriptions of all of which may be found in [7]. This wide variety of scalarizing methods, coupled with the large range of selection crietria from single-objective optimization mentioned in the previous paragraph, leads to a plethora of (scalarizing) multi-objective algorithms being possible with kriging; a selection of these are shown in Table 1. Despite this fact, surprisingly few have been explored in the literature: two which have been explored are labelled with their reference.

One of the potential algorithms, marked X in Table 1, was tested on an electromagnetic design problem. The voltage on, and position of, the focus electrode of an electron gun was varied so as to achieve two objectives: to focus the beam of electrons on the center of the anode as much as possible, and to make the electrons hit the anode face as perpendicular as possible. Formally, denoting the voltage on the focus electrode by V Volts, and

Table 1. The family of scalarizing multi-objective optimization algorithms made possible with kriging.

Selection Criteria	Scalarizing Method						
	GC	W	ε-C	WM	AF	LO	VF
POI							
EI			[3]	[8]			
GEI							
WEI							
CH							
ME							
Hybrids				X			

its perpendicular distance from the emitting surface by d cm, the objective functions to be minimized are $f_1(V,d) = \int_{\text{anode}} J(r) r^2 \, dS$ and $f_2(V,d) = \int_{\text{anode}} \frac{(v_x^2 + v_y^2)}{(v_x^2 + v_y^2 + v_z^2)} \, dS$ with $V \in [0,1000]$ and $d \in [4,10]$, where r is the radial distance from the center of the anode surface, $J(r)$ is the current density at r, and the integrals are taken over the surface of the anode. v_x, v_y and v_z are the components of the electron velocities as they hit the surface of the anode, which lies in the xy plane. Each analysis was carried out using the Vector Fields space charge solver, SCALA. The Pareto optimal points found, along with one of the solutions, are shown in Fig.1.



(a) (b)

Fig.1 (a) Pareto optimal front for electron gun problem, and (b) the right-most Pareto-optimal solution (which has the most parallel beam, as measured by objective f_2), as found using algorithm X from Table 1.

Conclusions

Many advances have been made in recent years in kriging-assisted single-objective optimization. Using scalarizing methods, it is possible to also use each in multi-objective optimization. Despite this, relatively few of the possibilities have been explored in the literature. This paper has made it explicit the range of multi-objective algorithms possible with kriging, and demonstrated the use of one of the algorithms on an electromagnetic design problem. Other algorithms from the family of scalarizing algorithms will be explored in the full paper.

References

[1] L. Lebensztain, C.A.R. Marretto, M.C. Costa and J-L. Coulomb, Kriging: a useful tool for electromagnetic devices optimization, IEEE Transactions on Magnetics, Vol. 40, No. 2, pp. 1196-1199, 2004.

[2] D. R. Jones, A Taxonomy of Global Optimization Methods Based on Response Surfaces, Journal of Global Optimization, Vol. 21, pp. 345-383, 2001.

[3] D. R. Jones, M. Schonlau and W. J. Welch, Efficient Global Optimization of Expensive Black-Box Functions, Journal of Global Optimization, Vol. 13, pp. 455-492, 1998.

[4] M. J. Sasena, P. Y. Papalambros and P. Goovaerts, Exploration of metamodelling sampling criteria for constrained global optimization, Engineering Optimization, Vol. 34, No. 3, pp. 263-278, 2002.

[5] A. Sobester, S. J. Leary and A. J. Keane, On the design of optimization strategies based on global response surface approximation models, Journal of Global Optimization, vol. 33, pp. 31–59, 2005.

[6] J. Villemonteix, E. Vazquez, and E.Walter, An informational approach to the global optimization of expensive-to-evaluate functions, to appear in the Journal of Global Optimization, 2007.

[7] K. M. Miettinen, Nonlinear Multiobjective Optimization, Kluwer Academic Publishers, 1999.

[8] J. Knowles, ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Transactions on Evolutionary Computation, vol. 10, no. 1, pp. 50 66, 2006.

PROBABILITY OF IMPROVEMENT METHODS FOR CONSTRAINED MULTI-OBJECTIVE OPTIMIZATION

G. I. Hawe, J. K. Sykulski

School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, U.K.
e-mail: gih@soton.co.uk, jks@soton.ac.uk, fax: +44 (0)23 8059 3709

Keywords: Constrained Multi-Objective Optimization, Kriging.

