

Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations

by

Michael James Sasena

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2002

Doctoral Committee:

Professor Panos Y. Papalambros, Co-Chair
Assistant Professor Pierre Goovaerts, Co-Chair
Associate Professor Jack Hu
Assistant Professor Kazuhiro Saitou
Trudy Weber, General Motors Corporation

© Michael J. Sasena 2002
All Rights Reserved

For Jolene,
who has been there to share in the joys and agonies
that have defined the journey.

ACKNOWLEDGEMENTS

I owe a debt of gratitude to my co-chairs, Panos Papalambros and Pierre Goovaerts, for their valuable guidance and efforts to keep me focused. Working with you both these past few years has been a wonderful learning experience. I would also like to thank the rest of my committee for their insightful questions which have improved the quality of the thesis. Special thanks to Don Jones of General Motors for his help with both the EGO and DIRECT algorithms and to Matt Reed of UMTRI for his help with the reach effort study. Your enthusiasm for research is contagious.

My research has been partially supported by the Automotive Research Center at the University of Michigan, a US Army Center of Excellence in Modeling and Simulation of Ground Vehicles and by the General Motors Collaborative Research Laboratory at the University of Michigan. This support is gratefully acknowledged.

Of course, all this couldn't have happened without my fellow ODE members. Nestor, Sig, Julie, Shinji, Zhifang, Ryan, Nnaemeka, Chris, George, Hyung Min, Matt, John, J, Panayiotis, Hosam, Zhijun, Ilkin, Lara, Michael, Olena, Elena, Ruchi, Alex, and Adam – you have all been a blast to work with and will be missed! Thanks also to countless friends who have shown such wonderful support and encouragement.

Finally, a special thank you to my family and my fiancée. You have helped me tremendously by reminding me of the other important things in life, most importantly my faith. Your support has enabled me to achieve so many wonderful things that I couldn't have done on my own.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	viii
LIST OF TABLES	xiii
LIST OF APPENDICES	xiv
CHAPTER	
1. Introduction	1
2. Literature Review	10
2.1 Review of Surrogate Modeling Techniques	10
2.1.1 Least squares	11
2.1.2 Interpolating and smoothing splines	12
2.1.3 Fuzzy logic and neural networks	13
2.1.4 Radial basis functions and wavelets	14
2.1.5 Wiener process	15
2.1.6 Kriging	15
2.2 Current Research in Optimization through Approximation . .	16
2.2.1 Response surface methodology	16
2.2.2 Sequential quadratic programming (SQP)	17
2.2.3 Trust region framework for using surrogates	18
2.2.4 Metamodels in robust design	20
2.2.5 Variable complexity response surface modeling	21
2.2.6 Approximation methods in decomposition	22
2.2.7 Exploiting functional cost disparities	24
2.3 Origins and Development of EGO	24
2.3.1 Bayesian analysis	25
2.3.2 Extending the P-algorithm to multiple dimensions .	26
2.3.3 Choice of statistical models	27

2.3.4	Choice of infill sampling criterion	28
2.3.5	Convergence properties and stopping rules	30
2.3.6	Miscellaneous advancements	33
2.4	Chapter Summary	33
3.	Understanding Kriging	35
3.1	History and Terminology	35
3.2	General Kriging Approach	36
3.3	Derivation of the Prediction Formula	39
3.4	Parameter Estimation	43
3.5	Effect of the SCF and DOE on Model Accuracy	45
3.6	How Does DACE Differ from Geostatistics?	46
3.6.1	Model fitting	46
3.6.2	Anisotropy	48
3.6.3	Global search window	49
3.7	Improved DACE Modeling	50
3.7.1	MLE fitting revisited	50
3.7.2	Non-interpolating kriging models	54
3.8	Chapter Summary	59
4.	Infill Sampling Criteria	60
4.1	A Taxonomy of Criteria	60
4.1.1	Kushner's criterion	61
4.1.2	Expected improvement	63
4.1.3	Generalized expected improvement	64
4.1.4	Cool criterion	65
4.1.5	Lower confidence bounding function	66
4.1.6	WB ₁ : Locating the threshold-bounded extreme	67
4.1.7	WB ₂ : locating the regional extreme	68
4.1.8	WB ₃ : minimizing surprises	69
4.1.9	Maximum variance	70
4.1.10	Switching criterion	70
4.2	Analytical Examples	71
4.2.1	Example 1: Mystery function	71
4.2.2	Example 2: Branin function	72
4.2.3	Example 3: Six hump camelback function	73
4.3	Methodology and Comparison Metrics	73
4.4	Results	75
4.4.1	Solution efficiency	81
4.4.2	Solution accuracy	81
4.4.3	Global searching properties	83
4.4.4	Periodic local search	84

4.5	Discussion	89
4.6	Chapter Summary	90
5.	Constrained Bayesian Analysis	91
5.1	Methods for Constraint Handling	91
5.1.1	Probability and penalty methods	92
5.1.2	Expected violation method	95
5.1.3	Constrained ISC method	98
5.1.4	Justification of choosing constrained ISC method	98
5.2	Quantifying Constraint Satisfaction	100
5.3	Disconnected Feasible Regions	109
5.3.1	Locating an initial feasible point	109
5.3.2	Locating subsequent feasible points	112
5.3.3	Demonstrations	114
5.4	Equality Constraints	119
5.5	Constraint Activity	121
5.6	Chapter Summary	123
6.	Exploiting Disparities in Function Computation Time	125
6.1	Motivation	125
6.2	Filter Method	127
6.3	Analytical Examples	128
6.3.1	Example 1: Constrained mystery function	129
6.3.2	Example 2: Reverse constrained mystery function	130
6.3.3	Example 3: Three-hump camelback function	130
6.3.4	Example 4: Goldstein-Price function	131
6.3.5	Example 5: Test function #2	132
6.4	Results	133
6.5	Discussion	134
6.6	Chapter Summary	136
7.	Bells and Whistles	137
7.1	Locating the Infill Sample	137
7.2	Model Fitting Frequency	139
7.3	Intelligent Sampling Strategies	141
7.4	Termination Criteria	143
7.5	Chapter Summary	145
8.	Vehicle Product Platform Design Study	146
8.1	Problem Description	146

8.2	Results	150
8.3	Discussion	151
8.4	Chapter Summary	153
9.	Reach Effort Study	154
9.1	Problem Description	155
9.2	Adaptive Sampling	159
9.2.1	What is adaptive sampling?	159
9.2.2	How does it behave?	162
9.2.3	Why use it?	166
9.3	Initial Assessment	172
9.4	Subject Testing	175
9.4.1	Results	178
9.4.2	Discussion	182
9.5	Chapter Summary	183
10.	Conclusions and Future Work	184
	APPENDICES	190
	BIBLIOGRAPHY	212

LIST OF FIGURES

Figure

1.1	Example of a simulation-based design problem	3
1.2	Flowchart of the EGO algorithm	4
1.3	Demonstration of EGO	6
2.1	Neural network with one hidden layer of 3 nodes	14
3.1	Illustration that errors in regression function depend on x	37
3.2	Illustration of the impact of the global trend	45
3.3	Example of variogram fitting	48
3.4	Test function #1	51
3.5	Example of MLE fitting	52
3.6	Likelihood function for data sets A (left) and B (right).	53
3.7	Example of MLE fitting	54
3.8	Likelihood function for data sets C (left) and D (right).	55
3.9	Noisy test function (data points are shown as circles)	58
3.10	Kriging predictions for interpolating and smoothed models	58
3.11	Kriging variance for interpolating and smoothed models	58
4.1	Example of ISC plot for comparisons	62
4.2	Kushner's criterion for two values of ϵ	63

4.3	Expected improvement criterion for different g values	65
4.4	Lower confidence bounding criterion for two values of b	67
4.5	Sampling criterion for WB_1 and WB_2	68
4.6	Sampling criterion for WB_3 and Maxvar	70
4.7	Mesh and contour plots of Example 1: mystery function	72
4.8	Mesh and contour plots of Example 2: Branin function	72
4.9	Mesh and contour plots of Example 3: six hump camelback function	73
4.10	Median value of comparison metrics for Example 1	78
4.11	Median value of comparison metrics for Example 2	79
4.12	Median value of comparison metrics for Example 3	80
4.13	Comparison of the relative difficulty in locating the next iterate . .	84
4.14	Comparison of metrics for standard approach (black) and periodic local search (white) for Example 1	86
4.15	Comparison of metrics for standard approach (black) and periodic local search (white) for Example 2	87
4.16	Comparison of metrics for standard approach (black) and periodic local search (white) for Example 3	88
5.1	Contour plot of EI criterion for Equation (5.1)	93
5.2	Contour plot of the adjusted EI criterion for Equation (5.1)	94
5.3	Contour plot of the adjusted EI criterion for Equation (5.1) in the presence of additional sample points	95
5.4	Differences in the feasibility of sampling regions for the probability and penalty methods	96
5.5	Plot of expected violation for Equation (5.3)	97
5.6	Contour plot of test function #2	102

5.7	Plots of the EI criterion for Equation (5.4)	103
5.8	Plots of the WB_2 criterion for Equation (5.4)	103
5.9	Comparison of constraint satisfaction metrics	104
5.10	Differences between the quantification of constraint satisfaction . . .	107
5.11	Differences between the quantification of constraint satisfaction (close-up)	108
5.12	Contour plot of the newBranin example	111
5.13	Plot of ISC_{13} for the newBranin example	111
5.14	Plot of ISC_{14} for the newBranin example	113
5.15	Plot of ISC_{14} for the newBranin example (second iteration)	113
5.16	Plot of the Gomez #3 test function	115
5.17	Plot of the Gomez #3 constraint function	116
5.18	Results of ISC_{14} on the Gomez #3 example	117
5.19	Progression of the ISC_{14} criterion on the Gomez #3 example	118
5.20	Splitting an equality constraint into two inequality constraints . . .	120
5.21	Illustration of active and tight constraints	122
6.1	Possible Problem Statement Types	127
6.2	Example 1 with feasible region shaded, optimum shown as filled circle	129
6.3	Example 2 with feasible region shaded, optimum shown as filled circle	130
6.4	Example 3 with feasible region shaded, optimum shown as filled circle	131
6.5	Example 4 with feasible region shaded, optimum shown as filled circle	131
6.6	Example 5 with feasible region shaded, optimum shown as filled circle	132

7.1	Breakdown of the computational costs per iteration	140
8.1	Value curves for the two objectives	147
8.2	Null platform and Pareto set for vehicle study	151
9.1	Example of a test subject reaching for the target	155
9.2	Reach effort study testing equipment	156
9.3	Automotive interior layout design	157
9.4	Example of ray-based experimental design	158
9.5	Button placement constraints around test subject	159
9.6	Contour plots of the reach effort for two virtual subjects	163
9.7	Progress of adaptive sampling for variance ISC	164
9.8	Error in the predicted reach effort	164
9.9	Adaptive sampling results using variance ISC	165
9.10	Adaptive sampling results using 7-8 zone ISC	166
9.11	Contours of the 7-8 zone of virtual subject A	170
9.12	Contours of the 7-8 zone of virtual subject B	171
9.13	Confidence intervals for the 7 and 8 contours of the reach effort . . .	171
9.14	Resulting sample for first subject	172
9.15	Actual reach effort values assigned during initial testing	174
9.16	Initial set of 11 reaches	176
9.17	DOE for ISC_1 for Subjects A (left) and B (right)	179
9.18	Reach effort interpolating model for Subjects A (left) and B (right) based on DOE from ISC_1	179

9.19	Reach effort smoothed model for Subjects A (left) and B (right) based on DOE from ISC_1	180
9.20	DOE from ISC_2 for Subjects A (left) and B (right)	180
9.21	Comparison of predicted (x-axis) to actual (y-axis) reach effort using ISC_2 for subjects A (left) and B (right)	181
9.22	DOE from ISC_3 for Subjects A (left) and B (right)	181
9.23	Comparison of predicted (x-axis) to actual (y-axis) reach effort using ISC_3 for subjects A (left) and B (right)	181
A.1	Flowchart of the superEGO algorithm	192
A.2	Flow of information between superEGO and its subroutines	193

LIST OF TABLES

Table

4.1	Cooling schedule for the Cool criterion	66
4.2	List of criteria compared	76
4.3	Median value of performance metrics for Example 1	77
4.4	Median value of performance metrics for Example 2	77
4.5	Median value of performance metrics for Example 3	78
5.1	List of metrics for quantifying constraint satisfaction	100
5.2	Comparison of constraint satisfaction metrics	104
6.1	Problem types and associated solution strategies	128
6.2	List of analytical examples	129
6.3	Summary of comparisons to EGO (in percentage) for each category	134
6.4	Relative improvement (in percentage) over EGO in iterations required	135
6.5	Relative improvement (in percentage) over EGO in time required . .	135
8.1	Percent improvement over the original EGO	151
9.1	Comparison of RMS errors resulting from various sampling strategies	168

LIST OF APPENDICES

Appendix

A.	Software Manual	191
A.1	Background	191
A.2	Framework and Important Files	192
A.3	The Subroutines	195
A.3.1	myfun_exp and myfun_cheap	195
A.3.2	fitkrige	196
A.3.3	predictkrige	198
A.3.4	UMDIRECT	199
A.3.5	fmincon	201
A.4	Syntax and Options for SuperEGO	202
A.5	Examples of How to Execute SuperEGO	208
A.5.1	Example 1	208
A.5.2	Example 2	209
A.5.3	Example 3	210

CHAPTER 1

Introduction

What is design? One definition posited by Papalambros is that design “is a decision-making process” [66]. This dissertation focuses on the kinds of decisions facing the designer who must assign parameter values describing an artifact in order to best achieve some desired performance. For example, the designer could be deciding what thickness to assign a structural member that should be as light as possible while supporting the given loads.

But how does one go about making such decisions? How could one make this process more efficient? These are questions that have pushed engineers, scientists, and mathematicians to develop tools to aid designers in decision making. Many such tools belong to a field known as design optimization, whereby numerical algorithms are used to solve problems of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to: } & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} . \end{aligned} \tag{1.1}$$

The designer is to choose values for the vector of design variables, \mathbf{x} , that minimize the scalar objective function, $f(\mathbf{x})$, while satisfying the inequality constraints,

$\mathbf{g}(\mathbf{x})$, and the equality constraints, $\mathbf{h}(\mathbf{x})$. When the functions in Equation (1.1) are nonlinear, problems of this form are referred to as *nonlinear programming*.

The general nature of nonlinear programming has led to specialized fields within optimization. One field has focused on solving structural optimization problems which have a characteristically large number of design variables and constraints. Another field, Multidisciplinary Design Optimization (MDO), has focused on how to solve problems involving a variety of engineering fields effectively by solving the large problem as a collection of smaller subproblems. The field of interest in this dissertation is simulation-based optimization.

Simulation-based optimization addresses problems where the objective and/or constraint functions are not expressed with closed-form analytical equations, but with so-called “black box” computer simulations. Typically the functions (a) are noisy or discontinuous (i.e., non-smooth) and/or (b) require a long time to compute. Each of these features causes specific difficulties that must be addressed in simulation-based optimization.

Figure 1.1 shows the fuel economy of an automobile as a function of the final drive ratio. A computer simulation was used to compute the response, and each function evaluation required approximately 60 seconds on a Pentium III 500 MHz processor. Note that the response is both noisy and has several discontinuous jumps. Numerically finding the final drive ratio that maximizes the fuel economy is a simulation-based design optimization problem that poses several challenges.

Some traditional design optimization techniques (e.g., Newton’s method) use gradient information from finite differencing to guide a sequential strategy towards the maximum. The noisy nature of the simulation shown in Figure 1.1 may render finite differencing ineffective at estimating the gradient, thereby preventing the algorithm

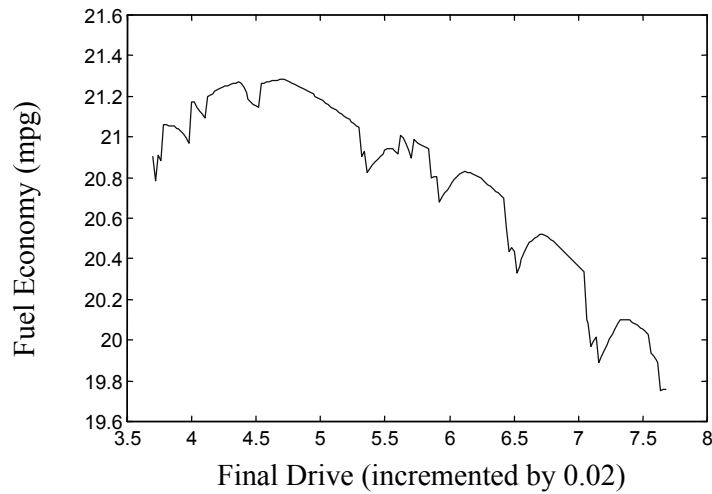


Figure 1.1: Example of a simulation-based design problem

from successfully finding the maximum. The discontinuities may also lead the algorithm to a local peak (i.e., a local optimum), but prevent it from finding the best of all peaks (i.e., the global optimum). One could start the search from several initial points to increase the chances of finding the global optimum, but the expense of the simulation may prevent such a strategy from being practical.

A common approach to deal with these problems is the use of approximations. Because approximations are models of a simulation which is itself a model of reality, they are often called *metamodels*. The terms approximations, surrogate models and metamodels will be used synonymously throughout this dissertation. The goal of using a surrogate model is to provide a smooth functional relationship of acceptable fidelity to the true function with the added benefit of quick computational speed. The approximation could be used in conjunction with a gradient-based algorithm or in entirely different ways. The details of how to build and exploit approximations effectively keep simulation-based optimization a thriving research area.

Some optimization algorithms utilize statistical models of the functions to define

an *infill sampling criterion* (ISC). The criterion determines which design points to sample next (the so-called infill samples). The infill samples are then evaluated on the true functions and the models updated. This is a completely different method than algorithms that rely on a search path because the sampling criterion could place the next iterate anywhere at all in the design space.

A wide variety of optimization algorithms search the design space in this global manner via statistical models. Throughout this dissertation, we will refer to this class of algorithms as *Bayesian analysis algorithms* in the spirit of key articles such as Žilinskas’ “One-step Bayesian Method of the Search for Extremum of an One-Dimensional Function” [102] and Mockus, Tiesis and Žilinskas’ “The Application of Bayesian Methods for Seeking the Extremum” [60]. Simply put, Bayesian analysis is the use of statistical models to predict future outcomes.

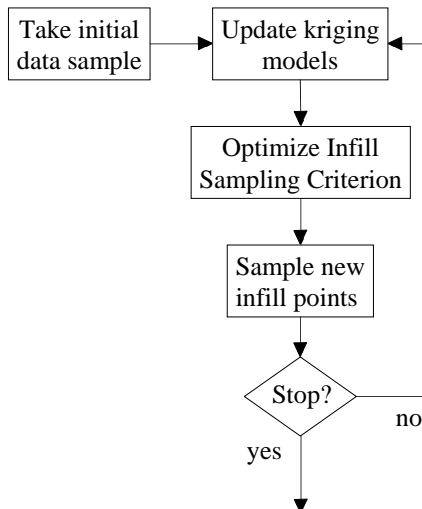


Figure 1.2: Flowchart of the EGO algorithm

One particular Bayesian analysis optimization algorithm is known as Efficient Global Optimization (EGO), developed by Schonlau, Welch and Jones [83]. It employs kriging models as the approximation method, which also provide a very useful

measure of local uncertainty. Because of the generality of the EGO framework and the appeal of the kriging models, EGO was chosen as the launching point for this dissertation. The basic outline of EGO is summarized below (see also Figure 1.2).

1. Use a space-filling design of experiments to obtain an initial sample of the true functions.
2. Fit a kriging model to each function.
3. Numerically maximize a sampling criterion known as the *expected improvement* function to determine where to sample next.
4. Sample the point(s) of interest on the true functions and update the kriging models.
5. Stop if the expected improvement function has become sufficiently small, otherwise return to 2.

To demonstrate EGO's search strategy for traditional optimization problems, a one dimensional multimodal example is shown in Figure 1.3. The w-shaped dashed line is the true objective function we wish to minimize, while the solid line is the kriging approximation conditional to the sample points shown as circles. The plot at the bottom is the sampling criterion, normalized to facilitate comparisons between iterations.

The infill sampling criterion, known as the expected improvement function, determines where EGO will evaluate the functions. It tends to choose the design points most likely to improve the accuracy of the model and/or have a better function value than the current best point. The progression of the example shown in Figure 1.3 demonstrates this behavior. After the initial sample of four points is taken, the

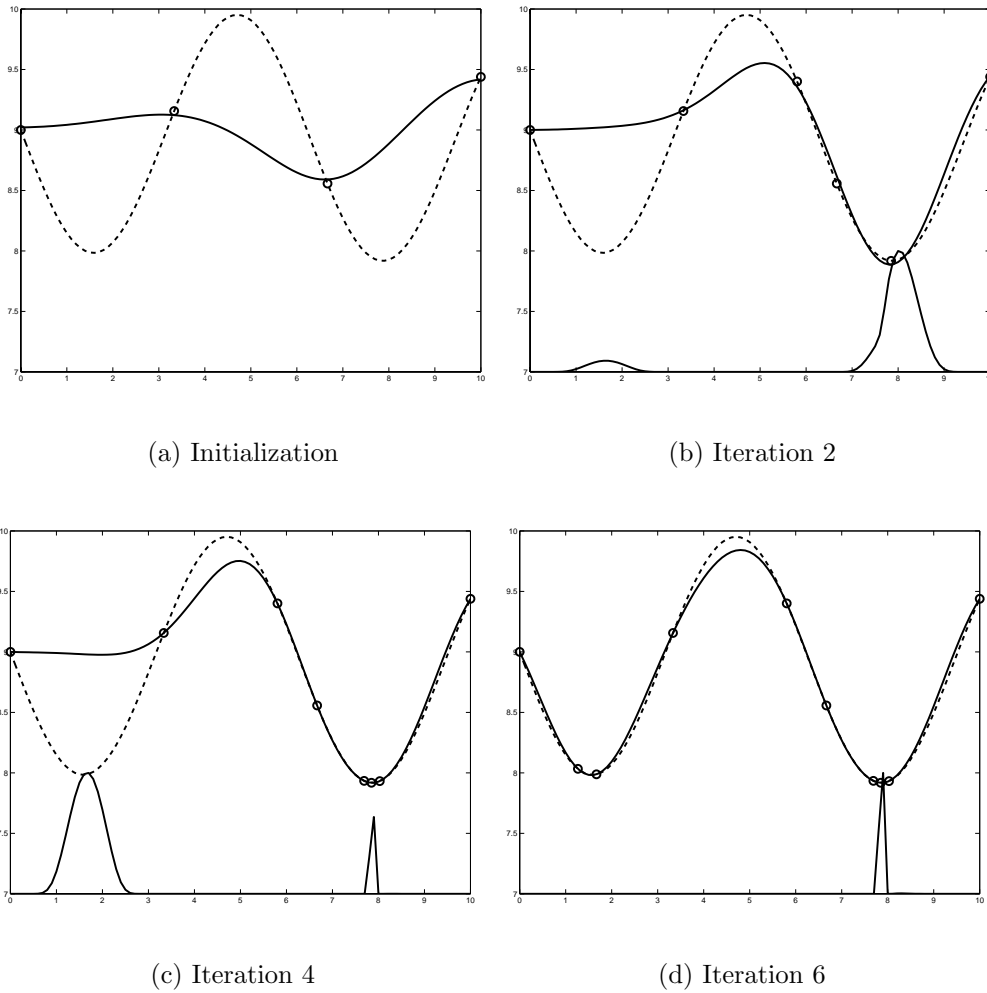


Figure 1.3: Demonstration of EGO. Dashed line is the true function, solid line is the approximation, circles are the sample points. The plot at the bottom is the sampling criterion.

resulting kriging model is a poor fit compared to the true function. However, the expected improvement function leads the algorithm to sample points where the uncertainty in the model is highest. After two iterations, the model has improved in the region of the local optimum on the right, and the expected improvement function leads EGO to sample another few points in that region where there is a high probability that a better point can be found. By the fourth iteration, the region on the right has been explored, but the uncertainty in the model on the left portion

of the model drives superEGO to sample points in that region. After six iterations, both local optima have been discovered and the true solution has been found quite accurately.

From this example, one can see that a Bayesian analysis algorithm such as EGO does not follow a search path. It selects points from anywhere in the design space depending on where the sampling criterion is highest. The optimum is found as the model accuracy is eventually improved in regions with a high probability of yielding good designs.

Because the process of fitting the kriging models and locating the maximum of the infill sampling criterion are optimization problems themselves, the overhead associated with EGO can be significant. Other methods such as genetic algorithms or gradient-based algorithms on the other hand require very little computational effort in determining where to evaluate the functions next. However, they require a large number of function evaluations to converge on a good solution. The benefit of the overhead of Bayesian analysis algorithms is that each iteration uses as much information as possible in determining where to evaluate the functions next, enabling them to locate good solutions with fewer iterations. This makes the Bayesian analysis algorithms best suited to situations where the functions are expensive, and the designer cannot afford to perform a large number of function evaluations.

The goal of this thesis is to advance the efficiency and flexibility of Bayesian analysis optimization algorithms via an adaptation of the EGO algorithm. Several areas have been identified where EGO could be expanded and/or improved.

1. *Improving the kriging modeling in Step 2.* Kriging is a data interpolation scheme that has its roots in the field of geostatistics. In the 1980's a group of statisticians began adapting the technique for cases where the data came from

deterministic computer simulations. This form of data collection and approximation, known as Design and Analysis of Computer Experiments (DACE), is used in EGO and by most other mathematicians and engineers working in the field of optimization. The DACE procedure for determining the kriging model parameters may occasionally lead to poorly fit models. We propose a simple strategy for detecting a specific type of modeling error and a method for correcting the situation. We also demonstrate how techniques for incorporating measurement error can improve the capabilities and robustness of the models.

2. *Using alternative ISC in Step 3.* EGO uses a criterion known as the expected improvement criterion to select the next iterate. Other Bayesian analysis methods use different criteria. This thesis compares a wide variety of sampling criteria to examine their effectiveness at solving analytical and simulation-based problems. In addition, the ability to use alternative ISC allows our approach a new degree of flexibility that dramatically broadens its application domain.
3. *Improving the constraint handling.* The vast majority of Bayesian analysis was done on unconstrained problems. Some authors used techniques to transform the constrained problem into an unconstrained one via penalty methods or similar approaches. This dissertation incorporates the inequality constraints directly into the ISC subproblem of Step 3. The advantages and possibilities are discussed.
4. *Taking into consideration the computational time of each function.* Another contribution of this thesis is to take into account the computational time of each function. In EGO, each function is considered expensive, and therefore a kriging model is made for all functions in the optimization model. The ap-

proach taken here is to exploit situations in which some of the problem's functions are *not* expensive. By using information from these inexpensive functions directly in the ISC subproblem, significant savings can be made in the number of function evaluations and in the clock time required to reach good solutions.

These improvements collectively allow Bayesian analysis to solve a wider variety of optimization problems and do so in less time. To distinguish the work at hand from that done in the original EGO algorithm, the version enhanced by the proposed changes will be referred to as *superEGO* to signify that it can solve a *superset* of the problems EGO was originally designed to solve.

The remainder of this thesis is organized as follows. Chapter 2 provides a review of the several approximation techniques and their current uses in simulation-based optimization as well as of the development of Bayesian analysis. In Chapter 3, an explanation of the kriging technique is accompanied by a comparison of DACE and kriging. Chapter 4 explores alternative infill sampling criteria and provides comparative results. Chapter 5 examines a variety of approaches for extending Bayesian analysis to constrained optimization and proposes a new method for locating sample points in problems with disconnected feasible regions. Chapter 6 demonstrates the benefits of exploiting discrepancies in functional computation time. A variety of issues related to the implementation details of the superEGO algorithm are discussed in Chapter 7. In Chapter 8, a simulation-based study is presented for an automotive product platform problem. In Chapter 9, an application involving an adaptive testing procedure for an ergonomics study is presented to demonstrate the benefits of the alternative ISC and of incorporating inexpensive information. The final conclusions and considerations for future work are given in Chapter 10.

CHAPTER 2

Literature Review

As described in the Introduction, one can use approximations to enhance design optimization in a number of ways. One can categorize their uses by the type of approximation and the way in which the approximations are integrated into the optimization algorithm. This chapter begins by briefly describing several commonly used approximation techniques. The second section reviews many of the relevant areas of optimization to which approximations have been successfully applied. The third section focuses specifically on the algorithm under consideration in this work. The origins and development of the algorithm are presented so that the contributions of this thesis are put into perspective.

2.1 Review of Surrogate Modeling Techniques

In this section, a brief overview of many different surrogate modeling techniques is given. The purpose is not to explain each in detail. Rather, the aim is to illustrate the wide variety of approximations available. It should be made clear that the most important technique for this dissertation is kriging (Section 2.1.6). While kriging is presented in this overview, the details of the technique are left for the next chapter.

2.1.1 Least squares

The simplest and most familiar metamodels are polynomials. For a given data set $(x_i, y_i), i = 1$ to n , one can determine the regression coefficients β_i in, say, a quadratic model of the form

$$\hat{y}(x) = \beta_0 + \beta_1 x + \beta_2 x^2, \quad (2.1)$$

where x is the input variable, $\hat{y}(x)$ is the predicted value at that input value. This is usually done by minimizing the mean of the sum of squared errors (MSSE) of the predicted output values at the x_i

$$\text{MSSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}(x_i))^2. \quad (2.2)$$

For the set of sampled data, Equation (2.1) is written as

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad (2.3)$$

or in matrix notation:

$$\mathbf{X}\beta = \mathbf{y}. \quad (2.4)$$

Solving for the vector β gives the least squares estimate

$$\beta = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y}. \quad (2.5)$$

With the optimal values of the coefficients chosen, a simple polynomial equation relating the inputs and outputs is built. This type of metamodel can be performed for any degree polynomial equation by simply changing the number of regression

coefficients included in the model. The polynomial can also be readily extended to multivariate expressions.

Statistical techniques known as analysis of variance (ANOVA) can then be carried out with this new functional relationship to determine the significance of any particular term in the polynomial. The model can be revised by changing the form of the regression equation, adding or deleting terms in a heuristic fashion, until a satisfactory representation is achieved. Increasing the number of terms or data points increases the size of the \mathbf{X} matrix to be inverted; but in general, the fitting time for least squares models is negligible.

Oftentimes, the behavior of the data cannot be captured by a single equation over the range of the data. For example, if the data tend to have highly nonlinear behavior, a quadratic, or even cubic polynomial fit would have difficulty following the data. One possible solution is to use as high an order polynomial as possible. If there are, say, ten data points that have been sampled, then a ninth order polynomial consisting of ten regression coefficients can be determined by least squares. This solution, however, is not a very good one. High order polynomials tend to oscillate wildly between data points at the extremes of the range [28]. For this reason, they are not considered appropriate surrogate models. Another solution is to include terms other than polynomials in the regression equation.

2.1.2 Interpolating and smoothing splines

Splines, where the polynomials are defined in a piecewise manner rather than as a single expression for the entire data set, provide an alternative solution to the problem of fitting highly nonlinear data. Simply put, several low order polynomials are fit to the data, each over a separate range defined by the knots or break points. Then,

boundary conditions are placed on the polynomials to ensure that the pieces match up with a prescribed order of continuity. Most often, the cubic polynomial pieces with C^2 continuity are used. By enforcing C^2 continuity, the pieces have identical value, slope, and curvature at the knot locations. In the case of interpolating splines, the knots are taken at the data points so that the spline passes exactly through each data point. Therefore, while interpolating splines may be more true to the sampled data than least squares polynomials (i.e., a single polynomial), they can exhibit more waviness in between the data points.

There has been considerable work done in the past thirty years on another regression model known as the smoothing spline. Most notable are the defining works by Wahba and Craven [22], [98]. By adjusting a weight factor known as the smoothing parameter, a compromise can be reached between the goodness of fit seen in interpolating splines and the smoothness of the approximating functions seen in least squares models.

2.1.3 Fuzzy logic and neural networks

Even though splines do not require the heuristic, fit and analyze methods that least squares models use, they still make the assumption that the data can be locally fit by a cubic polynomial. There are many other methods that avoid these assumptions by letting the data be fit in a more free form. Two popular methods in use are known as fuzzy logic and neural networks. In the former, variables are classified as belonging to sets that define an input as say, high, medium or low. Rules are then defined to describe how the dependent variable responds to the various input sets. By describing how closely an input belongs to one set or another, an approximate response can be defined [76].

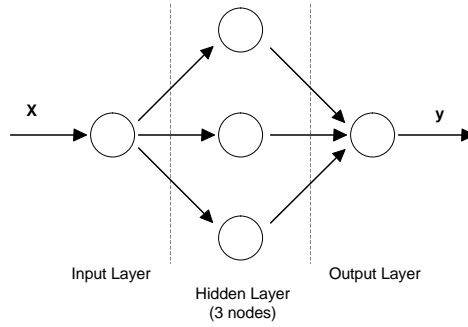


Figure 2.1: Neural network with one hidden layer of 3 nodes

For neural networks, only the set of input and corresponding output values are required. As the name suggests, the technique is analogous to the way the human brain learns. The input values are passed to a layer of nodes which alter the data using transforms such as the sigmoid or tansig functions (the “hidden layer” in Figure 2.1). The outputs of the hidden layer nodes are then passed to the output layer where they are recombined and scaled. Nonlinear optimization techniques are used to “train” the model parameters of the nodes so that the network can predict the output or response for a given input accurately [88]. Having multiple nodes and layers allows for highly nonlinear models. Because no prior knowledge of the behavior of the response is necessary, these methods have gained in popularity throughout various fields.

2.1.4 Radial basis functions and wavelets

Two other metamodels that have found wide use in image filtering and reconstruction are radial basis functions and wavelets. The radial basis function metamodel was originally developed by Hardy in 1971 as an interpolating scheme. Fifteen years later, the work of Dyn made radial basis functions more practical by enabling them to smooth data [26]. As the name suggests, the form of these metamodels is a basis

function dependent on the Euclidean distance between the sampled data point and the point to be predicted. Radial basis functions cover a wide range of metamodels, and one particular choice of basis function leads to the well-known thin plate splines. Wavelet metamodels are also relatively recent in their development [89]. They are similar to Fourier transforms, but wavelet transforms have the ability to maintain space or time information. They are also much faster and require fewer coefficients to represent a regular signal [39].

2.1.5 Wiener process

The Wiener process model is an interpolating function that was used as the basis for the original Bayesian analysis optimization algorithms described below in Section 2.3. It is a one dimensional function, but techniques have been used to successfully adapt it to multidimensional space. The Wiener process is a type of *Brownian motion* or *Markov process*, that is, the conditional expected value and variance are functions only of the nearest neighboring sample points. More specifically, the Wiener process is a piecewise linear interpolator between the data points with a quadratic variance model that returns to zero at each of the sample points. While this model is quite simplistic, it was widely used in early approximation-based optimization because of its ease of use [69]. More information on the Wiener process can be found in [50] and [53].

2.1.6 Kriging

Another type of metamodel that is currently gaining popularity is kriging. In all of the above metamodels, a fundamental assumption is that $y(x) = \hat{y}(x) + \epsilon$, where the residuals ϵ are independent, identically distributed normal random variables, $\epsilon \sim N(0, \sigma^2)$. The main idea behind kriging is that the errors in the predicted values,

ϵ_i , are *not* independent. Rather, they take the view that the errors are a systematic function of x . The kriging metamodel, $\hat{y}(x) = f(x) + Z(x)$, is comprised of two parts: a polynomial $f(x)$ and a functional departure from that polynomial $Z(x)$. While the idea behind kriging is simply put here, the details are left for the next chapter.

2.2 Current Research in Optimization through Approximation

At this point, we are ready to examine the current uses of approximation in the context of optimization. This section reviews the state-of-the-art as well as the more classical uses of approximation. It is not meant to be a comprehensive list. Rather, it serves to observe what areas of design optimization are taking advantage of approximation methods, and where there are opportunities for further development.

2.2.1 Response surface methodology

The technique known as Response Surface Methodology (RSM) is not an algorithm so much as a set of tools grouped together to analyze a problem. As Myers and Montgomery define it, “[RSM] is a collection of statistical and mathematical techniques useful for developing, improving and optimizing processes. It also has important applications in the design, development, and formulation of new products, as well as in the improvement of existing product designs.” [62]. The area has been studied extensively, and there are many comprehensive books that provide a good background [14], [15], [46], [61], [62].

RSM is a sequential process. To begin, an initial point is guessed. With design of experiments theory, data points are chosen and sampled in a small space around the initial point. From there, a linear least squares model is fit to the local area, and ANOVA is conducted to verify the adequacy of the polynomial model. Using

basic calculus, the direction of greatest improvement is computed and a new area of interest in the design space is chosen. The process iterates until a linear fit is no longer adequate, at which point a quadratic model is fit and the optimal setting of the design variables determined.

Given the sequential nature of RSM, global exploration and mapping of the design space is not employed. Rather, RSM is a local exploration method. As Mitchell and Morris quip, “One approaches the problem of finding the maximum response, say, in the same way that a myopic person might climb a hill.” [57].

2.2.2 Sequential quadratic programming (SQP)

Perhaps the most widely used approximation method in optimization is the Taylor series expansion, an eminent example being the Sequential Quadratic Programming (SQP) algorithm. Once a starting point is chosen, the algorithm makes a local quadratic polynomial approximation to the true function (i.e., a second order Taylor series expansion) and a linear approximation to the constraints. This is done most frequently using finite differencing to approximate the gradients and a quasi-Newton approximation of the Hessian of the Lagrangian. SQP then proceeds to determine a feasible descent direction from the quadratic programming approximate problem. A step is taken in that direction and the process of forming a local quadratic programming approximation is repeated until convergence occurs. It essentially works like RSM, but is automated, uses quadratic approximations at each iteration, and is able to account for constraints.

While the functional form of SQP can be much more complicated, this explanation serves to show how local polynomial models are frequently used in design optimization. SQP has found diverse uses in design optimization and is perhaps the

most frequently used nonlinear optimization algorithm. As it has been studied extensively, current implementations of SQP are quite mature. The reader interested in more detail is referred to Papalambros and Wilde [66].

2.2.3 Trust region framework for using surrogates

Researchers at the Institute for Computer Applications in Science and Engineering (ICASE), have recently shown how a Trust Region algorithm can be modified to incorporate more general approximations than the standard second order Taylor series expansion [3]. To begin, it is important to understand the standard Trust Region algorithm.

The fundamental idea behind the Trust Region method is very much similar to SQP. At each iteration, k , the algorithm begins by making a local quadratic approximation q_k of the Lagrangian of the form

$$f(\mathbf{x}_k + \mathbf{s}) \approx q(\mathbf{x}_k + \mathbf{s}) = f(\mathbf{x}_k) + \mathbf{g}_k^t \mathbf{s} + \frac{1}{2} \mathbf{s}^t \mathbf{B}_k \mathbf{s} \quad (2.6)$$

where \mathbf{s} is the next step to be taken (i.e., the change in design variables), \mathbf{g}_k is the gradient of f at \mathbf{x}_k , and \mathbf{B}_k is the Hessian of f at \mathbf{x}_k . As opposed to SQP, the Trust Region algorithm restricts the length of the step within some trust region radius, d . Hence, the Trust Region subproblem is to find the vector \mathbf{s} that solves the following:

$$\begin{aligned} \min_{\mathbf{s}} \quad & q_k(\mathbf{x}_k + \mathbf{s}) \\ \text{subject to:} \quad & \|\mathbf{s}\| < \delta_k \end{aligned} \quad (2.7)$$

Once this problem has been solved approximately (i.e., an exact solution is not required for convergence), a decision is made whether or not to accept the step based upon the realization of an improvement over $f(\mathbf{x}_k)$. After each iteration, the size of the trust region radius can be updated. If the model q_k has performed well in

predicting the actual improvement in f , then it should be allowed a larger radius for taking its next step. Conversely, if q_k has poorly predicted the actual improvement in f , then the trust region can be shrunk to reflect the lack of confidence in the model. Models that do an acceptable job leave the size of the trust region alone.

The measure for determining how well a model has performed is as follows:

$$r = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{s})}{f(\mathbf{x}_k) - q(\mathbf{x}_k + \mathbf{s})} \quad (2.8)$$

Once the positive constants $r_1 < r_2 < 1$ and $c_1 < 1$, $c_2 > 1$ are chosen, then the update to the trust region radius is determined as:

$$\delta_{k+1} = \begin{cases} c_1 \|\mathbf{s}\| & \text{if } r < r_1 \\ \min\{c_2 \|\mathbf{s}\|, \Delta\} & \text{if } r > r_2 \\ \|\mathbf{s}\| & \text{otherwise.} \end{cases} \quad (2.9)$$

where Δ is an upper bound on the trust region radius. Common values for r_1 , r_2 , c_1 and c_2 are 0.25, 0.75, 2 and 0.5, respectively [66].

The ICASE group demonstrated that one can replace the standard quadratic approximation, q_k , with a more general approximation, a_k , without sacrificing the global or local convergence properties of the Trust Region algorithm. The only conditions placed on this approximation is that it must match both the value and gradient of the objective function at the point \mathbf{x}_k . This now provides an even more powerful tool for optimization, as global metamodels such as kriging or neural nets can replace the standard local quadratic models. However, the drawbacks are that this framework is only developed for unconstrained optimization problems. Obviously, the majority of engineering design applications require some form of constraints. This is an area that can be improved with existing extensions of the Trust Region algorithm to constrained optimization.

2.2.4 Metamodels in robust design

Recent work has applied approximation methods to robust design. In robust design, separate noise and control factors are identified. The former refers to values over which the designer has no direct control, e.g., humidity in the production line facility. The latter are the more traditional design variables used in optimization, e.g., part dimensions. The often competing goals of robust design are to find a design that is both optimal according to some design performance criteria and as insensitive as possible to variations in the noise factors. The latter is usually measured by the variance of the response. Two recent publications from Georgia Tech [47], [87] demonstrate their approach to robust design optimization through surrogate modeling.

To begin, a metamodel is fit as a function of both the noise factors, \mathbf{z} , and the control factors, \mathbf{x} . The metamodel consists of a quadratic response model, although any surrogate model could have been chosen. By evaluating the model while varying the noise factors, a mean response

$$\mu_{\hat{y}} = f(\mathbf{x}, \mathbf{z})$$

is defined. The response model is then used to estimate the robustness of the function through the variance of the response as

$$\sigma_{\hat{y}}^2 = \sum_{i=1}^k \left(\frac{\partial f}{\partial z_i} \right)^2 \sigma_{z_i}^2$$

where k is the number of noise factors and $\sigma_{z_i}^2$ is the variance of the i^{th} noise factor.

A similar method for estimating the variance of the response involves evaluating the surrogate model for a great number of noise factor settings to find the mean estimated response values and their variance directly from the model. This Monte

Carlo-type approach results in more accurate estimates of the variance provided the response model fits well.

A third method is to use product arrays to vary the noise factors several times for a single setting of the control factors. In this way, an explicit model of the response and variance can be made directly from the simulation or true function itself. The downside is the large number of runs required and the small number of noise factor settings used for each control setting, thereby leading to a questionable estimation of variance.

Regardless of the method chosen, a model for both the mean and variance of each response is now available. With these models, a Compromise Decision Support Problem (DSP) is formulated. DSP is a multiobjective optimization problem based on goal programming. The objective is to minimize the deviation function (usually a weighted sum of individual deviations), which defines how well the design variables meet their goals. These goals can be both target values for the mean response models (or smaller-the-better functions) and target values for the variance models. In this way, a compromise can be struck to balance minimization and robustness. A comprehensive review of DSP and its application to robust design can be found in Mistree [56].

2.2.5 Variable complexity response surface modeling

Some researchers have created more efficient metamodels by only modeling the most promising region of the design space. This approach is called Variable Complexity Response Surface Modeling (VCRSM) [8], [32], [45]. It is a very simple, yet effective way of reducing the cost and increasing the usefulness of metamodels.

If one could construct a set of simple (e.g., algebraic) constraints that defined

either infeasible or unreasonable designs, then one could quickly evaluate which designs were good and which were bad. This is especially valuable for large scale design. The goal is to make an approximation to the computer simulation using only points that are in the reasonable design space. To that end, an initial design of experiments (DOE) is performed to define a box around some nominal feasible design. Then a factorial design is generated within the box to test points against the easy-to-evaluate constraints. If the points are deemed infeasible, they are discarded immediately or scaled back towards the center point. If more computationally expensive constraints are available, additional DOE sweeps are performed. Eventually, only “reasonable” designs remain. These are then run through the full simulation and stored for later approximations. The designer can use the data for any type of model they see fit. In the optimization phase, the expensive simulation is replaced with the model in the reasonable design space. If a call is made to a point outside this domain, then either the original simulation or some analytical function which smoothly transitions with the metamodel is used. In the latter case, a simple penalty function can be used to steer local optimization algorithms back towards the “reasonable” design space.

2.2.6 Approximation methods in decomposition

Multidisciplinary Design Optimization (MDO) has provided a difficult challenge for engineering optimization. Expensive computer simulations may be required to calculate important design quantities for several different engineering disciplines. The results from one simulation are often fed into another such that the entire design process involves the coupling of several complex analyses. For example, a computational fluid dynamics code could predict the forces that the passing air imposes on an airplane wing. With that information, a finite element analysis could

be conducted to estimate the stress distribution throughout the wing. Each of these codes depends on a distinct, but not mutually exclusive, set of design variables.

Because of the sometimes disjoint nature of the analyses, it is possible to decompose the overall analysis into subspaces of contributing disciplines. Each subspace is defined by the design variables it can change and by the effect it can have on the entire system. One of the more difficult problems in MDO decomposition methods is the question of how to guarantee convergence at the system level and how to resolve conflicting changes made by different subspaces. A research group from the University of Notre Dame has worked on a framework in which approximate models are used to address these problems [9], [74].

To begin, a group of initial designs is selected and evaluated through the system level simulation. The results are put into a database and used to create response surface approximations. In the original work, the approximations were quadratic models, but neural networks were used in later studies. Next, subspace optimization is performed. Each discipline changes its own set of design variables in an attempt to improve the system level merit function or to bring the design towards the feasible region. However, changing the design variables from one subspace often effects what happens in the others. Rather than having each subspace discipline run the other subspace discipline-specific codes to find how their changes effect others, an approximate model of the other disciplines is used. In this way, designers now have information to efficiently approximate the impact their decisions have on nonlocal responses.

The goal of the subsystem level optimization is not necessarily to find the best possible subspace design. Rather, the framework presented in the work is set up so that the subspace designers are able to suggest a list of candidate designs. These

new designs are then run through the system level analysis code and added to the database. The response surfaces are updated and now the fully approximate system coordination problem is addressed. Because the system level problem is approximated by the response surfaces, it can be more efficient than using the full simulation. Convergence checks are made, and if necessary, the subspace level optimization is begun again.

2.2.7 Exploiting functional cost disparities

An interesting feature common to many computer simulations is that the time required to evaluate one function (e.g., the objective function) can differ by orders of magnitude from others (e.g., the constraint functions). Nelson suggested ways to take advantage of this characteristic in a modified Trust Region algorithm [64]. He directly uses functions (either the objective function or constraints) that are inexpensive to compute while keeping the traditional quadratic approximation for expensive functions. By doing so, subsequent iterates are more accurate, thereby reducing the total number of calls made to the expensive functions. Even though this requires more calls to the inexpensive functions, the algorithm makes fewer calls to the expensive functions, and thus the overall time required to solve the optimization problem is reduced. While theoretical convergence is not fully proven, several examples in the work provide experimental evidence to support the claim.

2.3 Origins and Development of EGO

In this section we describe the Efficient Global Optimization (EGO) algorithm that forms the basis of this thesis. While the algorithm did not appear until the late 1990's, its origins date back to the 1960's. EGO fits into a general class of optimization algorithms which we will refer to as *Bayesian analysis algorithms*. These

global searching algorithms use statistical models from an initial data sample to determine where to next evaluate the functions via an auxiliary optimization problem. In this section, the historical perspective of Bayesian analysis algorithms and their development over time is examined. More specifically, issues such as the ability to solve problems in multidimensional space, the choice of sampling criteria, convergence properties and stopping criteria are discussed.

2.3.1 Bayesian analysis

The first known implementation of a Bayesian analysis optimization algorithm was presented by Kushner in 1964 [51]. In this work, the weaknesses of gradient-based optimization algorithms were discussed. Kushner stated that local optimizers are well known for not overcoming local minima when searching for the global minimum. Intuitively, one would prefer to have an algorithm that searched the entire design space at each iteration rather than relying on purely local information. To that end, Kushner introduced Bayesian analysis using the Wiener process model.

In the original work, the algorithm was designed for one-dimensional, unconstrained problems. Using a statistical model of the objective function, Kushner's method searched for the optimum by locating the point on the metamodel with the highest probability of improving upon the current best point f_{min} by some amount ϵ , that is,

$$\max_x P(\hat{y}(x) < f_{min} - \epsilon) .$$

Kushner controlled how local or how global the algorithm searched the space by changing ϵ as the iterations progressed. Starting with a large ϵ (more global search), it then narrowed to a more local refinement by reducing ϵ . Because the intervals between sample points are independent for the Wiener process, the algorithm could

analytically search each interval to calculate the maximum of the probability measure. The interval containing the highest probability measure was chosen, a new sample taken, and the Wiener process model updated. The search terminated when the probability of finding a better point became sufficiently small. While the formulae may deviate from what is now used in EGO, the concept was identical - search for a better point based on a statistical model that predicts the probability of exceeding the current best design.

It took quite some time before the Bayesian approach took root in the field of global optimization. Some early research into Bayesian methods can be found in Part One of Dixon and Szegö [25]. It wasn't until the 1980's that the *P-algorithm* as it came to be known [105] began to see more widespread attention. Advances were made in both the theoretical and practical aspects.

The remainder of this section highlights some of the research in Bayesian analysis that eventually led to the work at hand. Other literature reviews on the topic can be found in the following references [11], [25], [58], [92], [96], [106]. The most recent review found at the time of this dissertation is that of Jones [41].

2.3.2 Extending the P-algorithm to multiple dimensions

One of the key shortcomings to Kushner's original work was the fact that the search was restricted to a single variable due to the use of the Wiener process. By the late 1980's, many researchers had extended Bayesian analysis to multidimensional space through a variety of techniques. Stuckman's implementation [91] of the P-algorithm used 1-D projections of the Wiener process along lines connecting each sample point. As with the original P-algorithm, each line segment was searched to determine the best location for the next sample. Perttunen used Delauney trian-

gulization on Stuckman’s algorithm to divide the space into n dimensional simplices [67], [71], [68]. This was a significant development because iterates were no longer restricted to lie along line segments between previously sampled points. Elder also used a simplex division approach in his GROPE algorithm [27], another implementation of the P-algorithm. Strongin used a Peano mapping technique to solve multidimensional problems [90]. Both Mockus and Žilinskas also successfully extended Bayesian techniques to n dimensions [58],[59], [103], [106]. It should be noted that most researchers advise the use of Bayesian algorithms only for small problems, say $n \leq 10$ [106].

2.3.3 Choice of statistical models

Another avenue of research has been the statistical model used to locate the next sample point. Originally, the Wiener process was used because the Markovian properties provided a simple statistical model that was separable for each interval between the sample points. Many Wiener-based algorithms were introduced [4], [51], [53], [54], [75], [105].

Perttunen investigated variations on the Brownian motion model [69]. One particular model considered was the Ornstein-Uhrbeck process, a close relative of the kriging model used in this thesis. Perttunen and Stuckman applied a rank transform to the sampled data in order to better satisfy the assumption of normality in the model [70], [71]. This transform was also shown to increase the efficiency of the algorithm. Locatelli allowed for the Wiener model parameter σ to be fit iteratively during the optimization, enabling a more accurate model of the variance [53]. Calvin and Žilinskas incorporated derivative information into the model for one-dimensional functions with continuous derivatives [18].

