FELIPE ANTONIO CHEGURY VIANA

SURROGATE MODELING TECHNIQUES AND HEURISTIC OPTIMIZATION METHODS APPLIED TO DESIGN AND IDENTIFICATION PROBLEMS

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
2008

FELIPE ANTONIO CHEGURY VIANA

SURROGATE MODELING TECHNIQUES AND HEURISTIC OPTIMIZATION METHODS APPLIED TO DESIGN AND IDENTIFICATION PROBLEMS

Tese apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de **DOUTOR EM ENGENHARIA MECÂNICA.**

Área de Concentração: Mecânica dos Sólidos e Vibrações

Orientador: Prof. Dr. Valder Steffen Jr.

UBERLÂNDIA – MG

2008

Dados Internacionais de Catalogação na Publicação (CIP)

V614s Viana, Felipe Antonio Chegury, 1980-
Surrogate modeling techniques and heuristic optimization methods applied to design and identification problems / Viana, Felipe Antonio Chegury. - 2008.
156 f. : il.

Orientador: Valder Steffen Jr.

Tese (doutorado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica.
Inclui bibliografia.

1. Otimização matemática - Teses. 2. Vibração - Teses. I. Steffen Jr, Valder. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Mecânica. IV. Título.

CDU: 621:51-7

Elaborada pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

“Nothing shocks me. I'm a scientist.”
Harrison Ford, as Indiana Jones

To my wife Nádia and my daughter Bruna.

ACKNOWLEDGMENTS

I would like to express my gratitude to my parents Celso and Valéria. The good education they gave to me, their unconditional love, appreciation, and incentive were and will always be fundamental in my life.

I thank Nádia and Bruna for dreaming my dreams with me, and for all love they have devoted to me all these years.

I am grateful to have true friends in Anthony, Rômulo and Anderson for lending me a shoulder when I had a bad day and for sharing with me the happy moments.

I also thank my academic advisor Prof. Dr. Valder Steffen for his guidance, patience, and encouragement throughout my graduate school. It has been an amazing experience to work under his supervision.

I would like to especially thank my advisory committee members, Prof. Luiz Góes, Prof. Francisco Soeiro, Prof. Domingos Rade, and Prof. Sezimária Saramago, for their willingness to serve on my committee, for evaluating my thesis, and for offering constructive criticism that has helped improved this work.

I would like to thank Dr. Garret Vanderplaats, Dr. Gerhard Venter, and Dr. Vladimir Balabanov, from Vanderplaats Research and Development Inc.; Dr. Luiz Góes, Mr. Nei Brasil, and Mr. Benedito Maciel from the Technological Institute of Aeronautics in São José dos Campos; Dr. Sérgio Butkewitsch, Dr. Marcus Leal, Mr. Marcelo Zanini and Mr. Sandro Magalhães from EMBRAER; Dr. Raphael Haftka, and Dr. Tushar Goel from the University of Florida in Gainesville; and Dr. Wei Shyy from the University of Michigan in Ann Arbor for the enriching opportunity to work with them along this doctoral research.

My special and sincere thanks to the staff and colleagues at FEMEC-UFU, for making the graduate school a great experience.

Finally, I am deeply thankful to the Brazilian research agencies CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the financial support during my PhD studies.

SURROGATE MODELING TECHNIQUES AND HEURISTIC OPTIMIZATION METHODS APPLIED TO DESIGN AND IDENTIFICATION PROBLEMS

TABLE OF CONTENTS

VIANA, F. A. C., 2005, **Surrogate Modeling Techniques and Heuristic Optimization Methods Applied to Design and Identification Problems**. 2008. 142 pages. PhD Thesis. Universidade Federal de Uberlândia, Uberlândia, Brazil.

Abstract

Advances in computer throughput have helped to popularize numerical optimization as an engineering tool. However, they also favor an increase in complexity of the state-of-the-art simulation. As a result, the computational cost of complex high-fidelity engineering simulations often makes it difficult to rely exclusively on simulation for optimization. This doctoral research presents an effort in combining global optimization and surrogate modeling techniques as a way to rationally use the computer budget and increase the information level obtained during the optimization task. The above mentioned techniques were used in the solution of the continuous-discrete problems of the optimal design of a vehicular structure and aircraft structural components; identification of aircraft longitudinal stability and control derivatives and non-linear landing gear model and the improvement of surrogate models through extra simulations. Besides, the solution of the combinatorial problem of the optimal Latin Hypercube has been implemented. At the end of the research, the main learning is that, as it also happens with classical optimization algorithms, the success in using heuristic methods is highly dependent on a number of factors, such as the level of fidelity of the simulations, level of previous knowledge of be problem, and, of course, computational resources. This way, the use of variable fidelity and surrogate models together with heuristic optimization methods is a successful approach, since heuristic algorithms do not require gradient information (i.e., resources are directly used for the search, and there is no propagation of the errors due to the computation of the gradients); and they have the trend to find the global or near global solution. In some cases, a cascade-type combination of heuristic and classical optimization methods may be a suitable strategy for taking advantage of the global and local search capabilities of the individual algorithms.

Keywords: Heuristic optimization, surrogate modeling, design optimization, inverse problems, variable-fidelity modeling.

VIANA, F. A. C., 2005, **Técnicas de Meta-Modelagem e Métodos Heurísticos de Otimização Aplicados a Problemas de Projeto e Identificação**. 2008. 142 páginas. Tese de Doutorado. Universidade Federal de Uberlândia, Uberlândia, Brazil.

Resumo

Avanços na capacidade de processamento computacional popularizaram a otimização numérica como uma ferramenta de engenharia. Contudo, eles favoreceram também o aumento na complexidade das simulações. Como resultado, o custo computacional de simulações complexas de alta fidelidade em engenharia dificultam o uso exclusivo de simulações em otimização. Esta pesquisa de doutorado representa um esforço em combinar técnicas de otimização global e meta-modelagem como uma forma de usar racionalmente os recursos computacionais e aumentar o nível de informação obtida durante a tarefa de otimização. As técnicas mencionadas acima foram usadas na resolução dos problemas contínuo-discretos do projeto ótimo de uma estrutura veicular e componentes estruturais aeronáuticos; identificação de derivadas de controle e estabilidade longitudinal de aviões e modelo não linear de trem de pouso; e melhoramento de meta-modelos via adição de simulações. Além disso, a solução do problema combinatorial do hipercubo latino ótimo também foi implementado. Ao final da pesquisa, a principal lição é que, assim como também acontece com algoritmos clássicos de otimização, o sucesso no uso de métodos heurísticos é altamente dependente do problema, nível de fidelidade das simulações, nível das informações já conhecidas do problema, e, obviamente, recursos computacionais. Desta forma, o uso de fidelidade variável e meta-modelagem juntamente com métodos heurísticos de otimização é uma estratégia bem sucedida, uma vez que métodos heurísticos não requerem informação sobre o gradiente (isto é, os recursos são diretamente usados na busca e não há propagação dos erros devido ao cálculo dos gradientes); e eles têm a tendência em encontrar a solução global ou próxima dela. Em alguns casos, uma combinação em cascata de métodos de otimização heurísticos e clássicos pode compor uma estratégia viável para aproveitar as capacidades de busca global e local dos algoritmos individuais.

Palavras-Chave: Métodos heurísticos de otimização, meta-modelagem, otimização de projeto, problemas inversos, modelagem por fidelidade variável.

Nomenclature

Chapter 1

$f(\mathbf{x})$	objective function
$\mathbf{f}(\mathbf{x})$	vector of objective functions
$g(\mathbf{x})$	inequality constraint
$h(\mathbf{x})$	equality constraint
n_{cnstrt}	total number of constraints (inequality and equality)
n_{dv}	number of design variables
$n_{ineqcnstrt}$	number of inequality constraints
$\mathbf{x}$	vector of design variables
$\mathbf{x}^*$	optimal solution
x_i	ith design variable
$y(\mathbf{x})$	actual function
$\hat{y}(\mathbf{x})$	surrogate model of the actual function

Chapter 2

$\tilde{f}(\mathbf{x})$	scaled version of $f(\mathbf{x})$
$\tilde{g}(\mathbf{x})$	scaled version of $g(\mathbf{x})$
$\tilde{h}(\mathbf{x})$	scaled version of $h(\mathbf{x})$
n_{obj}	number of objective functions
p	number of points in a DOE
$BestPRESS$	surrogate with lowest $PRESS$ value for a given DOE
DOE	design of experiments
CVE	cross-validation error
$J(\mathbf{x})$	functional that combines objective and constraint functions
KRG	kriging
PRS	polynomial response surface
RBNN	radial basis neural networks
$RMSE$	root mean square error
SVR	support vector regression
$PRESS$	prediction sum of squares

Chapter 3

c_1	PSO self trust parameter
c_2	PSO population trust parameter
n_{pop}	population size
ACO	Ant Colony Optimization
CR	DE crossover probability
DE	Differential Evolution
DM	dispersion measure of the population
ESEA	Enhanced Stochastic Evolutionary Algorithm
F	DE weighting factor
LC	LyfeCycle algorithm
$\mathbf{P}$	population matrix
PSO	Particle Swarm Optimization
γ	ACO dissolving rate
σ^{ACO}	aggregation of the ACO population around the current minimum

Chapter 4

$\hat{\beta}(\mathbf{x})$	ratio correction surrogate
$\hat{\delta}(\mathbf{x})$	difference correction surrogate
$y_{HF}(\mathbf{x})$	high fidelity analysis
$y_{LF}(\mathbf{x})$	low fidelity analysis
N_{mxp}	number of extra actual function evaluations

Chapter 5

$BestRMSE$	most accurate surrogate of the initial DOE (basis of comparison for all other surrogates)
LM	Levenberg-Marquardt algorithm
NMSDS	Nelder-Mead simplex direct search
R_a^2	PRS adjusted correlation coefficients
ϕ_p	criterion which leads to the maximization of the point-to-point distance in a DOE

CHAPTER I

INTRODUCTION

As Venkataraman and Haftka (2004) discussed, the engineering and scientific communities as consumers of computing devices have greatly benefited from the growth in the computer throughput. As a result, the design of engineering systems, ranging from simple products, such as a child's toy, to complex systems, such as an aircraft or automobiles, has become computer centric. This is seen in modern computer-based methodologies of Computer-Aided Design and Computer-Aided Engineering (CAD/CAE), instead of a time consuming process of prototype trial and error. Current software packages allow since three-dimensional drafting to numerical optimization.

However, the truth is that the increases in computer processing power, memory and storage space have alleviated but not eliminated computational cost and time constraints on the use of optimization. In the words of Venkataraman and Haftka (2004): "this is due to the constant increase in the required fidelity (and hence complexity) of analysis models". In short, analysis as well as optimization algorithms have incorporated the hardware advances by increasing the complexity, fidelity, and scope. The bottom line is that the developing in hardware throughput still feeds the advances in both mathematical formulation and numerical implementation of analysis and optimization algorithms, and there is no evidence of changes in years to come. This scenario poses a challenge to optimization, as a discipline: to remain attractive as an engineering tool, while incorporating the technological progress.

This research presents an attempt of combining modern global optimization and surrogate modeling techniques as a way to rationally use the computer budget and increase the information level obtained during the optimization task. The efforts are concentrated in the field of the heuristic optimization methods, where the potential for global optimization, easiness to code, and robustness are encouraging. More precisely, Ant Colony Optimization (ACO), Differential Evolution (DE), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), LifeCycle Optimization (LC), and the Enhanced Stochastic Evolutionary Algorithm (ESEA) form the set of algorithms studied in this thesis. Surrogate modeling are then used to reduce the computational effort intrinsic to these optimization algorithms. Instead of a single and predefined surrogate, this research adopts the use of a large set of different models in order to prevent against badly fitted surrogates. Kriging models (KRG), Polynomial Response

Surfaces (PRS), Radial Basis Neural Networks (RBNN) and Support Vector Regression (SVR) are used in this research. When a large set is used, the selection of a single surrogate follows an estimator of the root mean square error ($RMSE$) based on data points, i.e. the prediction sum of squares ($PRESS$). This surrogate is then called $BestPRESS$.

The main objectives of this work are: (i) studying of the modern optimization techniques; (ii) combining surrogate modeling and heuristic algorithms; (iii) developing general-purpose optimization routines; and (iv) performing applications in direct and inverse problems of engineering. The set of applications, all real world problems, are important to suggest guidelines for the practitioner. These algorithms were employed to solve the following list of continuous/discrete optimization problems:

1. Vehicular three-dimensional structure design optimization.
2. Optimization of aircraft structural components using heuristic algorithms and multi-fidelity approximations.
3. Aircraft longitudinal stability and control derivatives identification by using LifeCycle and Levenberg-Marquardt optimization algorithms.
4. Parameter identification of a non-linear landing gear model.
5. Improvement of surrogate models using heuristic optimization algorithms and a pre-defined number of extra high-fidelity analyses.
6. Optimization of the Latin hypercube design of experiments.

As a result, the contributions of this research can be summarized as:

- The use of non-conventional methods based on heuristic optimization techniques for system identification was explored. Heuristic algorithms presented the required robustness to deal with corrupted experimental data that are typical in this type of application.
- The use of heuristic methods coupled with statistical tools for the solution of design problems was consolidated. This approach presents an alternative to overcome the limitations of classic methods and decrease the computational burden associated with heuristic optimization methods.
- The implementation of both a general-purpose optimization toolbox (SIMPLE Toolbox) and a surrogate modeling toolbox (SURROGATES Toolbox), which are tested in a number of applications.

Next sections introduce the basic concepts on optimization used in this research.

1.1. Fundamental Concepts in Surrogate Modeling

To reduce the computational time of complex engineering simulations, very often surrogate models, also known as meta-models, are employed replacing actual simulation or experimental models. Essentially, surrogate modeling consists in using statistical techniques to build approximations of expensive computer simulation codes. This way, given the vector of variables of interest, $\mathbf{x}$, if the true nature of a model is:

$$y(\mathbf{x}) = F(\mathbf{x}) \ , \tag{1.1}$$

then a surrogate model is:

$$\hat{y}(\mathbf{x}) = G(\mathbf{x}) \ , \tag{1.2}$$

and

$$y(x) = \hat{y}(x) + \mathrm{error}(x) \ , \tag{1.3}$$

where $\mathrm{error}(x)$ represents both the approximation and measurement (random) errors.

Figure 1.1 summarizes the general statistical procedure to generate and use surrogates.

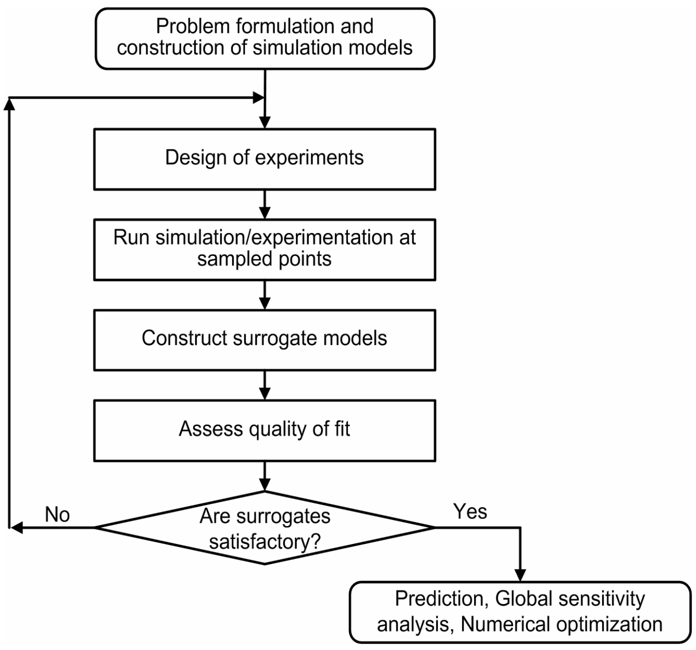

Problem formulation and construction of simulation models
Design of experiments
Run simulation/experimentation at sampled points
Construct surrogate models
Assess quality of fit
Are surrogates satisfactory?
No
Yes
Prediction, Global sensitivity analysis, Numerical optimization

Figure 1.1. Stages of the surrogate-based modeling.

The internal steps of the loop can be detailed as:

1. Design of experiments: the design space is sampled in order to reveal its contents and tendencies (Owen, 1992; Morrsi and Mitchell 1995; and Montgomery, 1997). At this step, the gain of as much information as possible must be balanced with the cost of simulation/experimentation.

2. Run simulation/experimentation at sampled points: at this phase, the simulation (or experimentation) unfeasible regions of the design space can be revealed.

3. Construct surrogate models: the surrogate model is fitted to the collected data (Myers and Montgomery, 1995; Wasserman, 1993; Martin and Simpson, 2005; and Smola and Scholkopf, 2004). It may imply in the solution of a system of linear equations or even in the solution of an optimization problem.

4. Assess quality of fit: the precedent steps are sufficient to build a first tentative model, which overall quality and usefulness has to be evaluated by adequate sets of metrics (Box et al., 1978; Sacks et al., 1989; and Meckesheimer et al., 2002).

Table 1.1 shows several options for previously discussed techniques used in surrogate modeling.

Table 1.1. Techniques for surrogate modeling.

Technique	**Examples**
Design of experiments	Factorial (full and fractional), Central Composite Design, Box-Behnken, D-optimal, Orthogonal Array, Latin Hypercube
Surrogate model	Polynomial Response Surface, Radial Basis Neural Networks, Kriging Models, Support Vector Regression
Verification of model accuracy	Root mean square error, Maximum absolute error, Coefficient of correlation (using test points), Prediction sum of squares, Coefficient of determination (R^2), Estimated prediction variance, Estimated process variance

1.2. Fundamental Concepts in Numerical Optimization

According to Vanderplaats (2005), in Mathematics, optimization is the discipline concerned with finding the maxima and minima of functions, possibly subjected to constraints. Despite its name, optimization does not necessarily mean finding the optimum solution to a problem; instead, it means finding the more suitable solution to a problem (which is the case of compromise solutions and robust solutions). An example of an optimization problem is the following: maximize the profit of a manufacturing operation while ensuring that none of the resources exceed certain limits and also satisfying as much of the demand faced as possible. Thus, extending the previous discussion, the optimization work is a mathematical problem compound by the following elements (Haftka and Gürdal, 1992; Marler and Arora, 2004; and Vanderplaats, 2005):

- Design space: where all possible solutions for a problem are considered (also known as search space). Each element of the design space is called design variable. The vector of design variables, $\mathbf{x}$, is composed by the elements $x_i,\ i = 1,\ 2,\ \ldots,\ n_{dv}$, which means that the number of design variables is n_{dv}. The bounds of the design space are given by lower and upper limits for each design variable, $x_i^l \leq x_i \leq x_i^u,\ i = 1,\ 2,\ \ldots,\ n_{dv}$. Design variables can be continuous (i.e., real values within a range), or discrete (i.e., certain values defined in a list of permissible values). When optimization is solved numerically, scaling design variables is important to avoid ill conditioning problems. This process consists in mapping the boundaries of each design variable to a new design space with boundaries $[0 \quad 1]$ or $[-1 \quad 1]$.

- Objective function: also known as cost function, fitness function or evaluating function. This is the way to evaluate each point of the design space. The objective function is denoted as $f(\mathbf{x})$ for single objective problems and $\mathbf{f}(\mathbf{x})$ for multi-objective problems.

- Constraints: impose restrictions to the system. A design point which satisfies all the constraints is called feasible, while a design point which violates even a single constraint is called infeasible. The collection of all feasible points is called feasible domain, or occasionally the constraint set. Figure 1.2 shows modifications in the original design space that are caused by constraints.

- Optimization algorithms: the algorithm that will search for the solution of the optimization problem.

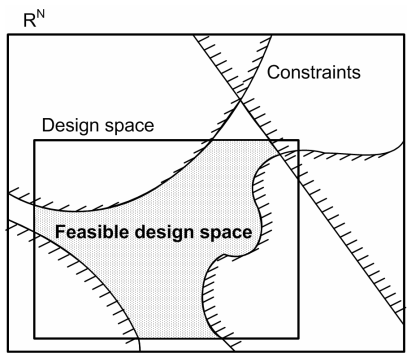


Figure 1.2. Design space definition.

In the single objective case, numerical optimization solves the following nonlinear, constrained problem: find the point, $\mathbf{x}^*$, in the design space that will minimize the objective function for a given set of system parameters, possibly observing a set of constraints. In other words, the standard formulation of an optimization problem is:

$$minimize \quad f(\mathbf{x}), \tag{1.4}$$

subject to:

$$\begin{cases} x_i^l \leq x_i \leq x_i^u, & i = 1,\ 2,\ \ldots,\ n_{dv}\ , \\ g_j(\mathbf{x}) \leq 0, & j = 1,\ 2,\ \ldots,\ n_{ineqcnstrt}\ , \\ h_j(\mathbf{x}) = 0, & j = n_{ineqcnstrt} + 1,\ n_{ineqcnstrt} + 2,\ \ldots,\ n_{cnstrt}\ , \end{cases} \tag{1.5}$$

where:

- $f(\mathbf{x})$ is the objective function,
- $x_i^l \leq x_i \leq x_i^u$ imposes the side constraints to the design space,
- n_{dv} is the number of design variables,
- $n_{ineqcnstrt}$ is the number of inequality constraints,
- n_{cnstrt} is the total number of constraints (inequality and equality), and
- $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$ are the inequality and equality constraints, respectively.

About the optimization algorithms, they can be classified according to the level of information required to solve the problem. This way, as it can be found in Vanderplaats (2005):

- Zero-order methods: use direct evaluation of the functions (also called direct-search methods). These methods present advantages such as easiness to program, capability of dealing with non-convex and discontinuous functions, and in many cases capability of working with discrete design variables. The price paid for this generality is that these methods often require a large number of function evaluations.

- First-order methods: require the evaluation of the gradient of the functions. Since more information is used during the search, these methods are expected to be more efficient in terms of the number of function evaluations as compared with zero-order methods. However, the price paid is that gradient information must be supplied. In addition, they have difficulties to deal with local minima and discontinuity on the first derivatives.

- Second-order methods: use function values, gradient and the Hessian matrix (the square matrix of second order partial derivatives of a function). As they increase the amount of used information, they are expected to be more efficient. However, while it does not solve the difficulties with local minima, this fact also implies in the additional hardness of the Hessian computation.

Other than the level of information, the approach used by the algorithm to manipulate this information allows a second classification. In this sense, classical methods are based on Algebra and Differential Calculus. Alternatively, in heuristic methods, the most appropriate solutions of several found are selected at successive stages for use in the next step of the search. They are called heuristic methods since the selection and update of solutions follow a heuristic rather than a formal mathematical approach (Rasmussen, 2002; and Haupt and Haupt, 2004). Figure 1.3 shows a concise representation of both classifications for the optimization algorithms.

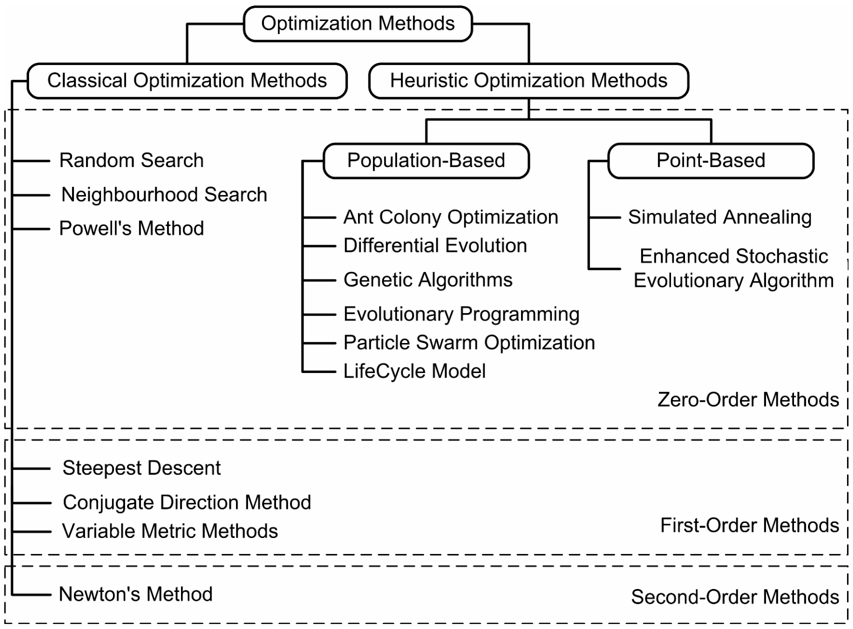


Figure 1.3. Classification of optimization algorithms.

1.3. Classification of the Optimization Problems

In terms of classification, in this research work, the optimization problems with continuous and eventually discrete-continuous variables are divided in two groups, namely direct problems and inverse problems. The distinction between them depends on the causes and effects relationship of the system. Figure 1.4 shows a scheme that illustrates the concepts in discussion. This way, in direct problems the inputs are used to determine either the best configuration of the given system or the response of the system to a pre-defined set of inputs. As an example of direct problems can be cited those in rigid body dynamics (in particular, articulated rigid body dynamics) that often require mathematical programming techniques, since rigid body dynamics attempts to solve an ordinary differential equation subjected to a set of constraints, which are various nonlinear geometric constraints such as “these two points must always coincide,” “this surface must not overlap to any other,” or “this point must always lie somewhere on this curve.” On the other hand, the inverse problem consists in using the results of actual observations to infer about either the values of the parameters characterizing the system under investigation or the inputs that generate a known set of outputs. As an example of inverse problems can be cited the identification of the proper boundary conditions and/or initial conditions such as: a) determination of thermal, stress/strain, electromagnetic, fluid flow boundary conditions on inaccessible boundaries, and b) determination of initial position, velocity, acceleration or chemical composition.

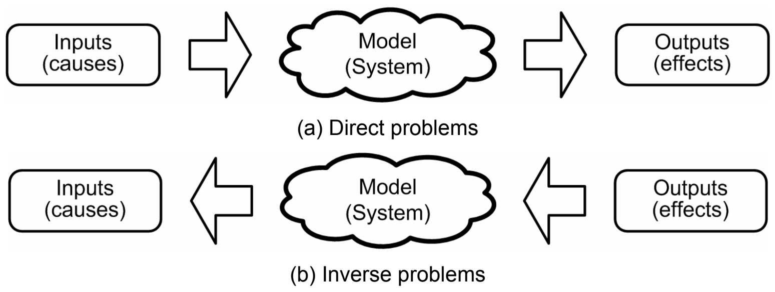


Figure 1.4. Classification of problems according to the causes and effects context

In addition, some difficulties that are intrinsic to inverse problems may arise:

- an accurate model of the system is required, since the results of the identification procedure rely upon the model used;
- identification problems are sensitive to noise found in the experimental data; and

- experimental data are incomplete either in the spatial sense (responses are available only in a limited number of positions along the structure), as in the time sense (responses are obtained in a given time interval and sampling frequency).

Alternatively, it is also possible that instead of finding the optimal set of values for different design variables, the optimization problem consists in finding the optimal combination of elements in a vector. As in the literature (Nemhauser and Wolsey, 1988; Papadimitriou and Steiglitz, 1998), here, these problems are classified as combinatorial optimization problems. In this category, instead of varying the values of each design variable, the optimizer changes the position of the design variables within the vector of design variables. The vehicle routing problem is an example of combinatorial optimization problem. Often the context is that of delivering goods located at a central depot to customers who have placed orders for such goods. The goal is minimizing the cost of distributing the goods, while serving the customers with a fleet of vehicles. Another example is the knapsack problem. It consists in the maximization problem of the best choice of essentials that can fit into one bag to be carried on a trip. Given a set of items, each with a cost and a value, the goal is to determine the number of each item to include in a collection so that the total cost is less than a given limit and the total value is as large as possible.

1.4. Literature Review

There is a vast literature about the basis of surrogate modeling and the application in benchmark and real world problems. Wang and Shan (2007) reviewed the state-of-the-art surrogate-based techniques from a practitioner's perspective according to the role of surrogate modeling in supporting design optimization, including model approximation, design space exploration, problem formulation, and solving various types of optimization problems. Simpson et al. (2001) surveyed the theoretical and practical aspects of several meta-modeling techniques, including Design of Experiments, Response Surface Methodology, Neural Networks, Inductive Learning and Kriging. Simpson et al. (2004) presented a discussion about concepts, state-of-the-art, applications and future directions of approximation methods in multidisciplinary analysis and optimization. Queipo et al. (2005) provided a discussion of the fundamental issues in surrogate-based analysis and optimization, including loss function and regularization criteria for constructing the surrogates, design of experiments, surrogate selection and construction, sensitivity analysis, convergence, and optimization. Zerpa et al. (2005) used multiple surrogates for optimization of an alkaline–surfactant–polymer flooding processes incorporating a local weighted average model of the individual surrogates. Goel et al. (2007) explored different approaches in which the weights associated with each surrogate model are determined based on the global cross-validation error measure called prediction sum of squares.

The literature about heuristic optimization algorithms is vast as well. Coello (2005) provided a brief introduction to this class of algorithm including some of their applications and current research directions. Van Veldhuizen and Lamont (2000) and Marler and Arora (2004) presented a survey of current continuous non-linear multi-objective optimization concepts and methods. Michalewicz (1995) and Coello (2002) reviewed methods for handling constraints by heuristic methods and tested them on selected benchmark problems; and discussed their strengths and weaknesses. About the algorithms themselves, especially those embraced in this research, Ant Colony Optimization is presented in details in Dorigo et al. (1996) and Pourtakdoust and Nobahari (2004); Differential Evolution is comprehensively discussed in Storn and Price (1997) and Kaelo and Ali (2006); Genetic Algorithms are well explained in Haupt and Haupt (2004) and Michalewicz and Fogel (2000); Particle Swarm Optimization is described in Kennedy and Eberhart (1995) and Venter and Sobieszczanski-Sobieski (2003); and finally, LifeCycle Optimization is introduced in Krink and Løvberg (2002) and recently applied to a real-word problem in Flores et al. (2007). Enhanced Stochastic Evolutionary Algorithm was first introduced by Saab and Rao (1991) and later applied to the optimization of the Latin hypercube design in Jin et al. (2005) and Viana et al. (2007c).

1.5. Scope of Current Research

In short, the goal of the present work is to develop methodologies for applying heuristic optimization techniques to the solution of optimization problems in Engineering. The research, conducted in the context of a doctoral thesis, has the following main objectives:

- Study of the modern optimization techniques, embracing since the problem definition to the solution of the optimization problem.
- Combining surrogate modeling and heuristic algorithms to the solution of optimization problems.
- Developing general-purpose optimization routines for academic use enabling its application to both direct and inverse problems.
- Using the above mentioned aspects in the solution of engineering problems. The applications studied must contain both direct and inverse problems.

The organization of this work is as follows. Chapter II gives more details about surrogate modeling and numerical optimization; it includes an overview about design of experiments as well as the basic surrogate models and the ensemble of surrogates as well as a revision of the formulation of the optimization problem, including constrained and multi-objective techniques as they are used by heuristic algorithms. Chapter III discusses the optimization algorithms used in this work as well as it presents some of the contributions of the present research on the implementation of such methods. Chapter IV discusses the different proposed approaches on how to couple surrogate modeling and heuristic algorithms to optimization. Chapter V presents the applications studied during this doctoral research. Finally, Chapter VI highlights the major conclusions of the present research work and delineates the scope of future work on this topic.

CHAPTER II

SURROGATE MODELING AND NUMERICAL OPTIMIZATION

2.1. Introduction

Chapter I has already introduced the basic concepts on surrogate modeling and numerical optimization. This chapter discusses how the current doctoral research approached these techniques. This is done by presenting:

1. The interaction between the general user of numerical optimization and the set of components involved in the solution of an optimization problem.
2. The proper formulation of optimization algorithms and surrogate modeling techniques addressed by this research.

Figure 2.1 shows a possible scenario on how the user approaches the problem to be solved (at this point there is no distinction between direct, inverse or combinatorial problems). It is worth mentioning that among the different activities illustrated in Figure 2.1, the conception and decision phases ("Concept problem" and "Take decision", respectively) are not covered by the present study. For the remaining three, the set of different exemplified software makes it easy to see the insertion in the Computer-Aided Engineering, where the computational tools are used together with real world data. In a real world context, these activities may not appear in cascade or there may be others to be considered. System Engineering (Kossiakoff and Sweet, 2002) techniques are usually employed to guide the solution of the problem, by maximizing the acquired amount of information.

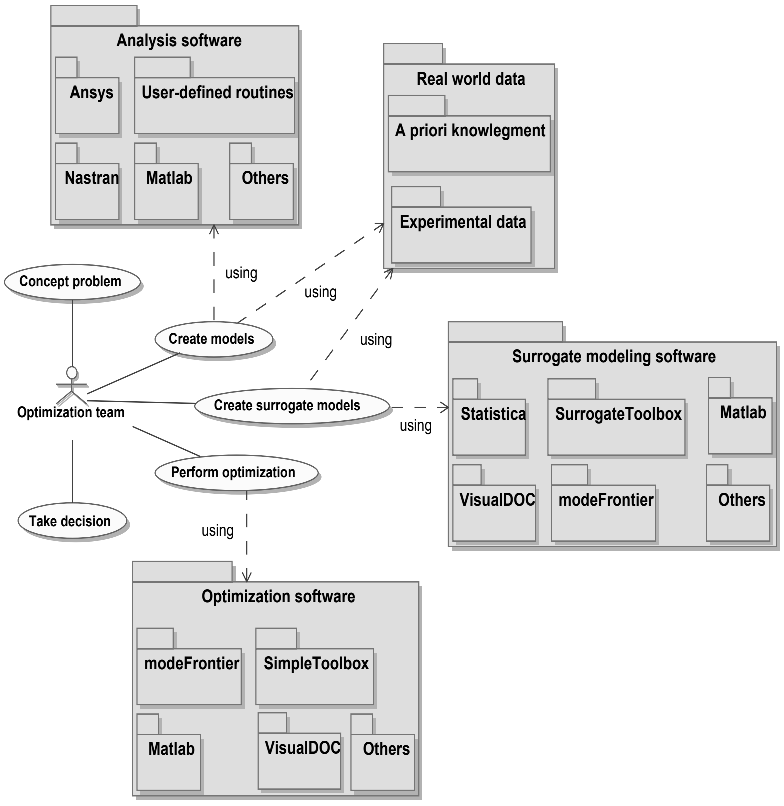
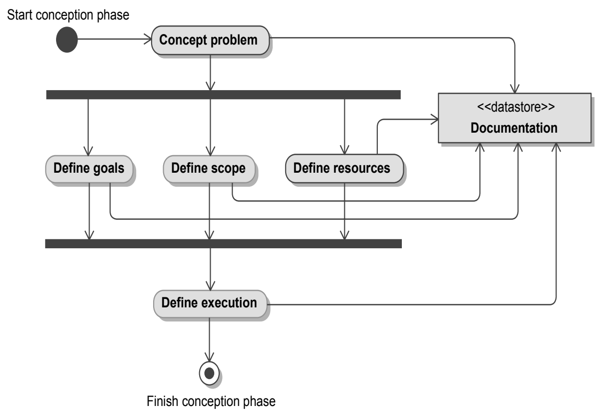


Figure 2.1. Optimization environment.

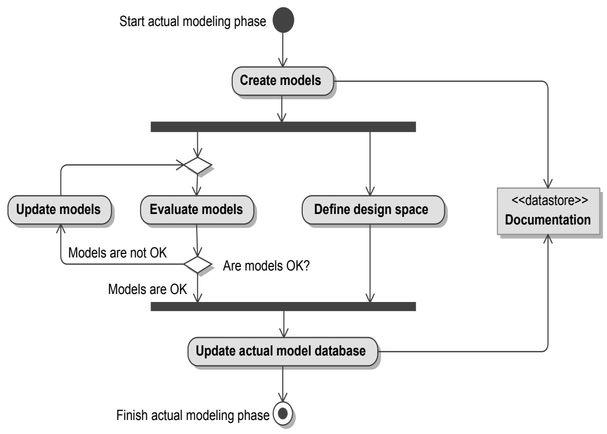
More details are given below:

- Concept problem: this is the process through which goals, scope and resources are defined. The focus is on the needs and required functionality, documenting and then proceeding to design synthesis and system validation while considering the complete problem. This phase often involves contribution from diverse technical disciplines (not only Engineering but also resource management). Figure 2.2-(a) illustrates a possible flowchart for this phase.

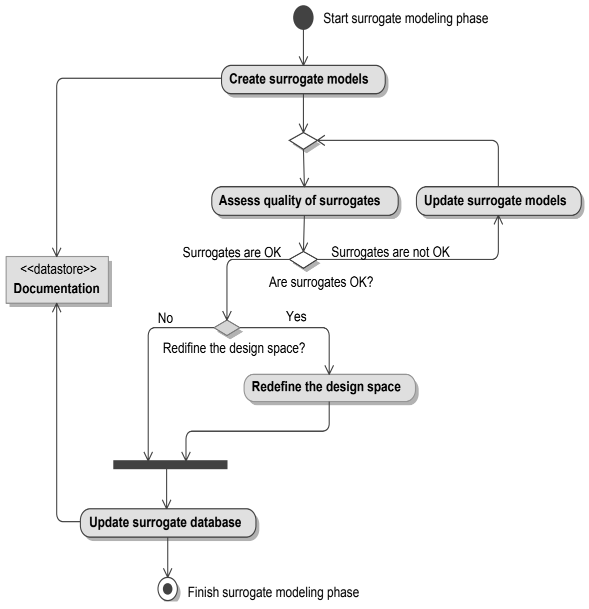
- Create models: this is the phase in which high-fidelity (possibly computationally or experimentally expensive) models are created. “Models” are intended to represent the physical reality of the systems under investigation. The level of accuracy in representing the reality defines the level of fidelity of the model. In the present work, to avoid misunderstandings, the expression “actual model” is sometimes used to define the high-fidelity models. Finally, the expression “expensive models” stands for difficulties in the model evaluation related to time or resource consumptions. Figure 2.2-(b) shows a possible cycle for this phase.



(a) Conception phase.



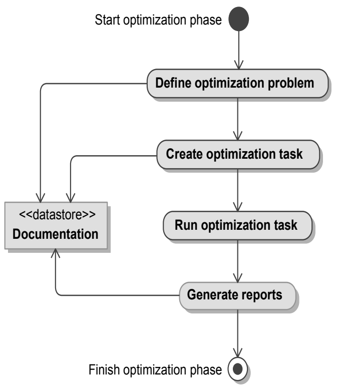
(b) Modeling phase.

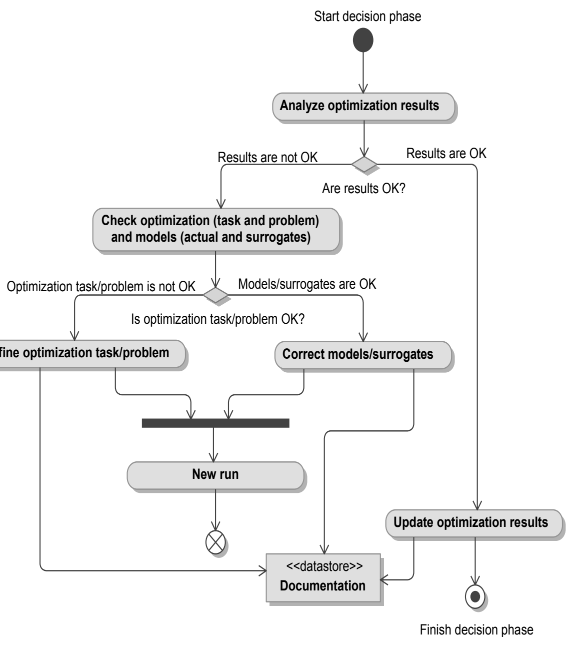


Start surrogate modeling phase
Create surrogate models
Assess quality of surrogates
Update surrogate models
Surrogates are OK
Surrogates are not OK
Are surrogates OK?
<<datastore>>
Documentation
No
Yes
Redifine the design space?
Redefine the design space
Update surrogate database
Finish surrogate modeling phase

(c) Surrogate modeling phase.

(d) Optimization phase.





Start decision phase
Analyze optimization results
Results are not OK
Results are OK
Are results OK?
Check optimization (task and problem) and models (actual and surrogates)
Optimization task/problem is not OK
Models/surrogates are OK
Is optimization task/problem OK?
Redifine optimization task/problem
Correct models/surrogates
New run
Update optimization results
<<datastore>>
Documentation
Finish decision phase

(e) Decision phase.

Figure 2.2. Activity diagram of different phases of the Engineering problem solution.

- Create surrogate models: when the actual models are extremely expensive to be used, surrogate models are constructed. This phase is based on statistical procedures to replace the actual models. It frequently encompasses resources for building and evaluating the fidelity level of the resulting surrogate models. Figure 2.2-(c) exemplifies the sequence of activities in this phase.

- Perform optimization: in this phase the optimization problem is defined and solved. Figure 2.2-(d) suggests a possible scheme for this phase.

- Take decision: this procedure is supported by all previous ones. In a more general sense, practical considerations (such as investment, cost-benefit ratio, marketing trends) are combined with the results of the modeling and optimization phases. Here, the focus is given to the optimization process instead, as illustrated in Figure 2.2-(e).

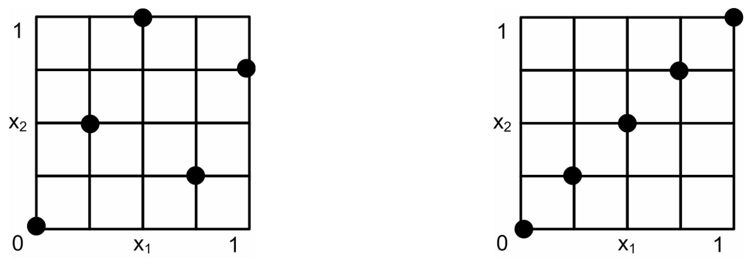
Despite the beauty of each of the phases, this work addresses the aspects of the use of heuristic optimization techniques on the solution of engineering problems. It means that during the conception phase, the optimization team already decided to use this class of optimization algorithms.

2.2. Surrogate Modeling Framework

2.2.1. Design of Experiments (DOE) and Latin Hypercube Sampling

The location of experimental/simulation data points is very important for generating accurate surrogate models, while maintaining a reasonable number of data points. Design of Experiments (Myers and Montgomery, 1995; and Montgomery 1997) aids in the process of point selection, extracting as much information as possible from a limited number of design points. There are many different criteria available for creating a design of experiments. The criterion of space filling design, which aims to cover as much of the design space, seems to be the most suitable for the use with heuristic optimization algorithms. Most of the time in this research, the space filling design called Latin Hypercube design of experiment was used. This sampling scheme, proposed by McKay et al. (1979) and Iman and Conover (1980), presents several advantages, such as: (i) number of points (samples) is not fixed; (ii) orthogonality of the sampling points (different points do not have the same projection in any dimension); (iii) sampling points do not depend on the surrogate model that will be constructed; and (iv) different configurations can be constructed for the same number of variables and number of sampling points.

The Latin Hypercube design is constructed in such a way that each one of the n_{dv} dimensions is divided into p equal levels and that there is only one point for each level (Bates et al., 2004). The final Latin Hypercube design then has p samples. Figure 2.3 shows two possible Latin Hypercube designs for $n_{dv} = 2$ and $p = 5$. Note that the Latin Hypercube design is constructed using a random procedure. This process results in many possible designs, each being equally good in terms of the Latin Hypercube conditions. However, a design that is ill suited for creating a surrogate model is possible, even if all the Latin Hypercube requirements are satisfied, as illustrated in Figure 2.3.



(a) Latin Hypercube 01, (b) Latin Hypercube 02.

Figure 2.3. Latin Hypercube DOEs for $n_{dv} = 2$ and $p = 5$.

