National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

On the Optimal Use of Metamodel-Assisted Evolutionary Algorithms in Aerodynamic Applications

Diploma Thesis

Michalis Dimitrios

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2022

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my professor, K. Giannakoglou for guiding me during the process of completing this diploma thesis and for trusting me with this particular subject. I would further like to thank him for his undying trust in me throughout this strenuous process. I would also like to thank Dr. Varvara Asouti who provided me with her valuable help and constant support, despite her busy schedule.

Finally, I thank all my friends for being there for me throughout my studies and my family for supporting me for all those years.

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

On the Optimal Use of Metamodel-Assisted Evolutionary Algorithms in Aerodynamic Applications

Diploma Thesis

Michalis Dimitrios

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2022

Abstract

In this diploma thesis, the implementation of Metamodel-Assisted Evolutionary Algorithms (MAEAs) in the optimization process of various common engineering cases is tested. Two main MAEA-based optimization methods are utilized and are affiliated with the approach followed in the training of the metamodels, i.e. on-line and off-line training. Both these methods are implemented using an external, Python-based software, called Surrogate Model Toolbox (SMT), and are compared to plain EAs in terms of efficiency and computational cost. The optimization is carried out using EASY (Evolutionary Algorithm SYstem) in-house software that is developed by the Parallel CFD & Optimization (PCOpt) Unit of NTUA. From the various built-in surrogate models found in EASY, Radial Basis Functions (RBFs) are utilized in this thesis. However, the optimization via the use of EASY can be additionally assisted by external metamodels, which are found in SMT software. From those external surrogate models, namely Kriging, its applications in reduced design space using Partial Least Squares, i.e. KPLS and KPLSK, and RBFs are utilized in this thesis. Each optimization method is initially implemented in two simple pseudo-engineering optimizattion problems, i.e. welded beam and speed reducer case, and subsequently in the shape optimization of a 2D isolated airfoil. The Reynolds Averaged Navier-Stokes equations of compressible flows are solved using PUMA (Parallel solver, for Unstructured grids, for Multi-blade row computations, including Adjoint) CFD solver that is developed by PCOpt/NTUA.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Περί Βέλτιστης Χρήσης Μεταπροτύπων στους Εξελικτικούς Αλγορίθμους με Εφαρμογές στην Αεροδυναμική

Διπλωματική Εργασία

Μιχάλης Δημήτριος

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2022

Περίληψη

Στο πλαίσιο αυτής της διπλωματικής εργασίας μελετάται η εφαρμογή εξελικτικών αλγορίθμων (EAs) υποβοηθούμενων απο μεταμοντέλα (Metamodel-Assisted Evolutionary Algorithms MAEAs) σε διάφορες εφαρμογές μηχανολογικού ενδιαφέροντος. Δύο είναι οι κύριες μέθοδοι βελτιστοποίησης με εφαρμογή των MAEAs και σχετίζονται με τον τρόπο εκπαίδευσης των μεταμοντέλων, δηλαδή συνδεδεμένα (on-line) και αποσυνδεδεμένα (off-line) από την εξέλιξη. Και οι δύο αυτοί μέθοδοι εφαρμόζονται με τη βοήθεια ενός εξωτερικού λογισμικού με βάση την Python, που ονομάζεται Surrogate Model Toolbox (SMT), και συγκρίνονται με τους κοινούς EAs με βάση την αποτελεσματικότητα και το υπολογιστικό κόστος που προκύπτει από τη χρήση τους. Η βελτιστοποίηση σε κάθε περίπτωση πραγματοποιείται με τη χρήση του EASY (Evolutionary Algorithm SYstem), ενός λογισμικού που αναπτύχθηκε από τη Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης (ΜΠΥΡΒ) του ΕΜΠ. Από τα διάφορα ενσωματωμένα μεταμοντέλα που υπάρχουν στον EASY, οι συναρτήσεις ακτινικής βάσης (Radial Basis Functions (RBFs) χρησιμοποιούνται στην παρούσα διπλωματική εργασία. Ωστόσο, η βελτιστοποίηση μέσω του EASY μπορεί να υποβοηθηθεί από εξωτερικά μεταπρότυπα, τα οποία είναι διαθέσιμα στο SMT. Από αυτά τα εξωτερικά μεταμοντέλα, σε αυτή τη διπλωματική εργασία γίνεται χρήση κυρίως του Kriging, των παραλλαγών του για χώρο σχεδιασμού μειωμένων διαστάσεων χάρη στην εφαρμογή της μεθόδου μερικών ελαχίστων τετραγώνων (Partial Least Squares), κυρίως του KPLS και του KPLSK, καθώς και των RBFs. Κάθε μια απο τις μεθόδους βελτιστοποίησης εφαρμόζεται σε πρώτο στάδιο σε απλά προβλήματα ψευδο-μηχανικής, κυρίως στην περίπτωση της συγκολλητής δοκού και του μειωτήρα ταχύτητας, ενώ στη συνέχεια στη βελτιστοποίηση μορφής μιας διδιάστατης αεροτομής. Η επίλυση των εξισώσεων Reynolds-Averaged Navier-Stokes συμπιεστού ρευστού γύρω από την αεροτομή γίνεται με τη χρήση ενός CFD επιλύτη, που ονομάζεται PUMA και αναπτύχθηκε από τη ΜΠΥΡΒ/ΕΜΠ.

Acronyms

EA	Evolutionary Algorithm
SOO	Single Objective Optimization
MOO	Multi Objective Optimization
CFD	Computational Fluid Dynamics
MAEA	Metamodel Assisted Evolutionary Algorithm
PSM	Problem-Specific Model
DoE	Desing of Experiments
LCPE	Low-Cost Pre-Evaluation
EASY	Evolutionary Algorithm SYstem
PCOpt	Parallel CFD & Optimization unit
NTUA	National Technical University of Athens
PUMA	Parallel solver, for Unstructured grids, for Multi-blade row computations, including Adjoint
RBF	Radial Basis Function
PLS	Partial Least Squares
SMT	Surrogate Model Toolbox
DB	Database
MDB	Metamodel Database
LH	Latin Hypercube
LHD	Latin Hypercube Design

LHS	Latin Hypercube Sampling
ESE	Enhanced Stochastic Evolutionary
FFD	Full Factorial Design
RMSE	Root Mean Squared Error
NRMSE	Normalised Root Mean Squared Error
BLUP	Best Linear Unbiased Predictor
MSE	Mean Squared Error
COBYLA	Constrained Optimization BY Linear Approximation
RNG	Random Number Generator
RANS	Reynolds Averaged Navier Stokes
GPU	Graphic Processing Unit
NURBS	Non Uniform Rational B-Splines
LTT	Lab of Thermal Turbomachines
CUDA	Compute Unified Device Architecture
MAE	Mean Absolute Error

Contents

Chapter 1

Introduction

Optimization problems arise in every ripple of the scientific spectrum, from economics to engineering. The art of mimicking nature's ability to select the optimal candidate solution from a larger population has captivated the entire scientific field for centuries and, thus, the Evolutionary Algorithms (EAs) were created. EAs are metaheuristic population-based search methods and are inspired by Darwinian evolution. They fall under the greater category of stochastic optimization and do not require the computation of derivatives, unlike the deterministic or gradient-based methods, which allows them to calculate the global minimum more efficiently regardless of the continuity or the differentiability of the objective function. Moreover, stochastic methods and therefore EAs, can be used with great success in both single-objective (SOO) and multi-objective optimization (MOO). The latter is implemented uniquely via stochastic methods with the computation of the Pareto front, which serves as an optical representation of non-dominated candidate solutions.

The merit of EAs lies in the ability to effectively and effortlessly accommodate the problem-specific evaluation software, e.g. Computational Fluid Dynamics (CFD). However, such software is commonly expensive in terms of computational cost and thus the optimization time is severely prolonged. In an attempt to reduce the wall clock time of the optimization the use of surrogate models, or metamodels, is introduced. Metamodels approximate, as accurately as possible, the initial evaluation model/objective function by using a data-driven approach, which is based on statistical analysis of the observed data. They tend to have considerably lower computational cost, although generally they lack in accuracy. The introduction of metamodels results in Metamodel-Assisted EAs (MAEAs)[1].

In order for the surrogate models to be utilized by the evaluation software, they must first be trained to fit the problem-specific evaluation model and are, therefore, classified in two main categories accordingly. Off-line trained surrogate models are built and updated statically, i.e. separately from the evolution. The termination or continuation of the optimization depends on a process which assesses the deviation between the optimal candidate solution obtained by the surrogate model and the one obtained using the exact problem-specific model (PSM). The necessary patterns for the training of this global model are collected via the use of various Design of Experiments (DoE)[3] schemes, e.g. Random [12], Factorial [24, 26, 25] and Latin Hypercube [13, 14].

On the other hand, on-line trained surrogate models are built dynamically, i.e. both metamodels and the exact PSM are implemented in the entirety of the EA

population during the evolution in a well coordinated scheme, which results in the training of a separate metamodel for each individual to be evaluated. Responsible for the selection of promising individuals is the surrogate model, either global or local, via a Low-Cost Pre-Evaluation (LCPE) [28] process that determines which individuals are fit for exact reevaluation using the costly CFD evaluation software. The training process ceases when a user-defined number of evaluations has been performed. In this thesis, optimization via both MAEA methods, i.e. off-line and on-line, is facilitated by EASY (Evolutionary Algorithm SYstem)[27] that is developed by the Parallel CFD & Optimization Unit of NTUA (PCOpt/NTUA).

The effectiveness of both methods, i.e. on-line and off-line, depends heavily on the efficacy of both the selected surrogate model and the training process, i.e. complexity of the process, quality and adequacy of the training data. Therefore, selected metamodels will be evaluated based on various criteria, such as goodness of fit, estimated computational cost, complexity of training and overall robustness. These criteria will be implemented individually in a plethora of surrogate models, namely Radial Basis Functions (RBFs) [43, 44, 45] and Kriging[30, 32], along with its variations in reduced design space using Partial Least Squares (PLS) regression, e.g KPLS[39] and KPLSK[42]. Responsible for the metamodel training, is a Python-based, open-source software called Surrogate Modelling Toolbox (SMT)[2], which is highly efficient in deterministic methods since it offers gradient prediction modules. However, its ability to work just as efficiently with stochastic methods and more importantly with EASY software, makes SMT suitable for accommodating the process of training surrogate models with the potential to replace or update the library of built-in metamodels available in EASY.

The purpose of this diploma thesis is to assess the performance (and way of implementation) of MAEA-based optimization in various applications. The first part of the study is focused on observing the performance of MAEAs w.r.t. conventional stochastic optimization methods; particularly in comparison to EAs. The second part is focused on improving the implementation of MAEAs by selecting surrogate models with enhanced qualities, e.g. reduced training time, improved response and overall model fitting. The quality of each metamodel is tested on various optimization problems of scaling difficulty, ranging from low-dimensional pseudo-engineering optimization roblems, i.e. welded beam and speed reducer case, to airfoil shape optimization with aerodynamic criteria using the CFD solver, called PUMA (Parallel solver, for Unstructured grids, for Multi-blade row computations, including Adjoint) and developed by PCOpt/NTUA.

1.1 Optimization

An optimization process aims at maximizing or minimizing a mathematical function via stochastic or deterministic methods w.r.t n_c constraints. This mathematical function is called objective function and is commonly denoted by $\vec{f}(\vec{\beta}) \in \mathbb{R}^n$. For n objectives a constrained optimization problem can be described as follows:

$$\begin{aligned} min \;\; \vec{f}(\vec{\beta}) = min \left\{ f_1(\vec{\beta}), f_2(\vec{\beta}), \ldots, f_n(\vec{\beta}) \right\} \\ \text{subject to } \; c_j(\vec{\beta}) \leq c_j^{thres} \;\; , j = 1, n_c \end{aligned} \tag{1.1}$$

where c_j^{thres} is the nominal threshold of each constraint imposed by the user for the purpose of the optimization. The input values to the objective function are called design variables and are commonly denoted by:

$$\vec{\beta} = \left[\beta_1, \beta_2, \ldots, \beta_{n_\beta} \right] \tag{1.2}$$

where n_β is their number or interchangeably the number of problem dimensions. Multi-objective optimization (MOO) consists of n objectives:

$$\vec{f}(\vec{\beta}) = \left[f_1(\vec{\beta}), f_2(\vec{\beta}), \ldots, f_n(\vec{\beta}) \right] \tag{1.3}$$

which are often conflicting and thus the minimization of each objective does not yield the optimal minimization of the objective function vector $\vec{f}(\vec{\beta}) \in \mathbb{R}^n$. The most common approach for solving such a problem w.r.t. two objectives is the depiction of the entirety of $\vec{f}(\vec{\beta})$ component values in a mutual plot. The vertical and horizontal axis of such a plot correspond to the range of values of the respective objective and the resulting plot is called Pareto front. A solution in Pareto front is called non-dominated if none of the objectives can be improved in value without degrading some of the other objective. The set of all the non-dominated solutions is the Pareto frontier.

In single objective optimization (SOO), the output of the objective function to a single design variable vector $\vec{\beta} \in \mathbb{R}^{n_\beta}$ input is a scalar quantity:

$$\vec{f}(\vec{\beta}) = f(\vec{\beta}) = f \tag{1.4}$$

1.2 Evolutionary Algorithms

Evolutionary algorithms are inspired by the Darwinian evolutionary theory and they have therefore assimilated its key elements. In complete correspondence the evolutionary process revolves around selecting the predominant/elite individuals from a greater sample. This sample consists of λ offspring which were created by μ parents during a generation of the evolutionary algorithm. The population involved in a generation, denoted by g, is classified in the three aforementioned categories that are denoted by P_a^g, P_λ^g or P_μ^g to refer to elites, offspring or parents respectively. EAs constantly form new generations by updating the three main population groups until convergence is reached. The optimal candidate solution in SOO or the non-dominated ones in MOO are included in the elite population set P_a^g. In order to gain better insight into (μ,λ) EAs, their structure is further decomposed[4]:

EAs-1. **Initialization**
The generation counter g is set to zero, marking the initialization of the algorithm. The main objective of EAs is selecting the optimal candidate in SOO or a set of non-dominated individuals in MOO problems from the offspring population set P_λ^g in each generation. This set is initialized randomly at start of the evolution.

EAs-2. **Offspring evaluation**
Each individual $\vec{\beta}$ in the offspring population P_λ^g is evaluated on the PSM. The outcome of the evaluation $\vec{f}(\vec{\beta})$ is archived in the database (DB).

EAs-3. **Computation of cost/fitness function**
Every individual $\vec{\beta} \in P^g \subset \mathbb{R}^{n_\beta}$ with $P^g = P_a^g \cup P_\lambda^g \cup P_\mu^g$, is assigned a scalar value $\Phi(\vec{\beta})$, where $\Phi(\vec{\beta})$ is a cost or fitness function computed as such:

$$\Phi(\vec{\beta}) = \Phi\left(\vec{f}(\vec{\beta}), \{\vec{f}(\vec{z}) \mid \vec{z} \in P^g \setminus \{\vec{\beta}\}\}\right) \in \mathbb{R} \tag{1.5}$$

In MOO problems sorting algorithms are implemented, namely NSGA[5], SPEA[6], NSGA-II[7] and SPEA-II[8]. Such algorithms assign a cost value $\Phi(\vec{\beta})$ to every vector $\vec{f}(\vec{\beta})$ based on dominance criteria in the objective space. In SOO problems such methods are redundant, since it suffices to compare the values obtained from a single objective function and hence $\Phi(\vec{\beta}) \equiv f(\vec{\beta})$.

EAs-4. **Identification of elites**
The cost/fitness function serves as a metric for the selection of the optimal candidate solution, where lower $\Phi(\vec{\beta})$ values indicate a more suitable candidate solution $\vec{\beta}$ in minimization problems; the opposite applies in maximization problems. Depending on the number of objectives, the currently optimal solution in SOO or a set of non-dominated candidate solutions in MOO are stored in the temporary set P_e.

EAs-5. **Elitism**
The elite population set of the next generation is updated via a process of elitism that is applied to the set $P_{e,a} = P_{\alpha}^{g} \cup P_{e}$. This process dictates the number of of elite individuals to replace the worst offspring in the current population P_{λ}^{g} along with the probability for a random elite to be selected as a parent.

EAs-6. **Parent selection**
The parents in the next generation P_{μ}^{g+1} are selected from the wider set $P_{\mu,\lambda}^{g} = P_{\lambda}^{g} \cup P_{\mu}^{g}$ after they have outperformed other parents in a tournament.

EAs-7. **Crossover and mutation**
The set of offspring in the next generation P_{λ}^{g+1} is subsequently formed via the use of operators that mimic natural evolution, i.e. crossover/ recombination and mutation. Crossover is responsible for the generation of a new offspring by using a recombination of the prominent genetic features of each parent. Mutation on the other hand, alters one or more genetic features in order to introduce diversity in the selected population.

EAs-8. **Termination**
The next generation is now fully formed and the convergence of the process is tested. If the convergence criteria are not yet met, then $g \leftarrow g+1$ and the optimization is repeated beginning from step EAs-2. If however, a predetermined threshold of PSM evaluations has been reached or the elite population remains unchanged for a user-selected number of iterations, then EA terminates.

The aforementioned steps outline the function of EAs and will be subsequently combined into the flowchart form of figure 1.1.

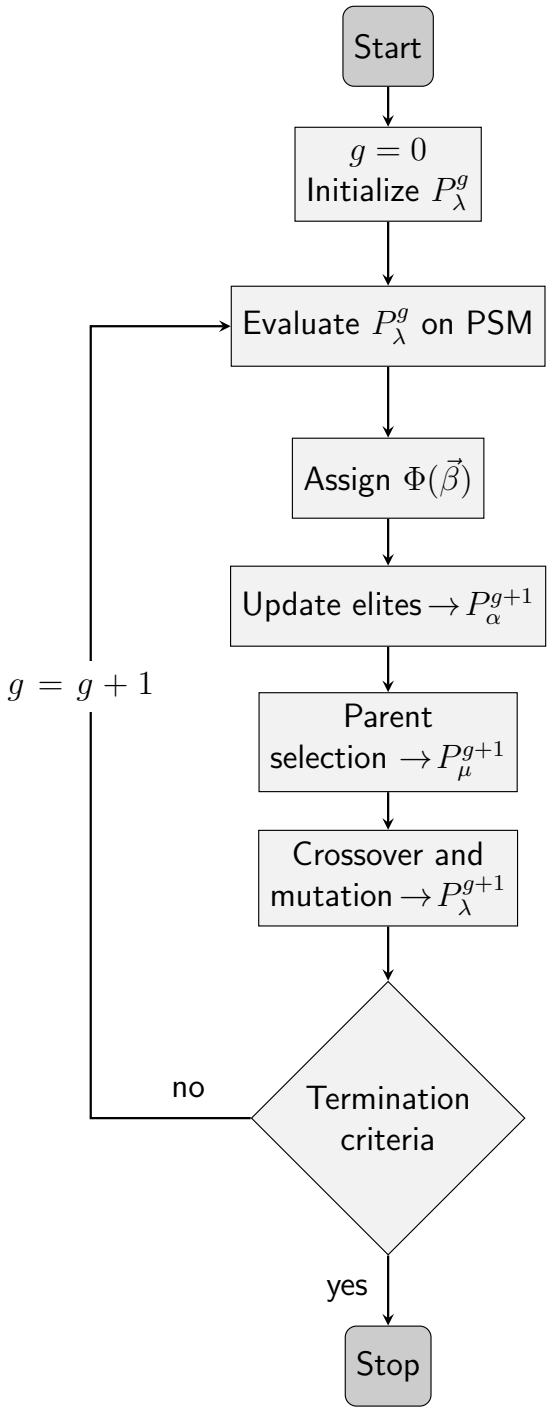


Figure 1.1: Flowchart of Evolutionary Algorithms

Most optimization problems are usually subject to constraints, which EAs handle via one of the following ways or a combination of those:

1. Penalty functions
2. Conversion of constraints into objectives
3. Correlation operators

EASY in particular, mainly uses the first method, which penalizes any value that exceeds a certain threshold. That upper bound of acceptable constraint values is called nominal threshold value c_j^{thres} and is firstly introduced in equation 1.1. Once a constraint exceeds this value ($c_j(\vec{\beta}) > c_j^{thres}$), an exponential penalty function $f_l(\vec{\beta})$ is triggered for each objective function $l \in [1, n]$:

$$f_n(\vec{\beta}) = f_n(\vec{\beta}) + \prod_{j=1}^{n_c} exp\left(\alpha_j \frac{c_j - c_j^{thres}}{c_j^{relax} - c_j^{thres}}\right) \tag{1.6}$$

where n and n_c is the number of optimization objectives and constraints respectively, α_j a user-defined positive constant and c_j^{relax} a user-defined constraint value that is called relaxation threshold value. It is by definition larger than the nominal threshold value $c_j^{relax} > c_j^{thres}$ and is introduced in order to prompt the evolution process from terminating in its early stages, when the candidate solutions commonly defy the imposed constraints. When a candidate solution exceeds the relaxation threshold its fitness function $\Phi(\vec{\beta})$ receives a death penalty i.e. an almost infinitely large value that practically renders the solution unsuitable for further evolution. Equation 1.6 operates when the candidate solutions reside in the $c_j^{thres} < c_j < c_j^{relax}$ range and penalizes them depending on their distance from the nominal threshold value c_j^{thres} (see figure 1.2). Nominal threshold determines, therefore, which solutions are feasible and which are not.

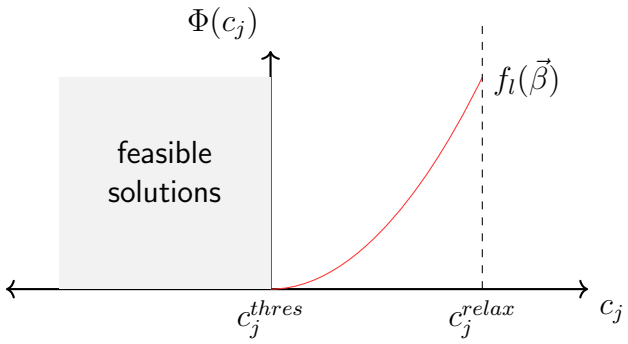


Figure 1.2: Penalisation of feasible and infeasible solutions in EASY

Chapter 2

MAEAs with off-line training

Off-line trained surrogate models are built prior to the evolution and are trained primarily on a dataset of training patterns, which cover the entirety of the design space and are collected via the implementation of various DoE techniques. The training process is disconnected from the evolution and, thus, this method is described as static. MAEAs with off-line training can be decomposed in the following steps:

OFFL-1. **Design of Experiments (DoE)**
One of the various DoE techniques is applied and the sampling process initiates, which involves the selection of n_{doe} observations $\vec{\chi} \in \mathbb{R}^{n_\beta}$ from within the imposed bounds of the design space. Subsequently, the necessary objective function values $\vec{f}(\vec{\chi})$ are computed on the PSM. Consequently, the resulting n_{doe} $(\vec{\chi}, \vec{f}(\vec{\chi}))$ observed pairs are archived in a database reserved for the training of the metamodels which is referred to as metamodel database (MDB). Any untried point in the design space that is not archived in the MDB, i.e. each candidate solution, is denoted by $\vec{\beta} \in \mathbb{R}^{n_\beta}$.