Other researchers departed from the Wiener process model completely. A significant consequence was that solving problems in R^n became straightforward because the model itself allowed for it. Gutmann successfully applied Radial Basis Function models to a statistical-based algorithm [35]. Björkman and Holström have implemented Gutmann’s algorithm in Matlab [13]. The kriging model-based EGO algorithm [83], [84], [44], is the algorithm most closely related to this thesis. The algorithm was originally named Stochastic Processes Analysis of Computer Experiments or *SPACE* in Schonlau’s Ph.D. thesis.

2.3.4 Choice of infill sampling criterion

Once the choice of a statistical model has been made, one can determine which design point to sample next. This is done by numerically finding the optimum of some auxiliary function. Throughout this thesis, this criterion is called the infill sampling criterion (ISC), taken from the geostatistics literature. However, one should note that the same concept is also given the name *risk function* or *loss function* in the Bayesian analysis literature [10].

As described above, Kushner’s original work used the following criterion:

$$\max_x P(\hat{y}(x) < f_{min} - \epsilon). \quad (2.10)$$

Stuckman implemented a version of this criterion for special cases where the minimum objective function value, or some approximation thereof, is known *a priori* [93]. He shows examples in fields such as curve fitting where the theoretical least squares minimum is $f_{min} = 0$, and we want to find a solution ϵ close to that bound.

Other researchers have attempted to minimize the average deviation from the best known point. Mockus’ algorithm [58], known also as the one-step Bayesian

method, quantified this as:

$$\min_x \frac{1}{\sqrt{2\pi}\hat{\sigma}(x)} \int_{-\infty}^{\infty} \min(y, c) \exp \frac{-(y-\hat{y}(x))^2}{2\hat{\sigma}^2(x)} dy, \quad (2.11)$$

where $\hat{y}(x)$ and $\hat{\sigma}^2(x)$ are conditional mean and variance of the model at the point x , and $c = f_{min} - \epsilon$, $\epsilon > 0$. An adaptive version of this criterion [103] utilizes

$$\max_x \min_i (\hat{\sigma}(x_i)/(\hat{y}(x_i) - c), \quad (2.12)$$

where c is defined as above and $\hat{y}(x_i)$ and $\hat{\sigma}^2(x_i)$ are the conditional mean and variance of the model at the sample point x_i .

Another group of researchers developed a criterion to quantify the benefit expected from sampling at a given location. Locatelli applied this to the Wiener process model [53], [54], while Jones, Schonlau and Welch applied this to kriging models [44]. The so-called expected improvement function, shown below, forms the basis for EGO.

$$\max_x (f_{min} - \hat{y}(x)) \cdot \Phi(z) + \hat{\sigma}(x) \cdot \phi(z), \quad (2.13)$$

where

$$z = \frac{f_{min} - \hat{y}(x)}{\hat{\sigma}(x)},$$

and $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative density and probability density functions of the standard normal distribution, respectively.

Cox and John used an approach whereby a lower bounding function was minimized at each iteration [20], [21].

$$\min_x \hat{y}(x) - b\hat{\sigma}(x), \quad (2.14)$$

where b is a user-controlled parameter that defines how globally or locally to search.

Cox and John reported results using $b = 2$ and $b = 2.5$.

Gutmann [35] proposed a criterion based on the smoothness of a radial basis function model of the objective function. Gutmann’s algorithm works significantly differently than those mentioned above. At each iteration, a target value f^* for the objective function is set (similar to $f_{min} - \epsilon$ in the P-algorithm), and the algorithm seeks to find which location x is most likely to yield such a value. Gutmann selects the next iterate based on the following measure of the “bumpiness” of the resulting model:

$$\min_x (-1)^{m_0+1} \mu(x) (\hat{y}(x) - f^*)^2, \quad (2.15)$$

where m_0 and $\mu(x)$ are model parameters. In other words, it seeks the location for x that is most consistent with the previously sampled data.

There are obviously many more choices for sampling criteria. Sasena, Papalambros and Goovaerts [82] compare several of those listed here and others. Chapter 4 of this thesis examines the issue of sampling criteria in more depth.

With all these methods, it is important to keep in mind practicality. Solving the design optimization problem via Bayesian analysis requires the additional optimization problem of minimizing the ISC at each iteration. Because this drastically adds to the CPU time, researchers often remark that such algorithms are best suited to situations where the objective function is expensive to compute [51]. It is also posited that exact minimization of the ISC is not critical because the purpose of the ISC is simply to determine the next sample location [60].

2.3.5 Convergence properties and stopping rules

Like most global optimization algorithms, proving convergence for Bayesian algorithms is difficult. A theorem by Törn and Žilinskas [96] states, as paraphrased by Jones, [41] “in order to guarantee convergence to the global minimum [of a con-

tinuous function], we must converge to every point in the domain". As long as an algorithm periodically samples in still unexplored regions, it will converge in the limit. While this is not terribly practical, it at least gives a *necessary* property for a globally convergent algorithm. A number of authors describe Bayesian algorithms that adhere to this theory of convergence [18], [54], [102], [105], [106]. Gutmann shows a convergence proof for his Radial Basis Function based algorithm [35].

Work has also been done on convergence rates for global algorithms. Mockus defines the density ratio as the ratio of density of observations in the vicinity of an optimum to the average density of observations [59]. He states that the density ratio can be regarded as a replacement of rate of convergence. Calvin also discusses the convergence rate specific to the P-algorithm [16]. One of the main findings is that the local convergence of the P-algorithms is quite poor [17], [96]. This is due to the piecewise continuous nature of the Wiener process. The algorithm can quickly identify the region around the global optimum, but cannot accurately refine the solution because the statistical model cannot smoothly match the true continuous function locally.

To address this problem, several researchers have created hybrid approaches. The P*-algorithm of Žilinskas [104] uses hypothesis testing to check if a locally optimal region has been found. If so, an SQP-type algorithm is employed to refine the solution, then global searching recommences. This method has successfully increased the local efficiency of the P-algorithm. In a similar approach, Locatelli's two-phase algorithm [53] periodically switches from a global search to a local refinement about the best point. Locatelli and Schoen also show that their algorithm terminates in a finite amount of time [54].

Equally important to the convergence proof is the convergence criterion. For most

Bayesian algorithms, the search is terminated by a limit on the number of function evaluations. Other termination criteria have been investigated, however. Most of these involve statistically predicting the benefits from additional sampling.

It is first important to understand the k -step look ahead or k -sla rule. As Betrò defines, “a k -sla rule calls for stopping the sampling process as soon as the current cost [of stopping prematurely] is not greater than the cost expected if at most k further observations are taken” [11]. An optimal strategy would be one that correctly identifies the N observations required to identify the global optimum within the specified tolerance. Of course, the inability to numerically define such a conditional probability leads to 1-sla rules whereby the next iterate is assumed to be the last. This approximation of the optimal search strategy makes the algorithm tractable [58], [106]. Betrò and Schoen examine the benefits of applying a 2-sla rule [12]. They comment that the added computational difficulty does not justify the meager gains in solution efficiency.

The 1-sla rule leads to intuitive termination criteria that stops the algorithm once the cost of sampling an additional observation exceeds the expected improvement over the current best point. Defining the cost of sampling is left to the user. Such a criterion works quite naturally with algorithms that use Equation (2.13) as the ISC and has been implemented in several algorithms [44], [54], [83].

Cox and John propose another criterion based on their lower confidence bounding function shown in Equation (2.14) [20], [21]. For a value of $b = 2$, Cox and John show that the probability of the actual function being below the lower bounding function is less than 2.5%. Their algorithm terminates if the current best point is below the best value of the lower bounding function computed at a predetermined set of prediction sites.

2.3.6 Miscellaneous advancements

In recent years, numerous other advancements have been made in Bayesian analysis. Schonlau discussed the advantages of searching for several new sample locations at each iteration [83]. To accomplish this, the expected improvement criterion of Equation (2.13) is numerically solved several times before sampling the new points. Jones [41] proposes selecting multiple candidates by using several different targets for improvement at each iteration. This can be done either by using several values for ϵ in Equation (2.10) or for b in Equation (2.14). Doing so allows the algorithm to choose several candidate sample locations at each iteration, some using local search, some using global search. Clustering analysis can then be used to select the most efficient subset of candidates.

A major area where little research has been done to date is constrained optimization. Björkman and Holström [13] use a penalty method to transform the constrained optimization problem to an unconstrained one. Schonlau [83] multiplies the expected improvement by the probability of feasibility to transform to an unconstrained problem. The limitations of both these approaches is discussed by Sasena, Papalambros and Goovaerts [80]. In related papers, we propose incorporating the constraints directly into the ISC subproblem using a constrained optimizer [81], [79]. Constrained Bayesian analysis represents one of the main contributions of this dissertation and is discussed further in Chapter 5.

2.4 Chapter Summary

As seen in this chapter, there are a multitude of approximation techniques in the open literature. This dissertation focuses on one particular technique, kriging. This decision is not baseless. Other researchers such as Simpson [86] have made

the argument that kriging provides a suitable approximation technique for design optimization that uses computer simulations to evaluate the model functions.

Another point illustrated in this chapter is that optimization algorithms can make use of approximations in a wide variety of ways. Bayesian analysis is chosen for further research because it is both simple and very flexible. As kriging models have been selected for the statistical basis, the EGO algorithm serves as an appropriate starting point for this thesis. The framework of EGO will be used to explore new ideas in alternative sampling criteria and constraint handling.

In the next chapter, the kriging technique is explored in more depth.

CHAPTER 3

Understanding Kriging

In this chapter, the approximation method known as kriging is discussed in detail. The prediction equations are derived, and the model fitting procedures explained. Differences between two forms of kriging are examined - geostatistics and DACE. Finally, some strategies for improving the robustness of the fitting procedure are proposed.

3.1 History and Terminology

There has been much interest of late in a form of curve fitting and prediction known as kriging. The name refers to a South African geologist, D. G. Krige who first used the statistics-based technique in analyzing mining data in the 1950s and 1960s [49]. His work was furthered in the early 1970s by authors such as Matheron [55] and formed the foundation for a entire field of study now known as geostatistics [23], [34].

At the end of the 1980s, kriging was taken in a new direction as a group of statisticians applied the techniques of kriging to deterministic computer experiments [77], [78], [100]. This new research avenue was quickly explored by a variety of authors as the potential for more general engineering work became apparent. Their process

of experimental design and using kriging models was named Design and Analysis of Computer Experiments (DACE), much as RSM refers to a set of tools for modeling and optimizing a system. The form of the kriging equations used in DACE branched off from traditional geostatistics in the sense that it concentrated and specialized on only a small portion of the available knowledge. To distinguish this form of the kriging equations from those found in geostatistics, care will be taken in this chapter to use the specific terms DACE and geostatistics when differentiating between the two.

3.2 General Kriging Approach

The following two sections are devoted to developing the equations found in kriging. Much of the derivation below follows the presentation of Schonlau [83]. Here, a general overview of kriging is given. In the next section, the formulae used in fitting the model and using it in prediction are derived.

The majority of metamodeling techniques rely on the decomposition $Y(x) = f(x) + \epsilon(x)$, where the residuals $\epsilon(x)$ are assumed to be independent and identically distributed (normal distribution), i.e., $\sim N(0, \sigma^2) \forall x$. The $Y(x)$ is capitalized in order to specify it as being a random variable. Any particular realization of that random variable would be written in lower case, as $y(x)$. The main idea behind kriging meta-models is that the error in the predicted values, $\epsilon(x)$, are *not* independent. Rather, they take the view that the errors are a systematic function of x . To understand this statement better, consider the following illustration where a quadratic equation is fit via least squares to a given data set. Figure 3.1 shows a narrow region around two sample point of some true function $y(x)$ (solid line) and its associated regression model $f(x)$ (dashed line). At the sample point x_i , we know the true function value,

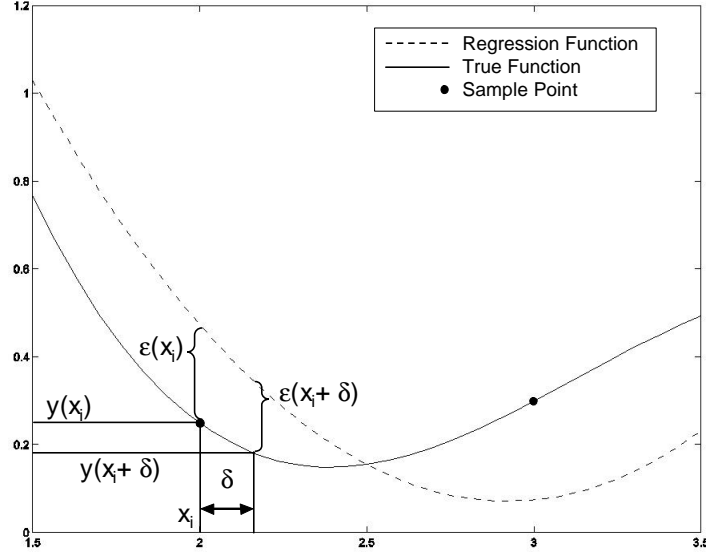


Figure 3.1: Illustration that errors in regression function depend on x

$y(x_i)$ and therefore the error $\epsilon(x_i) = y(x_i) - f(x_i)$. At a location $x_i + \delta$, for some small δ , we only know the value of regression function $f(x_i + \delta)$. Kriging theory posits that if the error at x_i is large, then there is good reason to believe that the error at $x_i + \delta$ is also large. This is because there is a systematic error associated with the functional form $f(x)$. The closer a point x is to x_i , the more likely $f(x)$ is to deviate from the $y(x)$ by the same amount as $\epsilon(x_i)$.

A kriging metamodel in multidimensions, which takes the form $Y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x})$, is comprised of two parts: a polynomial $f(\mathbf{x})$ and a functional departure from that polynomial $Z(\mathbf{x})$. This can be written as

$$Y(\mathbf{x}) = \sum_{j=1}^k \beta_j f_j(\mathbf{x}) + Z(\mathbf{x}), \quad (3.1)$$

where $f_j(\mathbf{x})$ are the basis functions (i.e., the polynomial terms), β_j are the corresponding coefficients, and $Z(\mathbf{x})$ is a Gaussian process. More precisely, $Z(\mathbf{x})$ represents uncertainty in the mean of $Y(\mathbf{x})$ with $E(Z(\mathbf{x})) = 0$ and

$$\text{Cov}(Z(\mathbf{w}), Z(\mathbf{x})) = \sigma_z^2 \mathbf{R}(\mathbf{w}, \mathbf{x}). \quad (3.2)$$

Here, σ_z^2 is a scale factor known as the process variance that can be tuned to the data and $R(\mathbf{w}, \mathbf{x})$ is known as the spatial correlation function (SCF).

The choice of the SCF determines how the metamodel fits the data. There are many choices of $R(\mathbf{w}, \mathbf{x})$ to quantify how quickly and how smoothly the function moves from point \mathbf{x} to point \mathbf{w} . One of the most common SCF's (shown here for one-dimensional points w and x) used in DACE is

$$R(w, x) = e^{-\theta|w-x|^p}, \quad (3.3)$$

where $\theta > 0$ and $0 < p \leq 2$. As with all choices of the SCF, the function goes to zero as $|w - x|$ increases. This shows that the influence of a sampled data point on the point to be predicted becomes weaker as their separation distance increases. The magnitude of θ dictates how quickly that influence deteriorates. For large values of θ , only the data points very near each other are well correlated. For small values of θ , points further away still influence the point to be predicted because they are still well correlated. The parameter p is a measure of smoothness. The response becomes increasingly smooth with increasing value of p . One should also note the interpolating behavior of the SCF. Because $R(\mathbf{x}, \mathbf{x}) = 1$, the predictor will go exactly through any measured data point. With more advanced techniques, one can allow the prediction to be non-interpolating. Such a method is presented in Section 3.7.2.

To extend the SCF to several dimensions, common practice in DACE is to multiply the correlation functions. The so-called *product correlation rule* applied to Equation (3.3) results in

$$R(\mathbf{w}, \mathbf{x}) = \prod_{d=1}^D e^{-\theta^d |w^d - x^d|^{p^d}}, \quad (3.4)$$

where the superscript $d = 1$ to D refers to the dimension.

The metamodels are robust enough to allow a different choice of SCF for each dimension. For example, a different choice of θ and p is possible for each dimension $j = 1$ to d , but some authors suggest that using a single correlation parameter gives sufficiently good results [77]. In the work at hand, θ and p are fit for each dimension.

3.3 Derivation of the Prediction Formula

In deriving the prediction equations, it is necessary to first define several quantities and formulae. We begin by revisiting Equation (3.1). The polynomial term of the model comprises of the $k \times 1$ vector of regression functions

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})]^t$$

and the associated $k \times 1$ vector of constants

$$\beta = [\beta_1, \beta_2, \dots, \beta_k]^t .$$

We next define the $n \times k$ expanded design matrix

$$\mathbf{F} = \begin{bmatrix} \mathbf{f}^t(\mathbf{x}_1) \\ \mathbf{f}^t(\mathbf{x}_2) \\ \vdots \\ \mathbf{f}^t(\mathbf{x}_n) \end{bmatrix} .$$

Notice that each row is one $\mathbf{f}(\mathbf{x})$ vector and there are n rows for the n sampled points.

If we notate the stochastic process as

$$\mathbf{z} = [Z(\mathbf{x}_1), Z(\mathbf{x}_2), \dots, Z(\mathbf{x}_n)]^t$$

then for the output of the sampled data, $\mathbf{y} = [y_1, y_2, \dots, y_n]^t$, one can rewrite equation (3.1) as

$$\mathbf{y} = \mathbf{F}\beta + \mathbf{z} . \tag{3.5}$$

Finally, let \mathbf{R} be the $n \times n$ correlation matrix with element i, j defined by $R(\mathbf{x}_i, \mathbf{x}_j)$ as in Equation (3.4) and let

$$\mathbf{r}_\mathbf{x} = [R(\mathbf{x}_1, \mathbf{x}), R(\mathbf{x}_2, \mathbf{x}), \dots, R(\mathbf{x}_n, \mathbf{x})]^t$$

be the $n \times 1$ vector of correlations between the point at which to predict, \mathbf{x} , and all the previously sampled data points \mathbf{x}_i .

The goal of this derivation is to determine what is called the Best Linear Unbiased Predictor (BLUP). To do so, we will need a measure of the prediction error to quantify the best covariance model and a constraint to ensure unbiasedness. This can be written as a minimization problem using Lagrange parameters for the constraints. The term unbiasedness refers to the fact that the expected value of the linear predictor must equal the expected value of Equation (3.5). A linear predictor is one of the form $\mathbf{c}_\mathbf{x}^t \mathbf{y}$. That is, it uses a linear combination of the sampled data points, \mathbf{y} , with the linear weighting vector $\mathbf{c}_\mathbf{x}$. Thus, an unbiased linear predictor is one for which $E(\mathbf{c}_\mathbf{x}^t \mathbf{y}) = \mathbf{c}_\mathbf{x}^t \mathbf{F} \beta$, since $E(Z(\mathbf{x})) = 0$ by definition. This also means that by similar reasoning, $E(\hat{y}(x)) = \mathbf{f}_\mathbf{x}^t \beta$. Equating these two for all β , we obtain the set of k unbiasedness constraints:

$$\mathbf{F}^t \mathbf{c}_\mathbf{x} = \mathbf{f}_\mathbf{x} . \quad (3.6)$$

We now show that for any linear predictor $\mathbf{c}_\mathbf{x}^t \mathbf{y}$ of $Y(\mathbf{x})$, the mean squared error of prediction is

$$\begin{aligned} E(\mathbf{c}_\mathbf{x}^t \mathbf{y} - Y(\mathbf{x}))^2 &= E[\mathbf{c}_\mathbf{x}^t \mathbf{y} \mathbf{y}^t \mathbf{c}_\mathbf{x} + Y^2(\mathbf{x}) - 2\mathbf{c}_\mathbf{x}^t \mathbf{y} Y(\mathbf{x})] \\ &= E[\mathbf{c}_\mathbf{x}^t (\mathbf{F} \beta + \mathbf{z}) (\mathbf{F} \beta + \mathbf{z})^t \mathbf{c}_\mathbf{x} + (\mathbf{f}_\mathbf{x}^t \beta + Z(\mathbf{x}))^2 \\ &\quad - 2\mathbf{c}_\mathbf{x}^t (\mathbf{F} \beta + \mathbf{z}) (\mathbf{f}_\mathbf{x}^t \beta + Z(\mathbf{x}))] \\ &= (\mathbf{c}_\mathbf{x}^t \mathbf{F} \beta - \mathbf{f}_\mathbf{x}^t \beta)^2 + \mathbf{c}_\mathbf{x}^t \sigma_z^2 \mathbf{R} \mathbf{c}_\mathbf{x} + \sigma_z^2 - 2\mathbf{c}_\mathbf{x}^t \sigma_z^2 \mathbf{r}_\mathbf{x} . \end{aligned}$$

If we substitute the unbiasedness constraint into the above equation, we eliminate the first term and obtain

$$E[\mathbf{c}_x^t \mathbf{y} - Y(\mathbf{x})]^2 = \mathbf{c}_x^t \sigma_z^2 \mathbf{R} \mathbf{c}_x + \sigma_z^2 - 2\mathbf{c}_x^t \sigma_z^2 \mathbf{r}_x . \quad (3.7)$$

Introducing the Lagrange multipliers λ_i , $i = 1$ to k for the constraints, the best linear unbiased predictor is the one that minimizes the error above. The formal statement is

$$\min_{\mathbf{c}_x} [\mathbf{c}_x^t \sigma_z^2 \mathbf{R} \mathbf{c}_x + \sigma_z^2 - 2\mathbf{c}_x^t \sigma_z^2 \mathbf{r}_x] - \lambda [\mathbf{F}^t \mathbf{c}_x - \mathbf{f}_x] \quad (3.8)$$

$$\text{subject to: } \mathbf{F}^t \mathbf{c}_x - \mathbf{f}_x = 0$$

Taking the partial derivative of the objective function with respect to \mathbf{c}_x yields

$$\sigma_z^2 \mathbf{R} \mathbf{c}_x - \sigma_z^2 \mathbf{r}_x - \mathbf{F} \lambda = 0 .$$

Notice that the factor of two was removed from the first two terms because defining a new choice of Lagrange multipliers $\lambda' = 2\lambda$ is permissible. From optimization theory, we now have two sets of equations to solve in the unknown vectors \mathbf{c}_x and λ :

$$\mathbf{F}^t \mathbf{c}_x = \mathbf{f}_x .$$

$$\sigma_z^2 \mathbf{R} \mathbf{c}_x - \mathbf{F} \lambda = \sigma_z^2 \mathbf{r}_x$$

Rewriting this system of equations in matrix form yields:

$$\begin{bmatrix} \mathbf{0} & \mathbf{F}^t \\ \mathbf{F} & \sigma_z^2 \mathbf{R} \end{bmatrix} \begin{bmatrix} -\lambda \\ \mathbf{c}_x \end{bmatrix} = \begin{bmatrix} \mathbf{f}_x \\ \sigma_z^2 \mathbf{r}_x \end{bmatrix} . \quad (3.9)$$

Solution of this system leaves the BLUP as

$$\hat{y}(\mathbf{x}) = \mathbf{c}_x^t \mathbf{y} = \begin{bmatrix} \mathbf{f}_x^t & \mathbf{r}_x^t \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{y} \end{bmatrix} \quad (3.10)$$

which is more easily seen as

$$\hat{y}(\mathbf{x}) = \mathbf{f}_\mathbf{x}^t \hat{\beta} + \mathbf{r}_\mathbf{x}^t \mathbf{R}^{-1}(\mathbf{y} - \mathbf{F} \hat{\beta}) , \quad (3.11)$$

where

$$\hat{\beta} = (\mathbf{F}^t \mathbf{R}^{-1} \mathbf{F})^{-1} \mathbf{F}^t \mathbf{R}^{-1} \mathbf{y} \quad (3.12)$$

is the generalized least squares estimator of β .

Often, one is also interested in the mean squared error (MSE) of the estimate. This measure provides an estimate of how reliable the prediction is at a given location. The MSE is defined as the square of the expected value of the difference between the predictor and the true value. We have already defined that quantity in Equation (3.7).

$$\begin{aligned} MSE[\hat{y}(\mathbf{x})] &= E[\mathbf{c}_\mathbf{x}^t \mathbf{y} - Y(\mathbf{x})]^2 \\ &= \sigma_z^2 [1 + \mathbf{c}_\mathbf{x}^t \mathbf{R} \mathbf{c}_\mathbf{x} - 2 \mathbf{c}_\mathbf{x}^t \mathbf{r}_\mathbf{x}] . \end{aligned}$$

Substituting in the matrices from Equation (3.9) leaves

$$MSE[\hat{y}(\mathbf{x})] \equiv \hat{\sigma}^2(\mathbf{x}) = \sigma_z^2 \left[1 - \begin{bmatrix} \mathbf{f}_\mathbf{x}^t & \mathbf{r}_\mathbf{x}^t \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f}_\mathbf{x} \\ \mathbf{r}_\mathbf{x} \end{bmatrix} \right] . \quad (3.13)$$

Another, less complicated formula for the kriging variance derived in the same Schonlau work [83] is

$$MSE[\hat{y}(\mathbf{x})] \equiv \hat{\sigma}^2(\mathbf{x}) = \sigma_z^2 (1 - \mathbf{r}_\mathbf{x}^t \mathbf{R}^{-1} \mathbf{r}_\mathbf{x}) . \quad (3.14)$$

The only difference between Equations (3.13) and (3.14) is that the former takes into account the estimation of β and is therefore slightly more accurate. Using either formula, the measure will be small near sampled data points and large when predictions are made far away from the known data locations. Note the use of the

subscript z to differentiate between the process variance parameter, σ_z^2 of Equation (3.2) and the kriging variance, $\sigma^2(\mathbf{x})$, of Equations (3.13) and (3.14). Care is taken to use this notational convention throughout the thesis.

In summary, the kriging models estimate the mean value and variance as

$$\hat{y}(\mathbf{x}) = \mathbf{f}_x^t \hat{\beta} + \mathbf{r}_x^t \mathbf{R}^{-1}(\mathbf{y} - \mathbf{F} \hat{\beta}) \quad (3.15)$$

$$\hat{\sigma}^2(\mathbf{x}) = \sigma_z^2 \left[1 - \begin{bmatrix} \mathbf{f}_x^t & \mathbf{r}_x^t \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f}_x \\ \mathbf{r}_x \end{bmatrix} \right] \approx \sigma_z^2 (1 - \mathbf{r}_x^t \mathbf{R}^{-1} \mathbf{r}_x) . \quad (3.16)$$

3.4 Parameter Estimation

With Equations (3.11) and (3.14) above, one can predict the response and MSE at any untried location. However, the question still remains of what values to use for the parameters σ_z^2 in Equation (3.2) and θ and p in the SCF of Equation (3.3). The DACE approach applies maximum likelihood estimation (MLE). While it will not be proven here, it can be shown [83] that the MLE of β is the generalized least squares estimator shown in Equation (3.12). The MLE of σ_z^2 is

$$\hat{\sigma}_z^2 = \frac{1}{n} (\mathbf{y} - \mathbf{F} \hat{\beta})^t \mathbf{R}^{-1} (\mathbf{y} - \mathbf{F} \hat{\beta}) \quad (3.17)$$

One can then write the log-likelihood of the stochastic process as

$$-\frac{1}{2} [n \ln(\hat{\sigma}_z^2) + \ln(\det(\mathbf{R}))] \quad (3.18)$$

Equation (3.18) is a function of the unknown SCF parameters θ and p only. Thus, one can use numerical optimization techniques to iterate on these unknowns until the maximum of this likelihood function (i.e., the maximum likelihood estimate) is obtained.

Practically speaking, this optimization procedure can be very time consuming because of the inversion of the $n \times n$ \mathbf{R} matrix required in Equation (3.17) for each set of covariance model parameters evaluated. If one has a large data sample, say more than 100 data, then the process can become cumbersome. Out of these practical concerns, some simplifications can be made. For one, the value of p is often fixed at 2 for all dimensions. The SCF is most often robust enough to adequately fit the data with only changes in θ . Also, many suggest [77] that the global trend can be reduced to a simple constant term model (i.e., $f(\mathbf{x}) = \beta$) without a significant loss in model fidelity. This simplification, known as *Ordinary Kriging* in geostatistics, results in the following formulae

$$\hat{y}_{OK}(\mathbf{x}) = \hat{\beta} + \mathbf{r}_{\mathbf{x}}^t \mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\beta}) , \quad (3.19)$$

where

$$\hat{\beta} = (\mathbf{1}^t \mathbf{R}^{-1} \mathbf{1})^{-1} \mathbf{1}^t \mathbf{R}^{-1} \mathbf{y} , \quad (3.20)$$

and $\mathbf{1}$ is an $n \times 1$ vector ones.

Figure 3.2 illustrates the impact that the polynomial order of the global trend has on the resulting model. Three models are fit to the data with a constant term, linear, and quadratic global trend shown as solid, dashed and dotted lines, respectively. As one can see in Figure 3.2(left), the difference between the models is practically negligible in the region around the data. This demonstrates why there is little reason to fit anything other than the simplest model, a constant term global trend.

However, when one examines the behavior of the models in extrapolation (Figure 3.2(right)), the differences are drastic. When predictions are made far from existing data (i.e., extrapolation), $\mathbf{r}_{\mathbf{x}}$ becomes a vector of zeros, and the kriging model degenerates to the global trend. For this reason, one must be extremely cautious in

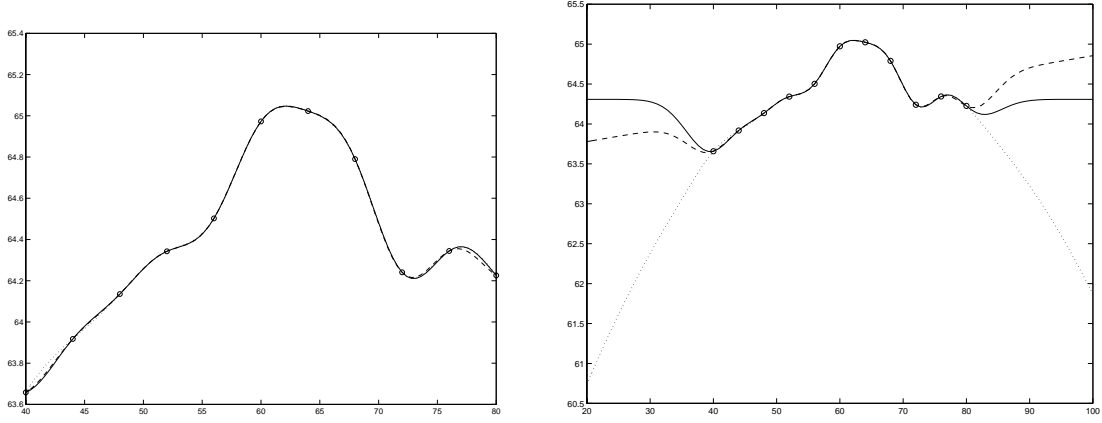


Figure 3.2: Illustration of the impact of the global trend. Different models are fit with a constant term, linear, and quadratic global trend shown as solid, dashed and dotted lines, respectively.

the choice of global trend functions if the kriging models are to be used in extrapolation. The general consensus among researchers, however, is to avoid using kriging for extrapolation whenever possible.

3.5 Effect of the SCF and DOE on Model Accuracy

One question that arises is how much the choice of the SCF influences the accuracy of the kriging model. Simpson addressed the question with an empirical study using five different SCFs including the exponential model shown in Equation (3.3) [86]. He used a testbed of six analytical functions to assess the accuracy of the models. Three measures were used to quantify the accuracy of the metamodel: the maximum absolute error, the root mean square error (RMSE), and the cross validation root mean square error. These measures were all normalized by the corresponding sample range so that responses of different magnitudes could be compared directly. Simpson found that the exponential model performed at least as well as all others to which it was compared. For that reason, the work here is at least partially justified in using the exponential SCF for all kriging metamodels.

Another question that arises is how much the initial choice of design points to be sampled (i.e., the design of experiments) influences the accuracy of the kriging model. Simpson also addressed this issue by comparing nine different space-filling and classical designs. The same examples as before were tested to measure the average error, RMSE, and maximum error associated with each DOE. It was found that the classical DOE's worked reasonably well for a small number of design variables, but problems with four or more variables showed better accuracy with the space-filling designs. There was no specific space-filling design that performed better than any of the others. Therefore, little attention will be paid to the type of space-filling design used in this work. If the DOE's used perform reasonably well, no effort will be made to find better experimental designs.

3.6 How Does DACE Differ from Geostatistics?

The DACE and geostatistical approaches to kriging, while quite similar, have some important differences. In this section, we look at some of the most significant differences in order to better understand the form of the DACE approach taken in this dissertation.

3.6.1 Model fitting

The most significant difference between DACE and geostatistics is the model fitting method. The standard practice in DACE is to use maximum likelihood estimation to find the covariance model parameters. Doing so involves numerically maximizing the closed-form expression of Equation (3.18). It is most often used as a black-box fitting method; however, it is prone to error because it is not completely robust. At times, the optimization algorithm may fail to find the true minimum of Equation (3.18), and the model will fit poorly. This is often the case because the

likelihood function can be multimodal with sharp peaks and valleys. Other times, the likelihood function itself may be monotonic, forcing the covariance model parameters to the extreme values, resulting in a poor fit.

Geostatisticians, on the other hand, commonly fit models with the so-called *variogram*, a plot of the average dissimilarity between sampled data points versus their separation distance. Large variogram values indicate little correlation between data points at that separation distance (called a *lag*). Typically, the variogram function takes small values at small lags where correlation is strong, then increases until it reaches a plateau, or *sill*. The lag distance at which it reaches the sill is called the *range*. These sill and range are analogous to σ_z^2 and θ in DACE modeling Equations (3.2) and (3.3), respectively. Under the assumption of second-order stationarity, the variogram

$$\gamma(h) = \frac{1}{2} \text{Var}[Z(x), Z(x+h)]$$

is related to the covariance function of Equation (3.2) as:

$$\gamma(h) = \sigma_z^2(x) - \text{Cov}(Z(x), Z(x+h)) , \quad (3.21)$$

where h is the lag distance between points.

The geostatistician calculates variogram values from the initial data (called the *experimental variogram*) for a small number of lags. Once the form of the covariance function has been chosen (e.g., the exponential model of Equation (3.3)), the sill and range are changed until the variogram model approaches the experimental variogram values in a least-squares sense. Finding appropriate parameters can be done either with numerical minimization of the least squares error or visual fitting with a graphical software package. Figure 3.3 shows an example of a variogram fitting. The experimental variogram values for 10 lag values are shown as circles. Two different

variogram models are shown. They have the same sill, but the dashed model has a longer range and appears to fit the experimental variogram better. Once model of the variogram has been established, the geostatistician can then transform the variogram into the covariance model using relation (3.21) and proceed with prediction.

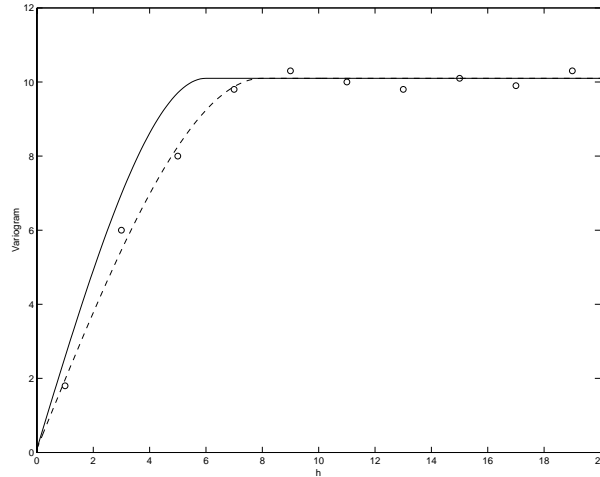


Figure 3.3: Example of variogram fitting for two covariance models of different range but identical sill. Experimental variogram points are shown as circles.

In short, the two approaches to model fitting are polar opposites. DACE modeling treats model fitting as a black-box, despite the lack of robustness, in order to create an automated fitting procedure. Geostatistics, however, choose a more labor-intensive, interactive approach in order to allow the user greater control over the fitting process.

3.6.2 Anisotropy

Anisotropy refers to situations where the spatial variability is a function of both distance and *direction*. Geostatistics accounts for two kinds of anisotropy: geometric and zonal [34]. In the former, the sill of the covariance function is the same everywhere, but the model has a larger range in some directions. The latter accounts for having variogram models that have different sills in different directions. DACE

modeling applies a very particular anisotropic model whereby different θ and p values are estimated for the different dimensions and a simple the product rule is applied.

3.6.3 Global search window

In general, model prediction requires solution of the *kriging system*, a set of linear equations:

$$\lambda_{SK} = \mathbf{r}_x^t \mathbf{R}^{-1} , \quad (3.22)$$

where λ_{SK} is the vector of so-called simple kriging weights. The size of the \mathbf{R} matrix is $n \times n$ where n is the number of sample points. In geostatistics, only the nearest neighbors of the point at which to predict are used in the kriging system. The *search window* determines how many neighbors to include and the maximum separation distance below which a sample point is included. Typically, a two-dimensional data set would use only 16 data points in the search window, making the inversion of the \mathbf{R} matrix much easier. As predictions are made around the design space, the search window shifts to include different sample points. Doing so also results in a local mean, $\hat{\beta}$, that varies throughout the design space to adapt to the local information by changing the \mathbf{R} matrix in Equation (3.12).

The practice in DACE modeling is to use a global search window. That is, *all* sample points are included in the kriging system of Equation (3.22) regardless of the prediction point location. This no longer requires a search for the nearest neighbors, but the resulting \mathbf{R} matrix is much larger and more difficult to invert. However, if many predictions are to be made using the same model, computational efficiency can be increased. The quantity $\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\hat{\beta})$ in Equation (3.11) is fixed because the same data are used everywhere, and therefore it can be pre-computed and stored with the kriging model. Overall, the DACE approach is more efficient

because it only requires evaluating $\mathbf{r}_\mathbf{x}$ at each prediction point. However, using the global search window means that the global trend function, $\mathbf{f}_\mathbf{x}^t\beta$, remains the same throughout the design space because \mathbf{R} is held fixed.

3.7 Improved DACE Modeling

The modeling practice of DACE could be greatly improved by incorporating some of the tools from the more general field of geostatistics. We investigate some minor modifications to the DACE approach to improve the robustness and flexibility of the resulting kriging models, some of which were suggested by Jones [42].

Some modifications were made to improve the conditioning of the \mathbf{R} matrix. As sample points begin to cluster around the optimum, rows of the \mathbf{R} matrix become very similar, resulting in a nearly singular matrix that must be inverted. One way to improve the conditioning of the matrix is to avoid values of $p = 2$ in the SCF of Equation (3.3). Such a model, referred to as the Gaussian model, is known to result in unstable kriging systems due to the behavior of the covariance model for short separation distances [72]. To improve the models, we limit the parameter range to $0 < p \leq 1.99$.

We have also applied a variety of other covariance models that are generally more stable than the Gaussian. The reader is referred to Lin et al. for a discussion of the differences between them [52]. There has been no consensus in the DACE community on the “best” covariance model to use in most situations. The results in this dissertation were gathered using the exponential model as described above.

3.7.1 MLE fitting revisited

In Section 3.6.1, it was noted that the log-likelihood function may behave monotonically, leading to poor model fitting. The phenomenon is illustrated using the

one-dimensional function shown in 3.4 that was created for this work. It is referred to as test function #1 throughout this dissertation and is defined as

$$f = -\sin(x) - \exp(x/100) + 10 , \quad (3.23)$$

with $x \in [0, 10]$.

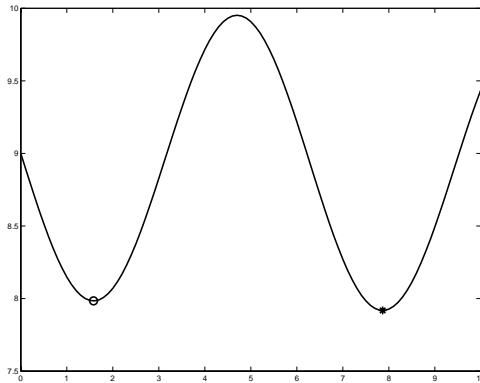


Figure 3.4: Test function #1

Two different sets of seven observations were taken and MLE was used to fit the models. Data set A consisted of seven equispaced points, whereas the inputs for data set B were located at $x = [0, 2, 4, 5, 6, 8, 10]$. Figure 3.5 shows that the model for the data set A (left) is quite erratic, while the data set B (right) produces a model that is almost identical to the original function (shown as a dashed line). Figure 3.6 plots the likelihood function of Equation (3.18) for a range of θ values. For this example, p was held fixed, making Equation (3.18) a function of θ only. For data set A, the function monotonically decreases with θ , leading to an “optimal” value θ_* that hits the upper limit on θ . With such a large θ_* value, the kriging model quickly degenerates to the global constant trend, $\hat{\beta}$ (the horizontal line in the graph), producing the erratic behavior. However, for data set B, the likelihood function has an interior optimum. The resulting model parameter leads to a much smoother kriging model.

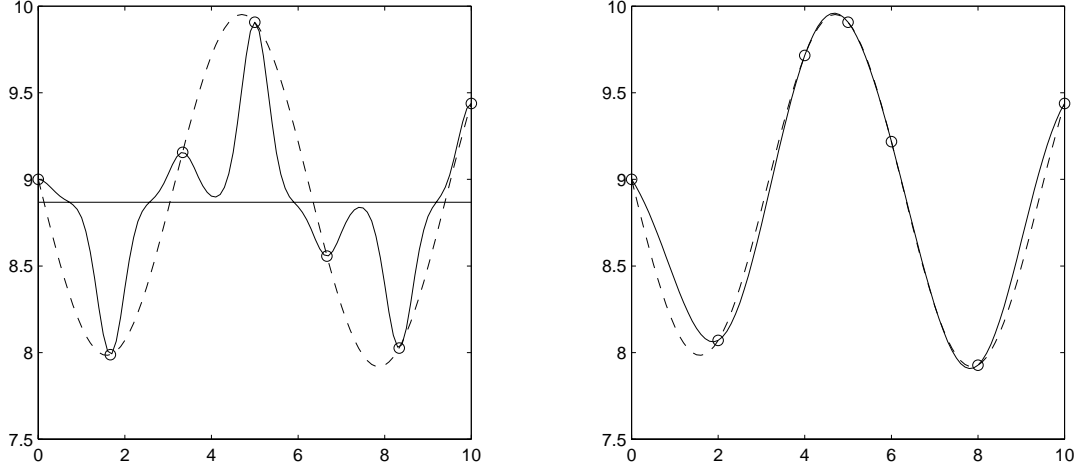


Figure 3.5: Maximum likelihood fitting for data set A (left) and B (right). The plot shows the kriging models (solid line), the true function (dashed line), and the data (circles). The horizontal line in the left graph is the global trend of the kriging model.

This begs the question, what can one do to detect and prevent the kind of model generated by data set A? One diagnostic would be to determine if the likelihood function was in fact monotonic. Further examination of Equation (3.18) reveals that the log-likelihood is comprised of two additive terms. As θ increases, the \mathbf{R} matrix approaches the identity matrix (no correlation). In this case, the second term drops out, leaving $-n \log(\hat{\sigma}_z^2)$ (the factor of 0.5 was left out). But for the case of no correlation, the generalized least squares estimate of σ_z^2 degenerates to the least squares estimate of the variance in the data, $\text{Var}(y)$. This signals that the log-likelihood function will eventually reach a plateau with a value of $-n \log(\text{Var}(y))$ for a large enough theta. One can therefore examine if following is true:

$$MLE(\theta_*, p_*) < -n \log(\text{Var}(y)) , \quad (3.24)$$

i.e., the optimal value of the log-likelihood function found during fitting is below the plateau value. If not, then there is a good chance that the likelihood function was monotonic and that the resulting model is a poor fit. One should apply this metric

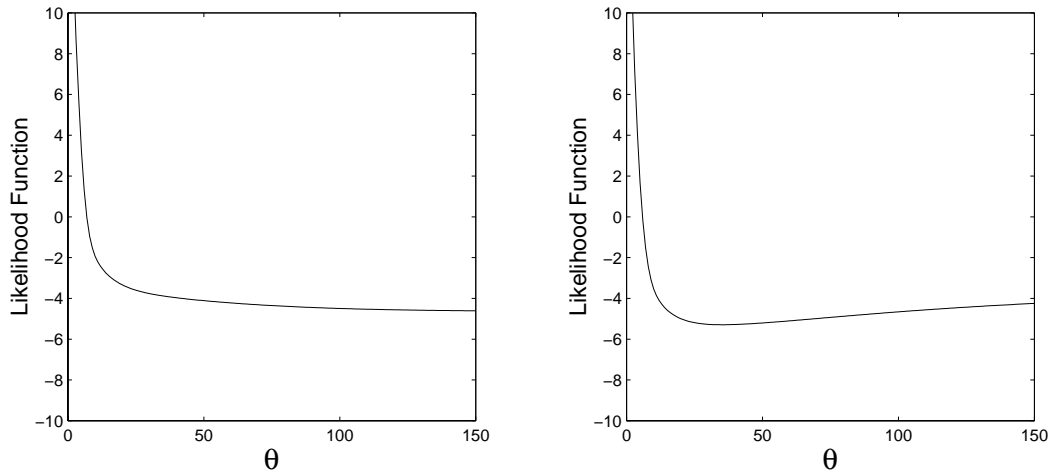


Figure 3.6: Likelihood function for data sets A (left) and B (right).

rather than simply checking if θ_* was at its upper limit because the numerical routine for finding the minimum of the function may result in an MLE value on the plateau, but with a θ_* value below the upper limit.

For data set A, one can verify that the $\text{MLE}(\theta_*)$ was indeed above the plateau value $-n \log(\text{Var}(y))$. To remedy the situation, we attempt to “convince” the MLE fitting process that there is indeed some correlation in the data by adding the data point $x = 0.1$, which is near an existing sample point. We refer to this data sample as set C. Having two neighboring sample points should improve upon the accuracy of the short-range variability. That is, the strong correlation between the two points should increase the acceptability of models with a smaller theta values (i.e., correlation that takes longer to decay). To verify that the success is not an artifact of having an extra sample point, another data set (set D) is created by moving one of the data points next to an existing point. Specifically, the data point at $x = 6.67$ was replaced by one at $x = 5.1$, near the point $x = 5.0$.

In Figure 3.7 we show the success of MLE fitting for data sets C and D. In both cases, the resulting kriging model is much more accurate than the original fit from

data set A. Figure 3.8 shows the associated likelihood functions. Notice that the optimal likelihood value is indeed below the plateau at $-n \log(\text{Var}(y))$, that is, the fitting function is not monotonic. This example illustrates another way in which the reliability of the MLE fit can be quickly assessed. Failure to find an optimal point below the plateau may lead the designer to sample an additional point nearby an existing sample to reduce the possibility of a monotonic likelihood function. While this approach is not full-proof, it does provide a simple means for checking a situation known to result in poor model fitting as well as a means to attempt to correct it.

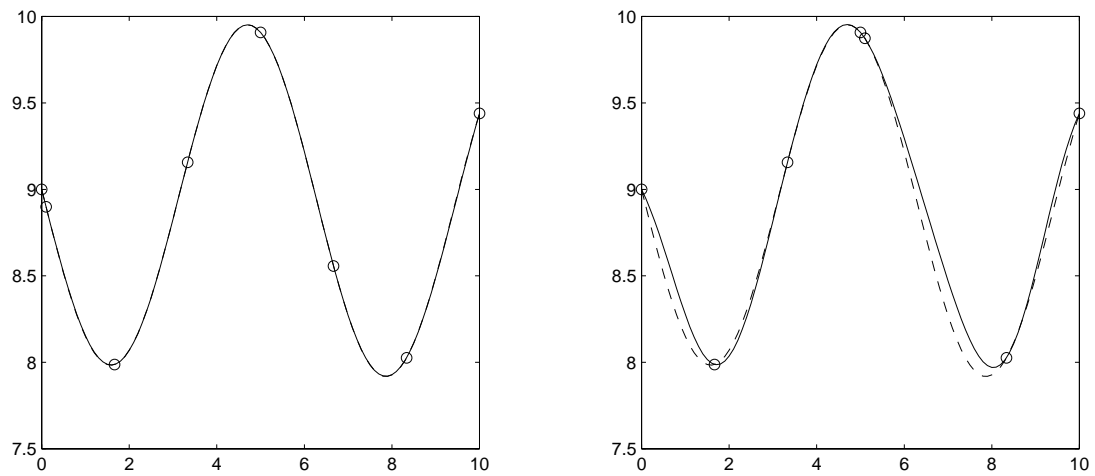


Figure 3.7: Maximum likelihood fitting for data set C (left) and D (right). The plot shows the kriging models (solid line), the true function (dashed line), and the data (circles).

3.7.2 Non-interpolating kriging models

The approach to kriging described above results in an interpolating surface. That is, at the sampled data locations, the predictor goes exactly through the data, and the kriging variance returns to zero. While this is appropriate for many engineering applications, there are cases where a non-interpolating version of the kriging model would be very beneficial – for example, when the data are highly noisy.

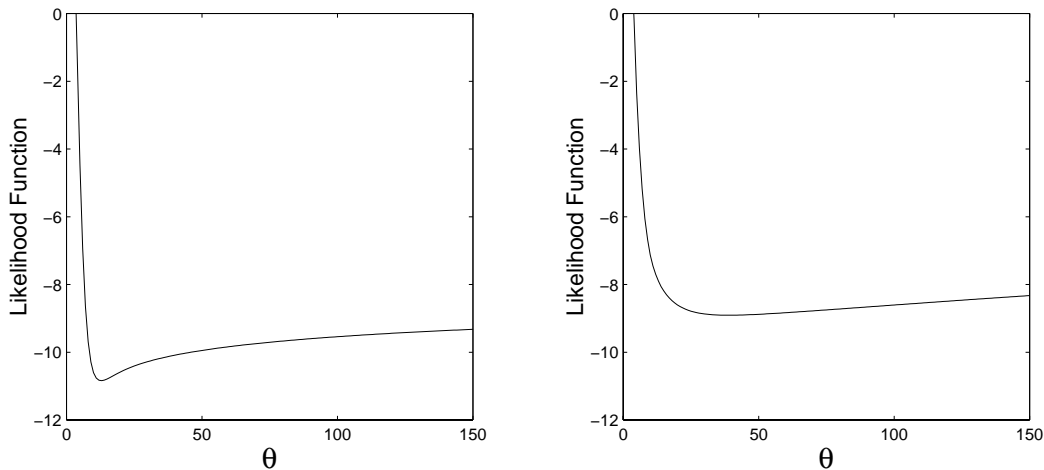


Figure 3.8: Likelihood function for data sets C (left) and D (right).

To allow kriging to smooth the data, an additional parameter is introduced into the SCF of Equation (3.4) to account for the measurement error, $E(\mathbf{x})$. The kriging model now takes the form $Y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}) + E(\mathbf{x})$, and the SCF appears as

$$R(\mathbf{w}, \mathbf{x}) = (1 - \text{nugget}) \prod_{k=1}^d e^{-\theta^k |w^k - x^k|^{p^k}}, \quad (3.25)$$

with $0 \leq \text{nugget} < 1$. The name refers to the *nugget effect* model used in geostatistics to describe the discontinuity of the covariance function as $|w - x|$ approaches zero. The scaling factor $(1 - \text{nugget})$ is the same as the ratio $\frac{\sigma_z^2}{\sigma_z^2 + \sigma_E^2}$ defined in the DACE literature [77], [78]. In either case, the additional parameter ranges between 0 and 1 and must be fit via MLE along with θ and p . We prefer the notation shown in Equation (3.25) simply because it is easier to view the value of *nugget* which is usually quite close to zero (e.g., 1e-5) rather than $\frac{\sigma_z^2}{\sigma_z^2 + \sigma_E^2}$ which is usually quite close to 1 (e.g., 0.99999). Typically, a single scaling factor is applied to the covariance model. It is, however, possible to fit a separate *nugget* parameter for each dimension as

$$R(\mathbf{w}, \mathbf{x}) = \prod_{k=1}^d (1 - \text{nugget}^k) e^{-\theta^k |w^k - x^k|^{p^k}}, \quad (3.26)$$

so that more smoothing is applied in some directions than others.