Abstract

This paper shows how the simultaneous consideration of multiple Kriging models can lead to useful metrics for the selection of design vectors in constrained multi-objective optimization. The savings in computational cost with such methods make them particularly useful for optimal electromagnetic design.

1 Introduction

In single-objective optimization, a single Kriging surface [7] may be constructed modelling the behaviour (with respect to the design variables) of the objective function in question. In [3], a variety of selection criteria are proposed for utilizing statistical information from a *single* Kriging model, for the purpose of selecting a design vector (or multiple design vectors) to evaluate in the search for the minimum. Methods also exist for incorporating constraint handling into these selection criteria, e.g. [9].

In multi-objective optimization, the multiple objectives may be combined into a single objective using a scalarizing method [8], and a method from single-objective optimization used for selecting design vectors, e.g. [1, 6]. Alternatively, each objective may be modelled by its own individual Kriging surface; this allows the uncertainty in each objective to be modelled separately. The simultaneous consideration of these uncertainties then allows useful metrics for selecting design vectors to be constructed [2, 5], which do not suffer from the loss of information which inevitably occurs when using scalarizing methods. This paper proposes a method of extending the handling of non-linear constraints into such metrics.

2 Constrained Multi-Objective Optimization

The constrained multi-objective optimization problem (CMOOP) may be phrased as:

$$\begin{array}{lll} \text{Minimize} & f_i(\mathbf{x}) & i = 1, \ldots, M \\ \text{subject to} & g_i(\mathbf{x}) \le 0 & i = 1, \ldots, J \\ & h_i(\mathbf{x}) = 0 & i = 1, \ldots, K \\ \text{where} & x_i^l \le x_i \le x_i^u & i = 1, \ldots, d \end{array} \quad (1)$$

where the design vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$. If a design vector satisfies all the constraints, it is feasible; otherwise it is infeasible. For any two feasible design vectors $\mathbf{x}^a$ and $\mathbf{x}^b$, if $f_i(\mathbf{x}^a) \le f_i(\mathbf{x}^b)\ \forall\, i$, and $\exists\, j$ such that $f_j(\mathbf{x}^a) < f_j(\mathbf{x}^b)$, then $\mathbf{x}^a$ is said to dominate $\mathbf{x}^b$; otherwise $\mathbf{x}^a$ is said to be non-dominated by $\mathbf{x}^b$. If $\mathbf{x}^a$ is non-dominated by $\mathbf{x}^b$ and $\mathbf{x}^b$ is non-dominated by $\mathbf{x}^a$ then $\mathbf{x}^a$ and $\mathbf{x}^b$ are said to be equivalent. All infeasible design vectors are deemed to be equivalent, and dominated by each of the feasible design vectors. The solution to (1) is the set of feasible design vectors which are non-dominated over the entire search space, known as the Pareto-optimal set.

3 Probability of Improvement with Constraints

Suppose that after sampling the design variable space (preferably using a space-filling experimental design, such as a Hammersley Sequence [4]), a set S of N_{par} non-dominated solutions exist (each of these solutions are both feasible, and non-dominated by the solutions not in S). Then it is desirable when sampling again to select a design vector which either dominates at least one (preferably more) solution in S *whilst being feasible*, or at the very least augments the set S (i.e. is equivalent to each solution in S *and is feasible*). In either case such a selection could be said to yield an improvement, as our solution set has improved. In *unconstrained* multi-objective optimization, by constructing Kriging surfaces for each objective individually, not only may the probability of an unevaluated design vector dominating at least one solution in S be calculated [5], but the probability of it dominating a particular number of solutions in S (representing a specific *level* of improvement), may also be determined [2]. This is illustrated schematically in Figure 1 for the case of a two-objective problem with N_{par}=5 non-dominated solutions. In the presence of constraints however, it is crucial to also ensure the feasibility of the design vectors being selected.