To overcome the above mentioned problem, the Optimal Latin Hypercube design was introduced to improve the space-filling property of the Latin Hypercube design. The Optimal Latin Hypercube design augments the Latin Hypercube design by requiring that the sample points be distributed as uniformly as possible throughout the design space. Unfortunately, the Optimal Latin Hypercube design results in a hard and time consuming optimization problem. For example, to optimize the location of 10 samples in 4 dimensions, the optimizer has to select the best design from more than 1022 possible designs. If the number of design variables is increased to 5, the number of possible designs is more than 6×10^{32}. To solve the Optimal Latin Hypercube design, it is necessary to formulate an optimization problem, the solution of which is the best design. To have an idea about how difficult this task can be, Ye et al. (2000) reported that generating an Optimal Latin Hypercube design with 25 samples in 4 dimensions using a column-wise/pair-wise algorithm could take several hours on a Sun SPARC 20 workstation. The Optimal Latin Hypercube was a case study of combinatorial optimization during this doctoral research. The achieved advances were reported in Viana et al. (2007a), where the generation of an Optimal Latin Hypercube design with 256 samples in 4 dimensions was as fast as 5 minutes in a PC with a 1000 MHz Pentium III Zeon processor. Chapter V gives more details about this implementation.

2.2.2. Surrogate Modeling Techniques

Viana et al. (2008a) have already shown that, for prediction purposes, it pays to generate a large set of surrogates and then pick the best of the set according to an estimator of the root mean square error called $PRESS$ (prediction sum of squares). The main features of the four surrogate models used in this study are described in the following sections. For all discussion, consider that the models are fitted with the set of p samples.

1. Kriging (KRG)

Kriging is named after the pioneering work of the South African mining engineer D.G. Krige. It estimates the value of a function as a combination of known functions $f_i(\mathbf{x})$ (e. g., a linear model such as a polynomial trend) and departures (representing low and high frequency variation components, respectively) of the form:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{p} \beta_i f_i(\mathbf{x}) + Z(\mathbf{x}) \ , \tag{2.1}$$

where $Z(\mathbf{x})$ is assumed to be a realization of a stochastic process with zero mean, process variance σ^2, and spatial covariance function given by:

$$\operatorname{cov}\left(Z(\mathbf{x}_i), Z(\mathbf{x}_j)\right) = \sigma^2 R\left(\mathbf{x}_i, \mathbf{x}_j\right) \ , \tag{2.2}$$

where $R\left(\mathbf{x}_i, \mathbf{x}_j\right)$ is the correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$.

The conventional KRG models interpolate training data. This is an important characteristic when dealing with noisy data. In addition, KRG is a flexible technique since different instances can be created, for example, by choosing different pairs of $f_i(\mathbf{x})$ and correlation functions. The complexity of the method and the lack of commercial software may hinder this technique from being popular in the near term (Simpson et al., 1997).

The Matlab code developed by Lophaven et. al (2002) was used to execute the KRG algorithm. More details about KRG are provided in Sacks et al. (1989), Simpson et al. (1997), and Martin and Simpson (2005).

2. Polynomial Response Surface (PRS)

The PRS approximation is one of the most well established meta-modeling techniques. In PRS modeling, a polynomial function is used to approximate the actual function. A second-order polynomial model can be expressed as:

$$\hat{y}(\mathbf{x}) = \beta_0 + \sum_{i=1}^{m} \beta_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} \beta_{ij} x_i x_j \ , \tag{2.3}$$

The set of coefficients β can be obtained by least squares and according to the PRS theory are unbiased and have minimum variance. Another characteristic is that it is possible to identify the significance of different design factors directly from the coefficients in the normalized regression model (in practice, using t-statistics). In spite of the advantages, there is a drawback when applying PRS to model highly nonlinear functions. Even though higher-order polynomials can be used, it may be too difficult to take sufficient sample data to estimate all of the coefficients in the polynomial equation, particularly in large dimensions.

The SURROGATES toolbox of Viana and Goel (2008) was used for PRS modeling. See Box et al. (1978) and Myers and Montgomery (1995) for more details about PRS.

3. Radial Basis Neural Networks (RBNN)

RBNN is an artificial neural network which uses radial basis functions as transfer functions. RBNN consist of two layers: a hidden radial basis layer and an output linear layer, as shown in Figure 2.4.

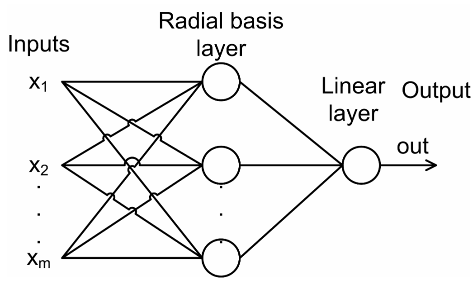


Figure 2.4. Radial basis neural network architecture.

The output of the network is thus:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{N} a_i \rho(\mathbf{x}, \mathbf{c}_i) \ , \tag{2.4}$$

where N is the number of neurons in the hidden layer, $\mathbf{c}_i$ is the center vector for neuron i, and a_i are the weights of the linear output neuron. The norm is typically taken to be the Euclidean distance and the basis function is taken to be the following:

$$\rho(\mathbf{x}, \mathbf{c}_i) = \exp\left(-\beta \|\mathbf{x} - \mathbf{c}_i\|^2\right) \ , \tag{2.5}$$

where $\exp(\cdot)$ is the exponential function.

RBNN may require more neurons than standard feed-forward/back-propagation networks, but often they can be designed in a fraction of the time it takes to train standard feed-forward networks. They work best when many training points are available, and this can be a crucial drawback in some applications. Another important point to consider is that the training process may lead to totally different configurations and thus different models.

The native neural networks Matlab® toolbox (MathWorks Contributors, 2002) was used to execute the RBNN algorithm. RBNN is comprehensively presented in Smith (1993), Wasserman (1993), and Cheng and Titterington (1994).

4. Support Vector Regression (SVR)

SVR is a particular implementation of support vector machines (SVM). In its present form, SVM was developed at AT&T Bell Laboratories by Vapnik and co-workers in the early 1990s (Vapnik, 1995). In SVR, the aim is to find $\hat{y}(\mathbf{x})$ that has at most ε deviations from each of the targets of the training inputs. Mathematically, the SVR model is given by:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{p}(a_i - a_i^*)K(\mathbf{x}_i,\mathbf{x}) + b \tag{2.6}$$

where $K(\mathbf{x}_i,\mathbf{x})$ is the so-called kernel function, $\mathbf{x}_i$ are different points of the original DOE and $\mathbf{x}$ is the point of the design space in which the surrogate is evaluated. Parameters a_i, a_i^*, and b are obtained during the fitting process.
Table 2.1 lists the kernel functions used in this work.

During the fitting process, SVR minimizes an upper bound of the expected risk unlike empirical risk minimization techniques, which minimize the error in the training data (which defines a_i, a_i^*, and b in Eq. (2.6)). This is done by using alternative loss functions. Figure 2.5 shows two of the most common possible loss functions. Figure 2.5-(a) corresponds to the conventional least squares error criterion. Figure 2.5-(b) illustrates the loss function used in this work, which is given by:

$$Loss(\mathbf{x}) = \begin{cases} \varepsilon, \text{ if } |y(\mathbf{x}) - \hat{y}(\mathbf{x})| \le \varepsilon \\ |y(\mathbf{x}) - \hat{y}(\mathbf{x})|, \text{ otherwise} \end{cases} \tag{2.7}$$

Table 2.1. Kernel functions (consider $\exp(\cdot)$ as the exponential function).

Gaussian Radial Basis Function (GRBF)	$K(\mathbf{x}_1,\mathbf{x}_2) = \exp\left(-\frac{\Vert\mathbf{x}_1-\mathbf{x}_2\Vert^2}{2\sigma^2}\right)$
Exponential Radial Basis Function (ERBF)	$K(\mathbf{x}_1,\mathbf{x}_2) = \exp\left(-\frac{\Vert\mathbf{x}_1-\mathbf{x}_2\Vert}{2\sigma^2}\right)$
Splines	$K(\mathbf{x}_1,\mathbf{x}_2) = 1 + \langle\mathbf{x}_1,\mathbf{x}_2\rangle +$ $\frac{1}{2}\langle\mathbf{x}_1,\mathbf{x}_2\rangle\min(\mathbf{x}_1,\mathbf{x}_2) - \frac{1}{6}(\min(\mathbf{x}_1,\mathbf{x}_2))^3$
Anova-Spline (Anova)	$K(\mathbf{x}_1,\mathbf{x}_2) = \prod_i K_i(x_{1i},x_{2i})$ $K_i(x_{1i},x_{2i}) = 1 + x_{1i}x_{2i} + (x_{1i}x_{2i})^2 +$ $(x_{1i}x_{2i})^2\min(x_{1i},x_{2i}) - x_{1i}x_{2i}(x_{1i}+x_{2i})(\min(x_{1i},x_{2i}))^2 +$ $\frac{1}{3}(x_{1i}{}^2 + 4x_{1i}x_{2i} + x_{1i}{}^2)(\min(x_{1i},x_{2i}))^3 +$ $-\frac{1}{2}(x_{1i}+x_{2i})(\min(x_{1i},x_{2i}))^4 +$ $\frac{1}{5}(\min(x_{1i},x_{2i}))^5$

The implication is that in SVR the goal is to find a function that has at most ε deviation from the training data. In other words, the errors are considered zero as long as they are inferior to ε.

According to Smola and Scholkopf (2004): “after that the algorithmic development seems to have found a more stable stage, one of the most important ones seems to be to find tight error bounds derived from the specific properties of kernel functions.” Another open issue in SVR is the choice of the values of parameters for both kernel and loss functions.

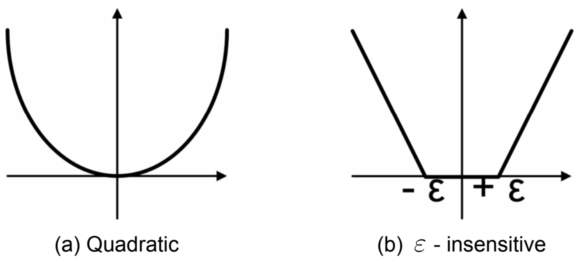


Figure 2.5. Loss functions.

The Matlab code developed by Gunn (1998) was used to execute the SVR algorithm. To learn more about SVR see Vapnik (1995), Smola and Scholkopf (2004), and Clarke et al. (2005).

5. Using a Large Set of Surrogates

Before start the discussion about the use of an ensemble of surrogates, it is important to clarify that when a set of surrogates is generated a common measure of quality must be employed in order to rank the surrogates. Since the set may be constituted by surrogates based on different statistical assumptions, this measure must be model-independent. One commonly used measure is the root mean square error ($RMSE$), which in the design domain with volume V is given by:

$$RMSE = \sqrt{\frac{1}{V}\int_V [y(\mathbf{x}) - \hat{y}(\mathbf{x})]^2 \, d\mathbf{x}} \ , \tag{2.8}$$

$RMSE$ is computed by Monte-Carlo integration at a large number of p_{test} test points:

$$RMSE = \sqrt{\frac{1}{p_{test}} \sum_{i=1}^{p_{test}} (y_i - \hat{y}_i)^2} \tag{2.9}$$

However, the use of an extra large set of test points is frequently prohibitive in most of the real world applications. This way, $RMSE$ is estimated by using cross-validation errors. A cross-validation error is the error at a data point when the surrogate is fitted to a subset of the data points not including that point. When the surrogate is fitted to all the other $p-1$ points, (so-called leave-one-out strategy), it is obtained the vector of cross-validation errors, $\tilde{\mathbf{e}}$. This vector is also known as the $PRESS$ vector ($PRESS$ stands for prediction sum of squares). Figure 2.6 illustrates the cross-validation errors at the third point of the DOE by fitting a PRS and a KRG model to the remaining four of the five data points of the function sin(x).

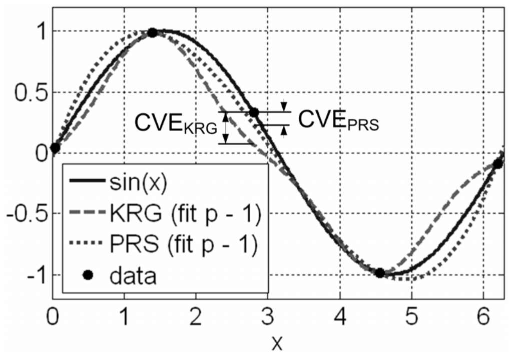


Figure 2.6. Cross-validation errors (CVE) for a polynomial response surface and Kriging surrogates.

However, the leave-one-out strategy is computationally expensive for large number of points. We then use a variation of the k-fold strategy (Kohavi, 1995). According to the classical k -fold strategy, after dividing the available data (p points) into p/k clusters, each fold is constructed using a point randomly selected (without replacement) from each of the clusters. Of the k folds, a single fold is retained as the validation data for testing the model, and the remaining $k-1$ folds are used as training data. The cross-validation process is then repeated k times with each of the k folds used exactly once as validation data. Note that k-fold turns out to be the leave-one-out when $k = p$.

The $RMSE$ is estimated from the $PRESS$ vector:

$$PRESS = \sum_{i=1}^{p} \tilde{e}_i^2 \ , \quad PRESS_{RMS} = \sqrt{\frac{1}{p} PRESS} \ , \tag{2.10}$$

where $\tilde{e}_i$ is the cross-validation error obtained at the *i-th* point of the DOE.

As shown in Viana et al. (2008b), since $PRESS_{RMS}$ is a good estimator of the $RMSE$, one possible way of using multiple surrogates is to select the model with best $PRESS$ value (called in the literature as $BestPRESS$ surrogate). Because the quality of fit depends on the data points the $BestPRESS$ surrogate may vary from DOE to DOE. This strategy may include surrogates based on the same methodology, such as different instances of Kriging (e.g., Kriging models with different regression and/or correlation functions). The main benefit from a diverse and large set is the increasing chance of avoiding (i) poorly fitted surrogates and (ii) DOE dependence of the performance of individual surrogates. Obviously, the key for the success when using $BestPRESS$ is the quality of the $PRESS$ estimator.

The SURROGATES toolbox of Viana and Goel (2008) is used for an easy manipulation of all different codes previously presented.

2.3. General Optimization Problem

The basic version of the heuristic optimization algorithms is defined for unconstrained problems with a single objective. Since most engineering problems are multi-objective and/or constrained problems in one way or another, it is important to add the capability of dealing with constraint functions in a multi-criteria context. For the sake of clarity, the optimization problem is restated in its general form as:

$$minimize \quad \mathbf{f}(\mathbf{x}), \tag{2.11}$$

subject to:

$$\begin{cases} x_i^l \le x_i \le x_i^u, & i = 1,\ 2,\ \ldots,\ n_{dv}\ , \\ g_j(\mathbf{x}) \le 0, & j = 1,\ 2,\ \ldots,\ n_{ineqcnstrt}\ , \\ h_j(\mathbf{x}) = 0, & j = n_{ineqcnstrt}+1,\ n_{ineqcnstrt}+2,\ \ldots,\ n_{cnstrt}\ , \end{cases} \tag{2.12}$$

where:

- $\mathbf{f}(\mathbf{x}) = \left[f_1(\mathbf{x}) \quad f_2(\mathbf{x}) \quad \ldots \quad f_{n_{obj}}(\mathbf{x}) \right]^T$ is the vector of objective functions. This vector is composed by objective functions that can sometimes support but more commonly conflict with each other.
- $x_i^l \le x_i \le x_i^u$ imposes the side constraints to the design space,
- n_{dv} is the number of design variables,
- $n_{ineqcnstrt}$ is the number of inequality constraints,
- n_{cnstrt} is the total number of constraints (inequality and equality), and
- $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$ are the inequality and equality constraints, respectively.

Figure 2.7 gives a graphical representation of the general optimization problem in terms of design and function space.

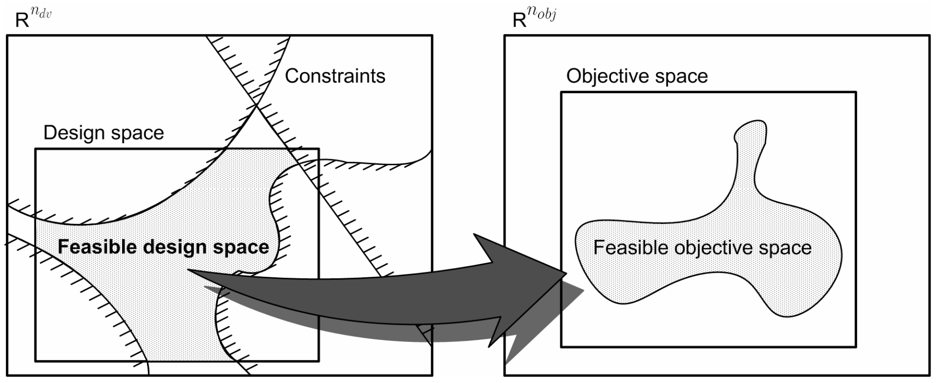


Figure 2.7. General optimization problem.

The addition of multiple objectives asks for the introduction of new concepts. While the solution of a single objective problem is a single design point, the solution of a multi-objective problem is a subspace of designs. The subspace is characterized by the condition that no objective function can be improved without some deterioration in another objective function.

This is known as “Pareto optimally” (Haftka and Gürdal, 1992; Marler and Arora, 2004). Formally, the Pareto optimal can be defined as follows (Marler and Arora, 2004):

- A point, $\mathbf{x}^* \in FeasibleDesignSpace$, is a Pareto optimal if there does not exist another point, $\mathbf{x} \in FeasibleDesignSpace$, such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1,\ 2,\ldots,\ n_{obj}$, and $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for at least one objective function.

The set of Pareto optimal points creates what is typically referred to as the Pareto front. Another important concept in multi-objective optimization is the concept of utopia point (Marler and Arora, 2004):

- A point, $\mathbf{f}^\circ(\mathbf{x}) \in FeasibleObjectiveSpace$, is an utopia point (or ideal point) if for each $i = 1,\ 2,\ldots,\ n_{obj}$, $f_i^\circ(\mathbf{x}) = \min(f_i(\mathbf{x}))$.

In general, the utopia point is unattainable, and the best solution is as close as possible to the utopia point. Such a solution is called a compromise solution and is Pareto optimal. Figure 2.8 illustrates these concepts by considering the example of two conflicting objectives. The shaded area defines non-optimal solutions.

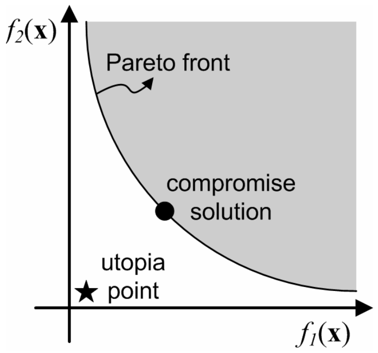


Figure 2.8. Pareto front.

As in the case of design variables, performing a mapping of the functions to a common range may be convenient. By doing this, they will not be erroneously interpreted when combined together by the handling techniques. There are different methods of scalarization, depending on whether a function is an objective or a constraint.

Considering the objective functions, the first approach is as simple as:

$$\tilde{f}_i(\mathbf{x}) = \frac{f_i(\mathbf{x})}{\left|f_i^{\max}(\mathbf{x})\right|}\ , \tag{2.13}$$

where $\tilde{f}_i(\mathbf{x})$ is the scaled version of $f_i(\mathbf{x})$ and $f_i^{\max}(\mathbf{x})$ is the maximum value possible for $f_i(\mathbf{x})$. Note that $f_i^{\max}(\mathbf{x}) \neq 0$ is assumed. The result, $\tilde{f}_i(\mathbf{x})$, is a non-dimensional objective function with an upper limit of one (or negative one) and an unbounded lower limit.

An alternative to Eq. (2.13) is:

$$\tilde{f}_i(\mathbf{x}) = \frac{f_i(\mathbf{x}) - f_i^{\circ}(\mathbf{x})}{f_i^{\circ}(\mathbf{x})} \quad . \tag{2.14}$$

In this case, the lower value of $\tilde{f}_i(\mathbf{x})$ is restricted to zero, while the upper value is unbounded.

However, a most robust approach is given as follows:

$$\tilde{f}_i(\mathbf{x}) = \frac{f_i(\mathbf{x}) - f_i^{\circ}(\mathbf{x})}{f_i^{\max}(\mathbf{x}) - f_i^{\circ}(\mathbf{x})} \quad . \tag{2.15}$$

In this case, the values assumed by $\tilde{f}_i(\mathbf{x})$ are between zero and one.

It is easy to see that the mapping as suggested by Eqs. (2.13) to (2.15), relies on $f_i^{\max}(\mathbf{x})$ and $f_i^{\circ}(\mathbf{x})$. Unfortunately, this information is not available. Indeed, there might be no reason for optimization if $f_i^{\circ}(\mathbf{x})$ was known. In practice, these values are either estimated or assumed as based on any a priori knowledgement regarding the problem. Vanderplaats (2005) suggests $f_i^{worst}(\mathbf{x})$ as the objective function associated with the initial design (also called baseline design) to replace $f_i^{\max}(\mathbf{x})$; and $f_i^{*}(\mathbf{x})$ as the desired value of this objective function to be used in the place of $f_i^{\circ}(\mathbf{x})$. The counterparts on the case of heuristic optimization algorithms are the worst value of the objective function $f_i(\mathbf{x})$ found in the initial population as suggestion for $f_i^{worst}(\mathbf{x})$, and in the case of lacking an initial guess for $f_i^{*}(\mathbf{x})$, the best value found in the initial population for $f_i(\mathbf{x})$ could be used as $f_i^{*}(\mathbf{x})$.

As in the case of objective functions, inequality constraint functions can also lie in different ranges. Then, when treated by penalty methods (as explained later), that can be misinterpreted as different level of violation instead of different ranges. In that case, it is desired that the scaled versions of the constraints have the same range (or order of magnitude) while restoring the formulation as presented in Eq. (2.12). Thus, assuming that an inequality constraint is as $g_j(\mathbf{x}) \leq g_j^{*}$, there are just two possibilities for the target value, g_j^{*}: $g_j^{*} = 0$, or $g_j^{*} \neq 0$. For the first case, i.e. $g_j^{*} = 0$, the scaled version is given by:

$$\tilde{g}_j(\mathbf{x}) = \frac{g_j(\mathbf{x})}{g_j^{\bullet}} \leq 0 \ , \tag{2.16}$$

where $\tilde{g}_j(\mathbf{x})$ is the scaled version of $g_j(\mathbf{x})$, and $g_j^{\bullet}$ is its order of magnitude.

For the second case, i.e. $g_j^* \neq 0$, $\tilde{g}_j(\mathbf{x})$ is computed using:

$$\tilde{g}_j(\mathbf{x}) = \frac{g_j(\mathbf{x})}{g_j^*} - 1 \leq 0 \ . \tag{2.17}$$

On other circumstances, the optimum must satisfy $g_j(\mathbf{x}) \geq g_j^*$, and as before, the scaled version depends on g_j^*. For $g_j^* = 0$, the scaled version is given by changing the inequality on Eq. (2.16). For $g_j^* \neq 0$, the scaled version is as:

$$\tilde{g}_j(\mathbf{x}) = 1 - \frac{g_j(\mathbf{x})}{g_j^*} \leq 0 \ . \tag{2.18}$$

Finally, for the same reasons, re-writing the set of equality constraints may be convenient. If the target value of $h_j(\mathbf{x})$ is $h_j^* = 0$, the scaled version, $\tilde{h}_j(\mathbf{x})$, is as:

$$\tilde{h}_j(\mathbf{x}) = \frac{h_j(\mathbf{x})}{h_j^{\bullet}} = 0 \ , \tag{2.19}$$

where $h_j^{\bullet}$ is the order of magnitude of $h_j(\mathbf{x})$.

If on the other hand, $h_j^* \neq 0$, then $\tilde{h}_j(\mathbf{x})$ is given by:

$$\tilde{h}_j(\mathbf{x}) = \frac{h_j(\mathbf{x})}{h_j^*} - 1 = 0 \ . \tag{2.20}$$

Now that all functions considered in the optimization problem have been properly scaled, the general optimization problem can be re-stated to be solved by using heuristic optimization algorithms. Equations (2.11) and (2.12) are combined in a functional, $J(\mathbf{x})$, to be minimized. This functional is obtained as follows (Marler and Arora, 2004):

$$J(\mathbf{x}) = F\left(\tilde{\mathbf{f}}(\mathbf{x})\right) + P\left(\tilde{\mathbf{g}}(\mathbf{x}), \tilde{\mathbf{h}}(\mathbf{x})\right) \ , \tag{2.21}$$

where $F\left(\tilde{\mathbf{f}}(\mathbf{x})\right)$ plays the role of the multi-objective handling technique that combines the vector of scaled objective functions, $\tilde{\mathbf{f}}(\mathbf{x})$, and $P\left(\tilde{\mathbf{g}}(\mathbf{x}),\tilde{\mathbf{h}}(\mathbf{x})\right)$ plays the role of the constrained optimization handling technique that combines the vectors of scaled inequality constraints, $\tilde{\mathbf{g}}(\mathbf{x})$, and equality constraints, $\tilde{\mathbf{h}}(\mathbf{x})$. $P\left(\tilde{\mathbf{g}}(\mathbf{x}),\tilde{\mathbf{h}}(\mathbf{x})\right)$ is zero, if no violation occurs, and positive otherwise (actually, the worst the violation the larger the value of $P\left(\tilde{\mathbf{g}}(\mathbf{x}),\tilde{\mathbf{h}}(\mathbf{x})\right)$).

2.4. Multi-Objective Optimization Techniques

When considering multi-objective problems, the algorithms offer alternatives either to compute the compromise solution (Van Veldhuizen and Lamont 2000; and Marler and Arora, 2004) or to build the Pareto front (Zitzler and Thiele, 1999; and Deb, 2001). In this work, the focus is on algorithms used to obtain the compromise solution.

Basically, there are two main strategies used for these techniques. The first consists in building a functional through the manipulation of the objective functions. The second one redefines the multi-objective problem as a constrained optimization problem. For the sake of simplicity, in this work only the most common methods following the first approach are covered.

2.4.1. Weighted Sum Method

This is the most common approach used in multi-objective optimization. The functional $F\left(\tilde{\mathbf{f}}(\mathbf{x})\right)$ is given by:

$$F\left(\tilde{\mathbf{f}}(\mathbf{x})\right) = \sum_{i=1}^{n_{obj}} w_i \tilde{f}_i(\mathbf{x}) \,, \tag{2.22}$$

where $\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_{n_{obj}} \end{bmatrix}^T$ is the vector of weighting factors, which indicate the importance of each objective, such as $\sum_{i=1}^{n_{obj}} w_i = 1$ and $w_i > 0$.

The results of the optimization problem formulated by Eq.(2.22) can vary significantly according to the choice of the weights. Consequently, it is possible to generate the Pareto front by systematically changing the values of the weights (Coello, 1999; and Gerace, 2007).

2.4.2. Compromise Programming Method

While the Weighted Sum Method has the strength of being able to generate the Pareto frontier, it is sensitive to the values of the weights if used to obtain the compromise solution. The Compromise Programming Method represents a variation of Eq.(2.22) pursuing more robustness in generating the compromise solution. The functional $F\left(\tilde{\mathbf{f}}(\mathbf{x})\right)$ is obtained by:

$$F\left(\tilde{\mathbf{f}}(\mathbf{x})\right) = \left[\sum_{i=1}^{n_{obj}} \left(w_i \tilde{f}_i(\mathbf{x})\right)^p\right]^{\frac{1}{p}} . \tag{2.23}$$

Other than the vector $\mathbf{w}$, the solution also depends on the value of the parameter p (generally $p = 2$). Since $\tilde{f}_i(\mathbf{x})$ is between zero and one (or at least $\tilde{f}_i(\mathbf{x}) \geq 0$), p is proportional to the amount of emphasis placed on minimizing the function with the largest difference between $f_i(\mathbf{x})$ and $f_i^{\circ}(\mathbf{x})$ (or, more practically, $f_i^*(\mathbf{x})$).

2.4.3. Weighted Min-Max Method

The Weighted Min-Max Method is presented as follows:

$$F\left(\tilde{\mathbf{f}}(\mathbf{x})\right) = \max_i \left(w_i \tilde{f}_i(\mathbf{x})\right) . \tag{2.24}$$

This is equivalent to the limit of Eq. (2.23) as $p \rightarrow \infty$. Therefore, Eq. (2.24) can provide the complete Pareto optimal set by varying the values of the weights as well (Marler and Arora, 2004).

2.5. Constrained Optimization Techniques

There exist several methods proposed for handling constraints by heuristic algorithms for numerical optimization problems (Michalewicz, 1995; and Coello, 2002). In this research, the focus is on those based on the concept of penalty functions.

Before calculating $P\left(\tilde{\mathbf{g}}(\mathbf{x}),\tilde{\mathbf{h}}(\mathbf{x})\right)$, it is necessary to pre-handle the set of constraints. The set of $\tilde{\mathbf{g}}(\mathbf{x})$ and $\tilde{\mathbf{h}}(\mathbf{x})$ functions are grouped to form a new set of functions, $c_j(\mathbf{x})$, that represents all the constraints. One of the possible options is given below:

$$c_j\left(\tilde{\mathbf{g}}(\mathbf{x}),\tilde{\mathbf{h}}(\mathbf{x})\right) = \begin{cases} \max\left(\tilde{g}_j(\mathbf{x}),0\right), & j = 1,\ 2,\ \ldots,\ n_{ineqcnstrt}\ , \\ \left|\tilde{h}_j(\mathbf{x})\right|, & j = m+1,\ m+2,\ \ldots,\ n_{cnstrt}\ . \end{cases} \tag{2.25}$$

Equation (2.25) presents a proper treatment for the inequality constraints, i.e., $c_j(\mathbf{x})$ is zero if there is no violation, and $\tilde{g}_j(\mathbf{x})$ otherwise. However, if the equality constraints follow rigorously the definition given by Eq. (2.12), it does not present a practical solution for the heuristic algorithms. The formulation as in $\left|\tilde{h}_j(\mathbf{x})\right|$ has the inconvenience of making infeasible regions of the design space close to $\tilde{h}_j(\mathbf{x}) = 0$. Then, the solution is transforming an equality function, $\tilde{h}_j(\mathbf{x})$, to an inequality one, $\tilde{g}_j(\mathbf{x})$, of the form:

$$\tilde{g}_j(\mathbf{x}) = \left|\tilde{h}_j(\mathbf{x})\right| - \varepsilon \leq 0 \ , \tag{2.26}$$

where ε is a tolerance to the violation, usually a "small" value.

Finally, with the redefinition of the equality constraints as inequality constraints as given by Eq. (2.26), Eq. (2.25) is re-written as:

$$c_j(\tilde{\mathbf{g}}(\mathbf{x})) = \max\left(\tilde{g}_j(\mathbf{x}), 0\right), \ j = 1,\ 2,\ \ldots,\ n_{cnstrt} \ , \tag{2.27}$$

and the functional for the constraints $P\left(\tilde{\mathbf{g}}(\mathbf{x}), \tilde{\mathbf{h}}(\mathbf{x})\right)$ turns into $P(\tilde{\mathbf{g}}(\mathbf{x}))$.

2.5.1. *Static Penalties Method*

In this method, the functional $P(\tilde{\mathbf{g}}(\mathbf{x}))$ is given by:

$$P(\tilde{\mathbf{g}}(\mathbf{x})) = \sum_{i=1}^{n_{obj}} r_i \left[c_i(\tilde{\mathbf{g}}(\mathbf{x}))\right]^{\beta} \tag{2.28}$$

where β is a constant (usually $\beta = 2$) and $\mathbf{r} = \begin{bmatrix} r_1 & r_2 & \ldots & r_{n_{cnstrt}} \end{bmatrix}^T$ is the vector of weighting factors used to express the penalty degree of each constraint.

Differently from what happens in multi-objective problems, for which the weights lead to different solutions on the Pareto front, here the weights may be used to distinguish the importance of individual constraints. While this can change the level of violation, it may not change the solution of the problem (i.e. the set of optimum design variables and the value of the objective functions). As a consequence, using the same value, r, for all elements of the vector $\mathbf{r}$ is common. In this case, Eq. (2.28) is re-defined as:

$$P(\tilde{\mathbf{g}}(\mathbf{x})) = r \sum_{i=1}^{n_{obj}} \left[c_i(\tilde{\mathbf{g}}(\mathbf{x}))\right]^{\beta} \tag{2.29}$$

2.5.2. Dynamic Penalties Method

The second and last method used in this research imposes dynamic penalties. The idea is to make the penalization harder along the evolution of the optimization process. At the beginning of the process there is little or no knowlegment about the problem. At this point, penalization could inhibit the proper exploration of the design space. As the optimization evolves, it is expected that infeasible regions get harder penalties. In practice, the functional $P(\tilde{\mathbf{g}}(\mathbf{x}))$, at the iteration t, is given by:

$$P(\tilde{\mathbf{g}}(\mathbf{x})) = (C \times t)^{\alpha} \sum_{i=1}^{n_{obj}} [c_i(\tilde{\mathbf{g}}(\mathbf{x}))]^{\beta} \ , \tag{2.30}$$

where C, α, and β are constants. As suggested in Michalewicz (1995), a reasonable choice for these parameters is $C = 0.5$ and $\alpha = \beta = 2$.

When using Eqs. (2.28), (2.29) or (2.30), heuristic optimization methods do not have any extra information (such as Lagrange multipliers). Then, these equations define the level of violation, if any, of each individual of the population. As a consequence, the penalty parameters ($\mathbf{r}$, r, or C) may assume large values in order to better differentiate the feasible region of the design space.

This chapter reviewed concepts about surrogate modeling and the optimization problem formulation as used by heuristic algorithms. The algorithms themselves are presented in the next chapter.

CHAPTER III

HEURISTIC OPTIMIZATION ALGORITHMS

3.1. Introduction

Chapter II showed the set of activities performed during the solution of an optimization problem as well as general concepts on surrogate modeling and formulation of the optimization problem as treated by heuristic optimization algorithms. This chapter presents the basic schemes of the algorithms and theoretical aspects of the contributions proposed during this doctoral research.

Even though classical methods (where the search process is guided by information regarding the gradient) have been widely used due to their computational efficiency, they have difficulties when a local minimum is found (often erroneously interpreted as the global one). On the other hand, Venter and Sobieszczanski-Sobieski (2003) discussed that in recent years non-gradient nature based, probabilistic search algorithms (which generally mimic some natural phenomena) have attracted much attention from the scientific community due to features such as easiness to code, ability of handle non-continuous functions, capability of using parallel architectures, and the trend of finding the global, or near global, solution. The large number of function evaluations required by these methods often presents a drawback when compared with classical gradient-based algorithms, despite the advances in computational throughput that have helped the use this class of algorithm in real world problems (Venter and Sobieszczanski-Sobieski, 2003 and Viana and Steffen Jr, 2005).

As introduced in Chapter I, the focus of this work is on heuristic and population-based optimization algorithms (i.e. algorithms that use a set of points updated via statistical rules for exploration of the design space during the optimization task). More precisely, Ant Colony Optimization (ACO), Differential Evolution (DE), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), LifeCycle Optimization (LC), and the Enhanced Stochastic Evolutionary Algorithm (ESEA) are the set of algorithms studied in this thesis. These methods generate new points in the design space by applying operators to current points and statistically moving toward more optimal places in the design space. They rely on a heuristic-driven search of a finite solution space and do not require function derivatives. Consequently, they can deal with discrete variables and non-convex and non-continuous functions. The following sections describe specific details as well as the common ground of the algorithms.

3.2. Ant Colony Optimization

Ant Colony Optimization (ACO) is inspired in the behavior of real ants and their communication scheme by using pheromone trail (Dorigo, 1992). When searching for food, real ants start moving randomly, and upon finding food they return to their colony while laying down pheromone trails (Socha, 2004). This means that if other ants find such a path, they return and reinforce it. However, over time the pheromone trail starts to evaporate, thus reducing its attractive strength. When a short and a long path are compared, it is easy to see that a short path gets marched faster and thus the pheromone density remains high. Consequently, if one ant finds a short path (from the optimization point of view, it means a good solution) when marching from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually encourages all the ants in following the same single path. ACO follows some basic concepts, as presented below (Socha, 2004):

- A search performed by a population of ants, i.e., by simple independent agents.
- Incremental construction of solutions.
- Probabilistic choice of solution components based on stigmergic information of pheromone. A stigmergic process is the process through which the results of a worker insect's activity act as a stimulus to further activities.
- No direct communication between ants.

The outline of a basic ACO algorithm is presented in Figure 3.1.

The question that arises is how ACO, first used to solve combinatorial problems (Dorigo et al., 1996, Dorigo and Gambardella, 1997), deals with continuous variables. A possible answer to this question would be the conversion of the continuous variables and the pheromone trail into a string of bits as proposed in the past by Monmarché et al. (1999). Instead, the present continuous version extends ACO in order to work directly with continuous variables.

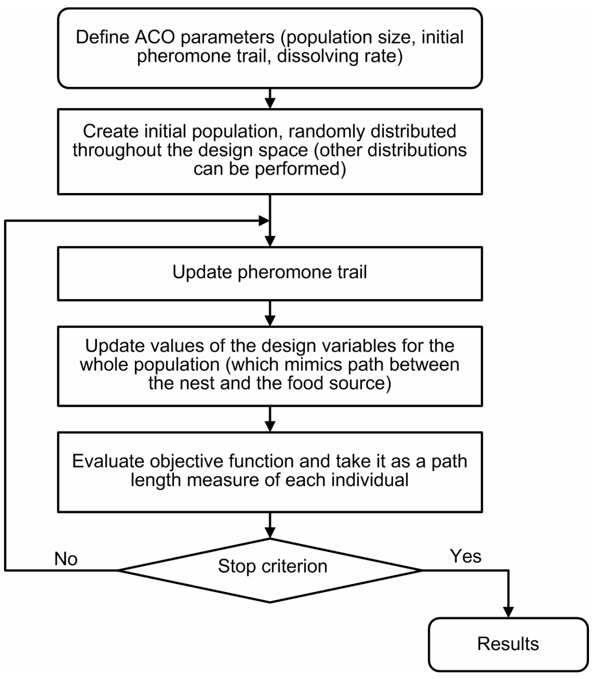

Define ACO parameters (population size, initial pheromone trail, dissolving rate)
Create initial population, randomly distributed throughout the design space (other distributions can be performed)
Update pheromone trail
Update values of the design variables for the whole population (which mimics path between the nest and the food source)
Evaluate objective function and take it as a path length measure of each individual
Stop criterion
No
Yes
Results

Figure 3.1. Basic scheme for ACO.

Following the idea of directly using continuous variables, a population of n_{pop} ants (herein referred to as individuals) is expressed as a $n_{pop} \times n_{dv}$ matrix, $\mathbf{P} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{n_{pop}} \end{bmatrix}^T$. The continuous version of ACO models the pheromone trail as an amount of pheromone laid on the path. As suggested by Socha (2004) and Pourtakdoust and Nobahari (2004), for the continuous model implementation, this is done by using a normal probability distribution function (PDF). Then, for the $i\text{-}th$ dimension of the design space, the pheromone trail, τ_i, is given by:

$$\tau_i(x) = \exp\left(-\frac{\left(x - x_i^*\right)^2}{2\left(\sigma_i^{ACO}\right)^2} \right) \tag{3.1}$$

$$\sigma_i^{ACO} = \sqrt{\frac{1}{n_{pop}-1}\sum_{j=1}^{n_{pop}}\left(z_j - \overline{z}\right)} \tag{3.2}$$

where:

- $\exp(\cdot)$ is the exponential function ;
- x_i^* is the $i\text{-}th$ coordinate of the best point found by the optimization task within the design space until the current iteration;

- σ_i^{ACO} is an index related to the aggregation of the population around the current minimum for the $i\text{-}th$ coordinate of the design space;
- $\mathbf{z}$ is the $i\text{-}th$ column of the population matrix $\mathbf{P}$; and
- $\overline{z}$ is the mean value of the vector $\mathbf{z}$.

The updating process of the values of each design variable for all individuals is the process in which, for a given iteration, each individual sets the values for the trial solution based on the probability distribution given by Eq. (3.1). Computationally, this is implemented through a random number generator based on a normal PDF. From Eq. (3.1), it can be noticed that each variable uses a different random number generator together with its respective PDF. About the pheromone scheme, it is possible to see that the concentration of pheromone increases in the area of the candidate to the optimum. This approach (also called as positive update) reinforces the probability of the choices that lead to good solutions. However, for avoiding premature convergence, negative update procedures are not discarded, as described by Socha (2004). In this work, a simple method to perform negative update is used: dissolving of the pheromone. The idea of this scheme is to spread the amount of pheromone by changing the current standard deviation (for each variable) according to the following equation:

$$\sigma_{new}^{ACO} = \gamma\sigma_{old}^{ACO} , \tag{3.3}$$

where $\gamma > 1$ is the dissolving rate (usually $\gamma = 1.25$). The dissolving rate affects the exploration capabilities, and consequently, the convergence of the algorithm.

At the initialization the algorithm:

- $\mathbf{x}_{\min}$ is randomly chosen within the design space using a uniform PDF;
- σ^{ACO} is taken as being 3 times greater than the length of the search interval.

The SIMPLE Optimization toolbox of Viana and Steffen Jr (2008) was used to run ACO algorithm.

3.3. Differential Evolution

Differential Evolution (DE) was proposed by Storn and Price (1995) as an algorithm to solve global optimization problems of continuous variables. Even though the DE community has adopted biological terms to describe arithmetic operations on real (and discrete) variables, i.e., mutation and recombination, the operations are more akin to dynamics/thermodynamics and physics than they are to biology. Selection is perhaps the

DE's most Darwinian aspect, but even here, competition for survival is more limited than in most evolutionary algorithms. The main idea behind DE is how possible solutions taken from the population of individuals are set, recombined and chosen to evolve the population to the next generation (Storn and Price, 1997, Ronkkonen et al., 2005). This way, the weighted-difference scheme seems to be the most important feature of the present heuristic.

In a population of individuals (seen as potential solutions), a fixed number of points (or vectors in the DE nomenclature) are randomly initialized, and then evolved over the optimization task to explore the design space and hopefully to locate the optimum of the objective function. At each iteration, new vectors are generated by the combination of vectors randomly chosen from the current population (this is also referred to as "mutation" in the DE literature). The out coming vectors are then mixed with a predetermined target vector. This operation is called "recombination" and produces a "trial vector". Finally, the "trial vector" is accepted for the next generation if and only if it yields a reduction in the value of the objective function. This last operator is referred to as "selection." As can be seen, the basic algorithm preserves some common aspects of the traditional simple GA (specially the nomenclature of some operators, such as selection, crossover and mutation).

Again, a population of n_{pop} individuals is expressed as a $n_{pop} \times n_{dv}$ matrix, $\mathbf{P} = \left[\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots \quad \mathbf{x}_{n_{pop}}\right]^T$. The outline of a basic DE is as shown in Figure 3.2.

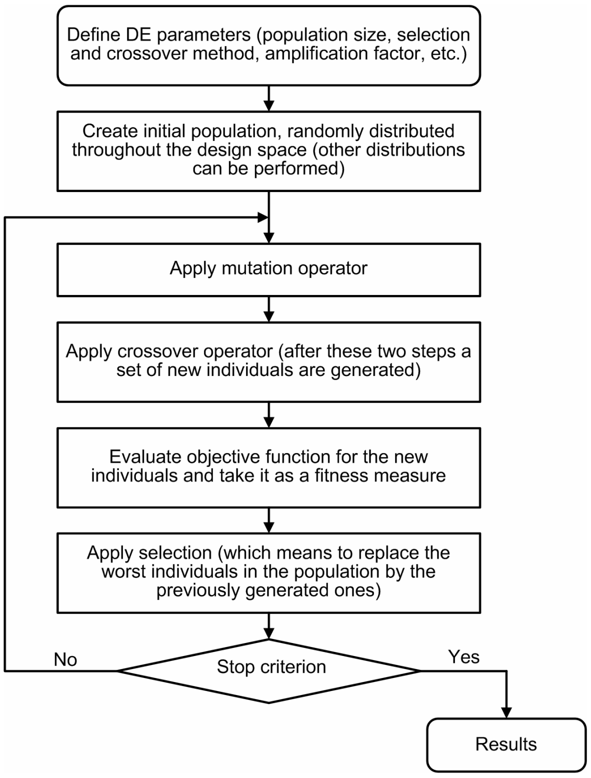

Define DE parameters (population size, selection and crossover method, amplification factor, etc.)
Create initial population, randomly distributed throughout the design space (other distributions can be performed)
Apply mutation operator
Apply crossover operator (after these two steps a set of new individuals are generated)
Evaluate objective function for the new individuals and take it as a fitness measure
Apply selection (which means to replace the worst individuals in the population by the previously generated ones)
No
Stop criterion
Yes
Results

Figure 3.2. Basic scheme for DE.