OFFL-2. **Training of the metamodel**
The n_t archived $(\vec{\chi}, \vec{f}(\vec{\chi}))$ pairs are used in the training of the metamodel. In the first optimization cycle, the number of training patterns n_t is equal to the n_{doe} observations. DoE techniques are applied mainly in MAEAs with off-line training and are used to collect training patterns from the entirety of the design space, resulting in the construction of a global metamodel.

OFFL-3. **Implementation of EAs**
The optimization process initiates subsequently via the use of the EASY software that implements EAs. The evolution follows the process described in section 1.2 with one main variation; the offspring evaluation in step 2 (EAs-2) is performed via the use of the trained surrogate model. The metamodel serves as a black box that approximates λ individuals, where λ is the number of offspring in the P_λ^g set, and provides the corresponding prediction of the objective function value $\hat{\vec{f}}(\vec{\beta})$, $\forall \vec{\beta} \in P_\lambda^g$. Each prediction $\hat{\vec{f}}(\vec{\beta})$ is assigned a scalar fitness function value $\hat{\Phi}(\vec{\beta})$, which in MOO problems is based on dominance criteria and $\hat{\Phi}(\vec{\beta}) \equiv \hat{f}(\vec{\beta})$ in SOO. The criterion that prohibits EAs from exceeding a selected number of evaluations is accordingly modified to fit the trivial computational cost of the metamodel.

OFFL-4. **Re-evaluation on the PSM**
The re-evaluation process initiates once the evolution has been completed and the optimal candidate solutions have been found. The best candidate solution in SOO or a set of $\lambda_{\alpha}^{(i)}$ non-dominated solutions in MOO, residing in the $P_e^{(i)}$ temporary set, are re-evaluated using the exact PSM. Index i is used to denote the current cycle of the MAEA algorithm using off-line training.

OFFL-5. **Termination**
The deviation between the metamodel and the PSM evaluated objective function values determines the convergence of the MAEA-based optimization. In case the convergence criteria are not met, the optimal candidate solution/s residing in the $P_e^{(i)}$ set at the end of the evolution or some others arbitrarily selected individuals are used to update the existing MDB. The outcome of their evaluation, i.e. $\vec{f}(\vec{\beta})$, $\forall \vec{\beta} \in P_e^{(i)}$, is subsequently added to the updated MDB and the i_{th} evolution terminates. The next cycle of the optimization initiates starting from step 1 (OFFL-1) and index i is set to $i \leftarrow i+1$. The evolution's inability to yield an optimal solution is indicative of a poorly trained surrogate model and, therefore, an improved metamodel needs to be trained. Consequently in step 1 (OFFL-1) of the $(i+1)_{th}$ cycle, DoE techniques are implemented to select n_{new_doe} new points and in the following step (OFFL-2) a new metamodel is built on n_t training patterns, where:

$$n_t = n_{doe} + \sum_{i=0}^{i} \left(\lambda_{\alpha}^{(i)} + n_{new\_doe} \right) \tag{2.1}$$

where $\lambda_{\alpha}^{(i)}$ is the number of elites selected in the i_{th} generation and n_{new_doe} a user-defined number of sample points that is sampled via DoE techniques at the start of each optimization cycle in order to fill the MDB and improve the fitting of the metamodel. The updated number of sample points will be denoted by $n_{doe}^{'}$ for simplicity, where:

$$n_{doe}^{'} = n_{doe} + \sum_{i=0}^{i} \left(n_{new\_doe} \right) \tag{2.2}$$

The aforementioned steps outline the function of MAEAs with off-line trained metamodels and will be subsequently combined into the flowchart form of figure 2.1.

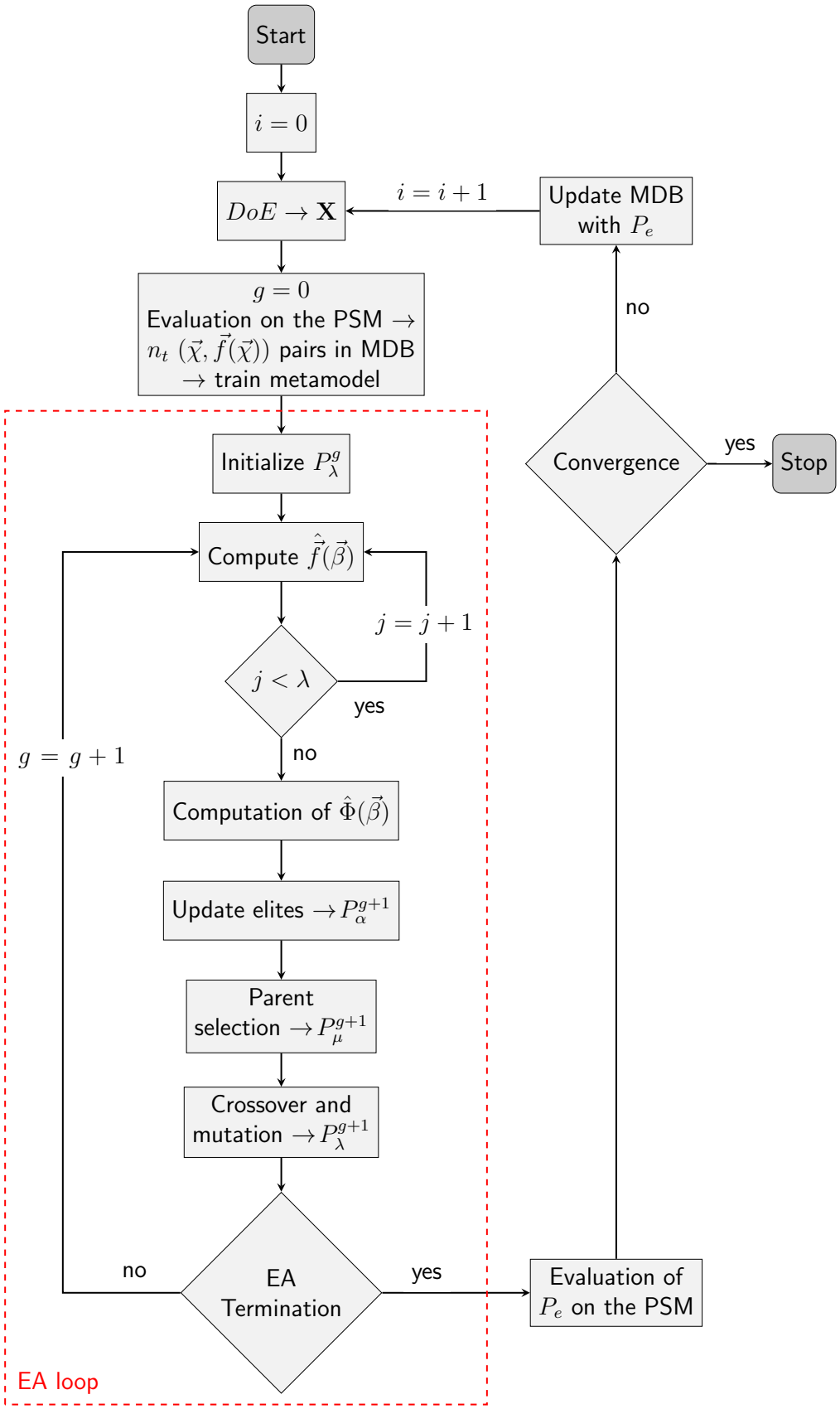


Figure 2.1: Flowchart of MAEAs using off-line trained metamodels

2.1 Design of Experiments (DoE)

The predominant characteristic of off-line trained MAEAs is the construction of a single global surrogate model [9]. The majority of necessary patterns for the training of this global metamodel are collected via the use of various Design of Experiments (DoE) techniques that sample the entirety of the design space. DoE is a statistical tool used for analyzing the interactions between the parameters that effect the performance of a system and controlling them in order to optimize its performance[3, 10, 11]. The most commonly used DoE techniques and the ones studied in this thesis are the following:

1. **Random sampling**
   The most common technique of removing bias from a design is randomization, which gives each sample point $\vec{\chi}=[\chi_1, \chi_2, \ldots, \chi_{n_\beta}] \in \mathbb{R}^{n_\beta}$ equal probability of being selected from the design space [12], as shown in figure 2.2.

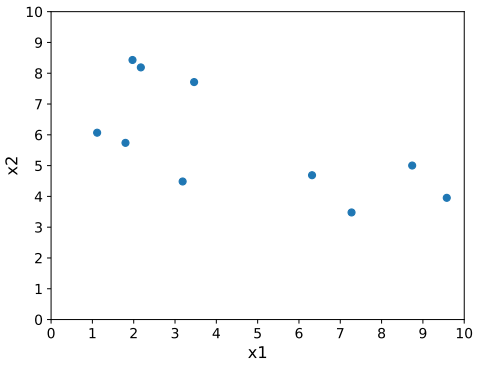


Figure 2.2: Random design in 2D space for $n_{doe}=10$ sample points

2. **Latin Hypercube Sampling (LHS)**
   A square grid containing a single sample point $\vec{\chi} \in \mathbb{R}^2$ per row and column is called a Latin Square. The generalization of this design in $n_\beta > 2$ dimensions results in the creation of a Latin Hypercube (LH)[13]. A Latin Hypercube Design (LHD) aims to improve the coverage of the design space and eliminate the probability of two coinciding sample points and is created via the implementation of LHS[14, 15] scheme. In LHDs, the design space in each dimension is stratified into n_{doe}[1] equiprobable and non-overlapping intervals[14], called strata. Subsequently, n_{doe} distinct values are selected, one from each stratum, and are paired to form the components $\chi_1, \chi_2, \ldots, \chi_{n_\beta}$ of each sample vector $\vec{\chi} \in \mathbb{R}^{n_\beta}$. As a result of the stratification, the LHD consists of n_{doe} distinct sample points and can be written as a $n_{doe} \times n_\beta$ matrix $\mathbf{X} = [\vec{\chi}_1, \vec{\chi}_2, \ldots, \vec{\chi}_{n_{doe}}]^T$, where each component $\vec{\chi}_i = [\chi_{i,1}, \chi_{i,2}, \ldots, \chi_{i,n_\beta}]$ represents an observation $\vec{\chi} \in \mathbb{R}^{n_\beta}$. LHDs can be enhanced with several optimality construction criteria, some of which are presented here [2, 71]:

[1]The original design ($i = 0$) consists of $n_t = n_{doe}$ points, while a separate design is constructed for every n_{new_doe} points sampled. Without loss of generality, we assume from this point forward that in the description of DoE we refer to the original design of n_{doe} points.

(a) **Centered LHD**
This construction criterion centers the selected values from within each hypercube, as shown in figure 2.3.

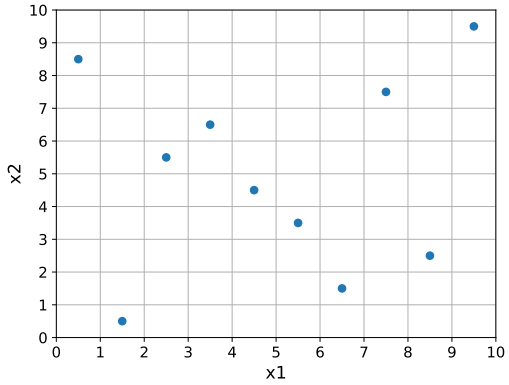


Figure 2.3: Centered LHD in 2D space. The grid has been modified to facilitate the visualization of $n_{doe} = 10$ strata in each dimension.

(b) **Maximin LHD**
This construction criterion was introduced by Johnson et al. [16] based on the idea that the Euclidean distance between sample points should be used as a metric for design construction. A maximin design, denoted by S_{Mm}, guarantees that every pair of points will never coincide by maximizing the minimum distance between them [17].

$$\max_{S \subset \mathbb{R}^{n_\beta}} \min_{\vec{\chi}_i, \vec{\chi}_j \in S} d(\vec{\chi}_i, \vec{\chi}_j) = \min_{\vec{\chi}_i, \vec{\chi}_j \in S_{Mm}} d(\vec{\chi}_i, \vec{\chi}_j) \quad , \forall i, j \in [1, n_{doe}] \tag{2.3}$$

Each selected point $\vec{\chi} \in S$, where S the selected design set, is the center of a sphere, the radius of which is calculated by the algorithm that produces the maximin design described by eq. 2.3. Consequently, the final design contains n_{doe} non-overlapping spheres. In a maximin LHD, the sample points must furthermore be selected from within within each hypercube, as shown in figure 2.4.

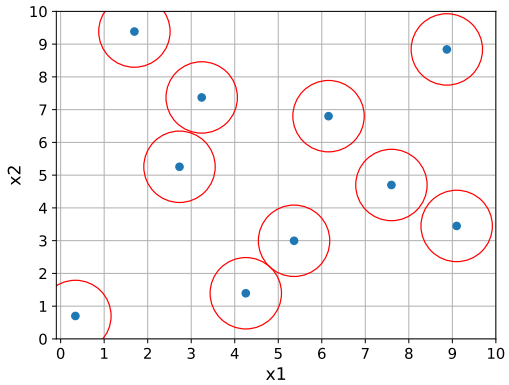


Figure 2.4: Maximin LHD in 2D space for $n_{doe} = 10$ sample points

(c) **Maximin Centered LHD**
Similar to maximin LHD with the exception that the selected sample points are centered within each hypercube, as shown in figure 2.5.

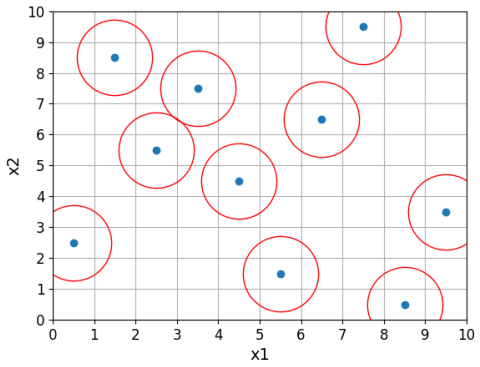


Figure 2.5: Maximin centered LHD in 2D space for $n_{doe} = 10$ sample points

(d) **Maxent LHD**
Information entropy as proposed by Shannon [18] is directly associated to the level of information available from a design. Shewry and Wynn [19] showed that maximizing the entropy of the response distribution at the sampled design sites $\mathbf{X}$ is equivalent to maximizing the gain of information of the response distribution at any untried location of the design space. If the response distribution is given by a stationary Gaussian process $Y(\cdot)$ with mean μ_Y, variance σ^2 and correlation function $R(\cdot)$, then the optimal design $S \subset \mathbb{R}^{n_\beta}$ can be found by maximizing the simplified entropy of the distribution of the responses at the design sites, as such:

$$\max_{\vec{\chi}_i, \vec{\chi}_j \in S} -ln\left[det\mathbf{R}(\vec{\chi}_i, \vec{\chi}_j)\right] \tag{2.4}$$

where $\mathbf{R}(\vec{\chi}_i, \vec{\chi}_j) = \prod_{l=1}^{n_\chi} exp(-\theta_l \left|\chi_{i,l} - \chi_{j,l}\right|^q)$ and q a positive integer with values 1 or 2, corresponding to an exponential or a Gaussian kernel, respectively. The parameters θ_l denote the degree of correlation between training points w.r.t. each design dimension $l \in [1, n_\beta]$. In a maxent LHD, the sample points must furthermore be selected from within within each hypercube, as shown in figure 2.6.

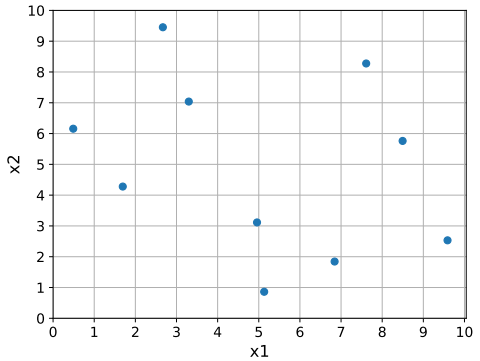


Figure 2.6: Entropy LHD in 2D space for $n_{doe} = 10$ sample points

(e) **Enhanced Stochastic Evolutionary (ESE) LHD**
This criterion is an enhancement to the existing global search, stochastic evolutionary (SE) algorithm, originally developed by Saab and Rao[20]. The need to further reduce the computational cost of SE resulted in the creation of Enhanced Stochastic Evolutionary algorithm (ESE)[21]. This new approach is based on utilizing efficient methods for evaluating various space-filling criteria, namely φ_p, entropy and centered L_2 discrepancy criterion. The first criterion was proposed by Morris and Mitchel (1995)[22] and is an extension of the maximin criterion. L_2 discrepancy is the most common expression of L_p discrepancy, which is a metric of non-uniformity of a DoE. The formula used to describe centered L_2 or CL_2 discrepancy was proposed by Hickernell (1998)[23]. The minimization of CL_2 discrepancy results in a uniform design. ESE combines these three aforementioned space-filling criteria to construct an optimal design. In a ESE LHD, the sample points must furthermore be selected from within within each hypercube, as shown in figure 2.7.

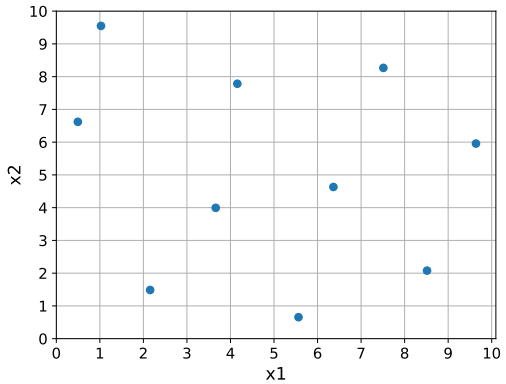


Figure 2.7: ESE LHD in 2D space for $n_{doe} = 10$ sample points

The quality of the LHD affects the convergence of the MAEA-based optimization process and therefore selecting the most cost-efficient construction criterion of an LHD is essential for the success of this method. An analysis performed in appendix A.3, concluded that ESE LHDs are the most suitable for the purpose of this thesis and therefore the LHS scheme is modified accordingly to produce such designs.

3. **Factorial sampling**
   In a factorial design, the relative importance of each design variable (factor) on the objective function is tested by replicating all the possible combinations of the factors. Each possible combination is replicated in a run of the design with a total of n_{doe} runs being performed. Each factor is assigned a number of discrete values in the [-1,1] range, called levels, where high and low influence are assigned a level of 1 and -1 respectively[24]. The change in response caused by an alteration in the level of each factor can, therefore, be correlated with the relative importance of each factor. The complete replicate of a factorial design that contains all possible combinations between n_β factors is called a Full Factorial Design (FFD). A conventional FFD is performed at 2 levels, i.e 1 and -1, which results in $n_{doe} = 2^{n_\beta}$ possible combinations[25]. However, the number of factorial runs n_{doe} is user-defined, i.e. $n_{doe} \neq 2^{n_\beta}$ or $n_{doe} \neq 3^{n_\beta}$, and the cost of constructing a FFD grows exponentially as the number of factors increases. In order to overcome the imposed restrictions, interactions between factors that yield the lowest response are neglected. The resulting design is a fractional factorial design[26]; such a design is depicted in the following case for $n_{doe} = 10$ sample points in figure 2.8:

Levels	**-1**	**-0.5**	**0**	**0.5**	**1**
χ_1	6	7.33	-	8.66	10
χ_2	150	-	175	-	200

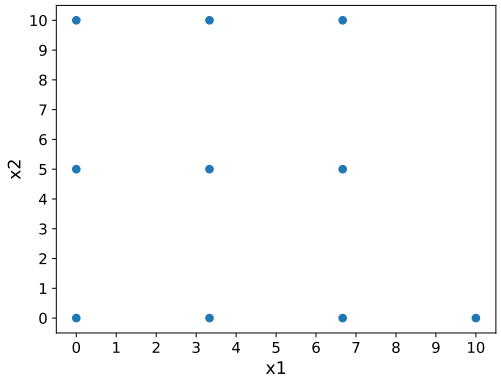
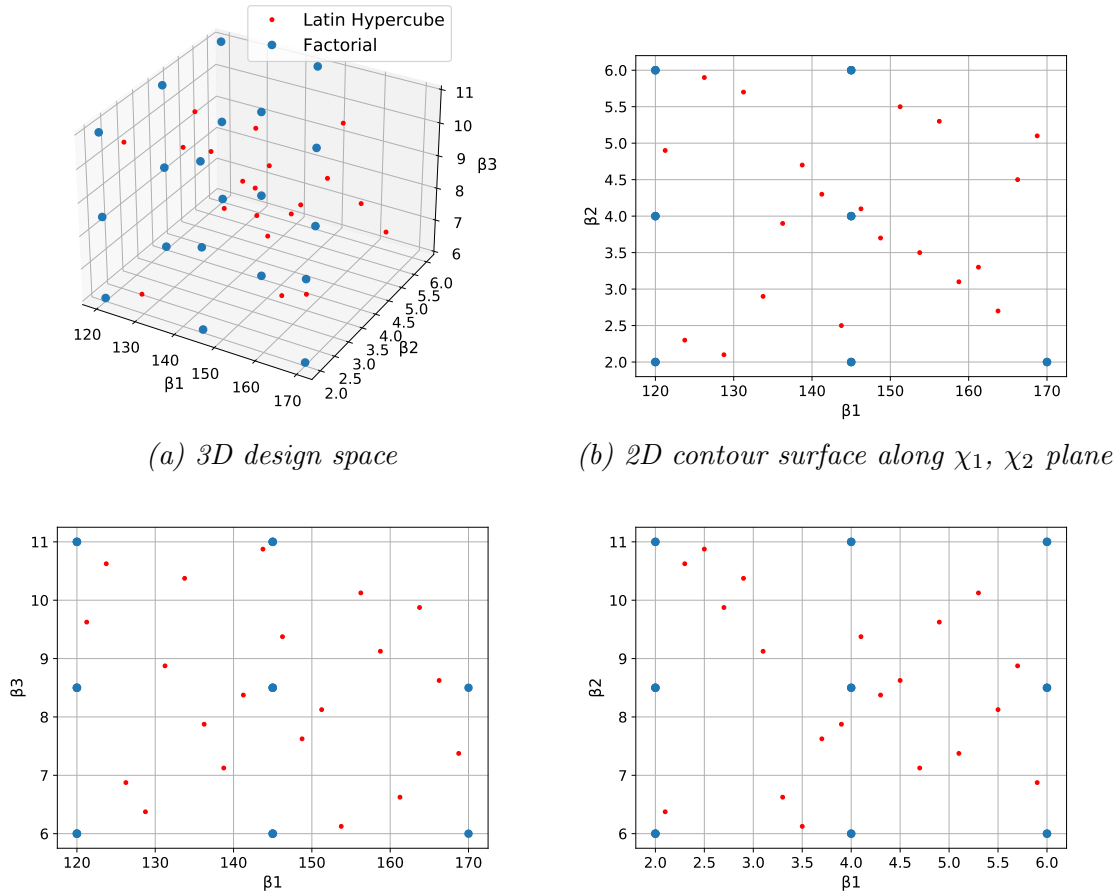


Figure 2.8: Example of Factorial design for 2 design variables with $n_{doe} = 10$ sample points

2.1.1 Comparison between DoE construction schemes

The selection of a suitable design is an essential step to the training of a surrogate model. For that reason the available DoE construction schemes are evaluated w.r.t. their effect on the training process of the metamodel. The comparison is limited to Factorial and LH designs, since they tend to be the most reliable in overall coverage of the design space and especially in the selection of a representative sample from the total population set. Random designs are eliminated from the assessment process, since they are considered unfit for large population sets due to equiprobable selection of each individual. Optimality space-filling criteria are not utilized in random designs and the outcome is a design that either contains a number of similar sample points or omits significant sample points that are of great importance to the training of the surrogate model [12].