As can be seen from Figure 1, for an improvement level n (achieving a level n improvement means that a design vector dominates n solutions in S, with a level 0 improvement meaning that it is equivalent to each solution in S), a design vector may map to $N_{par}-n+1$ regions of objective function space. Denote by $P(I^n(\mathbf{x}))$ the probability that an unknown design vector x will yield a level of improvement n (i.e. it will dominate exactly n

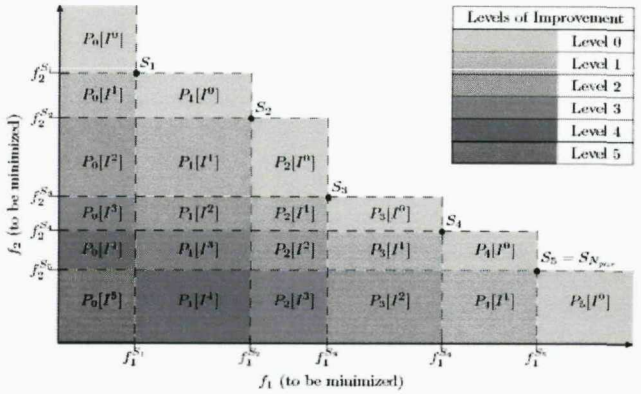


Figure 1: Levels of improvement for a two-objective problem with 5 non-dominated solutions.

existing non-dominated solutions *and* be feasible), and $P_i(I^n(\mathbf{x}))$ as the probability that design vector $\mathbf{x}$ will dominate the n non-dominated solutions S_{i+1}, S_{i+2}, ... , S_{i+n} (these sub-regions are labelled in Figure 1) *and* be feasible. Define

$$\Phi_p^i(\mathbf{x}) = \Phi\left(\frac{f_p^{S_i} - \hat{f}_p(\mathbf{x})}{s_p(\mathbf{x})}\right) \qquad (2)$$

for p=1,2, with

$$\Phi_1^0(\mathbf{x}) = 0\,,\ \Phi_1^{N_{par}+1}(\mathbf{x}) = 1 \qquad (3), (4)$$

$$\Phi_2^0(\mathbf{x}) = 1\,,\ \Phi_2^{N_{par}+1}(\mathbf{x}) = 0 \qquad (5), (6)$$

where $\hat{f}_p(\cdot)$ and $s_p(\cdot)$ are the Kriging prediction and standard error for each objective (p=1,2), and $f_p^{S_i}$ is the value of objective p for non-dominated solution S_i. Then

$$P\left(I^n(\mathbf{x})\right) = \prod_{i=1}^{J} \Phi\left(-\frac{\hat{g}_i(\mathbf{x})}{s_{g_i}(\mathbf{x})}\right) \times \prod_{j=1}^{K}\left[\Phi\left(\frac{\varepsilon_j - \hat{h}_j(\mathbf{x})}{s_{h_j}(\mathbf{x})}\right) + \Phi\left(\frac{\hat{h}_j(\mathbf{x}) - \varepsilon_j}{s_{h_j}(\mathbf{x})}\right)\right] \times \sum_{l=0}^{N_{par}-n}\left(\Phi_1^{l+1}(\mathbf{x}) - \Phi_1^{l}(\mathbf{x})\right)\left(\Phi_2^{n+l}(\mathbf{x}) - \Phi_2^{n+l+1}(\mathbf{x})\right) \qquad (7)$$

where $\hat{g}_i(\cdot)$ (i = 1,...J) and $\hat{h}_i(\cdot)$ (i=1,...,K) are the Kriging predictions for each of the individual constraint functions, s_{g_i} and s_{h_i} are their standard errors, and ε_i (i=1,...,K) are small tolerances chosen to transform each equality constraint into two inequality constraints.

In words, the design vector which maximizes the expression in (7) is that which is most likely to dominate n of the current non-dominated solutions, *and* be feasible. In total this gives N_{par}+1 different levels of improvement to maximize at each iteration; by grouping the design vectors which maximize each of these levels of improvement into clusters (using, e.g. the method proposed in [3]), and selecting a representative design vector from each cluster, a robust method (which is easily parallelized) – using the probability of improvement method – is made available for constrained multi-objective optimization. Although the description given here is for two-objective problems, the method is extensible to higher numbers of objectives.

4 Conclusions

A novel utility function, which is easily parallelized and which does not require normalization of the objective functions, has been proposed for computationally expensive constrained multi-objective optimization. It is ideally suited for applications such as electromagnetic design.

References

[1] G. I. Hawe, J. K. Sykulski. "A scalarizing one-stage algorithm for efficient multi-objective optimization", *To appear in IEEE Transactions in Magnetics*, (2008).