The DE community has established the following notation to determine how the mutation and crossover operators work together (Storn and Price, 1995; Storn and Price, 1997; and Ronkkonen et al., 2005): $DE\,/\,a\,/\,b\,/\,c$, where:

- a specifies the vector to be mutated, which currently can be one of the following:
  - "rand": a randomly chosen population vector,
  - "best": the vector of best objective value from the current population,
  - "rand-to-best": takes both randomly and best individuals.
- b is the number of difference vectors used, and
- c denotes the crossover scheme, which can be either "bin" (independent binomial experiments) or "exp" (exponential crossover).

3.3.1. Mutation

As described before, the DE mutation is the operator that first adds the weighted difference between two individuals to a third individual. Table 3.1 gives more details about each mutation scheme.

Table 3.1 Different implementations of the DE mutation.

Type	Updating equation	Target vector	Population size
$rand\,/\,1$	$\mathbf{x}_{trial} = \mathbf{x}_{r1} + F(\mathbf{x}_{r2} - \mathbf{x}_{r3})$	$\mathbf{x}_{r1}$	> 3
$rand\,/\,2$	$\mathbf{x}_{trial} = \mathbf{x}_{r1} +$ $F(\mathbf{x}_{r2} - \mathbf{x}_{r3} + \mathbf{x}_{r4} - \mathbf{x}_{r5})$	$\mathbf{x}_{r1}$	> 5
$best\,/\,1$	$\mathbf{x}_{trial} = \mathbf{x}_{best} + F(\mathbf{x}_{r2} - \mathbf{x}_{r3})$	$\mathbf{x}_{best}$	> 3
$\mathrm{best}/2$	$\mathbf{x}_{trial} = \mathbf{x}_{best} +$ $F(\mathbf{x}_{r1} - \mathbf{x}_{r2} + \mathbf{x}_{r3} - \mathbf{x}_{r4})$	$\mathbf{x}_{best}$	> 5
$rand\text{-}to\text{-}best\,/\,1$	$\mathbf{x}_{trial} = \mathbf{x} +$ $F(\mathbf{x}_{best} - \mathbf{x} + \mathbf{x}_{r1} - \mathbf{x}_{r2})$	$\mathbf{x}$	> 5
$rand\text{-}to\text{-}best\,/\,2$	$\mathbf{x}_{trial} = \mathbf{x} + F(\mathbf{x}_{best} - \mathbf{x}) +$ $F(\mathbf{x}_{r1} - \mathbf{x}_{r2} + \mathbf{x}_{r3} - \mathbf{x}_{r4})$	$\mathbf{x}$	> 6

where:

- $\mathbf{x}_{trial}$ is the trial vector (individual);
- $r1,\ r2,\ r3,\ r4,\ r5\ \in\ \{1,\ 2,\ \ldots, m\}$ are random integer indexes and mutually different;

- F is a real constant factor $\in [0, 2]$, which controls the amplification of the differential variation;
- x_{best} is the best individual of the current population; and
- x is the current individual.

In DE, the second step of the design variable update is the crossover operator. There are two types of crossovers: binary and exponential. The superiority of each crossover scheme over the other can not be uniquely defined, which means that this is a problem-dependent operator. Thus, the final trial vector, x_{trial}, is formed in such a way that the i-th coordinate is given by:

$$x_{trial} = \begin{cases} x_i^{mutation}, & \text{if crossover is applicable,} \\ x_i, & \text{otherwise,} \end{cases} \qquad i = 1, 2, \ldots, n_{dv}. \tag{3.4}$$

In practical terms, DE has a parameter that controls the crossover, namely the crossover probability, CR. In exponential crossover, the crossover is performed in a loop controlled by a uniform random number, rn, and a counter, i, that ranges from 1 to n_{dv}. The first time a randomly picked number between 0 and 1 goes beyond the CR value, no crossover is performed and the remaining of the n_{dv} variables is left intact (Storn and Price, 1995; Storn, 1996; Storn and Price, 1997; and Ronkkonen et al., 2005). Figure 3.3 shows a flowchart for the exponential crossover.

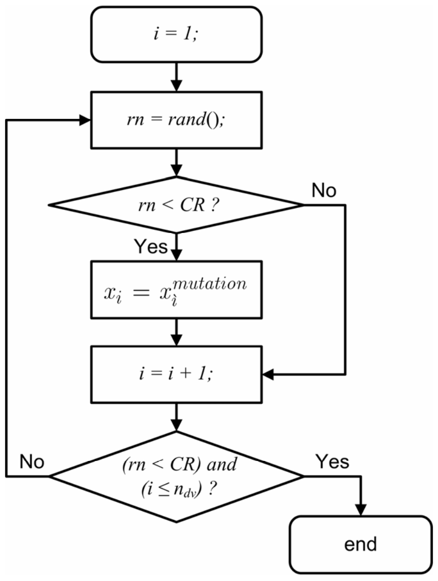


Figure 3.3. Basic scheme for the DE exponential crossover. Consider *rand*() as a function for generating a uniform random number.

In the binomial crossover, the operation is performed on each of the n variables whenever a randomly picked number between 0 and 1 is lesser than the CR value (Storn and Price, 1995; Storn, 1996; Storn and Price, 1997; and Ronkkonen et al., 2005). Figure 3.4 shows a flowchart for the binomial crossover.

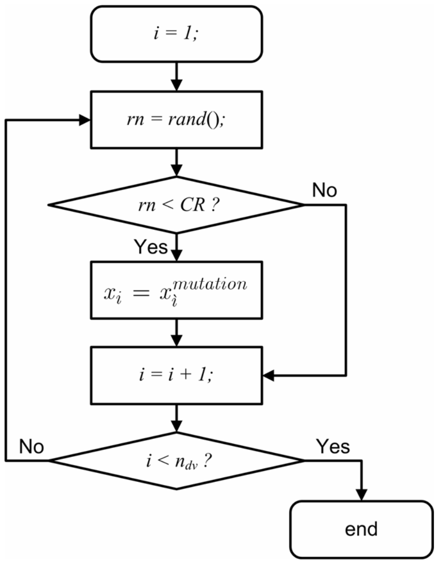


Figure 3.4. Basic scheme for the DE binomial crossover. Consider rand() as a function for generating a uniform random number.

It can be noticed that for high values of CR, the exponential and binomial crossovers yield similar results.

3.3.2. Selection

Selection is the simplest DE operator. It is used to decide whether or not the new vector, $\mathbf{x}_{new}$, should become a member of the next generation. The approach is straightforward: the objective function value of the new individual, $f(\mathbf{x}_{new})$, is compared with the one of the target vector, $f(\mathbf{x}_{\text{target}})$. If there is an improvement, $\mathbf{x}_{new}$ is set to the next generation; otherwise, $\mathbf{x}_{\text{target}}$ is retained.

3.3.3. Remarks about DE

Storn (1996) gives a set of practical rules useful when it comes to choose the DE parameters, namely:

- A population size of 10 times the number of design variables is a good choice for many applications. However, it is worth mentioning that various scientists have been successful in using a smaller population for DE.

- Most often the crossover probability, CR, must be considerably lower than one (e.g. 0.3). However, if no convergence can be achieved, $CR \in [0.8,\ 1]$ should be tried. This is especially true when using exponential crossover.
- The higher the population size is chosen, the lower the weighting factor F.
- F is usually chosen $\in [0.5,\ 1]$.

Last but not least, it is worth mentioning that by combining the six mutation schemes shown in Table 3.1 with the two previously presented crossover schemes, a total number of twelve DE strategies can be explored. By following the $DE/a/b/c$ notation: DE/rand/1/bin, DE/rand/1/exp, DE/rand/2/bin, DE/rand/2/exp, DE/best/1/bin, DE/best/1/exp, DE/best/2/bin, DE/best/2/exp, DE/rand-to-best/1/bin, DE/rand-to-best/1/exp, DE/rand-to-best/2/bin, and DE/rand-to-best/2/exp.

The code developed by Price et al. (2005) was used to execute the DE algorithm.

3.4. Genetic Algorithm

Genetic Algorithm (GA) is an optimization algorithm used to find approximate solutions to difficult-to-solve problems through the application of the principles of evolutionary biology to computer science. GA is based on Darwin's theory of survival and evolution of species, as explained by Gen and Cheng (1999), Michalewicz and Fogel (2000) and Haupt and Haupt (2004). GA uses biologically-derived concepts such as inheritance, mutation, natural selection, and recombination (or crossover). Due to the vast literature on GA, for the sake of simplicity, just an overview is provided in this text.

The algorithm starts from a population of random individuals, viewed as candidate solutions to a problem. During the evolutionary process, each individual of the population is evaluated, reflecting its adaptation capability to the environment. Some of the individuals of the population are preserved while others are discarded; this process mimics the natural selection in Darwinism. The members of the remaining group of individuals are paired in order to generate new individuals to replace those that are discarded in the selection process. Finally, some of them can be submitted to mutation, and as a consequence, the chromosomes of these individuals are altered. The entire process is repeated until a satisfactory solution is found. The outline of a basic GA is as shown in Figure 3.5.

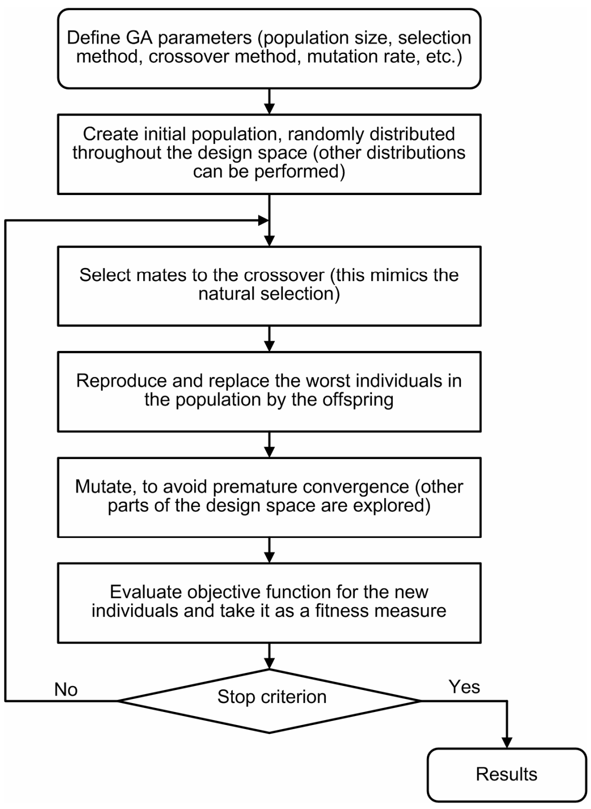

Define GA parameters (population size, selection method, crossover method, mutation rate, etc.)
Create initial population, randomly distributed throughout the design space (other distributions can be performed)
Select mates to the crossover (this mimics the natural selection)
Reproduce and replace the worst individuals in the population by the offspring
Mutate, to avoid premature convergence (other parts of the design space are explored)
Evaluate objective function for the new individuals and take it as a fitness measure
No
Stop criterion
Yes
Results

Figure 3.5. Basic scheme for GA.

Although the initially proposed GA algorithm was dedicated to discrete variables only, nowadays improvements are available to deal with discrete and continuous variables (see Gen and Cheng, 1999; Michalewicz and Fogel, 2000; and Haupt and Haupt, 2004).

The SIMPLE Optimization toolbox of Viana and Steffen Jr (2008) was used to run GA algorithm.

3.5. Particle Swarm Optimization

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart (1995), as emerged from experiences with algorithms inspired in the social behavior of some bird species. Consider the following situation: a swarm of birds searching for food around a delimited area. Suppose there is just one place with food and the birds do not know where it is. The success of one bird is shared by the whole swarm (learning from the experience of other individuals). In this sense, the adjustment between exploration (the capacity of individual search) and exploitation (taking advantage of someone else's success) is required. If there is little exploration, the birds will all converge on the first good place encountered. On the other hand, if there is little exploitation, the birds will slowly converge or they will try to find food alone. It is clear that the best policy is a trade-off between both policies.

In PSO, the flight of each bird (herein referred as individual) is modeled by using a velocity vector, which considers the contribution of the current velocity, as well as two other parts accounting for the self-knowledge of the individual and the knowledge of the swarm (herein referred to as population) about the search space. This way, the velocity vector is used to update the position of each individual in the population (Kennedy and Eberhart, 1995; Parsopoulos and Vrahatis, 2002; and Venter and Sobieszczanski-Sobieski, 2003). An outline of a basic PSO algorithm is shown in Figure 3.6.

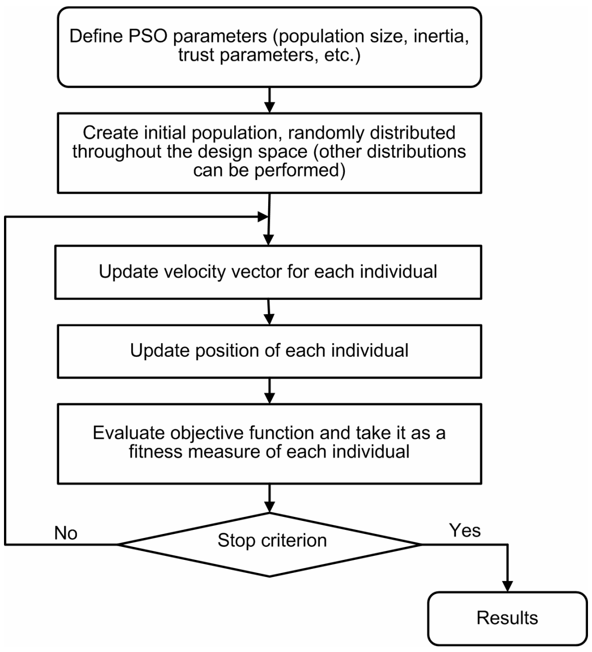

Define PSO parameters (population size, inertia, trust parameters, etc.)
Create initial population, randomly distributed throughout the design space (other distributions can be performed)
Update velocity vector for each individual
Update position of each individual
Evaluate objective function and take it as a fitness measure of each individual
Stop criterion
No
Yes
Results

Figure 3.6. Basic scheme for PSO.

The position of each individual is updated according to the following equation:

$$x_k^i = x_{k-1}^i + v_k^i \Delta t \ , \tag{3.5}$$

where x_k^i is the position of the individual i in the iteration k; v_k^i represents the corresponding velocity vector; and Δt is the time step (generally $\Delta t = 1$).

The velocity vector is updated as follows:

$$v_k^i = wv_{k-1}^i + c_1 r_1 \frac{(p^i - x_{k-1}^i)}{\Delta t} + c_2 r_2 \frac{(p_{k-1}^s - x_{k-1}^i)}{\Delta t} \ , \tag{3.6}$$

where w is the inertia of the individual; c_1 and c_2 are the two "trust" parameters; r_1 and r_2 are random numbers between 0 and 1; p^i is the best position found by the individual i up to the current iteration; and p_{k-1}^s is the best individual in the iteration $k-1$.

Equation (3.6) can be interpreted as a sum of three different contributions. The first term, pondered by the inertia, w, gives how much of “exploration” there will be in the resulting velocity. The second term, pondered by the self trust parameter, c_1, indicates how much the knowledge of the individual itself there will be in the resulting velocity. Finally, the term weighted by the trust on the population, c_2, shows how much of the population success will influence the resulting velocity. As a consequence, the larger the inertia value, the more global (as opposed to individualistic) will be the behavior. The trust parameters c_1 and c_2 must be set to balance the influence of the amount of knowledge that was acquired by the individual itself and the knowledge acquired by the population.

3.5.1. Initial Population

The initial population is created by randomly distributing the individuals throughout the search space. The initial position and the initial velocity vectors are given by:

$$x_0^i = x_{\min} + r_1(x_{\max} - x_{\min}) \ , \tag{3.7}$$

$$v_0^i = \frac{x_{\min} + r_2(x_{\max} - x_{\min})}{\Delta t} \ , \tag{3.8}$$

where $x_{\min}$ is the lower bound vector; and $x_{\max}$ is the upper bound vector for the variables.

3.5.2. Algorithm Parameters

The velocity vector update formula has some parameters that can be adjusted according to the problem, namely the trust parameters, c_1 and c_2, and the inertia weight, w.

Kennedy and Eberhart (1995) initially suggested $c_1 = c_2 = 2$, as a way to equally weight the contribution given by the individual and the population. Later, various works (Shi and Eberhart, 1998; Fourie and Groenwold, 2002; and Venter and Sobieszczanski-Sobieski, 2003) proposed that trust parameters could be set to different values, generally satisfying $c_1 + c_2 = 4$. In this work, the default values are $c_1 = 1.5$ and $c_2 = 2.5$, as used by Venter and Sobieszczanski-Sobieski (2003).

The inertia parameter, in general, is such that $0.8 \le w \le 1.4$. In addition, a scheme of dynamic reduction of w has several advantages (Venter and Sobieszczanski-Sobieski, 2003). It improves the convergence to the optimum solution and makes the problem less sensitive to the w parameter. The idea is to start with a large value for w, since the knowledge bases of the individual and the population are empty, and then, to reduce continuously that value along the optimization process, privileging the flying experiences. The following equation gives the updating scheme for w:

$$w_{new} = f_w w_{old} \ , \tag{3.9}$$

where w_{new} is the updated value; w_{old} is the previous value; and f_w is a constant between 0 and 1 (usually, $f_w = 0.975$).

The inertia is not updated each iteration. Instead, a dispersion measure, DM, as will be described in Section 3.7.1, of a subset of the individuals is used to guide when to apply the inertia reduction. If the DM value of this subset of individuals falls below a pre-defined threshold value, it is understood that the algorithm is converging to an optimum, then Eq. (3.9) is applied. As suggested by Venter and Sobieszczanski-Sobieski (2003), a subset of the best 20% of the individuals from the population is monitored and $DM_{thresh} = 0.1 \times D_{\max}$ is used as threshold for DM.

3.5.3. Dealing with Violated Constraints

When in an optimization problem the individuals violate the constraints, a strategy for repairing the violation has to be considered. As suggest by Venter and Sobieszczanski-Sobieski (2003), the idea of feasible directions (Vanderplaats, 2005) is used to achieve this goal. This way, when the constraints are violated, the velocity vector is recomputed according to the equation:

$$v_k^i = c_1 r_1 \frac{(p^i - x_{k-1}^i)}{\Delta t} + c_2 r_2 \frac{(p_{k-1}^s - x_{k-1}^i)}{\Delta t} \tag{3.10}$$

After obtaining the new velocity vector, the position is recalculated by using Eq. (3.5). The difference between this formula and the initial one is that the current velocity is neglected. Since just the elements of the best solution found by the individual and the population are considered, it is expected that this new velocity vector will bring the individual back to a feasible region of the search space. When this is not the case, Eq. (3.10) is applied repetitively until the individual falls in a feasible region.

The SIMPLE Optimization toolbox of Viana and Steffen Jr (2008) was used to execute PSO algorithm.

3.6. LifeCycle Algorithm

LyfeCycle (LC) algorithm is a hybrid heuristic optimization method inspired by the idea of life cycle stages, initially proposed by Krink and Løvberg (2004). From the mathematical point of view, natural algorithms such as ACO, DE, GA, and PSO are heuristic search methods of proven efficiency as optimization tools. LC is intended to avoid poor performance of individual methods and creates a self-adaptive optimization approach (Flores et al., 2007). Each individual, as a candidate solution, decides based on its success if it would prefer to belong to a population of an ACO, DE, GA, or PSO algorithm. This means that various heuristic techniques contribute to form a robust high performance optimization tool. The idea is that complex problems can be conveniently considered from the optimization viewpoint. As can be seen, the less successful individuals must change their status in order to improve their fitness. This means that the optimization approach does not follow a rigid scheme, in which various techniques are used sequentially in a cascade-type of structure. In other words, it is the mechanism of self-adaptation to the optimization problem that rules the procedure. A LC outline follows in Figure 3.7.

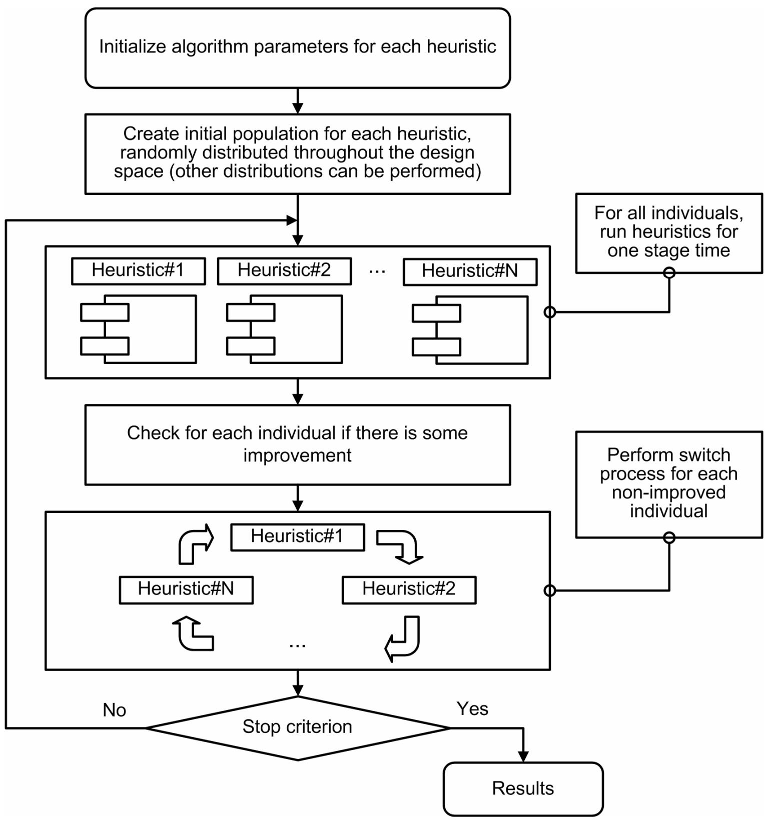

Initialize algorithm parameters for each heuristic
Create initial population for each heuristic, randomly distributed throughout the design space (other distributions can be performed)
For all individuals, run heuristics for one stage time
Heuristic#1
Heuristic#2
...
Heuristic#N
Check for each individual if there is some improvement
Perform switch process for each non-improved individual
Heuristic#1
Heuristic#N
Heuristic#2
...
No
Stop criterion
Yes
Results

Figure 3.7. Basic scheme for LC.

Since the algorithm is composed by various heuristics, it is necessary to set the parameters of every heuristic used in the LC. Nevertheless, there is a parameter inherent to the LC, namely the number of iterations that represent a stage of the LC, known as stage interval. At the end of each stage interval, the less successful individuals must change their stage in order to improve their fitness. During the optimization procedure the agents of each subpopulation commute to the other in such a way that its own fitness is improved. In other words, it is the mechanism of self-adaptation to the optimization problem that rules the procedure. To close the definition, LC stages must be presented. In the present work, two or more of the previously presented heuristics are used as stages. Other versions of the LC can be proposed by considering other heuristics and a mix of them, as shown by Krink and Løvberg (2004).

The SIMPLE Optimization toolbox of Viana and Steffen Jr (2008) was used to run LC algorithm.

3.7. Common Ground of the Basic Algorithms

Despite the differences on the inspiration of the previously presented algorithms, they share common points such as:

- Modeling a set of points on the design space as a population of individuals: each individual represents a vector of design variables, whose values are iteratively updated according to a heuristic rule.
- Ability to have the population split in clusters during the optimization task: this is when the population is divided in sub-populations, or clusters, that run the optimization separately from each other. Eventually, these clusters can exchange information when a migration operator is applied.
- Possibility of applying elitism and additional randomness operators: even though these two operators were first used in GA, they can be easily extended to all previously discussed methods. In fact, DE uses a sort of the elitism during the selection operator, and the PSO community has already tested a randomness operator as can be viewed in Venter and Sobieszczanski-Sobieski (2003).

Therefore, a unified framework is naturally proposed, as illustrated by Figure 3.8.

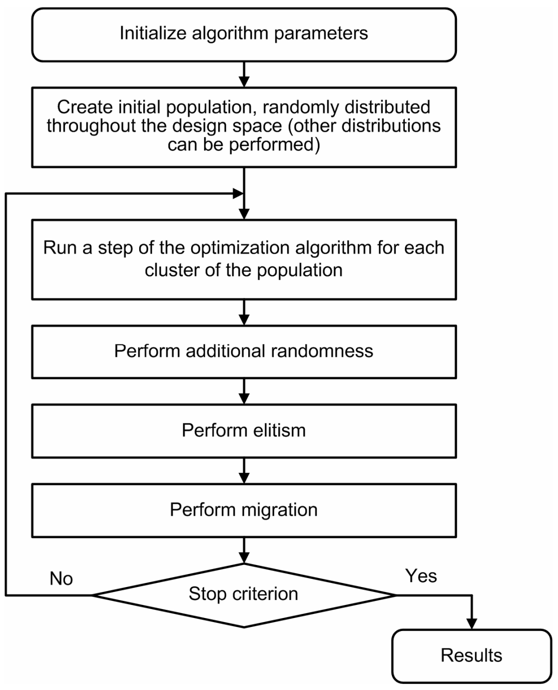

Initialize algorithm parameters
Create initial population, randomly distributed throughout the design space (other distributions can be performed)
Run a step of the optimization algorithm for each cluster of the population
Perform additional randomness
Perform elitism
Perform migration
No
Stop criterion
Yes
Results

Figure 3.8. Framework for population-based heuristic optimization algorithms.

Going deeper on the discussion of each step:

1. Initialize algorithm parameters: this is when different operators and parameters are set, including both general, such as migration scheme and population size, and heuristic-specific, such as the GA crossover operator or the PSO trust parameters.
2. Create the initial population: at this step, the design space is sampled. The intention is to raise as much information as possible. To accomplish that, a random sampling or other statistic-based sampling strategies are commonly applied.
3. Run a step of the optimization algorithm for each cluster of the population: as described in previous sections, each of the algorithms has an internal loop that besides dealing with specific data structures, basically, performs selection, generation and replacement of individuals in the population. It is important to notice that the evaluations of the functions occur during this step.
4. Perform additional randomness: as in the case of the GA mutation, an additional randomness can be applied to the new population in order to avoid premature convergence so that other parts of the design space can be explored. It is worth mentioning that since LC performs complete steps of each algorithm, there is no need to be redundant here. Therefore, LC has no additional randomness operator.
5. Perform elitism: a pre-defined number of individuals are kept from iteration to iteration according to their rank in the population.
6. Perform migration: when the population is divided in clusters, migration refers to the transfer of an individual from one cluster to another. This allows clusters to

communicate, increasing diversity. A migration rate must be chosen to determine how often individuals migrate to other clusters. Cantu-Paz (2001) reviews several migration schemes. In this work, a simple ring-type scheme is implemented, i.e., the $n\text{-}th$ cluster migrates into the $(n+1)\text{-}th$ cluster (the last cluster migrates into the first). A migration fraction parameter controls how many individuals move from one cluster to the other and the migration interval controls how many iterations occur before another migration occurs.

7. Stop criterion: this is necessary to stop the optimization task avoiding any additional function evaluations.

Now that the common framework is clear, the following sub-sections cover details about how to deal with discrete/integer variables, different implementations of the randomness operator, and a set of stopping criteria.

3.7.1. Dispersion Measure of the Population

Some of the operators are only applied when a dispersion measure of the population follows below a threshold. Since all design variables are normalized between 0 and 1, a convenient dispersion measure, DM, can be defined as follow:

$$DM = \frac{\bar{D}}{D_{\max}} \tag{3.11}$$

where:

- $\bar{D} = \frac{1}{n_{pop}} \sum_{i=1}^{n_{pop}} \|\mathbf{x}_i - \bar{\mathbf{x}}\|$: defines the dispersion of the population and it is equivalent to the mean value of the Euclidian distance between each individual and $\bar{\mathbf{x}}$.
- $\bar{\mathbf{x}}$: is the vector of mean values of each design variable for the population $\mathbf{P}$, i.e. $\bar{x}_j = \frac{1}{n_{pop}} \sum_{i=1}^{n_{pop}} \mathbf{P}_{ij}$ (i and j represents the design variable and the individual of the population, respectively).
- $D_{\max} = \frac{1}{2}\left\|\mathbf{1}_{n_{dv}\times 1}\right\|$: defines the maximum possible dispersion of the population. Each of the n_{dv} elements of the vector $\mathbf{1}$ is equal to 1. Assuming that all individuals are uniformly distributed on the borders of the design space, $D_{\max}$ is the mean value of the Euclidian distance between each individual and the center of the design space.

DM represents the ratio between the current dispersion of the population around its mean value and the maximum possible dispersion. Then, the smaller the DM value the more aggregated is the population.

Frequently, the operators are only applied when a dispersion measure of the population follows below a threshold, DM_{thresh}, which can be calculated by:

$$DM_{thresh} = thresh \times D_{\max} \tag{3.12}$$

where $0 \le thresh \le 1$ is the threshold value.

3.7.2. Discrete/Integer Variables

ACO, DE, GA, PSO and LC differ historically at this point, since ACO was originally introduced to solve combinatorial optimization problems, DE, PSO and LC were first used to deal with continuous problems and, GA was initially designed to solve discrete problems. However, this aspect does not make unfeasible the implementation of each algorithm to solve continuous, discrete or discrete-continuous problems. A simple modification can make the algorithms able to deal with the design variables as integer numbers. The approach is straightforward: the set of discrete variables of each individual is rounded to its closest integer value before evaluating the objective function. This method, although simple, has shown to be quite effective in several problems tested in this research.

3.7.3. Additional Randomness

To avoid premature convergence of the algorithms, additional randomness is introduced using a randomness operator. The randomness operator behaves like the mutation operator in GA. The randomness operator used here changes the vector of design variables that represents an individual of the population.

In the present implementation, the individuals to be modified are identified by using the dispersion measure DM, as in the case of the inertia reduction for the PSO algorithm. If DM falls below a pre-defined threshold value, it is assumed that the population is becoming too uniform. In this case, a random subset of the population is subjected to the randomness operator. As in the case of the GA mutation operator, a randomness operator rate, r_{frac}, controls the maximum size of this subset. Finally, $DM_{thresh} = thresh \times D_{\max}$ is used as threshold value for DM.

Table 3.2 summarizes the different implementations for the randomness operator used in this work.

3.7.4. Stopping Criteria

A convergence criterion is necessary to avoid any additional function evaluations after an optimum solution is found. Ideally, the convergence criterion should not have any problem specific parameters. In this work, the algorithms repeat an iterative loop until at least a stop criterion is achieved. The implemented stopping criteria are shown in Table 3.3.

Table 3.2. Randomness operators.

Name	**Description**
Boundary	It changes randomly one of the parameters of the individual either to its upper or its lower boundary value.
Non-uniform	It changes one of the parameters of the individual by using a Gaussian distribution. This distribution starts wide, and narrows to a point distribution as the current iteration approaches the maximum number of iterations.
Multi-non-uniform	It changes all of the parameters of the individual based on a Gaussian distribution. This distribution starts wide, and narrows to a point distribution as the current iteration approaches the maximum number of iterations.
Uniform	It changes one of the parameters of the individual by using a uniform probability distribution.

Table 3.3. Stopping criteria.

Criterion	**Description**	**Default value**
Iterations	Maximum number of iterations.	100
Function Evaluations	Maximum number of function evaluations.	$n_{pop} \times \text{Iterations}$
Time	Total time (in seconds) allowed for the optimization task.	$+\infty$
Objective limit	The lowest value allowed for the objective function.	$-\infty$
Iterations for convergence	Number of consecutive iterations without improvements allowed for the optimization task.	15
Time for convergence	Time without improvements allowed for the optimization task.	$+\infty$

The SIMPLE Optimization toolbox of Viana and Steffen Jr (2008) is also used for an easy manipulation of all different codes previously presented.

3.8. Enhanced Stochastic Evolutionary Algorithm

The Enhanced Stochastic Evolutionary Algorithm (ESEA) is a modified version introduced by Jin et al. (2005) in the Stochastic Evolutionary Algorithm, initially developed by Saab and Rao (1991). The ESEA, as shown in Figure 3.9, consists of an inner loop and an outer loop, as described in the following. The inner loop constructs new designs by an element-exchange approach and dictates whether to accept the designs based on a certain acceptance criterion dealing with the location of the points. A set of n_D designs is taken by exchanging two elements within the column and then the best of this n_D designs is chosen to be compared with the current best solution. The outer loop controls the entire optimization process by adjusting the threshold value, T_h, in the acceptance criterion of the inner loop. According to Jin et al. (2005), this dynamic adjustment of the acceptance criterion enables the algorithm to avoid local minima designs. During this process, $\mathbf{x}_{best}$ is used to keep track of the best design.

3.8.1. Inner Loop

The inner loop is responsible for constructing new designs by using an element-exchange approach. The new designs can be accepted or rejected, based on an acceptance criterion that deals with the location of the points. As can be seen in Figure 3.9 (b), the inner loop has M iterations. At iteration i, a set of n_D designs is created by exchanging two elements within the column of the current design, $\mathbf{x}$, with the best of these n_D designs, $\mathbf{x}_{trial}$, selected for comparison with the current solution, $\mathbf{x}$. The substitution of $\mathbf{x}$ by $\mathbf{x}_{trial}$ depends on the acceptance criterion given by:

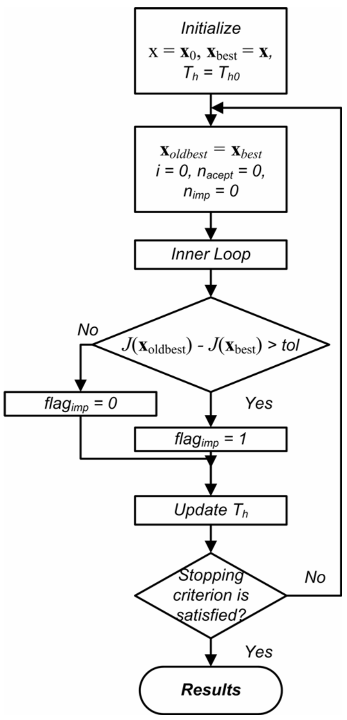
$$\Delta f \leq T_h \times \mathrm{rand}(0,1) \ , \tag{3.13}$$

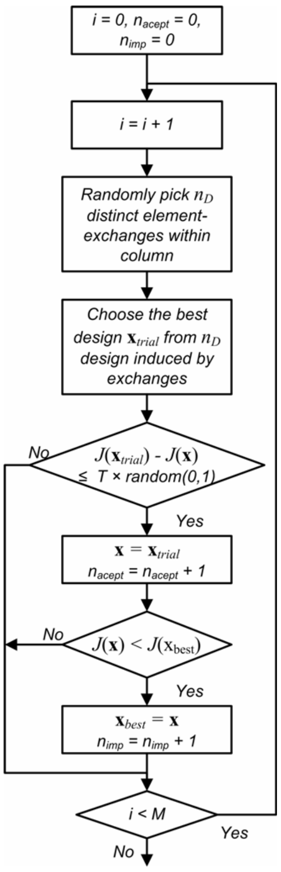
where:

- $\Delta J = J(\mathbf{x}_{trial}) - J(\mathbf{x})$,
- $\mathrm{rand}(0,1)$ is a function that generates uniform random numbers between 0 and 1, and
- $T_h > 0$ is the threshold control parameter.

$\mathbf{x}_{trial}$ will be accepted only if it satisfies $0 \le \Delta J \le T_h$ and the probability of acceptance given by:

$$P\left(S \ge {}^{\Delta J}\!/_{T_h}\right) = 1 - {}^{\Delta J}\!/_{T_h} \quad . \tag{3.14}$$


Initialize
$x = \mathbf{x}_0$, $\mathbf{x}_{best} = \mathbf{x}$,
$T_h = T_{h0}$
$\mathbf{x}_{oldbest} = \mathbf{x}_{best}$
$i = 0$, $n_{acept} = 0$,
$n_{imp} = 0$
Inner Loop
$J(\mathbf{x}_{oldbest}) - J(\mathbf{x}_{best}) > tol$
No
Yes
$flag_{imp} = 0$
$flag_{imp} = 1$
Update T_h
Stopping criterion is satisfied?
No
Yes
Results


$i = 0$, $n_{acept} = 0$,
$n_{imp} = 0$
$i = i + 1$
Randomly pick n_D distinct element-exchanges within column
Choose the best design $\mathbf{x}_{trial}$ from n_D design induced by exchanges
$J(\mathbf{x}_{trial}) - J(\mathbf{x})$
$\le T \times random(0,1)$
No
Yes
$\mathbf{x} = \mathbf{x}_{trial}$
$n_{acept} = n_{acept} + 1$
$J(\mathbf{x}) < J(\mathbf{x}_{best})$
No
Yes
$\mathbf{x}_{best} = \mathbf{x}$
$n_{imp} = n_{imp} + 1$
$i < M$
Yes
No

(a) Outer loop (b) Inner loop

Figure 3.9. Basic scheme for ESEA.

Using the scheme guided by Eqs. (3.13) and (3.14), the algorithm attempts to avoid local solutions. It can be noticed that a temporarily worse design, $\mathbf{x}_{trial}$, could be accepted to replace the current design, $\mathbf{x}$. Table 3.4 gives the values for the inner loop parameters, according to Jin et al. (2005).

3.8.2. Outer Loop

The outer loop controls the entire optimization process by adjusting the threshold value T_h in the acceptance criterion of the inner loop. Initially, T_h is taken as a small fraction of the initial design ($T_h = 0.005 \times J(\mathbf{x}_0)$) and its value is dynamically updated during the search process. The search process is divided in two main stages: (a) the *improving process*, which attempts to find a locally optimal design, and, (b) the *exploration process*, which try to escape from the locally optimal design. These two stages are now discussed in more detail.

Table 3.4. Inner loop parameters.

Parameter	**Value**	**Comments**
n_D	$n_e/5$, but no larger than 50	n_e is the number of all possible distinct $k=2$ element-exchanges in a column. In the Latin Hypercube case $n_e = \dfrac{n_{dv}!}{k!(n_{dv}-k)!}$.
M	$2 \times N \times n_{dv}/J$, but no larger than 100 (if the candidate solution is represented as a vector $N=1$; otherwise, N is the number of rows)	M is the number of iterations in the inner loop

Improving process: The search process switches to the improving process if the objective function is improved after completing an entire inner loop. This mechanism is controlled by $flag_{imp}$, i.e. $flag_{imp} = 1$. Once in the improving process, T_h is adjusted to rapidly find a local optimal design. This is done keeping the value of T_h small, so that only a better, or a slightly worse, design is accepted to replace $\mathbf{x}$. T_h is updated based on the acceptance ratio n_{acpt}/M (number of accepted design divided by the number of tries in the inner loop) and the improvement ratio n_{imp}/M (number of improved design divided by the number of tries in the inner loop). Thus, there are the following possibilities:

1. T_h will decrease if the acceptance ratio is larger than a small percentage (e.g., 10%) and the improvement ratio is less than the acceptance ratio.
2. T_h will remain constant if the acceptance ratio is larger than the small percentage and the improvement ratio is equal to the acceptance ratio (meaning that T_h is so small that only improving designs are accepted by the acceptance criterion).
3. T_h will increase otherwise. The following equations are used to decrease and increase T_h respectively:

$$T_h^{new} = \alpha_1 T_h^{old} \quad , \tag{3.15}$$

$$T_h^{new} = T_h^{old} / \alpha_1 \ , \tag{3.16}$$

where $0 < \alpha_1 < 1$. As in Jin et al. (2005), setting $\alpha_1 = 0.8$ worked well in all tests.

Exploration process: If no improvement is made after completing an entire inner loop, the search process will turn to the exploration process. This mechanism is controlled by $flag_{imp}$, i.e. $flag_{imp} = 0$. During the exploration process, T_h is adjusted to help the algorithm escape from a locally optimal design. Unlike the improving process, T_h fluctuates within a range based on the acceptance ratio. If the acceptance ratio is less than a small percentage (e.g., 10%), T_h will rapidly increase until the acceptance ratio is larger than a large percentage (e.g. 80%). If this happens, T_h will slowly decrease until the acceptance ratio is less than the small percentage. This process will be repeated until an improved design is found.

The following equations are used to decrease and increase T_h, respectively:

$$T_h^{new} = \alpha_2 T_h^{old} \ , \tag{3.17}$$

$$T_h^{new} = T_h^{old} / \alpha_3 \ , \tag{3.18}$$

where $0 < \alpha_3 < \alpha_2 < 1$. As in Jin et al. (2005), $\alpha_2 = 0.8$ and $\alpha_3 = 0.7$ worked well in all tests.

As a result, during the exploration process T_h increases rapidly (so that worse designs could be accepted) to help escaping a local optimal design. T_h decreases slowly for finding better designs after moving away from the local optimal design. For the whole algorithm, the maximum number of cycles is used as the stopping criterion.

ESEA was implemented during an internship of this doctoral research at Vanderplaats Research and Development, Inc. The current version of the VisualDOC general-purpose design optimization software from (VR&D contributors, 2006) uses this module to generate optimum Latin hypercube DOEs. Chapter V shows the results of this collaboration.

To close this chapter, a summary of the contributions to the basic algorithms is given:

- Multiple schemes for DE and GA: the developed computational code allows the use of multiple schemes for both DE and GA. For instance, DE can be set up with a combination of DE/rand/1/bin and DE/best/1/exp; or GA can be set up with two different selection operators (e.g. roulette and tournament).

- Additional randomness for ACO and DE: the general framework as presented by Figure 3.8 makes possible to include additional randomness in these two algorithms, which have limited capabilities of avoiding premature convergence.
- Dispersion measure on the normalized design space: in opposition to schemes based on the function space, the proposed approach prevents difficulties in detecting aggregation of the population when the function space has several minima.
- Implementation of ESEA in an environment of commercial software development to generate the optimal Latin hypercube.

Next chapter presents different approaches on how to couple surrogate modeling and heuristic algorithms during the solution of the optimization problem.

CHAPTER IV

MIXING SURROGATES AND HEURISTIC OPTIMIZATION ALGORITHMS

4.1. Introduction

Chapter I to Chapter III clarified concepts related to the general formulation of the optimization problem, surrogate modeling and heuristic algorithms. The level of accuracy in representing the physical reality was used to define the level of fidelity of the model; the higher the fidelity the more accurate is the model. Chapter III even showed theoretical aspects of some contributions of this research. This is the last theoretical chapter of this thesis; it discusses how to combine surrogate models and heuristic algorithms during the solution of the optimization problem.

As explained in Chapters I and II, surrogate models are used when computational models of high physical fidelity or numerical accuracy are prohibitively expensive in iterative procedures such as in numerical optimization. In this scenario, another alternative is the use of lower-fidelity models, which may not capture a particular feature of the physical phenomenon to the same degree of accuracy as its higher-fidelity counterpart, but may still have satisfactory predictive properties for the purposes of finding a good direction of improvement for the higher-fidelity model. The combination of high-fidelity, low-fidelity and surrogate models determine the model management framework, which aims at maximizing the use of lower-fidelity and surrogate models in iterative procedures with occasional, but systematic, recourse to higher-fidelity, and more expensive models for monitoring the progress of the optimization (Alexandrov et al. 1999, and Vitali et al. 2002).

Starting from the high-fidelity models, three approaches are described in the following sections: (i) surrogates directly replace actual models; (ii) variable-fidelity models, i.e. aiming to optimize high-fidelity but expensive models, the optimization is conducted with cheaper low-fidelity models corrected by surrogates; and (iii) simultaneous use of actual high-fidelity and surrogate models (during the optimization, surrogate models can eventually be updated).