The first difference between the two remaining DoE construction schemes is detected in the selection process of sample points. In both full and fractional factorial designs, the sample points are distributed as evenly as possible in the n_β-dimensional design space, utilizing its full capacity. In LHDs, the design space is stratified and the sample points are selected from within the created intervals via the use of some space-filling construction criterion. For up to $n_\beta = 3$ design variables the resulting designs can be replicated in 3D space as depicted in figure 2.9; in this example the design space is created by the bounds of each design variable in eq. A.2.



(a) 3D design space

(b) 2D contour surface along χ_1, χ_2 plane

(c) 2D contour surface along χ_1, χ_3 plane

(d) 2D contour surface along χ_2, χ_3 plane

Figure 2.9: Factorial and LH designs in 3D design space

The implementation of factorial sampling in 3D space for $n_{doe} = 20$ runs results in the creation of a fractional factorial design, which disregards a large section of the design space. For the same number of runs, on the other hand, LHS scheme spreads sample points optimally across the design space and yields, for this reason, better designs. In LHDs furthermore, the MDB is more diverse and complete, since it consists of n_{doe} distinct values, unlike in factorial designs where the influence of each design variable is tested on $\kappa_l < n_{doe}$ levels each. The responses of n_{doe} sample points can be written as a $n_{doe} \times n$ matrix $\mathbf{F} = [\vec{f}_1, \vec{f}_2, \ldots, \vec{f}_{n_{doe}}]^T$, where each component $\vec{f}_i = [f_{i,1}, f_{i,2}, \ldots, f_{i,n}]$ represents a response $\vec{f} \in \mathbb{R}^n$. The responses of $n_{doe} = 20$ sample points in eq. A.2 (see appendix chapter A.1) are depicted in figure 2.10.

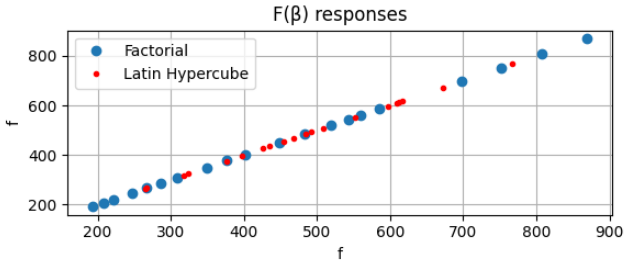


Figure 2.10: Comparison between $\mathbf{F}(\vec{\chi})$ responses to samples created via Factorial and LHS DoE scheme

The two DoE schemes yield similar objective function values, as seen in figure 2.10). However, a similarity in objective function responses cannot lead to any definite conclusions on the quality of the respective models. One of many metrics for metamodel quality is the Root Mean Square Error (RMSE) (see appendix chapter A.3). This metric depends on the order of magnitude of the observed values and the size of the sample, so it is merely used in the comparison of various metamodels when approximating the same PSM and trained on the same dataset. Consequently, a high RMSE is a characteristic of a model that has been selectively trained for only a narrow set of sample points, therefore lacking in robustness. This concept is tested in a KPLS model with fitting shown in figure 2.11.

(a) LHD, NRMSE = 0.003818 *(b) Factorial design, NRMSE = 0.010939*

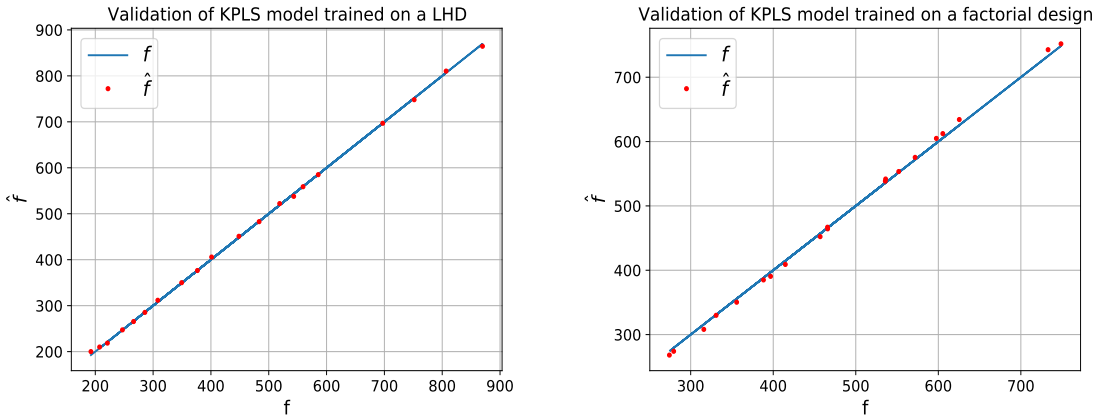


Figure 2.11: NRMSE of metamodels trained on a LHD and a factorial design

The RMSE is calculated using the following equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_{val}} \left(\hat{f}_i - f_i\right)^2}{n_{val}}} \tag{2.5}$$

where n_{val} is the number of validation points and $\vec{\chi_i} = \left[\chi_{i,1}, \ldots, \chi_{i,n_\beta}\right] \in \mathbb{R}^{n_\beta}$ the vector of the i_{th} training point. Validation points are selected from the design space via the implementation of any DoE technique and are used in the evaluation of the model[2]. Every set of sample points different than the one used to train the metamodel is considered a set of validations point. The values obtained via the use of the trained surrogate model are denoted by $\hat{f}$ and referred to as estimated values. In equation 2.5 the existence of a single objective is assumed and the both f and $\hat{f}$ are scalar quantities. In MOO problems, the RMSE is computed w.r.t. to each objective iteratively. In order to remove the dependency on the order of magnitude Normalised Root Mean Square Error (NRMSE) is introduced, which can be calculated from the following formula:

$$NRMSE = \sqrt{\frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \left(\frac{\hat{f}_i - f_i}{f_i}\right)^2} \tag{2.6}$$

NRMSE is dimensionless and assumes values in the $\mathbb{R}^+$, with values closer to zero indicating a well-trained metamodel. NRMSE is not restricted in a specific dataset but can rather be generalised to compare models of various orders of magnitude.

In addition to inferior metamodel quality, factorial designs are imposed with severe limitations when sampling high-dimensional design spaces. Even in its simplest form a FFD must consist of $n_{doe} = 2^{n_\beta}$ possible combinations. In 10 dimensions, the number of runs required to fully replicate the design is:

$$n_{doe} = (2)^{10} = 1024$$

If moreover the case in study is that of a 3D airfoil, then each set of design variable values $\vec{\beta}$ would correspond to a different airfoil shape. That results in 1024 different shapes and therefore to a beyond sustainability computational cost. The number of factorial runs needed for the creation of a factorial design in 10-dimensional space is $n_{doe} > 600$, which results experimentally from the implementation Python software. On the other hand, LHS scheme is suitable for creating high-dimensional designs and offers a better coverage of the design space combined with minimal impact on the computational cost, which leads to its selection as the main sampling scheme used in this thesis.

2.2 Communication between EASY and SMT in MAEAs with off-line training

In the evaluation phase of MAEAs with off-line training the PSM is replaced by a surrogate model, which is trained on n_t training patterns that are collected via the use of various DoE techniques. The creation of DoE, the training of the metamodel and the prediction of the objective function value are performed via the use of SMT. However, in order for SMT to facilitate the evolution performed by EASY (see appendix C), a set of modifications must be applied in order for the two programs to be compatible. Responsible for the establishment of a line of communication between the two software is a Python script, which is manually created and can be decomposed in the following sections:

1. **Sampling (Code 1)**
   The design space is defined, i.e. upper and lower bound of each design variable, along with the magnitude of the design, denoted by $n_{doe}^{'}$, and the DoE technique utilized to construct it. The $n_{doe}^{'}$ collected training patterns $\mathbf{X}$ are subsequently written in an ASCII text file *sample_points.dat*, prior to the termination of Code 1.

2. **Evaluation of sample points on the PSM (Code 2)**
   Code 2 contains the exact PSM. Both the input to the PSM, i.e. the observations $\mathbf{X} \in \mathbb{R}^{n_{doe}^{'} \times n_\beta}$ contained in *sample_points.dat*, and the yielded responses $\mathbf{F}(\vec{\chi}) \in \mathbb{R}^{n_{doe}^{'} \times n}$ are written in a plain ASCII text file *model_values.dat*, along with the constraints $\mathbf{C}(\vec{\chi}) = [\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_{n_t}]^T$, in case the optimization problem is constrained. Each component $\vec{c}_i = [c_{i,1}, c_{i,2}, \ldots, c_{i,n_c}]$ corresponds to the constraint vector $\vec{c}(\vec{\chi})$ of the i_{th} sample point $\vec{\chi} \in \mathbb{R}^{n_\beta}$.

3. **Training of metamodel (Code 3)**
   Codes 1 through 3 are incorporated in the *preprocessor.exe* executable. Code 3 in particular is responsible for training the selected surrogate model. In order to accomplish that, first $n_{doe}^{'}$ $(\vec{\chi}, \vec{f}(\vec{\chi}))$ observed pairs must be imported from *model_values.dat*. At the end of each off-line optimization cycle the MDB is updated with λ_e elites that are contained in the P_e set. Consequently, Code 3 is also responsible for incorporating λ_e $(\vec{\beta}, \vec{f}(\vec{\beta}))$ pairs in the training of the metamodel, after importing them from a plain ASCII text file *out.log* that is created by Code 6. Once the MDB is complete, a new surrogate model is trained at the start of each optimization cycle using n_t training pairs $(\vec{\chi}, \vec{f}(\vec{\chi}))$ (see eq. 2.1).

   In the case of a constrained optimization, the matrix of constraints $\mathbf{C}(\vec{\chi})$ is imported into Code 3 along with the objective function values matrix $\mathbf{F}(\vec{\chi})$, and a distinct metamodel is trained on each constraint or a single metamodel is trained for the entirety of the imposed constraints. Metamodels trained on constraints require n_t $(\vec{\chi}, \vec{c}(\vec{\chi}))$ training pairs to be built.

   Once the training is complete, the parameters of each trained metamodel are witten in a binary text file using the Python module *.pickle()*; in this way, it can be utilized in the evaluation of prominent solutions in each generation of the evolution. For some metamodels, however, *.pickle()* is not applicable, e.g.

RBF. In that case, a folder containing the cached data that are produced via the training process is used in order to store and reuse the saved surrogate model.

4. **Evaluation using the trained metamodel (Code 4)**
   The current script uses as input the file *task.dat*, which EASY creates, and contains a single offspring $\vec{\beta} \in P_{\lambda}^{g}$. A metamodel prediction $\hat{f}(\vec{\beta})$ is subsequently computed for every offspring $\vec{\beta} \in P_{\lambda}^{g} \subset \mathbb{R}^{n_{\beta}}$ by utilizing the stored metamodel via the use of *.pickle()* module. Code 4 is identical in form to *evaluation.exe*, but the PSM is replaced by a surrogate hence it is called *prediction.exe*.

5. **Selection of objectives and constraints (Code 5)**
   In order to establish the communication between EASY and the user, *post-processor.exe* is manually created. This script is responsible for writing the objectives and the imposed constraints of the optimization in a *task.cns* and a *task.res* file, respectively. Text files *task.res* and *task.cns* contain the predictions $\hat{\vec{f}}(\vec{\beta})$ and $\hat{\vec{c}}(\vec{\beta})$, respectively, of a single individual $\vec{\beta} \in P_{\lambda}^{g}$ and are read by EASY.

6. **Evaluation of elites using the trained metamodel (Code 6)**
   Code 6 performs the evaluation of the elite population P_e using the exact PSM. The only difference with Code 2 is that the inputs $(\vec{\beta}, \hat{\vec{f}}(\vec{\beta}))$, $(\vec{\beta}, \hat{\vec{c}}(\vec{\beta}))$, $\forall \vec{\beta} \in P_e$ are imported from *out_L1.log*, which is created by EASY at the end of each optimization cycle, and the corresponding exact PSM evaluations $(\vec{\beta}, \vec{f}(\vec{\beta}))$, $(\vec{\beta}, \vec{c}(\vec{\beta}))$ are written in *out.log*. Both these files follow ASCII text format.

Chapter 3

MAEAs with on-line training

On-line trained surrogate models are built in each generation of the evolution and hence this MAEAs method is described as dynamic. The most common approach is one that involves the implementation of a Low-Cost Pre-Evaluation (LCPE) process. LCPE is responsible for the selection of promising individuals via the implementation of local or global metamodels. The former are however more widely used, since they tend to approximate complex objectives function more effectively. MAEAs with metamodels trained on-line via LCPE phase can subsequently decomposed in the following discrete steps:

ONL-1. **Implementation of EAs**
The initialization of LCPE phase requires the implementation of conventional EAs for a number of generations. Each untried individual $\vec{\beta} \in P_\lambda^g$ is evaluated on the PSM and subsequently archived in the DB. Once a user-defined minimum number of individuals has been stored in the DB, LCPE[28] phase initiates.

ONL-2. **Low-cost Pre-evaluation**
LCPE phase initiates by training on the fly a local surrogate model for each untried individual $\vec{\beta} \in P_\lambda^g$. The training of each metamodel in SOO problems requires the selection of an appropriate set of training patterns from the vicinity of each individual $\vec{\beta}$. In MOO problems more sophisticated algorithms are required. Such a method is developed by PCOpt/NTUA and is called Training Pattern Selection (TPS)[29]. Using the trained local metamodels, the objective function value of each offspring s subsequently predicted and denoted by $\hat{\vec{f}}(\vec{\beta}), \forall \vec{\beta} \in P_\lambda^g$.

ONL-3. **Computation of fitness function**
This step is identical to step EAs-3. Each candidate solution $\vec{\beta} \in P_\lambda^g$, is assigned a scalar value. Depending on the process implemented to evaluate each candidate solution, i.e. PSM or LCPE using local metamodels, the fitness function is either exactly calculated ($\Phi(\vec{\beta})$) or predicted ($\widehat{\Phi}(\vec{\beta})$), respectively.

$$\Phi(\vec{\beta}_i) = \Phi(\vec{f}(\vec{\beta}_i), \{\vec{f}(\vec{z}) \mid \vec{z} \in P_\lambda^g \setminus \{\vec{\beta}_i\}\}) \in \mathbb{R} \quad \text{, for } i = 1, \lambda \tag{3.1}$$

or

$$\widehat{\Phi}(\vec{\beta}_i) = \widehat{\Phi}(\hat{\vec{f}}(\vec{\beta}_i), \{\hat{\vec{f}}(\vec{z}) \mid \vec{z} \in P_\lambda^g \setminus \{\vec{\beta}_i\}\}) \in \mathbb{R} \quad \text{, for } i = 1, \lambda \tag{3.2}$$

ONL-4. **Identification of elites**
The values of the fitness function assigned to each candidate solution are used to update the temporary set of elites P_e. In SOO problems $\Phi(\vec{\beta}) \equiv f(\vec{\beta})$, $\widehat{\Phi}(\vec{\beta}) \equiv \hat{f}(\vec{\beta})$ and there is only one optimal solution $\lambda_e = 1$. In MOO problems the best $\lambda_e < \lambda$ offspring are selected to populate the P_e set. In the latter category due to the large number of data the fitness function values are assigned via the implementation of the simplest sorting algorithm, e.g. NSGA[5] or SPEA[6], or via a simple ranking of non-dominated Pareto fronts. The final P_e set is formed as such:

$$P_e \triangleq \{\vec{\beta}_i \colon \widehat{\Phi}(\vec{\beta}_i) < \widehat{\Phi}(\vec{z}), \vec{z} \in P_\lambda^g \setminus P_e\} \quad , \text{for } i = 1, \lambda_e \tag{3.3}$$

The process of populating the P_e set continues until:

$$\lambda_{e,min} < \lambda_e < \lambda_{e,max} \tag{3.4}$$

where $\lambda_{e,min}, \lambda_{e,max}$ are user-defined lower and upper bounds of the number of elites λ_e, respectively.

ONL-5. **CFD evaluation**
Subsequently, λ_e elite candidate solutions contained in the P_e set are re-evaluated on the PSM and are stored in the DB. Depending on the deviation between metamodel and psm evaluated outcome, denoted by ε_{P_e}, either the evolution continues or new elites are selected (step ONL-4) and re-evaluated (step ONL-5). This criterion can be expressed mathematically as such:

$$\varepsilon_{P_e} = \left| \frac{\vec{f}(\vec{\beta}) - \hat{\vec{f}}(\vec{\beta})}{\vec{f}(\vec{\beta})} \right| < \varepsilon_\lambda \quad , \forall \vec{\beta} \in P_e \tag{3.5}$$

where ε_λ a user-defined value upon which the criterion is satisfied.

ONL-6. **Elitism**
The temporary set P_e is used to update the population of the current generation P_α^g. The process of elitism subsequently commences and leads to the formation of P_α^{g+1} set. This step is identical to step EAs-5.

ONL-7. **Crossover and mutation**
Crossover and mutation operators are applied to form the set P_λ^{g+1}, similarly to step EAs-6.

ONL-8. **Termination**
Once the process of implementing evolution operators is complete, the convergence of the on-line training process is tested. If a user-defined number of generations has been formed the process terminates, alternatively, the next generation initiates by setting $g = g + 1$. If a user-defined number of idle generations n_{idle} has been performed using metamodels in LCPE phase, then plain EAs are utilized and the counter of idle LCPE generations c_{idle} is reset to zero, as shown in figure 3.1.