A *nugget* value of zero means that the predictor goes exactly through the data. Increasing the *nugget* value results in smoothed models because collocated points no longer have perfect correlation. The *nugget* parameter can also be used to improve the stability of the kriging model. When superEGO is used for design optimization, points begin to cluster around the optimum (e.g., Figure 1.3(d)), making the \mathbf{R} matrix nearly singular and difficult to invert. In our implementation, a small nugget effect (e.g., 10^{-12}) is used to improve the conditioning without noticeably smoothing the data.

It is important to note that the scaling factor (*1-nugget*) is *not* applied to the diagonal of the \mathbf{R} matrix, and *is* applied to each element of the \mathbf{r}_x vector even when a prediction point coincides with a sample point. This is done because a point is perfectly correlated with itself (diagonal of \mathbf{R} matrix consists of ones), but would not be perfectly correlated with a second sample point at the same location (\mathbf{r}_x vector) in the presence of measurement error. An example of the resulting kriging system of equations $\mathbf{R}\lambda = \mathbf{r}_x$ is shown in Equation (3.27) for a set of three data points, where the point at which to predict coincides with data point 1.

$$\begin{bmatrix} 1 & \eta R_{12} & \eta R_{13} \\ \eta R_{12} & 1 & \eta R_{23} \\ \eta R_{13} & \eta R_{23} & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \eta \\ \eta R_{12} \\ \eta R_{13} \end{bmatrix}, \quad (3.27)$$

where λ_i is the kriging weight for the i^{th} data point, R_{ij} is the (interpolating) correlation between points i and j computed from Equation (3.4), and $\eta = (1 - \text{nugget})$ or $\eta = \frac{\sigma_z^2}{\sigma_z^2 + \sigma_E^2}$ depending upon the choice of notation for the scaling parameter. The solution to this system of equations appears in the kriging prediction formula of Equation (3.11) as $\mathbf{r}_x^t \mathbf{R}^{-1}$. The influence of the scaling parameter results in an approximation that does not pass through the data. However, if one were to *not* apply

the scaling parameter to the first element of $\mathbf{r}_\mathbf{x}$ where the prediction point coincides with data sample 1 (i.e., $r_1 = 1$), then the resulting approximation be smooth everywhere except at each sample point where it would have a discontinuous “spike” to the sampled value. For this reason, the scaling parameter is applied to each element of the $\mathbf{r}_\mathbf{x}$ even if it coincides with a sample point.

A one-dimensional example is shown to demonstrate the smoothing effect. A uniformly distributed random number between $\pm\frac{1}{2}$ is added to the quadratic function $y = (x - 5)^2$, $x \in [0, 10]$. Nineteen sample points are taken: 11 equally spaced points and 8 additional points near the minimum. Kriging models are fit to $[\theta, p]$ for the interpolating and $[\theta, p, \textit{nugget}]$ for the smoothing covariance models of Equations (3.4) and (3.25), respectively. Figure 3.10 shows the predicted value of the interpolating (solid line) and smoothed (dashed line) kriging models. In the area where the noise is prominent, the interpolating model oscillates in order to pass through each of the data samples, whereas the other model is much smoother in a least squares sense.

Figure 3.11 shows the kriging variance. For both the interpolating and smoothed models, the variance is lower in the middle because more data are available in that region. A consequence of the smoothing effect is that the variance model no longer returns to zero at the sample points. This is because the smoothing effectively increases the uncertainty in the model everywhere. In this example, there is little difference between the variance of the smoothed model at the sample points and the variance in locations far from the data. Because of this feature, one must be cautious when using Equation (3.14) as a measure of the uncertainty in smoothed kriging models. It is recommended that the interpolating covariance model of Equation (3.4) rather than Equation (3.25) be used in conjunction with Equation (3.14) for a more useful estimate of the model uncertainty.

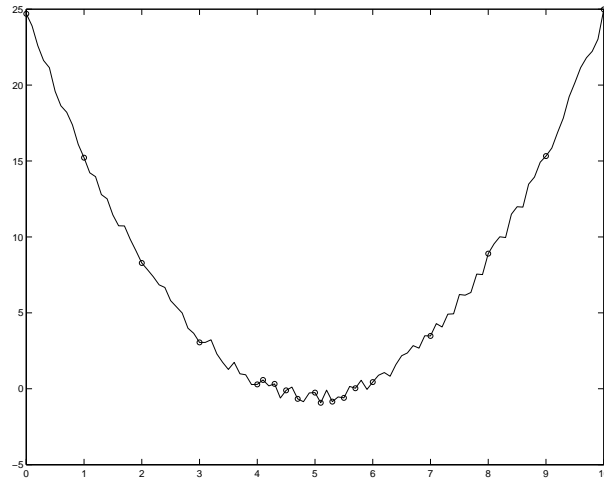


Figure 3.9: Noisy test function (data points are shown as circles)

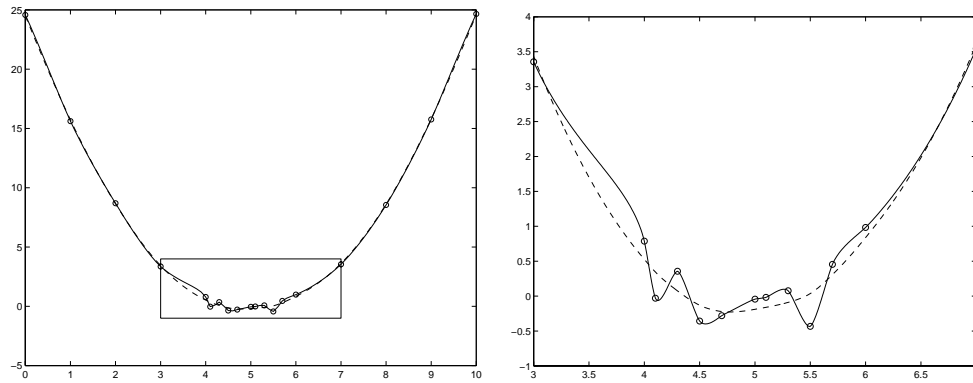


Figure 3.10: Kriging predictions for interpolating (solid) and smoothed (dashed) models (left). A close up of the boxed region is shown at right. Data points are shown as circles

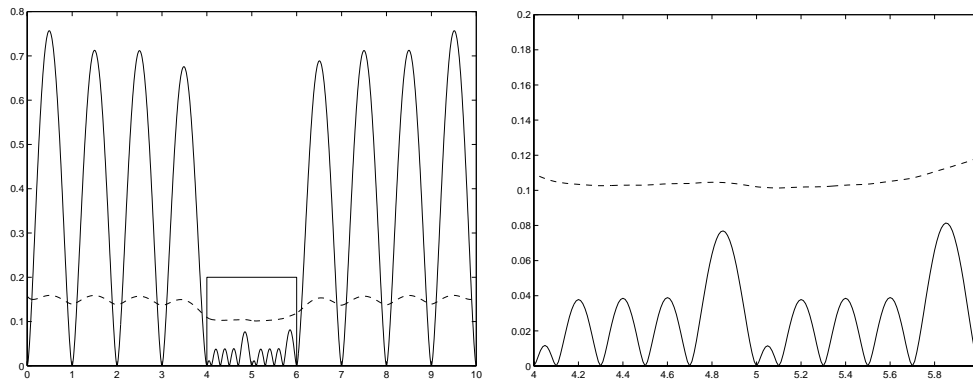


Figure 3.11: Kriging variance for interpolating (solid) and smoothed (dashed) models (left). A close up of the boxed region is shown at right.

3.8 Chapter Summary

This chapter has developed the equations for fitting a kriging model and predicting the corresponding mean and variance. The approach used in this work follows the DACE literature, but incorporates some enhancements. Specifically, an additional parameter was introduced into the covariance model in order both to improve the stability of the kriging system and to enable the non-interpolating kriging models. This version of kriging performs very well for smoothing over noisy data, but the variance models do not tend towards zero at the sampled data points if the smoothing effect is large. It was also demonstrated that the DACE procedure of MLE fitting is not robust, but has the advantage of full automation. Some efforts were made to detect and correct for poor model fits.

In the next chapter, we begin to use the kriging models with superEGO to explore a series of infill sampling criteria on unconstrained optimization test problems.

CHAPTER 4

Infill Sampling Criteria

The choice of infill sampling criterion is what differentiates most Bayesian analysis optimization algorithms. Moreover, the vast majority of them use criteria that search for the minimum of the approximate model, the location of maximum uncertainty, or some compromise between the two. They do, however, vary in the quantification of these ideas. In this chapter, several such criteria are compared on a set of unconstrained analytical test functions.

4.1 A Taxonomy of Criteria

In this section, a number of sampling criteria and their formulae are presented. Although many of them were developed independently, they often reveal striking similarities. We begin by describing Kushner's criterion [51] followed by the expected improvement function and variations thereof [53], [58], [79], [83]. The next criterion is the lower confidence bounding function of Cox and John [21]. We then describe three criteria proposed by Watson and Barnes for the field of geostatistics [99]. The three metrics - locating the threshold-bounded extreme, locating the regional extreme, and minimizing surprises - are abbreviated here as WB_1 , WB_2 , and WB_3 , respectively. Next, we describe the maximum variance criterion. Finally, we propose a hybrid

approach, the *switching* criterion.

It is necessary first to clarify some notation that will be used for many of the criteria below. Let f_{min} be the best sampled function value after n evaluations of the function $y = f(\mathbf{x})$. For simplicity, we will leave out the dependence on \mathbf{x} in this section, notating the predicted value $\hat{y}(\mathbf{x})$ as \hat{y} and the variance $\hat{\sigma}^2(\mathbf{x})$ as $\hat{\sigma}^2$. Also, the notational simplification

$$z = \frac{f_{min} - \hat{y}}{\hat{\sigma}} , \quad (4.1)$$

is used throughout the chapter and should not to be confused with the $Z(x)$, the Gaussian process of Equation (3.1).

Throughout this section, the behavior of the various criteria is illustrated using test function #1 shown in Equation (3.23) (see Figure 4.1). The w-shaped dashed line is the true objective function we wish to minimize, while the solid line is the kriging approximation conditional to the sample points shown as circles. The plot at the bottom is the sampling criterion, normalized to facilitate comparisons between the various criteria. Higher values for the criterion in these plots are better, but it is important to note we've also changed the sign because the ISC subproblem is a minimization problem.

4.1.1 Kushner's criterion

The first criterion, originally proposed by Kushner [51], is to maximize the probability of improving upon the best point, f_{min} by some amount ϵ , quantified as

$$\begin{aligned} \text{Kushner} &= P(\hat{y} < f_{min} - \epsilon) \\ &= \Phi \left(\frac{(f_{min} - \epsilon) - \hat{y}}{\hat{\sigma}} \right) , \end{aligned} \quad (4.2)$$

where $\Phi(\cdot)$ is the Gaussian cumulative distribution function (cdf). The value ϵ is a parameter that may either be chosen by the user, or used at its default value of

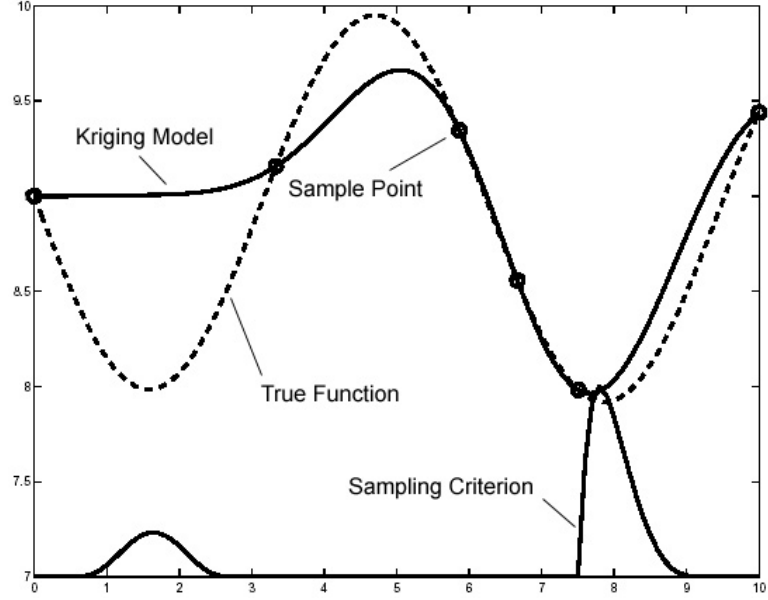
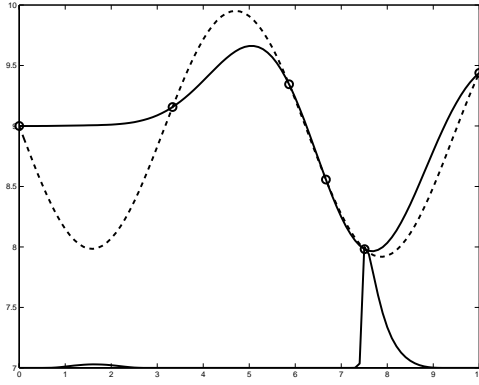
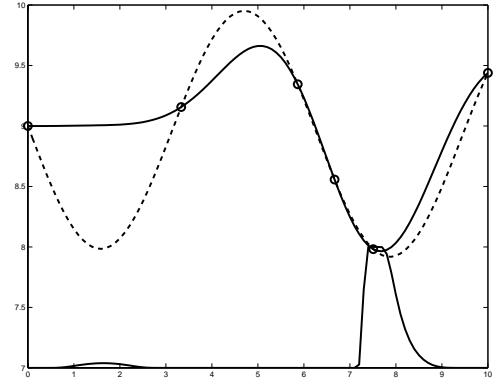


Figure 4.1: Example of ISC plot for comparisons. Dashed line is the true function, solid line is the approximation, circles are the sample points. The plot at the bottom is the sampling criterion.

0.1% of f_{min} . To demonstrate the effect of ϵ on the criterion, two plots are shown in Figure 4.2 for $\epsilon = 0.1\%$ of f_{min} and $\epsilon = 1\%$ of f_{min} . Note that the lower value of ϵ leads to a more local searching criterion because decreasing the numerator of the cdf tends to make the peak of the probability metric sharper in promising regions. Larger ϵ values flatten out the probability curve, making it wider with a lower peak value, but that effect is not as noticeable in Figure 4.2 because each ISC has been normalized in the plots.

There are potential problems with this criterion if a wide region results in identical P values. The criterion cannot distinguish between any of the candidate sampling locations in the region, although some would be intuitively more beneficial to sample than others based upon how redundant they are with existing sample locations. The flatness of the criterion may also pose numerical problems if gradient-based algorithms are used to solve the ISC subproblem. The methods used here however

(a) $\epsilon = 0.1\%$ of f_{min} (b) $\epsilon = 1\%$ of f_{min} Figure 4.2: Kushner's criterion for two values of ϵ

are derivative-free.

4.1.2 Expected improvement

As mentioned in the Chapter 1, the ISC used by EGO is known as the expected improvement function. It was developed in part to overcome the difficulty of having to choose a value for ϵ in Kushner's criterion. The improvement over the current best point is defined as

$$I = \max\{0, f_{min} - \hat{y}\} . \quad (4.3)$$

The expected value of the improvement (i.e., the *expected improvement*) is computed as

$$EI = \begin{cases} (f_{min} - \hat{y})\Phi(z) + \hat{\sigma} \phi(z), & \text{if } \hat{\sigma} > 0 \\ 0, & \text{if } \hat{\sigma} = 0 \end{cases}, \quad (4.4)$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function and the cumulative distribution function of the standard normal distribution, respectively, and z is defined in Equation (4.1).

Inspection of the expected improvement function reveals two important trends.

The first term in Equation (4.4) is the difference between the current minimum and the predicted value multiplied by the probability that $Y(\mathbf{x})$ is smaller than f_{min} and is therefore large where \hat{y} is likely smaller than f_{min} . The second term tends to be large where there is high uncertainty about whether or not \hat{y} will be better than f_{min} . Thus, the expected improvement is large for both regions of likely improvement and regions of high uncertainty. As the prediction variance vanishes at the sampled data points, the expected improvement also vanishes.

4.1.3 Generalized expected improvement

The generalized form of the expected improvement introduces a non-negative integer parameter g , as

$$I^g = \max\{0, (f_{min} - \hat{y})^g\}. \quad (4.5)$$

The resulting recursive formula for the generalized expected improvement is

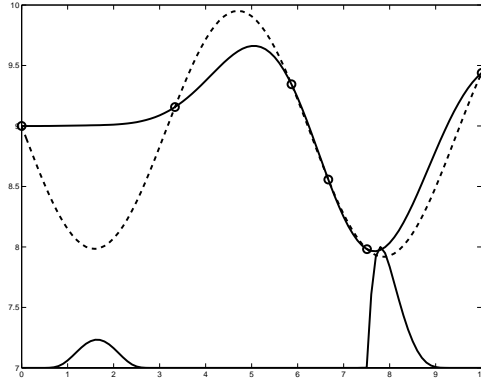
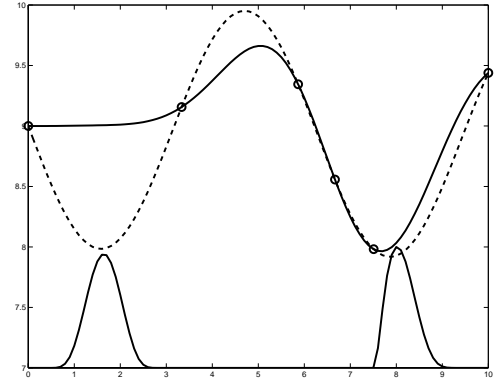
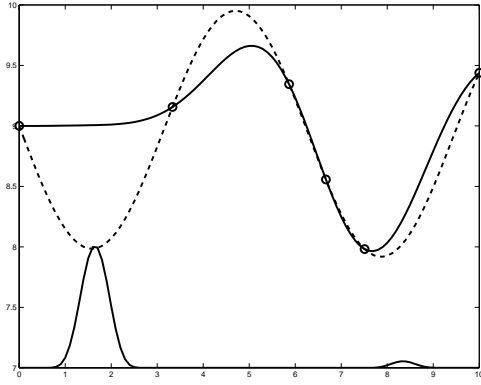
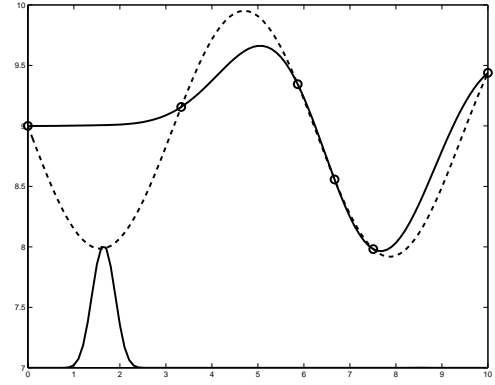
$$E(I^g) = s^g \sum_{k=0}^g (-1)^k \left(\frac{g!}{k!(g-k)!} \right) z^{g-k} T_k, \quad (4.6)$$

where

$$T_k = -\phi(z) z^{k-1} + (k-1)T_{k-2} \quad (4.7)$$

starting with $T_0 = \Phi(z)$ and $T_1 = -\phi(z)$. For a value of $g = 1$, the criterion degenerates to the ordinary expected improvement function of Equation (4.4).

The impact of the parameter g in Equation (4.6) is shown in Figure 4.3. For $g = 1$ (Figure 4.3(a)), the EI function rises significantly in two areas. The region on the left is sparsely sampled, leading to high model uncertainty, while the region on the right is where the probability of finding a better objective function value is high. For $g = 2$ (Figure 4.3(b)), the criterion places as much emphasis on the region of high uncertainty as it does the region of high likely improvement. For $g = 10$

(a) $g = 1$ (b) $g = 2$ (c) $g = 5$ (d) $g = 10$ Figure 4.3: Expected improvement criterion for different g values

(Figure 4.3(d)), the EI function rises only in the left region where model uncertainty is high. In summary, increasing the value of g shifts emphasis towards global search by giving more importance to the second term in Equation 4.4.

4.1.4 Cool criterion

Although the value used for g is extremely significant, there is no obvious way to select it. Too high of a value could prevent EGO from converging on a good solution in a reasonable amount of time. Too low of a value could lead EGO to overlook the global optimum as it searches too locally. By analogy to the Simulated Annealing

Table 4.1: Cooling schedule for the Cool criterion

Iteration	g value
1 - 4	20
5 - 6	10
7 - 9	5
10 - 11	2
12 - 14	1
≥ 15	0

algorithm, which starts searching globally then refines the search more locally as iterations continue, we propose starting with a large g value, reducing the value towards zero. The heuristic cooling schedule used in this chapter is shown in Table 4.1. The use of the EI function with this schedule is referred to as the *Cool* criterion. Note that this is akin to progressively decreasing the parameter ϵ as proposed by Kushner [51].

4.1.5 Lower confidence bounding function

As described in Chapter 2, Cox and John’s SDO algorithm employs a linear combination of the uncertainty in the model and its predicted value. The so-called lower confidence bounding (lcb) function appears as:

$$lcb = \hat{y} - b\hat{\sigma}, \quad (4.8)$$

where b is a user-defined parameter. Cox and John report results for $b = 2$ and $b = 2.5$. Recalling that the lcb function is to be minimized, locations that have either low function values or high uncertainty are favored by Equation (4.8). Increasing b increases the emphasis on high uncertainty, thereby forcing a more global searching strategy. Plots of the lcb function for two b values are shown in Figure 4.4. There is only a slight difference in the resulting ISC plots. However, the lcb function for $b = 2$ would indeed choose the next iterate in the local improvement region on the

right, whereas it would choose the high uncertainty region on the left for $b = 2.5$.

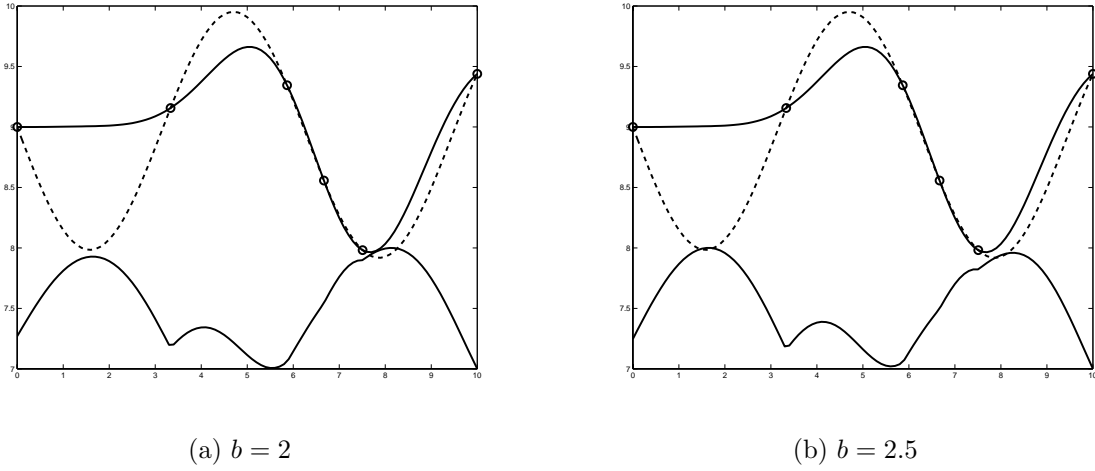


Figure 4.4: Lower confidence bounding criterion for two values of b

4.1.6 WB_1 : Locating the threshold-bounded extreme

This criterion was developed in the context of contamination testing where the objective was to locate points that maximize the probability that at least one of the infill samples exceeds some specified threshold. This is exactly analogous to Kushner's approach. We extend Watson and Barnes' concept to design optimization by using f_{min} as the threshold. The objective is now to maximize the probability of being no greater than f_{min} , which is computed as

$$WB_1 = \Phi \left(\frac{f_{min} - \hat{y}}{\hat{\sigma}} \right). \quad (4.9)$$

Notice that this formulation is exactly Kushner's criterion for $\epsilon = 0$, or the generalized EI function for $g = 0$. It is therefore extremely local in its search, and one must be confident that the model has found the region of the optimum for this criterion to be successful.

The behavior of this function is illustrated in Figure 4.5(a). The large peak on the right is because the cdf of a normal distribution is strictly increasing, making

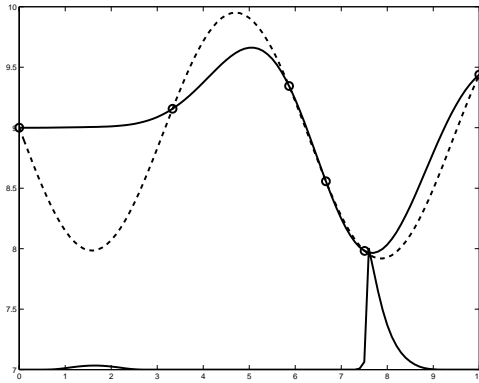
WB_1 largest for positive quantities, i.e., when the predicted value is smaller than the current best point. The smaller peak in the undersampled region on the left is due to the large uncertainty, where the large $\hat{\sigma}$ lowers the magnitude of the negative argument of the cdf, thereby increasing the value of WB_1 .

4.1.7 WB_2 : locating the regional extreme

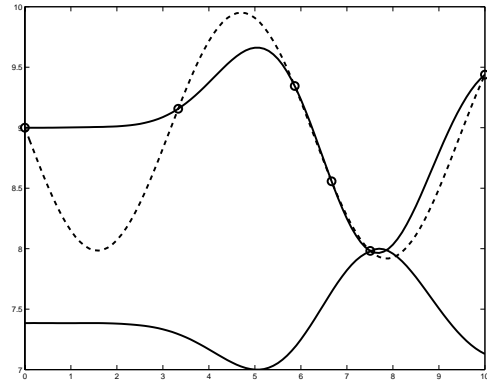
The next criterion attempts to minimize the expected value of the smallest observation once the infill samples have been added:

$$WB_2 = \begin{cases} \hat{y} + (f_{min} - \hat{y}) \Phi(z) + \hat{\sigma} \phi(z), & \text{if } \hat{\sigma} > 0 \\ 0, & \text{if } \hat{\sigma} = 0 \end{cases}. \quad (4.10)$$

The resulting formula is remarkably similar to the EI function in Equation (4.4). The only difference is the additional first term, which is the predicted value at the location of interest. This criterion thus gives slightly more weight to local search than does the EI function. It is also smoother since it does not return to zero at the sampled points, see Figure 4.5(b). This appealing trait is noteworthy, as it may help locate the maximum of the criterion.



(a) WB_1



(b) WB_2

Figure 4.5: Sampling criterion for WB_1 and WB_2

4.1.8 WB_3 : minimizing surprises

The third Watson and Barnes criterion aims to minimize the maximum probability that a true value deviates significantly from its predicted value. Watson and Barnes use the simplified expression

$$WB_3 = \min_{\mathbf{x}} \max_{\mathbf{v}} \text{Var}[Y(\mathbf{v})|\mathbf{S} \text{ and } \mathbf{x}], \quad (4.11)$$

where \mathbf{x} is the candidate infill sample point of interest, \mathbf{v} is a generic location in the design space, and $Y(\mathbf{v})$ is the random variable at the unobserved location \mathbf{v} . Note that the variance is conditional to both the sample set, \mathbf{S} , and the candidate infill sample, \mathbf{x} . One can compute the variance of $Y(\mathbf{v})$ because the updated variance (i.e., the variance of the model once the candidate infill samples have been added) does not depend on data values, but is a function of only their locations and a given covariance function. Evaluating WB_3 for a given candidate location requires locating the maximum variance. This minimax problem within the original design optimization problem adds significantly to the total run time. Thus it is best suited for problems where the objective function is extremely expensive to calculate.

A characteristic of the updated variance of $Y(\mathbf{v})$ is that it becomes constant if the distance between \mathbf{v} and the closest sample point exceeds the range of correlation. As a consequence, the profile of WB_3 values in Figure 4.6(a) is flat on the left side, which is undersampled. Also, adding infill samples far from areas that are not covered will not reduce the maximum variance. These characteristics can cause serious difficulties in locating the maximum of WB_3 . For the above reasons, WB_3 is not considered further in this chapter.

4.1.9 Maximum variance

Because of the inherent difficulty in solving the minimax problem of WB_3 , a simpler measure of uncertainty is needed. The *Maxvar* criterion uses the variance at the candidate location from Equation (3.14) and is to be maximized, see Figure 4.6(b). While WB_3 has the benefit of looking ahead to locate regions of high uncertainty in the *next* iteration, Maxvar is much more reliable and easy to compute. In the example, WB_3 and Maxvar would choose the next iterate at approximately the same location, lending further support for the dismissal of WB_3 due to its inefficiency.

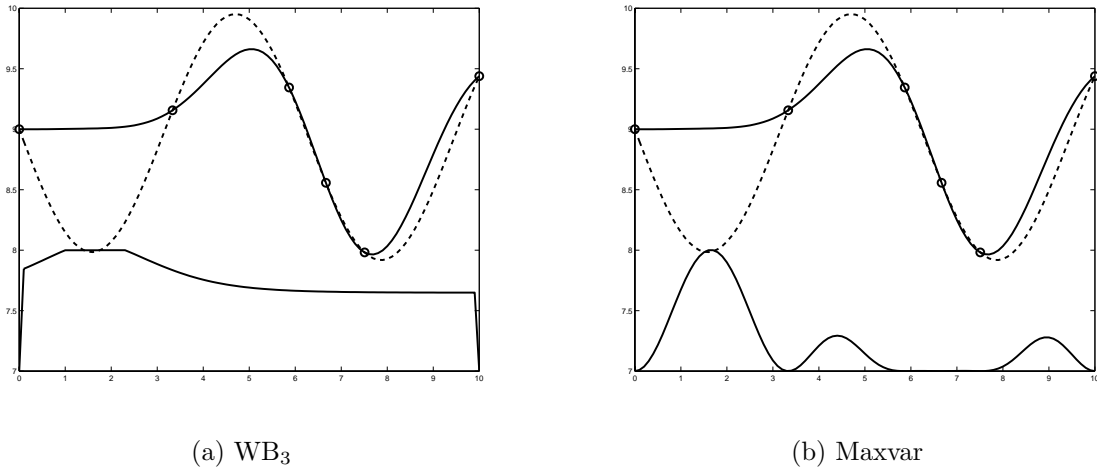


Figure 4.6: Sampling criterion for WB_3 and Maxvar

4.1.10 Switching criterion

The final criterion described here is referred to as the *switching* criterion. In all of the criteria described above except the last two, the selection process relies on some compromise between two goals: exploring regions likely to produce local improvements and sampling regions of high uncertainty. At each iteration, the above criteria may find a point that is a good compromise between the two goals, but does not excel at either one. We propose here that there is an advantage to *not*

compromising. Instead, the switching criterion alternates between improving local optima and sampling in regions of high variance as iterations proceed. The search strategy is somewhat similar to the two-phase algorithm proposed by Locatelli and co-authors [53] or Žilinskas' P*-algorithm [104].

For the results shown in this chapter, the criterion alternated in the following pattern. Five iterations were performed using the Maxvar criterion. Then, the algorithm switched to a criterion that searched for the minimum point on the approximate model. Once three consecutive points were sampled within 0.1% of the design space range of each other, it considered a local optimum as being achieved. It then switched back to five iterations of Maxvar, and the cycle repeated. This pattern was chosen on a purely subjective basis for the purposes of these examples.

4.2 Analytical Examples

The impact of each of the above criteria on superEGO is assessed using three unconstrained analytical examples. Test problems with two design variables were chosen for better visualization.

4.2.1 Example 1: Mystery function

The first example is a multimodal function in two dimensions with three local optima, one of which is the global minimum. The origins of this function are unknown to this author, hence the name *mystery function*.

$$f = 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7 \sin(0.5x_1) \sin(0.7x_1x_2) , \quad (4.12)$$

defined over the range $x_i \in [0, 5], i = 1, 2$. The global solution has a value of -1.4565 at $\mathbf{x} = [2.5044, 2.5778]$. Figure 4.7 shows the mesh and contour plots of the function. The global solution is shown as an asterisk on the contour plot.

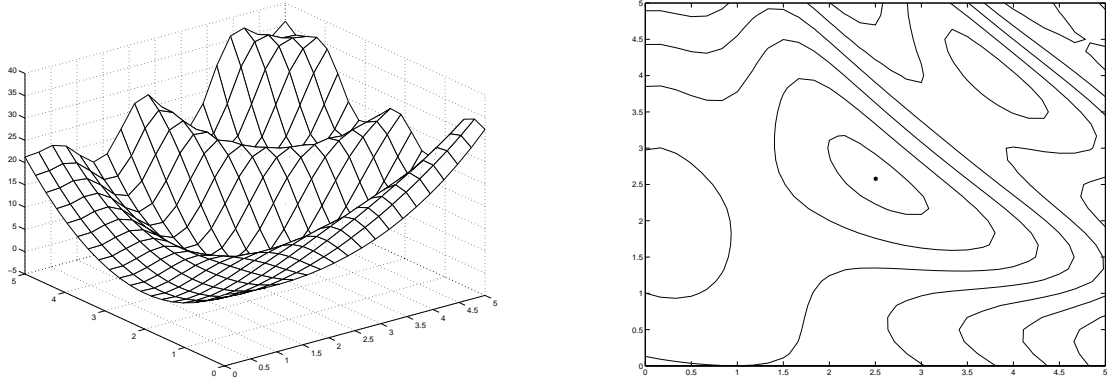


Figure 4.7: Mesh and contour plots of Example 1: mystery function

4.2.2 Example 2: Branin function

The second example is the Branin test function [25], defined as

$$f = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10 \quad (4.13)$$

with $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. The three global minima at $\mathbf{x} = [3.1416, 2.2750]$, $\mathbf{x} = [9.4248, 2.4750]$ and $\mathbf{x} = [-3.1416, 12.2750]$ shown as asterisks in Figure 4.8(b) have identical function values of 0.3979.

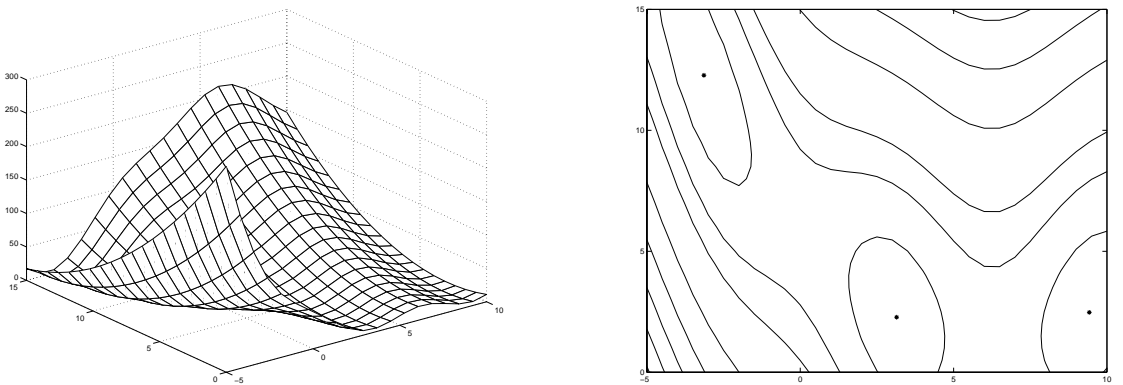


Figure 4.8: Mesh and contour plots of Example 2: Branin function

4.2.3 Example 3: Six hump camelback function

The third example is the six hump camelback function [37], defined as:

$$f = (4 - 2.1x_1^2 + x_2^3/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2, \quad (4.14)$$

$x \in [-10, 10], i = 1, 2$. It has six local minima (hence the name of the function), two of which are global. The solutions occur at $\mathbf{x} = [-0.0898, 0.7127]$ and $[0.0898, -0.7127]$ with a value of $f = -1.0316$. The original function yields values five orders of magnitude higher in regions away from the local optima. To make modeling such a function practical, we solved the problem on the subregion $x_1 \in [-2, 2]$ and $x_2 \in [-1, 1]$ which still includes all six local optima. The plots shown in Figure 4.9 are of that subregion.

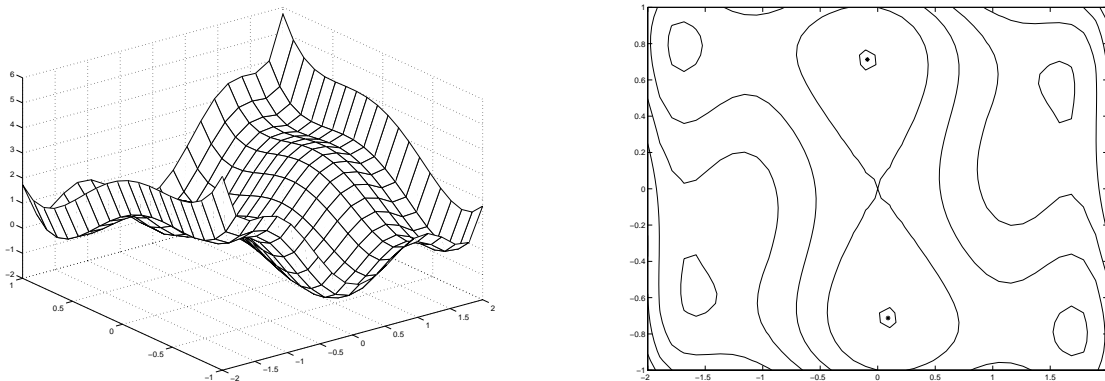


Figure 4.9: Mesh and contour plots of Example 3: six hump camelback function

4.3 Methodology and Comparison Metrics

In order to isolate the impact of the infill criteria for the analytical examples, it was necessary to reduce the influence of model accuracy. Thus for a given example, the same kriging model parameters were used for each criterion and for all iterations. Typically, the kriging model parameters θ , p , and *nugget* are fit at each iteration via

maximum likelihood estimation (MLE). In preliminary testing, it was observed that cross-validation fitting often provides more accurate kriging models than MLE fitting [80]. Thus a set of observations collected from previous experimentation was used to find good covariance model parameters by minimizing the Root Mean Squared error of cross-validation. While cross-validation was more expensive, no further model fitting was required during optimization. The reduced overhead more than made up for the initial computational costs.

In this work, the optimum of the infill sampling criteria was found via the DIRECT algorithm due to Jones [43]. It does not use gradient information and has global search properties that help it avoid the local optima inherent to the infill criteria. For the examples here, 50 iterations of DIRECT were used, requiring on the order of 300 evaluations of the infill criteria for each iteration of superEGO. More information on the search strategy can be found in Section 7.1.

Comparisons of the infill sampling criteria are difficult because there is no rigorous convergence criterion for superEGO. Schonlau proposed stopping the search once the ratio of the expected improvement to the current best sample value becomes sufficiently small [83]. Because this stopping rule has no meaning for the other criteria studied here, each optimization is stopped after 100 function calls have been made, and the following metrics are computed:

- $f_{1\%}$ *metric*: The number of iterations required before a point is sampled with an objective function value within 1% of the true solution.
- $x_{1\%}$ *metric*: The number of iterations required before a point is sampled within a box the size of $\pm 1\%$ of the design space range centered around the true solution.
- x_* *metric*: The Euclidean distance from the best sample point to the global solution, \mathbf{x}_* .

- *RMS metric*: The global modeling error. After 100 evaluations, the metamodel is compared to the true function on a 30 by 30 gridded set of locations. The resulting errors at $N = 900$ points are summarized by the RMS error, calculated as

$$RMS = \frac{1}{N} \sqrt{\sum_{i=1}^N (\hat{y}(\mathbf{x}_i) - f(\mathbf{x}_i))^2}.$$

The first two metrics measure how efficiently the algorithm finds the solution, while the third one measures how accurately it finds the solution. The last metric evaluates how accurately the final metamodel approximates the response over the entire design space. For all metrics, lower values are better.

To test the robustness of the results, 10 optimizations were performed for each criterion using a different set of 10 initial sample points. The same set of initial points was used across the different criteria. There were 11 criteria evaluated on three examples, optimized 10 times each – 330 optimizations in all. For each criterion, we calculated the average value, standard deviation and median for each metric. For cases where the algorithm was unable to sample a point within the $f_{1\%}$ or $x_{1\%}$ limits, the metric was assigned a value of 90, the maximum number of iterations allowed.

4.4 Results

A total of 11 sampling criteria were compared for the three examples. The abbreviations used for each is summarized in Table 4.2.

Tables 4.3 through 4.5 show the median value of the performance metrics over the 10 optimization runs for each example. In some instances, a criterion performed well for most of the 10 optimizations of a particular example, but was unable to satisfy the $f_{1\%}$ or $x_{1\%}$ limits in one or two cases. This significantly skewed the average values of the metrics because only 10 optimizations were performed per criterion per

Table 4.2: List of criteria compared

Abbrev.	Description
EI ₁	expected improvement function
EI ₂	generalized expected improvement for $g = 2$
Cool	generalized expected improvement with cooling schedule Table 4.1
K _{0.1%}	Kushner's criterion for $\epsilon = 0.1\%$ of f_{min}
K _{1%}	Kushner's criterion for $\epsilon = 1\%$ of f_{min}
lcb ₂	lower confidence bounding function for $b = 2$
lcb _{2.5}	lower confidence bounding function for $b = 2.5$
WB ₁	Watson and Barnes' threshold-bounded extreme criterion
WB ₂	Watson and Barnes' regional extreme criterion
Maxvar	maximum variance criterion
Switch	switching criterion for 5 iterations of Maxvar

example. Therefore, the median value is reported in the tables rather than the mean to reduce the impact of outliers. Bar graphs are given for the same data in Figures 4.10 through 4.12 to facilitate comparisons. Note that the x_* metric for the Maxvar criterion was far higher than the for the others and therefore extends above the limits of plot.

The results from this study are not intended to provide conclusive evidence of which criterion performs best for the comparison metrics. Rather, the goal is to understand how the balance of local/global search as defined by the various criteria impact their search strategy. Do the criteria designed for more local searching in fact find the optimum more accurately? Do the criteria designed for more global searching in fact provide a more accurate metamodel? How effective is the approach of not compromising? These questions and others will be addressed in the following sections.

Table 4.3: Median value of performance metrics for Example 1

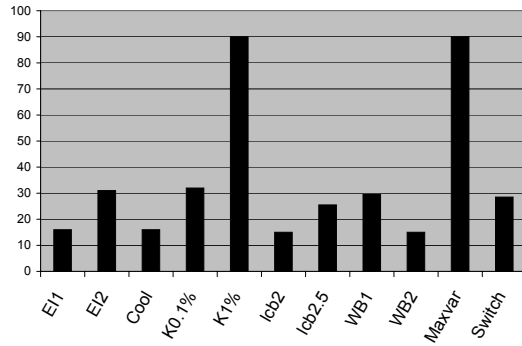
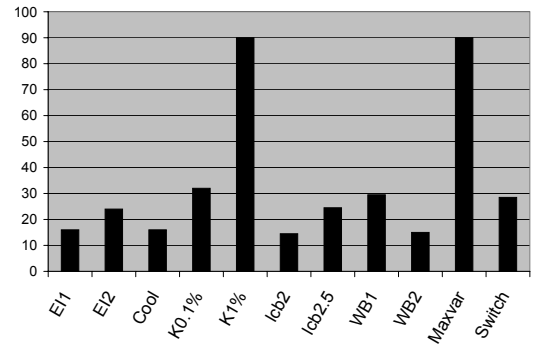
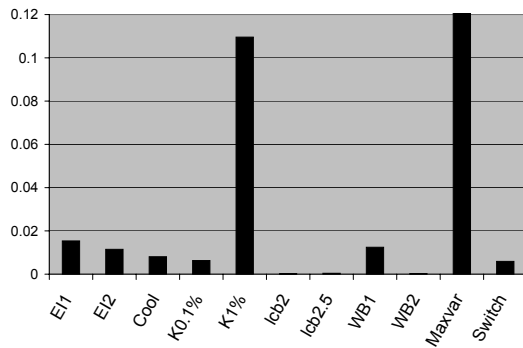
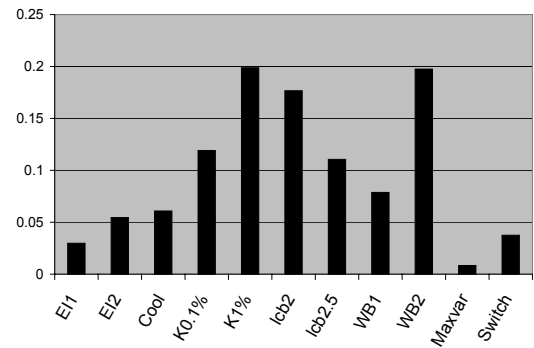
Criterion	$f_1\%$	$x_1\%$	x_*	RMS
El ₁	16	16	1.54e-2	2.98e-2
El ₂	31	24	1.15e-2	5.45e-2
Cool	16	16	8.06e-3	6.08e-2
K _{0.1%}	32	32	6.30e-3	1.19e-1
K _{1%}	90	90	1.10e-1	1.99e-1
lcb ₂	15	14.5	1.85e-4	1.77e-1
lcb _{2.5}	25.5	24.5	4.03e-4	1.11e-1
WB ₁	29.5	29.5	1.25e-2	7.87e-2
WB ₂	15	15	2.52e-4	1.98e-1
Maxvar	90	90	2.38e-1	8.37e-3
Switch	28.5	28.5	5.91e-3	3.75e-2

Table 4.4: Median value of performance metrics for Example 2

Criterion	$f_1\%$	$x_1\%$	x_*	RMS
El ₁	67.5	19.5	2.38e-2	2.85e-1
El ₂	39	25	3.63e-2	3.56e-1
Cool	22	18.5	1.91e-3	7.15e-1
K _{0.1%}	37	26.5	8.34e-3	8.21e-1
K _{1%}	32.5	26	1.24e-2	7.33e-1
lcb ₂	59	26	1.37e-2	7.26e-1
lcb _{2.5}	60	29	1.85e-2	6.13e-1
WB ₁	36	22	7.51e-3	6.93e-1
WB ₂	13.5	9	1.34e-3	1.18e0
Maxvar	90	90	6.70e-1	8.84e-3
Switch	35	16.5	3.06e-3	2.41e-2

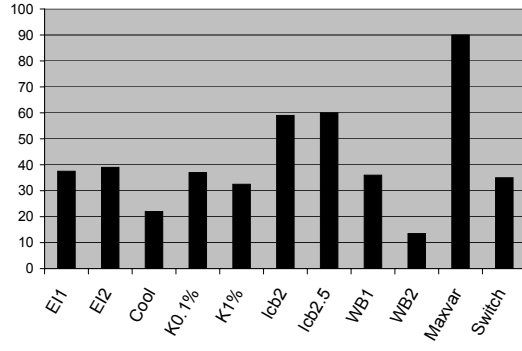
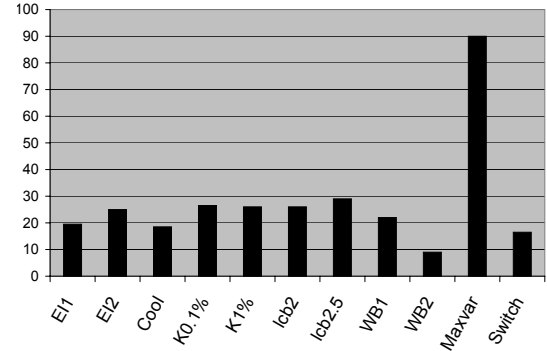
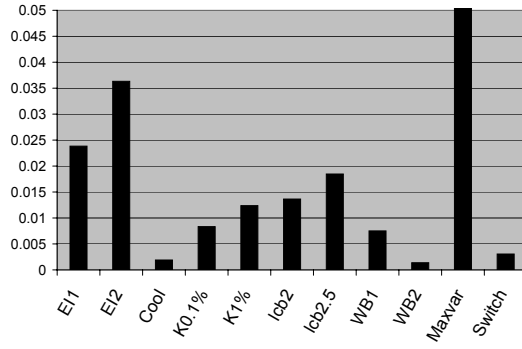
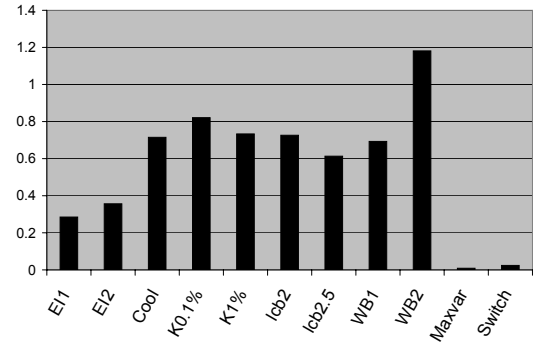
Table 4.5: Median value of performance metrics for Example 3

Criterion	$f_1\%$	$x_1\%$	x_*	RMS
El ₁	6.5	6.5	9.88e-3	5.97e-3
El ₂	15.5	15.5	9.61e-3	1.33e-2
Cool	14	14.5	7.38e-4	1.54e-2
K _{0.1%}	7.5	14.5	2.83e-3	1.84e-2
K _{1%}	45.5	45.5	1.86e-2	3.12e-2
lcb ₂	9	10.5	1.29e-4	2.98e-2
lcb _{2.5}	20.5	22	5.15e-4	2.10e-2
WB ₁	6	7.5	9.13e-4	1.37e-2
WB ₂	4.5	5	1.14e-4	3.17e-2
Maxvar	90	90	1.37e-1	1.73e-3
Switch	10	10.5	2.11e-4	5.34e-3

(a) $f_1\%$ metric(b) $x_1\%$ metric(c) x_* metric

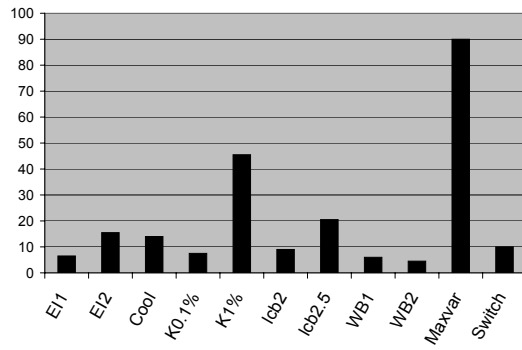
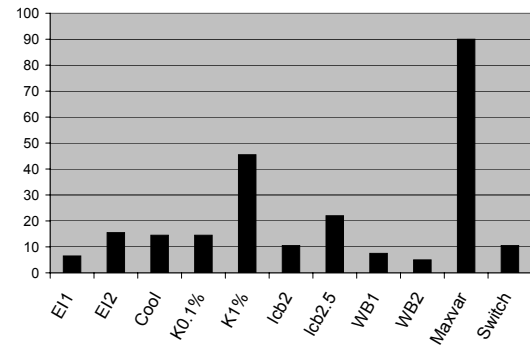
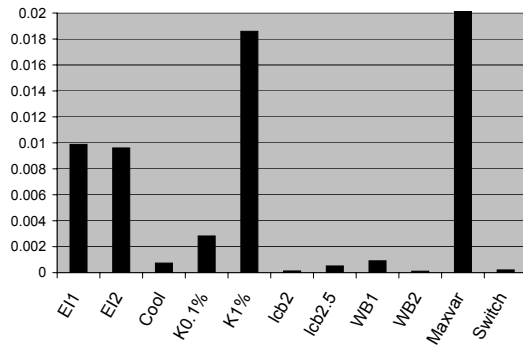
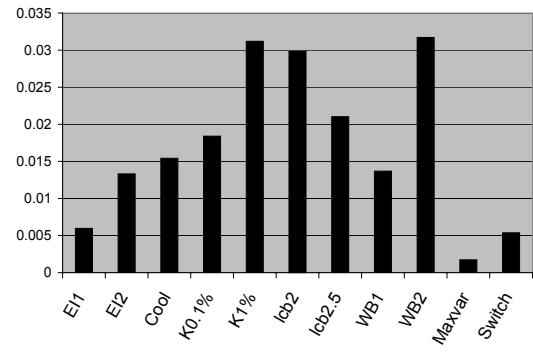
(d) RMS metric

Figure 4.10: Median value of comparison metrics for Example 1

(a) $f_1\%$ metric(b) $x_1\%$ metric(c) x_* metric

(d) RMS metric

Figure 4.11: Median value of comparison metrics for Example 2

(a) $f_1\%$ metric(b) $x_1\%$ metric(c) x_* metric

(d) RMS metric

Figure 4.12: Median value of comparison metrics for Example 3

4.4.1 Solution efficiency

Let us begin by examining how quickly the various criteria arrived at the solution as measured by metrics $f_1\%$ and $x_1\%$. As expected, the Maxvar criterion did not find solutions within the 1% limits before the 100 function call limit was reached. The variance reducing technique does not attempt in any way to improve upon the objective function, so any sample point within the $f_1\%$ or $x_1\%$ limits was purely coincidental. Theoretically, the generalized expected improvement, Kushner and lcb criteria can be made more local searching by decreasing their associated parameter values g , ϵ , and b , respectively. Generally speaking, the more local searching versions (EL_1 , $K_{0.1\%}$, WB_2) sampled points within the 1% limits more quickly than criteria for which the parameter values were higher (more global searching). While the results are not conclusive, they suggest that the parameter values may be tuned to control how quickly the algorithm arrives at a local solution. Of course, this comes at the risk of overlooking the global optimum. An efficient strategy may be to begin with a local search to quickly find a good solution before continuing on to revisit unexplored regions.