[2] G. I. Hawe, J. K. Sykulski. "An enhanced probability of improvement utility function for locating Pareto-optimal solutions", *Proceedings of the 16th Conference on the Computation of Electromagnetic Fields, COMPUMAG 2007, pp. 965 – 966*, (2007).

[3] D. R. Jones. "A taxonomy of global optimization methods based on response surfaces", *Journal of Global Optimization*, 21, pp. 345-383, (2001).

[4] J. R. Kalagnanam, U. M. Diwekar. "An efficient sampling technique for off-line quality control", *Technometrics*, 39 (2), pp. 131-148, (1997).

[5] A. J. Keane. "Statistical improvement criteria for use in multiobjective design optimization", *AIAA Journal*, 44 (4), pp. 879-891, (2006).

[6] J. Knowles. "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multi-objective optimization problems", *IEEE Transactions on Evolutionary Computation Journal*, 10 (1), pp. 50-66, (2006).

[7] L. Lebensztajn, C. A. R. Marretto, M. C. Costa, J-L. Coulomb. "Kriging: a useful tool for electromagnetic devices optimization", *IEEE Transactions on Magnetics*, 40 (2), pp. 1196-1199, (2004).

[8] K. M. Miettinen. *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, (1999).

[9] M. J. Sasena, P. Y. Papalambros, P. Goovaerts. "Exploration of metamodeling sampling criteria for constrained global optimization", *Engineering Optimization*, 34 (3), pp. 263-278, (2002).

Appendix F

List of Journal Papers

The following papers were published as a result of work carried out in this thesis:

1. Hawe, G. and Sykulski, J. K. (2007) Considerations of Accuracy and Uncertainty with Kriging Surrogate Models in Single-Objective Electromagnetic Design Optimization. IET Science, Measurement & Technology 1(1) pp. 37-47.

2. Hawe, G. and Sykulski, J. K. (2007) A Hybrid One-then-Two Stage Algorithm for Computationally Expensive Electromagnetic Design Optimization. COMPEL - International Journal for Computation and Mathematics in Electrical and Electronic Engineering 26(2) pp. 240-250.

3. Hawe, G. I. and Sykulski, J. K. (2008) A Scalarizing One-Stage Algorithm for Efficient Multi-Objective Optimization, to appear in IEEE Transactions on Magnetics.

4. Hawe, G. I. and Sykulski, J. K. (2008) Scalarizing Cost-Effective Multiobjective Optimization Algorithms Made Possible with Kriging, to appear in COMPEL.

Bibliography

[1] G. I. Hawe and J. K. Sykulski. The consideration of surrogate model accuracy in single-objective electromagnetic design optimization. In *Proceedings of 6th International Conference on Computational Electromagnetics*, pages 115–116, 2006.

[2] G. I. Hawe and J. K. Sykulski. Balancing exploration and exploitation using kriging surrogate models in electromagnetic design optimization. In *Proceedings of 12th Biennial IEEE Conference on Electromagnetic Field Computation*, page 226, 2006.

[3] G. I. Hawe and J. K. Sykulski. A hybrid one-then-two stage algorithm for computationally expensive electromagnetic design optimization. In *Proceedings of the 9th Workshop on Optimization and Inverse Problems in Electromagnetism*, pages 191–192, September 2006.

[4] G. I. Hawe and J. K. Sykulski. A hybrid one-then-two stage algorithm for computationally expensive electromagnetic design optimization. *COMPEL*, 26(2):240–250, 2007.

[5] G. I. Hawe and J. K. Sykulski. An enhanced probability of improvement utility function for locating pareto optimal solutions. In *Proceedings of the 16th Conference on the Computation of Electromagnetic Fields*, pages 965–966, 2007.

[6] G. I. Hawe and J. K. Sykulski. A scalarizing one-stage algorithm for efficient multi-objective optimization. In *Proceedings of the 16th Conference on the Computation of Electromagnetic Fields*, pages 967–968, 2007.

[7] G. I. Hawe and J. K. Sykulski. Scalarizing cost-effective multiobjective optimization algorithms made possible with kriging. In *Proceedings of the 16th International Symposium on Electromagnetic Fields in Mechatronics, Electrical and Electronic Engineering*, pages 232–233, 2007.