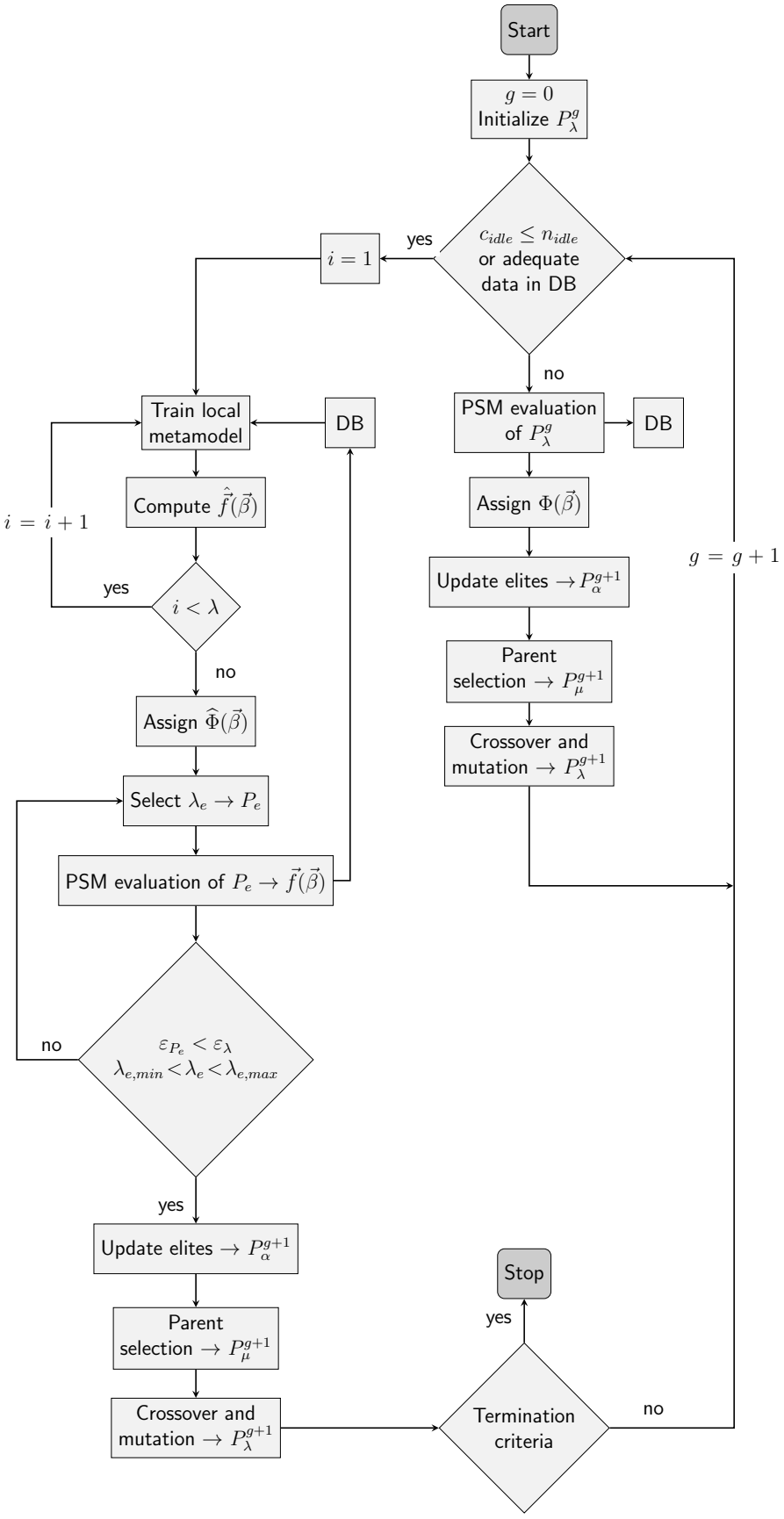


Figure 3.1: Flowchart of MAEAs using on-line trained metamodels

3.1 Communication between EASY and SMT in MAEAs with on-line training

The optimization based on MAEAs via the use of EASY[27] software is primarily focused on training metamodels on-line and therefore a number of metamodels are already archived in the database of EASY. In order to provide EASY with external metamodels trained on SMT software, the following Python scripts must be deployed:

1. **Evaluation of offspring using the exact model (Code 1)**
   This script contains the exact PSM and is responsible for computing the exact objective function vector $\vec{f}(\vec{\beta}), \forall \vec{\beta} \in P_{\lambda}^{g}$. Each one of those candidate solutions is imported from the file *task.dat*. The script is subsequently converted to an executable process, called *evaluation.exe*, and executed via *task.bat* batch file, which has the following structure:

```
@echo off
erase results.dat
evaluation.exe > nul
postprocessor.exe > nul
```

Listing 3.1: Structure of task.bat file that initiates the exact evaluation of offspring

2. **Training of metamodel (Code 2)**
   LCPE phase initiates by training a local metamodel for each individual $\vec{\beta} \in P_{\lambda}^{g}$. Both the n_t training patterns $\mathbf{X}$ and their corresponding exact model values $\mathbf{F}(\vec{\chi})$ are imported from a plain ASCII text file, which is called *meta.db* and is created by EASY. Code 2 is converted to *train.exe* and executed via a user-created batch file *meta_train.bat* that has the following structure:

```
@echo off
train.exe > nul
```

Listing 3.2: Structure of meta_train.bat file that initiates training of the metamodel

3. **Evaluation of offspring using the metamodel (Code 3)**
   Code 3 utilises each local metamodel, which is built in the vicinity of the i_{th} individual $\vec{\beta}_i \in P_{\lambda}^{g}$, in order to produce the evaluation $\hat{\vec{f}}(\vec{\beta}_i)$. Each individual is contained in a plain ASCII text file *meta.dat*, which is structured similarly to *task.dat*. Consequently, after converting the script to an executable *prediction.exe* Code 3 is executed iteratively for λ offspring via *meta_use.bat* batch file, which has the following structure:

```
@echo off
erase results.dat
prediction.exe > nul
postprocessor.exe > nul
```

Listing 3.3: Structure of meta_use.bat file that initiates the evaluation of some candidate solution $\vec{\beta} \in P_{\lambda}^{g}$ based on its personalised local metamodel

4. **Objectives and constraints (Code 4)**
   Code 4, which is called *postprocessor.exe*, is executed alternately via *task.bat* and *meta_use.bat* batch files in order to provide EASY with the exact $\vec{f}(\vec{\beta})$ or predicted $\hat{\vec{f}}(\vec{\beta})$ objective function value of each individual $\vec{\beta} \in P_{\lambda}^{g}$, respectively. EASY expects to read this value, or values if there are more than one objective, in *task.res* file. Any constraint $\vec{c}(\vec{\beta})$ is subsequently written in a *task.cns* file.

In EASY, the MAEA-based on-line construction process consists of the same fundamental steps, i.e. evaluation based on the PSM, training of the surrogate model and prediction based on the trained model, but no additional user-constructed scripts are needed to utilize the built-in metamodels of EASY, in contrast to SMT.

Chapter 4

Surrogate Models

Metamodels approximate the initial evaluation model by utilising a data-driven approach, which is based on statistical analysis of the observed data. Consequently, the selection of a suitable surrogate model is essential in the optimal utilization of MAEA-based optimization. In attempt to achieve homogeneity throughout this thesis, it is reminded that the observations' matrix formed of n_t training patterns is denoted by $\mathbf{X} = [\vec{\chi}_1, \vec{\chi}_2, \ldots, \vec{\chi}_{n_t}]^T$, where each component $\vec{\chi}_i = [\chi_{i,1}, \chi_{i,2}, \ldots, \chi_{i,n_\beta}]$ represents an observation $\vec{\chi} \in \mathbb{R}^{n_\beta}$. Their corresponding objective function values are included in matrix $\mathbf{F}(\vec{\chi}) = [\vec{f}_1, \vec{f}_2, \ldots, \vec{f}_{n_t}]^T$, where each component $\vec{f}_i = [f_{i,1}, f_{i,2}, \ldots, f_{i,n}]$ represents a response $\vec{f} \in \mathbb{R}^{n_\beta}$. In order to simplify the mathematical equations describing the surrogate models, $\mathbf{F}(\vec{\chi})$ is reduced to an 1-dimensional matrix by assuming, without loss in generality, that the optimization process has a single objective. From there, the respective equations describing a MOO can be easily formulated by combining the SOO equations iteratively for n objectives. In SOO, therefore, $\mathbf{F}(\vec{\chi}) = \mathbf{F} = [f_1, f_2, \ldots, f_{n_t}]^T \in \mathbb{R}^{n_t}$. Surrogate models are used to predict the objective function value $f(\vec{\beta})$ at any untried location of the design space, i.e. at each candidate solution $\vec{\beta} \in \mathbb{R}^{n_\beta}$. The theoretical background of every metamodel utilized via SMT in this thesis is subsequently presented.

4.1 Kriging

Kriging[30] is a surrogate model used for predicting the objective function value at any candidate solution $\vec{\beta} \in \mathbb{R}^{n_\beta}$ in the design space. In order to make the prediction Kriging uses an interpolation method that combines a deterministic term with the realization of stochastic process. The former is replaced by a regression model and the latter is the realization of the stationary process Gaussian $z(\vec{\beta}) \backsim N(0, C)$ with a zero mean and a covariance kernel $C(\vec{\beta})$ of the observations:

$$C(\vec{\chi}_i, \vec{\chi}_j) = \sigma^2 R(\vec{\chi}_i, \vec{\chi}_j) \tag{4.1}$$

where σ^2 the variance of the process and $R(\vec{\chi}_i, \vec{\chi}_j)$ the correlation between any two observations $\vec{\chi}_i, \vec{\chi}_j \in \mathbb{R}^{n_\beta}, \forall i, j \in [1, n_t]$. In order to improve the fitting of the Kriging model the distribution of training patterns in each problem dimension is normalized:

$$\vec{\chi}_{norm} = \frac{\vec{\chi} - \mu_{\vec{\chi}^{(j)}}}{\sigma_{\vec{\chi}^{(j)}}} \tag{4.2}$$

where $\vec{\chi}^{(j)} = [\chi_{1,j}, \chi_{2,j}, \ldots, \chi_{n_t,j}]^T \in \mathbb{R}^{n_t}$ is the column vector of the $n_t \times n_\beta$ matrix $\mathbf{X}$ and $\mu_{\vec{\chi}^{(j)}}$, $\sigma_{\vec{\chi}^{(j)}}$ the mean value and the standard deviation of the j_{th} observation, respectively. The correlation between any normalized training point $\vec{\chi}_{norm} \in S \subset \mathbb{R}^{n_\beta}$, where S is the design set, can be computed using one of the following correlation kernels [1] [2, 31]:

- Exponential Ornstein-Uhlenbeck process

$$R\left(\vec{\chi_i}, \vec{\chi_j}\right) = \prod_{l=1}^{n_\beta} exp\left(-\theta_l \left|\chi_{i,l} - \chi_{j,l}\right|\right) \tag{4.3}$$

- Gaussian

$$R\left(\vec{\chi_i}, \vec{\chi_j}\right) = \prod_{l=1}^{n_\beta} exp\left(-\theta_l \left(\chi_{i,l} - \chi_{j,l}\right)^2\right) \tag{4.4}$$

- Matérn 5/2

$$R\left(\vec{\chi_i}, \vec{\chi_j}\right) = \prod_{l=1}^{n_\beta} \left(1 + \sqrt{5}\left|\chi_{i,l} - \chi_{j,l}\right| + \frac{5}{3}\theta_l^2 \left(\chi_{i,l} - \chi_{j,l}\right)^2\right) exp\left(-\sqrt{5}\theta_l \left|\chi_{i,l} - \chi_{j,l}\right|\right) \tag{4.5}$$

- Matérn 3/2

$$R\left(\vec{\chi_i}, \vec{\chi_j}\right) = \prod_{l=1}^{n_\beta} \left(1 + \sqrt{3}\theta_l \left|\chi_{i,l} - \chi_{j,l}\right|\right) exp\left(-\sqrt{3}\theta_l \left|\chi_{i,l} - \chi_{j,l}\right|\right) \tag{4.6}$$

where θ_l are parameters that denote the degree of correlation between training points w.r.t. each design dimension $l \in [1, n_\beta]$. Kriging assumes that the estimated value of each correlation parameter θ is constant for each independent design variable and therefore for each design dimension, leading to the creation of an isotropic model[32]. The correlation patterns of the observed data and their corresponding covariance can be stated in the form of an orthogonal matrix $\mathbf{R}$ and $\mathbf{C}$, respectively:

$$\mathbf{R} = \begin{bmatrix} R(\vec{\chi}_1, \vec{\chi}_1) & \ldots & R(\vec{\chi}_1, \vec{\chi}_{n_t}) \\ \vdots & \ddots & \vdots \\ R(\vec{\chi}_{n_t}, \vec{\chi}_1) & \ldots & R(\vec{\chi}_{n_t}, \vec{\chi}_{n_t}) \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} C(\vec{\chi}_1, \vec{\chi}_1) & \ldots & C(\vec{\chi}_1, \vec{\chi}_{n_t}) \\ \vdots & \ddots & \vdots \\ C(\vec{\chi}_{n_t}, \vec{\chi}_1) & \ldots & C(\vec{\chi}_{n_t}, \vec{\chi}_{n_t}) \end{bmatrix} \tag{4.7}$$

Under the assumption of a SOO problem, the Kriging model computes the objective function value at any normalized point $\vec{\beta} \in \mathbb{R}^{n_\beta}$ outside the sampled design as such:

$$f(\vec{\beta}) = \mu_K + z(\vec{\beta}) \tag{4.8}$$

where the deterministic term μ_K is expressed as a constant, linear or quadratic regression model:

$$\mu_K = \sum_{j=1}^{k} \mathrm{w}_j p_j(\vec{\beta}) \tag{4.9}$$

[1] In all Kriging models from this point on, the notation of normalization will not be used for sampled points but will be implied for simplicity, so $\vec{\chi} \equiv \vec{\chi}_{norm}$ and $\vec{\chi} \equiv \vec{\chi}_{norm}$

where w_j is the j_{th} regression coefficient and $p_j : \mathbb{R}^{n_\beta} \mapsto \mathbb{R}$ are k chosen functions. The parameter k assumes various values to denote a constant, a linear or a quadratic regression model [33]. In a constant regression model, $k = 1$ and $p_1(\vec{\beta}) = 1$. In a linear regression model, $k = n_\beta + 1$ and the corresponding functions assume the following values:

$$p_1(\vec{\beta}) = 1,\ p_2(\vec{\beta}) = \beta_1, \ldots,\ p_k(\vec{\beta}) = \beta_{n_\beta} \tag{4.10}$$

In a quadratic regression model, $k = \frac{1}{2}(n_\beta + 1)(n_\beta + 2)$ and the functions p_j assume the following values:

$$\begin{aligned}
&p_1(\vec{\beta}) = 1,\ \ p_2(\vec{\beta}) = \beta_1, \ldots, \\
&p_{n_\beta+1}(\vec{\beta}) = \beta_{n_\beta},\ \ p_{n_\beta+2}(\vec{\beta}) = \beta_1^2, \ldots, \\
&p_{2n_\beta+1}(\vec{\beta}) = \beta_1\beta_{n_\beta},\ \ p_{2n_\beta+2}(\vec{\beta}) = \beta_2^2, \ldots, \\
&p_{3n_\beta}(\vec{\beta}) = \beta_2\beta_{n_\beta}, \ldots \\
&p_k(\vec{\beta}) = \beta_{n_\beta}^2
\end{aligned} \tag{4.11}$$

where $\beta_j \in \mathbb{R}$ is the component of any untried point $\vec{\beta}$ w.r.t. the j_{th} design dimension for $j \in [1, n_\beta]$.

- **Prediction with noise-free observations**

Kriging, when provided with the observed data that are collected via the implementation of DoE, can predict the value of any individual at any untried location of the design space accompanied by the measure of confidence of the prediction at that location. Under the assumption of a SOO problem, consider the linear predictor $\hat{f}(\vec{\beta})$ of the objective function at any untried point $\vec{\beta}$, given the prior observations $\mathbf{F} = [\vec{f}_1, \vec{f}_2, \ldots, \vec{f}_{n_t}]^T$:

$$\hat{f}(\vec{\beta}) = \vec{c}^{\,T}(\beta)\mathbf{F} \tag{4.12}$$

where $\vec{c}(\beta) \in \mathbb{R}^{n_t}$ is the $n_t \times 1$ vector of coefficients. Then the deviation between the predictor and the true objective function value:

$$\begin{aligned}
\hat{f}(\vec{\beta}) - f(\vec{\beta}) &= \vec{c}^{\,T}(\vec{\beta})\mathbf{F} - (\mu_K + z(\vec{\beta})) \xrightarrow[(4.9)]{(4.8)} \\
&= \vec{c}^{\,T}(\vec{\beta})\,(\mathbf{P}\vec{\mathrm{w}} + \mathbf{Z}) - (\vec{p}^{\,T}(\vec{\beta})\vec{\mathrm{w}} + z(\vec{\beta})) \\
&= \vec{c}^{\,T}(\vec{\beta})\mathbf{Z} - z(\vec{\beta}) + (\mathbf{P}^T\vec{c}(\vec{\beta}) - \vec{p}(\vec{\beta}))^T\vec{\mathrm{w}}
\end{aligned} \tag{4.13}$$

where $\mathbf{Z} = [z_1, z_2, \ldots, z_{n_t}]^T$ is the $n_t \times 1$ vector of errors at the observed points, $z(\vec{\beta})$ is the error at the untried location, $\vec{p}(\vec{\beta}) = [p_1(\vec{\beta}), p_2(\vec{\beta}), \ldots, p_k(\vec{\beta})]^T$ is the $k \times 1$ vector of the chosen functions at any untried input $\vec{\beta}$ and $\mathbf{P}$ the corresponding $n_t \times k$ matrix for the complete design of observed data, which for $i = 1, n_t$ training patterns $\vec{\chi}_i = [\chi_{i,1}, \chi_{i,2}, \ldots, \chi_{i,n_\beta}] \in S \subset \mathbb{R}^{n_\beta}$ is expressed as:

$$\mathbf{P} = \begin{bmatrix} p_1(\vec{\chi}_1) & p_2(\vec{\chi}_1) & \dots & p_k(\vec{\chi}_1) \\ p_1(\vec{\chi}_2) & p_2(\vec{\chi}_2) & \dots & p_k(\vec{\chi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(\vec{\chi}_{n_t}) & p_2(\vec{\chi}_{n_t}) & \dots & p_k(\vec{\chi}_{n_t}) \end{bmatrix} \tag{4.14}$$

The best linear unbiased predictor (BLUP) is obtained by selecting the vector $\vec{c}(\vec{\beta})$ that minimizes the mean squared error (MSE). In order to keep the predictor unbiased, we demand that the expected value of the predictor and objective function coincides at the design sites $\mathbf{X}$ [35]:

$$\begin{aligned} &E[\hat{f}(\vec{\beta}) - f(\vec{\beta})] = 0 \xrightarrow{(4.13)} E[\vec{c}^{\,T}(\vec{\beta})\mathbf{Z} - z(\vec{\beta}) + (\mathbf{P}^T\vec{c}(\vec{\beta}) - \vec{p}(\vec{\beta}))^T\vec{\mathrm{w}}] = 0 \rightarrow \\ &\vec{c}^{\,T}(\vec{\beta})E[\mathbf{Z}] - E[z(\vec{\beta})] + E[(\mathbf{P}^T\vec{c}(\vec{\beta}) - \vec{p}(\vec{\beta}))^T\vec{\mathrm{w}}] = 0 \xrightarrow{E[z(\vec{\beta})]=\mu_z=0} \\ &E[(\mathbf{P}^T\vec{c}(\vec{\beta}) - \vec{p}(\vec{\beta}))^T]\vec{\mathrm{w}} = 0 \rightarrow \mathbf{P}^T\vec{c}(\vec{\beta}) - \vec{p}(\vec{\beta}) = 0 \end{aligned} \tag{4.15}$$

Consequently, the MSE of the predictor is calculated as such:

$$\begin{aligned} E[\hat{f}(\vec{\beta})] &= E[\hat{f}(\vec{\beta}) - f(\vec{\beta})]^2 = E[(\vec{c}^{\,T}(\vec{\beta})\mathbf{Z} - z(\vec{\beta}))^2] \\ &= E[\vec{c}^{\,T}(\vec{\beta})\mathbf{Z}\mathbf{Z}^T\vec{c}(\vec{\beta})] - 2\vec{c}^{\,T}(\vec{\beta})\mathbf{Z}z(\vec{\beta}) + z^2(\vec{\beta})] \\ &= \sigma^2\left(\vec{c}^{\,T}(\vec{\beta})\mathbf{R}\vec{c}(\vec{\beta}) - 2\vec{c}^{\,T}(\vec{\beta})\vec{r}_{X\beta} + 1\right) \end{aligned} \tag{4.16}$$

where $\vec{r}_{X\beta} = [R(\vec{\chi}_1, \vec{\beta}), R(\vec{\chi}_2, \vec{\beta}), \dots, R(\vec{\chi}_{n_t}, \vec{\beta})]^T$ is the $n_t \times 1$ matrix denoting the correlation between the n_t observations and any untried candidate solution $\vec{\beta} \in \mathbb{R}^{n_\beta}$. $E[\hat{f}(\vec{\beta})]$ is minimized w.r.t. $c(\vec{\beta})$ and subject to the equality constraint $\mathbf{P}^T\vec{c}(\vec{\beta}) - \vec{p}(\vec{\beta}) = 0$ stated in eq. 4.15, when the Kriging BLUP at some untried point $\vec{\beta} \in \mathbb{R}^{n_\beta}$ is given by equation 4.17:

$$\hat{\mathrm{f}}(\vec{\beta}) = \vec{p}^{\,T}(\vec{\beta})\hat{\vec{\mathrm{w}}} + \vec{r}_{X\beta}\mathbf{R}^{-1}\left(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}}\right) \tag{4.17}$$

The regression coefficients of the BLUP are estimated at the observed design sites using generalised least-squares method. The $k \times 1$ vector $\hat{\vec{\mathrm{w}}} = [\mathrm{w}_1, \mathrm{w}_2, \dots, \mathrm{w}_k]^T$ of the estimates of $\vec{\mathrm{w}}$ are given by:

$$\hat{\vec{\mathrm{w}}} = \left(\mathbf{P}^T\mathbf{R}^{-1}\mathbf{P}\right)^{-1}\mathbf{P}^T\mathbf{R}^{-1}\mathbf{F} \tag{4.18}$$

The MSE of the Kriging predictor can be computed using equation 4.19:

$$MSE(\vec{\beta}) = \hat{\sigma}^2(1 - \vec{r}_{X\beta}^{\,T}\mathbf{R}^{-1}\vec{r}_{X\beta}) \tag{4.19}$$

which can be solved by adopting generalised least-squares estimates for the variance:

$$\widehat{\sigma}^2 = \frac{1}{n_t}(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}})^T\mathbf{R}^{-1}(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}}) \tag{4.20}$$

The computation of the Kriging predictor requires the inversion of the symmetric matrix of correlations $\mathbf{R}$, so the computational cost depends on the size of the training sample n_t. The calculation of $\mathbf{R} = \mathbf{R}(\vec{\theta})$ requires the computation of n_β correlation parameters θ, assuming an isotropic design, which are estimated using either maximum likelihood or cross validation method. The former method is more commonly used and dictates the selection of those parameters θ that maximize the likelihood function $l_F(\vec{\theta}|\mathbf{F})$ given the responses $\mathbf{F}$, which is a function of $\vec{\theta} = [\theta_1, \theta_2, \ldots, \theta_{n_\beta}]$ mathematically expressed as [34]:

$$l_F(\vec{\theta}|\mathbf{F}) = \frac{1}{(2\pi)^{n_t/2}(\sigma^2)^{n_t/2} det\mathbf{R}^{1/2}} exp\left[\frac{-(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}})^T\mathbf{R}^{-1}(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}})}{\sigma^2}\right] \tag{4.21}$$

Intuitively, this process tries to infer the design space population that is most likely to have generated the responses $\mathbf{F}$. The complexity of the previous equation decreases by computing $ln(l_F(\vec{\theta}|\mathbf{F}))$, since $ln(\cdot)$ is monotonous:

$$\begin{aligned} ln(l_F(\vec{\theta}|\mathbf{F})) = &- \frac{n_t}{2} ln(2\pi) - \frac{n_t}{2} ln(\sigma^2) - \frac{1}{2} ln(det\mathbf{R}) \\ &- \frac{(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}})^T\mathbf{R}^{-1}(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}})}{\sigma^2} \end{aligned} \tag{4.22}$$

After inserting equations 4.18 and 4.20 in eq. 4.22, the latter can be written in the concentrated ln-likelihood form where any constant terms are ignored:

$$\begin{aligned} ln(l_F(\vec{\theta}|\mathbf{F})) = &- \frac{n_t}{2} ln\left[\frac{1}{n_t}\left(\mathbf{F} - \mathbf{P}(\mathbf{P}^T\mathbf{R}^{-1}\mathbf{P})^{-1}\mathbf{P}^T\mathbf{R}^{-1}\mathbf{F}\right)^T\right. \\ &\left. \times\, \mathbf{R}^{-1}\left(\mathbf{F} - \mathbf{P}\left(\mathbf{P}^T\mathbf{R}^{-1}\mathbf{P}\right)^{-1}\mathbf{P}^T\mathbf{R}^{-1}\mathbf{F}\right)\right] + ln(det\mathbf{R}) \end{aligned} \tag{4.23}$$

Due to the dependency on the correlation $\mathbf{R}(\vec{\theta})$ on the number of training patterns n_t, the cost of maximizing $ln(l_F(\vec{\theta}|\mathbf{F}))$, and therefore $l_F(\vec{\theta}|\mathbf{F})$, increases as the number of observations n_t increases. In order to reduce the cost of solving this computationally expensive equation, a variety of algorithms are utilized, the most common of which is COBYLA algorithm (Constrained Optimization By Linear Approximation) [36], which uses linear approximations for the objective and constraint functions.