4.4.2 Solution accuracy

Let us next examine the issue of solution accuracy as measured by the x_* metric. The more local searching version of the Kushner criterion ($K_{0.1\%}$) yielded appreciably more accurate results than $K_{1\%}$. The lcb_2 function was slightly more accurate than the $lcb_{2.5}$ function for Examples 1 and 3, but based upon statistical hypothesis testing, the difference in Example 2 was insignificant.

Because only two values of the parameters ϵ and b were tested here, one cannot draw conclusions as to how effective changing parameter values is at producing a

more local search. The parameter b may need to be varied over a wider range in order to produce results that are significantly more local searching. An exhaustive sensitivity analysis is beyond the scope of this work, but may provide quantitative measures of the impact of these parameters on local search accuracy.

Because a wider set of variations on the EI function was tested, more discussion is merited on its local search accuracy. Changing the parameter g from 1 to 2 did not appear to have a strong impact on the criterion. In fact, the difference in the x_* metric was statistically significant only for Example 2, in which case the local accuracy for the EI₂ criterion was actually better than for EI₁. The WB₁ criterion is exactly the generalized EI function for $g = 0$ and therefore provides results for the most local searching version of the criterion. The accuracy of the solutions for WB₁ far surpassed EI₁ and EI₂ for Examples 2 and 3. This may indicate that there is a more significant difference between parameter values of $g = 0$ and $g = 1$ than there is for $g = 1$ and $g = 2$. For Example 1 however, there was no statistically significant difference between the x_* results for $g = 0, 1$ or 2 .

Additional insight into the question of parameter sensitivity can be gained from examining the results of the Cool criterion, as it used a wide range of g values for the generalized EI function over the course of optimization. Analysis of the Cool criterion verified that the initial few iterations produced data points spread throughout the design space, but later iterations honed in on a local optimum. In the implementation here, it continued using $g = 0$ from iteration 15 to the end, and thus settled on the most local searching version of the generalized EI function. The results of the x_* metric indicate that it was significantly more accurate at finding the optimum than the WB₁, EI₁ or EI₂ criteria. Further testing may increase the understanding of how parameters such as these impact the local searching accuracy of such criteria.

The WB_2 criterion is essentially the predicted function value added to the EI function and is therefore slightly more local searching. For all three examples used here, the x_* metric for the WB_2 criterion was substantially better than for the EI_1 criterion. This may be due to the relative ease of locating the criterion’s optima. As described above, the EI function returns to a value of zero at each sample point. As a consequence, the expected improvement function becomes extremely “bumpy” as sample points begin to accrue around the optimum. Such a function is difficult to optimize when solving the ISC subproblem. The WB_2 criterion does not exhibit this spiked behavior and is therefore more likely to produce a good iterate. To illustrate the difference between the two criteria, 30 points were sampled for the one dimensional example used in the plots of the previous section. Most of samples occur near the optimum on the right (see Figure 4.13). The two criteria have an optimum at the same location. However, it is apparent that the WB_2 criterion is much more likely to successfully locate the optimal infill sample during the ISC subproblem. For this reason, the WB_2 criterion may provide a more robust criterion than the expected improvement function.

4.4.3 Global searching properties

We next examine the global searching properties of the criteria. Because interpolating kriging models were used in these examples (i.e., $nugget = 0$), we have quantified the amount of global searching by the RMS of the approximation after 100 function calls. Note that in all three examples, the Maxvar criterion performed extremely poorly for the first three comparison metrics. However, it is designed to spread points throughout the design space and therefore provided a benchmark as the most global searching criterion. The switching criterion also placed a large em-

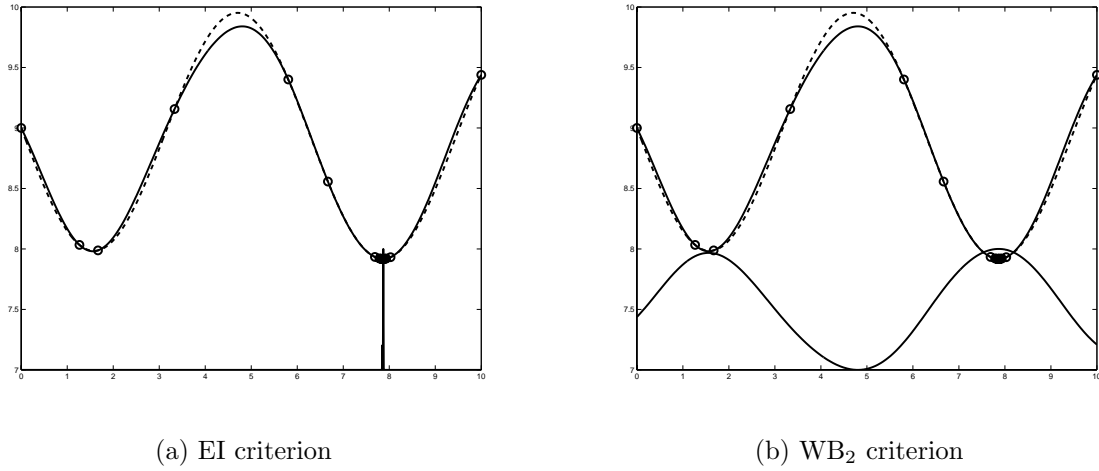


Figure 4.13: Comparison of the relative difficulty in locating the next iterate using the EI criterion versus the WB₂ criterion

phasis on global searching. Its RMS metric was the second best for Examples 2 and 3, and came in a close third to the EI function for Example 1.

As discussed in the last subsection, the parameters used in defining the criteria play an important role in how locally or globally they search. There was, however, no strong evidence here to suggest that more global searching parameter settings resulted in better RMS metrics. In fact, quite the opposite is true for many cases. For example, the EI₁ criterion actually yielded better RMS metrics than the EI₂ criterion. One possible explanation relates to the difficulty in solving the ISC subproblem as shown in Figure 4.13. It is conceivable that failure to locate the true optimum of the ISC subproblem led to sample points being more spread out around the design space because only local optima of the criterion could be found. These local optima would occur in between the sample points where the variance was at its peak.

4.4.4 Periodic local search

Based on the success of the switching criterion, it was hypothesized that some additional local search may improve upon the accuracy of the criteria. A periodic

local search was performed whereby the approximate model of the objective function was used as the sampling criterion to be minimized. Once every ten iterations, superEGO would find the minimum of the approximate model and sample the next iterate at that location, thereby increasing the solution accuracy. It would then return to whatever sampling criterion was specified by the user. Because 90 iterations were performed, the periodic local search occurred 9 times during each optimization.

Local searching versions of the same tests as above were performed. Comparisons between the two approaches are shown as bar graphs in Figures 4.14 to 4.16. The results from the standard approach and periodic local searching are shown as black and white bars, respectively. It should be mentioned that the local search performed here is exactly the same as that used in the switching criterion. For that reason, the difference between the standard and local searching variations of the switching criterion are not meaningful. The local searching version simply changed the cyclic pattern of the switching criterion.

In general, it appears that the periodic local search has limited success. In most cases, it significantly reduces the number of iterations required to sample a point within the $f_{1\%}$ or $x_{1\%}$ limits. However, it does not systematically do so. There are several cases where the standard technique is at least as efficient.

One fairly consistent result is that the periodic local search results in worse RMS metrics. This was expected because the nine iterates produced during the minimization of the metamodel tend to pile up near the optimum.

The periodic local search made a significant improvement in the x_* metric for all criteria in Example 3. However, there were several criteria in the first two examples for which the periodic local search actually resulted in a worse x_* metric, significantly so in many cases. The reason for this is not yet clear. It is possible that the difference

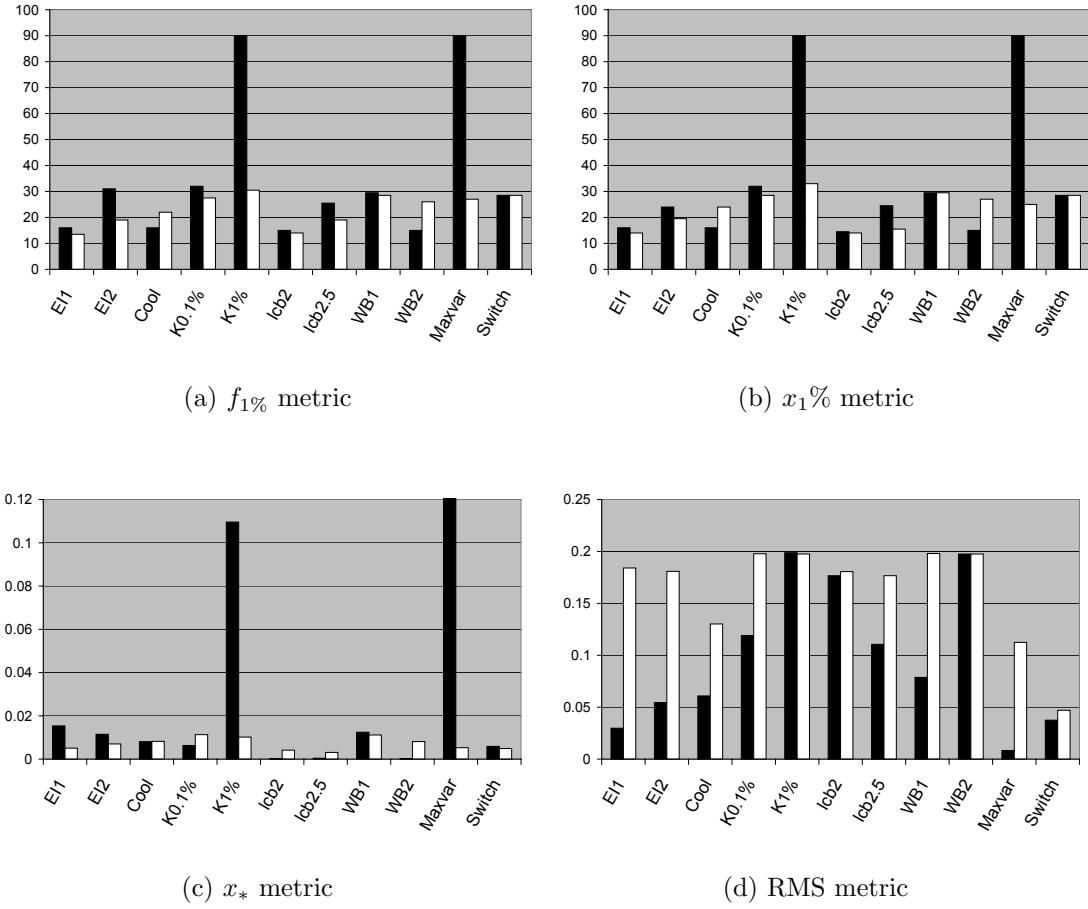


Figure 4.14: Comparison of metrics for standard approach (black) and periodic local search (white) for Example 1

in local search strategies was the cause. All the criteria except the switching criteria attempt to locate points with low objective function values using the z-statistic $\Phi(z)$ rather than purely minimizing \hat{y} as does the periodic local search. The former approach may be more successful than the latter. If that is the case, a more successful strategy may be to perform a periodic local search using the WB_1 criterion instead.

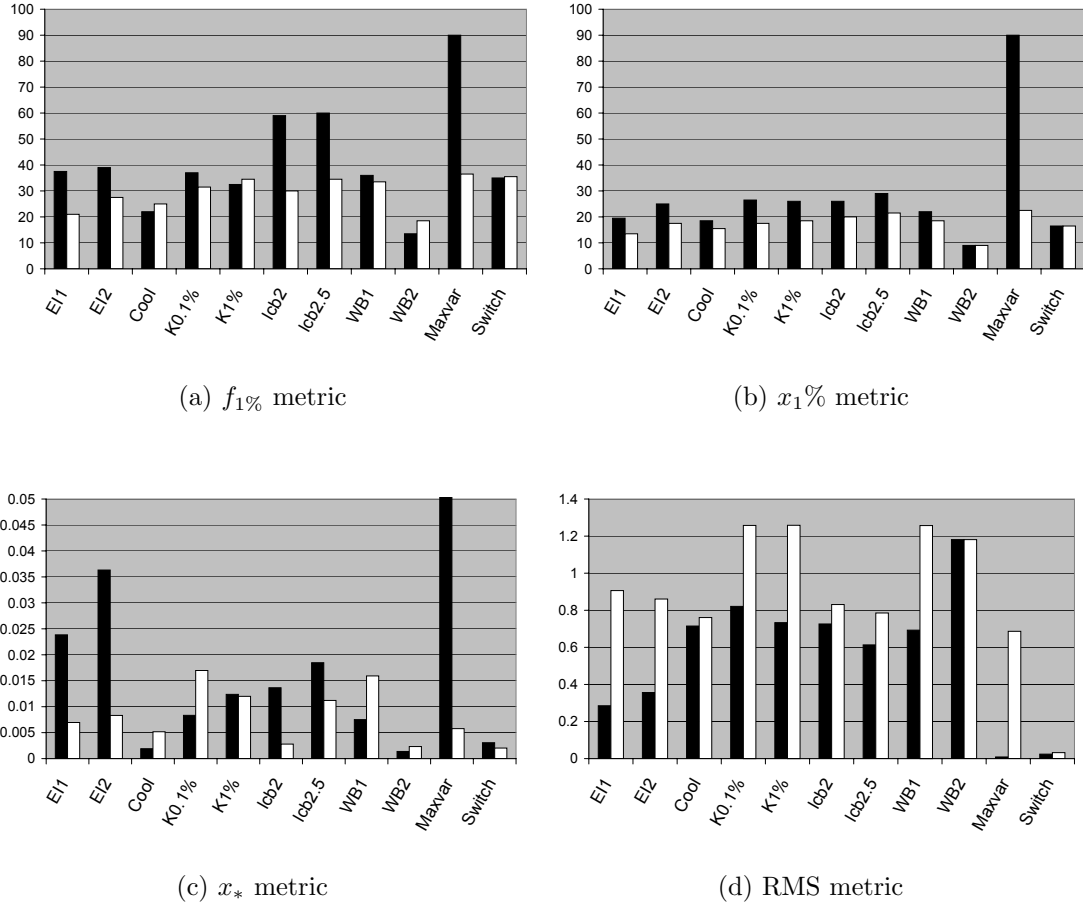


Figure 4.15: Comparison of metrics for standard approach (black) and periodic local search (white) for Example 2

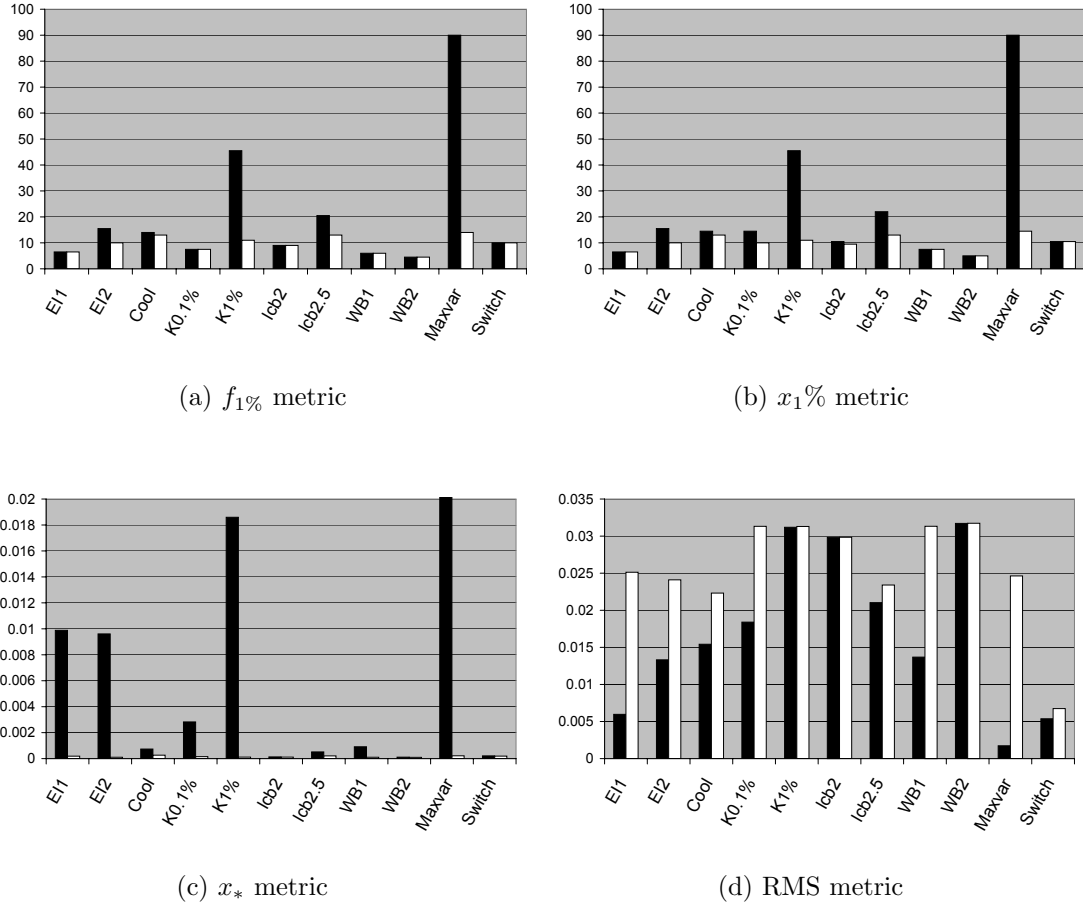


Figure 4.16: Comparison of metrics for standard approach (black) and periodic local search (white) for Example 3

4.5 Discussion

The main discussion of this chapter revolved around the balance between local and global searching. On one hand, the designer wishes to find the optimal solutions quickly and with as much accuracy as possible. On the other hand, there is the risk of overlooking the global optimum by overemphasizing the local search. There are also cases where the designer may wish to perform future analysis and what-if studies on the same problem. The ability to produce an accurate global approximation during the course of optimization is a valuable contribution to that end. However, it was shown that criteria that tend to be good at local searching are not always good at producing good global accuracy of the resulting approximation.

It was proposed that a more effective strategy may be to alternate between local and global search over the course of the optimization rather than to compromise between the two at each iteration. While the results here are not conclusive, they are quite promising. The switching criterion was the only one to perform well for all four metrics. In most cases, the switching criterion performed better than all others except the Maxvar criterion for the RMS metric. However, the switching criterion also performed quite competitively at the x_* metric as well, coming in either third or fourth in each example. It was not significantly less efficient than the other criteria. However, if efficiency of finding local solutions was of more importance, the number of Maxvar iterations performed in the switching cycle could have been adjusted.

The results here provide some evidence to support the claim that there are advantages to focusing on one objective at a time, rather than compromising at each iteration. The current implementation of superEGO only samples one new infill per iteration, making the switching criterion the only way of avoiding compromise. Other

researchers have performed similar Bayesian analysis studies that gather several new sample points per iteration [7], [41]. For each iteration, some of the new sample points are global searching and others are local searching, rather than compromising for the entire set of new infills. Similar to our own results, their research suggests that there are benefits to this approach. For future enhancements of superEGO, the ability to select multiple infill points per iteration would benefit from the findings of this research.

4.6 Chapter Summary

This chapter examined the differences between many of the sampling criteria proposed in the literature. Moreover, the criteria tend to produce similar search patterns that balance the objectives of finding local optima and searching in previously unexplored regions. It was shown, however, that some of the criteria can be guided more towards local or global search by changing the value of some criterion-specific parameters. It was also demonstrated that there may be advantages to alternating between local and global search rather than compromising between the two at each iteration.

The next chapter examines methods for including constraints in Bayesian analysis.

CHAPTER 5

Constrained Bayesian Analysis

The ability to incorporate nonlinear constraints is an important but as yet relatively unexplored aspect of Bayesian analysis. In this chapter, we examine several methods for extending Bayesian analysis to constrained optimization. In addition, a new method is proposed for locating feasible points within the design space, which is particularly useful for problems where the feasible design space is disconnected.

5.1 Methods for Constraint Handling

Bayesian analysis was originally developed for the optimization of simply bounded problems, that is, with constraints only on the ranges of the design variable values to be considered. More recent research has attempted to extend Bayesian analysis to include general, nonlinear constraints. This author is aware of only four approaches for doing so: the probability method of Schonlau [83], the penalty method [79], [80], the expected violation method of Audet et al. [7], and the constrained ISC method of Sasena et al. [81]. In this section we examine each method and discuss the rationale for choosing the constrained ISC approach.

5.1.1 Probability and penalty methods

The first two methods – the probability and penalty methods – are related in that they both attempt to transform the constrained optimization problem into an unconstrained one. In his work, Schonlau suggests multiplying the value of the expected improvement by the probability that the point is feasible. In this way, the probability of feasibility will reduce the magnitude of the expected improvement, driving it to zero where there is a very low likelihood of feasibility. Schonlau reports successful optimization of an engine piston subject to friction constraints using the probability method.

One concern is that the probability method will impact the value of the infill criterion too strongly, keeping the algorithm from exploring points directly along the constraint boundary where the true optimum lies. A proposed remedy is to use a penalty method, whereby a large constant (i.e., a penalty) is added to the criterion in order to restrict it from choosing infill samples in the infeasible region. Björkman and Holström apply a penalty method to their algorithm [13]. The penalty method will theoretically provide a sharp adjustment to the sampling criterion at the constraint boundary. By contrast, the probability-adjusted sampling criterion will transition into the infeasible zone more gradually. To illustrate the difference between the two, an optimization is performed on the mystery function from Chapter 4 with an additional constraint as shown in Equation (5.1).

$$\begin{aligned} \min f(\mathbf{x}) &= 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7 \sin 0.5x_1 \sin 0.7x_1x_2 \\ \text{subject to: } g(\mathbf{x}) &: -\sin(x_1 - x_2 - \frac{\pi}{8}) \leq 0 \end{aligned} \quad (5.1)$$

Figure 5.1 shows the contour of the EI function without adjustment in the region near the constrained optimum. The contours of the kriging approximation of the

objective function are shown as dashed lines. The constraint boundary cuts diagonally across, the bottom right being the feasible side. The circle on the top left is the nearest sample point, and the asterisk near the center is the location of the true optimum. The fact that the true optimum lies directly on the boundary of the constraint metamodel is an indication that the constraint was approximated accurately. The optimum of the EI function is shown as a triangle.

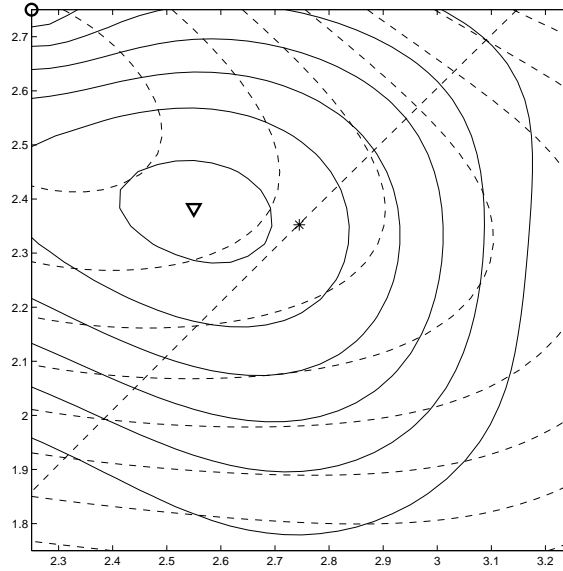


Figure 5.1: Contour plot of EI criterion for Equation (5.1). Dashed contours are of the true objective function. Sample point, true optimum, and optimum of ISC are shown as a circle, asterisk, and triangle, respectively.

Figures 5.2(a) and (b) show the contours of the probability-adjusted and penalty-adjusted EI criterion, respectively. The optimum for each criterion is shown as a triangle. Although both criteria drop off quickly in the infeasible region, the optimum of the penalty method is much closer to the constraint boundary than the probability method. This provides support to the claim that the probability method effectively pushes the infill sample point away from the constraint.

A similar test was run adding three sample points around the optimum. Figures 5.3(a) and 5.3(b) show the contours of the probability- and penalty-adjusted criteria,

respectively, with the additional sample points shown as circles and the optimum infill points shown as triangles. The two are extremely similar in shape and in the location of their optima. The probability method drops off slightly past the constraint boundary whereas the penalty method drops exactly at the boundary. The implications are unclear. Further testing must be done to understand what happens as points begin to cluster around the constraint boundary or as the number of constraint functions increases.

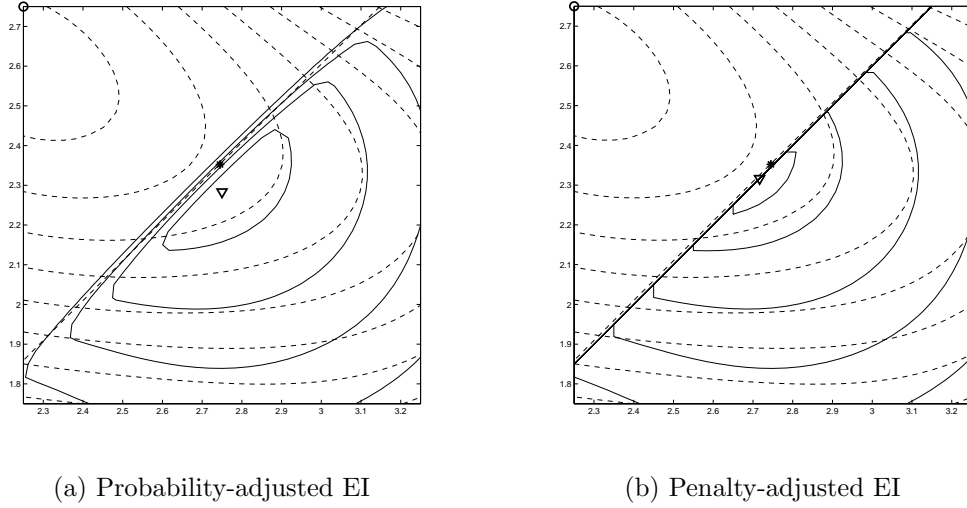
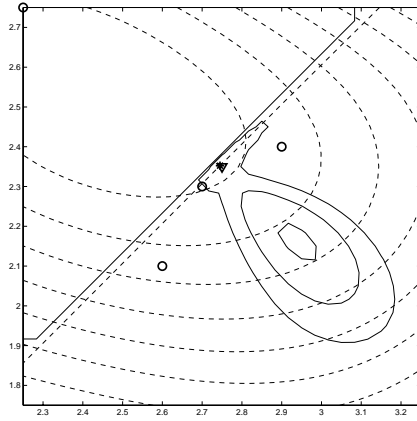
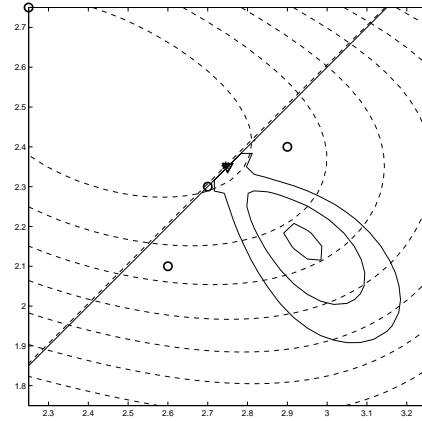


Figure 5.2: Contour plot of the adjusted EI criterion for Equation (5.1). Dashed contours are of the true objective function. Sample point, true optimum, and optimum of adjusted ISC are shown as a circle, asterisk, and triangle, respectively.

To visualize how the different methods behave, a 100 point optimization was performed on Equation (5.1). Figure 5.4 shows a comparison of the function calls made by the EI criterion with the probability method throughout (a) and with a switch to the penalty method after ten iterations (b). The contours are of the true functions, and the initial samples are shown as circles and the infill points as x's. While both methods cluster around the global optimum, the probability method samples more frequently in the infeasible space. The penalty method clusters



(a) Probability-adjusted EI



(b) Penalty-adjusted EI

Figure 5.3: Contour plot of the adjusted EI criterion for Equation (5.1) in the presence of additional sample points. Dashed contours are of the true objective function. Sample points, true optimum, and optimum of adjusted ISC are shown as a circle, asterisk, and triangle, respectively.

more sample points around the optimum on the feasible side of the constraint. For optimization problems where strict feasibility at each iteration is important, using a penalty method may prove useful.

5.1.2 Expected violation method

Another method for constraint handling was proposed by Audet and colleagues at Rice and Boeing [7]. They take an alternative approach whereby the constraints are inspected before the sampling criterion is considered. Their Constrained, Balanced, Local-Global Search (CBLGS) algorithm incorporates the expected improvement function of Equation (4.4) and is designed to search for $ngoal$ infill samples per iteration from among a finite set of candidate locations generated by a large Latin hypercube. They first calculate the *expected violation* of each candidate, defined as:

$$EV_i = \begin{cases} (\hat{g}_i - 0)\Phi\left(\frac{\hat{g}_i - 0}{\hat{\sigma}_i}\right) + \hat{\sigma} \phi\left(\frac{\hat{g}_i - 0}{\hat{\sigma}_i}\right), & \text{if } \hat{\sigma}_i > 0 \\ 0, & \text{if } \hat{\sigma}_i = 0 \end{cases}, \quad (5.2)$$

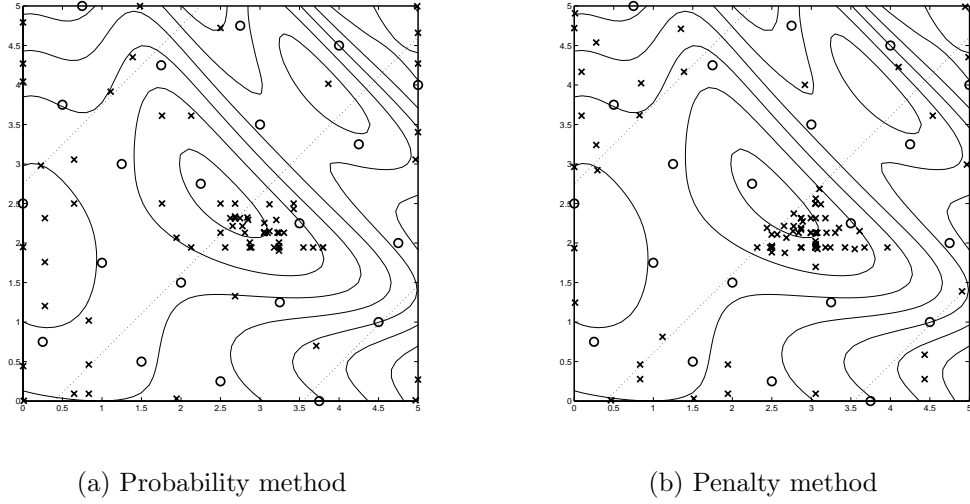


Figure 5.4: Differences in the feasibility of sampling regions for the probability and penalty methods. The contours are of the true functions, and the initial samples are shown as circles and the infill points as x's.

where \hat{g}_i and $\hat{\sigma}_i$ are the predicted mean and standard deviation of constraint i , respectively. The dependence on \mathbf{x} is left out for notational simplicity. Note that the expected violation is exactly analogous to the expected improvement function. It is low except in regions where the constraint is likely to be violated or where there is a large uncertainty in the model of the constraint. The expected violation is vector-valued for problems with multiple constraints.

The CBLGS algorithm then accepts any candidate for which the infinity norm (i.e., maximum) of the expected violation vector is within a user-specified limit. The expected improvement is then calculated only for the set of acceptable candidate points. Finally, the algorithm evaluates the true functions at the $ngoal$ number of candidate locations for which the expected improvement is largest, where $ngoal = 5$ in their implementation.

The expected violation as given by Audet et al. is not sufficiently defined. If a candidate point coincides with a previously sampled point, the expected violation will

be zero if the sample point is feasible. However, if the sample point is infeasible, there is no discussion of what value to assign the expected violation in such a case. In our implementation, we have arbitrarily assigned a value of 1000 to such an occurrence.

To help visualize the expected violation, we have used test function #1 as a constraint that must satisfy

$$g(\mathbf{x}) = -\sin(x) - \exp(x/100) + 10 \leq 9.4, \quad (5.3)$$

we get the expected violation shown in Figure 5.5. It has a large peak in the center and a smaller one on the right where it is fairly certain to be infeasible. It also has a smaller peak on the left where the uncertainty is high. The latter behavior is actually quite detrimental because it may prevent the optimization algorithm from sampling points in regions of high uncertainty.

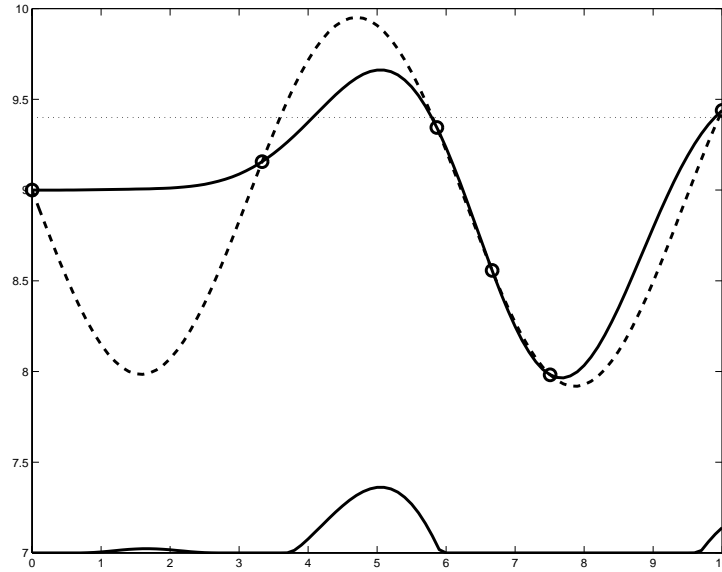


Figure 5.5: Plot of expected violation for Equation (5.3). Dashed line is the true function, dotted horizontal line is the constraint boundary, solid line is the approximation, circles are the sample points. The plot at the bottom is the expected violation.

5.1.3 Constrained ISC method

The fourth method for handling constraints, originally proposed by Sasena, Papalambros, and Goovaerts [81], is simply to solve the ISC subproblem as a constrained optimization problem. Rather than transforming the ISC, the information about constraint satisfaction is considered directly in the solution of the ISC subproblem. This is different than the expected violation method in that a constrained optimization problem is solved rather than an exhaustive enumeration of the expected violation at a predetermined set of candidate sample points. The constrained ISC method has been selected for the superEGO algorithm.

5.1.4 Justification of choosing constrained ISC method

Before continuing, let us briefly give some justification for choosing the constrained ISC method. First of all, we have rejected the probability method. Based on some preliminary analysis, it appears to have difficulty finding solutions directly along the constraint boundary. Another important shortcoming is that the probability method will not work with every kind of sampling criterion.

Consider, for example, the local search criterion described in Chapter 4 which uses the predicted value of the objective function, $\hat{y}(\mathbf{x})$, as the ISC. Multiplying $\hat{y}(\mathbf{x})$ by the probability of feasibility would not result in the next infill point being the constrained minimum of the $\hat{y}(\mathbf{x})$. For instance, if $\hat{y}(\mathbf{x})$ took on both positive and negative values, multiplying it by a number between 0 and 1 would not have the desired result. The strength of superEGO is its flexibility to offer a variety of sampling criteria. Using the probability method would restrict the set of applicable criteria, rendering it impractical for implementation within superEGO.

The penalty method, however, will work regardless of sampling criterion. In

infeasible regions, it simply increases the value of the ISC to be minimized, making it general enough for our purposes. There are however, shortcomings of the penalty method that cause us to reject it as well. For one, adding a penalty to the objective function of the ISC subproblem creates a large discontinuity exactly at its optimum. This makes accurately finding the optimum infill sample quite difficult, especially for the DIRECT algorithm which is used to solve the ISC subproblem. Another difficulty associated with penalty methods is finding a good value for the penalty parameter. For these reasons, the penalty method of constraint handling has been rejected for the purposes of this research.

The expected violation method for handling constraints has some advantages over the probability and penalty methods. First, it does not transform the sampling criterion, thereby avoiding any undesirable distortion. Second, the expected violation is computed before the ISC objective. Then the ISC objective is only computed for those candidates that “pass the test”. This is an efficiency measure that may reduce the time needed to solve the ISC subproblem.

There are shortcomings to the expected violation method that leads us to choose the constrained ISC method. The most significant is the way in which it locates the infill sample. Because it evaluates points from among a finite set of candidates, it may not be able to locate the optimum of the ISC subproblem as accurately as solving the constrained ISC subproblem as an optimization problem. In addition, the number of candidate points to evaluate must be very large for there to be any reliability in the selected infill. Audet and coauthors report using Latin hypercubes of 10,000 to 100,000 candidates when searching for the next iterate [7]. This is much more than would be required for a reliable constrained optimization, even for a moderately large number of design variables.

For the above reasons, we have decided to use a constrained optimization algorithm, namely DIRECT, to extend Bayesian analysis to constrained optimization via the constrained ISC approach.

5.2 Quantifying Constraint Satisfaction

Having decided upon the method for incorporating constraints into superEGO, we turn our attention to how to measure if a constraint has been satisfied. The difficulty lies in the fact that we only have approximations of our constraint functions. The question is how best to use the approximations to locate design points that satisfy the true constraint functions.

The constrained ISC method is quite general and allows for any number of methods for quantifying the constraint violation. The superEGO algorithm currently implements six different approaches. The choices are summarized in Table 5.1 below. There are three ways to quantify the satisfaction of each constraint: the probability of feasibility, the estimated feasibility, and the expected violation function. One may use each metric as a vector, or take the maximum of the vector.

Table 5.1: List of metrics for quantifying constraint satisfaction

Name	Formula	Description
Probability _v	$P(\hat{\mathbf{g}}(\mathbf{x}) \leq 0) \geq P_{tol}$	vector
Probability _s	$\max_i P(\hat{g}_i(\mathbf{x}) \leq 0) \geq P_{tol}$	scalar
Estimated _v	$\hat{\mathbf{g}}(\mathbf{x}) \leq \mathbf{0}$	vector
Estimated _s	$\max_i \hat{g}_i(\mathbf{x}) \leq 0$	scalar
EV _v	$\mathbf{EV}(\mathbf{x}) \leq 0.01$	vector
EV _s	$\max_i EV_i(\mathbf{x}) \leq 0.01$	scalar

The probability metric states that the probability of feasibility must be greater

than some tolerance. By default, $P_{tol} = 0.95$ in superEGO, but the user may specify any threshold. The estimated feasibility metric uses the predicted value of the constraint functions directly as constraints of the ISC subproblem. The tolerance on the constraint is that of the original design problem. The last metric is the expected violation of Equation (5.2), which is constrained to be below 0.01 as suggested by Audet et al [7].

To better understand these metrics, we defined a new example in Equation (5.4) which will be referred to as test function #2.

$$\begin{aligned}
 \min f(\mathbf{x}) &= -(x_1 - 1)^2 - (x_2 - 0.5)^2 \\
 \text{subject to: } g_1(\mathbf{x}) &: (x_1 - 3)^2 + (x_2 + 2)^2 e^{-x_2^7} - 12 \leq 0 \\
 g_2(\mathbf{x}) &: 10x_1 + x_2 - 7 \leq 0 \\
 g_3(\mathbf{x}) &: (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.2 \leq 0
 \end{aligned} \tag{5.4}$$

defined over $x_i \in [0, 1], i = 1, 2$. There are local optima at $\mathbf{x} = [0.2316, 0.1216]$ and $\mathbf{x} = [0.2017, 0.8332]$. The latter is the global solution which has a value of -0.7483, and g_1 and g_3 are active. Figure 5.6 shows the problem. At left is the objective function, which gets better away from the point $[1, 2]$. At right is a contour plot where the solid and dashed contours are of the objective and the constraints, respectively. The shaded region is the feasible design space. Because the two active constraints meet at the objective at an acute angle, this problem can be difficult to solve accurately by approximation-based methods.

We began by running a 50 point optimization on the example using the expected improvement as the ISC. It was observed that after superEGO found a design point reasonably close to the solution area, it began sampling the same point repeatedly. This behavior was examined further by creating plots of the sampling criterion. As

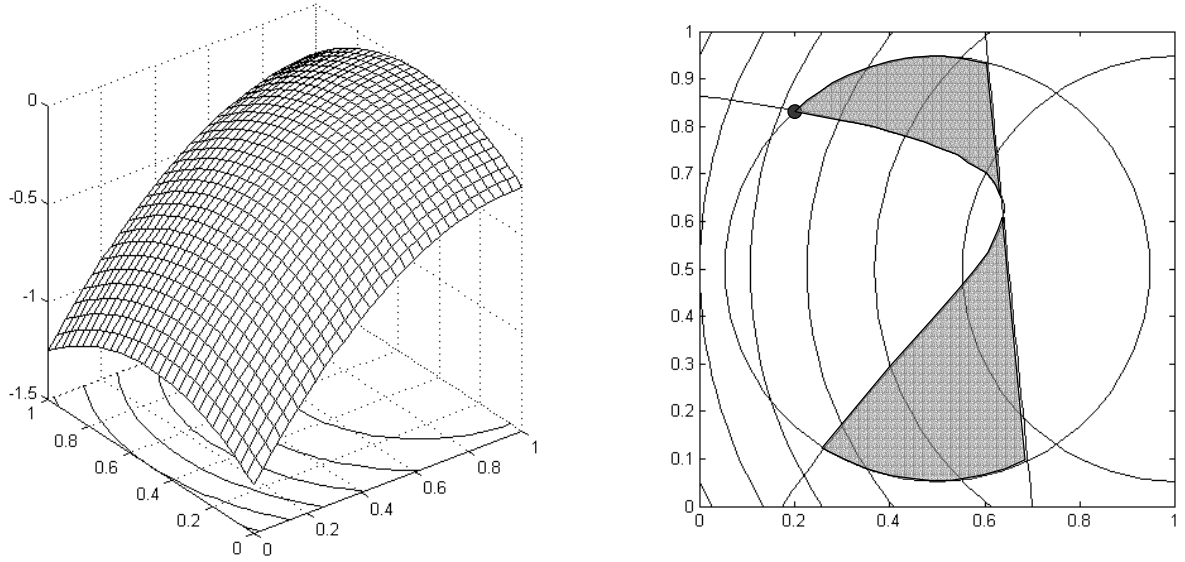


Figure 5.6: Contour plot of test function #2. Solid contours are of the objective, dashed lines are constraints. The problem is feasible in the shaded area. The optimum is shown as a filled in circle.

shown is Figure 5.7, the expected improvement function to be minimized is almost completely flat (with a value of zero) until the predicted value of the objective function drops below the current best point. This can be explained by the fact that the model had a very low variance, and therefore the expected improvement function took on non-negligible values only for regions of good improvement.

The problem we experienced occurred because of the constraint functions. They are shown in both plots as solid lines. In this example, the vast majority of the feasible design space is flat, with a only small corner below the plane. While in theory, the sampling criterion does in fact lead to a good design point at that corner, the practicality of finding such a point is a concern. Because there is no trend for most of the feasible space to guide the search to the corner, narrowing in on the solution was difficult.

To address this difficulty, we used the WB_2 criterion described in Section 4.1.7,

which adds the predicted value of the objective to the expected improvement function. This criterion, shown Figure 5.8, had a definite trend throughout the entire feasible design space, guiding the DIRECT algorithm more easily towards the optimum of the ISC subproblem. This provides additional support to favor WB_2 over the expected improvement criterion.

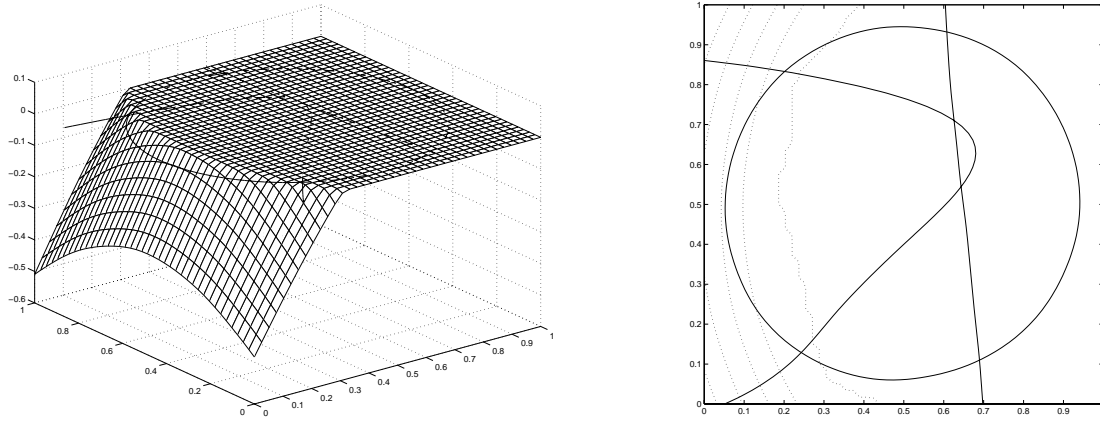


Figure 5.7: Plots of the EI criterion for Equation (5.4). The constraint boundaries are shown as dashed lines.

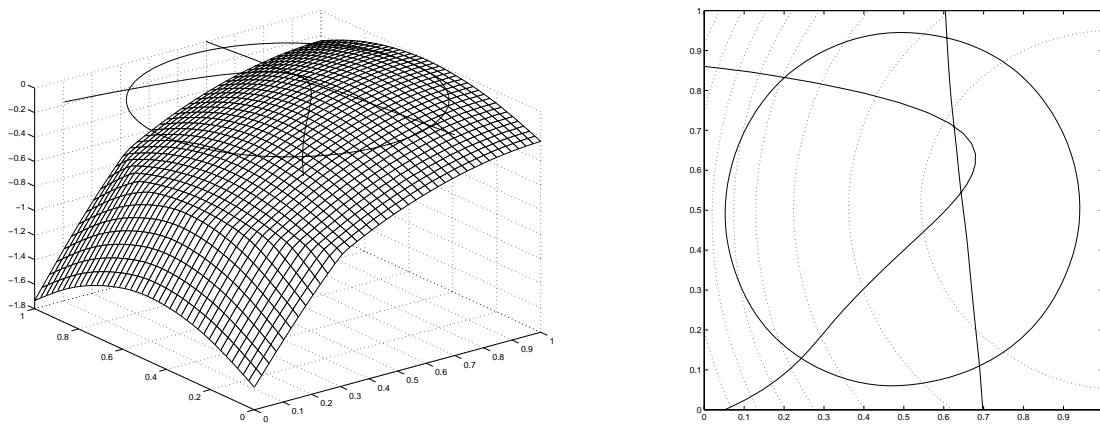


Figure 5.8: Plots of the WB_2 criterion for Equation (5.4). The constraint boundaries are shown as dashed lines.

Having decided upon a sampling criterion, we next examined the behavior of the six constraint satisfaction metrics by running a series of optimizations for 50

function evaluations. As with the methodology described in Section 4.3, we fixed the covariance model parameters for all iterations to reduce the influence of model fitting on the results. We also ran 10 optimizations for each constraint satisfaction metric to reduce the impact of the DOE for the initial 10 point sample.

Table 5.2 and Figure 5.9 show the comparison metrics used here: x_* and $x_{nearest}$. The former is the Euclidean distance from the best sampled feasible point to the known solution for Equation (5.4). The $x_{nearest}$ metric is the Euclidean distance from the known solution to the nearest point sampled, feasible or not.

Table 5.2: Comparison of constraint satisfaction metrics

Metric	x_*	$x_{nearest}$
Probability _v	2.20e-4	2.20e-4
Probability _s	2.20e-4	2.20e-4
Estimated _v	2.77e-5	1.56e-5
Estimated _s	2.20e-4	2.20e-4
EV _v	1.84e-1	5.49e-3
EV _s	2.49e-1	9.48e-3

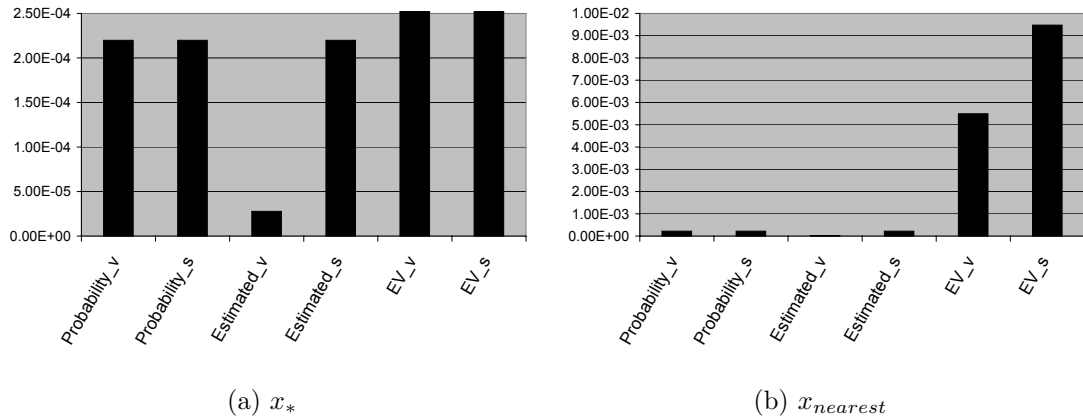


Figure 5.9: Comparison of constraint satisfaction metrics. Figure (a) cuts off the values for EV_v and EV_s because they are 3 orders of magnitude larger.

The results exhibit a few trends worth noting. For one, the scalar-valued treat-

ment of the constraint satisfaction metric was never better than the vector-valued treatment. This indicates that, while computationally simpler, there does not appear to be much gain in using the maximum of a constraint satisfaction metric rather than the vector itself. In fact, the results for the vector-valued Estimated_v metric was significantly better than for the scalar Estimated_s metric. The DIRECT algorithm is not significantly slower for a problem of ten constraints than it would be for one with a single constraint. Therefore, there is no impetus for using the scalar valued constraint satisfaction metric. A second observation of interest is that the Estimated_v metric appears to locate the constrained optimum more precisely than the others. The probability metric appeared to perform fairly well also, but using the expected violation metric did not perform well.

While we have only tested a single example function here, there is reason to believe that the conclusions may generalize. It was discussed above that the measuring the constraint satisfaction as the probability of feasibility may be too conservative, preventing the algorithm from sampling points close to the constraint boundary. As for the expected violation, note the significant difference between the x_* and $x_{nearest}$ metrics. This indicates that the expected violation may be too liberal, choosing candidates points that are in truth infeasible.

These interpretations are validated by the example here. Figure 5.10 shows the contours of the constraint satisfaction metric (solid lines) against the true constraint boundaries (dashed lines). The contours of the WB_2 metric are shown as dotted lines. For all plots, the same set of sample points was used, four of which were in the vicinity of the true optimum. At the global view, the only noticeable trend is that estimated value of the constraint appears to match the true constraint boundary more closely than either the probability or expected violation metrics. Figure 5.11

shows a close-up of the region around the optimum. At this view, it is clear that the probability metric is somewhat conservative, the estimated metric is difficult to distinguish from the true boundary, and the expected violation metric is too liberal.