4.2. First Approach: Direct Use of Surrogate Models

This is the simplest case of direct substitution of the high-fidelity models by their surrogate counterparts. When using this approach it pays the use of multiple surrogates instead of a single one for reasons such as the most accurate surrogate does not guarantee the global optimum, and fitting many surrogates and repeating optimizations is cheap as compared to the cost of the simulations. Figure 4.1 shows how this approach works. Most of the steps were already discussed in Chapter I and II, however here both full cycles of surrogate modeling and optimization are combined together. This is not an automatic process since steps involving (i) assessment of quality of fitting of the set of surrogates and selection of a reduced subset; (ii) assessment of the quality of the results; and (iii) redefinition of the design space and sampled points usually requires external intervention. As a consequence, the optimal solution is obtained gradually. At the end of each iteration, candidate solutions can be used in the decision of narrowing the design space, on the re-fitting of the set of surrogates or even as extra points for assessment of the quality of the surrogates.

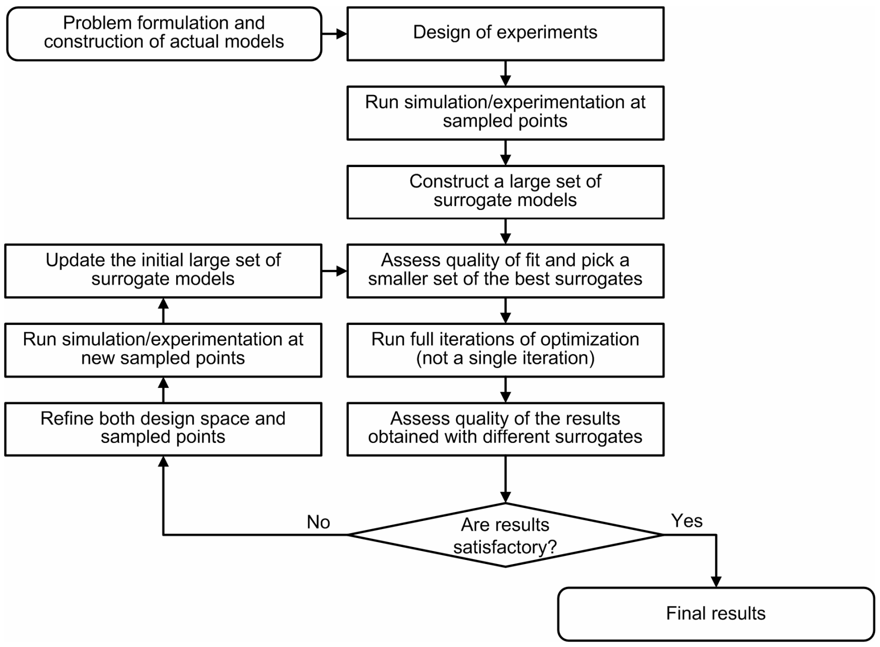

Problem formulation and construction of actual models
Design of experiments
Run simulation/experimentation at sampled points
Construct a large set of surrogate models
Update the initial large set of surrogate models
Assess quality of fit and pick a smaller set of the best surrogates
Run simulation/experimentation at new sampled points
Run full iterations of optimization (not a single iteration)
Refine both design space and sampled points
Assess quality of the results obtained with different surrogates
No
Are results satisfactory?
Yes
Final results

Figure 4.1. Basic scheme for direct use of surrogate models.

If multiple surrogates are used instead of the high-fidelity models, one could ask why not use classical gradient-based algorithms to solve the optimization problem. The answer is that surrogate modeling may also lead to local optima. In a scenario of easy function evaluations, heuristic algorithms may be a more suitable alternative since they present a trend to finding the global, or near global, solution. The bottom line could be the use of both heuristic and gradient-based algorithms in a cascade-type scheme in which the heuristic algorithm generate proper initial guesses for the gradient-based algorithm.

4.3. Second Approach: Use of Different Levels of Fidelity

This is the case in which surrogates are used to map the responses obtained from the low-fidelity to the high-fidelity space (i.e., they correct the responses obtained by using low-fidelity models). This allows the use of faster low-fidelity models during the optimization; while a reduced set of expensive high-fidelity models are run "off-line." A common approach is to use the ratio or difference between the low-fidelity and high-fidelity models at one or several points in order to correct the low-fidelity model at other points (Alexandrov et al. 1999, Vitali et al. 2002, Gano et al. 2004, and Marduel et al. 2006). Figure 4.2 shows a possible flowchart for this approach.

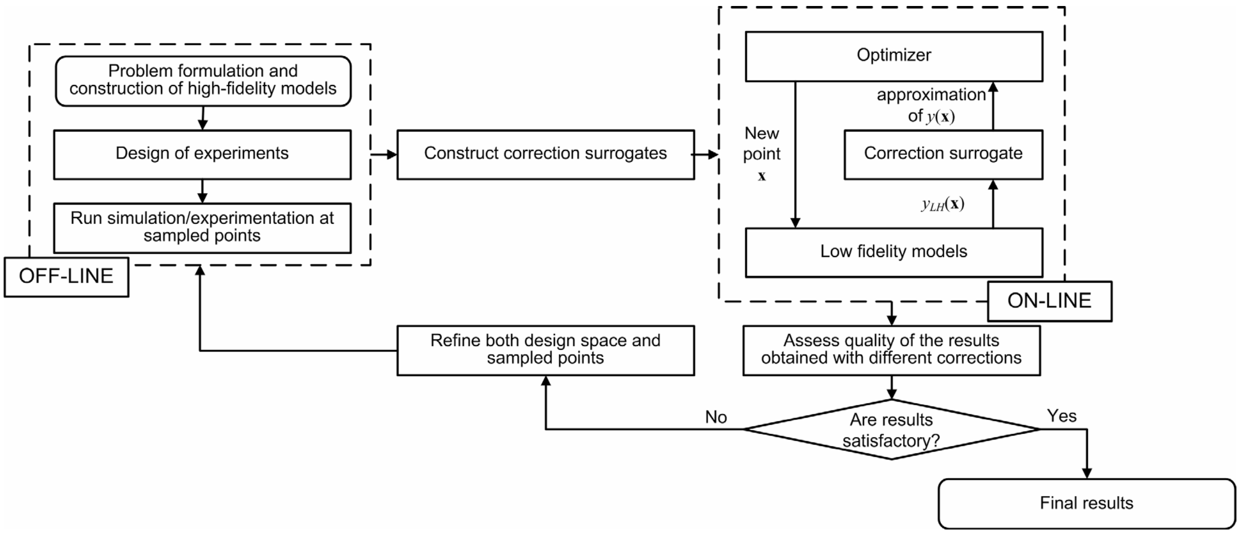

Problem formulation and construction of high-fidelity models
Design of experiments
Run simulation/experimentation at sampled points
OFF-LINE
Construct correction surrogates
Optimizer
approximation of $y(\mathbf{x})$
New point $\mathbf{x}$
Correction surrogate
$y_{LH}(\mathbf{x})$
Low fidelity models
ON-LINE
Refine both design space and sampled points
Assess quality of the results obtained with different corrections
No
Are results satisfactory?
Yes
Final results

Figure 4.2. Basic scheme for variable-fidelity framework.

Correction surrogates couple high-fidelity and low-fidelity methods of analysis at a number of points in the design domain. Here, two ways of coupling the ratio and difference methods of analysis are presented. The process starts with high and low-fidelity computation of the response of interest, y, at different p points. At these points, the ratio β:

$$\beta = \frac{y_{HF}}{y_{LF}} \ , \tag{4.1}$$

and the difference δ:

$$\delta = y_{HF} - y_{LF} \ , \tag{4.2}$$

are computed. The subscripts HF and LF in Eqs. (4.1) and (4.2) indicate the value of y obtained from high-fidelity and low-fidelity models, respectively.

Equation (4.1) may lead to a conditioning problem of β, when y_{LF} tends to zero. Marduel et al. (2006) show that adding a user-defined constant, K, sufficiently large, can prevent ill-conditioning. β is then defined by:

$$\beta = \frac{y_{HF} + K}{y_{LF} + K} \ , \tag{4.3}$$

The next step is to build the surrogate models $\hat{\beta}(\mathbf{x})$ and $\hat{\delta}(\mathbf{x})$ for β and δ, respectively. Here, there is no surrogate model to be preferably used. Instead, as in the previous approach, a set of different surrogates can also be used as a way to prevent against bad fitted surrogates.

The variable-fidelity approximation to y at any other point, $\mathbf{x}$, is obtained from a low-fidelity analysis as:

$$y(\mathbf{x}) \cong y_{\beta}(\mathbf{x}) \quad \text{or} \quad y(\mathbf{x}) \cong y_{\delta}(\mathbf{x}) \ , \tag{4.4}$$

$$\begin{aligned} y_{\beta}(\mathbf{x}) &= \hat{\beta}(\mathbf{x}) y_{LF}(\mathbf{x}) \ , \\ y_{\delta}(\mathbf{x}) &= \hat{\delta}(\mathbf{x}) + y_{LF}(\mathbf{x}) \ . \end{aligned} \tag{4.5}$$

In order to avoid the need to interface the low-fidelity analysis with the optimization software, Vitali et al. (2002) also propose the replacement of the low-fidelity model by its surrogate counterpart. Instead of calling the low-fidelity analysis program every time a value of $y(\mathbf{x})$ is needed, the optimizer calls for the surrogate model. This way:

$$y(\mathbf{x}) \cong \hat{\beta}(\mathbf{x}) \hat{y}_{LF}(\mathbf{x}) \quad \text{or} \quad y(\mathbf{x}) \cong \hat{\delta}(\mathbf{x}) + \hat{y}_{LF}(\mathbf{x}) \ . \tag{4.6}$$

Vitali et al. (2002) go even further by proposing that in addition to those given by Eqs. (4.4) and (4.6), two other approximations of $y(\mathbf{x})$ can be obtained by correcting the low-fidelity values $y_{LF}(\mathbf{x})$ by $\hat{\beta}(\mathbf{x})$ and/or $\hat{\delta}(\mathbf{x})$, then fitting another surrogate of $y_{\beta}(\mathbf{x})$ and/or $y_{\delta}(\mathbf{x})$, i. e.:

$$y(\mathbf{x}) \cong \hat{y}_{\beta}(\mathbf{x}) \quad \text{or} \quad y(\mathbf{x}) \cong \hat{y}_{\delta}(\mathbf{x}) \ , \tag{4.7}$$

At the end, the cost of running $y_{LF}(\mathbf{x})$ may be used to decide whether use computations following Eq. (4.4) or use only Eqs.(4.6) and (4.7). Once again, the use of multiple variable-fidelity approximations to $y(\mathbf{x})$ is encouraged as a way to generate a diverse set of candidate solutions for the optimization problem. In this sense, if Eqs. (4.4) to (4.7) are used, at the end of the optimization a set of candidate solutions must be evaluated, instead of a single one.

4.4. Third Approach: Coupling Actual High-Fidelity and Surrogate Models

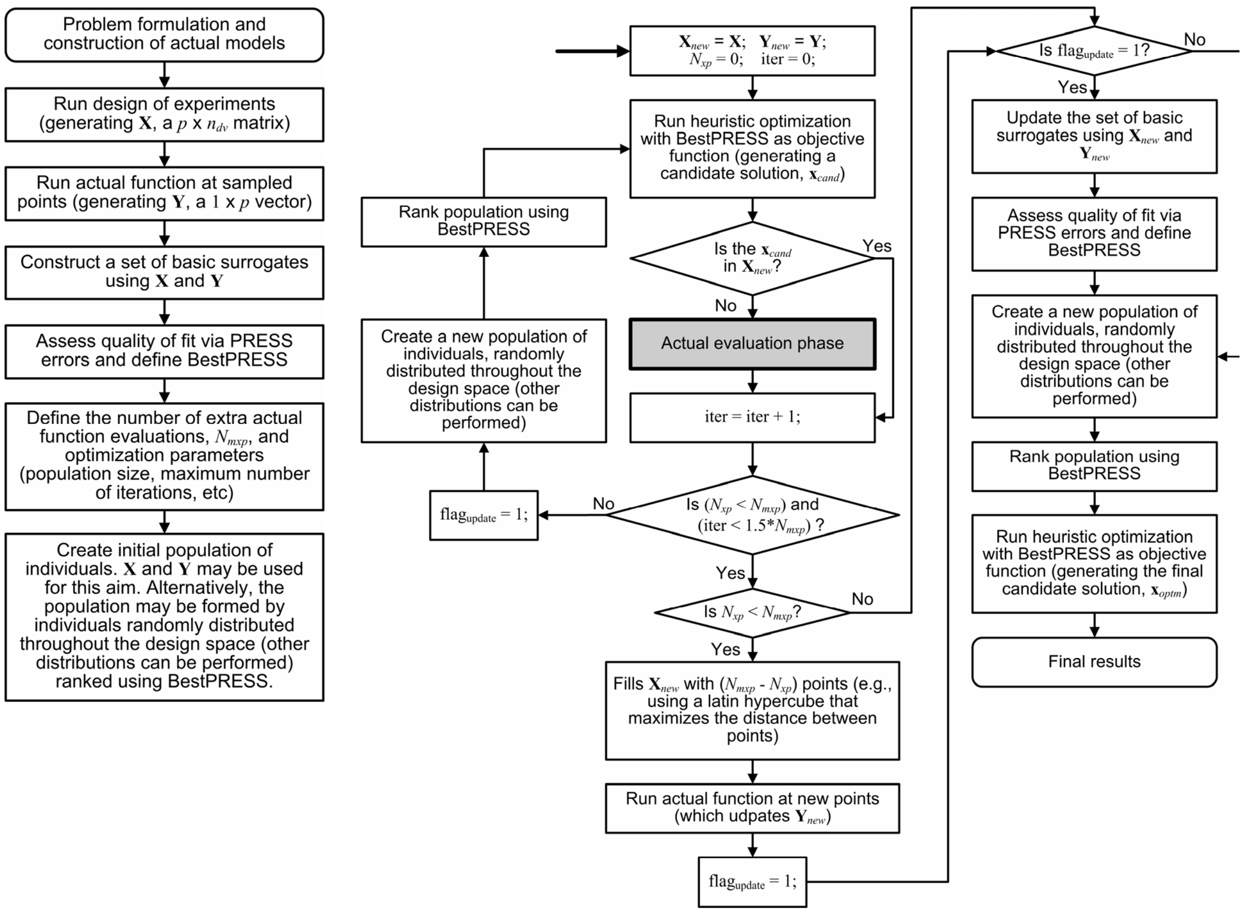
This is the case in which both high-fidelity and surrogate models are used during optimization. High fidelity models can be used both to check the accuracy and to update the surrogate models. There are a number of studies reporting the use of different surrogates and different strategies for coupling them with optimization algorithms. Liang et al. (2000) proposed a strategy for coupling heuristic algorithms with local search and quadratic response surface methods. However, when working with multimodal problems the accuracy of quadratic models can become questionable. Jin et al. (2002) presented a framework for coupling heuristic algorithms and neural-network-based surrogate models. This approach uses both the expensive and approximate models throughout the search, with an empirical criterion to decide the frequency at which each model should be used. Ong et al. (2003) employed both high-fidelity and support vector regression models (with linear-splines as kernel function) during the optimization. They propose an approach in which the local search is conducted with the surrogate models and the global search is performed with the actual high-fidelity models. Zhou et al. (2007) proposed a framework that uses different surrogate models during local and global search. The framework employs a data-parallel Gaussian process based global surrogate model to filter the heuristic algorithm population so that promising individuals are retained. Subsequently, these potential individuals undergo a trust-region gradient-based search strategy that employs radial basis function local surrogate models to accelerate convergence.

While several researchers have primarily been concerned with the choice among different surrogates, there has been virtually no work regarding the use of multiple surrogates. Thus, in this doctoral research, the following aspects are explored:

1. A large set of different surrogates is used during both off-line and on-line phases of the optimization.
2. Update of the surrogates during the on-line phase of the optimization through a pre-defined number of extra actual function evaluations (high-fidelity analysis).

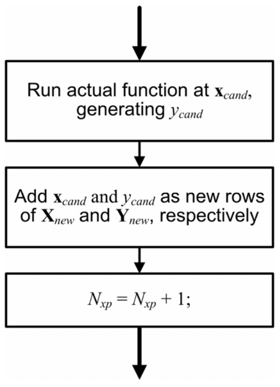
3. $BestPRESS$ (i.e. surrogate with the smallest $PRESS$ value of the set) is used for global search. Since $PRESS$ is used as an estimator of the $RMSE$ (RMS error); the idea is to pick up the surrogate with best global approximation capabilities.

A scheme with two variations is proposed, as seen in Figure 4.3. There is a common off-line phase as well as a common on-line phase (see Figure 4.3-(a) and (b)). During the off-line phase, the basic steps of the surrogate modeling (discussed in Chapter II) are performed. However, instead of the traditional use of a single surrogate model, here a set of different basic surrogates are generated. In the sequence, parameters for the optimization are defined. Other than parameters of the optimization method (such as population size, number of iterations, etc.), the number of extra actual function evaluations, N_{mxp}, must be defined. N_{mxp} acts like the computational budget used during the on-life phase for updating the surrogates. Using $BestPRESS$ as objective function, heuristic optimization algorithms are in charge of generating candidate solutions considered to be added the set of initial data points. If a candidate solution is not in the design matrix already, the high-fidelity analysis is performed. It is worth remembering that especially with discrete variables, different runs of the heuristic algorithm may lead to the same candidate solution. Thus, it is suggested that this loop run for up to $1.5 \times N_{mxp}$ iterations, instead of just for N_{mxp}. The two proposed variations differ on the actual evaluation phase. Figure 4.4-(a) illustrates the first and simplest approach, where the candidate solutions are added to the design matrix, $\mathbf{X}$ (and its respective actual function value to the $\mathbf{Y}$ vector). At the end of the loop, the resulting design matrix, $\mathbf{X}_{new}$ (and the respective $\mathbf{Y}_{new}$) is used for updating the set of basic surrogates. The update of the surrogates is performed just once, right before the final run of the heuristic algorithm, which finally generates the final candidate solution, $\mathbf{x}_{optm}$. The second approach updates the surrogates every time a new point is added to the design matrix, as shown in Figure 4.4-(b). It means that $BestPRESS$, used as objective function, may change in both shape and identity. It is easy to see that while the first approach is faster, the second one may benefit from the constant update of the surrogates. The downside of the second approach is the iterative computation of $PRESS$ errors. In high dimensions, this may be an issue and the use of alternative $PRESS$ approaches as the k -fold strategy may be considered (as discussed in Chapter II). In this case, instead of computing $PRESS$ errors through the leave-one-out strategy every time a new point is added to the design matrix, $\mathbf{X}_{new}$, $PRESS$ is performed using the k -fold strategy with pre-defined number of points and respective values of k. As an illustration of this question, consider the case of 12-point initial design matrix in a 2-dimensional space, and $N_{mxp} = 6$. Most probably, there are no computational issues in performing $PRESS$ calculations when the number of points in $\mathbf{X}_{new}$ is equal to 13, 14, 15, 16, 17, and 18. On the other hand, consider a 56-point initial design matrix in a 6-dimensional space, and $N_{mxp} = 14$. Here, the computation of $PRESS$ may be expensive and instead of employing the leave-one-out strategy, the k-fold strategy may be used when, for example, the number of points in $\mathbf{X}_{new}$ reaches 60, 63, 66, and 68, with k equals to 20, 21, 22, and 17; respectively. For the sake of simplicity, the flowchart of Figure 4.3-(b) and Figure 4.4-(b) show just the leave-one-out implementation (the k-fold variation is left as an exercise for the reader). The SIMPLE Optimization toolbox of Viana and Steffen Jr (2008) has all previously discussed functionalities already implemented.

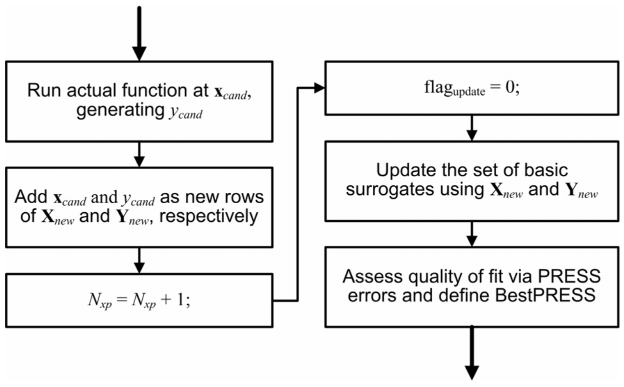

Problem formulation and construction of actual models
Run design of experiments (generating X, a p x n_dv matrix)
Run actual function at sampled points (generating Y, a 1 x p vector)
Construct a set of basic surrogates using X and Y
Assess quality of fit via PRESS errors and define BestPRESS
Define the number of extra actual function evaluations, N_mxp, and optimization parameters (population size, maximum number of iterations, etc)
Create initial population of individuals. X and Y may be used for this aim. Alternatively, the population may be formed by individuals randomly distributed throughout the design space (other distributions can be performed) ranked using BestPRESS.
X_new = X; Y_new = Y; N_xp = 0; iter = 0;
Run heuristic optimization with BestPRESS as objective function (generating a candidate solution, x_cand)
Is the x_cand in X_new?
Yes
No
Actual evaluation phase
iter = iter + 1;
Is (N_xp < N_mxp) and (iter < 1.5*N_mxp) ?
No
flag_update = 1;
Create a new population of individuals, randomly distributed throughout the design space (other distributions can be performed)
Rank population using BestPRESS
Yes
Is N_xp < N_mxp?
No
Yes
Fills X_new with (N_mxp - N_xp) points (e.g., using a latin hypercube that maximizes the distance between points)
Run actual function at new points (which udpates Y_new)
flag_update = 1;
Is flag_update = 1?
No
Yes
Update the set of basic surrogates using X_new and Y_new
Assess quality of fit via PRESS errors and define BestPRESS
Create a new population of individuals, randomly distributed throughout the design space (other distributions can be performed)
Rank population using BestPRESS
Run heuristic optimization with BestPRESS as objective function (generating the final candidate solution, x_optm)
Final results

(a) Off-line phase. (b) On-line phase.

Figure 4.3. Basic scheme for the coupling of actual high-fidelity and surrogate models.


Run actual function at $\mathbf{x}_{cand}$, generating y_{cand}
Add $\mathbf{x}_{cand}$ and y_{cand} as new rows of $\mathbf{X}_{new}$ and $\mathbf{Y}_{new}$, respectively
$N_{xp} = N_{xp} + 1;$

(a) Without update of surrogates.


Run actual function at $\mathbf{x}_{cand}$, generating y_{cand}
Add $\mathbf{x}_{cand}$ and y_{cand} as new rows of $\mathbf{X}_{new}$ and $\mathbf{Y}_{new}$, respectively
$N_{xp} = N_{xp} + 1;$
$\text{flag}_{\text{update}} = 0;$
Update the set of basic surrogates using $\mathbf{X}_{new}$ and $\mathbf{Y}_{new}$
Assess quality of fit via PRESS errors and define BestPRESS

(b) With update of surrogates.

Figure 4.4. Different alternatives for the “actual evaluation phase” of Figure 4.3-(b).

CHAPTER V

APPLICATIONS

The previous chapters showed the basis and theoretical contributions of this doctoral research. This chapter is devoted to the numerical applications. Ant Colony Optimization, Differential Evolution, Genetic Algorithms, Particle Swarm Optimization and LifeCycle Model were employed to solve the following list of direct/inverse optimization problems:

- Direct problems: (a) Vehicular three-dimensional structure design optimization; and (b) Optimization of aircraft structural components by using heuristic algorithms and multi-fidelity approximations.
- Coupling heuristic optimization and high-fidelity analysis to improve surrogate models in the region of the optima.
- Inverse problems: (a) Identification of a non-linear landing gear model using heuristic optimization; and (b) Aircraft longitudinal stability and control derivatives identification by using LifeCycle and a local optimization algorithm.
- Enhanced Stochastic Evolutionary Algorithm used to solve the combinatorial optimization of the optimal Latin hypercube design of experiments.

5.1. Direct Problems

5.1.1. Three-dimensional Vehicular Structure Design Optimization

This application explores discrete-continuous, multi-objective and constrained optimization using heuristic optimization methods. It consists in the design optimization of a vehicular three-dimensional structure (space frame type) by means of a set of nature-inspired optimization algorithms, namely ACO, GA, and PSO. The structure is used as a chassis of a high performance roadster vehicle. In general, for this type of vehicle, a chassis must be as rigid and lightweight as possible in order to fulfill drivability (the degree of smoothness and steadiness of acceleration of a car) and maneuverability (the quality of being capable of maneuvering or changing position) requirements. This application was fully reported in Viana et al. (2007b) and Oliveira et al. (2007).

1. Baseline Design Definition and Modeling

The use of three-dimensional vehicular structures became popular since 1960s, due to the interest in designing high performance race cars (Aird, 1997). Figure 5.1 shows one of the most common structures for this type of chassis.

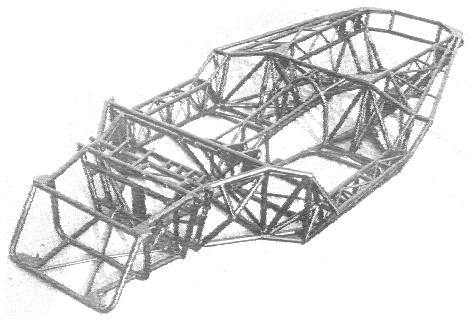


Figure 5.1. Space frame chassis in three-dimensional arrangement.

According to the literature (Thompson et al., 1998 and Aird, 1997), for design purposes the most effective strategy should consider both the mass reduction and the increase of the torsional stiffness. Translating this to the design level, the engineer must choose the material (by defining the material density, Poisson's ratio, and the Young's modulus) and design the proper geometry of the structure (by defining the total mass and moment of inertia). Even though steel alloy and aluminum alloy are the most attractive materials for the chassis construction, 1020 carbon steel was employed here as a low cost solution. Figure 5.2 illustrates the baseline design for a two-passenger vehicle as obtained by consulting the literature about high performance roadster vehicles (Thompson et al., 1998; and Aird, 1997). The lateral view shows indications of the suspensions anchorage points. Since lack of roof and the presence of side doors can make this structure flexible, left and right trusses were placed together with diagonal bars in the floor-plan (minimizing buckling tendencies). Table 5.1 gives more details about this initial design configuration. Regarding the finite element modeling of the structure, while the use of more sophisticated elements may be more accurate, they are also more expensive from the computational viewpoint, and thus, not convenient within the optimization framework. In this case study, bar elements were used as a compromise solution between accuracy and computational cost. Each element has twelve degrees of freedom (two nodes per element): translations along x, y and z nodal directions and rotations with respect to x, y and z nodal axes. The baseline design has $19{,}167$ degrees of freedom, mass of $139.52\ [Kg]$ and torsional stiffness of $1728.1\ \left[Kgf \cdot m/\deg\right]$.

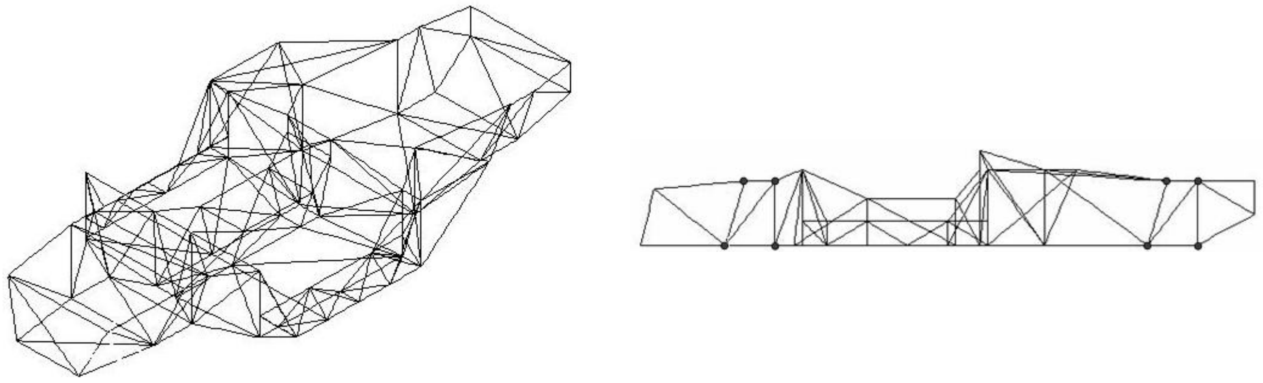


Figure 5.2. Baseline design of the space-frame chassis (lateral view indicates points of suspensions anchorage).

Table 5.1. Baseline details.

General		
Mass: $139.52\ Kg$	**Torsional stiffness:** $1{,}728.1\ \ Kgf \cdot m/\deg$	**Initial geometry:** circular tubes with $50.8 \times 10^{-3}\ [m]$ of external diameter and $1 \times 10^{-3}\ [m]$ of wall thickness
Wheelbase: $2.62\ m$	**Front and rear tracks:** $1.475\ m$	
Front suspension type: short arm	**Rear suspension type:** MacPherson with aggregate	
Material (carbon steel)		
Young's modulus: $2.1 \times 10^{11}\ \ \mathrm{N}/m^2$	**Poisson ratio:** 0.3	**Density:** $7850\ \ Kg/m^3$
FE model		
Element type: BEAM04	**NDOF per element:** 12	**NDOF of the model:** $18{,}675$

The torsional stiffness is numerically obtained through a static analysis of the structure. Figure 5.3 illustrates the boundary conditions for this aim. The torsional stiffness is then obtained by using the information about the stress and strain, as detailed in Aird (1997). The commercial finite element package ANSYS (Ansys Contributors, 2006) is used to run the finite element analysis. More details about the torsional stiffness computation for this specific application can be found in Oliveira (2007).

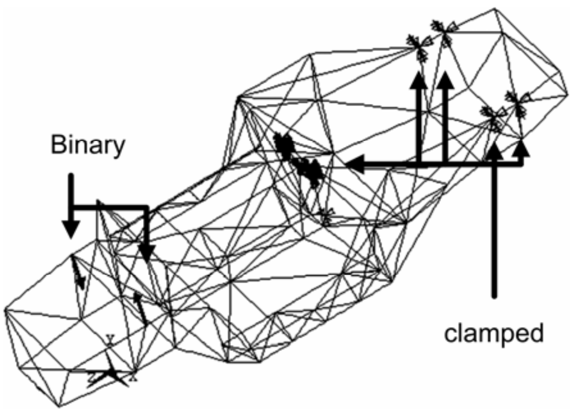


Figure 5.3. Boundary conditions for computing torsional stiffness.

2. Optimization Problem Formulation

Figure 5.4 shows a possible way of incorporating numerical optimization in the design cycle. Modeling of the structure and analysis of design requirements are steps of the “problem definition and formulation.” After this first stage, a baseline design is proposed. After each “optimization” step, the candidate solution is revised. If either the constraints were not satisfied or it is possible to achieve some extra improvement, the candidate solution is redefined as a baseline for a new iteration of the design cycle. This loop continues until the candidate solution satisfies the proposed goals. As a result of the optimization, it may be observed not only changes in the design in the sense of the nodal coordinates, but also changes in the finite element model. A candidate solution may lead the designer to include or eliminate elements in the structure.

The formulated optimization problem deals with two responses obtained from the finite element model, namely, the mass and the torsional stiffness (in a more realistic design scenario, it would be necessary to include other factors such as dynamic responses, manufacture constraints, etc.). A total number of 32 nodal coordinates on the central part of the chassis are assumed as continuous design variables and the external diameter of the tube is a discrete variable. The baseline design is placed at the center of the design space in such a way that and each node can be shifted to a position that has coordinates varying from 75% to 125% of the baseline counterparts. Figure 5.5 illustrates where the 32 nodal coordinates take place in the baseline design. The external diameter of the tube can assume values shown in Table 5.2. For all these diameter values, the thickness ranges from $1.06\times10^{-3}m$ to $3.00\times10^{-3}m$ (all values were obtained from commercial steel tubes found in the market). This way, the optimization is defined as a problem of 33 design variables (32 continuous and 1 discrete).

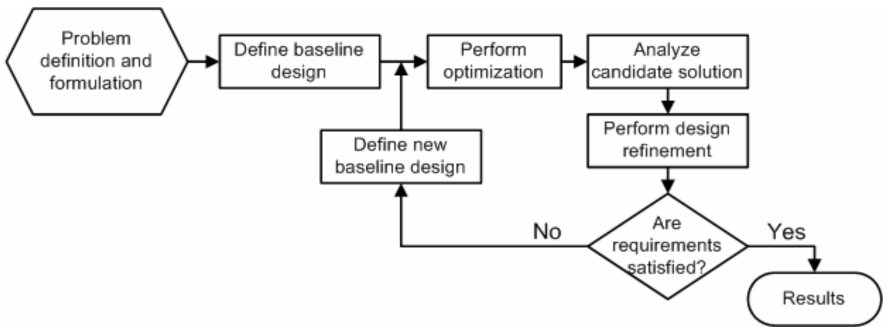

Problem definition and formulation
Define baseline design
Perform optimization
Analyze candidate solution
Perform design refinement
Define new baseline design
Are requirements satisfied?
No
Yes
Results

Figure 5.4. Optimization included in the design cycle.

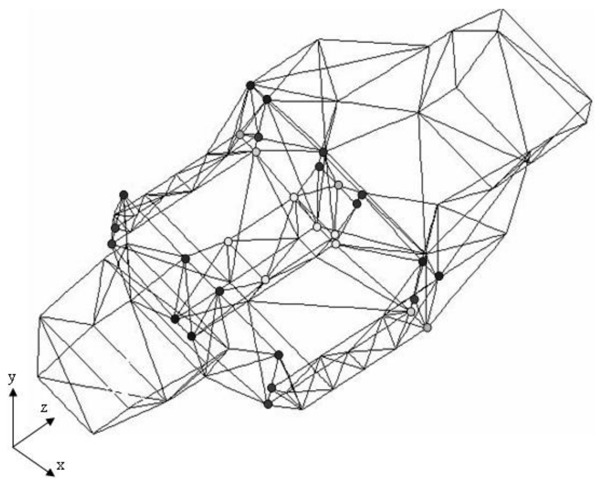


Figure 5.5. Nodal points for optimization.

Table 5.2. Possible values of the external diameter.

Design variable values	1	2	3	4	5
Corresponding diameter $\left[\times 10^{-3} m\right]$	38.10	41.27	44.45	47.60	50.80

Four different formulations were tested in the first “optimization” step of the design cycle. Table 5.3 gives the description of each of the four formulations. Static penalties and weighted sum methods were used to handle the constrained optimization, and the multi-objective problems, respectively.

Table 5.3. Formulation of the optimization problem.

Scenario	Description	Formulation
$\#1$	Optimization problem is defined as the minimization of a dummy objective function, while both the mass and the stiffness are considered as design constraints.	Objective: $J(\mathbf{x}) = 0$ Mass constraint: $G_1(\mathbf{x}) = \frac{M}{M_{limit}} - 1.0 \leq 0$ Stiffness constraint: $G_2(\mathbf{x}) = 1.0 - \frac{K}{K_{limit}} \leq 0$
$\#2$	Mass reduction is an objective function while stiffness increase is taken as a constraint.	Mass objective: $F(\mathbf{x}) = \frac{M}{M_{limit}}$ Stiffness constraint: $G(\mathbf{x}) = 1.0 - \frac{K}{K_{limit}} \leq 0$
$\#3$	Stiffness increase is an objective while mass reduction is a constraint.	Stiffness objective: $F(\mathbf{x}) = -\frac{K}{K_{limit}}$ Mass constraint: $G(\mathbf{x}) = \frac{M}{M_{limit}} - 1.0 \leq 0$
$\#4$	Multi-objective optimization: mass reduction and stiffness increase.	Mass objective: $F_1(\mathbf{x}) = \frac{M}{M_{limit}}$ Stiffness objective: $F_2(\mathbf{x}) = -\frac{K}{K_{limit}}$

3. Results and Discussion

Table 5.4 contains the optimal results obtained in the first “optimization” step, in which all heuristic methods were set up with a population size of 50 and number of iterations of 50, $M_{limit} = 100\ [Kg]$ and $K_{limit} = 1900\ \left[Kgf \cdot m/\deg\right]$. According to Table 5.4, although all algorithms performed well, considering both mass and torsional stiffness, PSO using formulation $\#1$ showed to be the best choice. This way, this is the only configuration for the “optimization” step in a second iteration of the design cycle. Table 5.5 summarizes all information about the only two cycles required to achieve an optimal design. In the second iteration, PSO was set up with a population size of 100 and number of iterations of 100, $M_{limit} = 80\ [Kg]$ and $K_{limit} = 2835\ \left[Kgf \cdot m/\deg\right]$.

The framework illustrated in Figure 5.4 showed to be capable of improving the baseline design in every cycle of optimization.

Table 5.4. Optimal results for the first step.

Method	Formulation	Mass		Torsional stiffness	
		Value $[Kg]$	**%of reduction**	**Value** $\left[Kgf \cdot m/\mathrm{deg}\right]$	**% of increasing**
ACO	#1	107.24	23.1	1897.7	9.8
	#2	115.93	16.9	1937.8	12.1
	#3	100	28.3	1765.08	2.1
	#4	105.31	24.5	1937.8	12.1
GA	#1	107.96	22.6	1774.54	2.7
	#2	126.34	9.45	1937.8	12.1
	#3	100.03	28.3	1809.56	4.7
	#4	100	28.3	1937.8	12.1
PSO	#1	**106.08**	**24**	**2106.25**	**21.9**
	#2	110.01	21.2	1937.8	12.1
	#3	108.55	22.2	1933.4	11.9
	#4	105.63	24.3	1868	8.1

Table 5.5. Summary of the optimization process.

Design cycle iteration	**Designs**	**Mass** $[Kg]$	**Torsional stiffness** $\left[Kgf \cdot m/\mathrm{deg}\right]$	**DOF of the model**
$\#1$	Baseline	139.5	1728.1	19167
	Candidate solution	106.1	2106.3	18609
$\#2$	New baseline	105.7	2266	18675
	New candidate solution	109	2880	18333
Final design refinement	Optimal	104	2810	17643
% Improvement (reduction for mass and increasing for stiffness)		25.5	62.6	---

4. Summary and Conclusions

This application demonstrated the use of heuristic optimization algorithms directly coupled with finite element models to solve the design optimization of a three-dimensional vehicular structure. The investigation showed the possibility of using the present techniques in real world environment. A combination of continuous and discrete design variables was successfully handled by the algorithms. It is also important to say that the algorithms were able to handle constraint functions and performed satisfactorily in the context of multi-criteria optimization design. The results are very encouraging in the sense that other system requirements, embracing for example the gravity center, dynamic and manufacture constraints, will be included in the optimization problem in further research work.

5.1.1. Optimization of Aircraft Structural Components by using Heuristic Algorithms and Multi-Fidelity Approximations

This application explores the use of the multi-fidelity framework coupled with heuristic optimization methods in order to reduce the computational cost in the on-line phase of the optimization. In this work, a suite of nature-inspired optimization methods, namely ACO, GA, PSO, and LC, is applied in the design of a flat pressure bulkhead reinforced by an array of beams. Variable fidelity is proposed as compromise solution between numerical performance and computational cost. This allows faster linear analyses during the optimization; whilst a reduced set of expensive non-linear analyses are run "off-line", enhancing the linear results according to the physics of the structure. Preliminary results of this application was reported in Viana et al. (2007d), the full report is in Viana et al. (2008d).

1. Component Description and Formulation

From the structural viewpoint, the pressurized cabin of a modern aircraft is a system of sealed pressure vessels containing an atmosphere near to that of sea level and its functional requirements include (Bruhn, 1973): (i) transmission of internal and external flight loads; (ii) necessity for non-structural cutouts such as doors and windows; (iii) shape efficiency for both aerodynamics and space allocation; and (iv) minimum structural weight. More specifically, devices called pressure bulkheads close the extremities of pressurized cabins. According to Niu and Niu (1999), pressure bulkheads should have a dome-like structure in preference to a flat one. It is also known, however, that some requirements, mostly related to space availability, impose the adoption of the flat geometry. This case study is although geared towards the non-ideal condition, which is a flat bulkhead. For a more realistic analysis, it should be considered that when the carrying of transverse loads entails large (more than a few tenths of the plate thickness) deflections in the same direction (see Figure 5.1), the plates deform into curved surfaces, giving rise to mid-surface stresses. Still, the edge supports resist the in-plane movements of the plates, leading to further membrane (mid-surface) action.

From the analysis point of view, all previous discussion justifies the use of non-linear finite element modeling to deal with such complex calculations. On the other hand, within a design optimization framework, several analyses are required iteratively and, due to their usually high computational cost, a feasibility obstacle can arise. In this case study, variable fidelity is used to overcome this issue. Faster linear analyses are employed during the optimization; and a reduced set of expensive non-linear analyses are run "off-line."

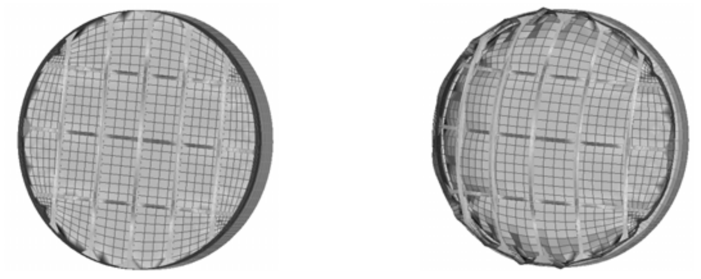


Figure 5.1. Displacement post-processing indicating large deflections at a pressurized flat bulkhead reinforced by beams.

2. Numerical Optimization of the Pressure Bulkhead

The optimization framework based on heuristic methods and on variable fidelity analyses, as described in Chapter IV, is used here. Table 5.1 to Table 5.3 contain useful design data, such as the general information about the pressure bulkhead; design variable descriptions and their respective boundary specifications; and finally, response descriptions and respective values for the baseline design and target design as well. All this information is important for the good understanding and definition of the optimization problem. It is worth mentioning that in Table 5.2, the normalization was performed to preserve interests in the technological content of this work.

The sets of four design variables and five responses that describe the pressure bulkhead modeling are listed in Table 5.2 and Table 5.3, respectively. The target values were chosen to be 80% of those at the baseline design (since all of them are quantities to be minimized). The bulkhead mass is set to be 2.5 times more important than the other responses (i.e. the weight coefficients are $w_i = 1$, for $i = 1 \text{ to } 4$ and $w_5 = 2.5$). The baseline responses are chosen as the worst accepted values, thus forcing the optimizer to move away from them in a clear attempt to improve the design.

Table 5.1. Pressure bulkhead description.

Web material	Al 2024
Number of horizontal reinforcement beams	3
Number of vertical reinforcement beams	5
Reinforcement beams material	Al 7050
Pressure differential	Equivalent to 30,000 ft

Table 5.2. Design variables values normalized by the bulkhead diameter (normalization makes the design variables non-dimensional).

Design variable	Lower bound	Baseline	Upper bound
x_1 : width of reinforcement beams	0.01	0.02	0.03
x_2 : height of reinforcement beams	0.040	0.05	0.06
x_3 : thickness of reinforcement beams	0.0008	0.0012	0.0016
x_4 : web thickness	0.0008	0.0012	0.0016

Table 5.3. Baseline design and target design response values.

Objectives	Baseline value	Target value
$f_1(\mathbf{x})$: ratio of ultimate tension stress at the reinforcement beams with respect to allowable	1.01	0.8
$f_2(\mathbf{x})$: ratio of ultimate compression stress at the reinforcement beams with respect to allowable	0.85	0.68
$f_3(\mathbf{x})$: web maximum displacement, normalized with respect to the bulkhead diameter	0.116	0.928
$f_4(\mathbf{x})$: maximum limit tension stress with respect to yield limit	0.19	0.15
$f_5(\mathbf{x})$: pressure bulkhead mass	22.56	18.05

For the sake of simplicity, the variable fidelity is implemented with a second order PRS model for the difference between the high and low fidelity analyses of responses $f_1(\mathbf{x})$ to $f_4(\mathbf{x})$. Since $f_5(\mathbf{x})$ is the pressure bulkhead mass, it does not require any correction (its fidelity does not depend on whether the calculations are linear or non-linear; eventually, small differences may appear due to either rounding or truncation).