- **Prediction with noisy observations**

In the case of noisy predictions, the correlation matrix $\mathbf{R} \in \mathbb{R}^{n_t \times n_t}$ is no longer orthogonal, since the values in the leading diagonal of the matrix are not equal to 1 due to the introduced errors. In such a case, the least squares estimate given by equations 4.18 and 4.20 will produce values that do not correspond to the physical model. In order to filter the noise, a parameter λ_R, referred to as nugget, is added to the leading diagonal of the matrix [37]. The nugget can be a vector $\vec{\lambda}_R = [\lambda_{R_1}, \lambda_{R_2}, \ldots, \lambda_{R_{n_t}}]$ and vary for each observation or a scalar value λ_R and be constant for all observations. Consequently, the correlation matrix $\mathbf{R}$ is replaced by the term $\mathbf{R} + \vec{\lambda}_R I$ as such:

$$\begin{aligned}
\hat{\mathrm{f}}(\vec{\beta}) &= \vec{p}^{\,T}(\vec{\beta})\hat{\vec{\mathrm{w}}} + \vec{r}_{X\beta}(\mathbf{R} + \vec{\lambda}_R I)^{-1}\left(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}}\right) \\
MSE(\vec{\beta}) &= \hat{\sigma}^2(1 - \vec{r}_{X\beta}^{\,T}(\mathbf{R} + \tilde{\lambda}_{\mathbf{R}}\mathbf{I})^{-1}\vec{r}_{X\beta}) \\
\hat{\vec{\mathrm{w}}} &= \left(\mathbf{P}^T(\mathbf{R} + \vec{\lambda}_R I)^{-1}\mathbf{P}\right)^{-1}\mathbf{P}^T(\mathbf{R} + \vec{\lambda}_R I)^{-1}\mathbf{F} \\
\widehat{\sigma}^2 &= \frac{1}{n_t}(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}})^T(\mathbf{R} + \vec{\lambda}_R I)^{-1}(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}})
\end{aligned} \tag{4.24}$$

where I is the $n_t \times n_t$ identity matrix.

4.2 KPLS

In an attempt to decrease the construction time of Kriging model in high-dimensional design spaces, the number of parameters $\vec{\theta}$ is decreased via the use of Partial Least Squares (PLS) method[38]. PLS is a statistical method used for observing the correlation between the design variables and the objective function by projecting the former in a design space of reduced dimensions h. This space is formed by h parameters, which are called principal components or latent variables, and are linear combinations of the design variables. In KPLS [39], the principal components $\mathbf{P_c} = [\vec{p_c}^{(1)}, \vec{p_c}^{(2)}, \ldots, \vec{p_c}^{(h)}]$ are retained via the implementation of the PLS method which seeks the best direction $\vec{D}^{(l)}$ that maximizes iteratively for h reduced dimensions the covariance between $\vec{p}^{(l)}$ and $\mathbf{F}^{(l-1)}$, where $\mathbf{F}^{(l-1)}$ are the responses at the observed design sites $\mathbf{X}^{(l-1)}$ for the $(l-1)_{th}$ principal component.

$$\vec{D}^{(l)} = \underset{\vec{D}^{(l)}}{\operatorname{argmax}} \ \vec{D}^{(l)^T}\mathbf{X}^{(l-1)^T}\mathbf{F}^{(l-1)}\mathbf{F}^{(l-1)^T}\mathbf{X}^{(l-1)}\vec{D}^{(l)} \ \text{, for } \ l = 1, h \tag{4.25}$$

which is maximized when $\vec{D}^{(l)^T}\vec{D}^{(l)} = 1$, i.e. $\vec{D}^{(l)} = [D_1^{(l)}, D_2^{(l)}, \ldots, D_{n_\beta}^{(l)}]^T$ is the $n_\beta \times 1$ eigenvector that corresponds to the scalar eigenvalue $\lambda_{eig} \in \mathbb{R}$ with the largest absolute value, which is estimated using the power iteration method proposed by Lanczos[40]. Let $\mathbf{\Delta}^{(l-1)} \equiv \mathbf{X}^{(l-1)^T}\mathbf{F}^{(l-1)}\mathbf{F}^{(l-1)^T}\mathbf{X}^{(l-1)}$, then each principal direction vector $\vec{D}^{(l)}$ maximizes the covariance of $\mathbf{\Delta}^{(l-1)}$. For the first iteration of the algorithm, $\mathbf{X}^{(0)} \equiv \mathbf{X} \in \mathbb{R}^{n_t \times n_\beta}$ and $\mathbf{F}^{(0)} \equiv \mathbf{F} \in \mathbb{R}^{n_t}$, assuming a SOO. With $D^{(l)}$ known, the principal component for the l_{th} iteration can be calculated:

$$\vec{p_c}^{(l)} = \mathbf{X}^{(l-1)}\vec{D}^{(l)} \tag{4.26}$$

where $\vec{p_c}^{(l)} = [p_{c_1}^{(l)}, p_{c_2}^{(l)}, \ldots, p_{c_{n_t}}^{(l)}]^T$ is the $n_t \times 1$ principal component vector for the l_{th} principal dimension. Subsequently, the matrices of the design space and its response are calculated and will be used to compute the values in the next iteration.

$$\begin{aligned} \mathbf{X}^{(l)} &= \mathbf{X}^{(l-1)} - \vec{p_c}^{(l)}\vec{\mathrm{w}}_x^{(l)} \\ \mathbf{F}^{(l)} &= \mathbf{F}^{(l-1)} - \mathrm{w}_F^{(l)}\vec{p_c}^{(l)} \end{aligned} \tag{4.27}$$

where $\vec{\mathrm{w}}_x^{(l)}$ and $\mathrm{w}_F^{(l)}$ are the regression coefficients of the l_{th} principal component for the local regression of $\mathbf{X}$ and $\mathbf{F}$, respectively, with the former being a $1 \times n_\beta$ matrix and the latter a scalar. Prior to the initialisation of the iterative process, matrices $\mathbf{X}, \mathbf{F}$ have been scaled and centered on the origin point of the initial coordinate system $O(0, 0, \ldots, 0)$; this has no impact on the correlation matrix $\mathbf{R}$. In addition, each resulting principal component is orthogonal to all the other principal components, since they compose the axes of the new coordinate system.

The completion of the iterative process results in the creation of a formatted design space of $h < n_\beta$ dimensions, which is defined by the coordinate system that the principal components form and is created by rotation of the original design space. This rotation can be quantified by the definition of a new matrix[41]:

$$\mathbf{D}_* = \mathbf{D}\left(\boldsymbol{W}_{\boldsymbol{x}}^T\mathbf{D}\right)^{-1} \tag{4.28}$$

In the previous equation, $\boldsymbol{W_x} = [\vec{w_x}^{(1)^T}, \vec{w_x}^{(2)^T}, \ldots, \vec{w_x}^{(h)^T}]$ is the $n_\beta \times h$ matrix containing the regression coefficients of h principal components for the local regression of $\mathbf{X}$ and $\mathbf{D} = [\vec{D}^{(1)}, \vec{D}^{(2)}, \ldots, \vec{D}^{(h)}]$ is the $n_\beta \times h$ matrix of principal direction vectors. $\mathbf{D}_* = [\vec{D}_*^{(1)}, \vec{D}_*^{(2)}, \ldots, \vec{D}_*^{(h)}] \in \mathbb{R}^{n_\beta \times h}$ is obtained by restating eq. 4.26 as such:

$$\vec{p_c}^{(l)} = \mathbf{X}^{(l-1)} \vec{D}^{(l)} = \mathbf{X}^{(0)} \mathbf{D}_*^{(l)} \tag{4.29}$$

where the scalar elements $D_{*1}^{(l)}, D_{*2}^{(l)}, \ldots, D_{*n_\beta}^{(l)}$ in each vector $\vec{D}_*^{(l)}$ measure the importance of each corresponding dimension in the construction of the l_{th} principal component, where its correlation with the response $\vec{f}$ is maximized. Respectively, the correlation parameters $\vec{\theta} \in \mathbb{R}^{n_\beta}$ in Kriging quantify the importance of each dimension in the calculation of the respective response $\vec{f}$. The estimation of such parameters via maximization of the likelihood function in eq. 4.23 is the most costly process of constructing the Kriging model. By assuming an isotropic and stationary process $R_l : S \times S \mapsto \mathbb{R}$, $\forall l \in [1, h]$, we can construct the KPLS kernel by using the scalar elements $D_{*1}^{(l)}, D_{*2}^{(l)}, \ldots, D_{*n_\beta}^{(l)}$ to replace $\vec{\theta}$ when measuring the importance of each one of the n_β dimensions for the l_{th} principal component.

$$\begin{aligned} R_{1:h}(\vec{\chi_i}, \vec{\chi_j}) &= \prod_{l=1}^{h} R_l\left(f_c^{(l)}(\vec{\chi_i}), f_c^{(l)}(\vec{\chi_j})\right), \quad \text{with } f_c^{(l)} : S \mapsto S \\ \text{and } & \begin{array}{l} \vec{\chi_i} \longmapsto \left[D_{*1}^{(l)}\chi_{i,1}, D_{*2}^{(l)}\chi_{i,2}, \ldots, D_{*n_\beta}^{(l)}\chi_{i,n_\beta}\right] \\ \vec{\chi_j} \longmapsto \left[D_{*1}^{(l)}\chi_{j,1}, D_{*2}^{(l)}\chi_{j,2}, \ldots, D_{*n_\beta}^{(l)}\chi_{j,n_\beta}\right] \end{array} \end{aligned} \tag{4.30}$$

where $f_c^{(l)}$ denotes some correlation function defined in the rotated n_β-dimensional design space of h principal components. This approach can be used to reconstruct two correlation kernels in order to decrease the number of parameters $\theta \in \mathbb{R}^h$:

- Exponential Ornstein-Uhlenbeck process

$$R(\vec{\chi_i}, \vec{\chi_j}) = \prod_{l=1}^{h} \prod_{k=1}^{n_\beta} exp\left(-\theta_l \left| D_{*k}^{(l)}\chi_{i,k} - D_{*k}^{(l)}\chi_{j,k} \right|\right) \tag{4.31}$$

- Gaussian

$$R(\vec{\chi_i}, \vec{\chi_j}) = \prod_{l=1}^{h} \prod_{k=1}^{n_\beta} exp\left(-\theta_l \left(D_{*k}^{(l)}\chi_{i,k} - D_{*k}^{(l)}\chi_{j,k} \right)^2\right) \tag{4.32}$$

The maximum likelihood given by eq.4.23 is subsequently estimated w.r.t. $\theta \in \mathbb{R}^h$, thus significantly decreasing the computational cost. With $\theta \in \mathbb{R}^h$ known, the correlation matrix is calculated and inserted in equations 4.17 and 4.19 that provide the KPLS prediction and the corresponding MSE.

4.3 KPLSK

The KPLSK model is used for improving the maximum likelihood function of Kriging described by equation 4.23. This improved maximum likelihood function is obtained by following the construction process of KPLS model with one variation. After the values of parameters θ have been calculated in the h-dimensional space via the use of KPLS model, KPLSK performs a local optimization of the likelihood function of Kriging by making it equivalent to the KPLS one [42]. The idea is to express the KPLS kernels, which are defined in a subset of the n_β-dimensional space, in the entirety of the n_β-dimensional space. In the subset $S \subset \mathbb{R}^{n_\beta}$, the equivalence of the KPLS and Kriging kernels eq. 4.32 can be proved for exponential kernels of order q. In this case, $q = 2$ to refer to the Gaussian correlation kernel:

$$\begin{aligned} R\left(\vec{\chi_i}, \vec{\chi_j}\right) &= \prod_{l=1}^{h} \prod_{k=1}^{n_\beta} exp\left(-\theta_l \left(D_{*k}^{(l)} \chi_{i,k} - D_{*k}^{(l)} \chi_{j,k}\right)^2\right) \\ &= \prod_{l=1}^{h} \prod_{k=1}^{n_\beta} exp\left(-\theta_l D_{*k}^{(l)^2} \left(\chi_{i,k} - \chi_{j,k}\right)^2\right) \\ &= exp\left(\sum_{k=1}^{n_\beta} \sum_{l=1}^{h} \left(-\theta_l D_{*k}^{(l)^2} \left(\chi_{i,k} - \chi_{j,k}\right)^2\right)\right) \\ &= exp\left(\sum_{k=1}^{n_\beta} \left(-\ \eta_k \left(\chi_{i,l} - \chi_{j,l}\right)^2\right)\right) \\ &= \prod_{k=1}^{n_\beta} exp\left(-\eta_k \left(\chi_{i,k} - \chi_{j,k}\right)^2\right) \end{aligned} \tag{4.33}$$

Consequently, $\eta_k = \sum_{l=1}^{h} \theta_l D_{*k}^{(l)^2}$ for $k = 1, 2, \ldots, n_\beta$ aids in the transition to the n_β-dimensional space, where it serves as a starting point for the local optimization of the Kriging likelihood function based on the values of parameters $\theta^{(l)}$ obtained via the use of the KPLS method for $l = 1, 2 \ldots, h$.

4.4 Radial Basis Function (RBF)

The comprehension of α RBF interpolation model initially requires defining radial functions. A function $\varphi : \mathbb{R}^{n_\beta} \to \mathbb{R}$ is called a radial function when its value at any given point $\vec{\beta} \in \mathbb{R}^{n_\beta}$ depends on the distance r between that point and some other fixed point, called the center of the RBF and denoted by c_e[43].

$$\varphi(\vec{\beta}) = g(\|\vec{\beta} - \vec{c}_e\|) = g(r) \tag{4.34}$$

where $\|\cdot\|$ denotes the Euclidean norm $\|\cdot\|_2$ and $g : [0, \infty) \to \mathbb{R}$ is a univariate radial basis function that depends solely on the distance r. The approximating model uses the n_t observed pairs $(\vec{\chi}, f(\vec{\chi}))$ to yield the interpolant $s(\vec{\beta})$ at an untried point $\vec{\beta} \in \mathbb{R}^{n_\beta}$, which is a linear combination of RBFs[44] $g(r_j)$, $\forall j \in [1, n_t]$.

$$\begin{aligned} s(\vec{\beta}) &= \sum_{j=1}^{n_t} \mathrm{w}_{t_j} g\left(\|\vec{\beta} - \vec{\chi}_j\|\right) = \sum_{j=1}^{n_t} \mathrm{w}_{t_j} g(r_j) \\ &\text{such that} \quad s(\vec{\chi}_i) = F(\vec{\chi}_i) = F_i \quad \text{, for } i = 1, n_t \end{aligned} \tag{4.35}$$

where each observation $\vec{\chi}_j$ serves as a center point for the RBF $g(r_j)$, with the distance between the j_{th} interpolation center and the i_{th} observation being equal to $r_j = \sqrt{(\beta_1 - \chi_{j,1})^2 + (\beta_2 - \chi_{j,2})^2 + \ldots + (\beta_{n_\beta} - \chi_{j,n_\beta})^2}$. Each RBF is additionally weighted by an interpolation coefficient w_{t_i}. The system described by eq. 4.35 is linear and solvable $\forall i, j \in [1, n_t]$:

$$\sum_{j=1}^{n_t} \mathrm{w}_{t_j} g\left(\|\vec{\chi}_i - \vec{\chi}_j\|\right) = F_i \quad \text{, for } i = 1, n_t \tag{4.36}$$

which in matrix form is written as follows:

$$\mathbf{G}\vec{\mathrm{w}}_t = \mathbf{F} \Leftrightarrow \begin{bmatrix} g\|\vec{\chi}_1 - \vec{\chi}_1\| & g\|\vec{\chi}_2 - \vec{\chi}_1\| & \cdots & g\|\vec{\chi}_{n_t} - \vec{\chi}_1\| \\ g\|\vec{\chi}_1 - \vec{\chi}_2\| & g\|\vec{\chi}_2 - \vec{\chi}_2\| & \cdots & g\|\vec{\chi}_{n_t} - \vec{\chi}_2\| \\ \vdots & \vdots & \ddots & \vdots \\ g\|\vec{\chi}_1 - \vec{\chi}_{n_t}\| & g\|\vec{\chi}_2 - \vec{\chi}_{n_t}\| & \cdots & g\|\vec{\chi}_{n_t} - \vec{\chi}_{n_t}\| \end{bmatrix} \begin{bmatrix} \mathrm{w}_{t_1} \\ \mathrm{w}_{t_2} \\ \vdots \\ \mathrm{w}_{t_{n_t}} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_{n_t} \end{bmatrix} \tag{4.37}$$

Consequently, the solution of the linear system results in the calculation of the interpolation coefficients $\vec{\mathrm{w}}_t$ and is executed during the training phase. This is the simplest method of implementing multivariate RBF interpolation.

It is often useful, however, to use a linear combination of conventional RBFs and a linear regression model consisting of low order polynomials, denoted by $p(\vec{\beta})$, and given by[45]:

$$s(\vec{\beta}) = \sum_{j=1}^{n_t} \mathrm{w}_{t_j} g\left(\|\vec{\beta} - \vec{\chi}_j\|\right) + \sum_{i=1}^{k} \mathrm{w}_i p_i(\vec{\beta}) \tag{4.38}$$

where w_i is the coefficient of the i_{th} polynomial and k is their number. The polynomial term $\sum_{i=1}^{k} \mathrm{w}_i p_i(\vec{\beta})$ is identical to the one used in Kriging model. Consequently, the distinction between Kriging and this method lays in the approach of the stochastic term, which in this case is expressed as the linear combination of RBFs.

Equation 4.38 can then be restated in matrix form as a linear system of the following form:

$$\mathbf{G}\vec{\mathrm{w}}_t + \mathbf{P}\vec{\mathrm{w}} = \mathbf{F} \tag{4.39}$$

where $\mathbf{P}$ is the matrix of known polynomials for the complete design presented in eq. 4.14. In order to form a solvable linear system one complementary equation must be added. By arbitrarily assuming that the objective function can be described by the same polynomial matrix $\mathbf{P}$ and a different coefficient matrix $\mathbf{w_d}$, as such $\mathbf{F} = \mathbf{P}\vec{\mathrm{w}}_d$. Consequently, eq. 4.39 can be restated as follows:

$$\begin{aligned} &\mathbf{G}\vec{\mathrm{w}}_t + \mathbf{P}\vec{\mathrm{w}} = \mathbf{P}\vec{\mathrm{w}}_d \rightarrow \\ &\mathbf{G}\vec{\mathrm{w}}_t = \mathbf{P}\left(\vec{\mathrm{w}}_d - \vec{\mathrm{w}}\right) = 0 \xrightarrow{\times \vec{\mathrm{w}}_t^T} \\ &\vec{\mathrm{w}}_t^T\mathbf{G}\vec{\mathrm{w}}_t = \vec{\mathrm{w}}_t^T\mathbf{P}\left(\vec{\mathrm{w}}_d - \vec{\mathrm{w}}\right) = 0 \end{aligned} \tag{4.40}$$

The left hand side must be zero if the following constraint is applied.

$$\vec{\mathrm{w}}_t^T\mathbf{P} = \left(\mathbf{P}^T\vec{\mathrm{w}}_t\right)^T = 0 \tag{4.41}$$

If subsequently this constraint is incorporated in eq. 4.39, the linear system takes the following form:

$$\begin{bmatrix} \mathbf{G} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \vec{\mathrm{w}}_t \\ \vec{\mathrm{w}} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ 0 \end{bmatrix} \tag{4.42}$$

The solution of the linear system results in the calculation of the interpolation coefficients w_t, w and is part of the training process. The polynomials used to facilitate the RBF interpolation can be of order 0,1 or 2 corresponding to a constant, linear or quadratic trend respectively.

Another variation of plain RBFs uses a constant trend that can be obtained from eq. 4.38 by setting $k = 1$ and $p_1(\vec{\beta}) = 1$.

$$s(\vec{\beta}) = \sum_{j=1}^{n_t} \mathrm{w}_{t_j} g\left(\|\vec{\beta} - \vec{\chi}_j\|\right) + \mathrm{w}_1 \tag{4.43}$$

The previous equation for n_t training patterns can be written in matrix form as follows:

$$\begin{bmatrix} \mathbf{G} & \vec{P} \\ \vec{P}^T & 0 \end{bmatrix} \begin{bmatrix} \vec{\mathrm{w}}_t \\ \mathrm{w}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ 0 \end{bmatrix} \tag{4.44}$$

where $\vec{P}$ is the $n_t \times 1$ matrix:

$$\vec{P} = \begin{bmatrix} p_1(\vec{\chi}_1) \\ \vdots \\ p_1(\vec{\chi}_{n_t}) \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4.45}$$

In order to perform the interpolation of n_t sample points, a plethora of radial basis functions $g(r)$ can be used. Some of the most common are presented in the following table:

RBF	**g(r)**	**Scaling parameters**	**Order**
Gaussian	$e^{(-\alpha r)^2}$	$a > 0$	0
Multiquadratic	$\sqrt{r^2 + \alpha^2}$	$a > 0$	1
Inverse Multiquadratic	$(1+(r\alpha)^2)^{-1/2}$	$a > 0$	0
Inverse Quadratic	$(1 + (r\alpha)^2)^{-1}$	$a > 0$	0
Thin Plate Spline	$r^{2c}log(r)$	$c \in \mathbb{N}$	c
Polyharmonic Spline	r^{2c-1}	$c \in \mathbb{N}$	$c - 1$

Table 4.1: Common radial basis functions

Gaussian basis functions are most commonly implemented with the scaling parameter α often being replaced by the parameter $d_0 > 0$, where $a = 1/d_0$. The restated formula that describes Gaussian RBFs is the following:

$$g(r) = g\left(\|\vec{\chi}_i - \vec{\chi}_j\|\right) = exp\left(-\frac{\|\vec{\chi}_i - \vec{\chi}_j\|^2}{d_0^2}\right) \tag{4.46}$$

where the scaling parameter d_0 is used to adjust the shape of the radial basis function $g(r)$ and can therefore affect the accuracy of the RBF interpolation. This parameter can be adjusted via the modifying the parameter $d0$. The effect of this parameter on the shape of the radial basis function $g(r)$ is presented in the following figure.