[8] G. I. Hawe and J. K. Sykulski. Considerations of accuracy and uncertainty with kriging surrogate models in single-objective electromagnetic design optimization. *IET Science, Measurement & Technology*, 1(1):37–47, 2007.

[9] G. I. Hawe and J. K. Sykulski. Scalarizing cost-effective multiobjective optimization algorithms made possible with kriging, 2008. to appear in COMPEL - International Journal for Computation and Mathematics in Electrical and Electronic Engineering.

[10] G. I. Hawe and J. K. Sykulski. Probability of improvement methods for constrained multi-objective optimization, 2008. to appear in Proceedings of Seventh International Conference on Computational Electromagnetics.

[11] G. I. Hawe and J. K. Sykulski. A scalarizing one-stage algorithm for efficient multi-objective optimization, 2008. to appear in IEEE Transactions on Magnetics.

[12] Opera Manager User Guide, 2007. Vector Fields.

[13] A. J. Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA Journal*, 44(4):879–891, 2006.

[14] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–374, 2001.

[15] M. J. Sasena, P. Papalambros, and P. Goovaerts. Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization*, 34(3):263–278, 2003.

[16] P. Neittaanmaki, M. Rudnicki, and A. Savini. *Inverse Problems and Optimal Design in Electricity and Magnetism*. Oxford University Press, 1996.

[17] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, The Santa Fe Institute, Santa Fe, NM, 1995.

[18] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[19] W. P. Baritompa, M. Dur, E. M. T. Hendrix, L. Noakes, W. J. Pullan, and G. R. Wood. Matching stochastic algorithms to objective function landscapes. *Journal of Global Optimization*, 31(4):579–598, 2005.

[20] P. Di Barba, B. Forghani, and D. A. Lowther. Discrete-valued design optimisation of a multiple-coil inductor for uniform surface heating. *COMPEL*, 24(1):271–280, 2005.

[21] J. Byun and S. Hahn. Topology optimization of electrical devices using mutual energy and sensitivity. *IEEE Transactions on Magnetics*, 35(5):3718–3720, 1999.

[22] D-H. Kim, K.S. Ship, and J. K. Sykulski. Applying continuum design sensitivity analysis combined with standard em software to shape optimization in magnetostatic problems. *IEEE Transactions on Magnetics*, 40(2):1156–1159, 2004.

[23] D-H. Kim, J. K. Sykulski, and D. A. Lowther. A novel scheme for material updating in source distribution optimization of magnetic devices using sensitivity analysis. *IEEE Transactions on Magnetics*, 41(5):1752–1755, 2005.

[24] D. N. Dyck and D. A. Lowther. Automated design of magnetic devices by optimizing material distribution. *IEEE Transactions on Magnetics*, 32(3), 1996.

[25] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[26] N. Srinivas and K. Deb. Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1994.

[27] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Technical Report 2000001, Indian Institute of Technology, Kanpur, India, 2000.

[28] J. Horn, N. Nafploitis, and D. Goldberg. A niched pareto genetic algorithm for multi-objective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87, 1994.

[29] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evoluionary Computation*, 8(2):149–172, 2000.

[30] E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 1998.

[31] E. Zitzler, M. Laumanns, and L. Thiele. SPEA-II: Improving the performance of the strength pareto evolutionary algorithm. Technical Report 103, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2001.

[32] M. Locatelli and G. R. Wood. Objective function features providing barriers to rapid global optimization. *Journal of Global Optimization*, 31:549–565, 2005.

[33] C. Khompatraporn, J. D. Pintér, and B. Zabinsky. Comparitive assesment of algorithms and software for global optimization. *Journal of Global Optimization*, 31:613–633, 2005.

[34] L. C. W. Dixon and G. P. Szego. The global optimisation problem: an introduction. In L. C. W. Dixon and G. P. Szego, editors, *Towards Global Optimisation*, volume 2, pages 1–15. North Holland, Amsterdam, 1978.

[35] B. Addis and M. Locatelli. A new class of test functions for global optimization, 2007. submitted to the Journal of Global Optimization.

[36] P. Di Barba. Nash equilibrium and pareto front for the optimal shape design in electromechanics. In *Proceedings of 6th International Conference on Computational Electromagnetics*, pages 123–124, 2006.

[37] T. Miyamoto, K. Kaneda, S. Noguchi, and H. Yamashita. A technique for selecting an optimal solution from among pareto-optima of multi-purposed electromagnetic apparatus design based on game theory. In *Proceedings of 12th Biennial IEEE Conference on Electromagnetic Field Computation*, page 120, 2006.