In order to keep the computational cost low (in terms of low-fidelity simulations), instead of a multi-objective approach for generating the Pareto front, the responses are combined into a functional whose minimization implies that all responses tend to a target value and depart from a non-desirable one. This simplifies the solution of the problem and can be considered acceptable given the general scope of this work. *Compromise Programming* (discussed in Chapter II) was used:

$$J(\mathbf{x}) = \sqrt{\sum_{i=1}^{n_{obj}} \left(w_i \frac{\left(f_i(\mathbf{x}) - f_i^*(\mathbf{x})\right)}{f_i^{worst}(\mathbf{x}) - f_i^*(\mathbf{x})} \right)^2} \tag{0.1}$$

where:

- $J(\mathbf{x})$ is a compromise objective function.
- $f_i(\mathbf{x})$ is the *i-th* response of interest, in a total of n_{obj}.
- $f_i^*(\mathbf{x})$ is the target value of the *i-th* response.
- $f_i^{worst}(\mathbf{x})$ is the worst (avoidable) value accepted for the *i-th* response.
- w_i is the weighting factor applied for the *i-th* response of interest.

The target and avoidable values are not necessarily design goals, but play an important role at the optimization problem by defining the tendency of the desired optima with respect to the baseline design. In the context of the present work, it is a useful approach since the heuristic techniques are defined for unconstrained and single-objective problems only. During the optimization task, each of the responses is calculated through a linear static finite element analysis (low-fidelity) and corrected by the PRS that correlate the result with the one obtained by means of non-linear finite element analysis (high-fidelity).

The samples for the regression are generated through a Central Composite Design CCD, as described in Box et al. (1978), is one of the most commonly used DOEs for quadratic polynomial response surfaces. CCD consists of three distinct sets of experimental runs: (1) a factorial (perhaps fractional) design in the factors studied, each having two levels; (2) the central point, whose values of each design variable are the medians of the design space; and (3) a set of axial points, experimental runs identical to the centre points except for one design variable, which will take on values both below and above the median of the design space, and typically both outside their range (all variables are varied in this way).In this application, CCD demanded 25 finite element simulations (both linear and non-linear). Following the idea presented in Chapter IV, most of the off-line computational effort is due to these runs, apart from isolated validations. According to this procedure, the obtained PRS are shown in the following equations:

$$\begin{aligned}\hat{\delta}_1(x) = -3.07513 + 1.57573x_1 + 0.75861x_2 + 0.97382x_3 + 0.23060x_4 \\ -0.41144x_1^2 - 0.11233x_2^2 - 0.23423x_3^2 + \\ -0.18688x_1x_2 - 0.32223x_1x_3 - 0.23890x_1x_4 - 0.14728x_2x_3\end{aligned} \tag{0.2}$$

$$\begin{aligned}\hat{\delta}_2(x) = -5.09962 + 1.68555x_1 + 1.33653x_2 + 0.96027x_3 + \\ -0.63840x_3^2 - 0.29912x_1x_2\end{aligned} \tag{0.3}$$

$$\hat{\delta}_3(x) = -7.77607 + 0.41315x_2 + 5.39596x_4 - 2.36998x_4^2 \tag{0.4}$$

$$\hat{\delta}_4(x) = -18.5759 - 2.6548x_1 - 1.2855x_2 - 2.4576x_3 + 6.3808x_4 + \\ -1.2053x_4^2 - 1.0776x_1x_3 \tag{0.5}$$

Table 5.4 gives the comparison of the adjusted correlation coefficients, R_a^2, of the PRS models for the responses, $\hat{f}(\mathbf{x})$, and for the correction surrogates, $\hat{\delta}(\mathbf{x})$. R_a^2 primarily measures the statistical accuracy of the PRS surrogates (the closer R_a^2 is to 1, the better). Mathematically:

$$R_a^2 = 1 - \frac{\sum_{i=1}^{p}(y_i - \hat{y}_i)^2}{(n_{dv} - p - 1)} \frac{(n_{dv} - 1)}{\sum_{i=1}^{p}(y_i - \overline{y})^2} \quad , \tag{0.6}$$

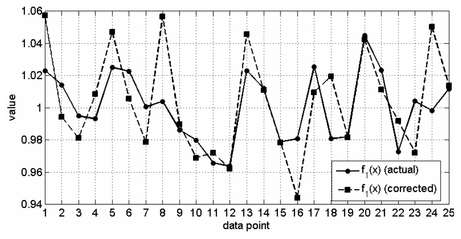
where p is the number of data points, y_i is the observed data, $\hat{y}_i$ is the prediction of the PRS model, and $\overline{y}$ is the mean of the observed data.

The success in using variable-fidelity over the direct meta-modeling of the responses can be observed in Table 5.4.

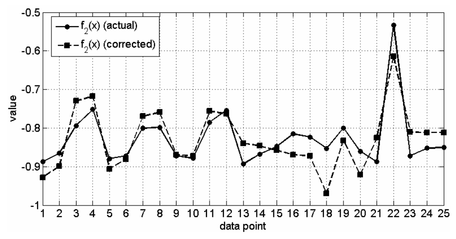
Table 5.4. Comparison of the adjusted correlation coefficients.

Surrogate of the response	$\hat{f}_1(\mathbf{x})$	$\hat{f}_2(\mathbf{x})$	$\hat{f}_3(\mathbf{x})$	$\hat{f}_4(\mathbf{x})$
	0.65	0.34	0.93	0.87
Correction surrogate	$\hat{\delta}_1(\mathbf{x})$	$\hat{\delta}_2(\mathbf{x})$	$\hat{\delta}_3(\mathbf{x})$	$\hat{\delta}_4(\mathbf{x})$
	0.99	0.96	0.982	0.952

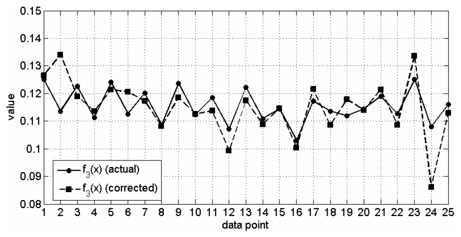
As expected from the theoretical analysis, it can be observed that (i) due to the non-linearities, it pays to use the variable-fidelity framework over the PRS models of the responses; and (ii) the correction PRS models adjust very well to the data. As a consequence, the corrected response values (low-fidelity analyses and correction PRS surrogates) and the actual non-linear results (high-fidelity analyses) have very close variation in the case of all four responses of interest, as shown in Figure 5.2. As a final verification, the data corresponding to the 25 runs of the CCD was used to calculate two sets of correlation coefficients, as shown in Table 5.5. All sets of data move in the same direction (positive correlations) and, in general, the correlation is greater when the correction surrogates are used with the linear results. Given the improvement on the predicting capabilities of the low-fidelity models due to the use of the correction surrogates adopted on the variable fidelity framework, it is now safe to perform the intended optimization procedures.



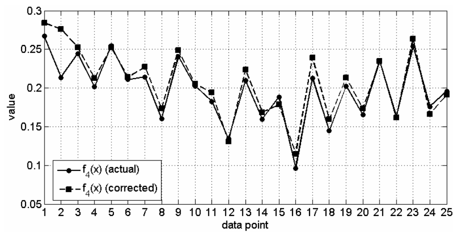
(a) Ratio of maximum tension stress at the reinforcement beams with respect to allowable one.



(b) Ratio of minimum compression stress at the reinforcement beams with respect to allowable one.



(c) Web maximum misplacement, normalized with respect to the bulkhead diameter.



(d) Maximum limit tension stress with respect to yield limit.

Figure 5.2. Variation of actual and corrected values of responses $f_1(\mathbf{x})$ to $f_4(\mathbf{x})$.

Table 5.5. Correlation coefficients of non-corrected and corrected linear results with respect to actual non-linear results.

Response	**Uncorrected linear to non-linear correlation coefficients**	**Corrected linear to non-linear correlation coefficients**
$f_1(\mathbf{x})$	51,32%	66,64%
$f_2(\mathbf{x})$	28,94%	79,28%
$f_3(\mathbf{x})$	70,09%	87,86%
$f_4(\mathbf{x})$	61,26%	94,62%

One of the aims of this work is to show the performance of the heuristic algorithms when combined with variable fidelity in a real world problem. Given the random nature of the algorithms used, results may vary from one run to another (or at least present slight differences). This way, in the first part of the study, each algorithm is tested 10 times (each time with different initial populations). The success of the algorithms in performing the initial global search is observed if the results of these repetitions are close to each other. Table 5.6 shows the setup used for each of the heuristic optimization algorithms. In all cases, each run of the algorithms will demand 1,500 low-fidelity function evaluations.

Secondly, a set of initial designs given by the heuristic methods fed the Nelder-Mead simplex direct search (NMSDS, detailed in Appendix D) in order to finish the local search.

NMSDS was chosen because it is a zero-order method as well and thus it has no privilege of information when compared with the heuristic optimization algorithms. Finally, one of the final designs is used to validate the results of the optimization performed with variable fidelity analysis. This is done by checking the differences between the uncorrected and corrected linear (low-fidelity) simulations with respect to the non-linear one (high-fidelity).

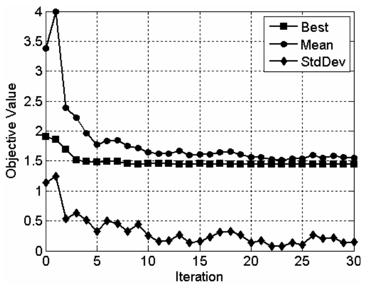
Table 5.6. Setup for heuristic optimization methods.

General	Population size	50
	Iterations	30
	Stop criterion	Maximum. number of iterations
ACO	Dissolving rate	1.25
GA	Number of individuals for elitism	2
	Selection function	simpleGASelectionRoulette
	Crossover function	simpleGACrossoverHeuristic
	Crossover fraction	0.8
	Mutation function	simpleGAMutationUniform
	Migration direction	'forward'
	Migration interval	20
	Migration fraction	0.2
PSO	Inertia (w)	1.4
	Self trust (c_1)	1.5
	Swarm trust (c_2)	2.5
	DT	1
	Vmax	0.2
	Mass extinction factor	0.975
	Coefficient of Variation	1
	Swarm Subset	0.2

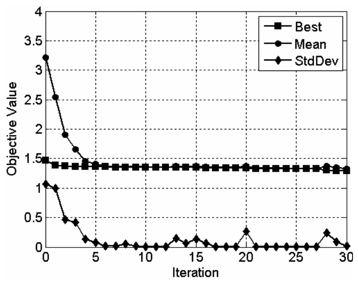
3. Results and Discussion

In the first part of the study, the performance and convergence of the heuristic algorithms is tested Figure 5.3 and Figure 5.4 show an arbitrary run executed with each of them. They help as an initial comparison basis to analyze the behavior of the used techniques. Figure 5.3 shows how the best, the mean and the standard deviation of the objective function values for the population of ACO, GA and PSO evolve during the optimization procedure. Figure 5.4 illustrates the evolution of the LC along the iterations.

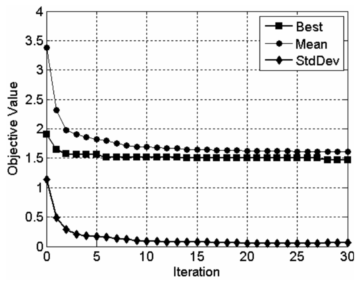
Figure 5.4-(a) shows which heuristic is conducting the optimization process at a given iteration. Figure 5.4-(b) shows the transitions due to its self-adaptation skills.



(a) ACO



(b) GA

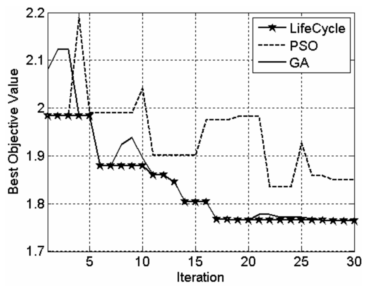


(c) PSO

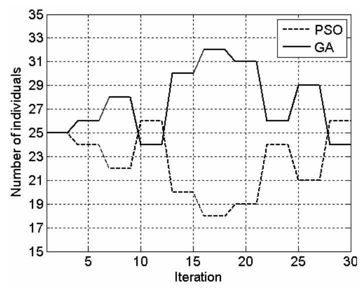
Figure 5.3. Sample of one run of the optimization for each basic algorithm.

In all cases shown above, good convergence both in terms of results (represented by the best value of the objective function) and in terms of the dispersion of the population (represented by the mean and standard deviation of the objective function) can be observed.

Figure 5.4-(a) illustrates how GA and PSO alternate in providing to LC the optimal solution. During the first 4 iterations, this is done by PSO, then the solution is few times updated by GA. Figure 5.4-(b) shows how this process is reflects the split of the population between GA and PSO. Since most of the time GA performs better than PSO, it can be observed that the population trends to perform GA rather than PSO.



(a) Best objective trace.



(b) Number of individuals trace.

Figure 5.4. Sample of one run of the optimization for the LC algorithm.

Given the random nature of the algorithms used, the profiles of Figure 5.3 and Figure 5.4 may vary from one run to another (final results may also vary slightly). This way, for the sake of the statistical analysis, each run was repeated 10 times and the best, worst, average, standard deviation and coefficient of variation of the results from each of the 10 repetitions were recorded. The average indicates the central tendency which the heuristic results tend to. The standard deviation indicates the stability of the optimal solutions, measuring the degree of convergence. The average value for is the average of values of the functional over the 10 runs; it does not correspond to the value obtained when running the analysis with the average design. Table 5.7 reports this information for each of the heuristic methods. The best and average results are shown in bold face. Table 5.7 shows that all algorithms converged fairly well. While GA achieved the best result, LC presents better dispersion (smaller differences among the best, average and worst designs). This is an evidence of the LC robustness, somehow expected, since LC is intended to take advantage of multiple heuristics simultaneously and to reduce the bias of a poorly performed algorithm. Altogether, Table 5.7 also indicates the success of all algorithms in pointing to a close region of the design space.

Table 5.7. Optimization results for all heuristic algorithms (performed alone).

Algorithm	**Statistics**	**Design variable**				**Objective function** $J(\mathbf{x})$
		$x_1 \times 10^{-3}$	$x_2 \times 10^{-3}$	$x_3 \times 10^{-3}$	$x_4 \times 10^{-3}$	
ACO	**Best**	25	41.94	1.5	1.08	**1.44**
	Average	24.96	40.19	1.5	1.03	**1.73**
	Worst	24.96	39.4	1. 5	1.02	1.76
	StdDev	0.035	0.73	0.0001	0.02	0.10
GA	**Best**	29.08	42.87	1.65	0.98	**1.3**
	Average	26.70	38.37	1.63	1.00	**1.67**
	Worst	22.5	47.4	1.47	1.06	1.9
	StdDev	2.55	5.65	0.09	0.04	0.16
PSO	**Best**	24.89	41.35	1.49	1.07	**1.47**
	Average	24.61	40.72	1.49	1.04	**1.76**
	Worst	24.3	38.87	1.48	1.06	1.81
	StdDev	0.42	1.59	0.007	0.03	0.10
LC	**Best**	25.00	40.28	1.50	1.03	**1.764**
	Average	24.62	40.47	1.5	1.04	**1.774**
	Worst	24.38	41.05	1.5	1.05	1.785
	StdDev	0.4	0.64	0.002	0.01	0.009

Table 5.8 shows what happens when running the objective function with the average result found by each algorithm. The response values are all obtained by linear analysis (low fidelity) and further on properly corrected. Indeed, despite the apparent differences among the average values (which suggests small advantage for GA), the four averages are statistically equivalent at a 95% significance level, since all of them belong to the corresponding confidence interval that goes from 1.667 to 1.828.

Table 5.8. Comparison of the average design and the corresponding responses for each algorithm.

Design		Algorithm			
		ACO	**GA**	**PSO**	**LC**
Design variables	$x_1 \times 10^{-3}$	24.96	26.703	24.614	24.624
	$x_2 \times 10^{-3}$	40.191	38.367	40.717	40.473
	$x_3 \times 10^{-3}$	1.5	1.625	1.49	1.499
	$x_4 \times 10^{-3}$	1.03	1.003	1.038	1.036
Responses (linear corrected)	$f_1(\mathbf{x})$	0.96	0.92	0.96	0.96
	$f_2(\mathbf{x})$	-0.72	-0.66	-0.72	-0.72
	$f_3(\mathbf{x}) \times 10^{-3}$	109.42	109.44	109.23	109.28
	$f_4(\mathbf{x})$	0.18	0.18	0.18	0.18
	$f_5(\mathbf{x})$	21.56	21.82	21.60	21.6
Functional	$J(\mathbf{x})$	1.77	**1.67**	1.78	1.77

In the second part of the study, the efficiency of heuristics in providing an initial guess for a classical algorithm is tested. Table 5.9 shows the results when running another step of optimization using NMSDS with initial designs as given by Table 5.8. All final designs present even closer values for the design variables, individual responses and respective functional, $J(\mathbf{x})$. Indeed, the initial designs were in a close region of the design space.

Table 5.9. Results when using NMSDS with initial designs given by different heuristics.

Design		Algorithm			
		ACO + NMSDS	**GA + NMSDS**	**PSO + NMSDS**	**LC + NMSDS**
Design variables	$x_1 \times 10^{-3}$	29.93	29.96	29.93	29.94
	$x_2 \times 10^{-3}$	33.06	33.04	33.04	33.04
	$x_3 \times 10^{-3}$	1.73	1.73	1.73	1.73
	$x_4 \times 10^{-3}$	0.96	0.96	0.96	0.96
Responses (linear corrected)	$f_1(\mathbf{x})$	0.89	0.89	0.89	0.89
	$f_2(\mathbf{x})$	-0.67	-0.67	-0.67	-0.67
	$f_3(\mathbf{x}) \times 10^{-3}$	112.34	112.35	112.34	112.35
	$f_4(\mathbf{x})$	0.17	0.17	0.17	0.17
	$f_5(\mathbf{x})$	21.79	21.79	21.79	21.79
Functional	$J(\mathbf{x})$	1.64	1.64	1.64	1.64

As seen in Chapter IV, the final step of the variable-fidelity approach is the validation of the optimization outcomes. This is done by running the high-fidelity simulations using the optimal design. Table 5.10 shows the results of an arbitrary run of the LC algorithm as performed lonely (simulation #10) and the results of LC + NMSDS (as in Table 5.9). In general, the non-linear and linear corrected simulations present just small differences for responses $f_1(\mathbf{x})$ to $f_4(\mathbf{x})$ ($f_5(\mathbf{x})$ is the pressure bulkhead mass and it does not depend on the level of fidelity). The similar results (bold face in Table 5.10) show once again that the candidate solutions found by the heuristic methods lay in a neighborhood of the design space, where the optimal may be. From the design viewpoint, the changes introduced by the optimizers seem to be sound choices. Material has been removed from the web to the reinforcement beams, in a clear effort to minimize the mass, which was given highest priority by means of the 2.5 weighting factor in the *Compromise Programming*. Most important, the optimizers were capable of finding a design that overall improved the involved quantities, despite their inherent conflicting nature. The small increase in the displacement is secondary in view of the improvements observed in the other responses.

Table 5.10. Validation of the optimal designs given by LC and LC + NMSDS.

Design		**Response**				
		$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$ $\times 10^{-3}$	$f_4(\mathbf{x})$	$f_5(\mathbf{x})$
Baseline (Non-linear)		1.01	-0.85	116.0	0.2	22.56
LC (simulation #10, $J(\mathbf{x}) = 1.76$)	Linear uncorrected	3.42	-5.35	1.71	0.09	20.95
	Linear corrected	0.92	-0.66	109.44	0.17	20.95
	Non-linear	**0.91**	**-0.65**	**117.5**	**0.17**	**20.95**
	Optimization effect (%)	-9.90	-23.53	+1.29	-15.00	-7.14
LC + NMSDS ($J(\mathbf{x}) = 1.64$)	Linear uncorrected	0.06	-0.0709	1.85	0.0016	21.79
	Linear corrected	0.89	-0.67	112.35	0.17	21.79
	Non-linear	**0.88**	**-0.62**	**118.20**	**0.18**	**21.79**
	Optimization effect (%)	-12.87	-27.06	1.90	-10.00	-3.41

4. Summary and Conclusions

In this application, the use of intensive computing heuristic techniques for the optimal design of aircraft structural components was described. It was explored (i) coupling of non-linear high-fidelity and linear low-fidelity analyses through the variable fidelity approach; and (ii) the performance of different heuristic optimization methods.

The study allows the following conclusions:

- Variable fidelity approach enabled the use of intensive computing heuristic techniques and expensive non-linear analyses.
- The four applied nature-inspired methods converged in the sense of pointing out the same region of the design space as a candidate to contain the most suitable design. This means that these techniques can be used for initial exploration of the design space, with the purpose of identifying the improvement trends and ruling out the inadequate design choices.

5.1. Inverse Problems

5.1.1. Aircraft Longitudinal Stability and Control Derivatives Identification by using LifeCycle and Levenberg-Marquart Optimization Algorithms

This application explores the use of global (heuristic) and local (classical) optimization algorithms *in solo* performance and in a cascade-type approach to accomplish the parameter identification of the longitudinal motion of a real aircraft by using the Output Error Method. The first algorithm is the LC Model (composed by Genetic Algorithms and Particle Swarm Optimization). The second one is the gradient-based technique named Levenberg-Marquardt algorithm (LM, detailed in Appendix D), which is a variant of the Gauss-Newton method. Flight test data, performed with a military aircraft (Xavante AT-26 FAB 4509), were used to feed the Output Error Method. In this context, both optimization algorithms were tested, in solo performance and in a cascade-type approach. This application shortly reported Góes et al. (2007), and later detailed in Viana et al (2008c).

1. Experimental Data

The identification of longitudinal stability derivatives from flight test data were performed using a set of experimental flight data performed with a military aircraft, namely the Xavante (AT-26 FAB 4516), as illustrated in Figure 5.1. The AT-26 FAB 4509 flight test instrumentation system installed on the aircraft is composed by an inertial sensor, an anemometric boom, a GPS sensor, a surface position sensor; and a data acquisition and record unity. It provides the following measures: elevator deflection (input data in the identification procedure), angle-of-attack, pitch rate (both output data).

Figure 5.1. Xavante AT-26 FAB 4516.

The angle-of-attack (an airflow measurement) requires an air data probe installed in a region of the aircraft in which the minimum disturbance of the inflow air by the aircraft is encountered. The pitch rate (being an inertial quantity) requires an inertial measurement unit generally installed near the center-of-gravity of the air vehicle in order to reduce the necessity of some corrections in acceleration measurements. The elevator deflection (a displacement) is generally measured by potentiometers directly installed in the aircraft control surfaces. Table 5.1 gives some information about the experimental work to evaluate some of the AT-26 FAB 4509 characteristics.

Table 5.1. AT-26 FAB 4509 data acquisition setup.

Measurement	**Acquisition Rate**	**Acquisition Accuracy**
Angle-of-attack	32 hz	0.11 %
Pitch Rate	32 hz	0.5 %
Elevator Deflection	16 hz	1.0 %

A single and manually applied longitudinal flight test maneuver was collected. It was specified in order to excite, mainly, the short period longitudinal dynamic model. Figure 5.2 shows the elevator deflection used as input in the identification procedure.

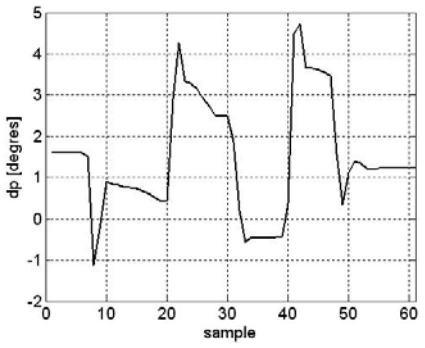


Figure 5.2. Elevator deflection.

2. Identification Problem Formulation

Figure 5.3 shows a scheme of the output error method, one of the most used estimation methods for aerodynamic parameter estimation from flight test data (Iliff, 1989) and the method used in this application. The output error method has several desirable statistical properties, including its applicability to nonlinear dynamical systems and the proper accounting of measurements noise (Iliff, 1989). In addition, it has no specific requirement about the optimization method. Even though most of works in the literature reports the use of classical methods, there are no limitations in using other approaches. In this scenario, three optimization strategies, described in Table 5.2, are explored to solve the identification problem.

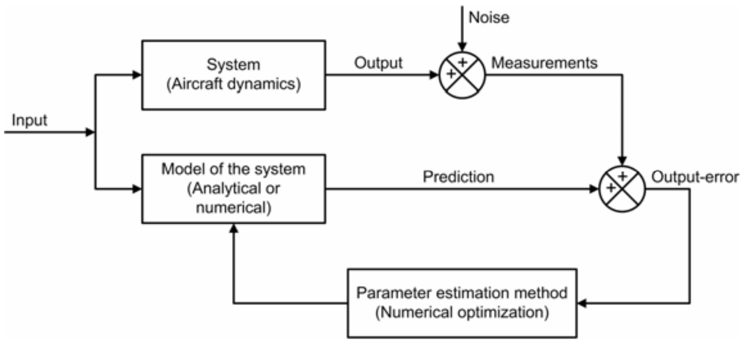

Noise
System (Aircraft dynamics)
Output
Measurements
Input
Model of the system (Analytical or numerical)
Prediction
Output-error
Parameter estimation method (Numerical optimization)

Figure 5.3. Block diagram of the estimation procedure.

Table 5.2 Optimization strategies used in the output error scheme.

Strategy	Optimization Method	Comments
$\#1$	LC	The main explored characteristics are the global search capability and the low level of knowledge about the search space.
$\#2$	LM	As a classical method, local search capabilities and small number of function evaluations are the main explored advantages.
$\#3$	LC and LM	The result of a first global search performed by LC is used as an initial guess in the LM method. This cascade approach aims to make use of the main advantages of both methods.

As can be found in Góes et al. (2007) and Viana et al (2008c), the longitudinal equations of motion of the aircraft for the short period mode are described in terms of 10 variables, which constitute the search space defined in Table 5.3. Parameters Z_α, Z_q, M_α, M_q, Z_{δ_e}, and M_{δ_e} are the stability and control derivatives, which represent the aerodynamic model of the aircraft. Parameters b_{x_1}, b_{x_2}, b_{y_1}, and b_{y_2} comprise measurement biases in the output and input variables, and initial conditions on the state variables.

Table 5.3. 10-dimensional search space.

Parameter	Search space	Parameter	Search space
Z_α	$\begin{bmatrix}-5 & 0\end{bmatrix}$	$M_{\delta e}$	$\begin{bmatrix}-10 & 0\end{bmatrix}$
Z_q	$\begin{bmatrix}0 & 5\end{bmatrix}$	b_{x_1}	$\begin{bmatrix}-0.1 & 0.1\end{bmatrix}$
M_α	$\begin{bmatrix}-5 & 0\end{bmatrix}$	b_{x_2}	$\begin{bmatrix}-0.1 & 0.1\end{bmatrix}$
M_q	$\begin{bmatrix}-5 & 0\end{bmatrix}$	b_{y_1}	$\begin{bmatrix}-0.1 & 0.1\end{bmatrix}$
$Z_{\delta e}$	$\begin{bmatrix}0 & 5\end{bmatrix}$	b_{y_2}	$\begin{bmatrix}-0.1 & 0.1\end{bmatrix}$

The model is considered to be known, and the identification process consists in determining the parameter vector, $\mathbf{x}$, which gives the best prediction of the output signal $y(t)$, using optimization criteria. The attainment of an estimate through optimization of a functional based on the prediction error of the plant involves the minimization of a nonlinear function. Thus, LC, LM or a composition of both is used here to estimate the parameters of the model. Therefore, the functional involves the prediction error:

$$e(k) = \hat{y}(k) - y(k), \tag{0.1}$$

where $\hat{y}(k)$ is the output prediction based on the estimate $\hat{\mathbf{x}}$ of the parameter vector $\mathbf{x}$ at the k sample.

Consider an identifiable dynamic system represented by a model $M(\mathbf{x})$ and output y. Suppose that $p(y|\mathbf{x})$ is the conditional probability Gaussian distribution of the random variable y with dimension m, mean $f(\mathbf{x})$ and covariance R, with dimension $m \times m$. $p(y|\mathbf{x})$ is known as the likelihood functional. Goodwin and Payne (1977) attribute this name due to the fact that it is a measure of the probability of occurrence of the observation y for a given parameter $\mathbf{x}$. Thus, the likelihood functional is:

$$p(y|\mathbf{x}) = \frac{1}{(2\pi)^{m/2}\left|R^T\right|^{n/2}}.\exp\left\{-\frac{1}{2}\sum_{k=1}^{n}[e(k,\mathbf{x})]^T\left[R^T\right]^{-1}[e(k,\mathbf{x})]\right\}. \tag{0.2}$$

The Maximum Likelihood Estimate (MLE) is defined as the value of $\mathbf{x}$ which maximizes this functional, in such a way that the best estimate of $\mathbf{x}$ is:

$$\hat{\mathbf{x}} = ArgMaxp(y|\mathbf{x}) \tag{0.3}$$

The maximization of $p(y|\mathbf{x})$ is equivalent to the minimization of $J(\mathbf{x})$, which is given by:

$$J(\mathbf{x}) = \sum_{k=1}^{n}\frac{1}{2}\left\{[e(k,\mathbf{x})]^T\left[R^T\right]^{-1}[e(k,\mathbf{x})] + \ln|R|\right\}. \tag{0.4}$$

3. Results and Discussion

Given the random nature of LC and the dependency of the initial guess of LM, the results may vary from one run to another. This way, for the sake of a reasonable statistical analysis, each one of the three strategies was repeated 20 times. For strategy $\#1$, LC was set with different and random initial population of 50 individuals and maximum number of 200 iterations. For strategy $\#2$, LM was set with different and initial guesses randomly picked within the design space. Finally, for strategy $\#3$, the results obtained when running LC were used as initial guesses for LM. Besides the best and the mean results, a dispersion measure called coefficient of variation (COV) was used for comparing the performance of the different strategies. COV is computed by dividing the standard deviation by the mean (i.e., it intends to filter the order of magnitude).

Table 5.4 to Table 5.6 report a set of identification results obtained following the three different optimization strategies. It is worth mentioning that the mean value for J is the mean value of the functional over the 20 runs; it does not correspond to the value obtained when running the analysis with the mean values of the design variables.

Table 5.4 shows the best and the mean results as well as COV. Results are in a close region of the design space for strategy $\#1$.

Table 5.4. Results for strategy $\#1$ (LC, 50 individuals and 200 iterations).

	Best run	**Mean**	**COV**
Z_α	-1.94	-2.23	-0.40
Z_q	1.00	0.99	0.41
M_α	-2.85	-2.01	-0.65
M_q	-0.83	-1.42	-0.53
$Z_{\delta e}$	2.74	2.57	0.37
$M_{\delta e}$	-5.78	-7.20	-0.26
b_{x_1}	-0.06	-0.01	-6.07
b_{x_2}	0.1	0.1	0.08
b_{y_1}	0.1	0.09	0.13
b_{y_2}	0.04	0.06	0.31
J	7.5×10^{-9}	2.1×10^{-8}	0.41

For strategy $\#2$, most of the runs did not find a feasible solution. Out of the 20 runs, 2 ended with an indetermination for the value of the functional J, 2 other runs ended with $J = +\infty$ and $J = -\infty$, and only 3 ended up with feasible solutions (revealing strong dependence on the initial guess when executing LM). Consequently, Table 5.5 was build computing only the remaining 16 runs. In addition to this, both mean and COV values reveal the dependency on the initial guess. Figure 5.4 helps the interpretation of these results.

Table 5.5. Results for strategy $\#2$ (LM with initial guesses randomly picked within the design space).

	Best run	**Mean**	**COV**
Z_α	-0.70	-20.87*	-1.19
Z_q	0.23	6.45*	3.31
M_α	-2.37	-2.26	-11.73
M_q	-1.18	1.29*	4.13
$Z_{\delta e}$	0.38	2.89*	10.4
$M_{\delta e}$	-7.16	0.002*	1.8×10^{4}
b_{x_1}	-0.01	-1.23	-2.92
b_{x_2}	0.20	1.88*	4.94
b_{y_1}	0.08	2.75*	6.80
b_{y_2}	0.02	2.63*	2.73
J	1.2×10^{-11}	7.5×10^{149}	4

*: boundary violation.

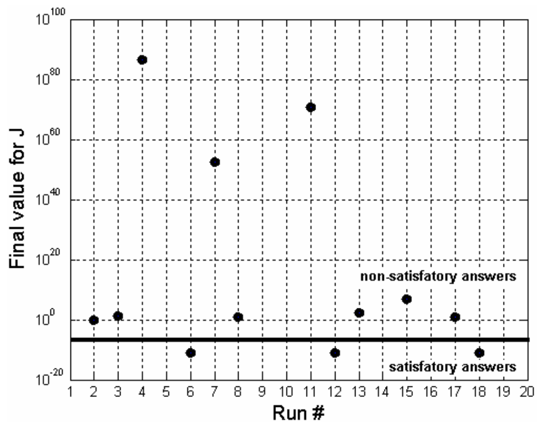


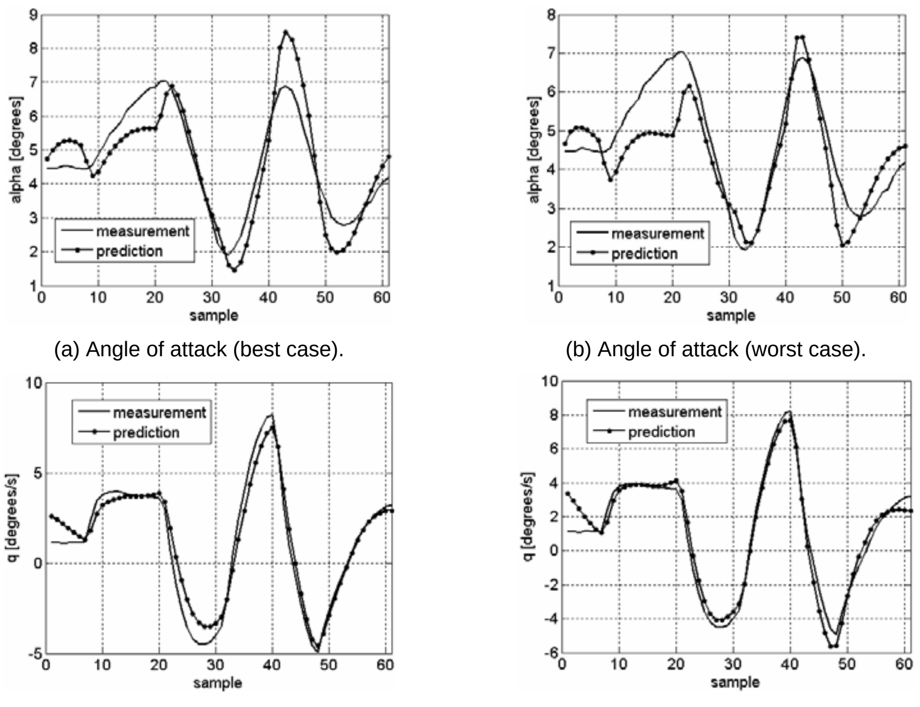
Figure 5.4. Final values of J for strategy $\#2$ (LM).

Table 5.6 shows performance when using strategy $\#3$. It is easy to see the capability of the strategy in finding nearly same solutions. Differently from results shown in Table 5.4 and Table 5.5, here there is very small dispersion and all runs end up in feasible solutions.

Now that is clear that the winning approach is the strategy $\#3$, the question that arises is how important the improvements driven by LM are. The answer is given when the predictions are compared. This way, Figure 5.5 and Figure 5.6 illustrate the prediction capabilities of strategy $\#1$ (LC alone) and strategy $\#3$ (LC+LM). Figure 5.5 helps to see that even though relatively close, the results found by LC are not good enough as in Figure 5.6, which illustrates the success of the LC+LM strategy.

Table 5.6. Results for strategy $\#3$ (initial guesses for LM provided by previous run of LC).

	Best run	**Mean**	**COV**
Z_α	-0.7012	-0.7012	-0.07 x 10^{-3}
Z_q	0.2308	0.2308	-0.10 x 10^{-3}
M_α	-2.3688	-2.3689	-0.2 x 10^{-3}
M_q	-1.1760	-1.1760	-0.2 x 10^{-3}
$Z_{\delta e}$	0.3841	0.3841	0.2 x 10^{-3}
$M_{\delta e}$	-7.1653	-7.1652	-0.1 x 10^{-3}
b_{x_1}	-0.0093	-0.0093	-0.3 x 10^{-3}
b_{x_2}	0.2037	0.2037	0.05 x 10^{-3}
b_{y_1}	0.0774	0.0774	0.05 x 10^{-3}
b_{y_2}	0.0182	0.0182	0.09 x 10^{-3}
J	1.2 x 10^{-11}	1.2 x 10^{-11}	2.2 x 10^{-5}



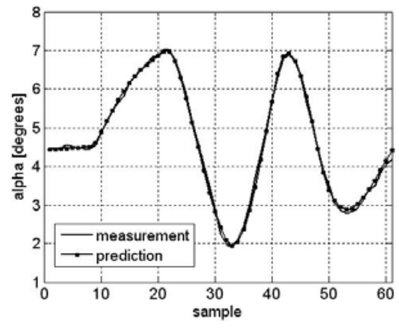
(a) Angle of attack (best case).

(b) Angle of attack (worst case).

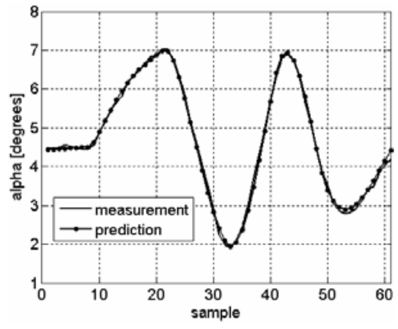
(c) Pitch angular rate (best case).

(d) Pitch angular rate (worst case).

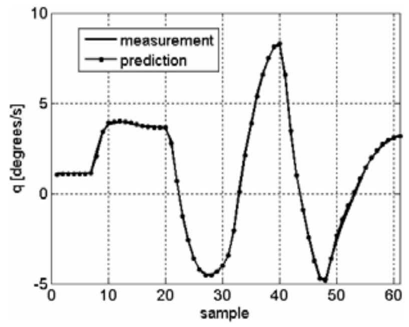
Figure 5.5. LC performance.



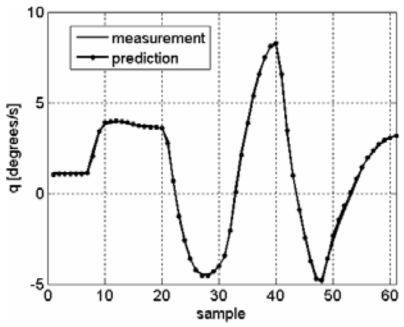
(a) Angle of attack (best case).



(b) Angle of attack (worst case).



(c) Pitch angular rate (best case).



(d) Pitch angular rate (worst case).

Figure 5.6. LC+LM performance.

4. Summary and Conclusions

This application presented an identification procedure to determine the longitudinal stability and control derivatives of a military aircraft. Within the framework of the Output Error method, the tested optimization algorithms are based on LC and LM methods. In the present application, GA and PSO represented the phases of the LC algorithm. The individual performances of both the algorithms were tested and then a cascade-type scheme was proposed aiming at taking advantage of the global search capabilities of LC and the local search capabilities of LM. Finally, the experimental investigation illustrated the possibility of using the proposed technique in real world environment. The results are very encouraging in the sense that more complex models, embracing non-linearities, will be analyzed in further research.

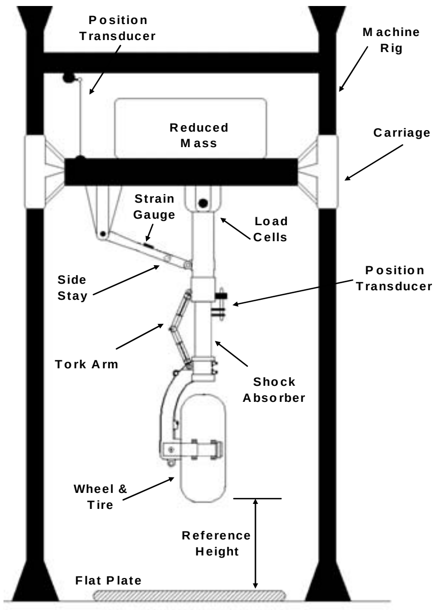
5.1.2. Identification of a Non-Linear Landing Gear Model using Heuristic Optimization

This application explores the parameter identification of a non-linear landing gear model proposed as an optimization problem solved by a heuristic optimization method, namely the LC algorithm (with GA and PSO representing the phases of the LC algorithm). The model is described in terms of the landing gear geometry, internal volumes and areas, shock absorber travel, tire type, and gas and oil characteristics of the shock absorber. The polytropic coefficient of the gas and the damping coefficient of the shock absorber are assumed as being unknown: they are considered as design variables. As an illustration, experimental drop test data, obtained under zero horizontal speed, were used in the non-linear landing gear model updating of a small aircraft. Preliminary results of this application was reported in Zanini et al. (2007), the full report is in Viana et al. (2008e).

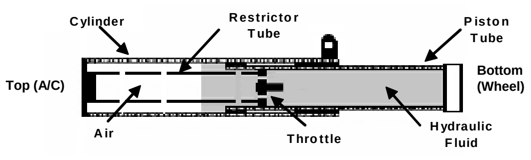
5. Drop Test Arrangements and Non-Linear Landing Gear Model

Figure 5.7-(a) illustrates a simplified drawing of the drop test installation and landing gear components. The landing gear is attached to a carriage that runs vertically between two rigs. This carriage is pre-loaded with the equivalent aircraft reduced mass. Position transducers are used to determine the vertical displacement of the carriage and deflection of the shock absorber. The difference between these parameters gives the tire deflection. Three-dimensional load cells together with strain gages are used to obtain the impact loads transferred to the aircraft. It is important to point out that, in this arrangement, no apparatus is used to simulate the aerodynamic lift.

The reduced mass used in the drop test represents the portion of the aircraft mass loaded on the landing gear during the landing impact. This mass can be calculated based on the aircraft static configuration (aircraft attitude on the ground) and certification requirements and it is further corrected to take into account the aircraft lift. A telescopic-type landing gear with an oil-pneumatic shock absorber was used. The main components of the oil-pneumatic shock absorber are shown in Figure 5.7-(b). The lower chamber contains hydraulic fluid that is forced to flow through small orifices (throttle) as a result of the piston displacement during landing. The impact energy is dissipated in this process and the air contained in the upper chamber compressed. After the maximum shock absorber deflection, part of the hydraulic fluid flows back into the lower chamber due to the expansion of the compressed air. In this extension stage, the numbers of orifices through which the fluid flows must be enough to guarantee that the tire stays in contact with the runway avoiding the rebound effect. The shock absorber considered in the present work uses eight orifices for compression and four orifices during extension. The “unsprung” mass corresponds to the sum of all movable parts of the landing gear as tire, wheel, piston tube, fork, etc. Part of the hydraulic fluid mass is also computed in the “unsprung” mass.



(a) Drop test sketch.



(b) Oil-pneumatic shock absorber.

Figure 5.7. Drop test arrangement.

Further details about the non-linear modeling of the landing gear can be found in Viana et al. (2008e). In the scope of this text, it is enough to give general ideas involved in the formulation of the equations of movement. Figure 5.8 shows the two degrees of freedom landing gear model used in this study. In this model, M_1 and M_2 represent respectively the reduced and “unsprung” masses. K_1 and b correspond to the pneumatic spring and damping coefficients of the shock absorber. The tire is represented by a pneumatic spring K_2. The vertical displacements of both reduced and “unsprung” masses are named as Z_1 and Z_2, respectively. The shock absorber deflection Z is obtained by the difference between the displacements Z_1 and Z_2.