These plots confirm the hypothesis that the expected violation metric resulted in a worse x_* metric for this example because most of the infills it chose were in fact infeasible. The behavior of the expected violation is such that it gradually increases around the infeasible zones (e.g., see Figure 5.5). If the expected violation does not reach a value of 0.01, it is considered an acceptable candidate point. The metric chose infills in the infeasible region because their expected violation was below the threshold. Determining an acceptable value for the threshold is difficult. Too low a value will result in the metric being too conservative, avoiding points directly along the constraint boundary. Too high a value will result in overconfidence, and many sample points will be placed in the infeasible region.

In general, the probability and expected violation metrics do not predict the constraint boundary as accurately as using the kriging models directly to estimate them. This is perhaps due to the fact that they incorporate the kriging variance. It has been the experience of this author that the kriging variance is not always a reliable measure of the uncertainty in the model. It is not unusual to fit a kriging model to a data set that matches the true function quite accurately, yet produces very high, or even erratic kriging variance values. Because using the estimated value of the constraint function does not include any information from the kriging variance, it may be more robust than either the probability or expected violation metrics.

Based upon the discussion above, it is recommended that the vector of constraint values estimated from the kriging models be used as the measure of constraint satisfaction.

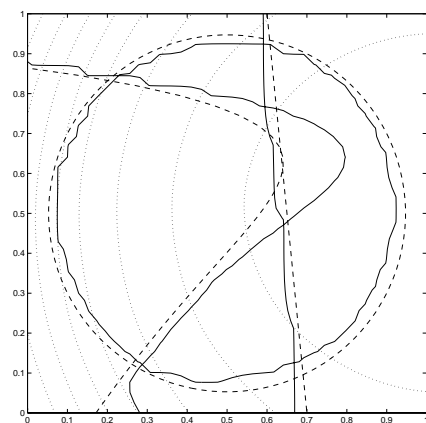
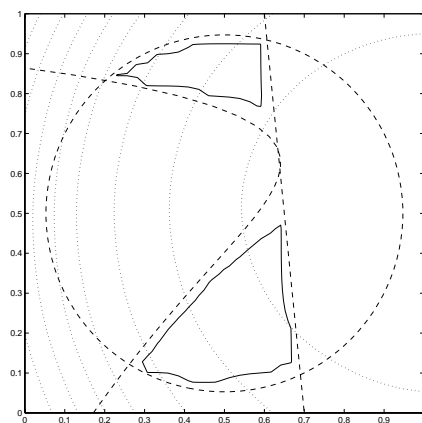
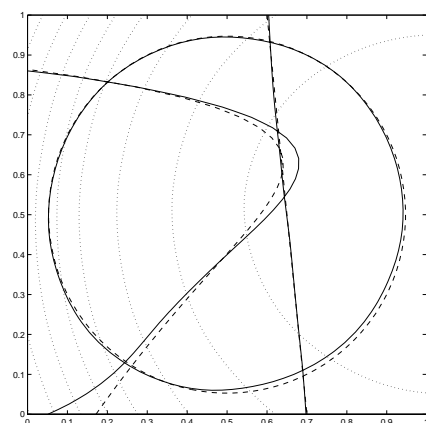
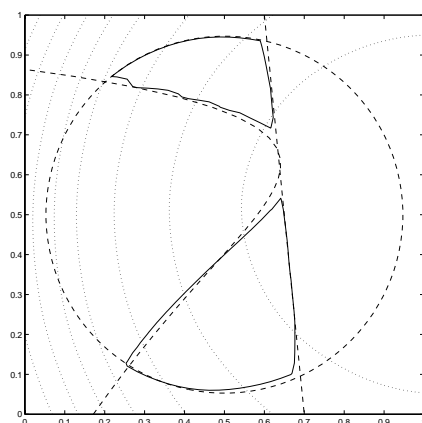
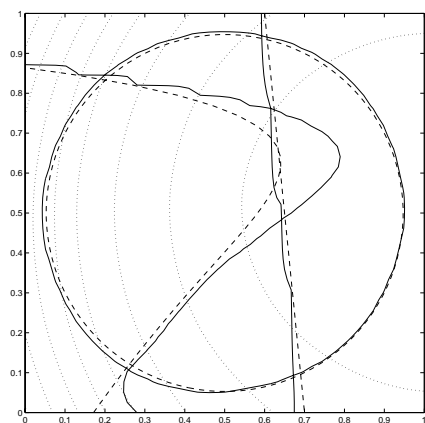
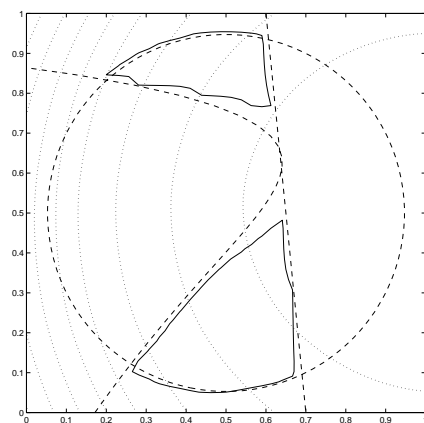
(a) Probability_v(b) Probability_s(c) Estimated_v(d) Estimated_s(e) EV_v(f) EV_s

Figure 5.10: Differences between the quantification of constraint satisfaction. The contours of the ISC are dotted lines, the true constraint functions are dashed lines and the ISC constraint functions are solid lines.

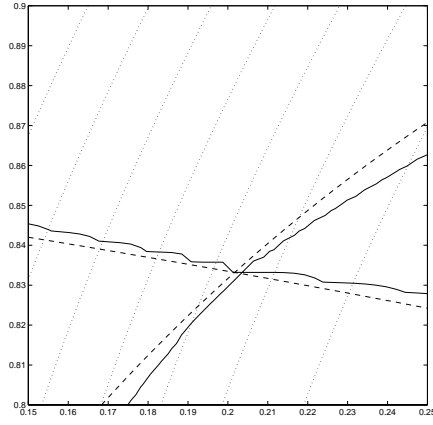
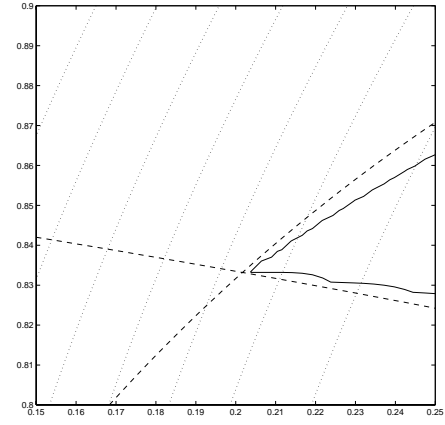
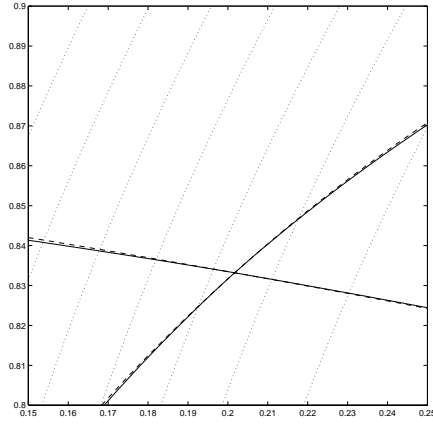
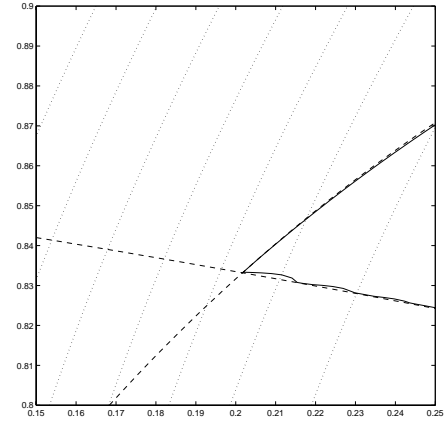
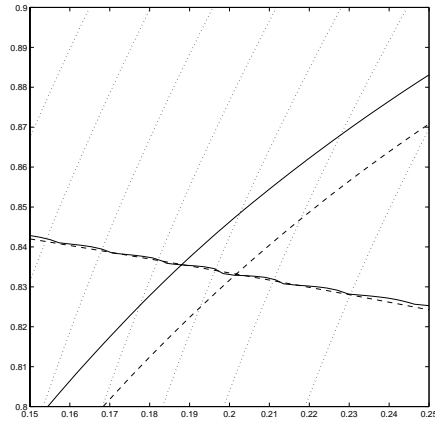
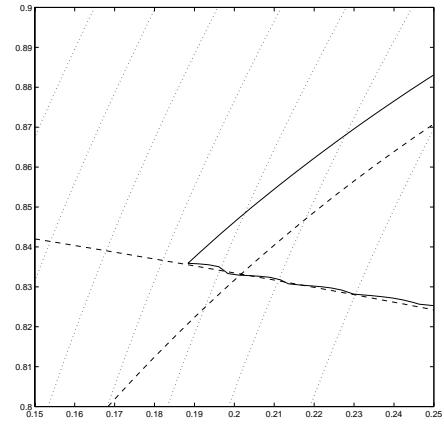
(a) Probability_v(b) Probability_s(c) Estimated_v(d) Estimated_s(e) EV_v(f) EV_s

Figure 5.11: Differences between the quantification of constraint satisfaction (close-up). The contours of the ISC are dotted lines, the true constraint functions are dashed lines and the ISC constraint functions are solid lines.

5.3 Disconnected Feasible Regions

This section examines the challenge of solving design problems where the size of the feasible design space is small relative to the range of the entire space. We also examine problems which have disconnected feasible regions (i.e., there are several “islands” of feasibility), making it quite difficult to locate the global optimum. We propose taking advantage of superEGO’s flexibility and utilize alternative sampling criteria to locate multiple feasible regions in a new way.

The proposed strategy is based upon the probability that a candidate design point is feasible. As will be discussed in Chapter 6, one may classify each constraint function as “inexpensive” or “expensive” based upon the time it takes to compute. For expensive constraint functions the probability of feasibility is estimated from the statistical models, as described in this chapter. For inexpensive constraints, the probability of feasibility takes a value of 0 or 1 because one can directly determine if the constraint is satisfied.

5.3.1 Locating an initial feasible point

When no feasible point has yet been found, one can simply search for the location in the design space that yields the highest probability of feasibility. We propose the following sampling criterion when searching for an initial feasible point:

$$ISC_{13}(\mathbf{x}) = \prod_{i=1}^m P(g_i(\mathbf{x}) \leq 0) , \quad (5.5)$$

The subscript 13 refers to the ISC number used in the superEGO code to identify this particular criterion.

Because the inexpensive functions are evaluated directly, the probability of fea-

sibility is simply:

$$P_{inexp} = \begin{cases} 1, & \text{if } g(\mathbf{x}) \leq 0 \\ 0, & \text{otherwise} \end{cases} . \quad (5.6)$$

For expensive constraints, the kriging model is used to estimate the probability of feasibility as:

$$P_{exp} = \Phi \left(\frac{0 - \hat{g}(\mathbf{x})}{\hat{\sigma}(\mathbf{x})} \right) , \quad (5.7)$$

where Φ is the Gaussian cumulative distribution function, $\hat{g}(\mathbf{x})$ is the kriging estimate of the constraint function and $\hat{\sigma}^2(\mathbf{x})$ is its associated kriging variance computed via Equations (3.11) and (3.13), respectively. While a constraint boundary of 0 is shown in the equations above, the user is free to set any constraint tolerance they wish.

To illustrate the behavior of the criterion, a two dimensional example with one constraint function is introduced. The objective is a simple quadratic function, but the Branin function is treated as a constraint whereby its value must be below 5. The example problem, referred to here as the *newBranin* example, is of the form:

$$\begin{aligned} \min f(\mathbf{x}) &= -(x_1 - 10)^2 - (x_2 - 15)^2 \\ g(\mathbf{x}) &= \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10 \leq 5 . \end{aligned} \quad (5.8)$$

A contour plot of the newBranin example is shown in Figure 5.12. The problem consists of three disconnected feasible regions (shown as dashed lines). The contours of the objective improve down and to the left. Thus the global optimum of the newBranin example occurs at $\mathbf{x} = [3.2730, 0.0489]$ (shown as an asterisk) with local optima (shown as circles) in the two other feasible regions.

A plot of the ISC_{13} criterion using an initial 21 point sample (none of which are feasible) on the newBranin example is shown in Figure 5.13. Because there is only one constraint function, ISC_{13} is the probability that the constraint will be satisfied.

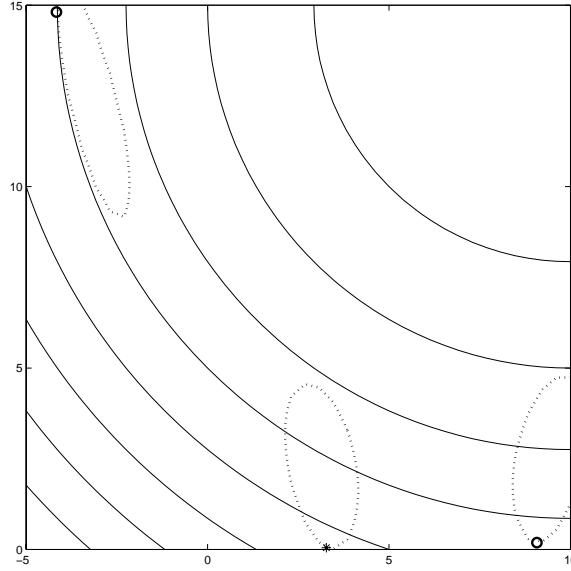


Figure 5.12: Contour plot of the newBranin example. Feasible region boundaries are shown as dashed lines, the local optima as circles, and the global optimum as an asterisk.

As expected, there are three regions of high probability of feasibility. The remainder of the design space is very unlikely to be feasible and therefore has a value of ISC_{13} close to zero.

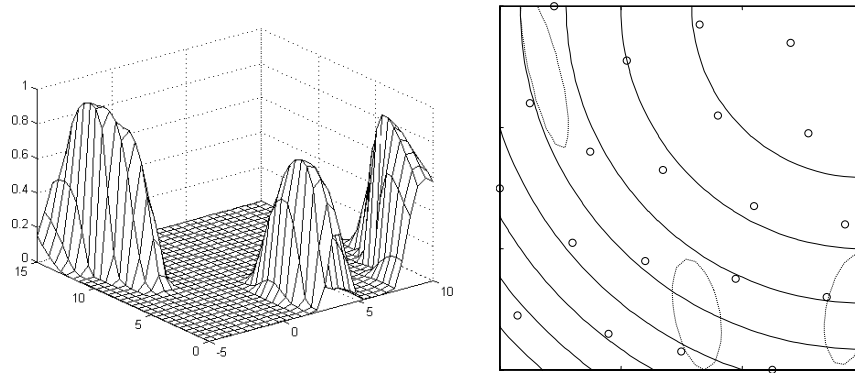


Figure 5.13: Mesh and contour plot of ISC_{13} for the newBranin example. Sample points are shown as circles.

Optimizing the ISC_{13} criterion searches for points where the probability of feasibility is largest. Of course, this will only work for cases where feasible points have yet to be found. Otherwise, there will be a region of probability 1, and the algorithm

will simply continue sampling in that region. Therefore, a second strategy must be employed to find subsequent feasible points.

5.3.2 Locating subsequent feasible points

Once an initial feasible point has been found, the challenge is to force the algorithm to search elsewhere for other feasible regions. We propose using an approach similar to *tunnelling* [48] whereby the ISC subproblem maximizes the distance from the current best point while searching for areas that have a high probability of feasibility. The two terms in the criterion are the probability (P) and the distance to the nearest feasible point (D).

$$ISC_{14}(\mathbf{x}) = \prod_{i=1}^m P_i \cdot D_i , \quad (5.9)$$

where P is calculated by Equations (5.6) and (5.7) above and the distance, D, is calculated as

$$D = \min_{x_{feas}} \left(\frac{\|\mathbf{x}_{feas} - \mathbf{x}\|}{range} \right) . \quad (5.10)$$

Dividing the distance by the range is done to balance the magnitudes of the distance in each dimension. Failure to do so might lead to the distance portion of ISC_{14} being dominated by one design variable. Multiplying the distance by the probability has the effect of reducing the value of the criterion wherever there is a low probability of feasibility.

To illustrate the behavior of ISC_{14} , we continue the same newBranin example. After the initial 21 point design, ISC_{13} guides superEGO to sample a point inside the feasible region in the upper left portion of the design space. ISC_{14} was then used to distinguish between these regions of likely feasibility by their distance from the feasible point. A plot of the sampling criterion is shown in Figure 5.14. The initial sample points are shown as circles, and the only feasible point is shown as

an x . Observe how the surface takes a noticeable dip in the immediate vicinity of the known feasible point. The algorithm is guided to the center of the the feasible region farthest away. Once that location is sampled, only the middle feasible region is left unexplored. Figure 5.15 shows the ISC_{14} now that two feasible points have been found. Note that the criterion is small in the immediate neighborhood of those points, but is high in the middle where the constraint is likely to satisfied and there are no nearby feasible points. Thus utilizing ISC_{13} and ISC_{14} successfully locates all three feasible regions in just three iterations.

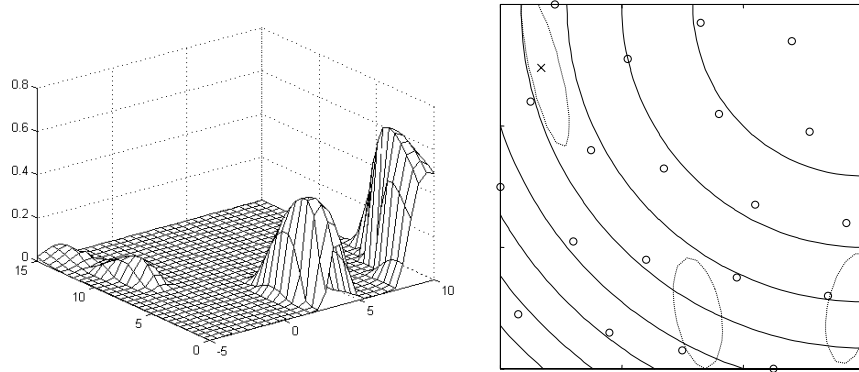


Figure 5.14: Mesh and contour plot of ISC_{14} for the newBranin example. Initial sample points are shown as circles, and the only feasible sample is shown as an x .

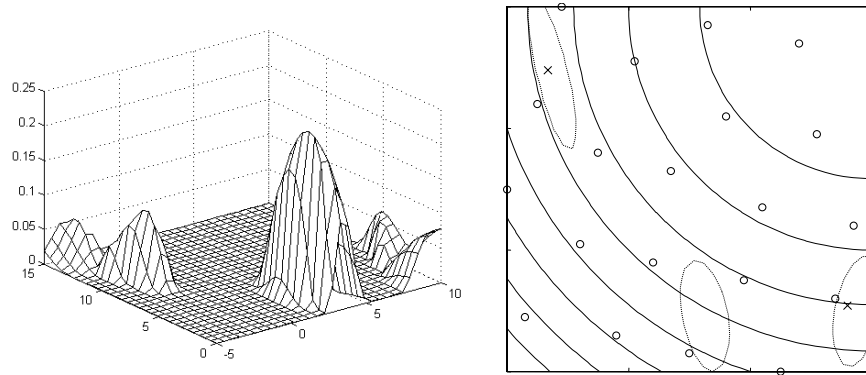


Figure 5.15: Mesh and contour plot of ISC_{14} for the newBranin example after an additional iteration. Initial sample points are shown as circles, and the two feasible samples are shown as x 's.

5.3.3 Demonstrations

In general, the proposed criteria appear to satisfy the need of finding multiple regions that are likely to be feasible in an efficient manner. We demonstrate the method below on a more difficult example, the Gomez #3 [33] test function:

$$\begin{aligned} \min f(\mathbf{x}) &= (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \\ \text{subject to: } g(\mathbf{x}) &: -\sin(4\pi x_1) + 2\sin^2(2\pi x_2) \leq 0 \end{aligned} \quad (5.11)$$

defined over the range $x_i \in [-1, 1], i = 1, 2$. Figure 5.16 shows a mesh plot of the objective and contours of the objective and constraint functions. The feasible solutions are constrained to lie within circles. The example is quite difficult to solve because the sine terms in the constraint function create 20 disconnected feasible regions. Gradient-based methods are not typically able to solve such problems efficiently because they require a large number of multistarts before the global solution can be found with any confidence. Approximation-based methods would also have difficulty with this problem because the constraint function (shown in Figure 5.17) is quite challenging to model accurately.

We first considered the constraint inexpensive and used Equation (5.6) to calculate the probability of feasibility. An initial 21 points sample yielded four feasible points. After 16 iterations with ISC₁₄, superEGO had sampled exactly one point in each of the 20 feasible regions, as shown in Figure 5.18. The initial sample points are displayed as circles and the infill samples as x's. This demonstrates that the sampling criterion has met the goal effectively. It was able to uncover feasible regions scattered throughout the design space with excellent efficiency.

Of course, the efficiency of the method is directly related to the accuracy of the kriging models of the constraint function. The next step was therefore to perform the

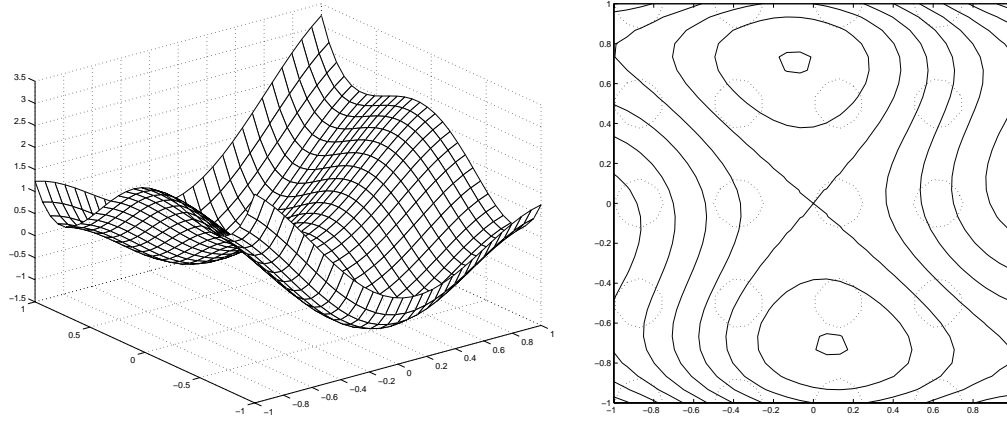


Figure 5.16: Plot of the Gomez #3 test function. The solid and dashed contours are the objective and constraint boundaries, respectively. The feasible space is inside each of the dashed circles.

same test with the constraint considered expensive. That required a kriging model to be fit to the constraint function and Equation (5.7) used to calculate the probability of feasibility. As mentioned above, modeling the highly nonlinear constraint function is quite difficult, but the proposed methodology works well in this example. The initial 21 point model of the constraint was highly inaccurate, but superEGO attempted to place points where ISC_{14} perceived the constraint likely to be satisfied. After a few iterations, the model became more accurate as the new points were sampled, and superEGO began to correctly identify new feasible regions.

Figure 5.19 shows the progression of the test as the number of iterations, n , increased. By iteration number 151, 19 of the 20 feasible islands had been sampled. It took over 300 function calls to locate the last feasible region. Note that unlike the previous test shown in Figure 5.18, there are multiple infill samples in many of the feasible regions. That is because at some iterations, there were not any regions of high probability of feasibility that were also far from the existing feasible samples. The model of the constraint could not identify the true feasible boundaries accurately

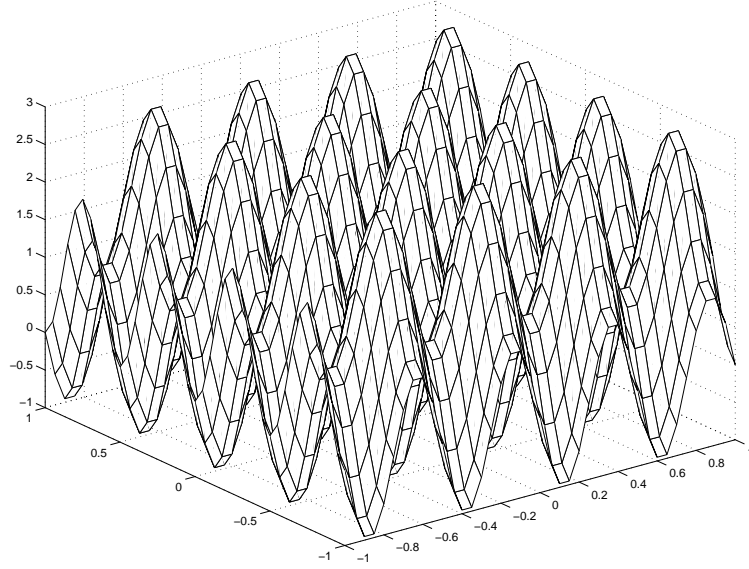


Figure 5.17: Plot of the Gomez #3 constraint function

enough to predict a high probability of feasibility in some of the unexplored feasible regions. The algorithm was therefore forced to pick points as far as it could from the known feasible points, sampling points within already identified feasible islands. However, as sampling continued, the model accuracy improved, and new feasible regions were identified.

In general, this demonstration was quite successful considering the difficulty of accurately modeling the constraint and the dependence of the probability calculation on the accuracy of that model. Due to the distance term in Equation (5.9), the method is able to spread points in the appropriate regions until the model accuracy begins guiding it towards the correct solution.

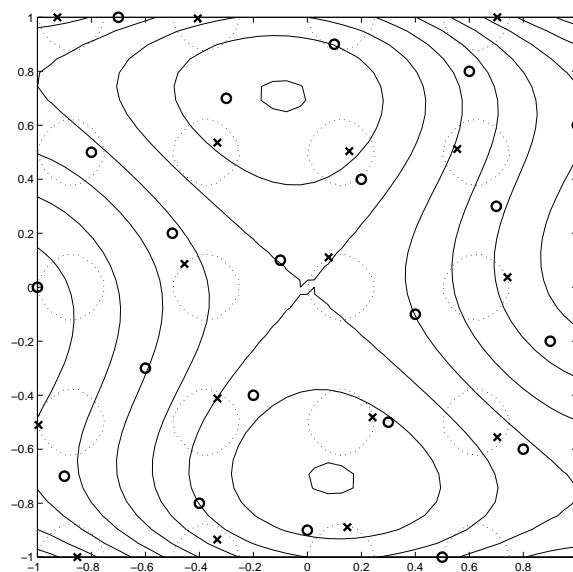


Figure 5.18: Results of ISC_{14} on the Gomez #3 example. The initial samples are shown as circles, and the infill samples as x's. Contours are of the true functions (feasible regions are inside the dashed circles).

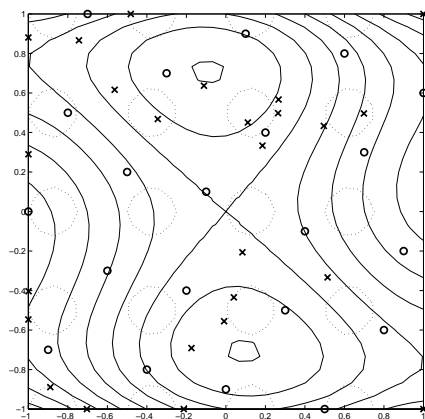
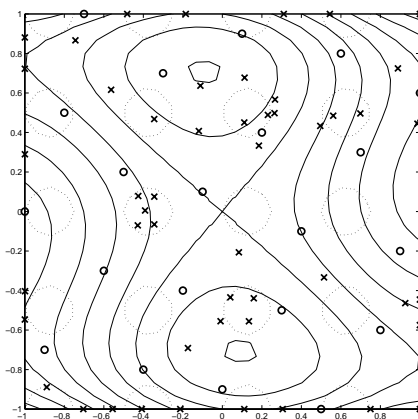
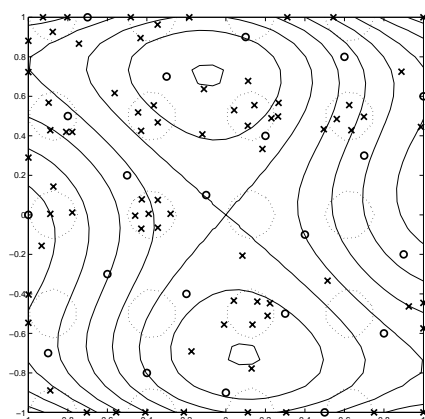
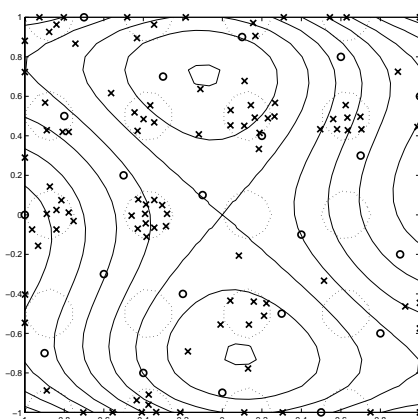
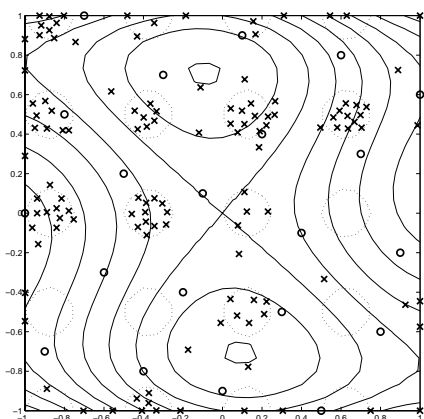
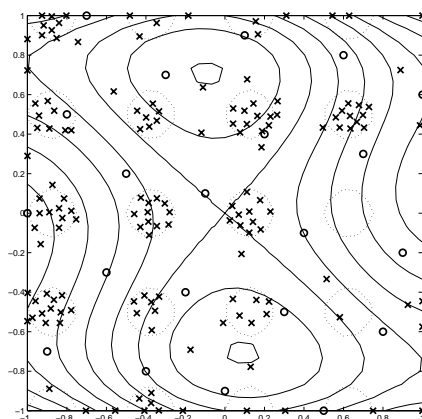
(a) $n = 25$ (b) $n = 50$ (c) $n = 75$ (d) $n = 100$ (e) $n = 125$ (f) $n = 151$

Figure 5.19: Progression of the ISC_{14} criterion on the Gomez #3 example for expensive constraint function as the number of iterations, n , increases.

5.4 Equality Constraints

All of the above discussion has been applied only to inequality constrained problems. To enable Bayesian analysis-based algorithms to solve problems including equality constraints, one approach is to bound each equality constraint as an inequality on both sides. As Audet and co-authors state [7],

However, since this paper deals with modeling of expensive simulations, it is assumed that there will be a fairly loose feasibility tolerance for equality constraints. Thus ...equalities will be considered to be inequality constraints with upper and lower bounds that happen to be somewhat close to each other.

The justification for converting an equality constraint into two inequality constraint is that the constraint need not be satisfied exactly due to the nature of the analysis models. This returns us to the discussion of constraint satisfaction metrics from Section 5.2. If the constraint boundary of an expensive simulation doesn't need to be found exactly, what difference does the choice of constraint satisfaction metric make?

As described in Section 5.2, there are three proposed constraint satisfaction metrics: the expected violation, probability of feasibility and estimated value metrics. There is reason to believe that the expected violation metric is slightly less conservative when predicting if a design point is feasible or not, and it may lead the algorithm to sample several points in the infeasible region. For some applications, there is no interest in designs that are even slightly infeasible. Thus, the expected violation metric is not recommended for general use because it may waste valuable computational resources.

For most applications then, there will be little practical difference between the estimated feasibility and probability of feasibility metrics for constraint satisfaction. The designer is satisfied as long as the search strategy finds designs that are feasible and reasonably close to the constraint boundary. This would not be the case however if the original problem contained equality constraints. Consider the example shown in Figure 5.20. A single equality constraint, $h(\mathbf{x})$ is transformed into two inequality constraints, $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$, that are somewhat close to each other (depending on the tolerance assigned by the user). Note that $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ will always be parallel to each other by virtue of their origin.

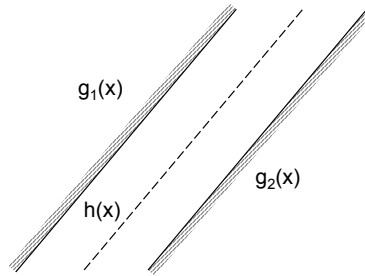


Figure 5.20: Splitting an equality constraint into two inequality constraints

Consider the effect of the constraint satisfaction metric on $g_1(\mathbf{x})$ during the solution of the ISC subproblem. A constraint satisfaction metric that is too liberal (e.g., the expected violation) may violate $g_1(\mathbf{x})$ and select candidate points too far up and to the left. A constraint satisfaction metric that is too conservative (e.g., the probability of feasibility metric) may satisfy $g_1(\mathbf{x})$, but in doing so may actually violate $g_2(\mathbf{x})$ if it places the candidate too far from the constraint boundary of $g_1(\mathbf{x})$. Therefore, a metric that is able to place iterates as close as possible to the constraint boundary is more likely to satisfy both sides of the split equality constraint. For this reason, it is recommended that the estimated value of the constraint be used as the constraint satisfaction metric for both equality and inequality constrained problems.

5.5 Constraint Activity

A constraint is called *active* if the solution to the optimization problem changes when the constraint is removed. Knowledge of which constraints are active is often useful to designers because it identifies the limiting factors on the design performance, thereby enabling them to focus their efforts. Some design optimization algorithms automatically generate information about constraint activity during their search process. Most common among these is the SQP algorithm described in Section 2.2.2. Upon termination, the Lagrange multipliers identify which constraints are active at the local optimum.

SuperEGO, like other Bayesian analysis algorithms, does not supply constraint activity information upon termination. Rather, the user may examine the values of the constraint functions at the best known design. Constraints with a value close to zero are likely to be active, although there is no theoretical guarantee of activity as with the Lagrange multipliers in SQP. It is simply the fact that the constraint is close to its limit that leads the designer to believe it may be active. An active constraint that also takes on a value of zero at the optimum is referred to as *tight*. Relying solely on the value of the constraint function at the optimum may not identify all active constraints.

Consider the example shown below. The objective function is the multipeak curve, and the constraint is shown as a vertical line, with the infeasible side hatched. The optimum, x_* , is shown as a circle. In Figure 5.21(a), the constraint is both *active*, because removing it would shift the solution to the valley on the right, and *tight*, because the solution lies exactly on the constraint boundary (i.e., $g(x_*) = 0$). However, the example in Figure 5.21(b) shows a constraint that is active, but not

tight – that is, $g(x_*) \neq 0$. Looking only at the value of $g(x_*)$ would not identify the constraint as being active.

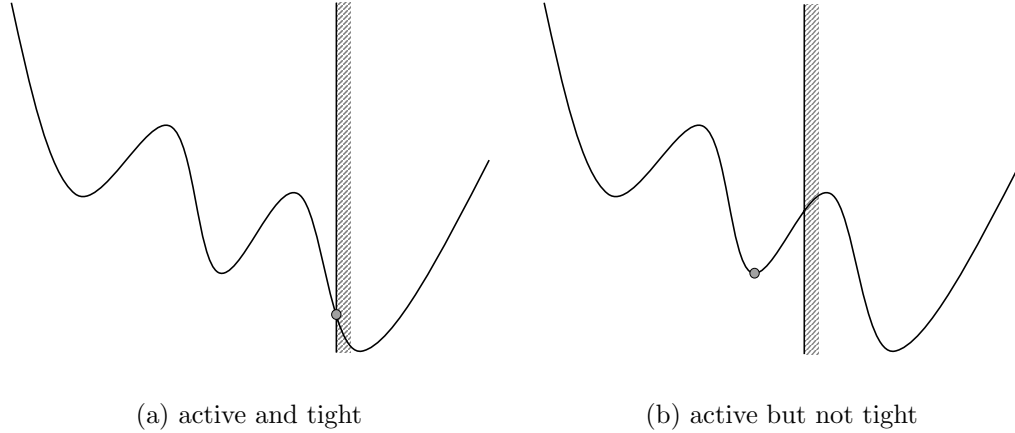


Figure 5.21: Illustration of active and tight constraints

While the example above demonstrates that using Bayesian analysis alone may incorrectly identify a constraint as inactive, active constraints in engineering design problems tend to be tight in most cases. Thus relying on the value of the constraints at the best sample point will provide the designer with the necessary information for most applications. If however, the designer is to place more confidence in identification of the active set, further work is required. One simple strategy would be to begin SQP from the best point found from superEGO. Presumably, it is near a local solution, enabling SQP to terminate efficiently. The results from SQP could then be examined to identify the active constraints from the Lagrange multipliers.

Other possibilities exist for enabling superEGO to better identify active constraints. At each iteration, the DIRECT algorithm is called to find the global solution to the ISC subproblem. It evaluates the approximate models thousands of times over the course of the optimization. The information from these searches could be analyzed to detect possibly active constraints. For example, if during all the func-

tions evaluations made by DIRECT, constraint $g_1(\mathbf{x})$ never exceeds a value of -100, there is a strong likelihood that the constraint is not active because it is satisfied everywhere that has been evaluated. Likewise, if constraint $g_2(\mathbf{x})$ was the only violated constraint for a candidate design that predicted a better objective function value than the best constrained solution, then there is a strong likelihood that $g_2(\mathbf{x})$ is active.

It is important to understand that DIRECT evaluates the kriging approximations to the constraints, not the constraints themselves. Therefore, any statements about the activity of a constraint arising from the analysis of the DIRECT search history must consider the accuracy of the models. The ideas presented here provide interesting avenues for future research that may enable Bayesian analysis algorithms to identify active constraints even if they are not tight.

5.6 Chapter Summary

This chapter discussed various approaches to handling constraints in Bayesian analysis. The approach proposed in this dissertation was to incorporate the constraints directly into the ISC subproblem. Doing so removed the problems with solution accuracy at the constraint boundary often associated with other methods. One drawback is that time consuming constrained optimization algorithms are now needed to solve the ISC subproblem. We have used a fairly efficient, robust global optimizer to satisfy this need. It was decided that the constraint satisfaction should be measured by the estimated values of the kriging models.

This chapter also demonstrated a new method for locating feasible points. This helps to overcome challenging optimization problems where the feasible region is either disconnected or relatively small. It was demonstrated that this method ap-

plied even to constraint functions that are difficult to approximate, so long as they are continuous. The chapter concluded with a brief discussion on solving equality constrained problems and identifying the active constraint set.

In the next chapter, we propose a method for incorporating inexpensive information in the ISC subproblem in order to improve the efficiency of superEGO.

CHAPTER 6

Exploiting Disparities in Function Computation Time

For approximation-based algorithms, an approximation is typically made for the objective function and each of the constraint functions. However, this may be quite inefficient if some of the functions (either objective or constraint) are *not* expensive to compute. This chapter discusses the benefits of exploiting situations where there is a large disparity in the computation time of functions within the optimization problem. By incorporating information from inexpensive functions in an intelligent way, analytical tests performed here required between 10% and 50% fewer iterations and 20% to 65% less time. A simulation-based study shown in Chapter 8 shows even greater benefits.

6.1 Motivation

In simulation-based optimization, the expense of the analysis models is often a prime concern. While surrogate modeling algorithms can reduce computational time, not all such algorithms can exploit differences in the computational costs of functions in the problem statement. In previous work by Nelson and Papalambros [64], inexpensive functions were used directly in a trust region algorithm, resulting in

fewer expensive analyses. This provides motivation to extend the work to surrogate model-based optimization algorithms such as EGO.

Originally, EGO was intended for problems where all the functions were expensive. The approach here does not have such limitations. It is not always clear whether to classify a simulation response as expensive or inexpensive, nor is it always feasible to segregate the two classes of functions into independent simulation runs. However, if one can do this, then eight types of well-defined optimization problems exist for the various combinations of expensive and inexpensive objective and constraint functions (see Figure 6.1).

EGO was originally designed for type 1 or 2 problems. The changes proposed here enable EGO to differentiate between all eight problem types, including the special case where all functions are considered inexpensive (types 3 and 4). Because DIRECT [43], another global optimization algorithm, is used to solve the ISC subproblem, EGO degenerates into DIRECT for those problem types. This chapter focuses on the more interesting cases of types 5 through 8, where both inexpensive and expensive functions exist. We propose using information from the inexpensive functions to avoid sampling the expensive functions when it is not beneficial to do so. The methods for making such decisions are referred to as filters. Table 6.1 lists the eight problem types and the method that shall be used to solve each. Expanding EGO to handle the entire array of possibilities allows the designer to solve the optimization problem efficiently, regardless of the cost of the functions. As mentioned in the Introduction, we have chosen the name *superEGO* for our algorithm because it is able to handle a *superset* of the types of problems that EGO was originally designed for.

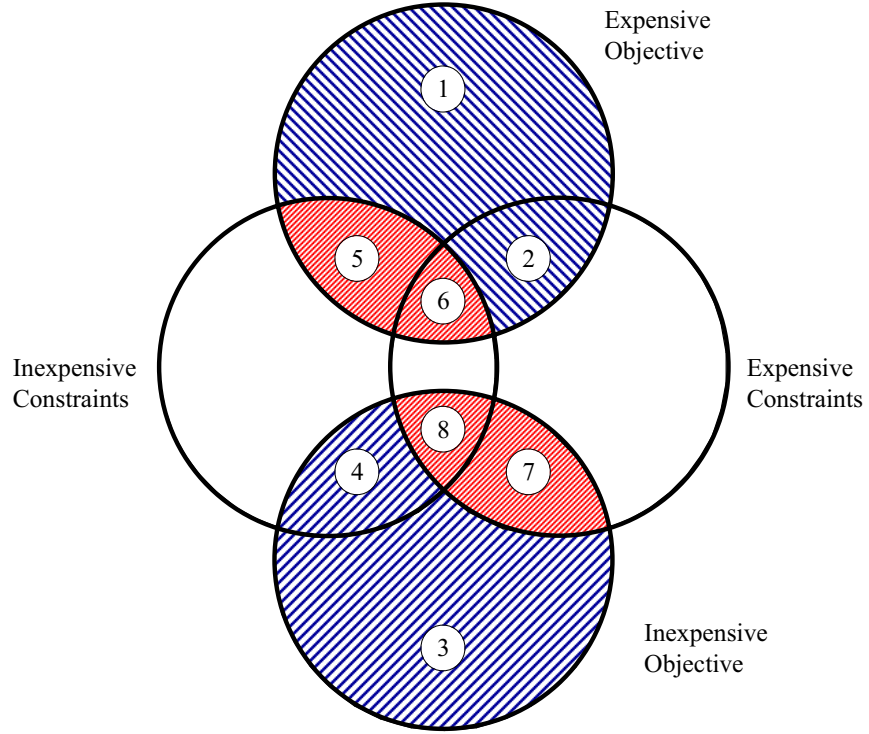


Figure 6.1: Possible Problem Statement Types

6.2 Filter Method

As explained in the previous chapter, our work takes advantage of the fact that a constrained ISC problem can be solved with a global searching, derivative-free algorithm, namely DIRECT. Using the estimated values of the constraint functions as the constraint satisfaction metric, the ISC problem of superEGO is solved as

$$\begin{aligned} \min f &= ISC(\mathbf{x}) \\ \text{subject to: } \hat{g}_i(\mathbf{x}) &\leq 0, \forall i \end{aligned} \tag{6.1}$$

where $\hat{g}_i(\mathbf{x})$ refers to the kriging model prediction of the i^{th} constraint at the candidate infill point \mathbf{x} .

Information from any inexpensive functions is used here to prevent suboptimal points from being evaluated by the expensive true functions. This is done by imposing

Table 6.1: Problem types and associated solution strategies

Type	Objective Function	Constraint Functions	Solution Strategy
1	Expensive	None	Unconstrained EGO
2	Expensive	Expensive	Constrained EGO
3	Inexpensive	None	Unconstrained DIRECT
4	Inexpensive	Inexpensive	Constrained DIRECT
5	Expensive	Inexpensive	filter type I
6	Expensive	Expensive & inexpensive	filter type II
7	Inexpensive	Expensive	filter type III
8	Inexpensive	Expensive & inexpensive	filter type IV

two additional constraints on the ISC problem of Equation (6.1). The first checks if any inexpensive constraint functions are infeasible, and the second checks if an inexpensive objective function is worse than the current best design point. The original constraint of keeping the predicted value of the expensive constraints feasible is retained as well. Thus the filter method changes the ISC problem to the following:

$$\begin{aligned}
 \min f &= ISC(\mathbf{x}) \\
 \text{subject to: } &\hat{g}_i(\mathbf{x}) \leq 0, i = 1, \dots, n_{expc} \\
 &g_j(\mathbf{x}) \leq 0, j = 1, \dots, n_{inexpc} \\
 &f_{inexp}(\mathbf{x}) \leq f_{min}
 \end{aligned} \tag{6.2}$$

where n_{expc} and n_{inexpc} are the numbers of expensive and inexpensive constraints, respectively, f_{inexp} is the inexpensive objective function of the original optimization problem, and f_{min} is the best feasible point found thus far. There are four filter types based on which kinds of functions exist as shown in Table 6.1.

6.3 Analytical Examples

The effectiveness of the filter method was evaluated for a suite of two-dimensional analytical examples. In the figures below, the feasible regions are shaded, and the

optima are shown as a filled circle. The variable bounds and solutions are summarized in Table 6.2.

Table 6.2: List of analytical examples

Example no.	Variable Bounds	Solution
1	$x_i \in [0, 5] \ \forall i$	$f(2.7450, 2.3523) = -1.1743$
2	$x_i \in [0, 5] \ \forall i$	$f(3.0716, 2.0961) = -0.5503$
3	$x_1 \in [-3, 3], \ x_2 \in [-1.5, 1.5]$	$f(1.7476, 0.8738) = 0.2986$
4	$x_i \in [-2, 2] \ \forall i$	$f(0.5955, -0.4045) = 5.6694$
5	$x_i \in [0, 1] \ \forall i$	$f(0.2017, 0.8332) = -0.7483$

6.3.1 Example 1: Constrained mystery function

Example 1 uses the mystery function in Equation (4.12) with an additional constraint (see Figure 6.2). The constraint is tight at the optimum, i.e., it is satisfied as an equality at the solution.

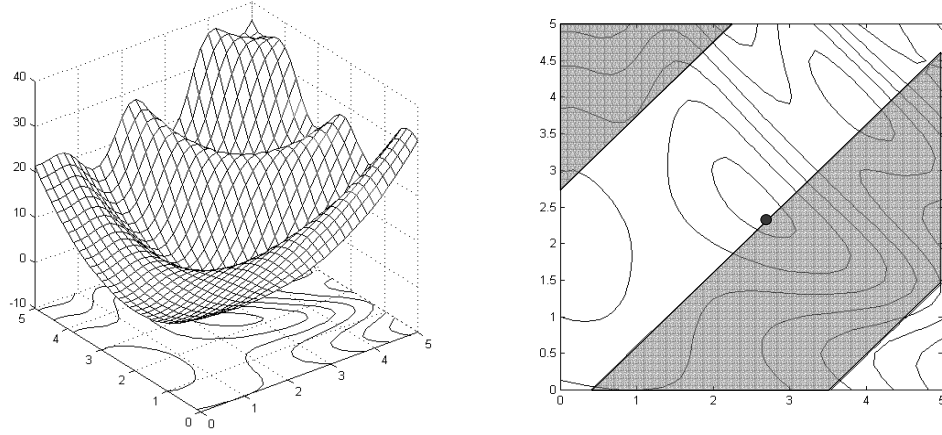


Figure 6.2: Example 1 with feasible region shaded, optimum shown as filled circle

$$\begin{aligned}
 \min f(\mathbf{x}) &= 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + \dots \\
 &\quad 7 \sin 0.5x_1 \sin 0.7x_1x_2 \\
 \text{subject to: } g(\mathbf{x}) &: -\sin(x_1 - x_2 - \frac{\pi}{8}) \leq 0
 \end{aligned} \tag{6.3}$$

6.3.2 Example 2: Reverse constrained mystery function

Example 2 switches the objective and constraint functions of Example 1 (see Figure 6.3). The feasible design space is relatively small and the constraint is tight at the solution.

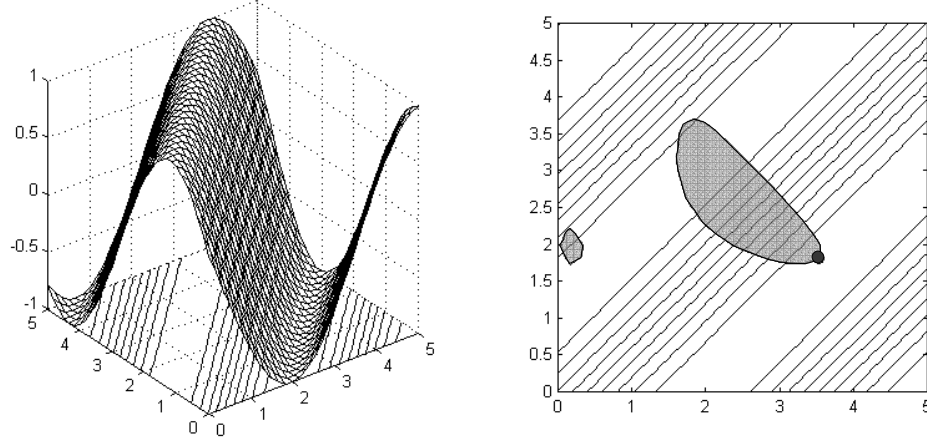


Figure 6.3: Example 2 with feasible region shaded, optimum shown as filled circle

$$\begin{aligned}
 \min f(\mathbf{x}) &= -\sin\left(x_1 - x_2 - \frac{\pi}{8}\right) \\
 \text{subject to: } g(\mathbf{x}) &: -1 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + \dots \\
 7 \sin 0.5x_1 \sin 0.7x_1x_2 &\leq 0
 \end{aligned} \tag{6.4}$$

6.3.3 Example 3: Three-hump camelback function

Example 3 is the three-hump camelback test function of Hardy [37] modified by adding a constraint (see Figure 6.4). In this case, the constraint is active, but not tight. In other words, removing the constraint changes the location of the optimum, but the solution does not lie along the constraint boundary.

$$\begin{aligned}
 \min f(\mathbf{x}) &= 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2 \\
 \text{subject to: } g(\mathbf{x}) &: (3 - x_1)^2 + (1 - x_2)^2 - 3 \leq 0
 \end{aligned} \tag{6.5}$$

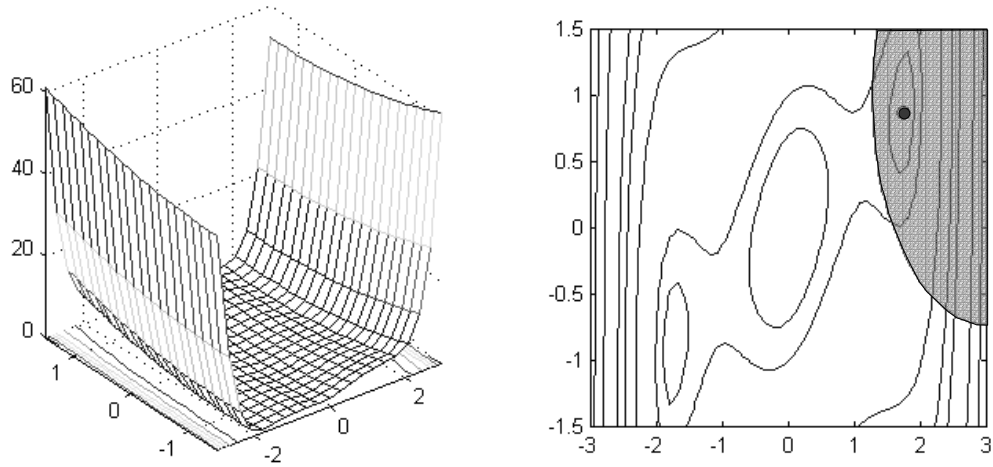


Figure 6.4: Example 3 with feasible region shaded, optimum shown as filled circle

6.3.4 Example 4: Goldstein-Price function

Example 4 is the Goldstein and Price test function [73], modified by adding two additional constraints. In order to improve the ability to model the objective function, a log-transform was used on the data as suggested by Schonlau [83]. Figure 6.5 shows the transformed function.