[38] K. M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[39] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of the Association for Computing Machinery*, 22(4):469–476, 1975.

[40] J. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980.

[41] M. A. Yukish. *Algorithms to Identify Pareto Points in Multi-Dimensional Data Sets*. PhD thesis, The Pennsylvania State University, 2004.

[42] K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[43] P. A. N. Bosman and D. Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):174–188, 2005.

[44] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: Ananalysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[45] J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[46] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

[47] C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, Lecture Notes in Computer Science, pages 584–593. Springer-Verlag, 1996.

[48] K. Deb, L. Thiele, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. Technical Report 113, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2001.

[49] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the Fa999 ACM Symposium on Applied Computing*, pages 351–357, 1999.

[50] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multi-objective optimization. In *Proc. 8th Int. Conf. Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, pages 792–802. Springer-Verlag, 2004.

[51] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[52] M. J. Reddy and D. N. Kumar. An efficient multi-objective optimization algorithm based on swarm intelligence for engineeing design. *Engineering Optimization*, 39(1):49–68, 2007.

[53] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, 1999.

[54] A. Canova, F. Freschi, and M. Repetto. Hybrid method coupling ais and zeroth order deterministic search. *COMPEL*, 24(3):784–795, 2005.

[55] S. I. Birbil and S-C. Fang. An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization*, 25:263–282, 2003.

[56] E. Ducheyne, B. De Baets, and R. De Wulf. Is fitness inheritance useful for real-world applications? In *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 31–42. Springer-Verlag, 2003.

[57] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[58] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 2006.

[59] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definitive functions. *Constructive Approximation*, 2:11–22, 1986.

[60] D. Gorissen. Towards an adaptive and flexible metamodeling toolbox. Technical report, University of Antwerp, COMS Research Group, University of Antwerp, Middelheimlaan, 2020 Antwerp, Belguim, 2006.

[61] G. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *ASME Transactions, Journal of Mechanical Design*, 2006. in press.

[62] A. A. Mullur and A. Messac. Metamodeling using extended radial basis functions: a comparitive approach. *Engineering with Computers*, 21:203–217, 2006.

[63] V. C. P. Chen, K-W. Tsui, R. R. Barton, and M. Meckesheimer. A review on design, modeling and applications of computer experiments. *IIE Transactions*, 38:273–291, 2006.

[64] G. Matherton. Principles of geostatistics. *Economic Geology*, 58:1246–1266, 1963.

[65] D. G. Krige. A statistical approach to some mine problems and allied problems at the witwatersrand. Master's thesis, University of Witwatersrand, 1951.

[66] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[67] N. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, 1993.

[68] M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, 1999.

[69] T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

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

[71] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[72] H. M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[73] M. Björkman and K. Holmström. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1:373–397, 2000.

[74] R. G. Regis and C. A. Shoemaker. Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37(1):113–135, 2007.

[75] H. P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley and Sons, 1981.

[76] M. Locatelli. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10:57–76, 1997.

[77] N.-H. Quttineh and K. Holmström. An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Presented at the Second International Workshop on Surrogate Modelling and Space Mapping for Engineering Optimization, November 2006.

[78] Emmanuel Vazquez and Julien Bect. On the convergence of the expected improvement algorithm, 2007. http://arxiv.org/abs/0712.3744v1.

[79] A. Sobester, S. J. Leary, and A. J. Keane. A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Struct Multidisc Optim*, 27:371–383, 2004.

[80] A. Sobester, S. J. Leary, and A. J. Keane. On the design of optimization strategies based on global response surface approximation models. *Journal of Global Optimization*, 33:31–59, 2005.

[81] H. J. Kushner. A new method of locating the maximum point of an arbitrary mulyipeak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106, 1964.

[82] D. R. Jones. Global optimization with response surfaces. presented at the Fifth SIAM Conference on Optimization, Victoria, Canada, 1996.

[83] K. Holmström. An adaptive radial basis algorithm (arbf) for expensive black-box global optimization. *to appear in the Journal of Global Optimization*, 2008.

[84] J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions, 2006. http://arxiv.org/abs/cs/0611143v2.

[85] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidiscplinary Optimization*, 26:369–395, 2004.