Viana et al. (2008e) shows details on the modeling of quantities that are used in the formulation of the optimization problem, namely, the shock absorber spring force (F_{K_1}), the shock absorber damping force (F_b), tire spring force (F_{K_2}), the vertical displacement of the reduced mass (X_1), the tire deflection (X_3), the shock absorber deflection (Z), and the values of the maximum vertical forces (F_{V_1} and F_{V_2}). F_{V_1} is the vertical force that corresponds to the first half of the shock absorber deflection stage. F_{V_2} is the vertical force that corresponds to the second half of the deflection stage. Figure 5.9 illustrates the behavior of both F_{V_1} and F_{V_2}.

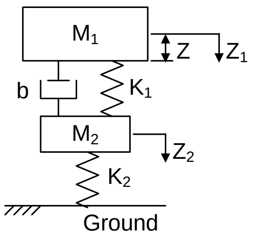


Figure 5.8. Landing gear model graphical representation.

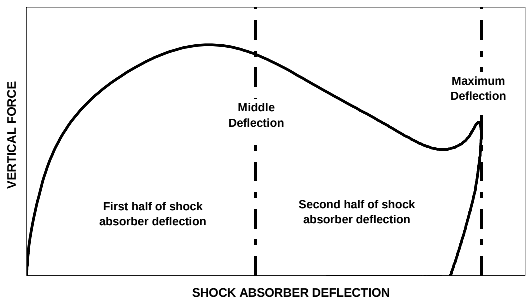


Figure 5.9. Vertical force vs. deflection curve.

The equations of motion that represent the landing gear dynamic behavior are obtained by combining the Lagrangian and Rayleigh's dissipation functions and replacing the terms that represent the shock absorber spring force (F_{K_1}), tire spring force (F_{K_2}) and damping force (F_b). Thus, after some algebraic manipulation:

$$\begin{cases} \ddot{Z}_1 = -\dfrac{1}{M_1}\cdot F_{K_1} - \dfrac{1}{M_1}\cdot F_b + g \quad , \\ \ddot{Z}_2 = \dfrac{1}{M_2}\cdot F_{K_1} - \dfrac{1}{M_2}\cdot F_{K_2} + \dfrac{1}{M_2}\cdot F_b + g \quad . \end{cases} \tag{0.5}$$

In order to obtain the numerical solution, it is assumed that $X_1 = Z_1$, $X_2 = \dot{Z}_1 = \dot{X}_1$, $X_3 = Z_2$, and $X_4 = \dot{Z}_2 = \dot{X}_3$.

It is also possible to write four first order differential equations, as presented below:

$$\begin{cases} \dot{X}_1 = X_2 \quad , \\ \dot{X}_2 = -\dfrac{1}{M_1}\cdot F_{K_1} - \dfrac{1}{M_1}\cdot F_b + g \quad , \\ \dot{X}_3 = X_4 \quad , \\ \dot{X}_4 = \dfrac{1}{M_2}\cdot F_{K_1} - \dfrac{1}{M_2}\cdot F_{K_2} + \dfrac{1}{M_2}\cdot F_b + g \quad . \end{cases} \tag{0.6}$$

Shock Absorber Spring Force (F_{K_1})

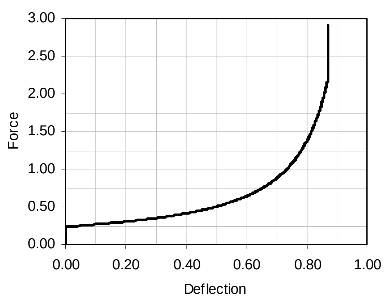
The non-linear spring characteristic of the shock absorber is taken into account by considering the compression of the air and of the hydraulic fluid interconnected with a common force. Considering a fluid with uniform mass density and constant secant bulk modulus (hydraulic fluid stiffness), the force due to hydraulic fluid compression is given by:

$$F_{K1} = P_0 \cdot A \cdot \left(\frac{v_0}{v_0 - A \cdot s_1} \right)^n = \frac{\beta \cdot A^2 \cdot s_2}{V_0} \quad , \tag{0.7}$$

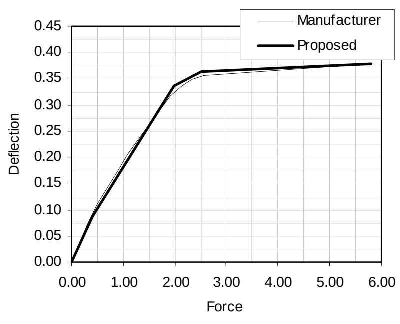
where:

- P_0 is the absolute air pressure for the fully extended shock absorber,
- A is the piston reference area,
- v_0 is the air volume for the fully extended shock absorber,
- s_1 is the shock absorber deflection due to the air compression,
- n is the polytrophic coefficient (constant value between 1.0 and 1.4 for adiabatic and isothermal compression, respectively),
- β is the secant bulk modulus,
- s_2 is the shock absorber deflection due to hydraulic fluid compression, and
- V_0 is the hydraulic fluid volume for the fully extended shock absorber.

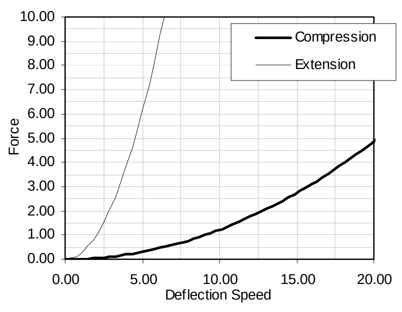
The sum of the deflections obtained from air and fluid compression, (s_1) and (s_2) respectively, corresponds to the difference between the reduced and "unsprung" masses displacements. Figure 5.10-(b) illustrates the absorber force versus deflection.



(a) Shock absorber force vs. deflection curve.



(b) Tire load vs. deflection curve.



(c) Hydraulic force vs. deflection speed.

Figure 5.10. Details of the landing gear model.

Tire Spring Force (F_{K_2})

The tire was modeled as a pure spring element. The tire load versus deflection curve supplied by the manufacturer was divided into four linear regions. Therefore, the tire stiffness assumed four different values depending on the tire deflection at each simulation step. A comparison between the proposed force vs. deflection curve and that obtained from the tire manufacturer is presented in Figure 5.10-(b).

Hydraulic Resistance Force (F_b)

The hydraulic resistance in the shock absorber resulted from the difference of pressure associated with the fluid flow through the orifice. In the considered landing gear, the orifice area is enough in relation to the diameter of the shock absorber so that the jet velocities and Reynolds numbers are sufficiently large, exhibiting a fully turbulent flow. As a result, the damping force varies with the square of the telescoping velocity rather than linearly with the velocity. As a consequence, the hydraulic resistance (F_b) is given by:

$$F_b = \frac{\rho \cdot {A_h}^3 \cdot \dot{Z}^2}{2 \cdot (C_d \cdot A_n)^2} = \frac{\dot{Z}}{\left|\dot{Z}\right|} \cdot DS \cdot g \cdot \dot{Z}^2 \qquad (0.8)$$

where ρ is the mass density of the hydraulic fluid, A_h is the hydraulic area, $\dot{Z}$ is the shock absorber deflection velocity, C_d is the coefficient of discharge, A_n is the net orifice area, DS is a constant given by $DS = \dfrac{\rho \cdot {A_h}^3}{2 \cdot g \cdot (C_d \cdot A_n)^2}$, and g is the acceleration of gravity.

Figure 5.10-(c) is obtained by using the quadratic relation between the hydraulic force and the shock absorber deflection speed and by considering that the number of orifices reduces during shock absorber extension to avoid rebound effect.

Vertical Force Transferred to the Reduced Mass (F_v)

The vertical force transferred to the reduced mass was based on the D'Alembert's principle. Thus:

$$F_v = M_1 \cdot g - M_1 \cdot \ddot{Z}_1 \qquad (0.9)$$

6. Identification Problem Formulation

The parameters that cannot be accurately determined by theoretical methods were selected for identification: the polytrophic coefficient (n), and the damping parameter (DS). The corresponding identification problem consists in finding the set of design variables $\mathbf{x} = \begin{bmatrix} n & DS \end{bmatrix}^T$ that minimizes the objective function given by:

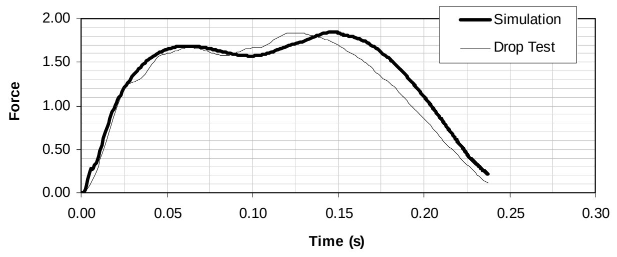
$$J(\mathbf{x}) = \frac{\left|X_{1trial}^{\max} - X_{1\exp}^{\max}\right|}{X_{1\exp}^{\max}} + \frac{\left|X_{3trial}^{\max} - X_{3\exp}^{\max}\right|}{X_{3\exp}^{\max}} + \frac{\left|Z_{trial}^{\max} - Z_{\exp}^{\max}\right|}{Z_{\exp}^{\max}} + \frac{\left|F_{V_1 trial}^{\max} - F_{V_1\exp}^{\max}\right|}{F_{V_1\exp}^{\max}} + \frac{\left|F_{V_2 trial}^{\max} - F_{V_2\exp}^{\max}\right|}{F_{V_2\exp}^{\max}} \quad . \tag{0.10}$$

The polytropic coefficient ranges between 1.0 and 1.4 for isothermal and adiabatic processes, respectively. The discharge coefficient was theoretically calculated by taking into account the diameters of the orifices of the shock absorber and the Reynolds' number for the upstream flow condition. The bounds of the search space for the damping parameter were arbitrarily defined as $\pm 15\%$ of the theoretical value DS_{TH}.

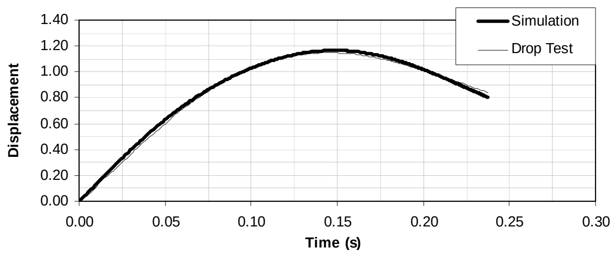
7. Results and Discussion

LC was used as the algorithm to solve the optimization problem. It was set up with 10 individuals, running along 50 iterations. The identified values are the following: polytropic coefficient $n = 1.05$, and damping parameter $DS = 622 \ kg.s^2 / m^2$. Currey (1984) recommended using a polytropic coefficient of 1.1 for shock absorbers without separator piston between gas and oil. Since this value is used only as reference and some variations can occur, for the studied shock absorber, which has oil in contact with the gas, the obtained value of 1.05 is considered as a good estimate. Similarly, the obtained damping parameter is approximately $+9\%$ above of the theoretical value, which represents also a good matching.

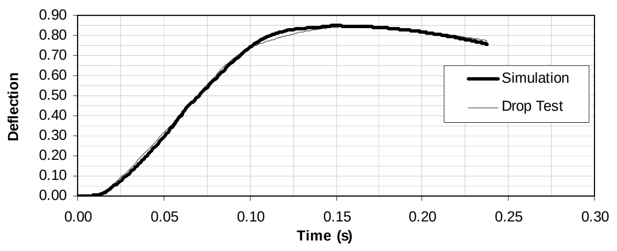
Figure 5.11 shows the graphics obtained for the simulations with the identified parameters and the drop test cases. Table 5.7 compares some experimental and simulated quantities. Again, the good performance and robustness of LC together with the proposed mathematical model were demonstrated. As can be seen, the agreement between experimental and simulated curves is satisfactory. The magnitude of the obtained loads variation is acceptable for a landing gear in the context of airplane design. Specifically for the tire deflection and vertical force, given by X_3 and F_v, respectively, small differences between experimental and theoretical results are found. This is mainly due to the flexibility effect of the landing gear fork that was not considered in the proposed mathematical model.



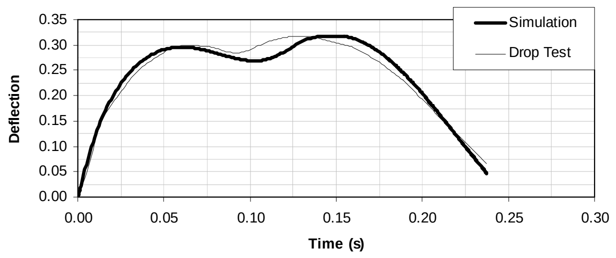
(a) Reduced mass – vertical force (F_v).



(b) Reduced mass – vertical displacement (X_1).



(c) Shock absorber deflection (Z).



(d) Tire deflection (X_3).

Figure 5.11. Comparison between simulation and experimental data.

Table 5.7. Experimental and simulated results for the landing gear.

Parameter	**Drop Test**	**Simulation**	**Error (%)**
$F_{V_1}^{\max}\ (N)$	$56,650.4$	$57,781.0$	2.0
$F_{V_2}^{\max}\ (N)$	$65,212.3$	$63,278.5$	-3.0
$X_1\ (m)$	0.315	0.318	1.2
$X_3\ (m)$	0.088	0.082	-6.1
$Z\ (m)$	0.229	0.231	1.1

8. Summary and Conclusions

This application was dedicated to the use of a heuristic optimization technique to solve the inverse problem represented by the identification of an aircraft landing gear model. For this aim, it was shown that the multi-objective function formulation makes use of both a set of measures that describe the behavior of the system and its numerical counterparts to build a representative functional, which once minimized gives a possible solution for the identification problem. The behavior of the landing gear was simulated as a non-linear model. A good match between the theoretical results and those obtained from the experimental drop-test was found. Therefore, it can be concluded that the proposed identification approach was proved successful; encouraging future research involving more complex models, such as the ones in which wheel rotation for the representation of drag loads and landing gear flexibility are included (in addition to the two degrees of freedom considered in the present model).

5.2. Coupling Actual High-Fidelity and Surrogate Models

5.2.1. Coupling Heuristic Optimization and High-Fidelity Function Analysis to Improve Surrogate Models

This section shows how to couple heuristic optimization methods and high-fidelity analysis to improve surrogate models. This was discussed during the Section 4.4 of Chapter IV. The proposed approach is based on (i) the use of a large set of surrogates; (ii) the online update of the surrogates; and (iii) the use of $BestPRESS$ for global search. In few words, the first variation updates the set of surrogates after refining any initial DOE with a pre-defined number of points. The second variation also refines any initial DOE with a pre-defined number of points; however, the update is performed every time a new point is added to the set. Here, DE was used to conduct the optimization and the performance of these approaches is studied through a set of analytical functions.

1. Numerical Experiments

A. Basic Surrogates

Table 5.8 shows the 24 different basic surrogates used during the investigation (see Chapter II for more details). The toolbox of Lophaven et al. (2002), the native neural networks Matlab toolbox (Mathworks Contributors, 2002), the code developed by Gunn (1998), and the SURROGATES toolbox of Viana and Goel (2008) were used to execute the KRG, RBNN, SVR, and PRS algorithms, respectively. SURROGATES ToolBox is also used for an easy manipulation of all these different codes. No attempt was made to improve the predictions of any surrogate by fine tuning its respective parameters. Six different Kriging surrogates are generated by varying the regression and correlation models, and 16 different SVR surrogates are created by varying the kernel function, the loss function ($\varepsilon\text{-insensitive}$ or quadratic) and the SVR parameters (C and ε). The suggestions of Cherkassky and Ma (2004) are followed for the computation of C and ε. Additionally, for SVR, details about the kernels are given in Table 2.1. See Chapter II for more details about each surrogate technique.

B. Performance Measures

Since surrogates with different statistical assumptions are used, the root mean square error ($RMSE$, defined in Sub-section 2.2.2 of Chapter II) is chosen as measure of fit quality for the surrogates. From the set of basic surrogates, two derived surrogates are used to check the performance. They are $BestPRESS$ and the most accurate surrogate of the initial

DOE, $BestRMSE$ (basis of comparison for all other surrogates). $BestRMSE$ would be selected if the performances of the surrogates were known *a priori*. In order to check the effectiveness of the strategies in adding points to the initial DOE, $BestRMSE$ is selected just once, i.e., the best surrogate of the ensemble as fitted with the initial set of data.

Table 5.8. Information about the set of 24 basic surrogates.

Surrogates		**Modeling technique**	**Details**
1	KRG-Poly0-Exp	Kriging model	Poly0, Poly1, and Poly2 indicate zero, first, and second order polynomial regression model, respectively. Exp and Gauss indicate general exponential and Gaussian correlation model, respectively. In all cases, $\theta_0 = 10 \times \mathbf{1}_{ndv\times 1}$, and $10^{-2} \le \theta_i \le 200$, $i = 1,2,\ldots,ndv$. 6 different Kriging surrogates are chosen by varying the regression and correlation models.
2	KRG-Poly0-Gauss		
3	KRG-Poly1-Exp		
4	KRG-Poly1-Gauss		
5	KRG-Poly2-Exp		
6	KRG-Poly2-Gauss		
7	PRS2	Polynomial response surface	Full model of degree 2.
8	RBNN	Radial basis neural network	$Goal = (0.05\bar{y})^2$ and $Spread = 1/3$.
9	SVR-Anova-E-Full	Support vector regression	Anova, ERBF, GRBF and Spline indicate the kernel function (ERBF and GRBF kernel functions ware set with $\sigma = 0.5$). E and Q indicate the loss function as $\varepsilon\text{-insensitive}$ and quadratic, respectively. Full means that $C = \infty$ and $\varepsilon = 1 \times 10^{-4}$, while Short01 and Short02 mean that $\varepsilon = \sigma_y/\sqrt{k}$ and $\varepsilon = 3\sigma_y\sqrt{\ln(k)/k}$, in that order, while $C = 100\max\left(\lvert\bar{y} + 3\sigma_y\rvert, \lvert\bar{y} - 3\sigma_y\rvert\right)$, where $\bar{y}$ and σ_y are the mean value and the standard deviation of the function values at the design data, respectively. 16 different SVR surrogates are chosen by varying the kernel function, the loss function ($\varepsilon\text{-insensitive}$ or quadratic) and the SVR parameters (*C* and ε) define these surrogates.
10	SVR-Anova-E-Short01		
11	SVR-Anova-E-Short02		
12	SVR-Anova-Q		
13	SVR-ERBF-E-Full		
14	SVR-ERBF-E-Short01		
15	SVR-ERBF-E-Short02		
16	SVR-ERBF-Q		
17	SVR-GRBF-E-Full		
18	SVR-GRBF-E-Short01		
19	SVR-GRBF-E-Short02		
20	SVR-GRBF-Q		
21	SVR-Spline-E-Full		
22	SVR-Spline-E-Short01		
23	SVR-Spline-E-Short02		
24	SVR-Spline-Q		

Each basic surrogate and $BestPRESS$ are compared with the best surrogate of the initial DOE ($BestRMSE$). $\%difference$ is defined such as the percent difference by choosing a specific model over $BestRMSE$:

$$\%difference = 100\frac{RMSE_{BestRMSE} - RMSE_{Surr}}{RMSE_{BestRMSE}} \quad . \tag{0.11}$$

At the initial DOE, for $BestPRESS$, it is expected that $\%difference \leq 0$, which means that there may be losses and the best case scenario is when one of the basic surrogates (hopefully $BestPRESS$) coincides with $BestRMSE$. With the final set of points, $\%difference$ for $BestPRESS$ may turns to positive, which express improvement over $BestRMSE$.

C. Analytical Examples

To test the effectiveness of the various approaches, a set of analytical functions widely used as benchmark problems in optimization is employed (Dixon and Szegö, 1978). These are:

1. Branin-Hoo function (2 variables):

$$y(\mathbf{x}) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10 \ , \quad -5 \leq x_1 \leq 10, \ \ 0 \leq x_2 \leq 15 \ . \tag{0.12}$$

2. Camelback function (2 variables):

$$y(\mathbf{x}) = \left(\frac{x_1^4}{3} - 2.1x_1^2 + 4\right)x_1^2 + x_1x_2 + \left(4x_2^2 - 4\right)x_2^2 \ , \quad -3 \leq x_1 \leq 3, \ -2 \leq x_2 \leq 2 \ . \tag{0.13}$$

3. Hartman functions (3 and 6 variables):

$$y(\mathbf{x}) = -\sum_{i=1}^{q} a_i \exp\left(-\sum_{j=1}^{m} b_{ij}\left(x_j - d_{ij}\right)^2\right) , \quad 0 \leq x_j \leq 1 \ , j = 1, \ 2, \ \ldots, \ m. \tag{0.14}$$

Hartman3, with 3 variables and Hartman6 with 6 variables were used. For both, $q = 4$, $\mathbf{a} = \begin{bmatrix} 1.0 & 1.2 & 3.0 & 3.2 \end{bmatrix}$, and other parameters are given in Table 5.9.

Table 5.10 shows details about the data set generated for each test function. Naturally, the number of points used to fit surrogates increase with dimensionality. Since quality of fit may vary with the initial DOE, results based on 100 different instances of latin hypercube designs (created with the MATLAB latin hypercube function `lhsdesign`, set with the “`maxmin`” option with 100 iterations) for all the problems are presented. For all examples, 5 different latin hypercube designs are used to test the surrogates (also created with `lhsdesign`, but set with the "`maxmin`" option and only 10 iterations). The $RMSE$ is taken as the mean of the values for the 5 DOEs. The number of extra points presented in Table 5.10 refers to those points allocated by the heuristic optimization. The number of points for update of surrogates refers to when the approach suggested on Figure 4.4 (Chapter IV) performs the step illustrated on Figure 4.4-(b). k-folds refers to the value of k used for the k-fold strategy of computation of the $PRESS$ errors. This way, for the Hartman6 function, when the number of points is equal to 60, k = 20, when it is 63, k = 21, and so on.

Table 5.9. Parameters used in Hartman function.

Hartman3	$\mathbf{B} = \begin{bmatrix} 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \\ 3.0 & 10.0 & 30.0 \\ 0.1 & 10.0 & 35.0 \end{bmatrix}$ $\mathbf{D} = \begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$
Hartman6	$\mathbf{B} = \begin{bmatrix} 10.0 & 3.0 & 17.0 & 3.5 & 1.7 & 8.0 \\ 0.05 & 10.0 & 17.0 & 0.1 & 8.0 & 14.0 \\ 3.0 & 3.5 & 1.7 & 10.0 & 17.0 & 8.0 \\ 17.0 & 8.0 & 0.05 & 10.0 & 0.1 & 14.0 \end{bmatrix}$ $\mathbf{D} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$

Table 5.10. Specifications for the Latin hypercube DOEs.

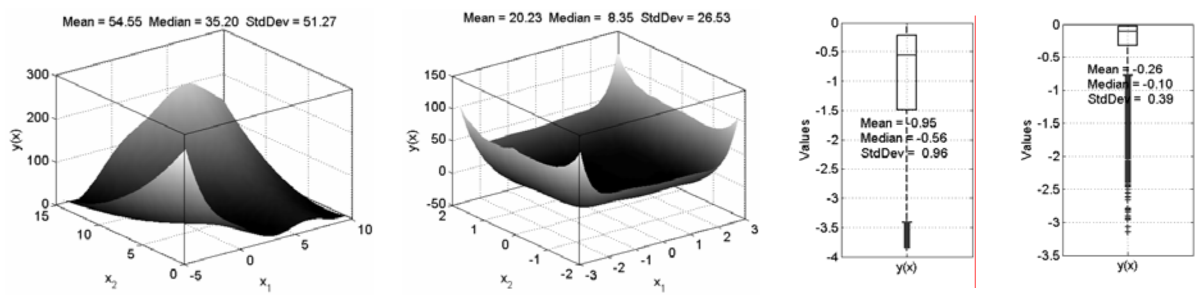
Test problem	**# design variables**	**# points for fitting**	**# extra points**	**# points for update of surrogates**	***k*-folds**	**# points for test (in each of the 5 DOEs)**
Branin-Hoo	2	12	6	13, 14, …, 18	13, 14, …, 18	2000
Camelback	2	12	6	13, 14, …, 18	13, 14, …, 18	2000
Hartman3	3	20	10	21, 22, …, 30	21, 22, …, 30	2000
Hartman6	6	56	14	60, 63, 66, 69, 70	20, 21, 22, 23, 14	2000

In addition, the $RMSE$ on the neighborhood of the optimum of each function is also studied. The setup for this part is given in Table 5.11. The DOEs were generated with the MATLAB `lhsnorm` function (latin hypercube sample with a normal distribution) set with $\mu = \mathbf{x}^*$, and $\sigma = 0.001 \times diag\left(\mathbf{1}_{n_{dv} \times 1}\right)$.

Table 5.11. Point of optimum and number of points used to estimate the $RMSE$ at the region close to the optimum.

Test problem	$\mathbf{x}^*$	**# points for test**
Branin-Hoo	$\begin{bmatrix}9.4248 & 2.4750\end{bmatrix}^T$	1000
	$\begin{bmatrix}3.1416 & 2.2750\end{bmatrix}^T$	1000
	$\begin{bmatrix}-3.1416 & 12.2750\end{bmatrix}^T$	1000
Camelback	$\begin{bmatrix}0.0898 & -0.7127\end{bmatrix}^T$	1000
	$\begin{bmatrix}-0.0898 & 0.7127\end{bmatrix}^T$	1000
Hartman3	$\begin{bmatrix}0.1 & 0.5559 & 0.8522\end{bmatrix}^T$	1000
Hartman6	$\begin{bmatrix}0.2017 & 0.15 & 0.4769 & 0.2753 & 0.3117 & 0.6573\end{bmatrix}^T$	1000

Figure 5.12 illustrates the complexity of the problems. For the two-dimensional cases, plots using the test points reveal the presence of high gradients. For all other cases, the test points were used to obtain boxplots of the functions, which show variation in the values of the functions by more than one order of magnitude. See Appendix C for details about boxplots.



(a) Branin-Hoo (b) Camelback (c) Hartman3 (d) Hartman6

Figure 5.12. Plot of test functions.

Table 5.12 shows the setup used for DE, the method used inside the algorithm proposed in Chapter IV, Figure 4.3, for all test functions.

2. Results and Discussion

The numerical experiments are intended to:

- measure the improvement due to the addition of points in the data set in the (i) $RMSE$; (ii) region close to the optimum;

- check the effectiveness of the two different strategies proposed in Figure 4.3 in adding points to the initial data set; and
- explore how to take advantage of a large set of surrogates.

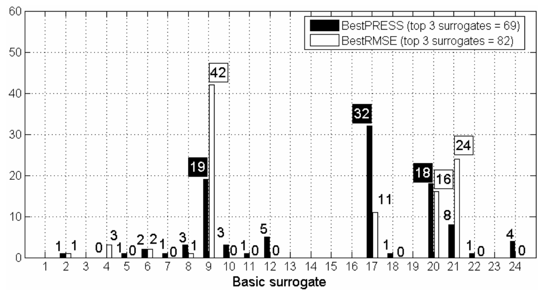
Table 5.12. Setup for DE algorithm.

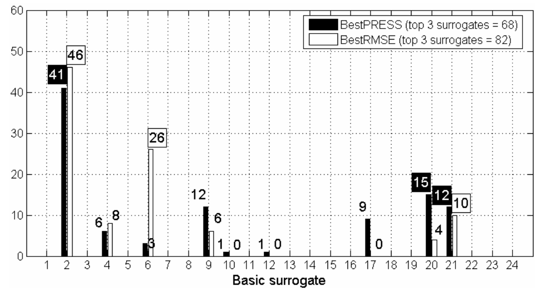
DE scheme	Amplification Factor	Crossover Probability	Max. number of Iterations	Population size
'rand-to-best/1/exp'	0.8	0.5	20	Same number of points used to fit surrogates (see Table 5.10).

The first bullet is first quantified by comparing the correlation between $PRESS_{RMS}$ and $RMSE$ across the 24 surrogates (i.e., for a given DOE, the correlation is computed between the vectors of $PRESS_{RMS}$ and $RMSE$ values for the different surrogates) and the effect on the performance measures. The second bullet is quantified by the $\%difference$ for each of the 24 surrogates and $BestPRESS$ as compared with the best surrogate ($BestRMSE$) obtained with the initial set of data.

Figure 5.13 shows the frequency of best $PRESS_{RMS}$ and the best $RMSE$ for each of the surrogates for the Branin-Hoo function. It can be observed that for both $PRESS_{RMS}$ and $RMSE$, the best surrogate depends (i) on the problem, i.e. no single surrogate or even modeling technique is always the best; (ii) on the DOE, i.e. for the same problem, the surrogate that performs the best can vary from DOE to DOE; (iii) adding points to the initial DOE may change the identity of the best surrogates; and (iv) the best 3 surrogates according to both $PRESS_{RMS}$ and $RMSE$ tend to be the same.

Figure 5.14 shows a histogram of the correlations between $PRESS_{RMS}$ and $RMSE$ for different surrogates, where the histograms represents the 100 DOEs. The Branin-Hoo and Hartman6 functions are chosen for illustrating what happens in low and high dimension, respectively. In a given DOE, the correlation is computed between the sets of $PRESS_{RMS}$ and $RMSE$ values. While none of the strategies to add points seems to be significantly better than the other, the correlation appears to improve with the number of points.


(a) Branin-Hoo, 12 points.


(b) Branin-Hoo, 18 points. Actual evaluation phase, without update of surrogates.

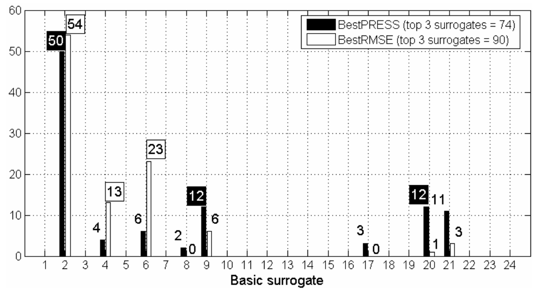
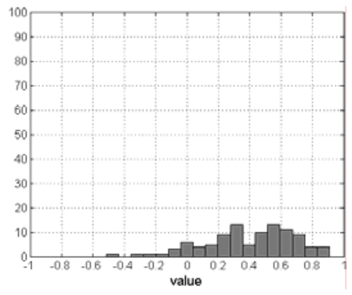
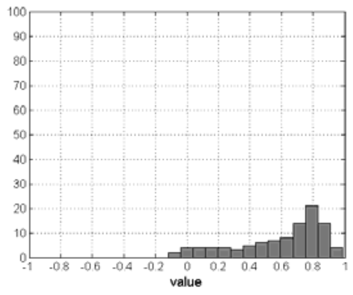
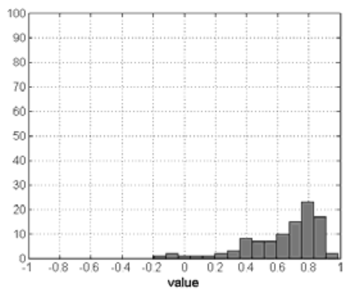
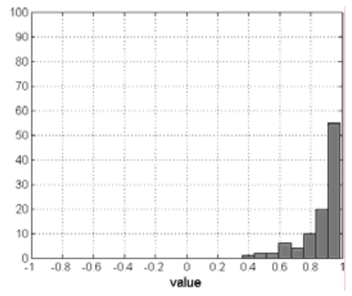

(c) Branin-Hoo, 18 points. Actual evaluation phase, with update of surrogates.

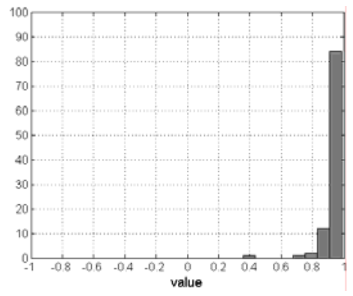
Figure 5.13. Frequency, in number of DOEs (out of 100), of best $PRESS_{RMS}$ and $RMSE$ for each basic surrogate for the Branin-Hoo function (numbers on the ordinate indicate the surrogate as in Table 5.8).


(a) Branin-Hoo, 12 points.


(b) Branin-Hoo, 18 points. Actual evaluation phase, without update of surrogates.


(c) Branin-Hoo, 18 points. Actual evaluation phase, with update of surrogates.


(d) Hartman6, 56 points.


(e) Hartman6, 70 points. Actual evaluation phase, without update of surrogates.

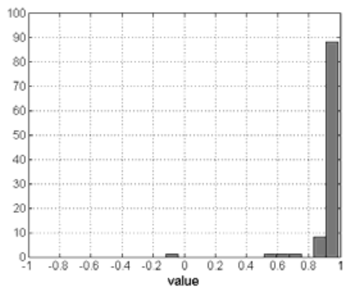
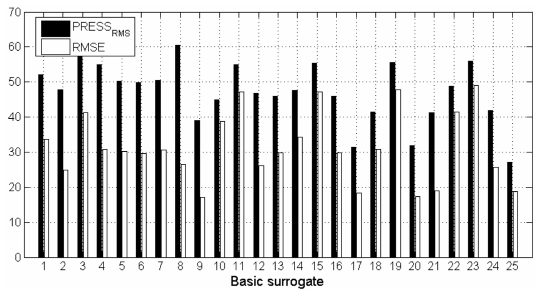
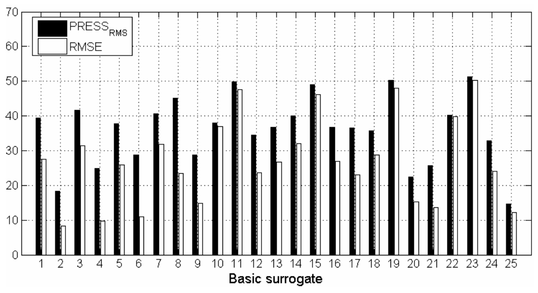

(f) Hartman6, 70 points. Actual evaluation phase, with update of surrogates.

Figure 5.14. Frequency of the correlation between $PRESS_{RMS}$ and $RMSE$ (out of 100 experiments) for the Branin-Hoo (low-dimension) and Hartman6 (high-dimension).

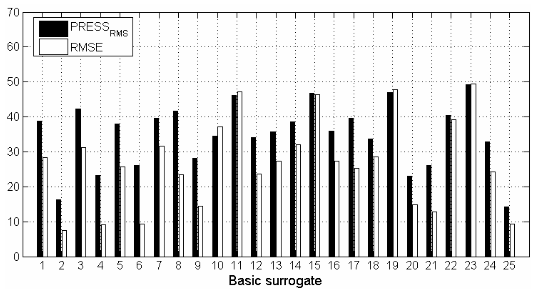
Figure 5.15 complements Figure 5.14, showing the median of $PRESS_{RMS}$ and $RMSE$ over the 100 DOEs.



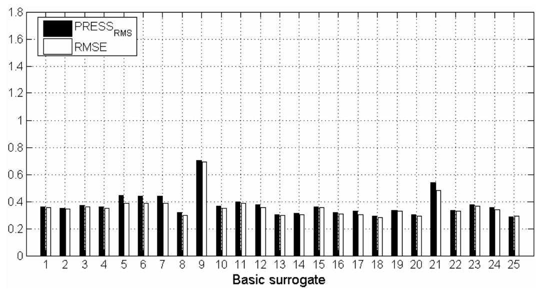
(a) Branin-Hoo, 12 points.



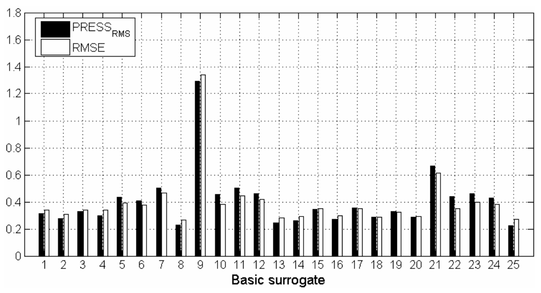
(b) Branin-Hoo, 20 points. Actual evaluation phase, without update of surrogates.



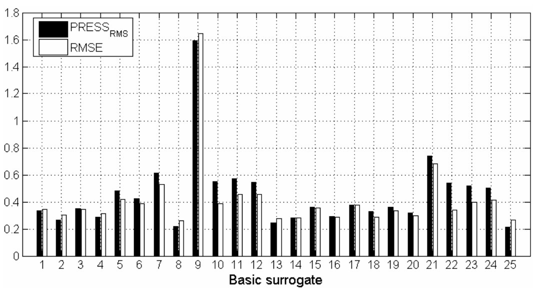
(c) Branin-Hoo, 20 points. Actual evaluation phase, with update of surrogates.



(d) Hartman6, 56 points.



(e) Hartman6, 70 points. Actual evaluation phase, without update of surrogates.



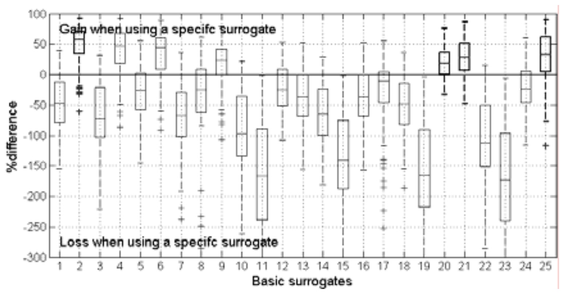
(f) Hartman6, 70 points. Actual evaluation phase, with update of surrogates.

Figure 5.15. Median over 100 DOEs of $PRESS_{RMS}$ and $RMSE$ (lower values indicate better fits) for basic surrogates for the analytical examples (numbers on the ordinate indicate the surrogate as in Table 5.8; the 25th surrogate is $BestPRESS$).

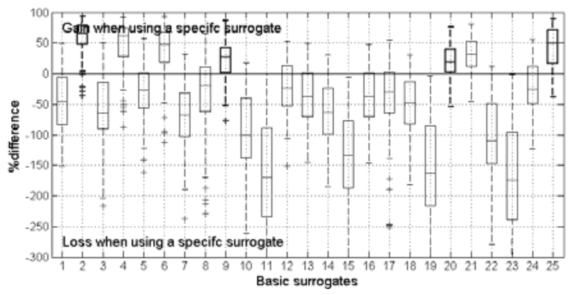
Once again, while there is no significant difference on the strategies, the agreement between both criteria improves with increasing number of points. Figure 5.15 and Figure 5.14 show that not only the correlation is good but also the actual values of $PRESS_{RMS}$ and $RMSE$ agreed well, especially for an increasing number of points. Altogether, $PRESS_{RMS}$ is good just for filtering out bad surrogates when there are few points (low-dimensional problems, i.e., 2 and 3 variables), and it can identify the sub-set of the best surrogates when there are more points available. Up to this point, while the benefits of the DOE refinement are already clear, there is no significant difference in performance when the two different strategies illustrated in Figure 4.4 are compared. In addition, since the identity of the best surrogate is DOE and problem dependent, it is also clear the advantage of using a large set of surrogates and then $BestPRESS$ as a way to prevent against bad fitted surrogates.

Next, the improvement over the $BestRMSE$ by adding points is studied. Figure 5.16 and Figure 5.17 illustrates the $\%difference$ for all basic surrogates and $BestPRESS$ for the examples of the Branin-Hoo and Hartman6 functions. For Figure 5.16, test data were generate as specified in Table 5.10. For Figure 5.17, test data were generate as specified in Table 5.11. Contrarily to what can be expected, adding points to the initial DOE does not always improve the performance of the basic surrogates. On the other hand, $BestPRESS$ shows to be not only more robust but also capable to gain with the extra simulations performed. It is also clear that, especially in high dimensions, the strategies for adding points benefit the region of the optima more intensively than the whole design space. Even though this is a trivial observation, it is worth to point out the potential of the $BestPRESS$ for optimization.

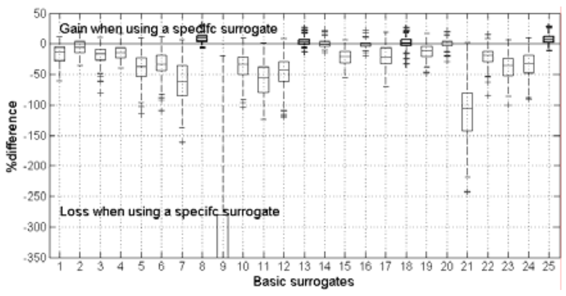
Table 5.13 provides the mean, median and standard deviation of the $\%difference$ in the $RMSE$ for the best 3 surrogates and $BestPRESS$. Test data were generate as specified in Table 5.10. The positive or negative signs of $\%difference$ indicate gain or loss in terms of $RMSE$ for the specific surrogate compared with the best surrogate of the original DOE. In general, $BestPRESS$ is the second best surrogate. It can be observed that (i) over the design space, the gains are limited by both the number of design variables (see the contrast between Branin-Hoo and Hartman6) and the complexity (see the contrast between Branin-Hoo and Camelback); and (ii) at the region of the optima, $BestPRESS$ always presented gains higher than 30%.



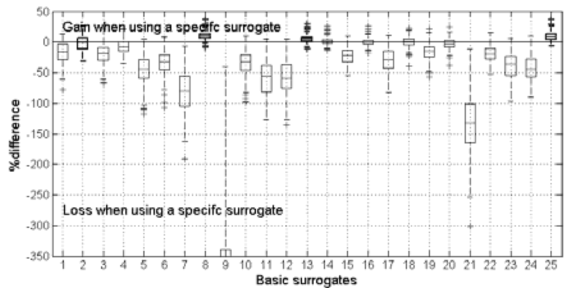
(a) Branin-Hoo, 20 points. Actual evaluation phase, without update of surrogates.



(b) Branin-Hoo, 20 points. Actual evaluation phase, with update of surrogates.

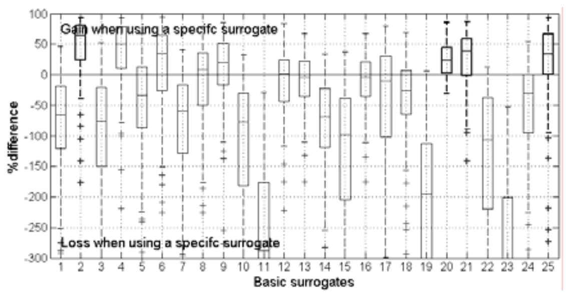


(c) Hartman6, 70 points. Actual evaluation phase, without update of surrogates.

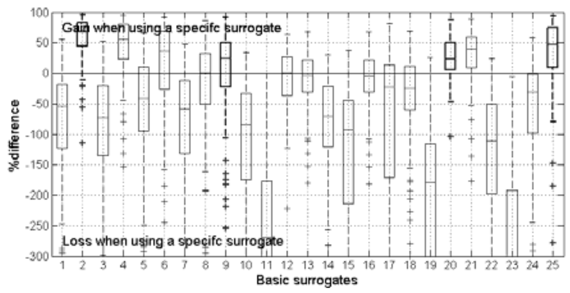


(d) Hartman6, 70 points. Actual evaluation phase, with update of surrogates.

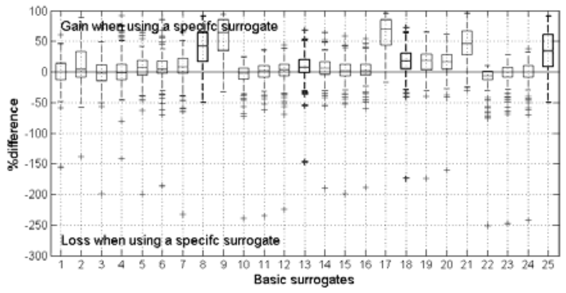
Figure 5.16. $\%difference$ in the $RMSE$, defined in Eq. (0.11), over the design space (numbers on the ordinate indicate the surrogate as in Table 5.8; the 25th surrogate is $BestPRESS$. See Appendix C for details about boxplots).



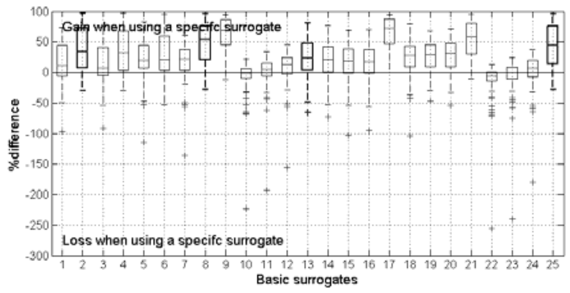
(a) Branin-Hoo, 20 points. Actual evaluation phase, without update of surrogates.



(b) Branin-Hoo, 20 points. Actual evaluation phase, with update of surrogates.