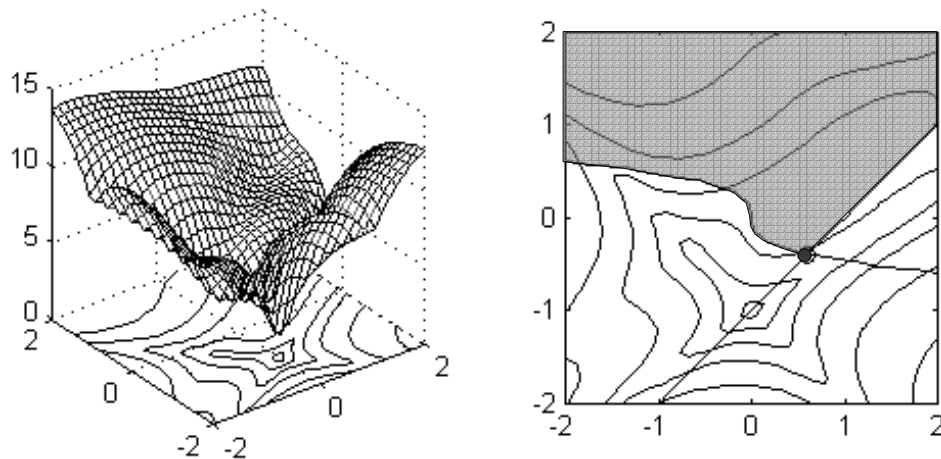


Figure 6.5: Example 4 with feasible region shaded, optimum shown as filled circle

$$\begin{aligned}
\min f(\mathbf{x}) &= (1 + A(x_1 + x_2 + 1)^2)(30 + B(2x_1 - 3x_2)^2), \\
\text{where } A &= 19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2, \\
\text{and } B &= 18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2 \\
\text{subject to: } g_1(\mathbf{x}) &: -3x_1 + (-3x_2)^3 \leq 0 \\
g_2(\mathbf{x}) &: x_1 - x_2 - 1 \leq 0
\end{aligned} \tag{6.6}$$

6.3.5 Example 5: Test function #2

Example 5 is the test function #2 of Equation (5.4), shown again in Figure 6.6 for convenience.

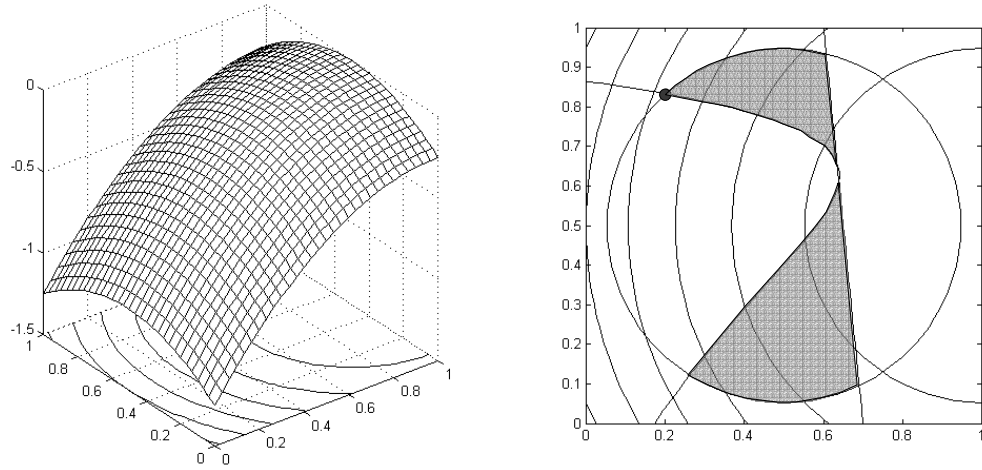


Figure 6.6: Example 5 with feasible region shaded, optimum shown as filled circle

$$\begin{aligned}
\min f(\mathbf{x}) &= -(x_1 - 1)^2 - (x_2 - 0.5)^2 \\
\text{subject to: } g_1(\mathbf{x}) &: (x_1 - 3)^2 + (x_2 + 2)^2 e^{-x_2^7} - 12 \leq 0 \\
g_2(\mathbf{x}) &: 10x_1 + x_2 - 7 \leq 0 \\
g_3(\mathbf{x}) &: (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.2 \leq 0
\end{aligned} \tag{6.7}$$

6.4 Results

To reduce the initial sample locations' influence on the results, the examples were run using a series of 50 initial 21-point (11-point in the case of example 5), random Latin Hypercube design of experiments (DOE). Using the same initial DOE's, superEGO was run for the various filter types by making different choices as to which functions were considered inexpensive. Results labelled as coming from EGO simply refer to those where all functions were considered expensive. In all cases, our own superEGO algorithm was used to gather the data.

For each example, the algorithm was stopped once it located a feasible point within 5% of the global minimum value. EGO and each filter type were then compared with respect to clock time and the number of iterations required (one function evaluation per iteration). The kriging model parameters were held fixed for all iterations so that the choice of model fitting frequency did not impact the study. Only examples 4 and 5 ran filter types II and IV because those filters require multiple constraints. Two tests for each of those filter types were run by swapping the constraints considered to be expensive.

With a large amount of data gathered for five examples, 50 optimizations each, the results for each approach are shown as the average across all examples and initial experimental designs. For the following three tables, the results of superEGO are categorized as better, the same, or worse according to how many function evaluations were required compared to EGO for a given optimization run. Table 6.3 summarizes the breakdown of these categories.

There are several trends worth noting. For one, superEGO is *not* systematically better than EGO, yet is at least as good in the vast majority of cases. Also, filter

Table 6.3: Summary of comparisons to EGO (in percentage) for each category

Comparison	filter type I	filter type II	filter type III	filter type IV
better	59.2	52.0	59.2	74.5
same	30.0	40.5	36.8	24.0
worse	10.8	7.5	4.0	1.5

types III and IV are worse than regular EGO less often than filter types I and II. This is most likely due to the fact that the former consider the objective function inexpensive. Using this information forces superEGO to find feasible points at least as good as the current best at each iteration, thereby reaching good solutions more quickly than for filter types I or II.

Tables 6.4 and 6.5 show the relative improvement (or worsening) in the number of iterations and time required to find the optimum. For each filter type, the average difference is shown alongside the average standard deviation across examples. The overall averages for each filter are weighted by the percentages of occurrence in each category as shown in Table 6.3. From Tables 6.4 and 6.5 one can observe that, on average, superEGO outperforms EGO for any given Filter Type, both in terms of function evaluations and time required. Also, filter types III and IV have both higher relative improvement and lower standard deviations for both time and iterations required. This is another indication that exploiting the information from an inexpensive objective function can consistently lead to significantly improved efficiency.

6.5 Discussion

This research attempted to quantify the gains in efficiency to a surrogate model optimization algorithm made by exploiting information from the inexpensive func-

Table 6.4: Relative improvement (in percentage) over EGO in iterations required

Fn Call Comparison	filter type I		filter type II		filter type III		filter type IV	
	avg	std dev	avg	std dev	avg	std dev	avg	std dev
better	50.7	14.7	40.1	16.6	60.7	18.8	63.6	15.9
worse	-62.0	62.6	-151.2	229.0	-68.3	27.1	-44.4	7.9
overall	23.3	15.5	9.5	25.8	33.2	12.2	46.7	12.0

Table 6.5: Relative improvement (in percentage) over EGO in time required

Fn Call Comparison	filter type I		filter type II		filter type III		filter type IV	
	avg	std dev	avg	std dev	avg	std dev	avg	std dev
better	63.7	11.6	49.2	14.6	72.9	14.8	77.0	10.9
same	18.6	17.7	15.1	5.6	19.2	3.7	33.2	4.3
worse	-23.3	49.4	-126.0	231.8	-62.3	67.1	8.1	11.7
overall	40.7	17.5	22.3	27.3	47.7	12.8	65.4	9.3

tions. By adding constraints to the ISC subproblem of Equation (6.1), the algorithm avoids evaluating locations known to be suboptimal. While this does not guarantee that the solution will be found more quickly, it tended to perform more efficiently most of the time. For the analytical tests, direct use of inexpensive functions decreased the number of required iterations by between 10% and 50% and the required time between 20% and 65%.

The results of this study point to new avenues of increased efficiency. The filter constraints used here are not the only kind one can use. For example, one may want to use information from the expensive functions to determine if a candidate infill sample location is in an area of poor model reliability. For some iterations, the desire to improve the expensive model functions locally could override the desire to sample only feasible and/or better locations. Our framework is generic enough to allow for such a filter. Another promising avenue is the ability of these filters to serve as convergence criteria, a current weakness of EGO. For example, one could

constrain the ISC problem to improve upon the best known point according to some statistical confidence interval such as the lcb function proposed by Cox and John [21] (see Chapter 4). Failure to find a feasible solution to the ISC problem would thus terminate the algorithm.

6.6 Chapter Summary

This chapter proposed that inexpensive information from the optimization problem should be directly incorporated into the ISC subproblem. That information could take the form of a feasibility check using the constraints that are quick to evaluate, or an objective function improvement check if the objective is quick to evaluate. This additional information was shown to produce a significant savings in the time required to find a good solution.

In the next chapter, we look at a variety of implementation details that to some extent impact the superEGO algorithm.

CHAPTER 7

Bells and Whistles

A variety of issues pertaining to the actual implementation of superEGO are discussed below. While none of them are critical research topics of this dissertation, it is still beneficial to explain the details of the heuristics. Specifically, we describe the methods for solving the auxiliary optimization problems (the ISC subproblem and fitting the kriging models) and the selection of the sampling criterion and stopping rule.

7.1 Locating the Infill Sample

At each iteration, superEGO must solve the ISC subproblem, an auxiliary optimization problem. The problem tends to be multimodal and may be constrained even if the original problem is unconstrained (e.g., if the filter method of Chapter 6 is applied). The ISC functions – either analytical expressions or evaluations of a kriging model – are inexpensive to evaluate. In short, locating the next iterate requires a good global optimization algorithm.

Some researchers have applied strategies where the choice of the next iterate is restricted to a predetermined set of candidate points [21], [95]. The sampling criterion is evaluated at the candidate locations, and the best observed value chosen

as the next iterate. The approach used in this thesis is to use continuous optimization techniques, thereby allowing the next iterate to be anywhere within the design space. While our approach is much more general, it is more difficult to implement.

The DIRECT algorithm of Jones [40], [43] has been selected to solve the ISC subproblem. In order to complete the psychoanalysis trio of the EGO and superEGO, the ISC DIRECT step of the process is affectionately referred to as the *ID* [31]. A Matlab-based implementation of the DIRECT algorithm known as UMDIRECT was recently developed by this author and other graduate students at the University of Michigan. It works both for constrained and unconstrained problems and was used for the results in Chapters 3, 4, 9 and Chapter 5. Results for Chapters 6 and 8 were gathered using another Matlab implementation of DIRECT from the Tomlab research group in Sweden [38].

Simply put, DIRECT works by subdividing the design space into hyper-rectangles at each iteration. The decision of which hyper-rectangles to divide is based upon Lipschitz theory and has been generalized to constrained optimization. The constraint functions for the ISC subproblem may take on any of four possible forms: kriging models of the expensive constraints of the main design problem, direct evaluation of the inexpensive constraints from the main design problem, analytical constraints imposed by the filtering method described in Chapter 6, or any special constraints that are specific to the infill sampling criterion itself.

In some cases, a local search step was added to UMDIRECT as suggested by Jones [40]. Doing so helps to overcome the slow local convergence of DIRECT. Matlab's constrained SQP algorithm, `fmincon`, was used for the local search steps [19]. However, this author has experienced occasions where `fmincon` went into an infinite loop for unknown reasons. Such an occurrence prevented superEGO from

continuing; and therefore, most results were gathered with the local search feature disabled. In the future, a more robust local search algorithm will be incorporated into UMDIRECT to increase its efficiency.

For the implementation of DIRECT used in the ID portion of superEGO, the algorithm stops once it has performed either 50 iterations or $200*d$ function evaluations, where d is the number of design variables, whichever occurs first. However, the user has the option of changing those limits. These are the same stopping rules applied when DIRECT is called to fit the kriging models. Additional information on the University of Michigan implementation of DIRECT may be found in the Appendix.

7.2 Model Fitting Frequency

The frequency of model fitting has a significant impact on the efficiency of superEGO. After fitting the covariance model from the initial data sample, the question arises of whether or not to refit the model at each iteration. On one hand, refitting the covariance parameters may lead to more accurate kriging models, and therefore better results at each iteration. On the other hand, fitting the models is a time consuming process that may dominate the computational time of the iteration if the functions in the problem statement aren't extremely costly.

To demonstrate this tradeoff, Figure 7.1 shows the breakdown of the computational costs for a simulation-based optimization run using superEGO. The example is a 3 variable problem with an expensive objective function and 12 expensive constraints. The function evaluation time is fairly consistent at around 70 seconds. The time required to solve the ISC subproblem increases approximately linearly with the number of sample points and ranges between 100 and 300 seconds. The time for

fitting models is relatively insignificant when existing parameters are used, but every 10 iterations, the covariance model parameters are refit via MLE. Refitting covariance models requires inverting the $n \times n$ covariance matrix, where n is the number of sampled points, for every iteration of the fitting procedure. On a Pentium IV, 1.7 GHz personal computer, the cost of fitting all 13 covariance models is between 30 minutes and two hours and increases linearly with the number of sample points.

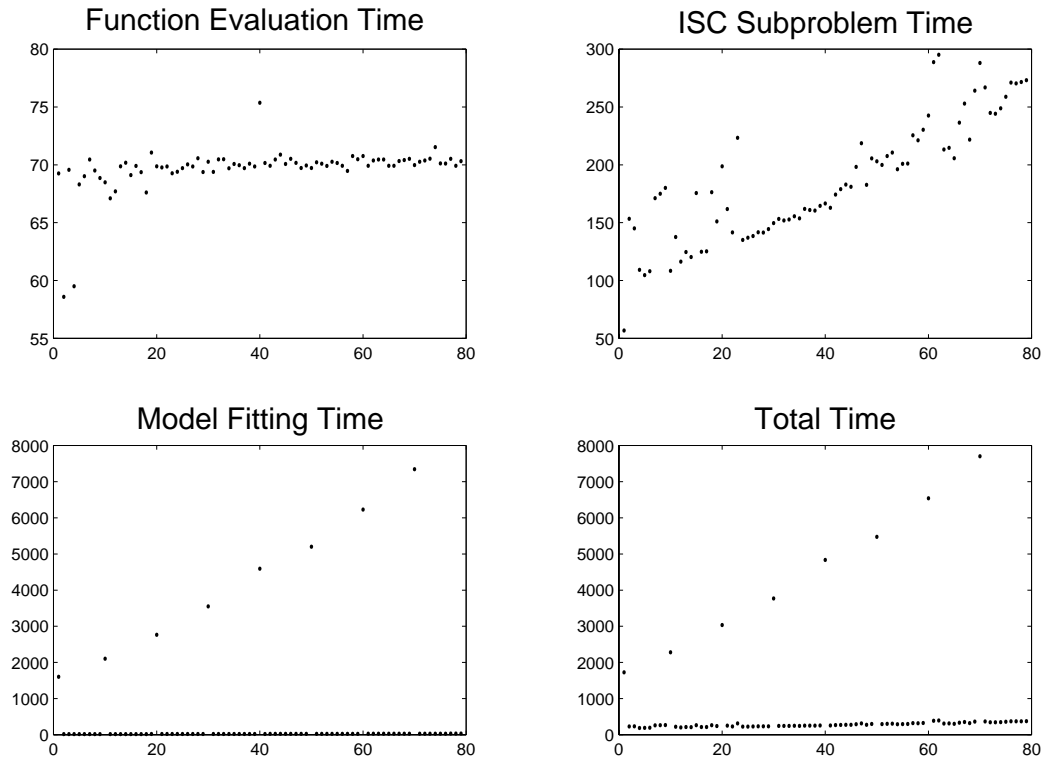


Figure 7.1: Breakdown of the computational costs per iteration (times on the y-axis are in seconds)

When looking at the overall costs per iteration, it is clear that the model fitting steps dominate the computation costs. Therefore, the user must be careful in deciding how frequently to refit the covariance models. For examples such as this where model fitting can exceed two hours, it is not practical to refit models more than a few times during the optimization. Otherwise, the cost of fitting models far exceeds performing

an exhaustive search of the design space using the true functions.

Ideally, the model parameters would start to stabilize towards a set of values as the iterations progress. At some point, the program would stop refitting the models and keep those parameter values for the rest of the optimization. However, the experience of this author has been that even small changes in the data set can lead to large changes in the fitted model parameters. This has happened even when the algorithm used to fit the data was the gradient-based SQP algorithm started from the previous model values or the global searching DIRECT algorithm. In our approach, we refit the models with UMDIRECT every 10 iterations unless the user specifies otherwise.

7.3 Intelligent Sampling Strategies

Throughout most of the discussion in this dissertation, a single sampling criterion has been used during the optimization. However, a more general approach is to let superEGO itself select the sampling criterion as iterations progress. Some heuristic strategies have been implemented and are described below.

Oftentimes, Bayesian analysis approaches have difficulty in refining the local solutions. To combat this problem, superEGO does a local search every 10 iterations to find the best point on the approximate models, as suggested in Chapter 4. During the local search, superEGO uses the predicted value of the constraint functions as the ISC subproblem constraints. Due to modeling error, these may or may not be true local optima, but at least it guides the algorithm more efficiently towards those points. This feature was disabled for many of the examples shown here because we were demonstrating specific search strategies. Using the periodic local search would have skewed the interpretation of their behavior.

SuperEGO also looks for special cases before beginning the ISC subproblem. First, it checks if a feasible point has been found. If not, then it begins using the probability of feasibility criterion (see Section 5.3) until a feasible point has been found, at which point it switches back to the criterion originally selected. The algorithm also checks if the objective function is inexpensive to compute and if the user has selected one of the improvement-based criteria (i.e., expected improvement, generalized expected improvement, cool criterion, or Kushner’s criterion). If both conditions are true, then superEGO uses the actual value of the objective function as the sampling criterion rather than estimating the improvement with a kriging model. In this way, it guides the iterations more accurately by avoiding the use of approximate models.

Another intelligent search strategy that has been implemented is for solving problems with disconnected feasible regions. As with other search strategies, superEGO begins by searching for an initial feasible point. Once a feasible point is obtained, it switches to a local search mode to refine this local optimum. After the change in the location of the next iterate has become sufficiently small (less than 0.1% of the design space range for three consecutive iterations), it switches back to ISC_{14} described in Section 5.3 to search for new feasible regions. If another feasible point is found sufficiently far away (greater than 5% of design space range) from the nearest feasible sample, another local search is launched. The user may override the default the conditions for switching between local search and feasible region search. In addition, the user may specify that superEGO not sample new feasible regions unless $\hat{y}(\mathbf{x}) - b\sigma(\mathbf{x}) \leq f_{min}$ if the objective function is expensive or unless $y(\mathbf{x}) \leq f_{min}$ if the objective is inexpensive.

Search strategies such as those described above are automatically implemented

by choosing the appropriate sampling criterion (see Appendix). Some relatively sophisticated search methods are easy to apply. More advanced users with an understanding of Matlab can also define custom search strategies relatively easily. In short, the flexibility of the framework allows for superEGO to perform a large variety of complicated search strategies.

7.4 Termination Criteria

As discussed in Section 2.3.5, few stopping criteria for Bayesian analysis algorithms exist in the open literature. The most prevalent is to terminate the algorithm once the number of function evaluations has exceeded some specified limit. Another somewhat common criterion is to stop once the expected improvement from further sampling drops below a user-defined threshold. Of course, for this criterion to be meaningful, one of the improvement-based criteria must be used.

A third stopping rule comes from the lower confidence bounding (lcb) function of Cox and John shown in Equation (2.14). At each iteration, the sampling criterion is evaluated at a fixed set of prediction sites and the best point is chosen. In other words, the ISC subproblem is solved from a finite set of candidates. Their algorithm terminates if the objective function value of the best sample point is below the best value of the lcb function among the prediction sites. Since our algorithm does not restrict the iterates to a finite set of candidates, it is difficult to determine if the global minimum of the ISC subproblem has been found. Incorporating the lcb stopping rule within superEGO would require DIRECT to run for a larger number of iterations to have more confidence that the best value of the lcb has indeed been found. Failure to do so may result in premature termination of superEGO. Evaluating the lcb function at a set of prediction sites could however be done after solving the ISC subproblem,

regardless of which sampling criterion was chosen.

In short, the expected improvement termination rule restricts the user to a subset of the available sampling methods, and the lcb function termination rule is computationally inefficient for this framework. Therefore, superEGO mainly relies on the function evaluation limit for termination. That said, there are other termination rules that have been implemented in our framework as well.

First of all, there are special cases besides the function evaluation limit where the algorithm automatically terminates. In some cases, the user may be able to define a goal for the objective function value, i.e., a value below which they are indifferent to further improvement. If an improvement-based sampling criterion is chosen, the goal can be used to select the target for improvement [93] (e.g., the value of ϵ in Equation (2.10)). However, it also provides a simple stopping rule and has been incorporated into superEGO. Another special case occurs when all of the functions in the problem are labelled “inexpensive”. As discussed in Chapter 6, superEGO degenerates into DIRECT for such an event. Therefore, one iteration of superEGO should provide a satisfactory solution from the DIRECT algorithm, and superEGO terminates.

Two other stopping rules are currently implemented in superEGO. The first stops the algorithm if N feasible design points have been found. By default, $N = 1$, but the user may define any number they wish. This criterion is useful for situations where superEGO is applied to identify a feasible starting point for a different algorithm. The second stopping rule terminates the algorithm if the last N iterations have produced sample points that are all located within tol percent of each other, where the distance between points is measured as a percentage of the total range of the design space. By default, $N = 3$ and $tol = 0.1\%$, but the user is free to specify their values. This stopping rule is useful for cases where the search tends to converge upon

a solution rather than scattering points throughout the design space.

While many of the stopping rules described in the above two paragraphs are only useful for certain applications, they all have two desirable qualities: they are easy to evaluate, and they are meaningful regardless of the sampling criterion chosen, making them well suited for the flexible superEGO framework. Further research will hopefully provide new stopping rules that have these qualities but are less heuristic in nature.

7.5 Chapter Summary

In this chapter, several implementation details of superEGO were described. It is hoped that some of the discussion here may spark future research. In the following chapters, we present two case studies that illustrate the benefits of the increased flexibility and efficient constraint handling.

CHAPTER 8

Vehicle Product Platform Design Study

In this chapter, a simulation-based case study is presented using a vehicle system analysis package known as ADVISOR [63]. The purpose of this study is to provide empirical evidence that taking the cost of the functions into consideration can significantly reduce the time required to solve a simulation-based optimization problem.

8.1 Problem Description

The terms *product platform* or *product family* refer to a group of artifacts that have been designed such that some of their components are identical. An example would be a line of handheld power tools. While the shape and purpose of each tool may vary within a given family of products, they often use the same motor. A product platform benefits the product costs by reducing design time, inventory costs, and manufacturing costs. Product platform design problems provide interesting Pareto optimality studies and have been explored for a variety of applications [29], [65].

This study presents the design of a family of vehicles. Two vehicles – premium compact (PC) and lower midsize (LM) vehicles – are to share a common engine while having independent final drive ratios. Sharing the engine is a cost effective way to

create a product family, and keeping the final drive (a simple component compared to the engine) separate allows for the two vehicles to be “tuned” differently. The premium compact is designed to maximize fuel economy while meeting six performance constraints. The lower midsize is a slightly larger vehicle designed to minimize 0–60 mph acceleration time while meeting 25 mpg fuel economy and performance constraints 10% more demanding than for the premium compact. This results in a three variable design problem with a bicriterion objective ($w_{PC}f_{PC} + w_{LM}f_{LM}$) and 12 constraints in addition to the simple bounds.

In order to scale the objectives of the two vehicles better, a transformation based on utility theory was applied. Three parameters – the baseline, ideal, and critical points – define the *value curve*, a unitless measure of a product’s worth. These parameters were chosen according to market data on existing designs. For the premium compact’s fuel economy and the lower midsize’s 0–60 time, the actual performance predicted by ADVISOR was transformed to a value roughly between 0 and 1. The value curves for the two objectives in this problem are shown in Figure 8.1. Equation (8.1) summarizes the formal statement of the optimization problem.

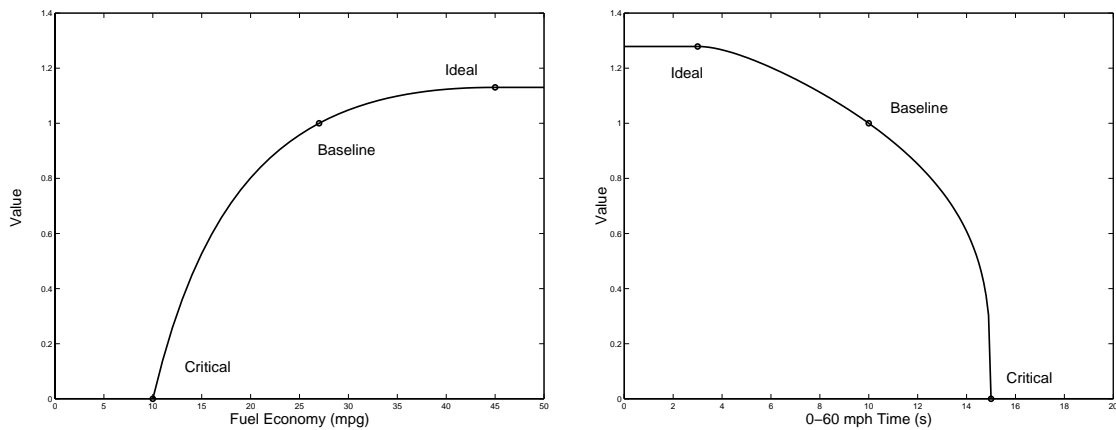


Figure 8.1: Value curves for the two objectives

$$\begin{aligned}
& \text{minimize} && w_{PC} \cdot \text{value}(\text{fuel economy}_{PC}) + w_{LM} \cdot \text{value}(t_{0-60\ LM}) \\
& \text{with respect to:} && \mathbf{x} = \{\text{engine size, final drive}_{PC}, \text{final drive}_{LM}\} \\
& \text{subject to:} && g_1 : t_{0-60\ PC} \leq 12\ \text{s} \\
& && g_2 : t_{0-85\ PC} \leq 24\ \text{s} \\
& && g_3 : t_{40-60\ PC} \leq 6\ \text{s} \\
& && g_4 : 5\text{-sec distance}_{PC} \geq 100\ \text{ft} \\
& && g_5 : \text{max acceleration}_{PC} \geq 0.5\ \text{g's} \\
& && g_6 : \text{max speed}_{PC} \geq 90\ \text{mph} \\
& && g_7 : \text{fuel economy}_{LM} \geq 25\ \text{mpg} \\
& && g_8 : t_{0-85\ LM} \leq 21.6\ \text{s} \\
& && g_9 : t_{40-60\ LM} \leq 5.4\ \text{s} \\
& && g_{10} : 5\text{-sec distance}_{LM} \geq 110\ \text{ft} \\
& && g_{11} : \text{max acceleration}_{LM} \geq 0.55\ \text{g's} \\
& && g_{12} : \text{max speed}_{LM} \geq 99\ \text{mph} \\
& && g_{13}, g_{14} : 50\ \text{kW} \leq \text{engine size} \leq 120\ \text{kW} \\
& && g_{15}, g_{16} : 3 \leq \text{final drive ratio}_{PC} \leq 4.5 \\
& && g_{17}, g_{18} : 3 \leq \text{final drive ratio}_{LM} \leq 4.5
\end{aligned} \tag{8.1}$$

where w_{PC} and w_{LM} are the Pareto weights in the multicriterion objective. The metric t_{0-60} refers to the 0 to 60 mph acceleration time (smaller is better). The five second distance is the distance the vehicle can travel from a dead stop in five seconds (larger is better). The maximum acceleration and speed metrics are the best performance the vehicle is able to produce, which must meet minimum standards. The values chosen for the constraint bounds are derived from the performance of the baseline vehicle model.

The simulation chosen to predict the vehicles' performance is known as the AD-

Vanced vehIcle SimulatOR (ADVISOR) [63]. Developed by the National Renewable Energy Laboratory (NREL), ADVISOR version 3.2 was released in August, 2001. One reason for the choice of simulation is public availability and reported work in the open literature, e.g., on the usefulness of ADVISOR in powertrain design [24] and on the validation of the simulation through the PNGV-sponsored FutureCar competition [85].

ADVISOR is a Matlab-based program which makes use of Matlabs Simulink toolbox to describe a physics-based model. It is a feed-backward, quasi-steady model of a total vehicle. Feed-backward, means that a driving cycle is specified first (e.g., the federal urban driving schedule, or FUDS), and then the simulation computes the load required at each of the power components such that the vehicle traces the prescribed driving schedule. Steady-state, means that ADVISOR approximates the continuous behavior of the vehicle by discrete time steps during which transient effects such as the rotational inertia of drivetrain components are ignored. The component models in ADVISOR are empirical, relying on input/output relations measured in the laboratory, and quasi-static, using data collected in steady state tests, corrected for transient effects. With steady state assumptions, efficiency tables can be used to trace back the power requirement from the wheels to the power components. For the speed and load at any time step, the instantaneous fuel consumption rate is found from a look-up table for the engine. The simulation then integrates over the entire driving schedule to compute any required information.

On a 550 MHz Pentium III computer, each vehicle analysis requires approximately 30 seconds for fuel economy and 7 seconds for performance calculations. Note that all performance metrics are computed at once. Because the performance constraints were too expensive relative to the fuel economy, they were replaced with

highly accurate spline models. Using the spline models as the “true” functions, five optimization studies were performed by varying the weights w_{PC} and w_{LM} between 0 and 1 such that $w_{PC} + w_{LM} = 1$. In addition, two optimization runs were performed to locate the so-called *null platform* where each vehicle is optimized independently. Filter type II was used for superEGO on all cases except for $w_{PC} = 0$ where filter type IV was used (see Chapter 6).

8.2 Results

The competing vehicle objectives are shown in the Pareto plot of Figure 8.2. Better designs occur up (better premium compact vehicle fuel economy) and to the left (better lower midsize vehicle acceleration). The null platform point, shown in the upper left corner, represents the best achievable vehicle designs if the engine were not shared. For the remaining points, similar shapes are for problems of a given (w_{PC}, w_{LM}) combination, with the x’s indicating the superEGO results.

Note the difference between EGO and superEGO solutions for a given (w_{PC}, w_{LM}) combination. Because its resulting convex hull (i.e., Pareto set) is further up and left, superEGO found a somewhat better combined objective than EGO in all cases except Pareto point one (\triangleleft) which was only 0.2% worse. Table 8.1 compares the time and number of iterations it took superEGO to locate a feasible point better than EGO’s best solution for a given optimization problem. In all but the one case, superEGO’s approach to constraint handling significantly improves the efficiency of the algorithm.

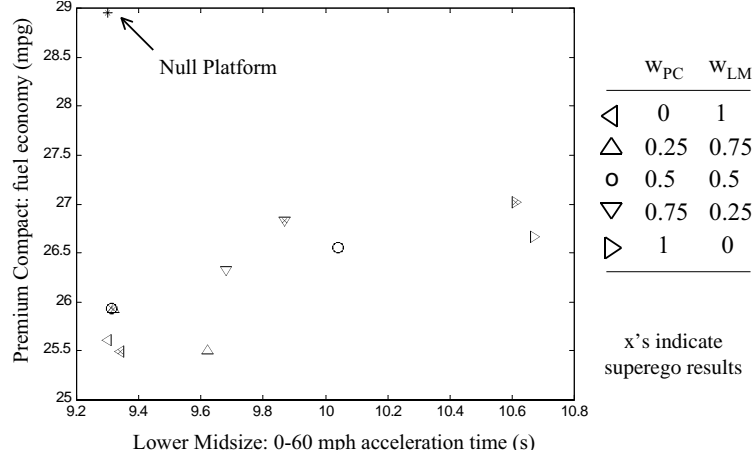


Figure 8.2: Null platform and Pareto set for vehicle study

Table 8.1: Percent improvement over the original EGO

	Pareto 1	Pareto 2	Pareto 3	Pareto 4	Pareto 5	Null 1	Null 2
	\triangleleft	\triangle	\circ	\triangleright	\triangledown	*	*
Iterations	-50.0	88.9	75.0	92.9	72.7	50.0	81.3
Time	-95.2	86.3	70.0	91.2	69.8	34.6	80.4

8.3 Discussion

The results of this study strongly support the claim that incorporating inexpensive information directly into the ISC subproblem yields good solutions more efficiently. In six of the seven optimization runs that were compared, superEGO required between 35% and 91% less time to arrive at a solution at least as good as the best solution obtained by EGO. For the remaining case, superEGO required one more iteration than EGO. Overall, the improvement in the number of iterations was slightly more significant than the savings in time. This shows that the benefits of using inexpensive information directly are not purely due to the time savings during the model fitting step. More accurate representation of the constraint functions lead to better solutions at each iteration. Had the expensive functions required more time to compute, the time savings would have been even more significant because fewer

iterations were required to reach good solutions.

This study also illustrates the difficulty in deciding whether to classify a function as “expensive” or “inexpensive”. Classifying a function as inexpensive reduces the number of models to fit and usually produces more accurate results for each ISC subproblem. However, for each iteration of superEGO, the inexpensive functions are evaluated on the order of $100n$ times, where n is the number of design variables.

As mentioned above, the ADVISOR calculations of the performance constraints were replaced by spline models in order for the filter method to be applied. This is because the fuel economy was only four times more computationally expensive than the set of performance constraints. Therefore the performance metrics were only “semi-expensive” in this example. The results would have taken much longer to generate if they had been classified as “inexpensive” because the time required for hundreds of performance evaluations during the ISC subproblem far exceeds a single fuel economy calculation. By creating spline models of those function *a priori*, solving the ISC subproblem becomes negligible compared to the fuel economy calculations. While it is not always efficient to generate accurate spline models, the point of this study was simply to demonstrate the behavior of superEGO in cases where there is a large disparity in function computation time. The spline models were necessary to demonstrate that behavior.

In summary, unless the difference in the computation time is at least two orders of magnitude, we suggest classifying the semi-expensive functions as “expensive”. Otherwise, the time required to solve the ISC subproblem will exceed the time saved by avoiding a model fitting step.

8.4 Chapter Summary

This chapter has demonstrated the capabilities of the superEGO algorithm to successfully solve a simulation-based optimization problem efficiently. Directly incorporating inexpensive constraint information into the ISC subproblem was shown to generate good solutions more quickly than the original EGO methodology. Using the techniques described in Chapter 6, we were able to reduce the number of iterations required to solve the optimization problem by 50 – 93%.

In the next chapter, we examine the second study which involves the design of experiments for an ergonomics test procedure.

CHAPTER 9

Reach Effort Study

The study presented here demonstrates how superEGO can be applied to a problem involving something other than traditional artifact design. Instead, superEGO is utilized to create adaptive experimental designs for an ergonomics test procedure that explores the reach range of a seated subject. A subjective measure of the *difficulty* of reaching to a specific target is obtained. SuperEGO creates a model of the difficulty and adaptively controls the data to be gathered from the test subject. The goal is to have superEGO determine a design of experiments (DOE) that adapts to each test subject. The resulting DOE is more efficient and yields more useful information than the current experimental procedure.

One of the arguments in this thesis is that optimization via Bayesian analysis should not be restricted to a single infill sampling criterion (ISC). Rather, the ability to tailor the algorithm easily to the problem at hand is a powerful property of the methodology proposed here. The unusual nature of this problem requires a sampling criterion created specifically for this application. In addition, inexpensive constraints are directly incorporated into the ISC subproblem to prevent a robot arm from coming too close to the test subject. Thus, this study demonstrates the flexibility of superEGO to solve completely different kinds of problems that EGO could not.

9.1 Problem Description

A reach effort study is currently underway at University of Michigan's HUMOSIM Laboratory. Figure 9.1 shows the setup for the experiment. A subject is seated in a mock-up of an industrial or workplace environment, with the chair oriented so that the reach target is in the subject's lateral plane. A computer controlling the target can raise and lower it as well as move it nearer to or farther from the subject (see Figure 9.2(a)). In the testing procedure, the target is positioned and the subject is asked to reach to the target and depress the button for two seconds. The subject then rates the difficulty of the reach on a scale of 1 to 10, with 10 being the most difficult. Maintenance of balance, the amount of exertion or degree of awkward posture required, and distance to the target all contribute to the difficulty of the reach. Once the subject presses the button, they use a small control panel (see Figure 9.2(b)) to quantify how difficult it was to reach the target on a scale of 1-10 (i.e., the *reach effort*). Because the number is input through a slider, the information is continuous. If the subject is unable to reach the button, the reach effort is assigned a value of 11.

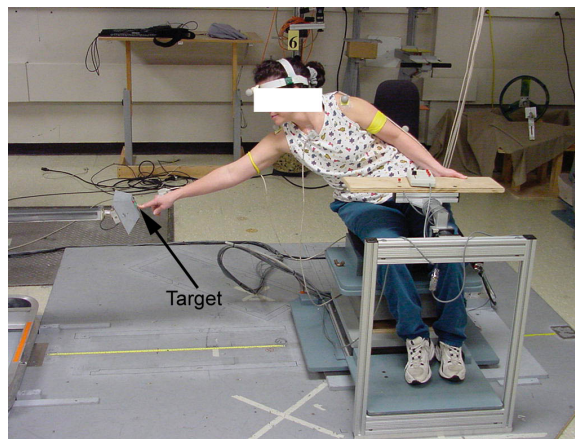
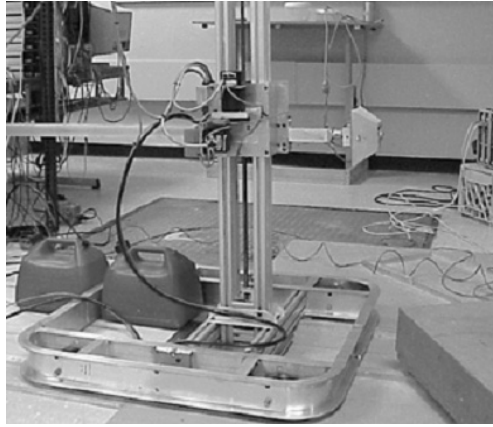
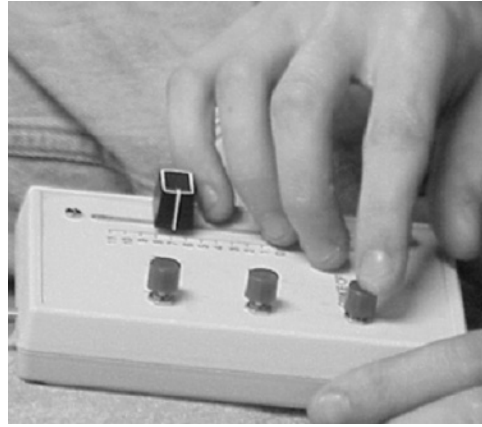


Figure 9.1: Example of a test subject reaching for the target



(a) robot arm



(b) slider

Figure 9.2: Reach effort study testing equipment

The objective of the test procedure is to obtain a mapping of the subject's reach effort, i.e., a data set with the reach effort for each associated target location. This is accomplished by obtaining a large data set from each subject using a number of different chairs. The current testing protocol requires a total of 400 button reaches - 200 in one chair, and 100 each in two other chairs - requiring five to six hours over the course of two days. The goal of this study is to reduce the required testing time without losing accuracy by creating more efficient designs.

Once data from a variety of subjects has been collected, reach effort models can be created to predict how difficult one may perceive a given reach to be. These models could take into account anthropometry (i.e., the size and shape) of the subject as well as their gender and age. The reach effort models could eventually be used to design the layout of the controls in an automobile interior as shown in Figure 9.3 or similar design domain.

The current testing protocol collects data along a finite number of rays in multiple planes. Only the right half of the test subject's reach zone is measured, assuming a



Figure 9.3: Automotive interior layout design

symmetric reach effort for the left half. The length of the rays are calculated before the experiment by estimating the maximum reach length of the subject. The target is placed at 0, 25, 50, 70, 80, 90, 95 and 100% of the distance from the subject to the expected maximum reach. Figure 9.4 shows an example of the current sampling scheme for the 90 degree (lateral) plane. The plot is shown from a perspective behind a subject extending their right arm in a lateral motion. The $[0, 0]$ reference is taken as the h-point, an ergonomic reference point at the mid-hip level along the body centerline. For convenience, the design shown below will be referred to as the *ray-based* experimental design.

It should be noted that the reaches are not ordered such that the subject samples all eight points in one radii, then moves to the next radii, etc. Rather, for a given rotational plane, the total set of 32 sample points (not including the 8 points directly overhead) is split such that 16 points are sampled in random order in one block of time. Then the plane is rotated for another set of 16 points, randomly chosen. Eventually, the total set of 200 points is sampled.

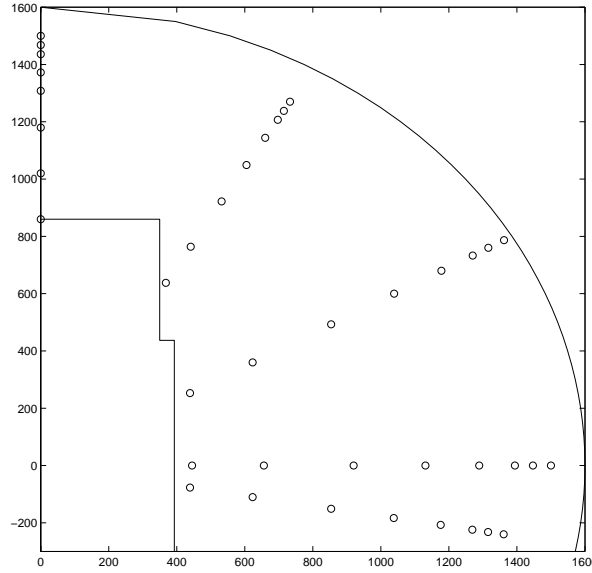


Figure 9.4: Example of ray-based experimental design

Throughout this study, testing was kept to a single plane (the 90 degree, or lateral, plane) to aid in the visualization and understanding of the methodology. The experiment placed the target in various locations around the test subject's right lateral plane. In addition, there were constraints to where the target may be placed.

1. *Don't hit the test subject!* This constraint is a rather important one. The protocol must ensure that the robot arm never encroaches upon the subject's personal space. More critically, it must never touch the subject. This area is referred to as the *exclusion zone*. It is modelled by two boxes around the subject as shown in Figure 9.5(a) and the lower left corner of Figure 9.4.
2. *Don't play keep away!* This constraint is intended to prevent placing the button in areas that are known to be too far for the subject to reach. Sampling a point there is a waste of time and energy. In the adaptive sampling scheme described below, this constraint is modelled as a sphere around the h-point. The radius is determined by the maximum reach length of the subject estimated in the

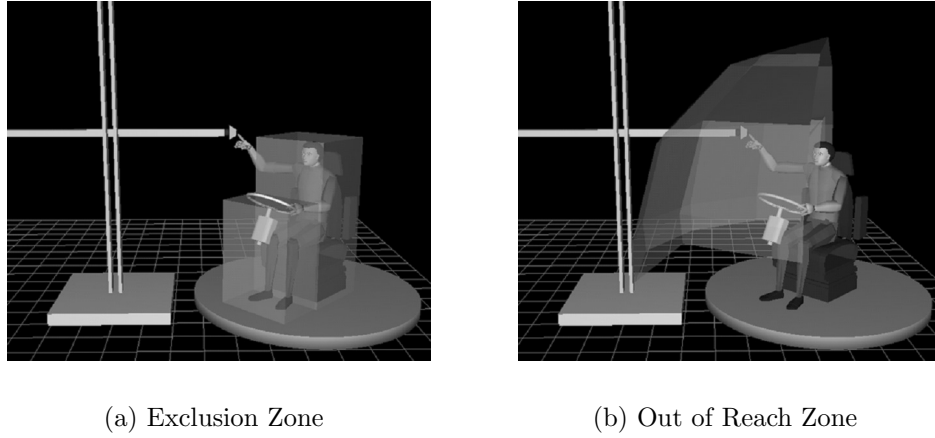


Figure 9.5: Button placement constraints around test subject

initial setup. This zone is referred to as the *out of reach zone* and is shown in Figure 9.5(b) and the arc in upper right of Figure 9.4. An alternative approach for this constraint that utilizes the kriging models of the reach effort will be described in Section 9.3.

9.2 Adaptive Sampling

In this section, we describe the approach taken for the study, known as adaptive sampling. The methodology is described and some initial tests done to illustrate their effectiveness. The case is made that adaptive sampling is necessary to fulfill the objectives of the study.

9.2.1 What is adaptive sampling?

An experimental run, be it a physical experiment or done on a computer, can be viewed as a black-box function. Input values are sent to the black-box, and the associated outputs are returned. Design of experiments (DOE) studies how to select the set of input values at which to evaluate the black-box in order to explore its behavior efficiently. There are many ways to quantify how well a particular

design achieves that goal. Among the most frequently used DOE's are factorial and composite designs, Latin hypercubes and orthogonal arrays. Each strategy attempts to spread points around the design space in some appropriate manner. The reader is referred to Montgomery [61] for more background on classical design of experiments.

In some applications, a particularly useful approach is to determine the set of input values sequentially, taking into account information from the previous experiments each time a new sample is requested. These so-called *adaptive designs* can be performed either in stages, where several new inputs are chosen at each stage, or fully adaptively, where each new input incorporates all prior information. Adaptive designs often appear in such fields as clinical research or allocation problems. For example, one class of problems known as *bandit problems* seeks to choose points from a finite set of alternative experiments with unknown outcomes at each of n stages in order to maximize overall yield. Clinical trials may use adaptive designs to determine whether to assign a test subject to the control or study group based upon the data collected thus far [101].

Adaptive designs are generally more complicated to generate because they require an optimization problem to be solved each time new input values are chosen rather than fixing the entire DOE before the experiments begin. The topic is currently an active area of research [5], [6], [30], [94] and specialized algorithms have been developed [36]. However, because adaptive design involves solving an optimization problem, general nonlinear programming techniques can be used to generate the designs. Due to the nature of the algorithm, superEGO is well suited to such problems.

The premise of this study is that the current testing protocol is extremely inefficient and yields poor reach effort models. SuperEGO is used to address both of

these shortcomings by creating a more efficient experimental design. In the proposed approach, superEGO takes a small initial data sample, then iteratively determines where to place the target next based on the previous reaches. Unlike the current sampling protocol, SuperEGO is able to select any point within the feasible space, not just points along discrete planes and rays. This greatly increases the accuracy of the models by filling the design space.

An important question in the adaptive sampling scheme, therefore, is what constitutes a “beneficial” sample point in this application. Obviously, the expected improvement function is not appropriate for the problem, so sampling criteria suitable for the needs of adaptive sampling must be applied. In much of the research on adaptive sampling, measures such as the integrated mean squared error or maximum mean squared error of the model are used as the criterion for selecting the next sample point [77], [78]. These criteria do not fully satisfy our needs, so we take advantage of the flexibility of superEGO by defining several sampling criteria designed specifically for this study. Doing so is straightforward and does not interfere with any of the remaining code (see Appendix A). For this study, two metrics were considered:

1. *Reduce local uncertainty throughout the design space.* For this criterion, superEGO searches for the location in the design space where the kriging variance (a measure of local uncertainty) is highest. Placing a sample point in that location will improve the local model accuracy. For this experiment, the variance is calculated as shown in Equation 3.14 in Chapter 3.

$$\text{ISC}_1 = \hat{\sigma}^2(\mathbf{x}) \tag{9.1}$$

2. *High definition of the 7-8 zone.* We would also like to ensure that the model

of the reach effort has a high accuracy for reach efforts between 7 and 8. This is a typical range of values about which automotive interior designers make decisions for control placement. SuperEGO searches for the location of maximum kriging variance conditional to the fact that predicted reach effort value is between 7 and 8. The goal is to spread the points out well in the 7-8 zone, thereby reducing the uncertainty in its shape and location. To quantify the criterion, the following expression is used:

$$ISC_2 = \begin{cases} \hat{\sigma}^2(\mathbf{x}), & \text{if } 7 \leq \hat{y}(\mathbf{x}) \leq 8 \\ 0, & \text{otherwise} \end{cases}. \quad (9.2)$$

9.2.2 How does it behave?

To evaluate the efficacy of the adaptive sampling approach, a pilot test was performed using analytical models to simulate the reach effort value a subject may assign for any given target location. Two analytical models were created to mimic the behavior of different test subjects. Virtual subject A was a quadratic polynomial with a log term, virtual subject B a pure quadratic model (see Figure 9.6). The virtual subjects were then used to evaluate the behavior of the adaptive sampling methodology. An initial 11 point sample was selected to include a number of points around the perimeter of the feasible space. It is a well known feature of kriging that the variance tends to be large around the outer edge of the design space which is less densely sampled. Populating points in this area at the onset attempted to reduce the number of far away samples chosen as infill samples during the procedure.

An initial test was run with subject A using ISC_1 , the kriging variance, as the sampling criterion. Figure 9.7 shows the progress of superEGO over 39 iterations (50 samples total). The initial 11 point sample is shown with circles, and infill sampling

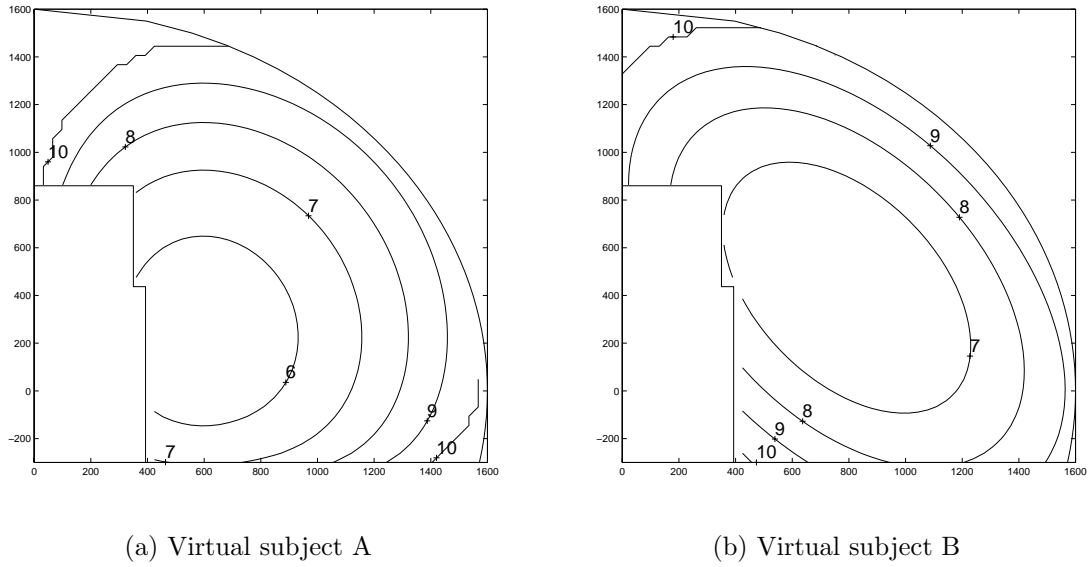


Figure 9.6: Contour plots of the reach effort for two virtual subjects. The exclusion zone and out of reach zone constraints are also shown.

points are shown as dots and x's, the most recent being the x's. The test reveals that the variance ISC behaves as expected. The infill points chosen tend to “fill the space”.

Because an analytical model was used as the test subject, the error from the predicted to the actual reach effort could be calculated. Obviously this is not a practical approach for actual testing, but it at least indicates the trend in the model accuracy improvement as infill samples are added. Figure 9.8 shows two measures of model error in the feasible region. The plots asymptotically approach zero error, as expected. The benefits from sampling more than 30 points become increasingly negligible, so we have chosen that as our limit for this demonstration.