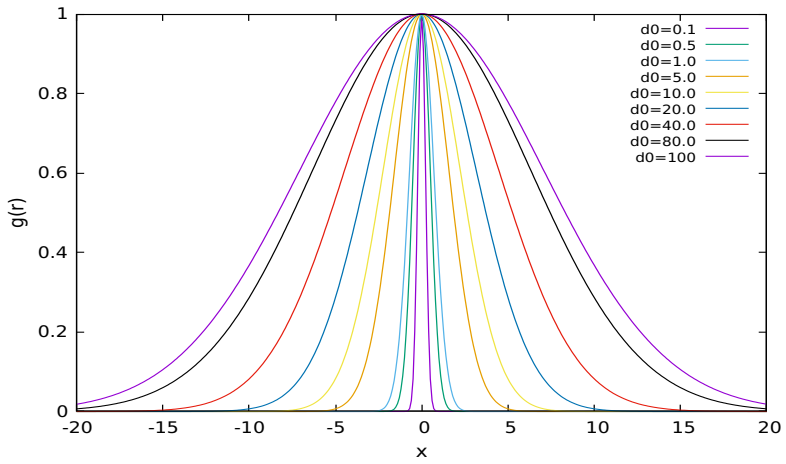


Figure 4.1: $g(\chi)$ RBF shape for various d_0 values when $\chi \in [-20, 20]$

EASY built-in RBFs are described by equation 4.35. In SMT,

Chapter 5

Numerical Cases

The efficacy of the selected metamodels is tested on a pair of pseudo-engineering optimization problems. The study involves a comparison between MAEA-based optimization using the metamodels selected in this thesis, MAEAs using EASY built-in RBF models and plain EAs. The entirety of the evaluations are performed on the multi-processor platform of the PCOpt/NTUA that consists of 3 clusters with combined computational power of 62 Teraflop. The outcome of the evaluation will provide important feedback regarding the potential of the selected surrogate models.

5.1 Welded Beam Design

The first case is a welded beam design [46], a SOO optimization problem where a beam is welded onto a rigid body (see figure 5.1). In this optimization case, the dimensions of the beam and the weld are modified in order for the overall construction cost to be minimized subject to constraints on shear stress, bending stress, buckling load and the end deflection. The design variables to be modified are four, i.e. the thickness of the welds h, the length of the welds l, the height of the beam t and the width of the beam b. Consequently, the vector of design variables assumes the following form $\vec{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4) = (h, l, t, b) \in \mathbb{R}^4$.

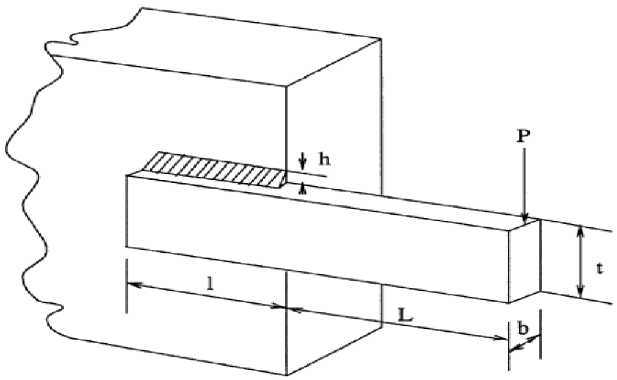


Figure 5.1: Welded beam design

The beam is made of 1010 steel and must be supported by an upper and a lower weld when a constant load $P = 6000$ lb is applied at distance $L = 14\ in$ from the rigid body. The fabrication cost of the welds is given by the equation:

$$f_w = (c_1 + c_2)h^2 l$$

where c_1 is the cost per unit volume of the weld material, c_2 the labour cost per unit weld volume and $V_w = h^2 l$ the volume of the weld material.

The fabrication cost of the beam is proportional to the amount of material in the beam:

$$f_b = c_3 tb(L + l) = c_3 tb(14.0 + l)$$

where c_3 is the cost per unit volume of the beam and $V_b = tb(L + l)$ the respective volume. The construction costs c_1, c_2, c_3 have been estimated:

- For the welds:
  $c_1 = 0.10471\ \frac{\$}{in^3}$ and $c_2 = 1\ \frac{\$}{in^3}$
- For the beam:
  $c_3 = 0.04811\ \frac{\$}{in^3}$

The overall fabrication cost can be written as:

$$f_{tot} = f_w + f_b = 1.10471h^2 l + 0.04811tb(14.0 + l) \tag{5.1}$$

The stress states that describe the optimization case are subsequently defined and will serve as the imposed constraints. The first is the shear stress of the welds that must not exceed the maximum allowable shear stress of the material $\tau_{max} \leq 13600$ psi. The shear stress of the welds is defined as such:

$$\tau = \sqrt{\tau_p^2 + 2\tau_p\tau_t cos\theta + \tau_t^2} \xrightarrow{cos\theta = l/2R} \sqrt{\tau_p^2 + \frac{l\tau_p\tau_t}{R} + \tau_t^2} \tag{5.2}$$

where R is the distance from the center of the cross-section of the beam, τ_p the primary stress of the weld throat and τ_t the torsional stress developed on the beam due to the torque M developed by the applied load P at its end. The equations describing the aforementioned static mechanical phenomena are the following:

$$\begin{aligned} R &= \sqrt{\frac{l^2}{4} + \left(\frac{h+t}{2}\right)^2} \\ \tau_p &= \frac{P}{\sqrt{2}hl} = \frac{6000}{\sqrt{2}hl} \\ \tau_t &= \frac{MR}{J} \\ M &= P\left(L + \frac{l}{2}\right) = 6000\left(14.0 + \frac{l}{2}\right) \end{aligned} \tag{5.3}$$

In torsional stress equation, the variable J is the polar moment of inertia of the weld:

$$J = 2\sqrt{2}hl\left[\frac{l^2}{12} + \left(\frac{h+t}{2}\right)^2\right]$$

The second stress that affects the quality of the design is the normal bending stress of the beam that must not exceed the maximum yield strength of the material $\sigma_{max} \leq 30000$ psi and is equal to:

$$\sigma = \frac{6PL}{bt^2} = \frac{504000}{bt^2} \tag{5.4}$$

The deflection at the end of the beam is the next constraint that must be incorporated into the optimization of the welded beam. The deflection of a cantilever beam of length $L = 14\ in$ must not exceed $\delta_{max} \leq 0.25\ in$ and is calculated as such:

$$\delta = \frac{4PL^3}{Ebt^3} = \frac{2.1952}{bt^3} \tag{5.5}$$

where E is the Young's modulus; for 1010 steel is equal to 30×10^6 psi.

Additionally, the structural integrity of the beam requires that the buckling load in the vertical direction must be greater than the applied load $P = 6000$ psi. The critical buckling load of the beam is calculated as such:

$$P_c = \frac{4.013\sqrt{\frac{EGt^2b^6}{36}}}{L^2}\left(1 - \frac{t}{2L}\sqrt{\frac{E}{4G}}\right) \tag{5.6}$$

where G is shearing modulus; for steel 1010 is equal to $G = 12 \times 10^6$ psi.

It is evident that the thickness of the welds h should not exceed the width of the beam and therefore the last imposed constraint is $h \leq b$. Consequently, the optimization of the welded beam design requires the minimization of a single objective, i.e. the fabrication cost of the structural design, in the 4-dimensional space formed by the design variables $\vec{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4) = (h, l, t, b) \in \mathbb{R}^4$ and bounded by the 5 imposed constraints.

$$\begin{aligned}
min\ \ & f(\vec{\beta}) = 1.10471\beta_1^2\beta_2 + 0.04811\beta_3\beta_4(14.0 + \beta_2) \\
\text{subject to}\ \ & c_1(\vec{\beta}) = \tau(\vec{\beta}) - \tau_{max} \leq 0 \\
& c_2(\vec{\beta}) = \sigma(\vec{\beta}) - \sigma_{max} \leq 0 \\
& c_3(\vec{\beta}) = \beta_1 - \beta_4 \leq 0 \\
& c_4(\vec{\beta}) = \delta(\vec{\beta}) - \delta_{max} \leq 0 \\
& c_5(\vec{\beta}) = P - P_c(\vec{\beta}) \leq 0
\end{aligned} \tag{5.7}$$

where the bounds of each design variable are $0.125 \leq \beta_1 \leq 10.0$, $0.1 \leq \beta_2 \leq 10.0$, $0.1 \leq \beta_3 \leq 10.0$ and $0.1 \leq \beta_4 \leq 10.0$. The formulas that describe $\tau(\vec{\beta})$, $\sigma(\vec{\beta})$, $\delta(\vec{\beta})$ and $P_c(\vec{\beta})$ can be found in equations 5.2, 5.4, 5.5 and 5.6, respectively.

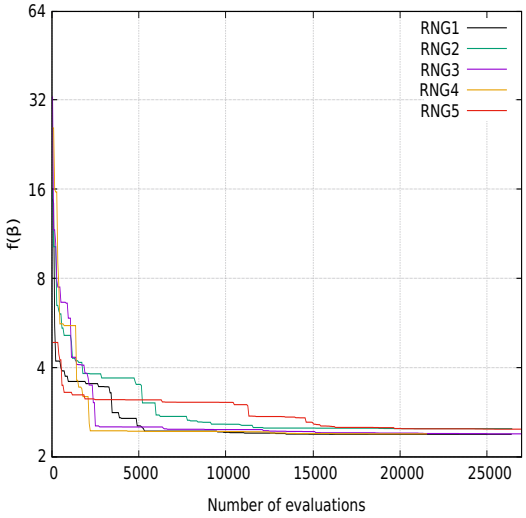
- **Optimization using EAs**

First, the welded beam design is optimized using plain EAs that utilise the problem-specific evaluation model. The optimization is performed using EASY software in order to identify the most suitable values for the parameters of the evolution, e.g. offspring and parents population size, mutation probability and crossover scheme. The evolution parameters identified via the use of EAs are later used to facilitate the evolution in MAEA-based optimization. The number of offspring and parent population is set to $(\mu, \lambda) = (20, 60)$ where 4 parents are combined to create a single offspring with one-point crossover. Gray binary encoding is used and 15 bits are assigned to each design variable. The optimization phase terminates after 27000 PSM evaluations have been performed and is repeated for 5 randomly initialised offspring populations P_λ^0 via the use of a Random Number Generator (RNG). The results are presented in table 5.1 and in figure 5.2.

Welded beam case					
	(μ, λ) **population**	**Best**	**Worst**	**Average**	**Average PSM eval.**
EAs	(20, 60)	2.38	2.49	2.43	27000

Table 5.1: Optimization of welded beam design using EAs

(a) Comparison between the convergence histories of 5 different λ initializations



(b) Convergence history of the optimal run

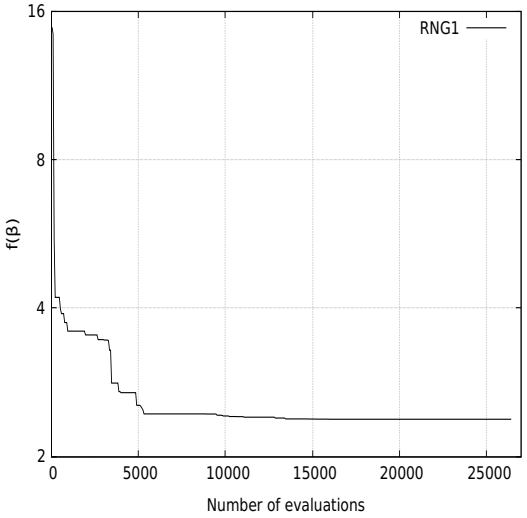


Figure 5.2: Convergence history of welded beam optimization case using EAs

The design variable vector that minimizes the construction cost of the welded beam using plain EAs initialized via RNG1 is $\vec{\beta} = [0.244, 6.194, 8.329, 0.244]$.

- **Optimization using MAEAs with off-line trained metamodels**

In MAEAs using off-line trained metamodels, both the objective $\mathbf{F}(\vec{\beta})$ and the imposed constraints $\mathbf{C}(\vec{\beta}) = [\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_{n_t}]^T$, where $\vec{c}_i = [c_{i,1}, c_{i,2}, \ldots, c_{i,n_c}]$, are approximated using surrogate models. Specifically, a global metamodel is built on the single objective and n_c unique metamodels on each imposed constraint. In this case, the objective function is approximated by a KPLS model, while each constraint is approximated via the use of Kriging model; responsible for the construction of the aforementioned metamodels is SMT software. Alternatively, a single surrogate model can be trained to approximate the entirety of the constraints but this approach resulted in a poorly trained surrogate model. However, even the first approach resulted in surrogate models with poor overall fitting, especially when approximating a function with design variables in the denominator that tend to zero. To solve this issue an approach is proposed where constraints with denominators that tend to 0 are reduced to polynomials. Let the original approach of unmodified constraints be case 1 and let the modified approach be case 2, then:

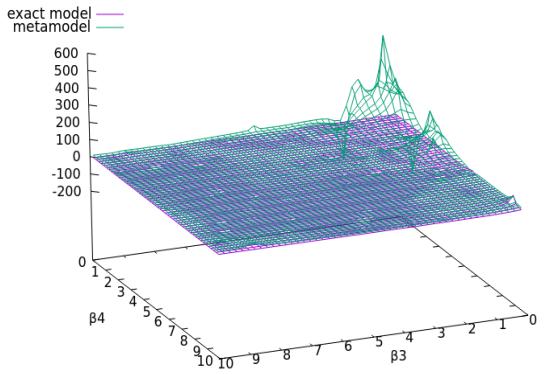
$$c_1(\vec{\beta}) = \sqrt{\tau_p^2 + \frac{\tau_p \tau_t \beta_2}{R} + \tau_t^2} - \tau_{max} \leq 0 \Rightarrow \quad \tau_p^2 + \frac{\tau_p \tau_t \beta_2}{R} + \tau_t^2 \leq \tau_{max}^2 \xrightarrow{R>0}$$

$$\tau_p \tau_t \beta_2 \leq -R\left[\tau_p^2 + \tau_t^2 - \tau_{max}^2\right] \xrightarrow{\tau_p \tau_t \beta_2 > 0} c_1(\vec{\beta})_{new} = \frac{R\left[\tau_p^2 + \tau_t^2 - \tau_{max}^2\right]}{\tau_p \tau_t \beta_2} - 1 \leq 0 \tag{5.8}$$

$$c_2(\vec{\beta}) = \frac{6PL}{\beta_4 \beta_3^2} - \sigma_{max} \leq 0 \xrightarrow{\beta_3, \beta_4 > 0} c_2(\vec{\beta})_{new} = 6PL - \sigma_{max} \beta_4 \beta_3^2 \leq 0 \tag{5.9}$$

$$c_4(\vec{\beta}) = \frac{4PL^3}{E \beta_4 \beta_3^3} - \delta_{max} \leq 0 \xrightarrow{\beta_3, \beta_4 > 0} c_4(\vec{\beta})_{new} = 4PL^3 - \delta_{max} E \beta_4 \beta_3^3 \leq 0 \tag{5.10}$$

(a) Case 1: Comparison between $c_4(\vec{\beta})$ and $\widehat{c}_4(\vec{\beta})$



(b) Case 2: Comparison between $c_4(\vec{\beta})$ and $\widehat{c}_4(\vec{\beta})$

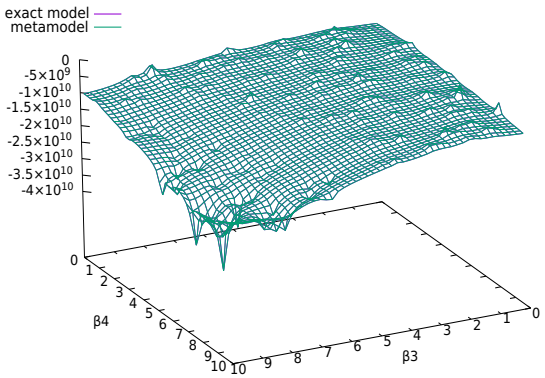
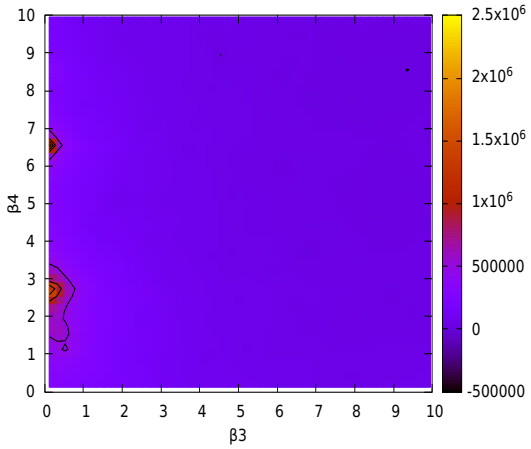
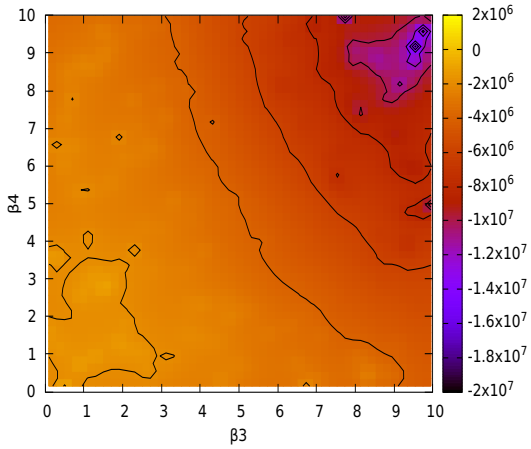


Figure 5.3: Error of the metamodel when approximating the original NRMSE = 1.206111 (left) and the modified NRMSE = $1.182408 \cdot 10^{-6}$ (right) equation $c_4(\vec{\beta})$ using an LHD composed of $n_t = 240$ training patterns

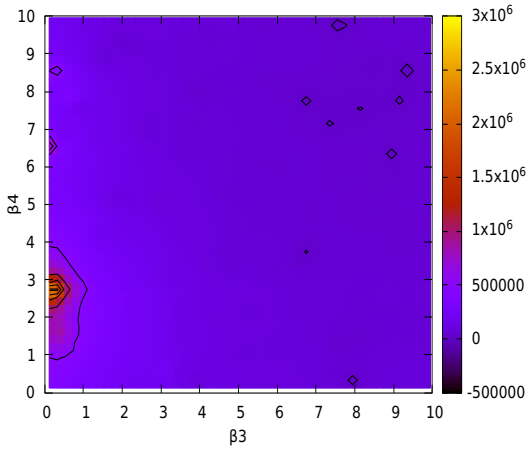
(a) Case 1: $c_2(\vec{\beta})$ function



(b) Case 2: $c_2(\vec{\beta})$ function



(c) Case 1: Prediction $\widehat{c}_2(\vec{\beta})$ function



(d) Case 2: Prediction $\widehat{c}_2(\vec{\beta})$

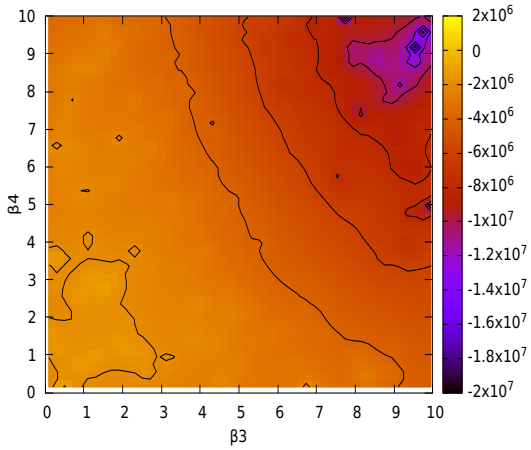


Figure 5.4: Contour projections to 2D plane for the original (left) and modified (right) function of the 2nd constraint

In both figures, i.e. 5.3 and 5.4, the visualization of the constraint function justifies the initial assumption of underfitted metamodels built $\forall\vec{\beta} \in \mathbb{R}^n_\beta$ when $\beta_1, \beta_4 \to 0$; such metamodels are trained in case 1. In case 2, the modification in constraints $c(\vec{\beta}_1), c(\vec{\beta}_2)$ and $c(\vec{\beta}_4)$ results in a better model fitting, NRMSE $= 1.182408 \cdot 10^{-6}$ compared to NRMSE $= 1.206111$ of case 1. The minimization of the welded beam case that through 5 runs is presented in table 5.2, where a maximum of 5000 evaluations per cycle are performed using MAEAs with off-line trained metamodels:

Case 2						
	(μ, λ) **population**	**Best**	**Worst**	**Average**	**Avg. metamodel eval./cycle**	**Avg. cycles**
MAEAs, off-line	(20, 60)	2.35	2.56	2.44	3168	3

Table 5.2: Optimization of welded beam design using MAEAs with off-line training

The optimal candidate solution obtained via this method is $\vec{\beta} = [0.336, 5.067, 7.323, 0.338]$. The corresponding value of each constraint and objective is presented in table 5.3.

Case 2	$\mathbf{c_1}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_2}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_3}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_4}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_5}(\vec{\boldsymbol{\beta}})$	$\mathbf{f}(\vec{\boldsymbol{\beta}})$
MAEAs	-0.412243	-29020.84	-0.019	-1065388211	-276.80	2.35
PSM	1129.41	-1633.52	-0.019	-0.235446	-261.95	2.35

Table 5.3: ***C****,* ***F*** *responses to* $\vec{\beta}$ *found via MAEAs with off-line training in case 2*

MAEAs with off-line training utilizing the approach of modified constraints (case 2) converge in candidate solutions that violate the first constraint $c_1(\vec{\beta})$. In order to determine the extent to which the design space has changed, the aforementioned approach is implemented in plain EAs that utilize the modified problem-specific model, denoted by PSM$'$ for simplicity. If PSM$'$-based EAs converge to an optimal solution that lies in the design space of the original PSM, then the modification in constraints did not lead to a significant change in the design space of candidate solutions and the unsatisfactory solutions are contributed to metamodel-related flaws. EAs using PSM$'$ find the optimal solution $\vec{\beta} = [0.272, 4.300, 7.856, 0.272]$. The corresponding value of each constraint and objective is presented in table 5.4

Case 2	$\mathbf{c_1}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_2}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_3}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_4}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_5}(\vec{\boldsymbol{\beta}})$	$\mathbf{f}(\vec{\boldsymbol{\beta}})$
PSM$'$	-0.000037	-273.75	0	-963206327	-529.75	2.15
PSM	3445.91	-16.286	0	-0.234001	-529.75	2.15

Table 5.4: ***C****,* ***F*** *responses to* $\vec{\beta}$ *found via EAs using the* PSM$'$

The implemented modifications seem to distort the design space significantly, since the optimal solution results in a design with welds that undergo massive shear stress $c_1(\vec{\beta}) = 17045.91$ psi. Consequently, optimal solutions found in case 2 do not satisfy the constraints imposed on the design space and result in a unsatisfactory welded beam design. However, metamodels trained off-line on the PSM do not converge to an optimal solution due to poor constraint model fitting. For this reason, a new approach is proposed, referred to as case 3, and is based on the observation that the entirety of optimal solutions found via the use both the PSM$'$ and metamodels trained on the PSM$'$(case 2) do not satisfy the first constraint $c_1(\vec{\beta})$. In case 3, therefore, constraints c_2, c_4 are modified according to equations 5.9 and 5.10, respectively, and the corresponding problem- specific model is denoted by PSM$''$. The implementation of EAs via the use of PSM$''$ yields the optimal solution $\vec{\beta} = [0.279, 5.679, 7.698, 0.284]$. The corresponding value of each constraint and objective is presented in table 5.5.