(c) Hartman6, 70 points. Actual evaluation phase, without update of surrogates.



(d) Hartman6, 70 points. Actual evaluation phase, with update of surrogates.

Figure 5.17. $\%difference$ in the $RMSE$, defined in Eq. (0.11), in the region of the optima (numbers on the ordinate indicate the surrogate as in Table 5.8; the 25th surrogate is $BestPRESS$. See Appendix C for details about boxplots).

Table 5.13. $\%difference$ in the $RMSE$, defined in Eq. (0.11), of the best 3 basic surrogates (according to how often they have the best $PRESS_{RMS}$, see Figure 5.13) and $BestPRESS$ (for the basic surrogates, the numbers indicate the identity as in Table 5.8)

Problem	Surrogate	Freq. of best $PRESS_{RMS}$	Over the design space			In the region of the optima		
			Median	Mean	StdDev	Median	Mean	StdDev
Branin-Hoo (without update of surrogates)	2	41	57	50	31	64	47	51
	20	15	19	20	25	23	25	26
	21	12	28	28	29	39	21	78
	BestPRESS	---	33	31	39	34	23	67
Branin-Hoo (with update of surrogates)	2	50	66	60	26	68	57	37
	9	12	27	21	32	24	-8	117
	20	12	19	21	26	23	26	31
	BestPRESS	---	49	43	33	47	35	57
Camelback (without update of surrogates)	1	11	13	11	22	14	-26	115
	5	19	31	29	19	40	24	62
	6	16	16	20	15	64	39	111
	BestPRESS	---	6	8	22	48	36	49
Camelback (with update of surrogates)	1	13	13	10	22	10	-35	123
	5	20	32	30	18	48	29	61
	17	15	-8	-18	41	73	62	32
	BestPRESS	---	6	9	21	46	31	68
Hartman3 (without update of surrogates)	2	49	29	27	24	40	31	56
	4	14	18	14	25	38	25	77
	13	14	13	14	16	-4	-30	87
	BestPRESS	---	21	21	25	33	25	67
Hartman3 (with update of surrogates)	2	46	32	26	29	63	48	46
	8	7	5	-14	88	31	21	63
	13	37	16	17	16	6	-8	79
	BestPRESS	---	24	24	19	34	31	55
Hartman6 (without update of surrogates)	8	67	10	9	7	42	40	30
	13	9	3	3	6	8	8	25
	18	18	2	1	9	18	16	28
	BestPRESS	---	8	7	7	34	35	31
Hartman6 (with update of surrogates)	2	6	-1	-3	13	34	37	35
	8	73	10	10	7	54	48	33
	13	17	4	5	6	23	26	29
	BestPRESS	---	8	9	7	45	44	34

3. Summary and Conclusions

In this application, the use of multiple surrogates, heuristic optimization methods, and extra high-fidelity simulations for minimum $RMSE$ in meta-modeling were explored. This was done by (i) the generation of a large set of surrogates and the use of $PRESS_{RMS}$ as a criterion for surrogate selection; and (ii) the use of heuristic optimization for the generation of extra points for high-fidelity simulation. In addition, two different strategies for updating the surrogates were studied. The first one does not include the update of the surrogates after the inclusion of each point in the data set. Contrarily, the second one updates the set of surrogates every time a new point is included in the data set.

Based on a set of standard test functions, the study leads to the conclusion that the benefits of both strategies depend on dimensionality and number of points:

- $BestPRESS$ is efficient avoiding poorly fitted surrogates and then indicating new points for extra simulation.
- While over the design space, the gains are limited by the number of design variables and the complexity of the function, both approaches presented above are robust in presenting gains in the region where the optima are located.

Additionally, there is no significant difference between the strategies used for adding points. This will be a subject for further research on this topic.

5.3. Combinatorial Optimization

5.3.1. On How to Implement an Affordable Optimal Latin Hypercube

It is worth mentioning that the reported methodology for obtaining Optimal Latin Hypercube was implemented during an exchange program at Vanderplaats Research and Development Inc., from May/2006 to July 2006. The resulting code is now part of the commercial package VisualDOC, one of the company's optimization software. This research was also properly reported in Viana et al. (2007a) and Viana et al. (2007c).

1. The Optimal Latin Hypercube

Chapter II has already discussed the importance of the Latin Hypercube design for computer generated experiments. Unfortunately, generating an Optimal Latin Hypercube design results in a difficult optimization problem that is traditionally solved by time consuming non-gradient based methods, for example a genetic algorithm. Solutions reported in the literature often exceed several hours for large number of points and large number of design variables (Morrsi and Mitchell, 1995; Ye et al., 2000; and Bates et al., 2004). Such a high computational cost limits the practical use of this important design of experiments.

The Optimal Latin Hypercube is obtained through the optimization problem of searching for a design $\mathbf{X}^*$ (a design matrix with n points and n_{dv} variables), which minimizes the objective function $f(\mathbf{X})$:

$$\mathbf{X}^* = \min f(\mathbf{X}) \ . \qquad (0.15)$$

The objective function is formulated to achieve the required uniform space-filling property and, as a result, to avoid situations such as that illustrated in Figure 2.3-(b). In this application, the ϕ_p criterion (Morris and Mitchell, 1995; Jin et al., 2005) was used for the objective function; which leads to the maximization of the point-to-point distance in the design (Johnson et al., 1990). A design is called a ϕ_p-optimal design, if it minimizes:

$$\phi_p = \left[\sum_{i=1}^{s} \Theta_i d_i^{-p} \right]^{1/p} \ , \qquad (0.16)$$

where:

- p is a positive integer, with a very large p, the ϕ_p-criterion is equivalent to the *maximin* distance criterion (Jin et al., 2005);
- d_i's are distinct distance values with $d_1 < d_2 < \ldots < d_s$;
- J_i is the number of pairs of points in the design separated by d_i; and
- s is the number of distinct distance values.

By sorting all the point-to-point distance d_{ij} $(i \geq 1,\ j \leq n,\ i \neq j)$, the distance list $(d_1,\ d_2,\ \ldots,\ d_s)$ and the index list $(\Theta_1,\ \Theta_2,\ \ldots,\ \Theta_s)$ can be obtained. To close the ϕ_p definition, the inter-sited distance can be expressed as follows:

$$d_{ij} = d\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left[\sum_{k=1}^{n_{dv}} \left|x_{ik} - x_{jk}\right|^2\right]^{1/2} . \qquad (0.17)$$

Up to this point, only the optimization problem was defined. However, in order to efficiently obtain an Optimal Latin Hypercube design, more than the problem definition is required. An appropriate optimization algorithm, with an affordable computational cost for evaluating the objective function, is also important. To overcome the high computational cost associated with the existing approaches the method used in this work is supported by (a) an adaptation and an enhancement of a global search algorithm, i.e., the Enhanced Stochastic Evolutionary Algorithm (i.e., ESEA, already presented in Chapter III), and (b) an efficient method for evaluating the objective function to reduce the computational cost.

2. Efficient Approach for Evaluating the Optimality Criterion

Since the objective function is evaluated whenever a new design of experiments is constructed, the efficiency of this evaluation becomes very important for creating the Optimal Latin Hypercube design within a reasonable time frame. Consider the evaluation of the ϕ_p based on Eq. (0.16). It can be seen that this process includes three parts, i.e.:

1. evaluation of all the point-to-point distances;
2. sorting of these distances to obtain a distance list and index list; and
3. evaluation of the ϕ_p value.

However, it can be observed that after an exchange $(x_{i_1k} \leftrightarrow x_{i_2k})$ only elements in rows i_1 and i_2 and columns i_1 and i_2 are changed in the distance matrix, $\mathbf{D}$. Thus, if Eq.

(0.16) could be written to take advantage of this fact, an efficient way for the calculation of ϕ_p is provided. It would avoid unnecessary calculations and the sorting required by Eq. (0.16). In this case, the new ϕ_p is computed by:

$$\phi_p' = \left[\phi_p^p + \sum_{1 \le j \le n, j \ne i_1, i_2} \left(\left(d_{i_1 j}' \right)^{-p} - \left(d_{i_1 j} \right)^{-p} \right) + \sum_{1 \le j \le n, j \ne i_1, i_2} \left(\left(d_{i_2 j}' \right)^{-p} - \left(d_{i_2 j} \right)^{-p} \right) \right]^{1/p} , \tag{0.18}$$

where:

- $d_{i_1 j}' = d_{j_1 i}' = \left[\left(d_{i_1 j} \right)^2 + s\left(i_1, i_2, k, j \right) \right]^{1/2}$;
- $d_{i_2 j}' = d_{j_2 i}' = \left[\left(d_{i_2 j} \right)^2 + s\left(i_1, i_2, k, j \right) \right]^{1/2}$; and
- $s\left(i_1, i_2, k, j \right) = \left| x_{i_2 k} - x_{jk} \right|^2 - \left| x_{i_1 k} - x_{jk} \right|^2$.

Table 5.14 provides an indication of possible savings in computational time when using Eq. (0.18) to evaluate the objective function. T_{full} is the objective function calculated through Eq. (0.16) and $T_{enhanced}$ is the enhanced approach, using Eq. (0.18). Table 5.14 shows the wall-clock time.

Table 5.14. Time comparison between two ways of calculating the objective function, adapted from Jin et al. (2005).

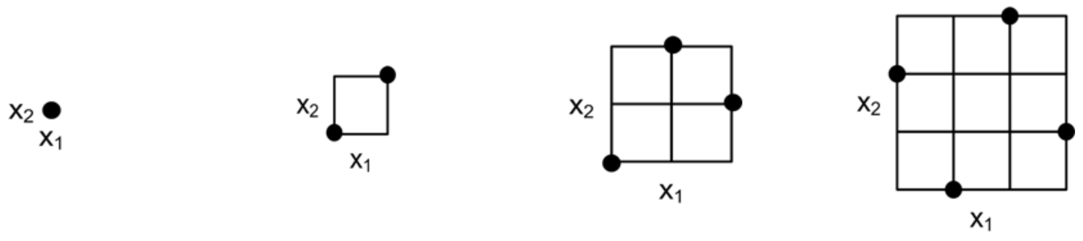
Latin Hypercube	12 x 4	25 x 4	50 x 5	100 x 10
$T_{enhanced}/T_{full}$	0.45	0.19	0.08	0.03

3. Structured Latin Hypercube Design

This section discusses the methodology proposed during this doctoral research for obtaining Optimal Latin Hypercube designs without using formal optimization. Using this methodology, the Optimal Latin Hypercube designs are obtained with minimal computational effort and in real time. The methodology exploits patterns of point locations for Optimal Latin Hypercube designs. Small building blocks with one or several points in each are used to recreate these patterns by taking into account the dimensionality of the problem and the required number of design points.

In its current form, the obtained Structured Latin Hypercube designs provided by this new methodology in two dimensional spaces cannot be improved using non-gradient optimization methods. In higher dimensional design spaces, the Structured Latin Hypercube designs generated using the proposed methodology are still good, but can be improved using non-gradient optimization methods. Even though the designs can be improved for higher dimensional problems, the proposed methodology still provides a powerful and practical tool for obtaining good Optimal Latin Hypercube designs in real time (at most, seconds).

The approach is quite simple and it is based on the hope that the simple n_{dv}-dimensional Latin Hypercube that can be constructed from a n_{dv}-dimensional seed design. Instead of a formal description of the approach, a practical example will be used to explain the methodology of creating a Structured Latin Hypercube design. Consider the case in which is desired a 16×2 Latin Hypercube, i.e., 16 points in 2 dimensions. First, a small Latin Hypercube design will be constructed to be used as a seed in the process. Figure 5.18 shows the some examples of 2-dimensional seed designs. Figure 5.18-(a) shows the seed used in this example. This seed can be as simple as just a $1 \times n_{dv}$ design (where n_{dv} is the number of dimensions of the problem, i.e. the number of design variables).



(a) 1x2 seed design. (b) 2x2 seed design. (c) 3x2 seed design. (d) 4x2 seed design.

Figure 5.18. Examples of seed designs for 2 design variables.

Second, the design space is divided into blocks, in such a way that each dimension is divided in the same number of blocks. The result is that each block can be filled using the seed design (defined previously). It is clear that these processes are inter-dependent. The seed size, i.e., the number of points in the seed design, and the final design size will determine the number of blocks in each dimension. In general, the following relations must be observed:

$$p = n_{blocks} \times n_{seed} \ , \tag{0.19}$$

$$n_{blocks} = (n_{divisions})^{n_{dv}} \ , \tag{0.20}$$

$$n_{seed} = \frac{p}{n_{blocks}} \quad . \tag{0.21}$$

For the seed design from Figure 5.18-(a), Figure 5.19 shows how the 16×2 Latin Hypercube mesh will be divided into blocks. It is important to point out that the fact that each block has four rows and four columns of the Latin Hypercube mesh does not mean that each block will have four points at the end of the process. Instead of that, this is a way to ensure the minimal distance between points in the final Latin Hypercube design.

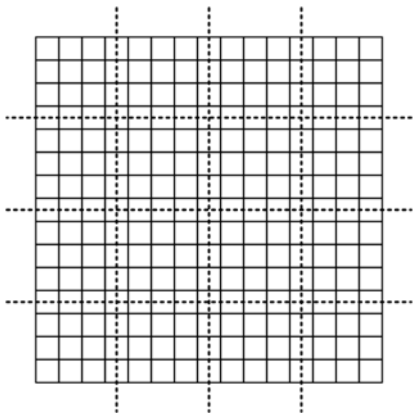


Figure 5.19. 16×2 Latin Hypercube mesh divided into blocks.

The seed design must be properly placed into each of the blocks. Figure 5.20 illustrates how this process works. The first step is to properly scale the “seed design” and then place it at the origin. Next, a set of shifts must be performed. The first one is to shift the seed to consecutive blocks following one of the dimensions. The second one is to shift the origin of the seed inside the mesh of the block. There is a coupling between these two processes. If the block shift is performed in the rows, the seed-origin shift must be performed in the columns, and vice-versa. This process is repeated until one of the dimensions is fulfilled. After that, the whole set of points placed in that dimension can be used to feedback the “shifting” process that continues fulfilling the next dimensions.

The biggest advantage of this approach is that there are no calculations to perform. All operations can be viewed as translations of an n_{seed}-points in the n_{dv}-dimensional hypercube. Although efficient for generating large designs, the previous algorithm fails to provide the flexibility for the number of points that the user may want. The approach described above is limited by the relationship:

$$p = \left(n_{divisions}\right)^{n_{dv}} \quad , \tag{0.22}$$

which follows directly from the Eqs. (0.20) and (0.21) for the smallest seed size of $n_{seed} = 1$. It means that the described algorithm is restricted to have the number of points to be equal to the integer power of the number of design variables.

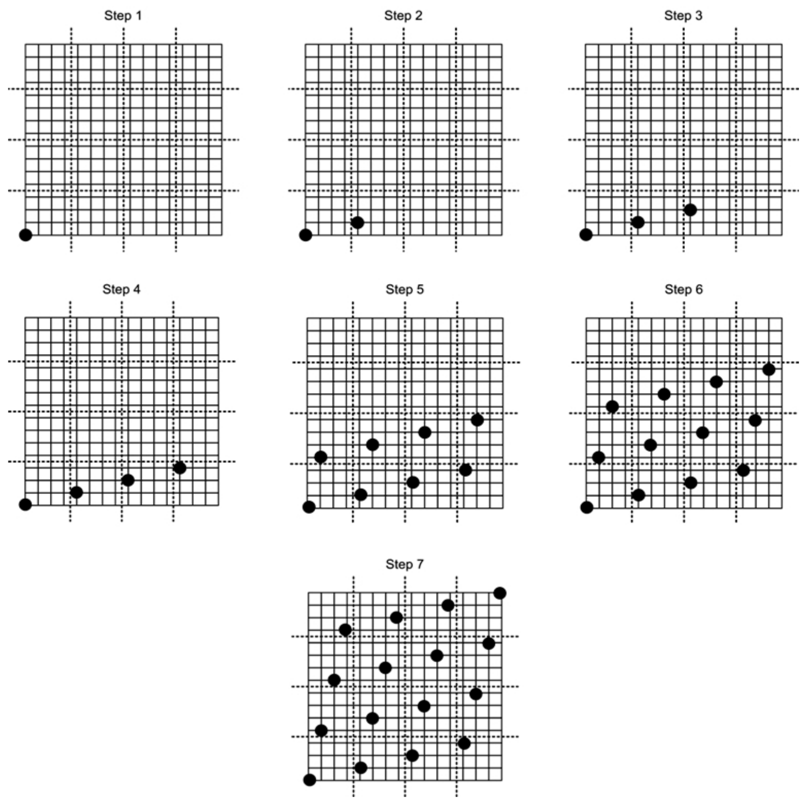
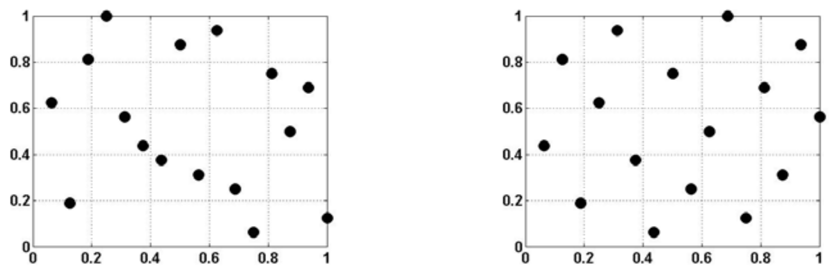


Figure 5.20. The process of creating the 16×2 Structured Latin Hypercube design.

The fact that the algorithm is capable of generating well distributed points that fill the design space well is used to overcome this deficiency. To generate the Structured Latin Hypercube design with any number of points, the first step is to generate the design that has at least the required number of points using the algorithm described above. If after generating the design we obtain the required number of points, the process is completed. If after applying the Structured Latin Hypercube algorithm the number of point is larger than required, a shrinking process is used to reduce the number of points to the desired one. The points are one-by-one removed from the initial Structured Latin Hypercube design discarding points that are the farthest from the center of the hypercube. The reason that such criterion works is that this criterion preserves the kernel of the initially generated design formed by well distributed points.

4. Results and Discussion

Figure 5.21 shows both the initial Latin Hypercube, i.e. before optimization, and the final Optimal Latin Hypercube, which is obtained by solving the optimization problem. The initial Latin Hypercube is a random design with good one-dimensional projective properties (in other words, there is only one point for each level), but with a poor space-filling property. This is typically for Latin Hypercube designs. In contrast, the Optimal Latin Hypercube design maintains the projective property while providing an excellent space filling property.



(a) Initial Latin Hypercube. (b) Final Optimal Latin Hypercube.

Figure 5.21. Latin Hypercube before and after optimization.

The appeal of the proposed methodology for generating Structured Latin Hypercube design is that virtually no computational time is required to create good Latin Hypercube designs. This empirical approach can be used either to quickly obtain a Latin Hypercube design with good space-filling properties or to generate a good starting point for a formal Optimal Latin Hypercube optimization procedure (like ESEA).

Table 5.15 shows the performance comparison of the ESEA procedure of generating Optimal Latin Hypercube designs starting from three different initial designs:

1. the worst design, where the points are located along the diagonal of the design space;
2. Structured Latin Hypercube design;
3. a random initial design.

For all cases, the stopping criterion used was a maximum number of 100 iterations.

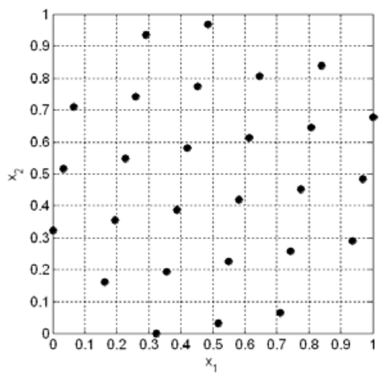
Table 5.15 also allows to directly comparing the values of criterion for the Structured Latin Hypercube design with the Optimal Latin Hypercube design obtained with ESEA.

Note that for the case of two design variables (225×2 and 1024×2), the optimization procedure was not able to improve the ϕ_p-criterion of the Structured Latin Hypercube design. This indicates that for the 2D cases, the proposed empirical methodology of creating the Structured Latin Hypercube design produced the optimum results at no computation cost. For the higher dimensional cases, the formal optimization was able to make relatively small improvements to the ϕ_p-criterion of the Structured Latin Hypercube design. This is an indication that the Structured Latin Hypercube design generated using the proposed methodology provides a reasonable approximation to the Optimal Hypercube designs in high dimensions.

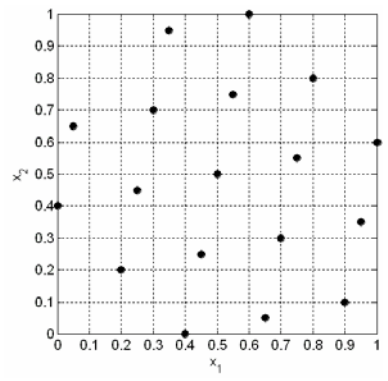
Table 5.15. Optimal Latin Hypercube designs generated using ESEA from three different initial designs.

Design size	**Performance**	**Worst case**	**Structured case**	**Random case**
225×2	Time [s]	143	68	147
	Iterations	251	101	237
	ϕ_p initial	0.5571	0.0752	0.5111
	ϕ_p final	0.0756	0.0752	0.0752
1024×2	Time [s]	11155	3868	7838
	Iterations	284	101	202
	ϕ_p initial	0.5743	0.0353	0.507
	ϕ_p final	0.0421	0.0353	0.0426
256×4	Time [s]	313	372	469
	Iterations	283	336	424
	ϕ_p initial	0.2793	0.0166	0.0385
	ϕ_p final	0.0110	0.0109	0.0108
243×5	Time [s]	609	556	550
	Iterations	547	498	494
	ϕ_p initial	0.2232	0.0130	0.0267
	ϕ_p final	0.0069	0.0068	0.0068
1024×10	Time [s]	34283	27772	29421
	Iterations	601	489	518
	ϕ_p initial	0.2297	0.0044	0.0109
	ϕ_p final	0.0023	0.0023	0.0023

Finally, Figure 5.22 shows how the shrinking process works. In order to generate a 18×2 Latin Hypercube, a 27×2 Optimal Latin Hypercube initially feeds the shrinking algorithm. The space filling property is preserved on the final design.



(a) Initial 27×2 Latin Hypercube.



(b) Final 18×2 Latin Hypercube.

Figure 5.22. Shrinking process.

5. Summary and Conclusions

The application discussed two affordable algorithms for constructing the Optimal Latin Hypercube Design of Experiments. The first one uses the ESEA and an efficient method for evaluating the optimality criteria (ϕ_p). The second one uses an empirical approach to creating Structured Latin Hypercube designs that are good approximations of Optimal Latin Hypercube designs.

Test cases in low dimension show that the Structured Latin Hypercube approach produces designs that could not be improved using a formal optimization approach. In higher dimensions, the formal optimization approach could make small improvements to the designs obtained from the empirical approach. Future work will include both the discussion about the influence of the “seed” on the final design and enhancements on the algorithm in order to work better in higher dimensions.

CHAPTER VI

SUMMARY, CONCLUSIONS AND FUTURE WORK

This chapter summarizes this research work, presents its main conclusions and contributions and discusses the prospects for future work.

As stated in the beginning, the main goal of this contribution was to develop methodologies for applying heuristic optimization techniques to the solution of problems in engineering. Firstly, fundamental concepts in surrogate modeling and numerical optimization were revisited. The set of activities performed during the solution of an optimization problem as well as general concepts on the formulation of the optimization problem as treated by heuristic optimization algorithms were discussed. Then, the heuristic optimization methods used in this research were presented. Besides the basic version of these methods, some theoretical contributions were also introduced. At this point, challenges behind the use of heuristic methods in real world problems, such as the large number of function evaluations and the proper optimization problem formulation, were analyzed. Other than the direct use of actual models, three different approaches for effective management of models with different levels of fidelity were illustrated. They are the direct use of surrogate models, the use of the variable fidelity framework, and the coupling of both actual and surrogate models. It is clear that surrogate models appear as an alternative to alleviate high computational costs associated with heuristic optimization methods. Finally, a set of applications solved during this research showed the success of using heuristic optimization techniques. Important lessons from this work and the scope of future work are briefly presented as follows.

First, a summary of the theoretical contributions to the basic heuristic optimization algorithms is given:

- DE and GA implemented with multiple instances of operators: the developed computational code allows the use of multiple schemes for both DE and GA. For instance, DE can be simultaneously set up with a combination of DE/rand/1/bin and DE/best/1/exp; or GA can be set up with two different selection operators (e.g. roulette and tournament).
- Additional randomness for ACO and DE: the general framework, as presented by Figure 3.6, makes possible to include additional randomness in these two algorithms,

whose basic implementations have limited capabilities of avoiding premature convergence.

- Dispersion measure on the normalized design space: in opposition to schemes based on the function space, the proposed approach prevents difficulties in detecting aggregation of the population when the function space contains several minima. In addition, this is also useful in the context of inverse problems, where besides the objective function, the set of design variables define the solution.
- Implementation of ESEA in an environment of commercial software development to generate the optimal Latin hypercube.

About mixing surrogates, extra actual function evaluations, and heuristic optimization algorithms, during the present doctoral research, two variations of a scheme for updating the surrogates were proposed. The following aspects are explored: (i) large set of different surrogates is used during both off-line and on-line phases of the optimization; (ii) update of the surrogates during the on-line phase of the optimization through a pre-defined number of extra actual function evaluations (high-fidelity analysis); and (iii) $BestPRESS$ (i.e. surrogate with the smallest $PRESS$ value of the set) is used for global search. Remembering that $PRESS$ is used as an estimator of the $RMSE$, the idea is to chose the surrogate with the best global approximation capabilities. The proposed variations differ in the update of the surrogates. While one updates the ensemble every time a new point is included in the data set, the other one performs the updating after the set of new points is generated.

Now, a summary and lessons from the applications:

1. Three-dimensional vehicular structure design optimization: this first application demonstrated the use of heuristic optimization algorithms directly coupled with finite element models to solve the design optimization of a three-dimensional vehicular structure. In the context of this doctoral work, this is an example of direct use of high-fidelity models. It is also important to say that the heuristic algorithms have successfully handled (i) a combination of continuous and discrete design variables, and (ii) constrained multi-criteria optimization. The results were encouraging regarding the possibility of adding system requirements, such as dynamic and manufacture constraints, during the optimization task.

2. Optimization of aircraft structural components by using heuristic algorithms and multi-fidelity approximations: in this application, the coupling of non-linear high-fidelity and linear low-fidelity analyses through the variable fidelity approach allowed the use of

intensive computing heuristic techniques for the optimal design of aircraft structural components. The study allows concluding that the use of variable fidelity together with heuristic optimization methods is a successful approach. The reasons are: (i) they do not require gradient information (which implies that the resources can be directly used for the search, and that there is no propagation of the errors due to the computation of the gradients based on corrected responses); and (ii) they have the trend to find the global or near global solution (which reduces the need of running multiple times the optimization problem).

3. Aircraft longitudinal stability and control derivatives identification by using LifeCycle and Levenberg-Marquardt optimization algorithms: this application presented an identification procedure to determine the longitudinal stability and control derivatives of a military aircraft by coupling together heuristic (global) and classical (local) optimization algorithms within the framework of the Output Error method. The individual performances of both the algorithms were tested and then a cascade-type scheme was proposed aiming at taking advantage of the global and local search capabilities of the individual algorithms. The results were very encouraging in the sense of adding complexity to the models, by embracing non-linearities for example, in further research.

4. Identification of a non-linear landing gear model using heuristic optimization: this application was dedicated to the identification of an aircraft landing gear model by using a heuristic optimization technique. A set of measures that describe the behavior of the system and its numerical counterparts are used together to build a representative functional, which once minimized gives a possible solution for the identification problem. A good match between the theoretical results and those obtained from the experimental drop-test was found. Encouraging future research involving more complex models, such as the ones in which the wheel rotation for the representation of drag loads and the flexibility of the landing gear are to be included (in addition to the two degrees of freedom considered in the present model).

5. Improving surrogate models by coupling heuristic optimization and extra high-fidelity function evaluations: in this application, the use of multiple surrogates, heuristic optimization methods, and extra high-fidelity simulations for minimization of the $RMSE$ in meta-modeling were explored. The key points of the methodology are (i) the generation of a large set of surrogates and the use of $PRESS_{RMS}$ as a criterion for

surrogate selection; and (ii) the use of heuristic optimization for the generation of extra points for high-fidelity simulation. Two different strategies for updating the surrogates were studied (with and without the update of the surrogates after the inclusion of each point in the data set). Based on a set of standard test functions, the study leads to the conclusion that (i) the benefits of both strategies depend on dimensionality and number of points; (ii) $BestPRESS$ efficiently indicates new points for extra simulation; and (iii) while the gains over the design space are limited by the number of design variables and the complexity of the function, the gains in the region of the optima are less sensitive.

6. On how to implement an affordable Optimal Latin Hypercube: the application discussed how to obtain the Optimal Latin Hypercube Design of Experiments by using either the ESEA and an efficient method for evaluating the optimality criteria (ϕ_p), or an empirical approach (creating Structured Latin Hypercube designs, which are good approximations of Optimal Latin Hypercube designs). Test cases in low dimension show that the Structured Latin Hypercube approach produces designs that could not be improved using a formal optimization approach. In higher dimensions, the formal optimization approach could make small improvements to the designs obtained from the empirical approach.

More generally, during this doctoral research the following points were achieved:

- The use of non-conventional methods for system identification was explored. Heuristic algorithms presented a robust alternative to the solution of problems in which experimental data are corrupted or in which the models are not sufficiently known.
- The use of heuristic methods coupled with statistical tools for the solution of design problems was consolidated. This approach has been used (i) as a way to overcome the limitations of classic methods and (ii) as an alternative to decrease the computational costs that are commonly associated with heuristic optimization methods.
- The use of the developed techniques to solve various representative real world problems.
- The implementation of both a general-purpose optimization toolbox (SIMPLE Toolbox) and a surrogate modeling toolbox (SURROGATES Toolbox).

The main learning is that, as it also happens with classical optimization algorithms, the success in using heuristic methods is highly dependent on the formulation of the problem. This also includes the level of fidelity, the quality of information already known, and the computational budget. In this sense, the choice of fidelity levels to be used and the proper handling strategy are both problem dependent. Additionally, since a large number of points of the design space is naturally visited when using heuristic algorithms, it is natural that the proper analysis of the whole set of points represents a complementary phase.

Aiming at amplifying the employment of heuristic optimization algorithms in real world problems, the following suggestions for future research work are given:

1. Heuristic optimization algorithms:

    - Comparison of the performance of different DE schemes against simultaneous use of DE schemes. The same study for GA (regarding GA operators for that matter) should be implemented. At the end of this proposed study, the following question should be answered: what is the best strategy to be followed, a single scheme or a composition of different operators (and how many)?
    - Development of a self-tuning approach for selecting different DE or GA schemes according to the performance in the early stages of the optimization. Independently of single or multiple operators, is it possible to choose the operators based on the first iterations of the optimization task?
    - Study on the effectiveness of the randomness operator for ACO and DE. When use it and when not?
    - LC outperforms the worst suited heuristic for a given problem. However, it also hurts the best one. This way, is it possible to speed up the selection of a heuristic method for a given problem?

2. Heuristic optimization in real world problems: consolidation of these algorithms through different test cases. Based on a larger set of applications, is it possible to build a general procedure that first evaluates the potential of using heuristic algorithms for a specific application and then gives general directions about their use?

3. Optimal Latin Hypercube: discussion about the influence of the “seed” on the final design and enhancements on the algorithm for the Structural Latin Hypercube, in order to work better in higher dimension problems.

4. Surrogate improvement through DOE refinement and heuristic optimization:

    - Enhancements on the update algorithm, especially in the phase of surrogate updates.
    - Application of the proposed approaches to real world problem.

APPENDIX A

SIMPLE OPTIMIZATION TOOLBOX

This is a short appendix about the SIMPLE Toolbox (SimpleToolBox), a set of functions for numerical optimization. The development is part of this PhD. For further reference, please check Viana and Steffen (2008).

A.1. What is the SIMPLE Optimization Toolbox?

SIMPLE Optimization ToolBox (SimpleToolBox) is a software that provides tools for creating, editing and solving general optimization problems. This way, you have access to a set of routines that can be easily used to deal with engineering design, inverse and multidisciplinary optimization problems. The current version includes:

- Multi-objective functions: weighted sum, compromise programming, and weighted min-max implementations.

- Constrained optimization functions: static penalties and dynamic penalties implementations.

- Solvers: Ant Colony Optimization, Differential Evolution, Genetic Algorithm, Particle Swarm Optimization, and LifeCycle Method.

- Report Generators: several plot functions, e.g. stopping criteria, objective statistics, and trace of the best score.

Examples:

- Mathematical functions: single objective, multi-objective, and constrained optimization.
- Design: PID controller, pressure vessel, and welded beam.
- Identification: fault diagnosis, frequency and time domain identification.

Figure A.1 illustrates the functionalities available in SimpleToolBox.

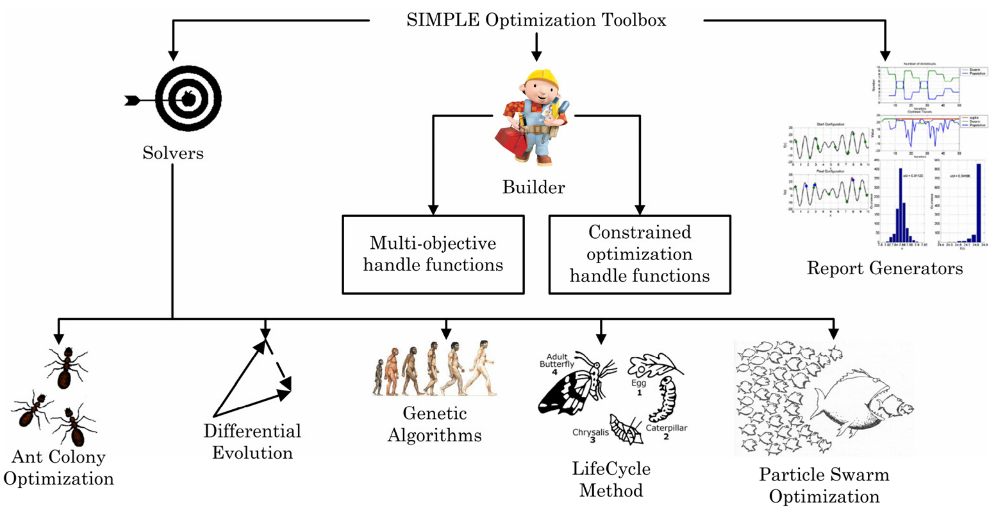


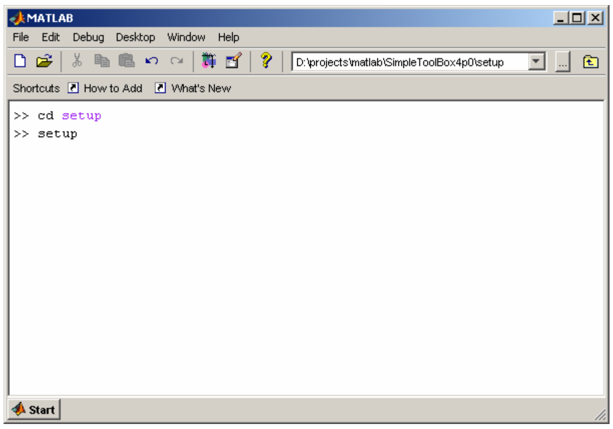
Figure A.1. SimpleToolBox facilities.

A.2. Installation and Uninstallation

Both the installation and the uninstallation processes are quite “simple”. To install SimpleToolBox, the first step is to download from <http://fchegury.110mb.com>. Then unzip the SimpleToolBox, open a MATLAB terminal and go to the directory where it is the SimpleToolBox (for example, `D:projects\matlab\SimpleToolBox4p0`, however, there is no preference for where the user should unzip it). Then type:

```
>> cd setup\
>> setup
```

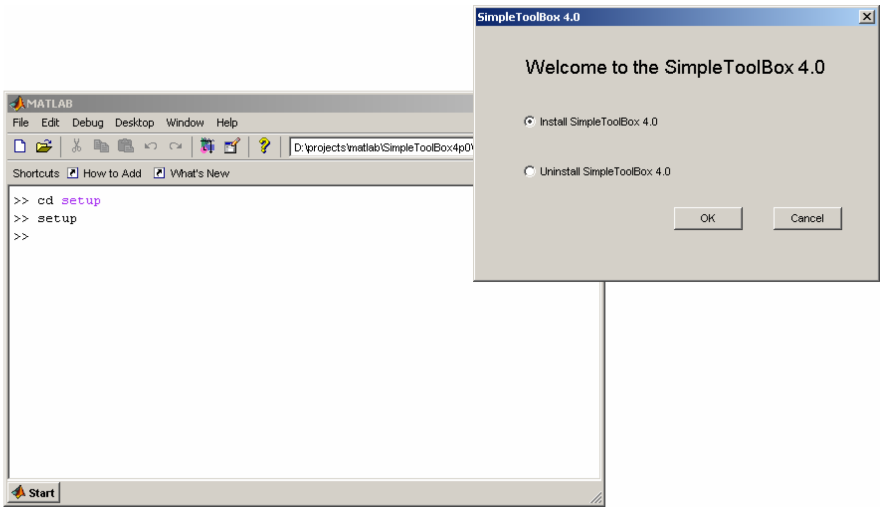
Figure A.2 shows what you would see.



```
>> cd setup
>> setup
```

Figure A.2. First step of installation (or uninstallation).

At this moment, the setup routine will help the user to install (or uninstall) the current version of the SimpleToolBox (see Figure A.3).


MATLAB
File Edit Debug Desktop Window Help
D:\projects\matlab\SimpleToolBox4p0\
Shortcuts How to Add What's New
Start
SimpleToolBox 4.0
OK
Cancel

```
>> cd setup
>> setup
>>
```

Welcome to the SimpleToolBox 4.0

(●) Install SimpleToolBox 4.0

( ) Uninstall SimpleToolBox 4.0

Figure A.3. Second step of installation (or uninstallation).

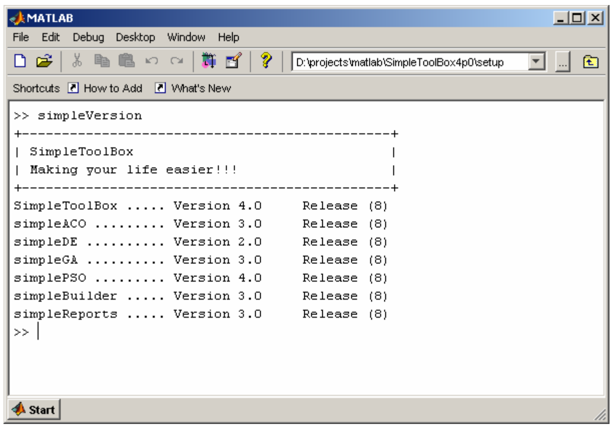
Observations:

1. The SimpleToolBox has been tested in both MATLAB® Version 7.0 Release 14 and MATLAB® Version 6.5 Release 13.
2. In practice, either the installation or uninstallation will change only the MATLAB search path. It means the NO files will be copied or moved.

To check the current version of the SimpleToolBox, open a MATLAB terminal and type:

```
>> srgtsVersion
```

Figure A.4 shows what you see for the current version.


MATLAB
File Edit Debug Desktop Window Help
D:\projects\matlab\SimpleToolBox4p0\setup
Shortcuts How to Add What's New
Start

```
>> simpleVersion
+-----------------------------------------------+
| SimpleToolBox                                 |
| Making your life easier!!!                    |
+-----------------------------------------------+
SimpleToolBox ..... Version 4.0     Release (8)
simpleACO ......... Version 3.0     Release (8)
simpleDE .......... Version 2.0     Release (8)
simpleGA .......... Version 3.0     Release (8)
simplePSO ......... Version 4.0     Release (8)
simpleBuilder ..... Version 3.0     Release (8)
simpleReports ..... Version 3.0     Release (8)
>> |
```

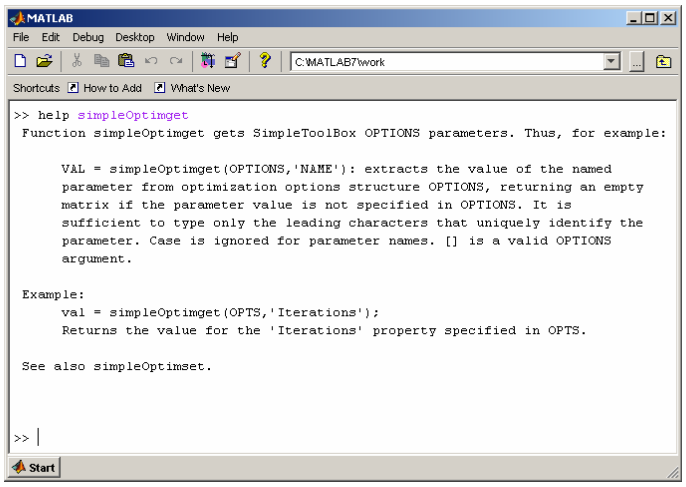
Figure A.4. SimpleToolBox version.

A.3. Help on Functions

All MATLAB functions in toolbox have prefix “simple” (except setup.m). To get help on a specific function just type:

```
>> help function_name
```

See Figure A.5 for an example.


MATLAB
File Edit Debug Desktop Window Help
C:\MATLAB7\work
Shortcuts How to Add What's New
Start

```
>> help simpleOptimget
 Function simpleOptimget gets SimpleToolBox OPTIONS parameters. Thus, for example:

      VAL = simpleOptimget(OPTIONS,'NAME'): extracts the value of the named
      parameter from optimization options structure OPTIONS, returning an empty
      matrix if the parameter value is not specified in OPTIONS. It is
      sufficient to type only the leading characters that uniquely identify the
      parameter. Case is ignored for parameter names. [] is a valid OPTIONS
      argument.

 Example:
      val = simpleOptimget(OPTS,'Iterations');
      Returns the value for the 'Iterations' property specified in OPTS.

 See also simpleOptimset.


>> |
```

Figure A.5. Help on simpleOptimget.

A.4. Bug Reports

To check the history of bug reports, go to BugBuster at <http://fchegury.110mb.com>.

A.5. Simpletoolbox/Surrogatestoolbox User's List

If the user wants to receive periodical information about both the SimpleToolBox and the SimpleToolBox, he/she can join to the SimpleToolBox/SurrogatesToolBox User’s List. It is easy; just send an e-mail to Felipe (fchegury@yahoo.com) with the following data: full name, e-mail address, and affiliation.

By doing that, the user will be informed about the state of development of those toolboxes, including updates on the code, change of standards, tendencies, etc. If the user no longer wants to receive e-mails from this list, he/she just have to send me an e-mail to Felipe (fchegury@yahoo.com) asking to exclude his/her address from the list.

APPENDIX B

SURROGATES TOOLBOX

This is a short appendix about the SURROGATES Toolbox (SurrogatesToolBox), a set of functions for surrogate modeling. The first functions were developed during the PhD work of Dr. Tushar Goel (Goel, 2007). During a exchange program in 2007, the development was conducted together with the present research. For further reference, please check Viana and Goel (2008).

B.1. What is the SURROGATES Toolbox?

SURROGATES Toolbox (SurrogatesToolBox) provides tools for creating and editing different surrogate models (commonly used to replace expensive simulations of engineering problems). The current version includes:

Design of Experiments:

- Face centered cubic design (FCC).
- Combination of a face centered cubic design and a ordinary (but improved) latin hypercube design (FCCLH).
- Mixed-level full-factorial design (FFD).
- Latin hypercube sampling (LHS is especially suitable for very large designs, when the native MATLAB function run out of memory).
- Combination of latin hypercube and D-optimal design (LHDoptimal).
- Filling of user-defined design with a latin hypercube design (LHFilling).
- Selection of points from a user-defined design according to a specific optimality criterion (OptimalSubDoE can use “MaxMin” or “Doptimal” criteria).