Given this initial success, a 30 point sample was taken for both virtual subjects (see Figure 9.9). The same initial 11 point sample (shown as circles) was used for both models, and the infill samples selected by superEGO are shown as x's. The two resulting sample sets are indeed different, indicating that the adaptive sampling can

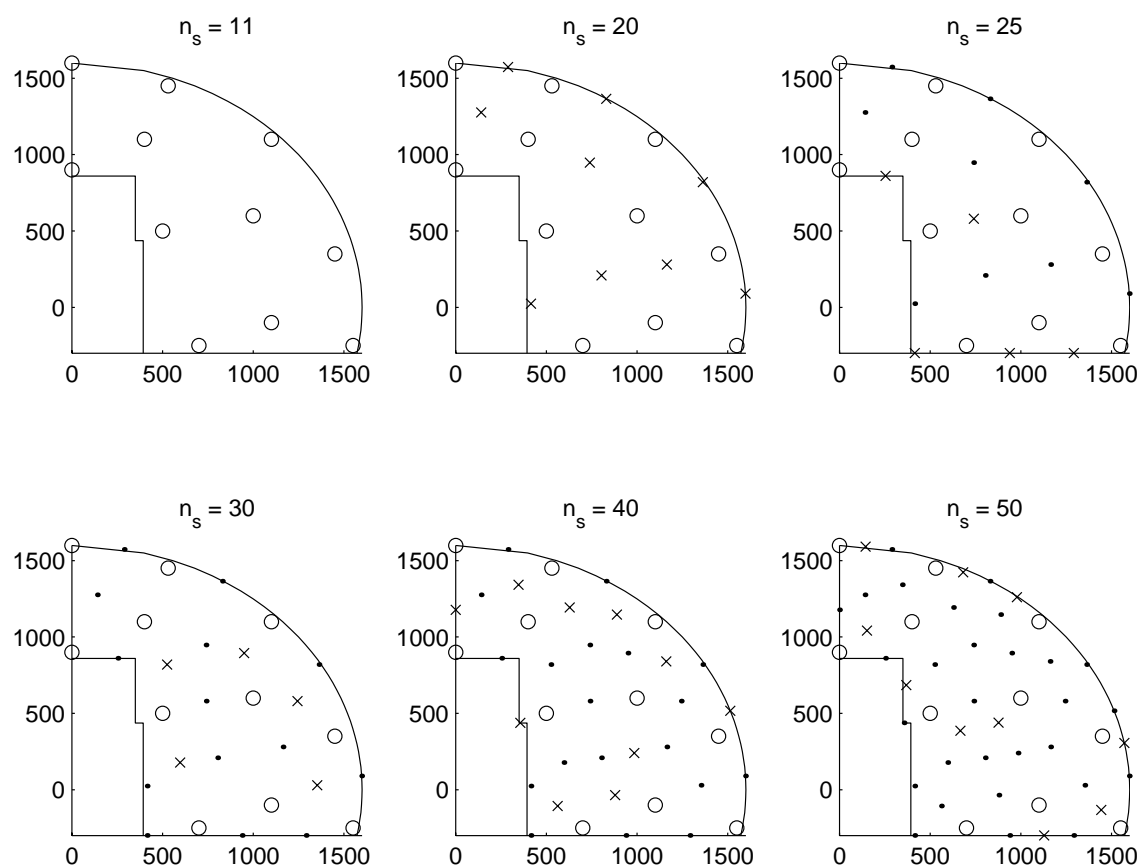


Figure 9.7: Progress of adaptive sampling for ISC_1 . Circles are initial samples, dots and x's are infill samples, x's being the most recent.

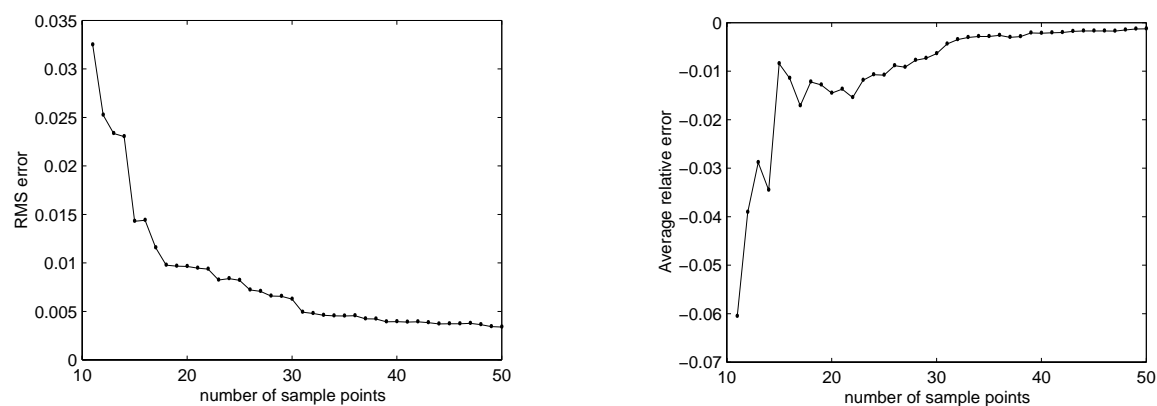


Figure 9.8: Error in the predicted reach effort as a function of the number of infill samples

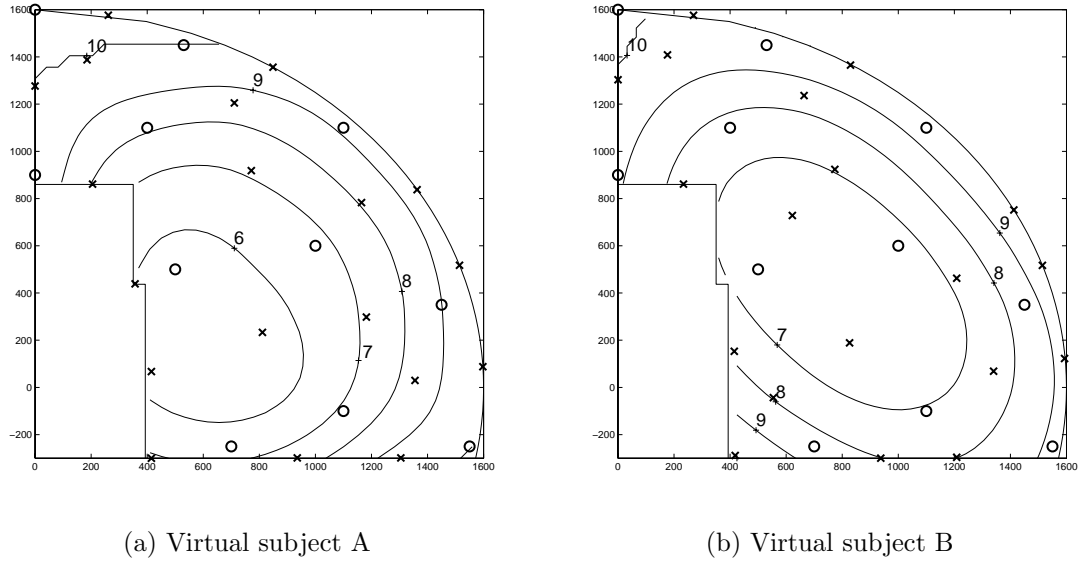


Figure 9.9: Adaptive sampling results using ISC_1 . Contours of the reach effort model are shown along with the constraints and sample point locations. Circles are initial samples, x's are infill samples.

effectively choose sample sets that depend on the subject. The models also appear to be a reasonable representation of the virtual subjects' reach effort. The accuracy deteriorates in the infeasible space which is unsampled, but that is irrelevant because the models are only meaningful in the feasible space.

A final test applied the 7-8 zone criterion of Equation (9.2) to generate a smaller 20 point sample around an area of interest. SuperEGO correctly placed the new infill samples in the region where the reach effort model predicted a value between 7 and 8 (see Figure 9.10). The modeling error may lead to sample points that are not actually in the 7-8 zone. But overall, the method tends to place points around the subjects' true 7-8 zone, thereby increasing the local accuracy of the models in that region. It is important to note that even though the shape of the 7-8 zone differs from subject A to subject B, the method successfully adapted the sampling pattern to the individual subject.

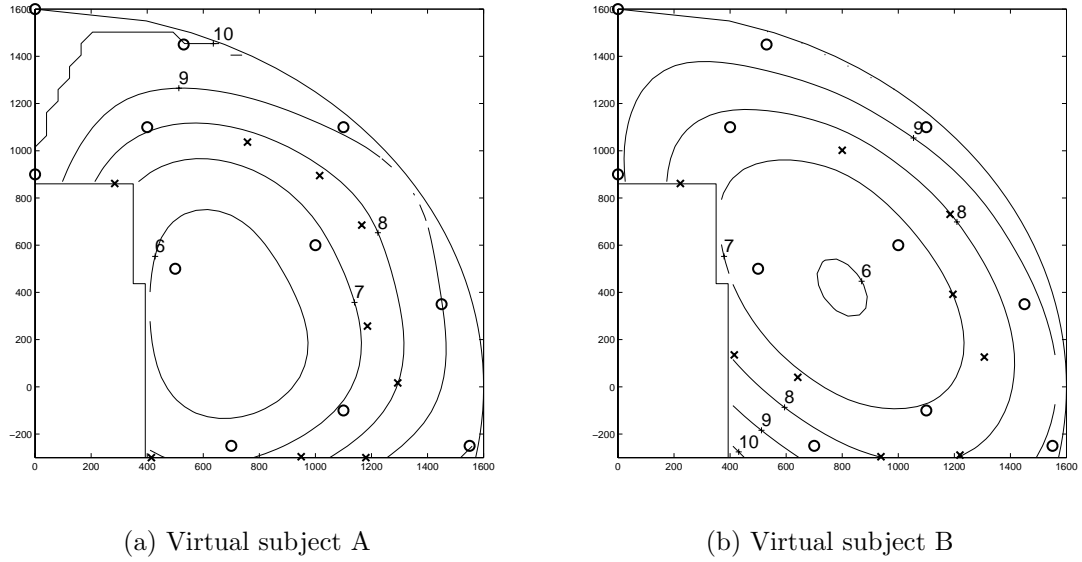


Figure 9.10: Adaptive sampling results using ISC_2 . Contours of the reach effort model are shown along with the constraints and sample point locations. Circles are initial samples, x's are infill samples.

9.2.3 Why use it?

Confident that the proposed methodology is able to generate adaptive designs, we now examine the benefits of doing so. In other words, why do we need adaptive sampling at all?

Consider first the demonstration shown in Figure 9.9. Applying the maximum variance criterion, different samples resulted in different experimental designs for the two subjects. This is because the kriging model parameter θ from Equation (3.3) was fit at each iteration via maximum likelihood, leading to different covariance models. The same θ values could have been used for each test subject and kept constant throughout the testing. The approximation would have still fit each data set well, but having identical covariance models would have resulted in *identical* experimental designs. This is because the kriging variance is a function of the covariance model, but *not* the response values.

If both subjects could have been sampled using the same set of covariance model parameters, resulting in identical designs, why use adaptive sampling? The answer is that maximizing the kriging variance for a fixed covariance model subject to the exclusion and out-of-reach zone constraints is *not* an adaptive procedure. For a given covariance model, neither the objective nor the constraints functions are impacted by the value of the reach effort given by the subject; therefore, a sampling design can be computed *a priori*. In order to make the process adaptive, one of two alterations must be made: refit the covariance models as data is sampled, or include constraints that are effected by the data.

In the virtual subject testing, we refit the covariance models to generate an adaptive process for ISC_1 . However, there are reasons one may wish to fix the covariance model parameters during sampling, as will be described in the next section. Therefore, additional constraints should be added to the ISC subproblem if the sampling is to be truly adaptive. It will be shown in Section 9.3 that defining the exclusion zone constraint as a kriging model results in experimental designs that adapt well to individual test subjects. It is important to note that tests using ISC_2 are always adaptive because the sampling criterion is influenced by the reach effort data through the condition $7 \leq \hat{y}(\mathbf{x}) \leq 8$ in Equation (9.2).

With a firmer understanding of what is and what is not adaptive sampling, we now justify the need for adaptive sampling through the results from the virtual subject testing described above. For each virtual subject, a kriging model was fit to a 40 point data sample collected at the locations specified by the ray-based approach shown in Figure 9.4. Kriging models were also fit to a smaller data sample collected using superEGO for two different sampling criteria, each initialized with the same 11 point sample. A total of 30 and 20 samples were collected utilizing ISC_1 and ISC_2 ,

respectively. Because the virtual subjects were analytical functions, the RMS error from the predicted to the actual reach effort could be calculated for each kriging model.

Table 9.1 compares the model accuracy for each subject for the two sampling criteria. Test 1 computed the accuracy over the entire feasible space for the 40 point ray-based design and the 30 point adaptive design from ISC_1 . Test 2 computed the accuracy in only the 7-8 zone for the 40 point ray-based and 20 point adaptive design from ISC_2 . The adaptive sampling strategy using ISC_1 provided a model with superior global accuracy and had an RMS error 3 times lower than the ray-based design even though it used 25% fewer sample points. Adaptive sampling also provided very accurate results in the local zone of interest when ISC_2 was applied for test 2. It was more than twice as accurate than the ray-based model for subject A while using half as many samples. The model from superEGO was slightly less accurate than the ray-based model for subject B, but one additional iteration of superEGO (21 points total) resulted in an RMS of 0.0198, better than the 40 point ray-based model.

Table 9.1: Comparison of RMS errors resulting from various sampling strategies. Measures of model accuracy are restricted to the feasible space for test 1 and to the 7-8 zone for test 2.

	Virtual subject A		Virtual subject B	
	superEGO	ray-based	superEGO	ray-based
Test 1	0.1072	0.3616	0.0664	0.1944
Test 2	0.0745	0.1786	0.0252	0.0220

While the RMS error helps identify the accuracy of the kriging models generated by various samples, it is important to also understand how accurately the borders of the 7-8 zone were defined. In Figure 9.11 we see the 7 and 8 contours of the

superEGO-based models are much closer to the true contours than those of the ray-based model for virtual subject A, which agrees with the information in Table 9.1. However, there is no discernable difference between the two models for virtual subject B as shown in Figure 9.12. The model from each predicts boundaries for the 7-8 zone that are very close to the true boundaries. This also agrees with Table 9.1.

It is important to examine the certainty with which each model defines the 7-8 zone as well. While the contours predicted by each model of subject B are comparable, there is a large discrepancy between the confidence intervals around those contours. Figure 9.13 shows the contours of the mean value (shown as a solid line) plus or minus 3σ (shown as dotted lines), for the reach effort values of 7 and 8. The meaning of the confidence interval, say for a reach effort of 7, is that the model predicts with 99.7% confidence that the true contour for a reach effort of 7 is between the plus or minus 3σ bounds. For each model, the intervals are tighter in regions where a data point is nearby, and looser (farther apart) in regions far from sample points.

It is clear from Figure 9.13 that the confidence intervals around the 7 and 8 boundaries are much tighter for the model from the 20 point superEGO design than for the 40 point ray-based design. This signifies that the designer can place a much higher degree of confidence in the location of the 7-8 zone as defined by the superEGO model than for the ray-based model even though half as many samples were taken. In general, the accuracy of the model from adaptive sampling shows that efficiency in experimental design can be achieved through adaptive sampling via superEGO. The advantages of adaptive sampling would likely be even more pronounced for a full three dimensional sampling.

While the results here support the claim that adaptive sampling can lead to im-

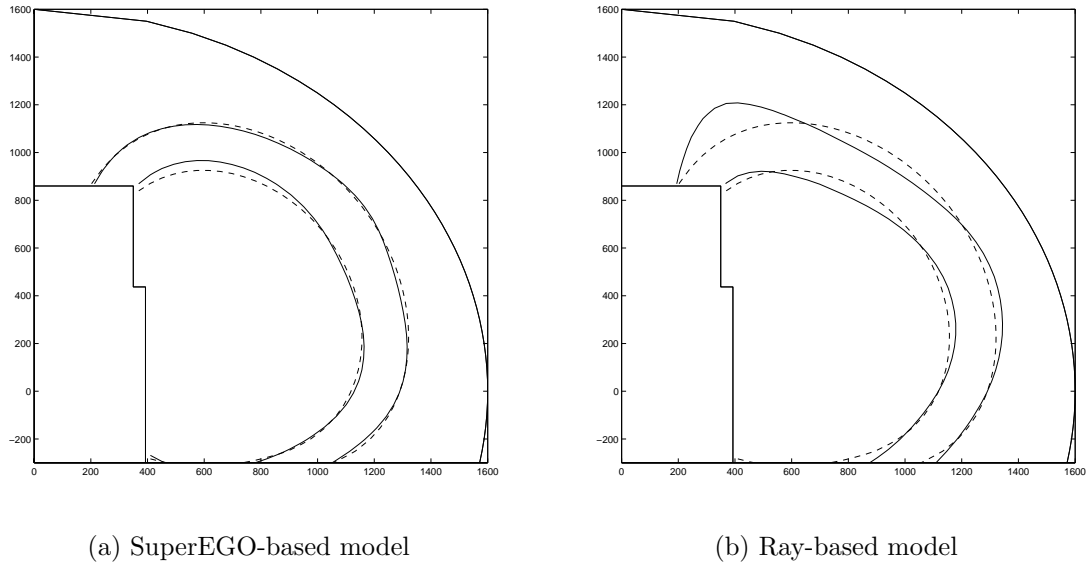


Figure 9.11: Contours of the 7-8 zone of virtual subject A for two experimental designs. The true contours are shown as a dashed line.

proved modeling accuracy and/or efficiency, can one justify the added complication? It would be much simpler to generate a generic space-filling design as a fixed DOE. Because the variation in the stature of test subject prohibits using a single DOE, one would have to scale the DOE for each subject. Doing so is not necessarily reliable because of the number of physical characteristics that determine reach range. Also, a subject may have physical limitations that prevent them from reaching a point that another person of the same size might easily reach. Individuals vary in how flexible they are as well, so two subjects of identical size may in fact rate identical reaches differently. Thus, it is not feasible to create a scalable, static design that would fulfill the needs of this experiment. In addition, there may be an interest in subject-specific information to be accounted for in the sampling (e.g., the location of a subject's 7-8 zone). The adaptive sampling scheme is necessary in such cases. For these reasons, adaptive sampling has been selected as an appropriate tool for the test procedure.

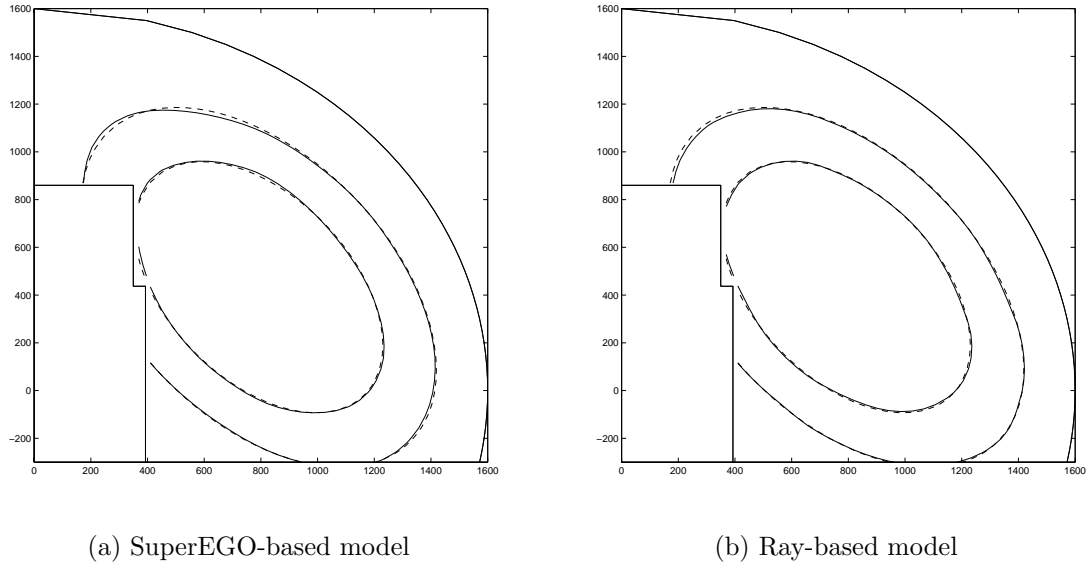


Figure 9.12: Contours of the 7-8 zone of virtual subject B for two experimental designs. The true contours are shown as a dashed line.

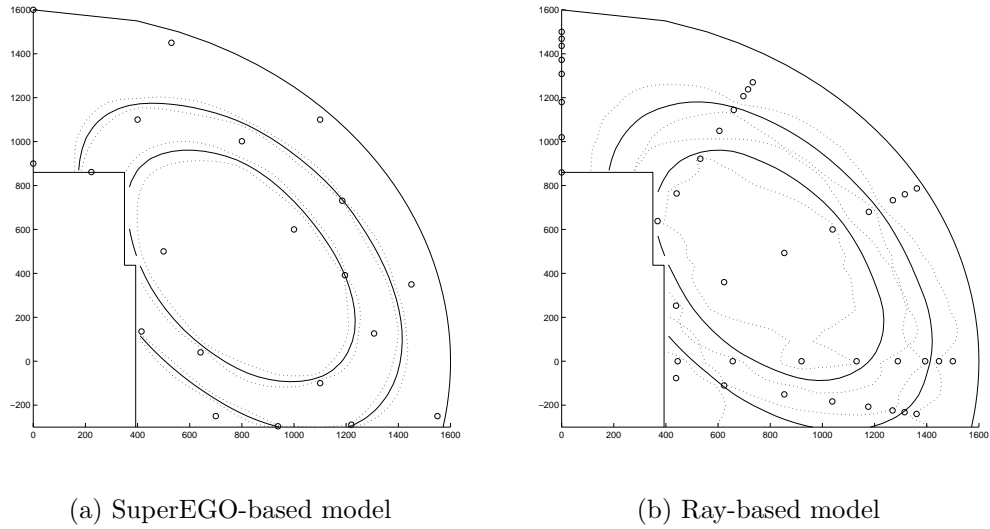


Figure 9.13: Confidence intervals for the 7 and 8 contours (solid lines) of the reach effort of virtual subject B. The superEGO generated design yields much tighter $\pm 3\sigma$ confidence bands (dotted lines) with half as many samples (circles).

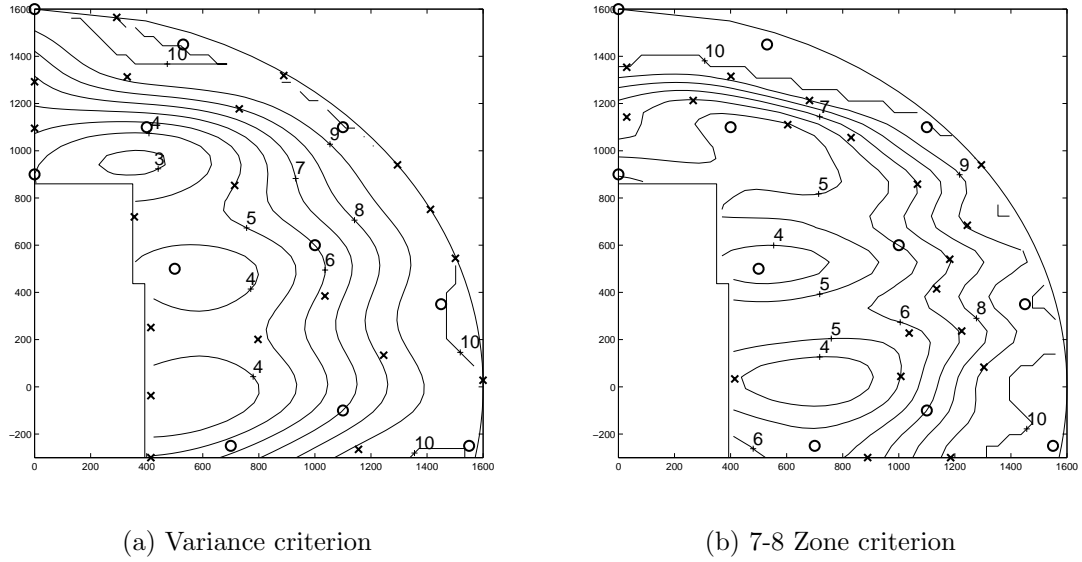


Figure 9.14: Resulting sample for first subject

9.3 Initial Assessment

After the virtual testing was complete, the behavior of adaptive sampling was examined on human test subjects. An initial test was done to better understand the practical difficulties associated with the experiment. Given that the purpose of the tests was to debug the methodology and not to generate results, I bravely used myself as the first test subject. As with the virtual subjects, we sampled 30 points in the 90 degree plane only. The same 11 point sample set was used to initialize the routine. Tests were performed with ISC_1 , the variance criterion and ISC_2 , the 7-8 zone criterion. The resulting 30 point samples are shown in Figure 9.14.

The adaptive sampling did not fully perform as anticipated during this experiment. SuperEGO fit the covariance model parameters every time a new point was added, and plots were made to show the resulting mean and variance of the models. Small changes in the data set unpredictably lead to such poor fits at times that the variance reached its maximum value (i.e., the sill of the variogram in geostatistics

terms) everywhere except in the immediate vicinity of a sample point. This was due to poor model parameters chosen during the MLE fitting process. Specifically, the situation described in Section 3.6.1 lead to very large theta values (i.e., no correlation). As a result, the variance was essentially identical everywhere, and the ISC could not determine the location of maximum uncertainty. The importance of reasonable reach effort models for this procedure cannot be understated.

In an effort to correct the situation, the value of p in the correlation model of Equation (3.3) was fixed at 1.99. Doing so resulted in models that did not behave erratically after MLE fitting was complete. This points to one of the difficulties in using Bayesian analysis successfully - development of a robust, accurate approximation technique. While kriging has generally performed quite well in this field, this author has experienced occasions where the robustness of the models fit by the standard DACE modeling MLE procedures is in question.

Even when the reach effort models were reasonable, they were not always as accurate as was needed. The specific problem encountered was that the reach effort model did not fully conform to our expectation of what the true reach effort should look like. For example, several times during the 7-8 zone criterion test, the procedure placed buttons in a location that I subsequently rated either below 7 or above 8.

Figure 9.15 shows the actual reach effort I assigned each sample point after the initial 11 points. Theoretically, one would expect the actual values to converge towards the 7-8 zone as iterations progressed. This did not occur during the initial testing. It is possible that more data is needed to accurately predict such a narrow band with confidence. It is also possible that a one integer range of 7-8 is impractical given the imprecise, subjective nature of the reach effort rating.

Another important lesson from these early experiments concerned the implemen-

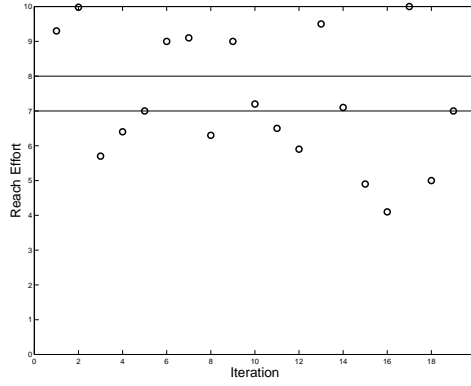


Figure 9.15: Actual reach effort values assigned during initial testing. Horizontal lines are the 7-8 zone limits.

tation of the out-of-reach zone. In the testing done in this experiment, the zone was defined simply by a 1600 mm radius around the subject’s h-point. The adaptive sampling tended to place the target along the 1600 mm limit because kriging models have high variance where little data exists. The constraint was therefore active in the solution to the ISC subproblem. Because the 1600 mm radius was clearly out of reach when I myself was the test subject, it was decided that a different approach was needed to prevent the adaptive sampling approach from “playing keep away”.

The proposed solution was to replace the 1600 mm radius constraint with one that used the predicted reach effort model as:

$$g_{outofreach} = \max(\hat{y}(x) - 10, 0) \quad (9.3)$$

The role of Equation (9.3) was to provide a means of preventing exceedingly far reaches in a way that adapts to how far the subject can actually reach. The implementation is such that the constraint has a zero (feasible) value for any location the predicted reach effort is less than 10 and linearly increases for predicted efforts above 10. The user could easily change to some other value besides 10 to either widen or narrow the feasible search space as fits the application.

It was also discovered that the ability to use non-interpolating kriging models greatly enhanced the accuracy of the approximations. By smoothing out the noise inherent to the subjective reach effort metric, the resulting models behaved more intuitively. The advantages of the smoothed models were an important factor in the success of subsequent experiments. With a better understanding of the adaptive sampling methodology, it was time to use test subjects unfamiliar with the project.

9.4 Subject Testing

For clarity's sake, we summarize the problem description for the final pilot experiment involving two test subjects. Subject A was a 181 cm tall, 84 kg, 25 year-old male. Subject B was a 157 cm tall, 52 kg, 24 year-old female. Before any measurements were recorded, they were each given about 10 reaches of various difficulty in order for them to begin building an internal system for assigning the reach effort value. Each subject was then given the same set of 11 initial sample point locations shown in Figure 9.16. The sample points were moved closer to the subjects than previous experiments in order to prevent “wasting” reach efforts on area that were clearly out of reach for subject B.

A kriging model of the subject's perceived reach effort was successfully fit using the initial data set, then the model parameters $[\theta, p, \text{nugget}]$ were held fixed for the rest of the experiment. Doing so reduced the waiting time between reaches and yielded more reliable reach effort models.

Three different sampling criteria were implemented in this study. The first attempted to create a space-filling design by selecting the location of maximum kriging variance, Equation (3.14), at each iteration. The other two criteria were used to increase the accuracy of the model where reach effort values were between 7 and 8.

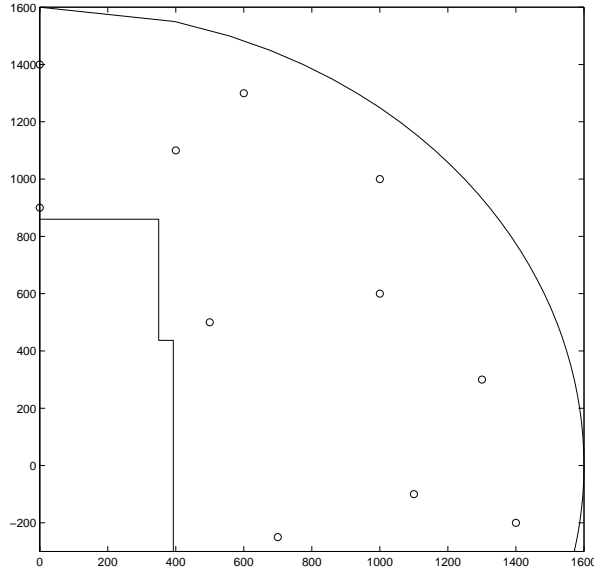


Figure 9.16: Initial set of 11 reaches

The difference between the last two criteria was that one predicted the 7-8 zone using an interpolating kriging model, while the other used a smoothed model. Points predicted to be outside the 7-8 zone were assigned an ISC value of zero. The ISC are summarized below.

$$ISC_1 = \hat{\sigma}^2(\mathbf{x}) \quad (9.4)$$

$$ISC_2 = \begin{cases} \hat{\sigma}^2(\mathbf{x}), & \text{if } 7 \leq \hat{y}_{interpolating}(\mathbf{x}) \leq 8 \\ 0, & \text{otherwise} \end{cases} \quad (9.5)$$

$$ISC_3 = \begin{cases} \hat{\sigma}^2(\mathbf{x}), & \text{if } 7 \leq \hat{y}_{smoothed}(\mathbf{x}) \leq 8 \\ 0, & \text{otherwise} \end{cases} \quad (9.6)$$

Three constraints were incorporated into the solution of the ISC problem at each iteration. They were the exclusion zone, out-of-reach, and predicted out-of-reach constraints. The first kept the target from getting too close to the subject and was modeled by the rectangular region shown in the lower left corner of Figure 9.16. The

out-of-reach constraints prevented placing the target in locations that were clearly out of the subject's reach, thus wasting time. The second constraint modeled the out-of-reach zone by a 1600 mm radius around the subject's h-point ($[0,0]$ on the graph). The third constraint modeled the out-of-reach zone by a smoothed kriging model of the reach effort. Predicted reach efforts above 10 were considered infeasible. While g_3 is more adaptive to the subject, g_2 was kept as a redundancy in case the reach effort model were to behave erratically. The three constraints are summarized below.

$$g_1 : \mathbf{x} \notin \text{Encroachment zone} \quad (9.7)$$

$$g_2 : \|\mathbf{x}\| \leq 1600mm \quad (9.8)$$

$$g_3 : \hat{y}_{smooth} \leq 10 \quad (9.9)$$

During the experiments, the kriging variance of an interpolating model was used to quantify the uncertainty in reach effort model. While the smoothing approach exhibited a more well-behaved reach effort model, there were problems with the variance calculations as demonstrated in Figure 3.11. Because the sampling criteria used the kriging variance to create a space-filling design, it was critical that the variance tend toward zero at the sampled locations and increase monotonically as the distance from neighboring samples increased. Thus, the interpolating model was selected. However, we have taken advantage of the flexibility of superEGO to calculate the objective of the ISC subproblem with variance of the interpolating model and the constraint of the ISC subproblem with predicted value of the smoothed kriging model.

In summary, the \mathbf{R} matrix of the kriging model was updated each iteration to incorporate the new data (without refitting the model parameters), then one of the

ISC's of Equations (9.4)-(9.6) was maximized subject to the constraints in Equations (9.7)-(9.9). The solution to the ISC subproblem was the location to place the target next. The subject reached for the target, rated the effort, and the process was repeated until a specified number of reaches were performed. For ISC_1 , a total of 30 points (including the initial 11) were sampled. Only 20 points were sampled for ISC_2 and ISC_3 .

9.4.1 Results

We begin by showing the resulting DOE's for each subject from ISC_1 (see Figure 9.17). The initial 11 points are shown as circles, with the remaining 19 infill points shown as x's. As expected, the two DOE's were quite different from each other. As discussed in Section 9.2.3, the kriging variance is homoscedastic, that is, it depends only on the location of the data, not the values thereof. So why then were the DOE's different? It is because the predicted out-of-reach constraint, g_3 , adapted to the user. As the kriging variance is high on the outer perimeter of the design space, where fewer data points are nearby, some infill points tended towards the edge of the feasible space. Because the predicted out-of-reach zone for subject A was beyond the 1600 mm radius, g_3 was dominated by g_2 in the majority of the space, and points were placed along the edge of the 1600 mm radius. However, for subject B, g_2 was dominated by g_3 because she could not reach that far, and points were placed along the edge of her predicted out-of-reach zone instead. Therefore, the infill locations for the two subjects quickly diverged. To compare the resulting reach effort models of the subjects after the 30 points, the interpolating and smoothed kriging models are shown in Figures 9.18 and 9.19, respectively.

The next two criteria were run for a total of 20 points each. Looking first at

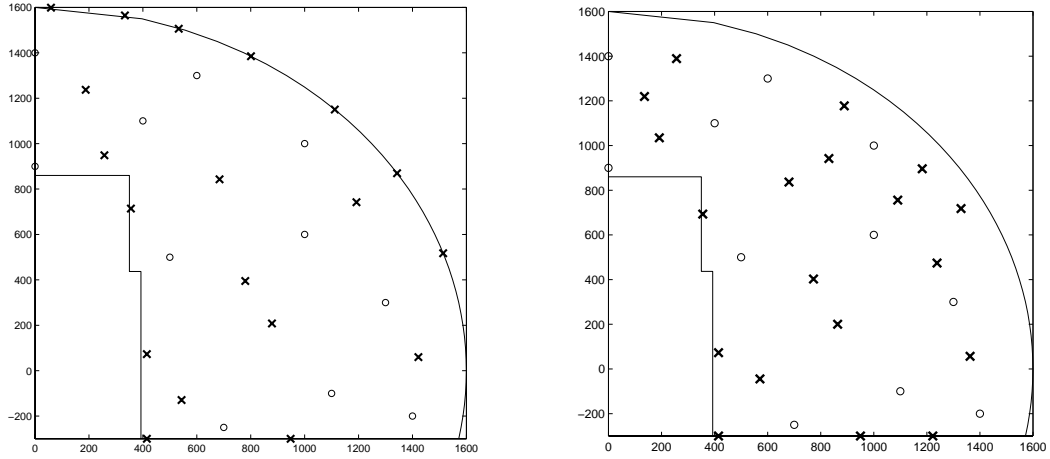


Figure 9.17: DOE for ISC_1 for Subjects A (left) and B (right)

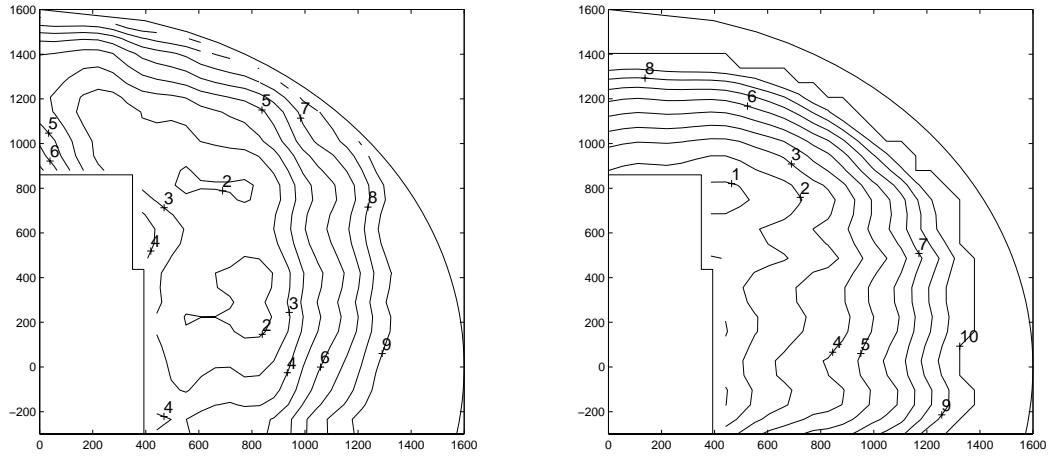


Figure 9.18: Reach effort interpolating model for Subjects A (left) and B (right) based on DOE from ISC_1

ISC_2 , Figure 9.20 shows the adaptive designs for the two subjects. For ISC_2 , the criterion estimates the 7-8 zone using the interpolating model. Figure 9.21 compares the predicted to the actual reach effort at each iteration. Similarly, Figures 9.22 and 9.23 show the adaptive designs and predicted vs. actual reach efforts for ISC_3 . For ISC_3 , the predicted 7-8 zone was estimated using the smoothing model. In both cases, the prediction accuracy was higher for subject B than for subject A. It is possible that subject B's smaller feasible design space made predictions more reliable, or that subject B had an underlying (true) reach effort that was more smooth than

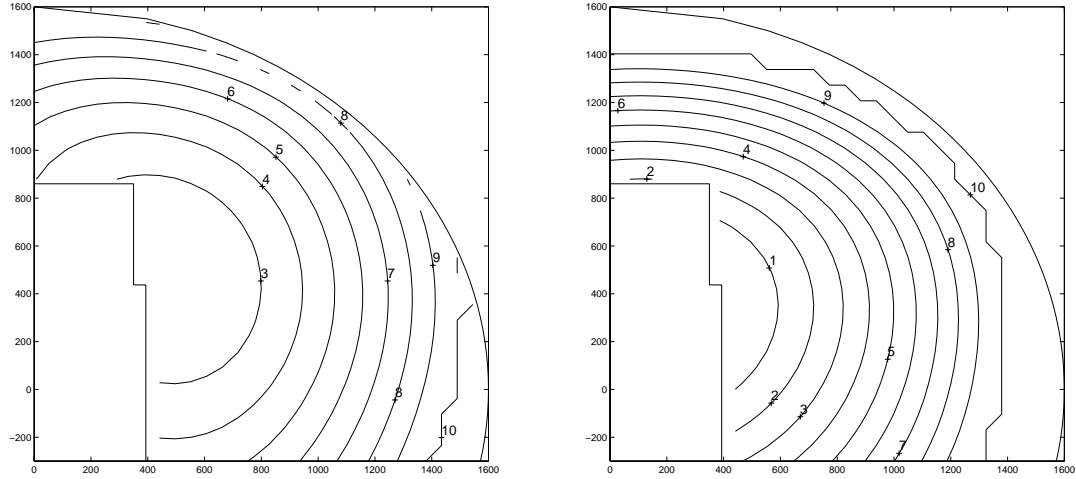


Figure 9.19: Reach effort smoothed model for Subjects A (left) and B (right) based on DOE from ISC_1

that of subject A, thereby facilitating the modeling. However, no general conclusions should be drawn from such a small pilot test.

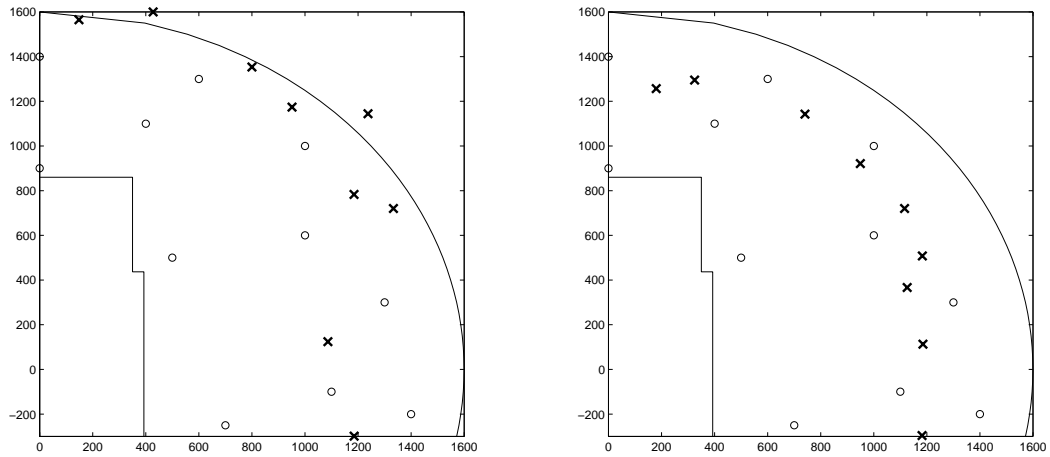


Figure 9.20: DOE from ISC_2 for Subjects A (left) and B (right)

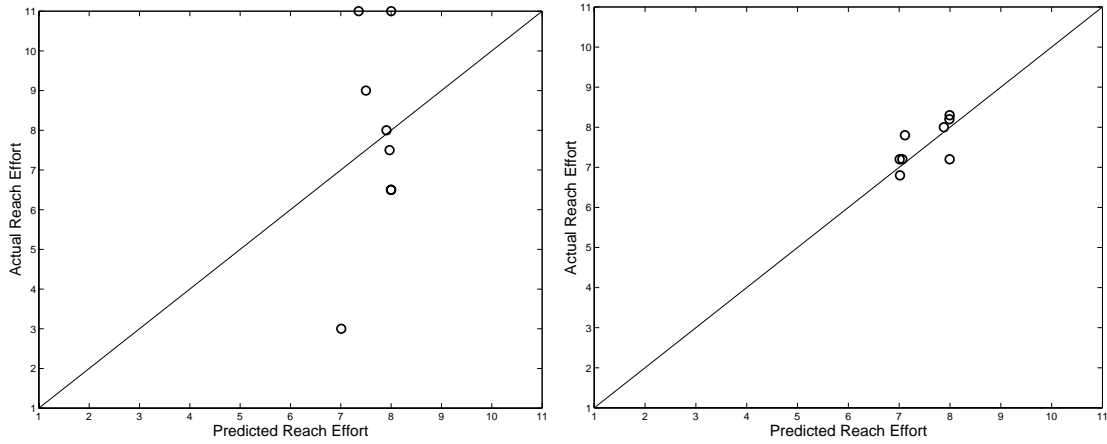


Figure 9.21: Comparison of predicted (x-axis) to actual (y-axis) reach effort using ISC_2 for subjects A (left) and B (right)

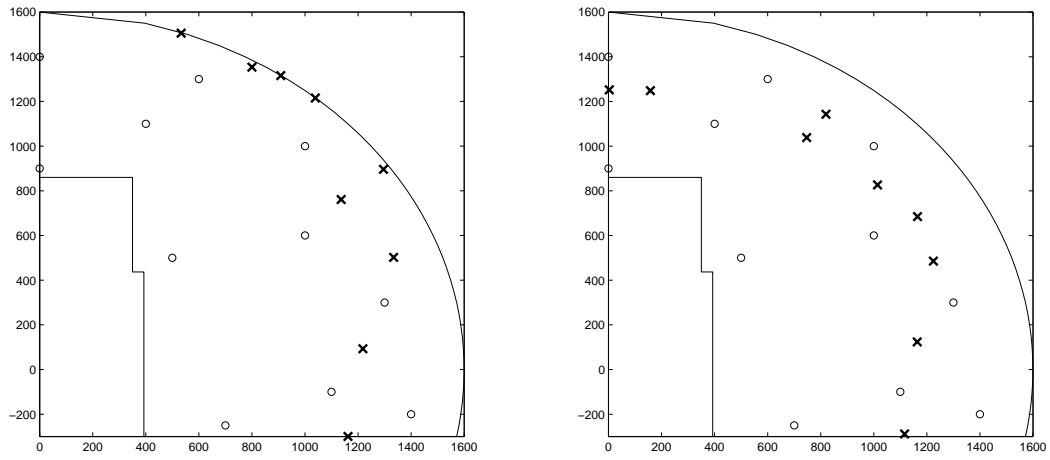


Figure 9.22: DOE from ISC_3 for Subjects A (left) and B (right)

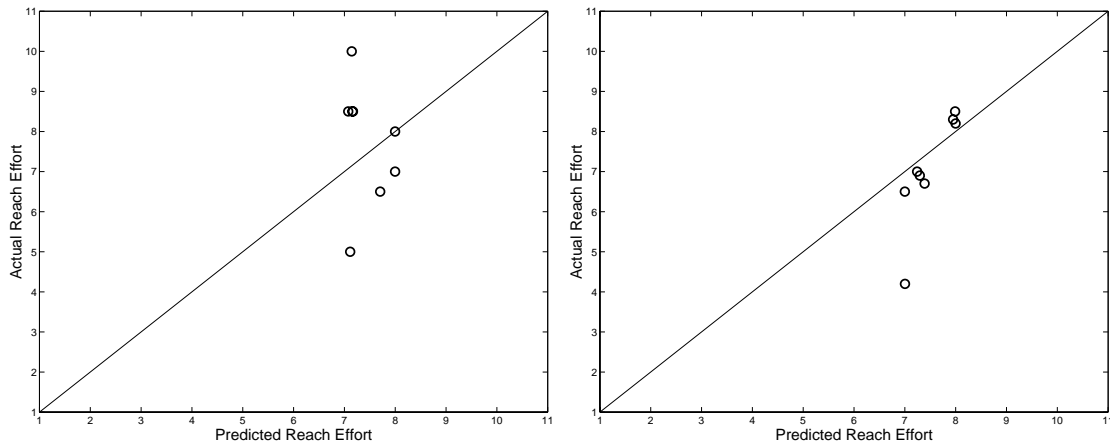


Figure 9.23: Comparison of predicted (x-axis) to actual (y-axis) reach effort using ISC_3 for subjects A (left) and B (right)

9.4.2 Discussion

A global, constrained, nonlinear optimization algorithm was used to create adaptive design of experiments. The generality of the framework lent itself well to exploring a variety of sampling criteria and constraint sets. It has been shown that the adaptive sampling successfully created experimental designs that adapted to the individual test subjects. For one subject, target locations were pushed further away because of his tall stature. For another, sample points were kept closer to the subject due to her shorter reach range. The integration of constraints into the solution of the ISC maximization problem allowed for irregularly shaped design spaces that took subject behavior into account. In addition, the ability to create smoothed kriging models was shown to be an important contribution to the study. The results of this pilot study have lead to some interesting possibilities.

Note from Figure 9.20 that two sample points for subject A were taken outside the 1600 mm radius. For that particular experiment, g_2 was eliminated from the constraint set to explore how well g_3 kept the samples within reach. The resulting DOE yielded 3 out-of-reach responses, compared to none for the ISC_3 experiment of Figure 9.22. While this does indicate that more targets were placed outside the subject's range, it did provide a reasonable constraint in that the subject was nearly capable of reaching the target. Occasionally placing the target just outside the subject's reach is not necessarily undesirable because having some sensed data helps clearly identify the maximum reach boundary.

Eliminating g_2 may prove essential in some cases. For example, subject A was actually able to reach many of the targets placed along the 1600 mm radius. An experiment designed to search for an individual subject's maximum reach range would be hindered by g_2 if the subject were tall because no out-of-reach samples

would be taken. One could measure the subject's height before testing and determine a suitable radius for g_2 , but adaptively setting the out-of-reach boundary via g_3 appears to be a more robust method. A sampling criterion similar to those applied here could be used effectively with g_1 and g_3 to adaptively locate the maximum reach range.

9.5 Chapter Summary

In this chapter, we have shown that determining an experimental design as the data is sampled can be an effective strategy. Using superEGO, models that are more accurate have been created using fewer data points than current procedures, resulting in an overall time and cost savings.

All four of the main contributions of the thesis were needed to complete the study. First, non-interpolating kriging models were created to accommodate the significant noise in the experimental data. Second, the flexibility of the framework was leveraged to easily create new sampling criteria specific to this problem. Third, this was a constrained optimization problem where analytical functions were used to describe the acceptable placement of the target. Fourth, because the constraints were inexpensive to compute, they were used directly during the ISC subproblem search without loss of accuracy. If approximations had been used instead of the constraints themselves, it is possible that any model inaccuracy could have led to a mistakenly violated constraint. Had the exclusion zone constraint been violated, the test subject would have been hit by the robot arm. Needless to say, direct evaluation of the inexpensive constraints was critical.

Having demonstrated the importance and success of the dissertation goals, we present the conclusions in the next chapter.

CHAPTER 10

Conclusions and Future Work

The goal of this dissertation was to advance the state of the art in design optimization via Bayesian analysis. Specifically, the main contributions focused on making Bayesian analysis more flexible and better suited to efficiently solve constrained optimization problems. These improvements were done by modifying an existing framework, the Efficient Global Optimization (EGO) algorithm of Schonlau, Welch and Jones [44].

We began by examining the kriging approximation and the relationship between the geostatistics and DACE approach to kriging in an effort to understand how the models in superEGO could be improved. A diagnostic for identifying certain poor model fits was proposed along with a corrective measure. The techniques for incorporating measurement error into the models were used for improving the conditioning of a nearly singular correlation matrix as design points cluster about the optimum. Smoothed models were later combined with interpolating variance in the reach effort study to better guide an adaptive sampling procedure.

To allow for flexibility, a modularized framework was created such that any quantifiable infill sampling criterion could be easily incorporated into the algorithm. A number of sampling criteria were implemented in the new superEGO framework to

illustrate the difference between many Bayesian analysis approaches. Also, a number of new criteria were introduced. A new technique for locating an initial feasible point in a highly constrained problem was successfully demonstrated. Another new technique was shown to have success at locating subsequent local optima in problems with disconnected feasible regions, even when the constraint function was difficult to model. New sampling criteria were also developed to enable Bayesian analysis to effectively generate adaptive experimental designs.

To advance the constraint handling abilities of Bayesian analysis, several techniques were compared. Some solve the infill sampling subproblem as an unconstrained problem by penalizing the criterion in some way for infeasible designs. The approach proposed in this dissertation was to directly incorporate the constraints into the infill sampling subproblem. The resulting method is very flexible and worked well on the examples, regardless of the sampling criterion chosen.

Significant advancements were made with respect to the efficiency of solving constrained optimization problems. By solving the infill sampling problem as a constrained problem, inexpensive constraints were directly and accurately accounted for during the search for the next iterate. The new search strategy was able to reduce the number of iterations required to achieve an acceptable solution by as much as 90%. When the time required for each function evaluation was significant, this also resulted in a substantial savings in overall time to solve the problem.

One of the remaining challenges for Bayesian analysis algorithms is the termination criterion. As described in the literature review, several researchers have developed stopping rules for specific sampling criteria. The flexibility of superEGO allows the user to select the stopping rule. However, while some rules work well for some choices of the sampling criterion, they lose meaning once a different sampling

criterion is selected. The challenge, therefore, is to find a suitable stopping rule (other than a function evaluation limit) that is applicable to Bayesian analysis-based optimization regardless of the sampling criterion.