Case 3	$\mathbf{c_1}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_2}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_3}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_4}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_5}(\vec{\boldsymbol{\beta}})$	$\mathbf{f}(\vec{\boldsymbol{\beta}})$
PSM$''$	-13.645	-355.91	-0.004	-904782848	-2906.86	2.56
PSM	-13.645	-21.170	-0.004	-0.233038	-2906.86	2.56

Table 5.5: ***C****,* ***F*** *responses to* $\vec{\beta}$ *found via EAs using the* PSM$''$

Unlike previous approaches, PSM$''$-based EAs converge to an optimal solution that satisfies the entirety of the constraints imposed to the original model. Metamodels trained off-line on the PSM$''$ through 5 runs yield the outcome shown in table 5.6.

Case 3						
	(μ, λ) **population**	**Best**	**Worst**	**Average**	**Average metamodel eval./cycle**	**Avg. cycles**
MAEAs, off-line	(20, 60)	2.90	3.55	3.12	2883	8

Table 5.6: Optimization of welded beam design using MAEAs with off-line training

The best solution $\vec{\beta} = [0.336, 5.067, 7.323, 0.338]$ obtained via MAEAs with off-line training, produces the constraint and objective function values shown in table 5.7 when evaluated on the PSM.

Case 3	$\mathbf{c_1}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_2}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_3}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_4}(\vec{\boldsymbol{\beta}})$	$\mathbf{c_5}(\vec{\boldsymbol{\beta}})$	$\mathbf{f}(\vec{\boldsymbol{\beta}})$
MAEAs	-469.56	-39385.13	-0.002	-928918812	-8492.04	2.90
PSM	-797.35	-2194.43	-0.002	-0.233450	-8522.98	2.90

*Table 5.7: **C**, **F** responses to $\vec{\beta}$ found via MAEAs with off-line training in case 3*

Consequently, the MAEA-based optimization of the welded beam case is performed by utilizing metamodels built off-line exclusively on PSM$''$, since MAEAs with surrogate models trained off-line on PSM$'$ and PSM either converge to prohibited by the constraints solutions or they do not converge after performing 20 optimization cycles, respectively. The convergence histories of EAs using PSM, PSM$'$ and PSM$''$ are subsequently presented in figure 5.5 all EA methods are initialized with the same offspring population set P_λ^0 corresponding to best solution.

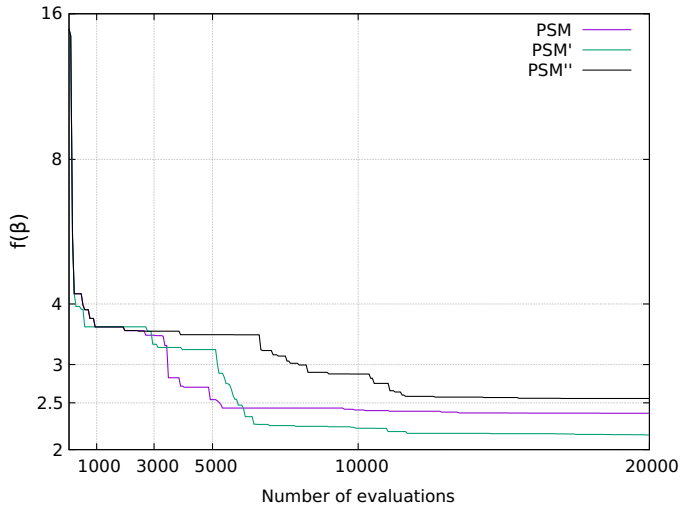


Figure 5.5: Welded beam design. Comparison between the convergence history of EAs performing evaluations on the PSM, PSM$'$ *and* PSM$''$

The deviation between metamodel predicted values and evaluated ones, using the PSM$''$ of either objective function or some constraint, can be observed in figures 5.6, 5.7 and 5.8. The surrogate models are trained on $n_t = n_{doe} = 240$ patterns $\mathbf{X}$ collected via the implementation of LHS that makes use of the ESE algorithm to construct an optimal space-filling design. At the end of each optimization loop, a new random design of $n_{new\_doe} = 20$ points and 1 elite are added to the initial LHD. The optimization process converged after 10 cycles and, therefore, at the end of the optimization the metamodels are retrained on $n_t = n'_{doe} = 429$ patterns. Each individual $\vec{\beta} \in P_\lambda^0$ selected in the first of the 5 total runs is used to validate both metamodels and calculate the NRMSE.

(a) Initial f, NRMSE$=0.00000269$ *(b) Final f,* NRMSE$=0.00000042$

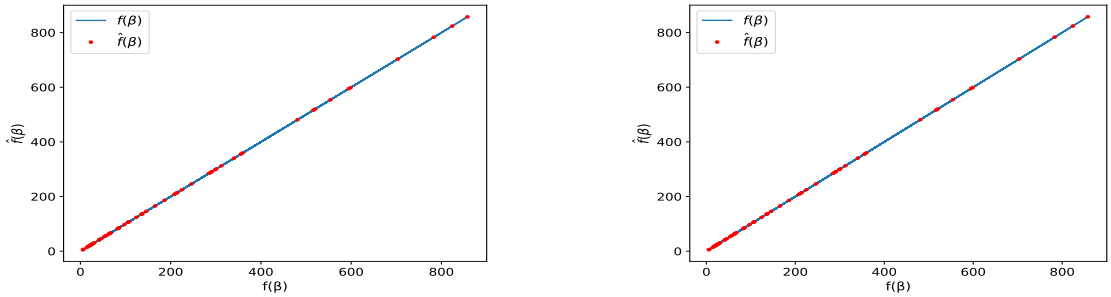


Figure 5.6: Case 3, welded beam case with RNG1. Deviation between exact PSM$''$ *values of the objective function $\vec{f}(\vec{\beta})$ and KPLS predictions $\hat{\vec{f}}(\vec{\beta})$. The initial KPLS model is trained on $n_{doe} = 240$ and the final on $n'_{doe} = 429$ training patterns.*

(a) Initial c_1, NRMSE$=0.7773964$ *(b) Final c_1,* NRMSE$=0.55329077$

(c) Initial c_2, NRMSE$=0.00000249$ *(d) Final c_2,* NRMSE$=0.00000063$

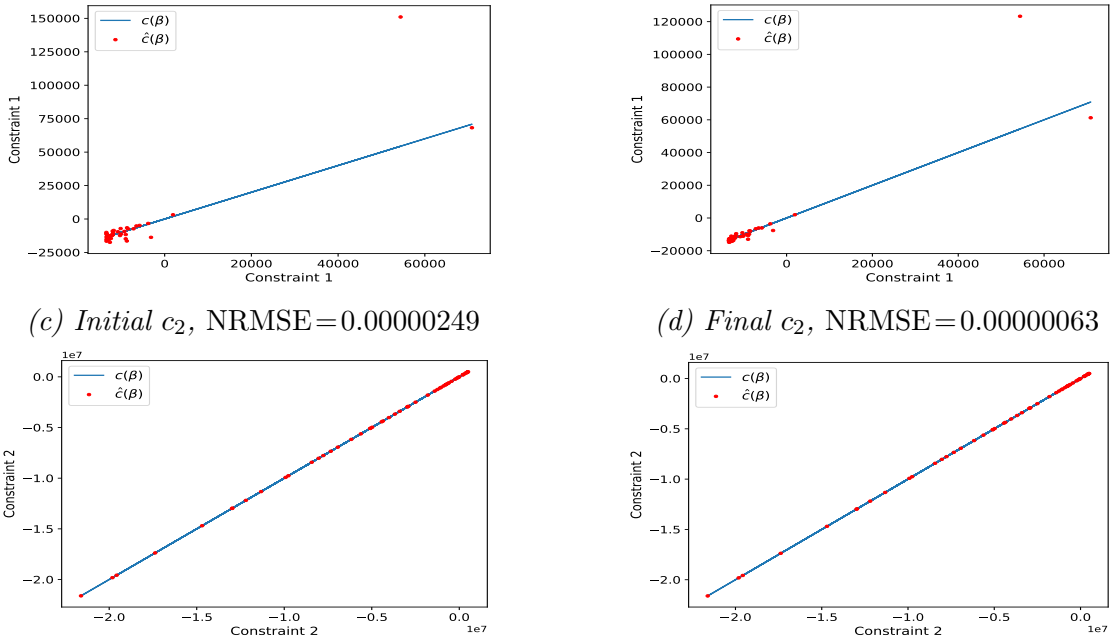
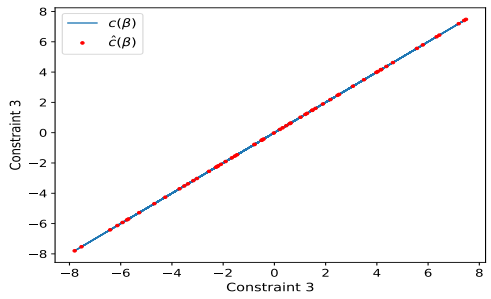
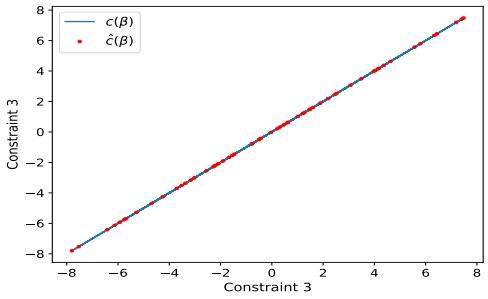


Figure 5.7: Case 3, welded beam case with RNG1. Deviation between exact PSM$''$ *constraint values $\vec{c}(\vec{\beta})$ and Kriging predictions $\hat{\vec{c}}(\vec{\beta})$ given via the implementation of SMT. The initial Kriging model, which approximates the first two constraint functions, is trained on $n_{doe} = 240$ and the final on $n'_{doe} = 429$ training patterns.*

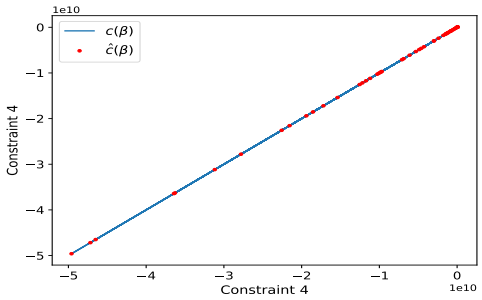
(a) Initial c_3, NRMSE = 0.00000008



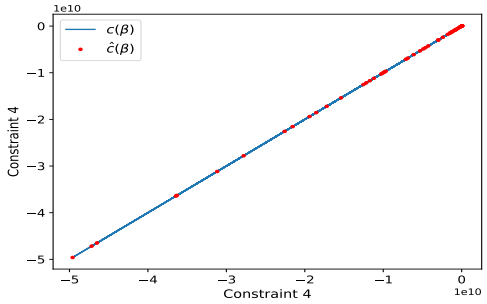
(b) Final c_3, NRMSE = 0.00000003



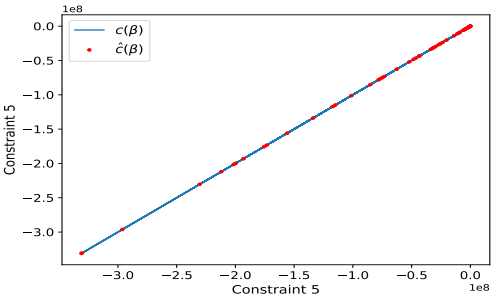
(c) Initial c_4, NRMSE = 0.00000617



(d) Initial c_4, NRMSE = 0.00000131



(e) Final c_5, NRMSE = 0.00000053



(f) Final c_5, NRMSE = 0.00000014

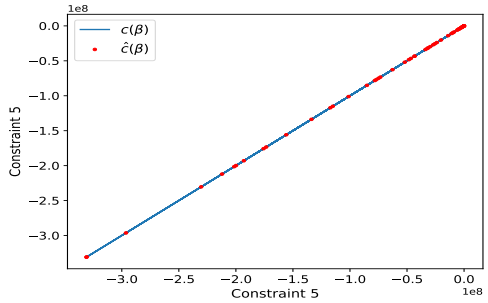


Figure 5.8: Case 3, welded beam case with RNG1. Deviation between exact PSM constraint values $\vec{c}(\vec{\beta})$ *and RBF predictions* $\hat{\vec{c}}(\vec{\beta})$ *given via the implementation of SMT. The initial Kriging model, which approximates the remaining three constraint functions, is trained on* $n_{doe} = 240$ *and the final on* $n'_{doe} = 429$ *training patterns.*

The NRMSE calculates the mean normalised deviation between the predicted and exactly evaluated on the PSM$''$ values on $\lambda = 60$ untried design sites $\mathbf{B}$, where $\vec{\beta} = \vec{\beta}_i = [\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,n_\beta}] \in P_\lambda^0$ is any untried point in the design space contained in the initial offspring population set. NRMSE serves as a metric of model fitting and indicates that all trained metamodels are well-fitted, except from the one built on the 1st constraint function $c_1(\vec{\beta})$. This underfitted surrogate model hinders the convergence of the optimization process, which reaches an satisfactory optimal solution after 8 cycles.

- **Optimization using MAEAs with on-line trained matamodels**

In the MAEA-based optimization of welded beam design using on-line trained metamodels, the LCPE phase is set to initiate once 480 exact evaluations are performed. In LCPE, personalised local metamodels are trained on $20 \leq n_t \leq 21$ training patterns and subsequently $2 \leq \lambda_e \leq 4$ individuals are re-evaluated using the exact evaluation model. A total of 10000 PSM evaluations are performed, unless 75 generations are formed without improving the current outcome, in which case the optimization terminates. Local metamodels are built by utilizing either the assisting software SMT or EASY. The former accommodates the use of Kriging, KPLS, KPLSK and RBFs, while the latter relies on Kriging and most often RBFs. In order to identify the most suitable metamodel in SMT for the welded beam optimization case, a comparison between each respective model is performed w.r.t. the convergence history and the produced outcome; the results of each comparison are presented in table figure 5.9 and table 5.8, respectively.

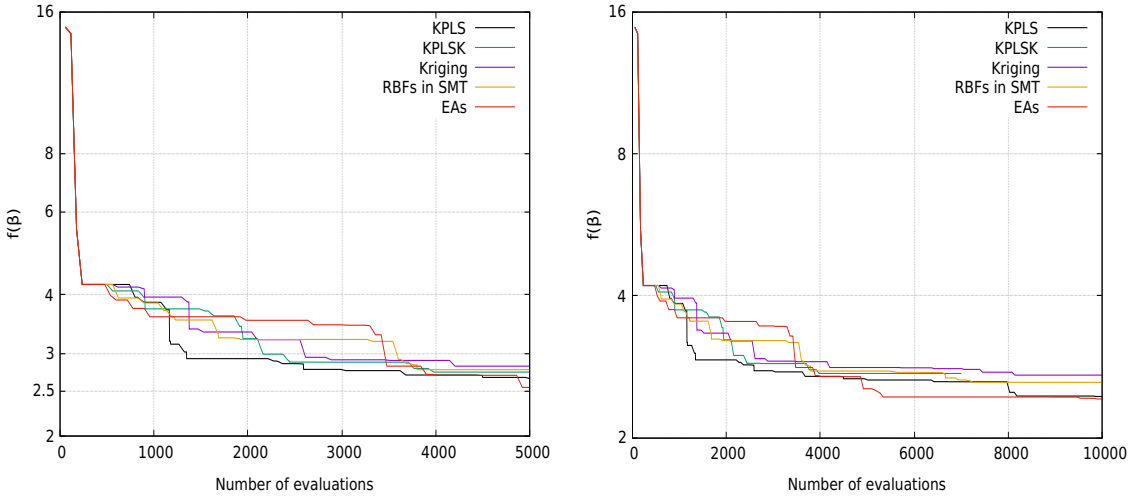


Figure 5.9: Welded beam design with RNG1. Comparison between the convergence history of EAs and MAEAs with metamodels trained on-line via SMT

Welded beam case					
MAEAs, on-line	(μ, λ) **population**	**KPLS**	**KPLSK**	**Kriging**	**RBFs**
SMT	(30, 100)	2.45	2.74	2.72	2.62

Table 5.8: Welded beam case with RNG1. Optimal candidate solution found using metamodels trained on-line via SMT

The comparison between convergence histories of the SMT built-in metamodels, depicted in figure 5.9, indicates that KPLS model is most suitable to facilitate the optimization of the welded beam case due to its fast convergence to the threshold of both 5000 and 10000 PSM evaluations. In plain EAs, RNG1 yields the best optimization outcome and for that reason MAEAs with on-line training are initialized with the same offspring population P_λ^0.

KPLS is yet to be compared to EASY built-in RBFs and plain EAs; the results are presented in table 5.9.

Welded beam design						
MAEAs, on-line	(μ, λ) **population**	**Best**	**Worst**	**Average**	**Avg. exact eval.**	**Avg. meta-model eval.**
SMT	(20, 60)	2.45	2.62	2.54	10000	11579
EASY	(20, 60)	2.38	2.82	2.53	10000	10738
Plain EAs	(20, 60)	2.42	3.05	2.59	10000	-

Table 5.9: Welded beam case. Comparison between the outcome of the optimization using MAEAs with on-line training and plain EAs

The design variable vector that minimizes the construction cost of the welded beam via the implementation of KLPS is $\vec{\beta} = [0.234, 5.717, 9.276, 0.239]$. For MAEAs utilizing EASY built-in RBFs the respective optimal design variable vector is $\vec{\beta} = [0.255, 5.664, 8.527, 0.260]$.

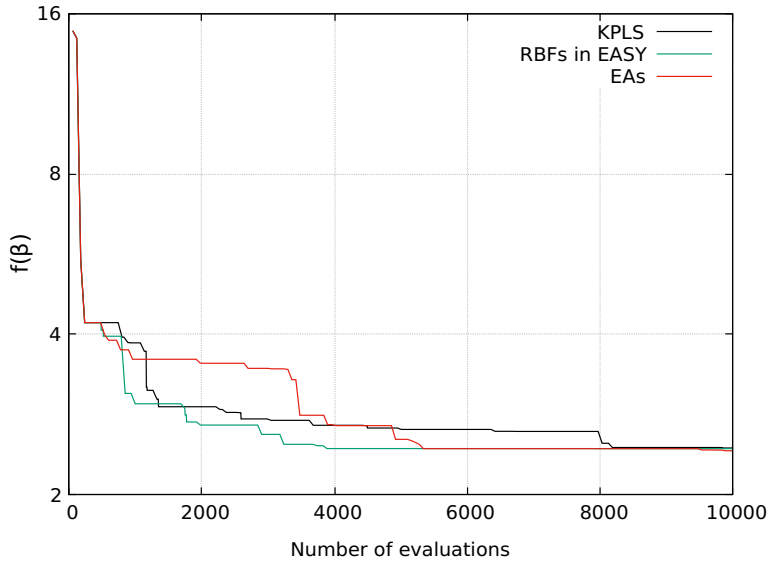


Figure 5.10: Welded beam case with RNG1. Comparison between the convergence histories of the optimization using EAs and MAEAs with metamodels trained on-line via SMT and EASY

In the comparison of convergence histories depicted in figure 5.10, EASY built-in RBFs are seemingly more accurate than KPLS model and yield a faster convergence. The impact of MAEAs on the computational cost is evident, since they outperform conventional EAs prior to the threshold of 10000 exact PSM evaluations.

5.1.1 MOO of Welded Beam Design

The welded beam case appears in the majority of the scholar literature as SOO problem, where the single objective is the minimization of the fabrication cost of the design. However, a variation of this optimization case also exists where the welded beam design is optimized w.r.t. to two objectives, which are the fabrication cost of the design and the deflection $\delta(\vec{\beta})$ of the beam. In this case, the deflection of the beam does not bound the design space of possible candidate solutions and the optimization problem assumes the following mathematical expression:

$$\begin{aligned} min \;\; & f_1(\vec{\beta}) = 1.10471\beta_1^2\beta_2 + 0.04811\beta_3\beta_4(14.0+\beta_2) \\ & f_2(\vec{\beta}) = \frac{2.1952}{\beta_4\beta_3^3} \\ \text{subject to} \;\; & c_1(\vec{\beta}) = \tau(\vec{\beta}) - \tau_{max} \leq 0 \\ & c_2(\vec{\beta}) = \sigma(\vec{\beta}) - \sigma_{max} \leq 0 \\ & c_3(\vec{\beta}) = \beta_1 - \beta_4 \leq 0 \\ & c_4(\vec{\beta}) = P - P_c(\vec{\beta}) \leq 0 \end{aligned} \tag{5.11}$$

where the bounds of each design variable are $0.125 \leq \beta_1 \leq 10.0$, $0.1 \leq \beta_2 \leq 10.0$, $0.1 \leq \beta_3 \leq 10.0$ and $0.1 \leq \beta_4 \leq 10.0$. The formulas that describe $\tau(\vec{\beta})$, $\sigma(\vec{\beta})$ and $P_c(\vec{\beta})$ can be found in equations 5.2, 5.4 and 5.6, respectively. In this case, the objectives are conflicting and their corresponding values are presented in figure 5.11.