Surrogates: Kriging, polynomial response surface, radial basis neural network, support vector regression, and weighted average surrogates.

Performance Metrics:

- Based on test points: correlation coefficient, RMS error, maximum absolute error, and error matrix.
- Based on data points: PRESS and cross-validation errors.

Global Sensitivity Analysis using two different approaches: Monte Carlo and Gaussian Quadrature.

Other capabilities:

- Selection of cross-validation sets following the “k-fold” strategy.
- Generation of a text report file.

Examples:

- Design of experiment examples.
- Braninhoo function.
- Engineering problem 1: Diffuser.
- Engineering problem 2: Radial turbine.
- Data structure update (from previous version to the current version of the toolbox).

Figure B.1 illustrates the functionalities available in SurrogatesToolBox.

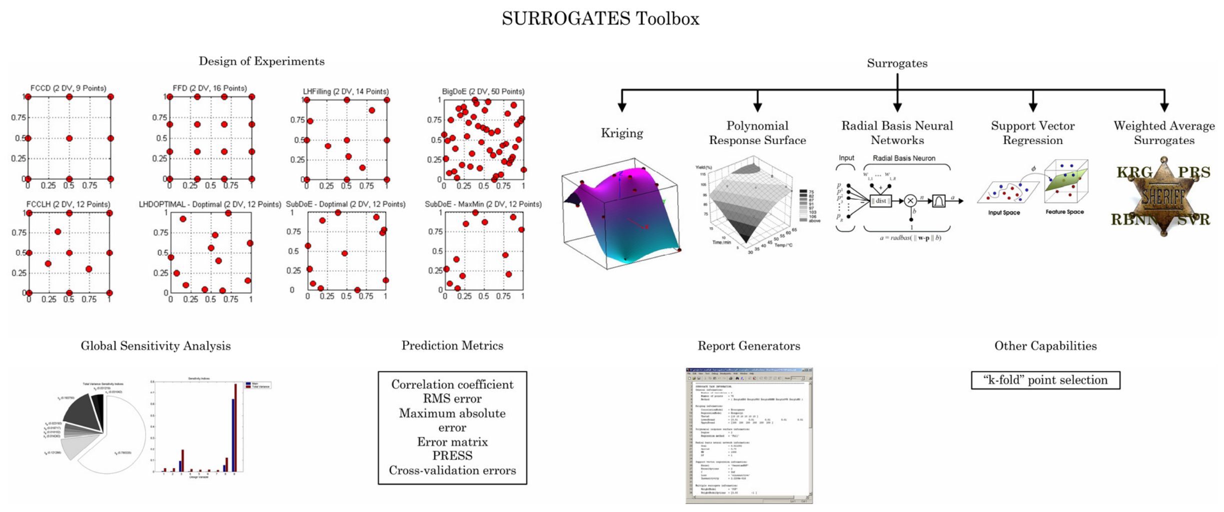


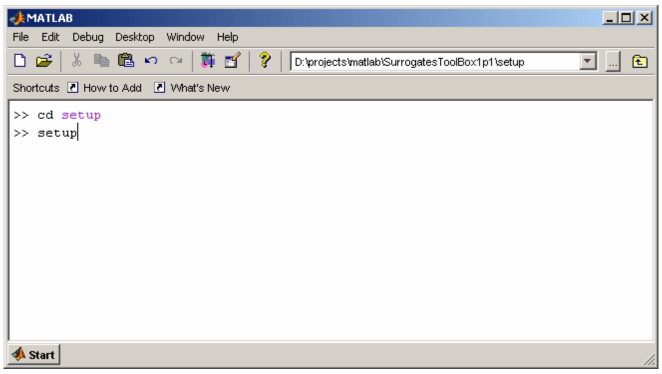
Figure B.1. SurrogatesToolBox facilities.

B.2. Installation and Uninstallation

Both the installation and the uninstallation processes are quite “simple”. To install SurrogatesToolBox, the first step is to download from <http://fchegury.110mb.com>. Then unzip the SurrogatesToolBox, open a MATLAB terminal and go to the directory where it is the SurrogatesToolBox (for example, `D:projects\matlab\SurrogatesToolBox1p1`, however, there is no preference for where the user should unzip it). Then type:

```
>> cd setup\
>> setup
```

Figure B.2 shows what you would see.

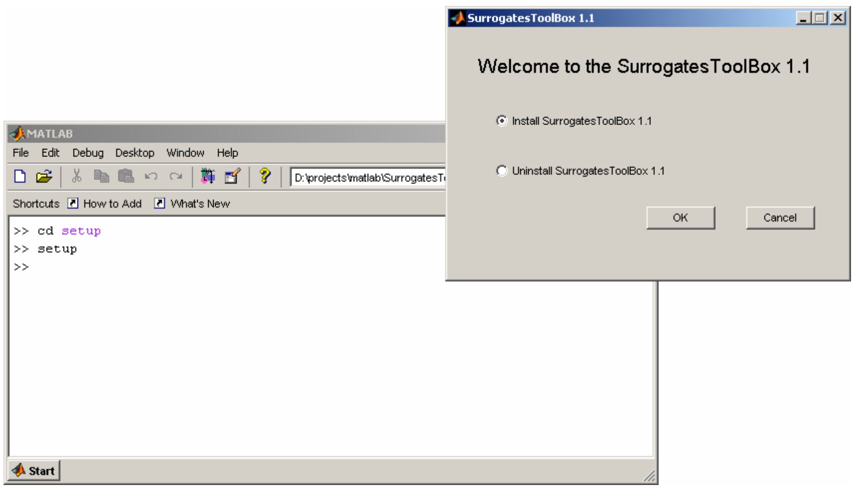

MATLAB
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
Start

D:\projects\matlab\SurrogatesToolBox1p1\setup

```
>> cd setup
>> setup
```

Figure B.2. First step of installation (or uninstallation).

At this moment, the setup routine will help the user to install (or uninstall) the current version of the SurrogatesToolBox (see Figure B.3).


SurrogatesToolBox 1.1
MATLAB
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New
OK
Cancel
Start

```
>> cd setup
>> setup
>>
```

Welcome to the SurrogatesToolBox 1.1

[✓ Install SurrogatesToolBox 1.1]

Uninstall SurrogatesToolBox 1.1

Figure B.3. Second step of installation (or uninstallation).

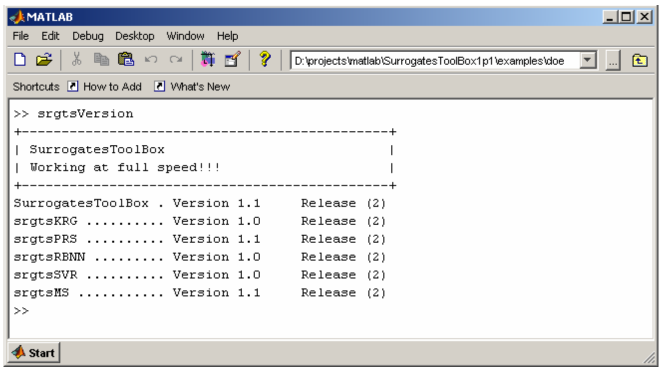
Observations:

1. The SurrogatesToolBox has been tested in both MATLAB® Version 7.0 Release 14 and MATLAB® Version 6.5 Release 13.
2. In practice, either the installation or uninstallation will change only the MATLAB search path. It means the NO files will be copied or moved.

To check the current version of the SurrogatesToolBox, open a MATLAB terminal and type:

```
>> srgtsVersion
```

Figure B.4 shows what you see for the current version.


MATLAB
File Edit Debug Desktop Window Help
D:\projects\matlab\SurrogatesToolBox1p1\examples\doe
Shortcuts How to Add What's New
Start

```
>> srgtsVersion
+-----------------------------------------------+
| SurrogatesToolBox                             |
| Working at full speed!!!                      |
+-----------------------------------------------+
SurrogatesToolBox . Version 1.1     Release (2)
srgtsKRG .......... Version 1.0     Release (2)
srgtsPRS .......... Version 1.1     Release (2)
srgtsRBNN ......... Version 1.0     Release (2)
srgtsSVR .......... Version 1.0     Release (2)
srgtsMS ........... Version 1.1     Release (2)
>>
```

Figure B.4. SurrogatesToolBox version.

B.3. Help on Functions

All MATLAB functions in toolbox have prefix "srgts" (except setup.m). To get help on a specific function just type:

```
>> help function_name
```

See Figure B.5 for an example.

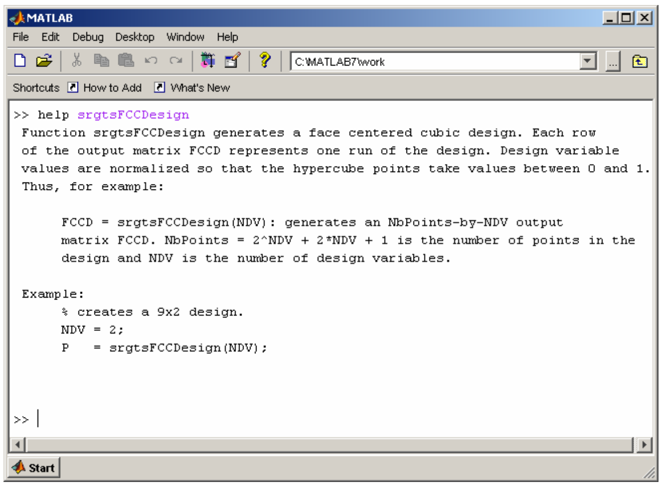
B.4. Bug Reports

To check the history of bug reports, go to BugBuster at <http://fchegury.110mb.com>.

B.5. Simpletoolbox/Surrogatestoolbox User's List

If the user wants to receive periodical information about both the SimpleToolBox and the SurrogatesToolBox, he/she can join to the SimpleToolBox/SurrogatesToolBox User's List. It is easy; just send an e-mail to Felipe (fchegury@yahoo.com) with the following data: full name, e-mail address, and affiliation.

By doing that, the user will be informed about the state of development of those toolboxes, including updates on the code, change of standards, tendencies, etc. If the user no longer wants to receive e-mails from this list, he/she just have to send me an e-mail to Felipe (fchegury@yahoo.com) asking to exclude his/her address from the list.


MATLAB
File Edit Debug Desktop Window Help
C:\MATLAB7\work
Shortcuts How to Add What's New
Start

```
>> help srgtsFCCDesign
 Function srgtsFCCDesign generates a face centered cubic design. Each row
 of the output matrix FCCD represents one run of the design. Design variable
 values are normalized so that the hypercube points take values between 0 and 1.
 Thus, for example:

      FCCD = srgtsFCCDesign(NDV): generates an NbPoints-by-NDV output
      matrix FCCD. NbPoints = 2^NDV + 2*NDV + 1 is the number of points in the
      design and NDV is the number of design variables.

 Example:
      % creates a 9x2 design.
      NDV = 2;
      P   = srgtsFCCDesign(NDV);


>> |
```

Figure B.5. Help on srgtsFCCDesign.

APPENDIX C

BOXPLOTS

In a boxplot, the box is defined by lines at the lower quartile (25%), median (50%), and upper quartile (75%) values. Lines extend from each end of the box to show the coverage of the rest of the data (i.e., they are plotted at a distance of 1.5 times the inter-quartile range in each direction or the limit of the data, if the limit of the data falls within 1.5 times the inter-quartile range). Outliers are data with values beyond the ends of the lines by placing a "+" sign for each point.

See an example given in the MATLAB tutorial (MathWorks Contributors, 2002). The following command lines create a box plot of car mileage grouped by countries.

```
>> load carsmall
>> boxplot(MPG, Origin)
```

Figure C.1 shows the results of the boxplot command.

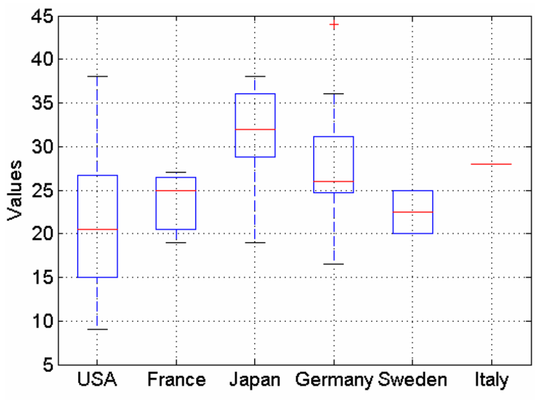


Figure C.1. Example of boxplot figure.

APPENDIX D

NELDER-MEAD SIMPLEX DIRECT SEARCH AND LEVENBERG-MARQUADT OPTIMIZATION METHODS

D.1. Nelder-Mead Simplex Direct Search

As seen in Lagarias et al. (1998), the Nelder-Mead simplex direct search (NMSDS) does not use numerical or analytic gradients. This method uses the simplex concept, which in the n_{dv}-dimensional space is characterized by the $n_{dv}+1$ distinct points in its vertices (e.g., if $n_{dv}=2$, a simplex is a triangle). At each step of the search, a new point in the neighborhood of the current simplex is generated. One of the vertices is replaced by this new point if the new function value is smaller than the values at the vertices of the simplex (which generates a new simplex). This step is repeated until the diameter of the simplex is less than a specified tolerance. The MATLAB *fminsearch* function (Mathworks Contributors, 2002), set with the default options, is used as implementation of the NMSDS algorithm.

D.2. Levenberg-Marquadt Algorithm

The Levenberg-Marquardt method (LM) is a second order variant of the Gauss-Newton method (Gill et al., 1981). This method, although complex, is suitable for a quadratic cost function, and is expected to converge quickly. First, $f(\mathbf{x})$ is approximated by a parabolic function $f_L(\mathbf{x})$ under the condition $\mathbf{x}_L$ (retaining only the 3 first Taylor series terms):

$$f_L(\mathbf{x}) \cong f_L(\mathbf{x}_L) + (\mathbf{x}-\mathbf{x}_L)^T \nabla_{\mathbf{x}}^T f(\mathbf{x}_L) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_L)^T \left[\nabla_{\mathbf{x}}^2 f(\mathbf{x}_L)\right](\mathbf{x}-\mathbf{x}_L). \quad \text{(D.1)}$$

Let the Jacobian of $f(\mathbf{x})$ be denoted by $\mathbf{J}$, then the LM method searches in the direction given by the solution $\mathbf{p}$ to the equations:

$$\left(\mathbf{J}_k^T\mathbf{J}_k + \lambda_k\mathbf{I}\right)\mathbf{p}_k = -\mathbf{J}_k^T f_k \ , \quad \text{(D.2)}$$

where λ_k are nonnegative scalars and $\mathbf{I}$ is the identity matrix.

The LM algorithm can be interpreted in the following manner: for small values of λ_k it behaves as the Gauss-Newton algorithm, while for high values of λ_k it behaves as the steepest gradient algorithm.

In the scope of this work, either NMSDS or LM are employed with initial designs given by the candidate solutions of the heuristic optimizers. This cascade-type scheme using a heuristic algorithm for global search and a classical algorithm for local search is believed to reduce the changes of fail because of local minima.

REFERENCES

AIRD, F. **Race Car Chassis Design and Construction**. Osceola, USA: Motorbooks International, 1997.

ALEXANDROV, N. M.; LEWIS, R. M.; GUMBERT, C. R.; GREEN, L. L.; and NEWMAN, P. A. **Optimization with Variable-fidelity Models applied to Wing Design**, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, USA, 1999, ICASE Technical Report N. 99-49.

ANSYS CONTRIBUTORS, ANSYS ICEM CFD Documentation Manuals, ANSYS Inc, 2006.

BATES, S.; SIENZ, J.; and TOROPOV, V. Formulation of the Optimal Latin Hypercube Design of Experiments Using a Permutation Genetic Algorithm. In: AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. Apr., 2004, Palm Springs, USA. **Proceedings of the 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference**.

BISHOP, C. M. **Neural Networks for Pattern Recognition**. Oxford University Press, 1995. pp. 364-371.

BOX, G. E. P.; HUNTER, W. G.; and HUNTER, J. S. **Statistics for Experimenters**. 1st edition. New York, USA: John Wiley & Sons, 1978.

BRUHN, E. F. **Analysis and Design of Flight Vehicle Structures**. Jacob Pub, 1973.

CANTU-PAZ, E. Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms. **Journal of Heuristics**. Vol. 7, N. 4, pp. 311-334, Jul. 2001.

CHENG, B.; and TITTERINGTON, D.M., 1994, Neural Networks: a Review from a Statistical Perspective. **Statistical Science**, Vol. 9, N. 1, pp. 2-30, Feb., 1994.

CHERKASSKY, V., and MA, Y. Practical Selection of SVM Parameters and Noise Estimation for SVM Regression. **Neural Networks**. Vol. 17, N. 1, pp. 113-126, 2004.

CLARKE, S. M.; GRIEBSCH, J. H.; and SIMPSON, T. W. Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses. **Journal of Mechanical Design**. Vol. 127, Issue 6, pp.1077-1087, Nov. 2005.

COELLO, C. A. C. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. **Knowledge and Information Systems**. Vol. 1, N. 3, pp. 129-156, 1999.

COELLO, C. A. C. Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. **Computer Methods in Applied Mechanics and Engineering**. Vol. 191, N. 11-12, pp.1245-1287, Jan. 2002.

COELLO, C. A. C. An Introduction to Evolutionary Algorithms and their Applications. **Lecture Notes in Computer Science**. Vol. 3563, pp.425-442, Aug. 2005.

CURREY, N. S. **Landing Gear Design Handbook**. Lockheed-Georgia Company, USA, 1984.

DEB, K. **Multi-Objective Optimization Using Evolutionary Algorithms**. 1st edition. Chichester, UK: Wiley, Jun. 2001.

DIXON, L. C. W.; and SZEGÖ, G. P. **Towards global optimization 2**. North-Holland, Amsterdam, The Netherlands, 1978.

DORIGO, M. **Optimization, Learning and Natural Algorithms**. 1992. PhD thesis, Politecnico di Milano, Italy.

DORIGO, M. and GAMBARDELLA , L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. **IEEE Transactions on Evolutionary Computation**, Vol. 1, N. 1, pp. 53-66, Apr. 1997.

DORIGO, M., MANIEZZO, V., and COLORNI, A. The Ant System: Optimization by a Colony of Cooperating Agents. **IEEE Transactions on Systems, Man, and Cybernetics-Part B**. Vol. 26, N. 1, pp.1-13, Feb. 1996.

FLORES, J. E. R.; VIANA, F. A. C.; RADE, D. A.; and STEFFEN Jr, V. Identification of External Forces in Mechanical Systems by using Lifecycle Model and Stress-Stiffening Effect. **Mechanical Systems and Signal Processing**. Vol. 21, Issue 7, pp.2900-2917, Oct. 2007.

FOURIE, P. C. and GROENWOLD, A. A. The Particle Swarm Optimization Algorithm in Size and Shape Optimization. **Structural and Multidisciplinary Optimization**. Vol. 23, N. 4 pp. 259-267, May, 2002.

GANO, S. E.; RENAUD, J. E.; and SANDERS, B. Variable Fidelity Optimization using a Kriging Based Scaling Function. IN: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. Aug. 30 – Sep. 1, 2004, Albany, USA. **Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference**.

GEN, M. and CHENG, R. **Genetic Algorithms and Engineering Optimization**. New York, USA: Wiley-Interscience, Dec., 1999.

GERACE, S. A.; KASSAB, A. J.; and DIVO, E. An Automated Approach to Multiobjective Shape Optimization for Engineering Design Problems. IN: INVERSE PROBLEMS, DESIGN AND OPTIMIZATION SYMPOSIUM, Apr. 16-18, 2007, Miami, USA. **Proceedings of IPDO2007**.

GILL, P. R.; MURRAY, W.; and WRIGHT, M. H. The Levenberg-Marquardt Method. **Practical Optimization**. London: Academic Press, pp. 136-137, 1981.

GOEL, T. **Multiple Surrogates and Error Modeling in Optimization of Liquid Rocket Propulsion Components**. 2007. PhD Thesis, Department of Mechanical and Aerospace Engineering, University of Florida, USA.

GOEL, T.; HAFTKA, R. T.; SHYY, W.; and QUEIPO, N. V. Ensemble of Surrogates. **Structural and Multidisciplinary Optimization**. Vol. 33, N. 3, pp. 199-216, Mar. 2007.

GÓES, L. C. S.; MACIEL, B. C. O.; NETO, N. S. B.; VIANA, F. A. C.; and STEFFEN Jr, V. On the Use of a Global Search Method and a Gradient Based Method for the Identification of Aircraft Longitudinal Stability and Control Derivatives. IN: INVERSE PROBLEMS, DESIGN AND OPTIMIZATION SYMPOSIUM, Apr. 16-18, 2007, Miami, USA. **Proceedings of IPDO2007**.

GOODWIN, G. C. and PAYNE, R. L. **Dynamic System Identification - Experiment Design and Data Analysis**. Academic Press, New York, USA, 1977.

GUNN, S. R. **Support Vector Machines for Classification and Regression**. Image Speech and Intelligent Systems Research Group, University of Southampton, UK, Technical Report, 1998.

HAFTKA, R. T. and GÜRDAL, Z. **Elements of Structural Optimization**. 3rd edition. Dordrecht, Netherlands: Kluwer Academic Publishers, 1992.

HAUPT, R. L. and HAUPT, S. E. **Pratical Genetic Algorithms**. 2nd edition. New York, USA: Wiley-Interscience Publication, May 2004.

ILIFF, K. W. Parameter Estimation for Flight Vehicles. **Journal of Guidance, Control and Dynamics**, Vol. 12, N. 5, pp. 609-622, 1989.

IMAN, R. L. and CONOVER, W. J. Small Sample Sensitivity Analysis Techniques for Computer Models, with an Application to Risk Assessment. **Communications in Statistics, Part A. Theory and Methods**. Vol. 17, pp.1749-1842, 1980.

JIN, R., CHENA, W. and SUDJIANTO, A. An Efficient Algorithm for Constructing Optimal Design of Computer Experiments. **Journal of Statistical Planning and Inference**. Vol. 134, N. 1, pp. 268-287, Sep. 2005.

JIN, Y.; OLHOFER, M.; and SENDHOFF, B. A Framework for Evolutionary Optimization with Approximate Fitness Functions. **IEEE Transactions on Evolutionary Computation**, Vol. 6, No. 5, pp. 481-494, 2002.

JOHNSON, M.; MOORE, L.; and YLVISAKER, D. Minimax and Maximin Distance Designs. **Journal of Statistics Planning and Inference**, Vol. 26, pp. 131-148, 1990.

KAELO, P. and ALI, M. M. A Numerical Study of Some Modified Differential Evolution Algorithms. **European Journal of Operational Research**. Vol. 169, Issue 3, pp.1176-1184, Mar. 2006.

KELLER, G. R. **Hydraulic System Analysis**. Published by the Editors of Hydraulics & Pneumatics Magazine. USA, 1974.

KENNEDY, J. and EBERHART, R. C. Particle Swarm Optimization. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS. Nov. 1995, Perth, Australia. **Proceedings of the 1995 IEEE International Conference on Neural Networks**. pp. 1942-1948.

KOHAVI, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. FOURTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE. Aug. 20-25, 1995, Montréal, Québec, Canada. **Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence**, pp. 1137–1143.

KOSSIAKOFF, A. and SWEET, W. N. **Systems Engineering Principles and Practice**. 1st edition. Hoboken, USA: Wiley-Interscience, Dec. 2002.

KRINK, T. and LØVBERG, M. The LifeCycle Model: Combining Particle Swarm Optimisation, Genetic Algorithms and Hill Climbers. **Lecture Notes in Computer Science**. Vol. 2439, pp.621-630. Feb. 2004.

LAGARIAS, J.C., REEDS, J. A., WRIGHT, M. H., and WRIGHT, P. E. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. **SIAM Journal of Optimization**, Vol. 9 N. 1, pp. 112-147, 1998.

LIANG, K.-H.; YAO, X.; and NEWTON, C. Evolutionary Search of Approximated N-dimensional Landscapes. **International Journal of Knowledge-Based Intelligent Engineering Systems**, Vol. 4, No. 3, pp. 172-183, 2000.

LOPHAVEN, S. N., NIELSEN, H. B., and SØNDERGAARD, J. **DACE – A MATLAB Kriging Toolbox**, Informatics and Mathematical Modelling, Technical University of Denmark, Technical Report IMM-TR-2002-12, 2002.

NEMHAUSER, G. L. and WOLSEY, L. A. **Integer and Combinatorial Optimization**. New York, NY, USA: Wiley-Interscience, 1988.

MARDUEL, X.; TRIBES, C.; and TRÉPANIER, J. Y. Variable-Fidelity Optimization: Efficiency and Robustness. **Optimization Engineering**. Vol. 7, pp.479–500, 2006.

MARLER, R. T. and ARORA, J. S. Survey of Multi-Objective Optimization Methods for Engineering. **Structural and Multidisciplinary Optimization**. Vol. 26, N. 6, pp.369–395, Apr. 2004.

MARTIN, J. D. and SIMPSON, T. W. Use of Kriging Models to Approximate Deterministic Computer Models. **AIAA Journal**. Vol. 43, N. 4, pp.853-863, 2005.

MATHWORKS CONTRIBUTORS, **MATLAB® The language of technical computing**. The MathWorks Inc., 2002.

MCKAY, M.D.; BECKMAN, R.J.; and CONOVER, W.J. A Comparison of Three Methods for Selecting Values of Input Variables from a Computer Code. **Technometrics**, Vol. 21, pp. 239-245, 1979.

MECKESHEIMER, M.; BOOKER, A. J.; BARTON, R. R.; and SIMPSON, T. W. Computationally Inexpensive Metamodel Assessment Strategies. **AIAA Journal**. Vol. 40, N. 10, pp.2053-2060, Oct. 2002.

MICHALEWICZ, Z. Genetic Algorithms, Numerical Optimization and Constraints. In: 6th INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, July 15-19 1995, Pittsburgh, USA. **Proceedings of the 6th International Conference on Genetic Algorithms**. pp. 151-158.

MICHALEWICZ, Z. and FOGEL, D. B. **How to Solve it**: Modern Heuristics. 1st edition. New York, USA: Springer-Verlag, Jan. 2000.

MONMARCHÉ, N.; RAMAT, E.; DROMEL, G.; SLIMANE, M.; and VENTURINI, G. **On the Similarities between AS, BSC and PBIL: toward the Birth of a New Meta-heuristic**, Ecole d'Ingénieurs en Informatique pour l'Industrie, Université de Tours, Technical Report E3i, 215, 1999.

MONTGOMERY, D.C. **Design and Analysis of Experiments**. 4th edition. New York, USA: John Wiley & Sons, 1997.

MORRIS, M. D. and MITCHELL, T. J. Exploratory Designs for Computational Experiments. **Journal of Statistics Planning and Interference**. Vol. 43, pp. 381-402, 1995.

MYERS, R.H. and MONTGOMERY, D.C. **Response Surface Methodology**: Process and Product optimization using Designed Experiments. 2nd edition. New York, USA: John Wiley & Sons, 1995.

NELLES, O. **Nonlinear System Identification**: from Classical Approaches to Neural Networks and Fuzzy Models. New York, USA: Springer-Verlag, Jan., 2001.

NIU, M. C and NIU, M. **Airframe Structural Design:** Practical Design Information and Data on Aircraft Structures. 2nd edition. Adaso Adastra Engineering Center, 1999, pp. 398.

OLIVEIRA, F. C. G. **CONTRIBUIÇÃO AO DESENVOLVIMENTO DE UMA ESTRUTURA VEICULAR TIPO SPACEFRAME USANDO MÉTODO DOS ELEMENTOS FINITOS E MÉTODOS HEURÍSTICOS DE OTIMIZAÇÃO NUMÉRICA**. 2007. MSc Dissertation, School of Mechanical Engineering, Federal University of Uberlândia, Brazil.

OLIVEIRA, F. C. G. ; VIANA, F. A. C.; BORGES, J. A. F.; and STEFFEN Jr, V. Three-dimensional Vehicular Structure Design combining Finite Element Modeling and Numerical

Optimization. IN: 19TH INTERNATIONAL CONGRESS OF MECHANICAL ENGINEERING, Brasilia, Brazil, Nov. 5-9, 2007. **Proceeding of COBEM 2007.**

ONG, Y. S., NAIR, P. B., and KEANE, A. J. Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling. **AIAA Journal**. Vol. 41, No. 4, Apr. 2003.

OWEN, A.B. Orthogonal Arrays for Computer Experiments, integration and visualization. **Statistica Sinica**. Vol. 2, pp.439-452, 1992.

PAPADIMITRIOU, C. H. and STEIGLITZ, K. **Combinatorial Optimization: Algorithms and Complexity**. Englewood Cliffs, New Jersey, USA: Prentice-Hall, Inc, 1998.

PARSOPOULOS, K. E. and VRAHATIS, M. N. Recent approaches to global optimization problems through Particle Swarm Optimization. **Natural Computing**. Vol. 1, N. 2-3, pp. 235-306, Jun. 2002.

POURTAKDOUST, S. H. and NOBAHARI, H. An Extension of Ant Colony System to Continuous Optimization Problems. **Lecture Notes in Computer Science**. Vol. 3172/2004, pp. 294-301, Nov., 2004.

PRICE, K, STORN, R., and LAMPINEN, J. **Differential Evolution**: A Practical Approach to Global Optimization. 1st edition. New York, USA:Springer, Dec. 2005.

QUEIPO, N. V.; HAFTKA, R. T.; SHYY, W.; GOEL, T.; VAIDYANATHAN, R.; and TUCKER, P. K. Surrogate-Based Analysis and Optimization. **Progress in Aerospace Sciences**. Vol. 41, N. 1, pp.1-28, Jan. 2005

RASMUSSEN, T. K. **Improving Particle Swarm Optimization by Hybridization of Stochastic Search Heuristics and Self-Organized Criticality**. May 2002. 80 pages. PhD thesis, University of Aarhus, Department of Computer Science, Denmark.

RONKKONEN, J.; KUKKONEN, S.; and PRICE, K. Real-parameter optimization with differential evolution. IN: 2005 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION, Sep. 2-5, 2005, Edinburgh, Scotland. **Proceedings of the 2005 IEEE Congress on Evolutionary Computation**, Vol. 1, pp. 506-513.

SAAB, Y. G. and RAO, Y. B. Combinatorial Optimization by Stochastic Evolution. **IEEE Transactions on Computer-Aided Design**, Vol. 10, pp. 525–535, 1991.

SACKS, J.; WELCH, W. J.; MITCHELL, T. J.; and WYNN, H. P. Design and Analysis of Computer Experiments. **Statistical Science**, Vol. 4, N. 4, pp.409-435, Nov. 1989.

SHI, Y. and EBERHART, R. Parameter Selection in Particle Swarm Optimization. **Lecture Notes in Computer Science**, Vol. 1447, pp. 591-600, 1998.

SIMPSON, T. W.; BOOKER, A. J.; GHOSH, D.; GIUNTA, A. A.; KOCH, P. N.; and YANG, R. J. Approximation Methods in Multidisciplinary Analysis and Optimization: a Panel Discussion, **Structural and Multidisciplinary Optimization**. Vol. 27, N. 5, pp 302-313, Jul. 2004.

SIMPSON, T. W.; POPLINSKI, J.; KOCH, P. N.; and ALLEN, J. K. On the Use of Statistics in Design and the Implications for Deterministic Computer Experiments. IN: DESIGN THEORY AND METHODOLOGY. Sep. 14-17, 1997, Sacramento, USA. **ASME-DETC97/DTM-3881**.

SIMPSON, T. W.; POPLINSKI, J. D.; KOCH, P. N.; and ALLEN, J. K. Metamodels for Computer-based Engineering Design: Survey and recommendations. **Engineering with Computers** , Vol. 17, N. 2, pp 129-150, Jul. 2001.

SMITH, M. **Neural Networks for Statistical Modeling**. 1st edition. New York, USA: Von Nostrand Reinhold, 1993.

SMOLA, A. J. and SCHOLKOPF, B. A Tutorial on Support Vector Regression. **Statistics and Computing**. Vol. 14, N. 3, pp. 199-222, Aug. 2004.

SOCHA, K. ACO for Continuous and Mixed-Variable Optimization. **Lecture Notes in Computer Science**. Vol. 3172/2004, pp. 25-36, Nov., 2004.

STORN, R., On the Usage of Differential Evolution for Function Optimization. IN: 1996 BIENNIAL CONFERENCE OF THE NORTH AMERICAN FUZZY INFORMATION PROCESSING SOCIETY, Jun. 19-22, 1996, Berkeley, USA. **Proceedings of the 1996 Biennial Conference of the North American Fuzzy Information Processing Society**. pp. 519-523.

STORN, R. and PRICE, K. **Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces**. International Computer Science Institute, Berkley, Technical Report TR-95-012, Mar. 1995.

STORN, R. and PRICE, K. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. **Journal of Global Optimization**, Vol. 11, N. 4, pp.341-359, Dec. 1997.

THOMPSON, L. L.; RAJU, S. and LAW, E. H. Design of a Winston Cup Chassis for Torsional Stiffness. IN: MOTORSPORTS ENGINEERING CONFERENCE AND EXPOSITION, Nov. 16-19, 1998, Dearborn, USA. **1998 Motorsports Engineering Conference Proceedings**.

VAN VELDHUIZEN, D. A. and LAMONT, G. B. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. **Evolutionary Computation**. Vol. 8, N.2, pp.125-147, 2000.

VANDERPLAATS, G. N. **Numerical Optimization Techniques for Engineering Design**. 4th edition. Colorado Springs, USA: Vanderplaats Research and Development, Inc., 2005.

VAPNIK, V. The Nature of Statistical Learning Theory. New York, USA: Springer, 1995.

VENKATARAMAN, S. and HAFTKA, R. T. Structural Optimization Complexity: What Has Moore's Law Done for Us? **Structural and Multidisciplinary Optimization**. Vol. 28, N. 6, pp.375-387, Dec. 2004.

VENTER, G. and SOBIESZCZANSKI-SOBIESKI, J. Particle Swarm Optimization. **AIAA Journal**. Vol. 41, N.8, pp.1583-1589, 2003.

VIANA, F. A. C.; BALABANOV, V.; VENTER, G.; GARCELON, J.; and STEFFEN Jr, V. Generating Optimal Latin Hypercube Designs In Real Time. IN: 7th WORLD CONGRESS ON STRUCTURAL AND MULTIDISCIPLINARY OPTIMIZATION, May 21-25, 2007a, Seoul, Korea. **Proceedings of the 7th World Congress on Structural and Multidisciplinary Optimization**. pp.2310-2315.

VIANA, F. A. C. and GOEL, T. **SURROGATES Toolbox User's Guide**. 2008. Disponível em: <http://fchegury.110mb.com>. Acesso em: fevereiro de 2008.

VIANA, F. A. C.; HAFTKA, R. T., STEFFEN Jr, V.; BUTKEWITSCH, S.; and LEAL, M. F. Optimal use of Multiple Surrogate for Reduced RMS Error in Meta-model. IN: 2008a NSF ENGINEERING RESEARCH AND INNOVATION CONFERENCE, Jan. 7-10, 2008a, Knoxville, USA. **Proceedings of 2008 NSF Engineering Research and Innovation Conference**.

VIANA, F. A. C.; HAFTKA, R. T., STEFFEN Jr, V.; BUTKEWITSCH, S.; and LEAL, M. F. Ensemble of Surrogates: a Framework based on Minimization of the Mean Integrated Square Error. IN: AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. Apr., 2008b, Schaumburg, Illinois, USA. **Proceedings of the 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference**.

VIANA, F. A. C., MACIEL, B. C. O., NETO, N. S. B., OLIVEIRA, M. F., STEFFEN Jr, V., and GÓES, L. C. S. Aircraft Longitudinal Stability and Control Derivatives Identification by Using Life Cycle and Levenberg-Marquardt Optimization Algorithms. **Inverse Problems in Science and Engineering**. Accepted for publication, 2008c.

VIANA, F. A. C., OLIVEIRA, F. C. G., BORGES, J. A. F., and STEFFEN Jr, V. Differential Evolution Applied to the Design of a Three-Dimensional Vehicular Structure. IN: IDETC/CIE 2007-ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Sep. 4-7, 2007b, Las Vegas, USA. **Proceedings of IDETC/CIE 2007 ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference**

VIANA, F. A. C. and STEFFEN Jr, V. Natural Methods Applied to Direct and Inverse Problems. IN: Workshop on Nonlinear Phenomena Modeling and their Applications, May 2-4, 2005, Rio Claro, Brazil. **Proceedings of the Workshop on Nonlinear Phenomena Modeling and Their Applications**.

VIANA, F. A. C. and STEFFEN Jr, V. **SIMPLE Optimization Toolbox User's Guide**. 2008. Disponível em: <http://fchegury.110mb.com>. Acesso em: fevereiro de 2008.

VIANA, F. A. C., STEFFEN Jr, V., VENTER, G., and BALABANOV, V. On How to Implement an Affordable Optimal Latin Hypercube. IN: 19th International Congress of Mechanical Engineering: COBEM 2007, Nov. 5-9, 2007c, Brasilia, Brazil. **Proceedings of 19th International Congress of Mechanical Engineering**.

VIANA, F. A. C.; STEFFEN Jr, V.; BUTKEWITSCH, S.; and LEAL, M. F. About the Optimum Design of an Aircraft Pressure Bulkhead by using Multi-Fidelity and LifeCycle Algorithm . IN: INVERSE PROBLEMS, DESIGN AND OPTIMIZATION SYMPOSIUM, Apr. 16-18, 2007d, Miami, USA. **Proceedings of IPDO2007**.

VIANA, F. A. C.; STEFFEN Jr, V.; BUTKEWITSCH, S.; and LEAL, M. F. Optimization of Aircraft Structural Components by using Nature-Inspired Algorithms and Multi-Fidelity Approximations. **Journal of Global Optimization**. Accepted for publication, 2008d.

VIANA, F. A. C.; STEFFEN Jr, V.; ZANINI, M. A. X.; MAGALHÃES, S. A.; and GÓES, L. C. S. Identification of a Non-Linear Landing Gear Model using Nature-Inspired Optimization. **Shock and Vibration**. Accepted for publication, 2008e.

VITALI, R.; HAFTKA, R. T.; and SANKAR, B.V. Multi-fidelity Design of Stiffened Composite Panel with a Crack. **Structural and Multidisciplinary Optimization**, Vol. 23, N. 5, pp. 347-356, Jun. 2002.

VR&D CONTRIBUTORS. **VisualDOC Design Optimization Software, Version 6.0, Getting Started Manual**. Colorado Springs, USA: Vanderplaats Research and Development, Inc., 2006.

WANG G. G. and SHAN, S. Review of Metamodeling Techniques in Support of Design Optimization. **Journal of Mechanical Design**. Vol. 129, pp.370-380, 2007.

WASSERMAN, P. D. **Advanced Methods in Neural Computing**. 1st edition. New York, USA: Van Nostrand Reinhold, May 1993.

YE, K. Q., LI, W. and SUDJIANTO, A. Algorithmic Construction of Optimal Symmetric Latin Hypercube Designs. **Journal of Statistics Planning and Interference**. Vol. 90, pp. 145-159, 2000.

ZANINI, M. A. X.; MAGALHÃES, S. A.; GÓES, L. C. S.; VIANA, F. A. C.; and STEFFEN Jr, V. On the use of Nature-Inspired Optimization Techniques applied to the Model Updating of a Landing Gear. In: ICED2007 - INTERNATIONAL CONFERENCE ON ENGINEERING DYNAMICS, April 16-18, 2007Algarve, Portugal. **Proceedings of ICED2007**.

ZERPA, L.; QUEIPO, N. V.; PINTOS, S.; SALAGER, J. An Optimization Methodology of Alkaline-surfactant-polymer Flooding Processes using Field Scale Numerical Simulation and Multiple Surrogates. **Journal of Petroleum Science & Engineering**. Vol. 47, Issues 3-4, pp.197-208, Jun. 2005.

ZHOU, Z.; ONG, Y. S.; NAIR, P. B.; KEANE, A. J.; and LUM, K. Y. Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization. **IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications And Reviews**. Vol. 37, No. 1, Jan. 2007.

ZITZLER, E. and THIELE, L. Multiobjective Evolutionary Algorithms: a Comparative Case Study and the Strength Pareto Approach. **IEEE Transactions on Evolutionary Computation**. Vol. 3, Issue 4, pp.257-271, Nov. 1999.

VIANA, F. A. C., 2005, **Técnicas de Meta-Modelagem e Métodos Heurísticos de Opimização Aplicados a Problemas de Projeto e Identificação**. 2008. 144 páginas. Tese de Doutorado. Universidade Federal de Uberlândia, Uberlândia, Brazil.

RESUMO EXTENDIDO EM PORTUGUÊS

Avanços na capacidade de processamento computacional popularizaram a otimização numérica como uma ferramenta de engenharia. Contudo, eles aparentemente também favoreceram o aumento na complexidade das simulações. Como resultado, o custo computacional de simulações complexas de alta fidelidade em engenharia dificultam o uso exclusivo de simulações em otimização. Esta pesquisa de doutorado representa um esforço em combinar técnicas de otimização global e meta-modelagem como uma forma de usar racionalmente os recursos computacionais e aumentar o nível de informação obtida durante a tarefa de otimização. Aspectos positivos dos métodos heuristicos de otimização, tais como o potencial para otimização global, facilidade para programação, e robustez foram explorados. Estes métodos também foram combinados com modelos de fidelidade variável, que reduz o esforço computacional intrínseco aos algoritmos heurísticos de otimização. As técnicas mencionadas acima foram usadas na solução dos problemas contínuo-discretos do projeto ótimo de uma estrutura veicular e componentes estruturais aeronáuticos; identificação de derivadas de controle e estabilidade longitudinal de aviões e modelo não linear de trem de pouso; e melhoramento de meta-modelos via adição de simulações; bem como na solução do problema combinatorial do hipercubo latino ótimo. Para a solução de problemas de otimização contínuo-discreta Otimização por Colônia de Formigas (OCF), Evolução Diferencial (ED), Algoritmos Genéticos (AG), Otimização por Enxame de Partículas, e Modelo do Ciclo de Vida foram usados. Para a solução do problema de otimização combinatorial o Algoritmo Estocástico Evolucionário Melhorado (AEEM) foi usado. Como resultado, algumas contribuições teóricas foram introduzidas à versão básica dos algoritmos heurísticos estudados, incluindo: (i) uso simultâneo de diferentes instâncias de operadores para o ED e AG; (ii) aleatoriedade adicional para OCF e ED; (iii) operadores de aleatoriedade disparados por uma medida de dispersão; e (iv) implementação do hipercubo latino ótimo em um ambiente de desenvolvimento de software comercial. Ao final da pesquisa, a principal lição é que, assim como também acontece com algoritmos clássicos de otimização, o sucesso no uso de métodos heurísticos é altamente dependente do problema, nível de fidelidade das simulações, nível das informações já conhecidas, e, obviamente, recursos computacionais. Desta forma, o uso de fidelidade variável juntamente com métodos heurísticos de otimização é uma estratégia bem sucedida, uma vez que métodos heurísticos não requerem informação sobre o gradiente (isto é, os recursos são diretametne usados na busca e não há propagação dos erros devido ao compto dos gradientes); e eles têm a tendência em encontrar a solução global ou próximo dela. Em alguns casos, uma combinação em cascata de métodos de otimização heurísticos e clássicos pode compor uma estratégia viável para aproveitar as capacidades de busca global e local dos algoritmos individuais. Ao acoplar simulações de alta fidelidade e otimização heurística para o melhoramento de meta-modelos, as estratégias propostas neste trabalho de doutorado indicam eficientemente novos pontos para simulações; e enquanto os ganhos sobre o espaço de projeto são limitados pelo número de variáveis e pela complexidade da função, os ganhos na região dos ótimos são menos sensíveis. Finalmente, uma vez que um grande número de pontos no espaço de projeto é naturalmente visitado no uso de algoritmos heurísticos, é natural que uma análise apropriada do conjunto de pontos seja uma fase complementar ao final da otimização.

Palavras-Chave: Métodos heurísticos de otimização, meta-modelagem, otimização de projeto, problemas inversos, modelagem por fidelidade variável.