[86] R. T. Marler and J. S. Arora. Function-transformation methods for multi-objective optimization. *Engineering Optimization*, 37(6):551–570, 2004.

[87] I. Kaliszewski. *Quantitative Pareto Analysis by Cone Separation Technique.* Kluwer Academic Publishers, 1994.

[88] L. Lebensztajn et al. A multi-objective analysis of a special switched reluctance motor. *COMPEL*, 24(3):931–941, 2005.

[89] M. T. M. Emmerich, K. C. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[90] A. J. Keane and J. P. Scanlan. Design search and optimization in aerospace engineering. *Philisophical Transactions of the Royal Society A*, 365(1859):2501–2529, 2007.

[91] D. Montgomery. *Design and Analysis of Experiments.* John Wiley and Sons, 2001.

[92] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa. A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal of Computing*, 17(3):263–289, 2005.

[93] A. A. Giunta, S. F. Wojtkiewicz Jr, and M. S. Eldred. Overview of modern design of experiments methods for computational simulations. In *Proceedings of the 41st AIAA Aerospace Sciences Meeting and Exhibit*, January 2003. AIAA-2003-0649.

[94] T. W. Simpson, D. K. J. Lin, and W. Chen. Sampling strategies for computer experiments: Design and analysis. *International Journal of Reliability and Applications*, 2:209–240, 2001.

[95] J. M. Hammersley. Monte carlo methods for solving multivariate problems. *Annals of the New York Academy of Science*, 86:844–874, 1960.

[96] J. R. Kalagnanam and U. M. Diwekar. An efficient sampling technique for off-line quality control. *Technometrics*, 39(3):308–318, 1997.

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

[98] A. I. Forrester. *Effecient Global Aerodynamic Optimisation Using Expensive Computational Fluid Dynamics Simulations.* PhD thesis, University of Southampton, Southampton, U.K., 2005.

[99] A. J. Keane and P. B. Nair. *Computational Approaches for Aerospace Design.* Wiley, 2005.

[100] A. I. J. Forrester, A. Sóbester, and A. J. Keane. Optimization with missing data. *Proceedings of the Royal Society A*, 462:935–945, 2006.

[101] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Waterloo, Canada, 1997.

[102] S. Jeong, K. Yamamoto, and S. Obayashi. Kriging-based probabilistic method for constrained multi-objective optimization problem. In *AIAA 1st Intelligent Systems Technical Conference*, Chicago, Illinois, 2004. AIAA-2004-6437.

[103] J-E. Käck. Constrained global optimization with radial basis functions. Technical Report MdH-IMa-2004, Mälardalen University, Västeras, Sweden, 2004.

[104] *Surrogate-Model-Based Method For Constrained Optimization*, Long Beach, CA, 2000. AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.

[105] M. Sasena, P. Papalambros, and P. Goovaerts. The use of surrogate modeling algorithms to exploit disparities in function computation time within simulation-based optimization. In *The Fourth World Congress of Structural and Multidisciplinary Optimization*, 2001.

[106] M. J. Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[107] E. S. Siah, M. Sasena, J. L. Volakis, P. Y. Papalambros, and R. W. Wiese. Fast parameter optimization of large-scale electromagnetic objects using DIRECT with kriging metamodeling. *IEEE Transactions on Microwave Theory and Techniques*, 52(1):276–285, 2004.

[108] T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, 17:129–150, 2001.

[109] G. Hawe. Multi-objective optimization in electromagnetic design. Technical report, University of Southampton, Southampton, U.K., 2005.

[110] Opera-3d User Guide, 2007. Vector Fields.

[111] Opera-2d User Guide, 2007. Vector Fields.

[112] L. Lebensztajn, C. A. R. Marretto, M. C. Costa, and J-L. Coulomb. Kriging: a useful tool for electromagnetic devices optimization. *IEEE Transactions on Magnetics*, 40(2):1196–1199, 2004.

[113] T. Goel and K. Deb. Hybrid methods for multi-objective evolutionary algorithms. Technical Report 2001004, Indian Institute of Technology, Kanpur, India, 2001.

[114] M. Farina. *Cost-effective Evolutionary Strategies for Pareto-optimal Front Approximation in Multiobjective Shape Design Optimization of Electromagnetic Devices.* PhD thesis, University of Pavia, 2001.