An ideal stopping rule would incorporate an estimate of the model accuracy or of the probability that the global optimum has been found. These metrics provide a natural criterion for stopping that apply regardless of the sampling criterion chosen. The tradeoff between this metric and the number of function evaluations may provide a stopping rule somewhat akin to the Akaike Information Criterion (AIC) [1], [2] for choosing an appropriate model while limiting the number of basis functions. Termination could occur if either the metric value is sufficiently high, or when the ratio of the metric value to the number of sample points reaches a certain threshold. Of course, the reliability of either approach is severely limited by the difficulty of accurately predicting the model accuracy or the probability of having found the global optimum. However, if sufficiently advanced statistical techniques are developed, such stopping rules may prove useful for a wide variety of applications.

More advanced statistical techniques may also improve the efficiency of constrained optimization. For many situations in design optimization, there is a strong correlation between constraint functions (e.g., the 0-60 time and 0-85 time constraints in the vehicle study). Principal Component Analysis may provide ideas for transforming and combining the responses, thereby reducing the number of kriging model to fit.

Efficiency becomes a prime concern when the number of design variables becomes large. Like most curve fitting techniques, kriging suffers from the “curse of dimensionality”. That is, it becomes extremely difficult to fit and/or unreliable for prediction as the number of dimensions increases. The examples shown in this thesis

were restricted to a small number of design variables to ease visualization. Readers are directed to Schonlau's work [83] to see how EGO handles problems of six to ten design variables. New strategies must be employed for solving problems with a large number of design variables.

One strategy would be to make the problem tractable by reducing the number of relevant design variables through sensitivity analysis, ANOVA, or some other appropriate statistical means. Failing that, the large problem may be solved as a collection of smaller problems through decomposition methods [97]. Another approach to solving large problems would be to replace the kriging models with a simpler approximation that can be fit easily in higher dimensions and couple it with a trend for the uncertainty that returns to zero at the sample points. The Wiener model mentioned in Section 2.1.5 may provide such an approximation. While the accuracy of the model would be inferior to kriging, the problem would no longer be hindered by the impracticality of fitting models, and the search process may eventually lead to better designs as the simple models adapt to the data collected.

Even if the number of design variables remains small, the computational burden of fitting and evaluating the kriging models can become prohibitive as the number of data sample points, n , increases beyond 300 or so, due to the inversion of the $n \times n$ \mathbf{R} matrix. It may become necessary to filter out data points that are redundant (i.e., have neighboring sample points with better objective function values) to reduce the size of the data set. Another approach for aiding prediction efficiency would be to use the geostatistics method of the moving search window (see Section 3.6.3) such that only data points near the point at which to predict are considered.

Perhaps the greatest obstacle to successful use of Bayesian analysis has been the reliability of automated model fitting. Some strategies were developed in this

dissertation to detect situations known to result in poorly fit models. However, the robustness of maximum likelihood estimation (MLE) is still in question. MLE fitting can still generate very poor models even if the diagnostic criterion of Equation (3.24) is satisfied. Because most designers are not experts in kriging modeling or statistics in general, the algorithm must either be intuitive enough for the average user to easily adapt it to their needs, or it must be robust enough that the average user will not need to adapt it. Some attempt has been made here to provide such an algorithm, but the lack of a robust automated model fitting process is still at issue. If the optimization is not successful, the user may have to examine the kriging models that came out of the algorithm and understand how to rectify the situation. The existence of problems such as these prevent Bayesian analysis from becoming an attractive option to designers.

Further work is currently underway to enhance the abilities of the smoothed kriging models. Rather than having a uniformly smoothed model, we have considered incorporating a measurement error that is a function either of the location in the design space or of the predicted value of the model. The method would have a wide range of applications. In the reach effort study for example, one could account for the fact that the subjects are more consistent when assigning very high or very low reach effort values than they are in assigning values in the medium range. The kriging model could potentially include more smoothing in the region of higher intrasubject variability and remain more true to the data at the extremes.

In summary, the goals of this dissertation have been met. Some suggestions were made to improve the kriging models, and new uses were found for combining interpolating and smoothed approximations. The utility of modularizing the ISC subproblem was demonstrated by a variety of new sampling criteria designed to solve

types of optimization problems that were previously either impossible or impractical given the algorithm's original capabilities. Constrained optimization problems were addressed more rigorously and the proposed method for handling constraints appears successful. Finally, incorporating inexpensive information directly into the ISC subproblem led to a significant reduction in the time required to find good solutions. The contributions of this dissertation have advanced the field of Bayesian analysis while pointing to opportunities for further research.

APPENDICES

APPENDIX A

Software Manual

This Appendix is intended to give a basic understanding of the Matlab implementation of superEGO. The comments in the code itself will provide more detail for advanced users. To receive a copy of the source code (available for academic use only), send an email to *msasena@umich.edu* listing your name, affiliation, and a brief synopsis of how you will use the code.

A.1 Background

SuperEGO is an optimization algorithm for solving problems of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to: } & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} . \end{aligned} \tag{A.1}$$

Equality constraints can be incorporated by defining two inequality constraints with appropriate bounds on either side. SuperEGO is designed for problems where the objective and/or constraint functions are expensive to compute. It solves the problems by making a global approximation of each response with a kriging model. Due to the nature of the approximations, superEGO is intended for problems where the number of design variables and expensive constraints are both relatively small (less than 10).

The basic procedure of superEGO is as follows (see also Figure A.1):

1. Use a space-filling experimental design to obtain an initial sample of true functions.
2. Fit a kriging model to each function.
3. Numerically maximize the infill sampling criterion (ISC) to determine where to sample next.
4. Sample the point(s) of interest on the true functions and update the kriging models.
5. Check for termination, otherwise return to 2.

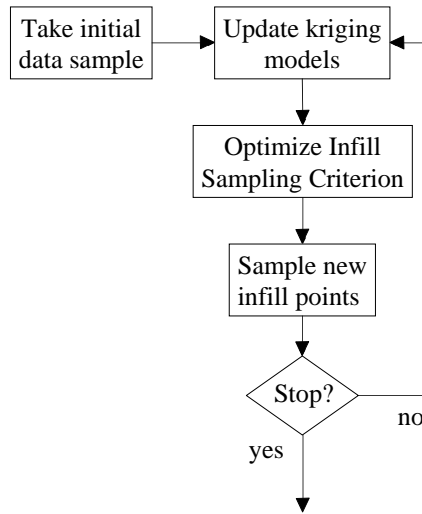


Figure A.1: Flowchart of the superEGO algorithm

A.2 Framework and Important Files

SuperEGO requires a number of files to run. The master program, `superEGO`, calls all the other subroutines needed including the function calling files created by

the user, which will be referred to as `myfun_exp` and `myfun_cheap` throughout this manual. The flow of information is summarized in Figure A.2. The solid boxes represent the various subroutines, with their respective names shown above.

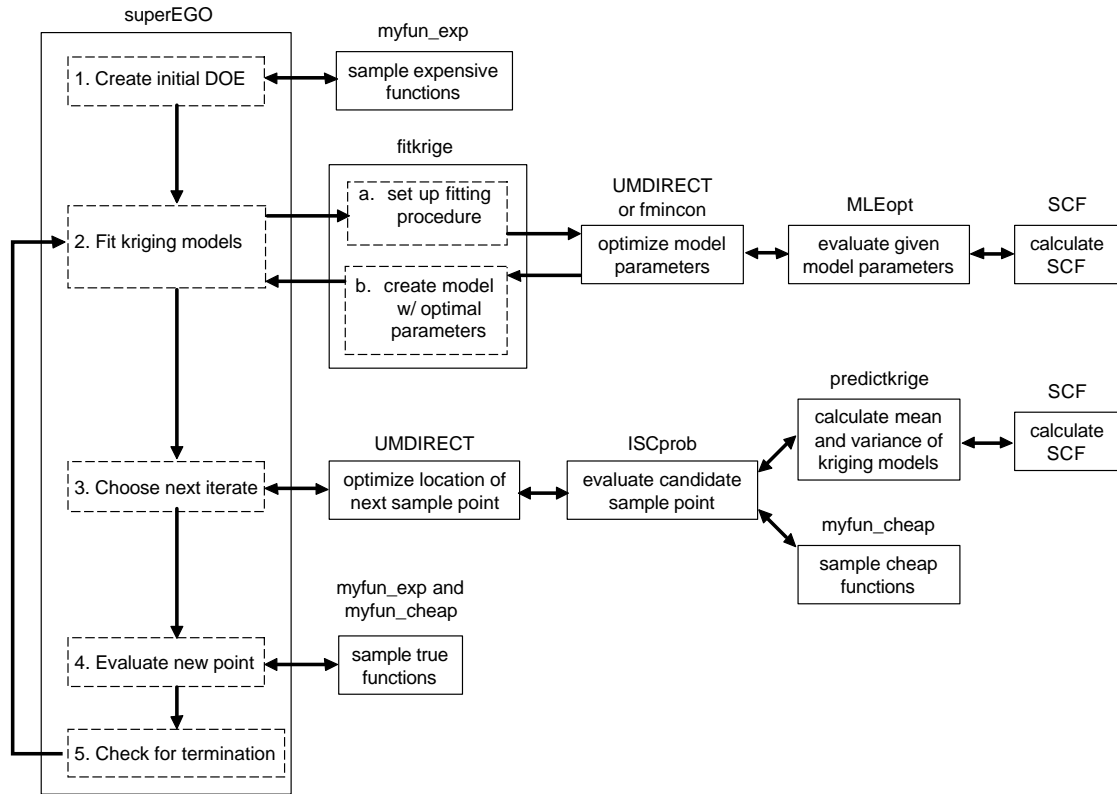


Figure A.2: Flow of information between `superEGO` and its subroutines

The main code (`superEGO`) runs the initial data sample by evaluating the expensive functions in the user-defined file `myfun_exp`. `SuperEGO` then calls out to the subroutine `fitkrige` to fit a kriging model to each of the expensive functions. `Fitkrige` in turn calls out to another algorithm to optimize the model parameters. That algorithm could be either `UMDIRECT`, a University of Michigan implementation of Jones' `DIRECT` algorithm, or `fmincon`, Matlab's SQP algorithm. The algorithm chosen depends on what fitting criterion is used, but most often, `UMDIRECT` is called.

The model fitting optimization algorithm calls out to **MLEopt** to compute the goodness of fit metric which requires **SCF** to calculate the spatial correlation function (SCF) between two sample points.

Once the optimal parameter values have been found, kriging models are created and control is returned to **superEGO**. In the next step, **UMDIRECT** is used to locate the next sample point by maximizing the infill sampling criterion (ISC). This auxiliary optimization problem is referred to as the ISC subproblem. The file **ISCprob** evaluates the sampling criterion at each candidate design point. It calls out to the subroutine **predictkrige** whenever the predicted mean and/or variance of the kriging models are needed to compute the objective or constraint functions of the ISC subproblem. **Predictkrige**, like **fitkrige**, requires the **SCF** subroutine to evaluate the covariance model. If the sampling criterion requires it, **ISCprob** also evaluates the inexpensive functions in the user-defined file **myfun_cheap**. The solution to the ISC subproblem is the location of the next sample. **SuperEGO** evaluates the true functions in **myfun_exp** and **myfun_cheap** at that location, checks if a new best feasible solution has been found and stores information from that iteration. After checking the termination criterion (usually the number of function evaluations), it either terminates or returns to the model fitting step. **Fitkrige** uses the same model parameters from the previous iteration most often, but determines a new set of optimal parameters every tenth iteration.

While many of the subroutines can be appended to **superEGO** to reduce the number of files needed, some must remain separate. When an auxiliary optimization problem is solved (e.g., finding optimal model parameters or maximizing the ISC), **UMDIRECT** or **fmincon** must evaluate the objective function of the auxiliary problem using a separate file. Therefore, at least two extra files are included in the source

code: `MLEopt` to evaluate the goodness of the model fit and `ISCprob` to evaluate the sampling criterion. In addition, the user must write three Matlab files: a script file which sets the **superEGO** options and starts the optimization, and two function files (`myfun_exp` and `myfun_cheap` in this example) which independently evaluate the expensive and inexpensive functions.

A.3 The Subroutines

While most of the subroutines used in **superEGO** are only used internally, some may be used independently. Here we describe the basic functionality of those functions, their set of input and output arguments, and the portions of code that the user may want to change. Detailed descriptions of the programs can be found in the header and comments of the Matlab files themselves. All programs have been coded by Michael Sasena except `UMDIRECT` which was coded by Ryan Fellini, Michael Sasena, and John Whitehead and `fmincon` which is part of Matlab's optim toolbox.

A.3.1 `myfun_exp` and `myfun_cheap`

These are the function files that the user must create to compute the objective and/or constraint functions of the design problem. **SuperEGO** distinguishes between so-called 'expensive' functions and 'inexpensive' functions which are evaluated hundreds of times per iteration. Doing so can result in an overall time savings when the inexpensive information is used to more intelligently dictate what expensive information is requested. Kriging models are fit only to those functions labelled 'inexpensive'. See Chapter 6 for more details.

If all functions are to be considered expensive, than all computations may be done in a single file. All functions to be considered inexpensive must be calculated using a separate function file. In both files, it is important that the first input argument be

the vector of design variables. Additional input arguments may be sent when calling these files, as will be discussed in Section A.5.

A.3.2 `fitkrige`

This function file creates a kriging model(s) for a given data set. While this function is called directly from `superEGO`, it is also useful for fitting kriging models independent of optimization. The syntax for calling `fitkrige` is as follows (with optional input arguments shown in *italics*):

```
kparam=fitkrige(x,y,method,lb,ub,theta,p,nugget,order)
```

<code>x</code>	matrix of data inputs
<code>y</code>	matrix of data outputs
<i><code>method</code></i>	choice of fitting method
<i><code>lb</code></i>	lower bounds on data input values
<i><code>ub</code></i>	upper bounds on data input values
<i><code>theta</code></i>	correlation model parameters
<i><code>p</code></i>	correlation model parameters
<i><code>nugget</code></i>	correlation model parameters
<i><code>order</code></i>	order of global trend

The first two inputs arguments are mandatory as they are the data set. Each column of `y` is a different response, thus the user may fit multiple models at once. The parameter `method` defines which fitting criterion is used (e.g., maximum likelihood), which algorithm is used when fitting the kriging model (e.g., `UMDIRECT`), and which parameters among `theta`, `p`, and `nugget` are to be fit rather than left at their nominal value. These variations lead to over 50 fitting methods, the full listing of which is found in the header of the `fitkrige` program. The default method is to use `UMDIRECT` to fit just `theta` via maximum likelihood.

The `lb` and `ub` row-vectors are used to specify the lower and upper bounds of the input data, respectively. This scales each input dimension to the $[0, 1]$ range, which helps the model fitting if there is a large disparity in the magnitude of the various inputs. If they are not specified, `fitkrige` uses the range of the data set, `x`, to define `lb` and `ub`. The input arguments `theta`, `p`, and `nugget` can be used to either specify a value for that model parameter (if `method=0`) or to provide a starting point for a gradient-based search (if `method=41-57`). Finally, the `order` of the polynomial in the global trend model can be specified to override the default of 0 (constant term model).

The output of the `fitkrige` program is the structure array `kparam`, which stores all information required to evaluate the kriging model. It includes the original data set itself, the `method` used in fitting, the values for the correlation parameters and `order`, and the range of the data set (to identify the scaling used). `Kparam` also precomputes and stores several terms of the kriging equations in order to reduce the computational effort of evaluating the kriging models. These include $\hat{\beta}$, $\hat{\sigma}_z^2$, \mathbf{R}^{-1} , and the product $\mathbf{R}^{-1} \cdot (\mathbf{y} - \hat{\beta})$.

The `fitkrige` program requires the following additional subroutines: `MLEopt`, `SCF`, `UMDIRECT`, and the Matlab `optim` toolbox. For some rarely used choices of `method`, the program calls out to the following subroutines: `errorcheck`, `predictkrige`, `variogram_aniso`, `WSS`, and `vmodel`.

In some cases, the user may wish to define a new fitting procedure. This could mean changing the optimization algorithm used to find the optimal parameter values or changing the criterion used to determine the goodness of the parameter values (e.g., maximum likelihood). There are two files they must edit: `fitkrige` and `MLEopt`. In `fitkrige`, there is a long list of ‘`elseif method == ...`’ statements

that set up the optimization run for each value of `method`. To create a new fitting procedure, the user must simply add a new ‘`elseif method == ...`’ branch defining how to call the optimization algorithm for the new procedure. Then, they can define a new fitting criterion by adding a new ‘`elseif criterion == ...`’ branch in the `MLEopt` file. To implement their new procedure, they must now simply call `fitkrige` with the new value of `method` they have defined. The user must take care not to use a value for either `method` in `fitkrige` or `criterion` in `MLEopt` that has already been defined.

A.3.3 predictkrige

Another useful program is the `predictkrige` function, whose purpose is to calculate the predicted mean and variance of the kriging models at a given location using Equations (3.11) and (3.13), respectively. This program can be used when making plots of the approximate responses, or within `ISCprob` to evaluate the mean and variance of the approximation at the candidate design point. The syntax for calling `predictkrige` is as follows (with optional input arguments shown in *italics*):

```
[ynew,yvar,lambda]=predictkrige(xnew,kparam,THETA,P,NUGGET,ORDER)
```

<code>xnew</code>	the design point(s) at which to evaluate <code>kparam</code>
<code>kparam</code>	the kriging model generated by <code>fitkrige</code>
<i><code>THETA</code></i>	correlation model parameters
<i><code>P</code></i>	correlation model parameters
<i><code>NUGGET</code></i>	correlation model parameters
<i><code>ORDER</code></i>	order of global trend

The first argument tells the program where to evaluate the kriging models. As each row in `xnew` is a design point, a matrix may be sent to evaluate multiple points at once, which is more efficient than using `predictkrige` within a loop. The second argument is the kriging model itself, created by `fitkrige`. The final four input

arguments are optional and may be used to override the SCF parameters in `kparam` when evaluating the models.

The `predictkrige` function returns up to three output arguments: `ynew`, `yvar`, and `lambda`. The first is the column vector (matrix if `kparam` contains multiple kriging models) of predicted values of the kriging model at `xnew`. The second output is the associated simple kriging variance. The third output returns the kriging weights for each prediction point defined in Equation (3.22). While most users will not need `lambda`, it is provided to help debug modeling errors. The `predictkrige` program requires only the `SCF` subroutine.

A.3.4 UMDIRECT

This program is the University of Michigan version of Jones' DIRECT algorithm [40], [43]. It is a derivative-free method for constrained (or unconstrained), non-linear optimization. DIRECT works by using Lipschitz theory to divide up the design space into hyper-rectangles. At each iteration, the set of hyper-rectangles most likely to contain good function values are further subdivided. The algorithm is used by `superEGO` to find optimal kriging model parameters and to solve the ISC subproblem. The syntax for calling `UMDIRECT` is as follows (with optional input arguments shown in italics):

```
results=UMDIRECT(fileInfo,lb,ub,options,restartFile)
```

<code>fileInfo</code>	information on the functions to be evaluated
<code>lb</code>	lower bounds on design space
<code>ub</code>	upper bounds on design space
<i><code>options</code></i>	DIRECT running parameters
<i><code>restartFile</code></i>	used to continue a previous DIRECT run

The first argument is a structure array with the following fields: **fName**, **gName**, **fParams**, **gParams**. The first two store the names of the function files that compute the objective and constraint functions, respectively. If the user would prefer to calculate both the objective and constraint functions in a single file, they must be assigned to **fName**. The **fParams** and **gParams** fields store any additional arguments that must be sent to the files **fName** and **gName** in order to execute them. The **lb** and **ub** arguments simply constrain the range of the design space. The **options** argument is another structure with the following fields:

maxfCount	limit on function calls (default = 200*(# variables))
maxIter	maximum number of iterations allowed (default = 50)
termType	termination criterion (default = fCount OR Iter exceeded)
conTol	allowable constraint violation (default = 1e-10)
display	amount of output sent to screen (default = show only final results)
saveFile	save the results to file at each iteration (default = don't save)
lgBalance	local / global balance parameter (default = 1e-4)
localSearch	allow for local optimization (default = use SQP)

Default values exist for all **options**, thus the user is not required to specify any of them. Several termination criteria exist, such as limiting the number of function evaluations or iterations, or stopping once very little progress has been made after a set number of function evaluations or iterations. The reader is referred to the comments in the header of **UMDIRECT** for more details. The **lgBalance** option can be set by the user to control the balance between local and global searching. However, the default value has been shown by Jones to be quite robust, and therefore does not require tuning to the specific problem in most cases. The **localSearch** option allows the user to turn on or off the local search capability. While allowing for local searching is often quite helpful at efficiently locating the local optima, it may launch

spurious local searches at times. If the function calls are inexpensive, the drawbacks are not significant, and local searching is recommended.

The final input argument, `restartFile`, is used to restart `UMDIRECT` in case the user would like to continue searching for a better solution beginning from the results of a previous run. It can be either the name of the `.mat` file containing the previous results, or the structure array of results itself.

The output of `UMDIRECT` is a structure array with the following fields:

<code>xBest</code>	best feasible design point found
<code>fBest</code>	objective function value at <code>xBest</code>
<code>gBest</code>	vector of constraint values at <code>xBest</code>
<code>fCount</code>	number of function evaluations performed
<code>iter</code>	number of iterations performed
<code>fChangeRate</code>	rate of change in <code>f</code> (used in <code>conWeight</code>)
<code>gChangeRate</code>	vector of rate changes for each constraint
<code>conWeight</code>	vector of constraint weights
<code>hist</code>	structure array containing history of best point
<code>rect</code>	structure array of rectangle properties
<code>inputs</code>	structure array of input arguments passed to <code>DIRECT</code>

`UMDIRECT` is a self-contained function. However, if its local search option is active, then it will call out to `fmincon` to find local solutions.

A.3.5 `fmincon`

This program is the Matlab SQP algorithm for constrained, non-linear optimization. In some cases, it is called by `superEGO` to find optimal model parameters or to solve the ISC subproblem. The reader is referred to Matlab's help for more information on this program.

A.4 Syntax and Options for SuperEGO

Now that the subroutines have been introduced, we are able to explain the main code, **superEGO**. The syntax for **superEGO**, which is very similar to **UMDIRECT**, is as follows (with optional input arguments shown in italics):

```
results = superego(fileInfo,lb,ub,options,restartFile)
```

fileInfo	information on the functions to be evaluated
lb	lower bounds on design space
ub	upper bounds on design space
<i>options</i>	superEGO running parameters
<i>restartFile</i>	used to continue a previous superEGO run

The **fileInfo** structure array has the following fields: **expFile**, **cheapFile**, **expParams**, **cheapParams** and **expObjFun**. The first two fields are the names of the function files that will be used to compute the expensive and inexpensive arguments, respectively. Any additional arguments that must be passed to these files are given via the **expParams** and **cheapParams** fields. The last field, **expObjFun**, indicates if the objective function is computed by the expensive (**fileInfo.expObjFun=1**) or cheap (**fileInfo.expObjFun=0**) function file. **SuperEGO** will store two additional fields when the algorithm terminates, **expFileNargout** and **cheapFileNargout**. These provide information as to how many output arguments are produced by each of the user-defined functions.

Similar to **UMDIRECT**, the **lb** and **ub** arguments define the range of the design space. The **options** argument is a structure array that can be used to override many of the default settings for **superEGO**. It contains the following fields:

- **display**: The amount of output sent to the screen. The default suppresses all output until the algorithm displays the final results (**options.display=0**). It

can be set to 1 to print each iteration, or to 2 to show additional information about the ISC subproblem at each iteration.

- **ISC:** Determines which sampling criterion to use when searching for the next iterate. There are a number of choices, and they are listed in the header of the `ISCprob` code. The default criterion (`options.ISC=9`) is the Watson and Barnes regional extreme function (WB_2) described in Section 4.1.7.
- **ISCParams:** Any additional parameters that may define the ISC. For example, if the user had specified `options.ISC=2` (generalized expected improvement), they could also set `options.ISCParams=5` to override the default value of g in Equation (4.6). Likewise, they could define `options.ISCParams` as a vector to create the cool criterion explained in Chapter 4. Of course, each criterion has its own definition of `ISCParams`, thus the user must inspect the criterion in the `ISCprob` file to determine how to use `ISCParams` if they want to override the default settings.
- **maxfCount:** Limits the number of function calls before termination. The default value is $50*d$, where d is the number of design variables.
- **termType:** The termination criterion to use. If none is specified, `superEGO` will terminate if any of the following occurs: `options.maxfCount` is exceeded, `options.fGoal` is met, or all functions are inexpensive and one iteration has been completed. Currently, three other options for termination criteria have been implemented. For `options.termType=1`, `superEGO` will terminate once the ratio of the value of the expected improvement function at the new iterate to the best feasible objective function value has stayed below the threshold $1e-3$ for three iterations in a row. This is a criterion proposed by Schonlau [83]. For

`options.termType=2`, `superEGO` will terminate once a feasible point has been found. For `options.termType=3`, `superEGO` will terminate if the last three iterations have not produced a change in the location of the next iterate of more than 0.1% of the range of the design space. For `options.termType=4`, `superEGO` will terminate once a sample point is taken within a given radius of a point specified by `options.termParams` (see below). This last criterion is useful for conducting experiments on problems with known solutions.

- **termParams**: Any additional parameters that may define the **termType** criterion. For example, the user may change the settings for `options.termType=2` described above such that `superEGO` will terminate once `options.termParams=5` feasible designs have been found instead of the default, 1. The user is referred to the `superEGO` file to determine what the **termParams** setting does for each **termType**.
- **fGoal**: Terminates `superEGO` once **fGoal** is reached. The default is set at $-\infty$ so that `superEGO` will only terminate by the specified **termType** or by the **maxfCount** limit.
- **conTol**: The allowable constraint violation. The default value is somewhat high ($1e-4$) because `superEGO` is designed for simulation-based optimization. The user may also define `options.conTol` as a vector in order to provide a different tolerance for each constraint. This is effective when some constraints are analytical functions, and some are simulations.
- **filter**: Apply the filtering method to the ISC subproblem (default=0, don't apply filtering). The inexpensive constraints will be met for all iterations for constrained ISC subproblems. However, the filter constraint $f(\mathbf{x}) < f_{min}$ is

only applied if `options.filter=1`. See Chapter 6 for more details about the filtering method.

- **expCon**: The method for handling expensive constraints during the ISC subproblem. The default value (`options.expCon=3`) means that the predicted value of each expensive constraint must be no greater than `options.conTol`. Other settings allow for expensive constraints to be handled using the probability of feasibility or the expected violation metric. See Section 5.2 and the `ISCprob` code for more information.
- **localSearch**: Enable periodic local search. The local search consists of locating the minimum of the kriging model of the objective function every tenth iteration and sampling the true functions at that location.
- **infillMethod**: Force a specific algorithm for ISC subproblem. The default is to use `UMDIRECT`, either constrained or unconstrained, depending on the problem statement and the choice of `options.ISC`. The user, however, may choose to force a specific method in order to override the default selection.

<code>options.infillMethod=1</code>	unconstrained Tomlab DIRECT
<code>options.infillMethod=2</code>	constrained Tomlab DIRECT
<code>options.infillMethod=3</code>	constrained SQP using <code>fmincon</code>
<code>options.infillMethod=4</code>	constrained Tomlab DIRECT on subregion
<code>options.infillMethod=5</code>	unconstrained <code>UMDIRECT</code>
<code>options.infillMethod=6</code>	constrained <code>UMDIRECT</code>

- **fitMethod**: Force a specific method for model fitting. This sets the value of the `method` input to the `fitkrige` program. The default is to use `UMDIRECT` to fit theta only via maximum likelihood (`options.fitMethod=21`).

- **fitOrder**: Force a specific value for the polynomial order of the global trend of the kriging model. This sets the value of the **order** input to the **fitkrige** program. The default is **options.fitOrder=0**, use a constant term model.
- **loadDOE**: Allows the user to pass their own DOE to initialize **superEGO**. The argument must be sent as an $n \times d$ matrix, where n is the number of design points to sample and d is the number of design variables. Also, the inputs must be scaled to the $[0,1]$ range for all variables. They will subsequently be scaled to the $[lb,ub]$ range by **superEGO**.
- **modelID**: Use predetermined values for kriging model parameters, if available. For some examples shown in this dissertation, good covariance model parameters have been found from fitting data samples. Those values were stored within **superEGO** to avoid having to fit models at each iteration. The user is referred to the **superEGO** code to see which functions have pre-computed model parameters. **SuperEGO** is able to determine if the problem uses one of these examples by checking the name of the problem files. The Gomez #3 example consists of the objective and one constraint function. If the user would like to consider the objective function as expensive (i.e., create a kriging model), but consider the constraint function as inexpensive (i.e., use the constraint function directly), they would set **options.modelID=[1]**. However, if they wanted the opposite, they would use **options.modelID=[2]**. If they wanted both functions to be labelled as expensive, they would use **options.modelID=[1 2]**. Leaving **options.modelID** undefined will let the algorithm fit the expensive functions via **fitkrige** rather than using the pre-computed model parameters. The user may also enter new kriging model parameters for their own models into the

main code in order for them to be recognized by `options.modelID`.

- **saveFile:** The name of file to save the results to at each iteration. If no filename is specified, the default is to save the results to a file named 'results_myfun_exp', where 'myfun_exp' refers to the name of the expensive function file. If only inexpensive functions exist, that file name will be used instead. The user may prevent **superEGO** from saving the results to a file each iteration by setting `options.saveFile='nosave'`.

As with **UMDIRECT**, a previous **superEGO** run can be continued by passing either the previous results file name, or the results structure array itself via the optional `restartFile` input argument. The results structure output from **superEGO** has the following fields:

<code>xBest</code>	best feasible design point found
<code>fBest</code>	objective function value at <code>xBest</code>
<code>gBest</code>	vector of constraint values at <code>xBest</code>
<code>lb</code>	lower bounds on design space
<code>ub</code>	upper bounds on design space
<code>fCount</code>	number of function evaluations performed
<code>fCountCheap</code>	number of inexpensive function evaluations performed
<code>iter</code>	number of iterations performed
<code>minIndex</code>	function evaluation at which <code>xBest</code> was found
<code>history</code>	structure array containing history of <code>xBest</code>
<code>inputs</code>	set of all row-vector inputs sent to the function file
<code>outputs</code>	set of all row-vector outputs ordered as $[f, g_1, g_2, \dots]$
<code>kparam</code>	the kriging model generated by <code>fitkrige</code>
<code>infill</code>	iterations' ISC choice and best found value
<code>flag</code>	indicates if superEGO converged or crashed
<code>prob</code>	structure array defining the problem statement
<code>fileInfo</code>	<code>fileInfo</code> argument originally passed to superEGO
<code>options</code>	<code>options</code> argument originally passed to superEGO

Note that the expensive constraints are listed first in the `results.outputs` matrix.

The user may wish to define a sampling criterion specific to their problem as was done for the case studies in this thesis. Doing so requires editing the portion of the `ISCprob` file that defines each ISC as `'elseif options.ISC == ...'`. The user must simply insert a new `elseif` branch and create the function to define `ISC_f`, making sure to not use a previously defined value for `options.ISC`.

A.5 Examples of How to Execute SuperEGO

In order to execute `superEGO`, the user must write a script file. Below are several examples of script files to set up the `superEGO` parameters and begin the optimization.

A.5.1 Example 1

In the first example, the user is setting up an ordinary optimization problem that consists of both expensive and inexpensive functions. The objective function is expensive.

```
fileInfo.expFile='myfun_exp'; %name of expensive function file
fileInfo.cheapFile='myfun_cheap'; %name of cheap function file
fileInfo.expParams=[10]; %extra parameter sent to 'myfun_exp.m'
fileInfo.expObjFun=1; %the objective function is expensive
lb=[0 5 0]; %lower bounds on design variables
ub=[10 25 50]; %upper bounds on design variables
options.maxfCount=75; %limit on number of function calls
options.conTol=[1e-3 1e-2]; %tolerance on the two constraints
options.display=1; %show progress every iteration
options.saveFile='myfile'; %name of file to save each iteration

myresults=superEGO(fileInfo,lb,ub,options); %start superEGO
```

Note that the names of the function evaluation files are passed as strings by putting single quotes on either side. The user may also include the `.m` after the filename, but it is not necessary to do so. Once the optimization is complete, the

results are stored in two ways: the structure array output of the function call to **superEGO** (named **myresults** in this example) is saved to the workspace, and the same structure array is saved to the Matlab data file **myfile.mat**, where it is named **results**.

A.5.2 Example 2

If the user wanted to continue with the same problem shown in Example 1 for another 25 function calls, they could restart **superEGO** with either the two following two lines of code:

```
newoptions.maxfCount=100; %set a higher limit on the fCount
mynewresults=superEGO([],[],[],newoptions,'myfile');
OR
newoptions.maxfCount=100; %set a higher limit on the fCount
mynewresults=superEGO([],[],[],newoptions,myresults);
```

Note that the user only had to define the specific **options** that they wished to change from the previous results. All the remaining **options** would take on the values from the original results stored in the file **myfile.mat** or the variable **myresults** in the workspace.

It should be noted that the user may also change either of the **lb** or **ub** input arguments when continuing an optimization run. If any of the problem bounds have been changed, say to narrow the search space, the user will be shown the old and new bounds and prompted to either accept the new bounds or terminate the run. If the **fileInfo** has been changed, an error occurs because the user is attempting to use previous results with a different function file.

A.5.3 Example 3

In some instances, the user may wish to keep more control over the sampling of the functions due to the extreme cost of the functions in the design problem. For example, they may wish to make an initial set of function calls before starting any optimization, or to reuse existing data. They may also wish to fit the model in order to more easily verify that the model is a good fit and visualize the results before starting the optimization.

```
%define a simple DOE of 4 experiments and run simulation
X = [[1 0 0];[-1 0 0];[0 1 0];[0 0 0]];
for i=1:4
    y(i) = myfun_exp(X(i,:));
end
y=y(:)'; %ensure it's a column vector

%create the kriging model
mymodel = fitkrige(X,y);
```

At this point, the user is free to inspect the kriging model stored in `mymodel`. Obviously, the illustration above is not practical because only 4 data points were used for a 3 dimensional data set. When the user is satisfied that the model sufficiently captures the trend, they may then import that information into `superEGO` and begin optimization by creating a structure array called `results` and passing it to `superEGO` as if it were from a previous optimization run to be restarted. The names of the fields must match exactly the names defined by the usual `superEGO results` array. Only the fields for the input and output data (and the kriging model if it has been fit) should be created in this way. `SuperEGO` will fill in the missing information itself.

```

%define results structure array
results.inputs=X;
results.outputs=y;
results.kparam=mymodel;

%define input arguments to superEGO
fileInfo.expFile='myfun_exp';
lb=[-1 -1 -1];
ub=[1 1 1];
options.maxfCount=10; %only run 6 more points using superEGO
options.infill=11; %use maximum variance criterion
options.saveFile='mynewresults'; %store new results to file
options.fitMethod=0; %just reuse model parameters w/o refitting

%start superEGO using previously gathered data
myresults=superEGO(fileInfo,lb,ub,options,results);

```

Note that the setting `options.fitMethod=0` is used so that the kriging model parameters will not be refit at each iteration. As long as the model built in the preliminary stages shown above is acceptable, then setting that option will reduce the overhead and fix the kriging model parameters at known good values to avoid potentially poor fits as during the optimization. Once the `options.maxfCount` limit has been reached, the algorithm will terminate. The user may then evaluate the results, determine how to proceed next, and continue the optimization as shown in Example 2. They are of course free to change the design variable bounds or any of the `options` settings during restart.

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and F. Csáki, editors, *Second International Symposium on Information Theory*, pages 267–280, 1973. Budapest: Akademiai Kiado.
- [2] H. Akaike. A new look at the statistical model identification. In *IEEE Transactions on Automatic Control AC-19*, pages 716–723, 1974. Budapest: Akademiai Kiado.
- [3] N. Alexandrov, J.E. Dennis, Jr., R.M. Lewis, and V. Torczon. A trust region framework for managing the use of approximate models in optimization. *Structural Optimization*, 15(1):16–23, 1998.
- [4] F. Archetti and B. Betrò. A probabilistic algorithm for global optimization. *Calcolo*, 16(3):335–343, 1980.
- [5] A. Atkinson, P. Hackl, and W. Müller, editors. *Proceedings of MODA6: Model Oriented Data Analysis*. Physica Verlag, 2001.
- [6] A. Atkinson, L. Pronzato, and H.P. Wynn, editors. *MODA5: Advances in model-oriented data analysis and experimental design: Proceedings of the 5th international workshop*. Physica Verlag, 1998.
- [7] C. Audet, J.E. Dennis, Jr., D.W. Moore, A. Booker, and P.D. Frank. A surrogate-model-based method for constrained optimization. In *Proceedings of the 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2000. Paper No. AIAA-2000-4891.
- [8] V.O. Balabanov, A.A. Giunta, O. Golovidov, B. Grossman, W.H. Mason, L.T. Watson, and R.T. Haftka. Reasonable design space approach to response surface approximation. *Journal of Aircraft*, 36(1):308–315, January - February 1999.
- [9] S.M. Batill, M.A. Stelmack, and R.S. Sellar. Framework for multidisciplinary design based on response surface approximation. *Journal of Aircraft*, 36(1):287–297, January - February 1999.
- [10] J.O. Berger. *Statistical Decisions Theory and Bayesian Analysis*. Springer Series in Statistics. Springer, New York, second edition, 1985.

- [11] B. Betrò. Bayesian methods in global optimization. *Journal of Global Optimization*, 1(1):1–14, 1991.
- [12] B. Betrò and F. Schoen. A stochastic technique for global optimization. *Computers and Mathematics with Applications*, 21(6–7):127–133, 1991.
- [13] M. Björkman and K. Holström. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1:373–397, 2000.
- [14] G.E.P. Box and N.R. Draper. *Empirical Model-Building and Response Surfaces*. J. Wiley & Sons, New York, 1987.
- [15] G.E.P. Box, W. Hunter, and J. Hunter. *Statistics for Experimenters*. Wiley, Inc., New York, 1978.
- [16] J. Calvin. Convergence rate of the p-algorithm for optimization of continuous functions. In P.M. Pardalos, editor, *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*. Kluwer Academic Publishers, Boston, 1999.
- [17] J. Calvin and A. Žilinskas. On the convergence of the P-Algorithm for one-dimensional global optimization of smooth functions. *Journal of Optimization Theory and Applications*, 102(3):479–495, 1999.
- [18] J.M. Calvin and A. Žilinskas. On convergence of a P-Algorithm based on a statistical model of continuously differentiable functions. *Journal of Global Optimization*, 19:229–245, 2001.
- [19] T. Coleman, M.A. Branch, and A. Grace. *Optimization Toolbox, For Use with Matlab: User’s Guide*. The MathWorks, Inc., Nattick, MA, 1999.
- [20] D.D. Cox and S. John. A statistical method for global optimization. In *Proceedings of the 1992 International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1241–1246, 1992.
- [21] D.D. Cox and S. John. SDO: A statistical method for global optimization. In M.N. Alexandrov and M.Y. Hussaini, editors, *Multidisciplinary Design Optimization: State of the Art*, pages 315–329. SIAM, Philadelphia, 1997.
- [22] P. Craven and G. Wahba. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerical Mathematics*, 31:377–403, 1978.
- [23] N. Cressie. Spatial prediction and ordinary kriging. *Mathematical Geology*, 20(4):405–421, 1997.
- [24] M. Cuddy and K. Wipke. Analysis of the fuel economy benefit of drivetrain hybridization. *SAE Paper 970289*, 1997.

- [25] L.C.W. Dixon and G.P. Szegö. *Towards Global Optimisation 2*, chapter The Global Optimisation Problem: An Introduction. North-Holland Publishing Company, New York, 1978.
- [26] N. Dyn, D. Levin, and S. Rippa. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal of Scientific and Statistical Computing*, 7(2):639–659, 1986.
- [27] J.F. Elder IV. Global R^d optimization when probes are expensive: the GROPE algorithm. In *Proceedings of the 1992 International Conference on Systems, Man, and Cybernetics*, volume 1, pages 577–582, 1992.
- [28] R.T. Farouki. Analytic computation geometry: A compendium for engineers. Coursepack for the University of Michigan course ME 554, 1996.
- [29] R. Fellini, P. Papalambros, and T. Weber. Application of a product platform design process to automotive powertrains. In *Proceedings of the 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2000. Paper No. AIAA-2000-4849.
- [30] N. Flournoy, W.F. Rosenberger, and W.K. Wong, editors. *New developments and applications in experimental design: Proceedings of the Joint AMS-IMS-SIAM Summer Research Conference*, volume 34 of *IMS Lecture Notes*. Institute of Mathematical Statistics, 1997.
- [31] S. Freud. *An Outline of Psychoanalysis*. Norton, New York, 1949.
- [32] A.A. Giunta, V. Balabanov, D. Haim, B. Grossman, L.T. Mason, and R.T. Haftka. Multidisciplinary optimisation of a supersonic transport using design of experiments theory and response surface modelling. *The Aeronautical Journal*, pages 347–356, October 1997.
- [33] S. Gomez and A. Levy. The tunnelling method for solving the constrained global optimization problem with several non-connected feasible regions. In A. Dold and B. Eckmann, editors, *Lecture Notes in Mathematics 909*, pages 34–47. Springer-Verlag, 1982.
- [34] P. Goovaerts. *Geostatistics for Natural Resources Evaluation*. Oxford University Press, New York, 1997.
- [35] H.M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.
- [36] J.P. Hardwick and Q.F. Stout. Flexible algorithms for creating and analyzing adaptive sampling procedures. In N. Flournoy, W.F. Rosenberger, and W.K. Wong, editors, *New Developments and Applications in Experimental Design: Selected Proceedings of a 1997 Joint AMS-IMS-SIAM Summer Conference*, *IMS Lecture Notes, Monograph 34*, pages 91–105. Institute of Mathematical Statistics, 1998.

- [37] J.W. Hardy. An implemented extension of branin's method. In L.C.W. Dixon and G.P. Szegö, editors, *Towards Global Optimization*, pages 117–142. North-Holland Publishing Company, Amsterdam, 1975.
- [38] K. Holstrom. Homepage. Technical report, Department of Mathematics and Physics, Mälardén University, Västerås, Sweden, 2001. <http://www.ima.mdh.se/tom/>.
- [39] M. Jansen, M. Malfait, and A. Bultheel. Generalized cross validation for wavelet thresholding. *Signal Processing*, 56:33–44, 1996.
- [40] D.R. Jones. The DIRECT global optimization algorithm. *Encyclopedia of Optimization*, 1:431–440, 2001.
- [41] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.
- [42] D.R. Jones. Personal communication. General Motors Technical Center, Warren, MI, 2002.
- [43] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181, 1993.
- [44] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.
- [45] M. Kaufman, V. Balabanov, S.L. Burgee, A.A. Giunta, B. Grossman, R.T. Haftka, W.H. Mason, and L.T. Watson. Variable-complexity response surface approximations for wing structural weight in HSCT design. *Computational Mechanics*, 18:112–126, 1996.
- [46] J. Kleijnen. *Statistical Tools for Simulation Practitioners*. Dekker, New York, 1987.
- [47] P.N. Koch, J.K. Simpson, and F. Mistree. Statistical approximations for multidisciplinary design optimization: The problem of size. *Journal of Aircraft*, 36(1):275–286, January - February 1999.
- [48] G.J. Kott and G. Gabriele. A tunnel based method for mixed discrete constrained nonlinear optimization. In *ASME Design Automation Conference*, 1998.
- [49] D.G. Krige. A statistical approach to some mine valuations and allied problems at the witwatersrand. Master's thesis, University of Witwatersrand, 1951.
- [50] H.J. Kushner. Stochastic model of an unknown function. *Journal of Mathematical Analysis and Application*, 5:150–167, 1962.

- [51] H.J. Kushner. A new method of locating the maximum of an arbitrary multi-peak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106, March 1964.
- [52] Y. Lin, K. Krishnapur, J.K. Allen, and F. Mistree. Robust concept exploration in engineering design: Metamodeling techniques and goal formulations. In *Proceedings 2000 ASME Design Engineering Technical Conference*, 2000. Paper No. DETC2000/DAC-14283.
- [53] M. Locatelli. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10:57–76, 1997.
- [54] M. Locatelli and F. Schoen. An adaptive stochastic global optimization algorithm for one-dimensional functions. *Annals of Operations Research*, 5:263–278, 1995.
- [55] G. Matheron. The theory of regionalized variables and its applications. Les Cahiers du Centre de Morphologie Mathematiques de Fontainebleau 5, Fontainebleau, France, 1971.
- [56] F. Mistree, O.F. Hughes, and B.A. Bras. The compromise decision support problem and the adaptive linear programming algorithm. In M.P. Kamat, editor, *Structural Optimization: Status and Promise*, pages 247–289. AIAA, Washington, D.C., 1993.
- [57] T.J. Mitchell and M.D. Morris. The spatial correlation function approach to response surface estimation. In *Proceedings of the 1992 Winter Simulation Conference*, pages 565–571, 1992.
- [58] J. Mockus. *Bayesian approach to global optimization*. Kluwer Academic Publishers, New York, 1989.
- [59] J. Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4:347–365, 1994.
- [60] J. Mockus, V. Tiešis, and A. Žilinskas. The application of bayesian methods for seeking the extremum. In *Toward Global Optimisation 2*, pages 117–130. North-Holland Publishing Company, New York, 1978.
- [61] D.C. Montgomery. *Design and Analysis of Experiments*. J. Wiley & Sons, New York, fourth edition, 1997.
- [62] R.H. Myers and D.C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. J. Wiley & Sons, New York, 1995.
- [63] National Renewable Energy Laboratory, Department of Energy. Homepage, 2002. URL: <http://www.ctts.nrel.gov/analysis/>.

- [64] S.A. Nelson II and P.Y. Papalambros. The use of trust region algorithms to exploit discrepancies in function computation time within optimization models. *ASME Journal of Mechanical Design*, 121(4):552–556, 1999.
- [65] S.A. Nelson II, M.B. Parkinson, and P.Y. Papalambros. Multicriteria optimization in product platform design. *ASME Journal of Mechanical Design*, 123(2):199–204, 2001.
- [66] P.Y. Papalambros and D.J. Wilde. *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press, New York, second edition, 2000.
- [67] C.D. Perttunen. A nonparametric global optimization method using the rank transformation. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 888–893, 1989.
- [68] C.D. Perttunen. A computational geometric approach to feasible region division in constrained global optimization. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 585–590, 1991.
- [69] C.D. Perttunen. A study of alternative stochastic models in kushner-based global optimization methods. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 597–601, 1991.
- [70] C.D. Perttunen and B.E. Stuckman. The rank transformation applied to a multi-univariate method of global optimization. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 217–220, 1989.
- [71] C.D. Perttunen and B.E. Stuckman. The rank transformation applied to a multivariate method of global optimization. *IEEE Transactions of the International Conference on Systems, Man, and Cybernetics*, 20(5):1216–1220, 1990.
- [72] D. Posa. Conditioning of the stationary kriging matrices for some well-known covariance models. *Mathematical Geology*, 21(7):755–765, 1989.
- [73] L. Pronzato, E. Walter, A. Venot, and J.F. Lebruchec. A general purpose global optimizer: Implementation and applications. *Mathematics and Computers in Simulation*, 26:412–422, 1984.
- [74] J.E. Renaud and G.A. Gabrielle. Approximation in nonhierarchical system optimization. *AIAA Journal*, 32(1):198–205, January 1994.
- [75] K. Ritter. Approximation and optimization on the wiener space. *Journal of complexity*, 6:337–364, 1990.
- [76] T.J. Ross. *Fuzzy Logic with Engineering Applications*. McGraw-Hill, New York, 1995.

- [77] J. Sacks, S.B. Schiller, and W.J. Welch. Design for computer experiments. *Technometrics*, 31:41–47, 1989.
- [78] J. Sacks, W.J. Welch, W.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989. (with discussion).
- [79] M.J. Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. Department of mechanical engineering, University of Michigan, Ann Arbor, MI, 2002.
- [80] M.J. Sasena, P.Y. Papalambros, and P. Goovaerts. Metamodeling sampling criteria in a global optimization framework. In *Proceedings of the 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2000. Paper No. AIAA-2000-4921.
- [81] M.J. Sasena, P.Y. 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.
- [82] M.J. Sasena, P.Y. Papalambros, and P. Goovaerts. Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization*, 34(3):263–278, 2002.
- [83] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Waterloo, Canada, 1997.
- [84] M. Schonlau, W.J. Welch, and D.R. Jones. Global versus local search in constrained optimization of computer models. Technical Report RR-97-11, Institute for Improvement in Quality and Productivity, University of Waterloo, Waterloo, Ontario, Canada, 1997.
- [85] R. Senger. Validation of ADVISOR as a simulation tool for a series hybrid electric vehicle using the virginia tech futurecar lumina. Master’s thesis, Department of Mechanical Engineering, Virginia Polytechnic Institute & State University, 1997.
- [86] T.W. Simpson. *A Concept Exploration Method for Product Family Design*. PhD thesis, Department of Mechanical Engineering, Woodruff School of Mechanical Engineering, Georgia Institute of Technology, 1998.
- [87] 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(2):129–150, 2001.
- [88] M. Smith. *Neural Nets for Statical Modeling*. Von Nostrand Rienhold, 1993.

- [89] G. Strang. Wavelets and dilation equations: a brief introduction. *SIAM Review*, 31:614–627, 1989.
- [90] R.G. Strongin. Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves. *Journal of Global Optimization*, 2:357–378, 1992.
- [91] B.E. Stuckman. A global search method for optimizing nonlinear systems. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 965–977, 1989.
- [92] B.E. Stuckman and E.E. Easom. A comparison of bayesian/sampling global optimization techniques. *IEEE Transactions of the International Conference on Systems, Man, and Cybernetics*, 22(5):1024–1032, 1992.
- [93] B.E. Stuckman and P. Scannell. A multidimensional bayesian global search method which incorporates knowledge of an upper bound. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, 1991.
- [94] S.K. Thompson and G.A.F. Seber. *Adaptive sampling*. John Wiley & sons, 1996.
- [95] V. Torczon and M.W. Trosset. Using approximation to accelerate engineering design optimization. In *Proceedings of the 7th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 1998. Paper No. AIAA-98-4800.
- [96] A. Törn and A. Žilinskas. *Global Optimization*. Springer-Verlag, New York, 1989.
- [97] T.C. Wagner. *A General Decomposition Methodology for Optimal System Design*. PhD thesis, University of Michigan, 1993.
- [98] G. Wahba. Spline models for observational data. In *Regional Conference Series in Applied Mathematics*, number 59. Society for Industrial & Applied Mathematics, 1998.
- [99] A. G. Watson and R. J. Barnes. Infill sampling criteria to locate extremes. *Mathematical Geology*, 27(5):589–608, 1995.
- [100] W.J. Welch, R.J. Buck, J. Sacks, H.P. Wynn, W.J. Mitchell, and M.D. Morris. Screening, predicting, and computer experiments. *Technometrics*, 34:15–25, 1992.
- [101] S. Zacks. Adaptive designs for parametric models. In S. Ghosh and C.R. Rao, editors, *Handbook of Statistics*, volume 13, chapter 5, pages 247–289. AIAA, Washington, D.C., 1993.

- [102] A. Žilinskas. One-step bayesian method of the search for extremum of an one-dimensional function. *Cybernetics*, 1:139–144, 1975.
- [103] A. Žilinskas. The use of statistical models for construction of multimodal optimization algorithms. In *Proceedings of the Third Czechoslovak-Soviet-Hungarian Seminar on Information Theory*, pages 219–224. Czechoslovak Academy of Science, Prague, 1980.
- [104] A. Žilinskas. Two algorithms for one-dimensional multimodal minimization. *Mathematische Operationsforschung und Statistik, Series Optimization*, 12:53–63, 1981.
- [105] A. Žilinskas. Axiomatic characterization of a global optimization algorithm and investigation of its search strategy. *Operations Research Letters*, 4(6):35–39, 1985.
- [106] A. Žilinskas. A review of statistical models for global optimization. *Journal of Global Optimization*, 2(2):145–163, 1992.