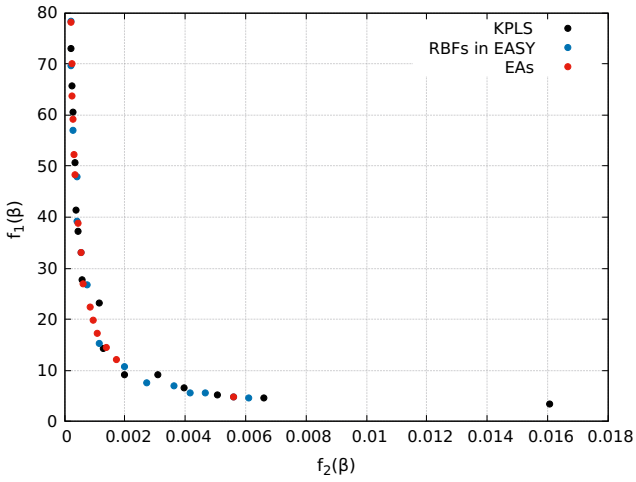


Figure 5.11: Pareto front of 15 non-dominated candidate solutions found in the MOO welded beam case for 1000 exact PSM evaluations

The first two fronts are obtained by optimizing the MOO welded beam case via the use of MAEAs with on-line training, namely EASY built-in RBFs and the KPLS model found in SMT. The LCPE phase is set to initiate once 240 exact evaluations are performed on the PSM. In LCPE, personalised local metamodels are trained on $20 \leq n_t \leq 21$ training patterns and subsequently $2 \leq \lambda_e \leq 6$ individuals are re-evaluated using the exact evaluation model. The comparison is completed by the front that results from the optimization of the MOO case via the use of plain EAs.

Since all fronts are seemingly overlapping, it is not evident which method yielded the best outcome. A way to determine this, is by calculating the hypervolume indicator [47] of each front $\mathcal{F} \subset \mathbb{R}^n$, which is defined as the measure of the region weakly dominated by $\mathcal{F}$ and bound by a reference point $\vec{x}_r \in \mathbb{R}^n$ and expressed as:

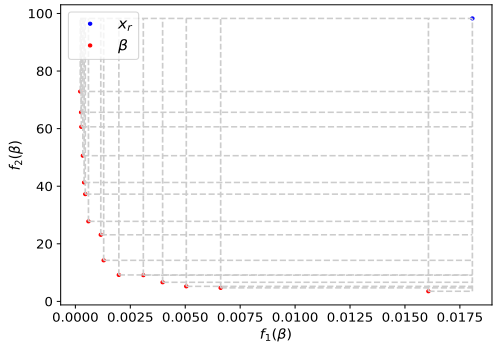
$$H(\mathcal{F}) = \Lambda\left(\{\vec{p} \in \mathbb{R}^n | \ \exists \vec{q} \in \mathcal{F} : \vec{q} \leq \vec{p} \text{ and } \vec{p} \leq \vec{x}_r\}\right) \tag{5.12}$$

where $H(\cdot)$ denotes the Lebesgue measure which is a way of assigning measure to subsets $\mathcal{F}$ of n-dimensional Euclidean space. In the 2D space, which is the case here, the Lebesgue measure $H(\mathcal{F})$ is equivalent to the area defined by each $\vec{q} \in \mathcal{F}$ and the reference point $\vec{x}_r \in \mathbb{R}^n$. The coordinates (x_{r_1}, x_{r_2}) of the reference point in 2D space are defined as:

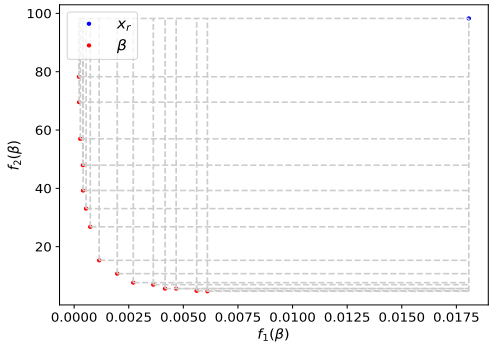
$$\begin{aligned} x_{r_1} &= \{q_1 \in \mathcal{F}_i : q_1 \geq \xi, \forall \xi \in \mathcal{F}_i, \forall i \in [1, n_f]\} + \xi_1 \\ x_{r_2} &= \{q_2 \in \mathcal{F}_i : q_2 \geq \xi, \forall \xi \in \mathcal{F}_i, \forall i \in [1, n_f]\} + \xi_2 \end{aligned} \tag{5.13}$$

where q_1, q_2 are the coordinates of $\vec{q} \in \mathcal{F}$ point in 2D space, n_f is the number of compared fronts and ξ_1, ξ_2 user-defined values. In this case, the parameters assume the values $(\xi_1, \xi_2) = (0.002, 20)$ and $(x_{r_1}, x_{r_2}) = (0.0181, 98.27)$ and yield the hypervolume indicators for each Pareto front shown in figure 5.12.

(a) KPLS, $H = 1.6222$



(b) RBFs in EASY, $H = 1.6215$



(c) EAs, $H = 1.6061$

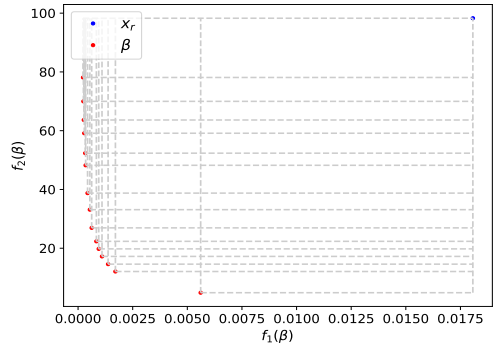


Figure 5.12: MOO welded beam case. Hypervolume indicator H for fronts formed via the use of EAs, on-line trained KPLS and RBFs in EASY

According to the H value, the best front is the one formed via the use of on-line trained KPLS. Consequently, $\vec{\beta} = [0.481, 3.383, 6.572, 0.481]$ is selected that yields the response $(f_1, f_2) = (6.41,\ 0.0031)$, since a minor increase in beam deflection yields a significant decrease in the fabrication cost.

5.2 Speed Reducer Design

The second optimization case is the SOO problem of minimizing the overall weight of a speed reducer. This design is used to reduce the output speed by increasing the output torque via the use two gears that are mounted to two separate shafts of diameter d_1 and d_2. The structure is enclosed within a housing, while a pair of pairings is used at the connection point of each shaft in order to reduce friction produced by the rotation movement of the shafts (see figure 5.13). The minimization of the overall weight of the structure is, therefore, refers to the minimization of the total weight of both gears and shafts. The speed reducer case[49] is optimized w.r.t. the following design variables:

- Face width of the gear b in [cm], where $2.6 \leq \beta_1 \leq 3.6$
- Teeth module m in [cm], where $0.7 \leq \beta_2 \leq 0.8$
- Number of pining teeth N_{teeth}, where $17 \leq \beta_3 \leq 28$
- Length between bearings of the first shaft L_1 in [cm], where $7.3 \leq \beta_4 \leq 8.3$
- Length between bearings of the second shaft L_2 in [cm], where $7.3 \leq \beta_5 \leq 8.3$
- Diameter of the first shaft d_1 in [cm], where $2.9 \leq \beta_6 \leq 3.9$
- Diameter of the second shaft d_2 in [cm], where $5.0 \leq \beta_7 \leq 5.5$

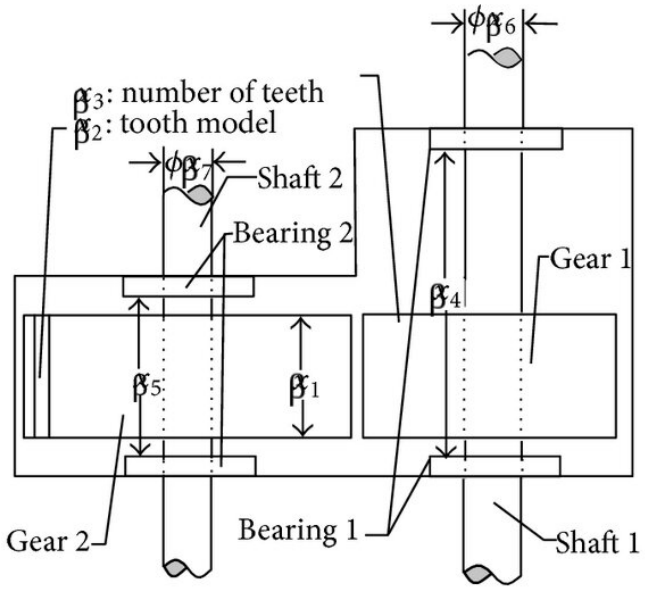


Figure 5.13: Speed reducer design

The volume of the speed reducer in $[cm^3]$ can be calculated using the following equation [48], which multiplied by the density of the material yields the weight of the speed reducer:

$$\begin{aligned} W_{spr} = \; & c_1 b m^2 \left(c_2 N_{teeth}^2 + c_3 N_{teeth} - c_4\right) - c_5 \left(d_1^{\,2} + d_2^{\,2}\right) \\ & c_6 \left(d_1^{\,3} + d_2^{\,3}\right) + c_1 \left(L_1 d_1^{\,2} + L_2 d_2^{\,2}\right) \end{aligned} \tag{5.14}$$

where the occurring parameters are calculated by Golinski [49]:

- $c_1 = 0.7854$
- $c_2 = 3.3333$
- $c_3 = 14.9334$
- $c_4 = 43.0934$
- $c_5 = 1.508$
- $c_6 = 7.4777$

and thus the previous equation can be restated as such:

$$\begin{aligned} W_{spr} = \; & 0.7854 b m^2 \left(3.3333 N_{teeth}^2 + 14.9334 N_{teeth} - 43.0934\right) - 1.508 \left(d_1^{\,2} + d_2^{\,2}\right) \\ & 7.4777 \left(d_1^{\,3} + d_2^{\,3}\right) + 0.7854 \left(L_1 d_1^{\,2} + L_2 d_2^{\,2}\right) \end{aligned} \tag{5.15}$$

The volume function is optimized in $\mathbb{R}^7$ space formed by the design variables $\{b, m, N_{teeth}, L_1, L_2, d_1, d_2\}$. The design space is bound by the imposed constraints that are associated with limitations on the bending stress of gear teeth, surface stresses, transverse deflections of the shafts due to transmitted force and stresses in shafts. Subsequently the imposed constraints are presented analytically. The upper bound on the bending stress of a gear tooth is given by the following formula:

$$\sigma_g = \frac{2M_g}{Y b m^2 N_{teeth}} \leq \sigma_{g_{max}} \Rightarrow \frac{27}{b m^2 N_{teeth}} \leq 1 \tag{5.16}$$

where $Y = 0.3937$ is the Lewis tooth form factor, M_g the bending moment for the gear teeth and $\sigma_{g_{max}} = 900 \;\; g/cm^2$ is the maximum bending stress of the gear teeth. Similarly the upper bound of the compressive stress of a gear tooth for both gears is defined as such:

$$P_{g_{1,2}} = \frac{2BM_g}{b m^2 N_{teeth}^2} \leq P_{g_{max1,2}} \Rightarrow \frac{397.5}{b m^2 N_{teeth}^2} \leq 1 \tag{5.17}$$

where $P_{g_{max1,2}} = 5800 \;\; g/cm^2$ is the maximum surface compressive stress for both gears and B is a coefficient dependent on Young's modulus of elasticity E.

The transverse deflections of the shafts due to the transmitted load P are required to not exceed the following bounds:

$$\text{Shaft 1:} \quad \delta_1 = \frac{1}{48}\frac{PL_1^2}{EI_1} \leq \delta_{1max} \Rightarrow \frac{1.93L_1^3}{mN_{teeth}d_1^4} \leq 1 \tag{5.18}$$

$$\text{Shaft 2:} \quad \delta_2 = \frac{1}{48}\frac{PL_2^2}{EI_2} \leq \delta_{2max} \Rightarrow \frac{1.93L_2^3}{mN_{teeth}d_2^4} \leq 1 \tag{5.19}$$

where I is the moment of inertia of the shafts and $\delta_{1max}, \delta_{2max}$ the maximum permissible transverse deflections of shaft 1 and 2 respectively.
Subsequently, the bending stress conditions for the shafts are limited based on the following formulas:

$$\text{Shaft 1:} \quad \sigma_{g_1} = \frac{M_{z_1}}{W_{x_1}} \leq \sigma_{g_{1max}} \Rightarrow \frac{\sqrt{\left(\frac{745L_1}{mN_{teeth}}\right)^2 + 16.9 \times 10^6}}{0.1d_1^3} \leq 1100 \tag{5.20}$$

$$\text{Shaft 2:} \quad \sigma_{g_2} = \frac{M_{z_2}}{W_{x_2}} \leq \sigma_{g_{2max}} \Rightarrow \frac{\sqrt{\left(\frac{745L_2}{mN_{teeth}}\right)^2 + 157.5 \times 10^6}}{0.1d_2^3} \leq 850 \tag{5.21}$$

where $\sigma_{g_{1max}} = 1100\ g/cm^2$, $\sigma_{g_{2max}} = 850\ g/cm^2$ are the maximum permissible bending stresses for shaft 1 and 2, respectively. W_x is strength section modulus if each shaft and M_z is moment of each shaft formulated by the equation:

$$M_z = \sqrt{M_g^2 + 0.75M_s^2}$$

where M_s is the torsional moment of each shaft.
In order to improve the optimization process, various dimensional restrictions are applied based on experience:

$$i)\ \frac{mN_{teeth}}{40} \leq 1 \qquad ii)\ \frac{5m}{b} \leq 1 \qquad iii)\ \frac{b}{12m} \leq 1 \tag{5.22}$$

Similarly a pair of restrictions are applied on the dimensions of shafts based on previous experience:

$$\begin{aligned} 1.5d_1 + 1.9 &\leq L_1 \\ 1.1d_2 + 1.9 &\leq L_2 \end{aligned} \tag{5.23}$$

The final optimization case is formulated as such:

$$
\begin{aligned}
min\ \ f(\vec{\beta}) = {} & 0.7854bm^2\left(3.3333N_{teeth}^2 + 14.9334N_{teeth} - 43.0934\right) - 1.508\left(d_1^{\,2} + d_2^{\,2}\right) \\
& 7.4777\left(d_1^{\,3} + d_2^{\,3}\right) + 0.7854\left(L_1 d_1^{\,2} + L_2 d_2^{\,2}\right) \\
\text{subject to}\ \ & c_1(\vec{\beta}) = \frac{27}{\beta_1 \beta_2^{\,2} \beta_3} - 1 \leq 0 \\
& c_2(\vec{\beta}) = \frac{397.5}{\beta_1 \beta_2^{\,2} \beta_3^{\,2}} - 1 \leq 0 \\
& c_3(\vec{\beta}) = \frac{1.93\beta_4^{\,3}}{\beta_2 \beta_3 \beta_6^{\,4}} - 1 \leq 0 \\
& c_4(\vec{\beta}) = \frac{1.93\beta_5^{\,3}}{\beta_2 \beta_3 d_7^{\,4}} - 1 \leq 0 \\
& c_5(\vec{\beta}) = \frac{\sqrt{\left(\frac{745\beta_4}{\beta_2 \beta_3}\right)^2 + 16.9 \times 10^6}}{110\beta_6^{\,3}} - 1 \leq 0 \\
& c_6(\vec{\beta}) = \frac{\sqrt{\left(\frac{745\beta_5}{\beta_2 \beta_3}\right)^2 + 16.9 \times 10^6}}{85\beta_7^{\,3}} - 1 \leq 0 \\
& c_7(\vec{\beta}) = \frac{\beta_2 \beta_3}{40} - 1 \leq 0 \\
& c_8(\vec{\beta}) = \frac{5\beta_2}{\beta_1} - 1 \leq 0 \\
& c_9(\vec{\beta}) = \frac{\beta_1}{12\beta_2} - 1 \leq 0 \\
& c_{10}(\vec{\beta}) = \frac{1.5\beta_6 + 1.9}{\beta_4} - 1 \leq 0 \\
& c_{11}(\vec{\beta}) = \frac{1.5\beta_7 + 1.9}{\beta_5} - 1 \leq 0
\end{aligned}
\tag{5.24}
$$

where the bounds of each design variable are $2.6 \leq \beta_1 \leq 3.6$, $0.7 \leq \beta_2 \leq 0.8$, $17 \leq \beta_3 \leq 28$, $7.3 \leq \beta_4 \leq 8.3$, $7.3 \leq \beta_5 \leq 8.3$, $2.9 \leq \beta_6 \leq 3.9$, and $5.0 \leq \beta_7 \leq 5.5$

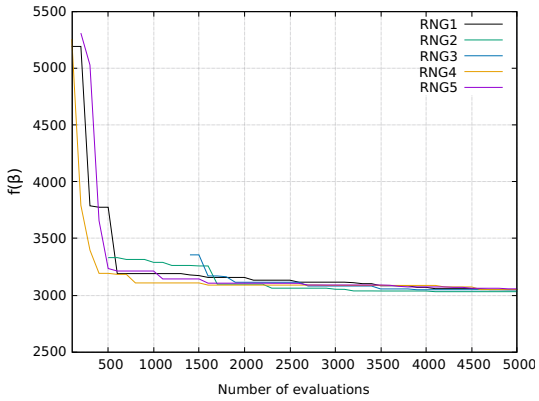
- **Optimization using EAs**

The optimization of the speed reducer design is performed via the use of plain EAs that utilize the PSM. Multiple experiments concluded that the optimal number of offspring and parent population is $(\mu, \lambda) = (30, 100)$ where 5 parents are combined to create a single offspring with two-point crossover. Gray binary encoding is used where 12 bits are assigned to each design variable, except for β_2, β_3 and β_7 that are assigned 8, 8 and 11 bits respectively. The optimization process terminates after 52000 total PSM evaluations have been performed and is repeated for 5 randomly initialised offspring populations P_λ^0 via the use of a RNG. The results are presented in in table 5.10 and figure 5.14.

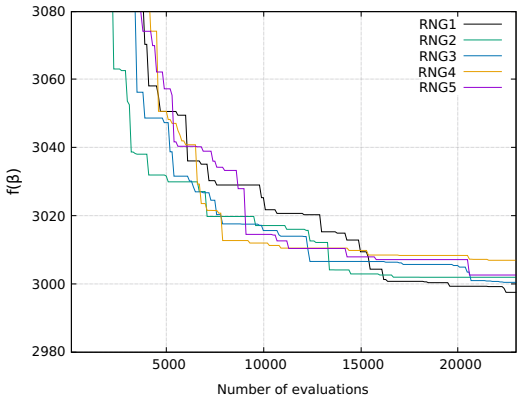
Speed reducer design						
	(μ, λ) **population**	**Best**	**Worst**	**Average**	**Exact model eval.**	**Average exact eval.**
EAs	(30, 100)	2994.91	3001.97	2997.74	52207	40246

Table 5.10: Optimization of speed reducer design using EAs

(a) Comparison between the convergence histories of 5 different P_λ^0 initializations



(b) Comparison between the convergence histories if $f(\vec{\beta})$ range is narrowed



(c) Convergence history of the optimal run

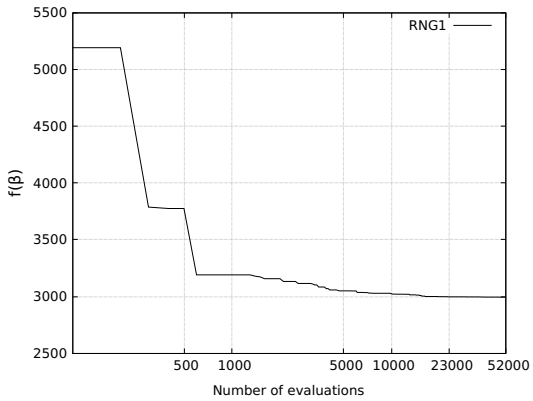


Figure 5.14: Convergence history of speed reducer optimization case using EAs

In 3 of the runs depicted in the previous figures, all evaluated individuals violate the imposed constraints in the first few generations and therefore their corresponding objective function values are penalised and do not appear in the range shown here.

- **Optimization using MAEAs with metamodels trained off-line**

In MAEA-based optimization using off-line trained metamodels, both the objectives $\mathbf{F}(\vec{\beta})$ and the imposed constraints $\mathbf{C}(\vec{\beta})$ are approximated using surrogate models. Specifically, a global metamodel is built on the single objective and n_c unique metamodels on each imposed constraint. Alternatively, a single surrogate model can be trained to approximate the entirety of the constraints with the same efficacy. In this case, the objective function is approximated by a KPLS model, while each constraint is approximated via the use of RBFs; responsible for the construction of the aforementioned metamodels is SMT software. In addition, the optimization of the speed reducer requires the formation of a mixed-integer design, since the number of teeth of the gear n_{teeth} assumes strictly integer values. The outcome of the optimization through 5 runs is presented in table 5.11, where 20000 evaluations per optimization cycle are performed utilizing the trained metamodel.

Speed reducer design						
	(μ, λ) **population**	**Best**	**Worst**	**Average**	**Average metamodel eval./cycle**	**Avg. cycles**
MAEAs, off-line	(30, 100)	3000.95	3017.68	3006.01	20000	1

Table 5.11: Optimization of speed reducer design using MAEAs with off-line training

The best candidate solution $\vec{\beta} = [3.502, 0.7, 17, 7.578, 7.779, 3.357, 5.287]$ obtained via MAEAs with off-line training, produces the constraint and objective function values shown in table 5.12 when evaluated on the PSM.

Speed reducer design			
	MAEAs, off-line	**PSM**	**Relative Error**
$\mathbf{c_1}(\vec{\boldsymbol{\beta}})$	-0.077830	-0.074335	0.045478
$\mathbf{c_2}(\vec{\boldsymbol{\beta}})$	-0.202314	-0.198362	0.019438
$\mathbf{c_3}(\vec{\boldsymbol{\beta}})$	-0.442547	-0.444165	0.003865
$\mathbf{c_4}(\vec{\boldsymbol{\beta}})$	-0.902293	-0.902275	0.000004
$\mathbf{c_5}(\vec{\boldsymbol{\beta}})$	-0.004910	-0.005490	0.120189
$\mathbf{c_6}(\vec{\boldsymbol{\beta}})$	-0.000146	-0.000187	0.204286
$\mathbf{c_7}(\vec{\boldsymbol{\beta}})$	-0.702511	-0.702500	0.000015
$\mathbf{c_8}(\vec{\boldsymbol{\beta}})$	-0.000203	-0.000453	0.645450
$\mathbf{c_9}(\vec{\boldsymbol{\beta}})$	-0.584013	-0.583144	0.001573
$\mathbf{c_{10}}(\vec{\boldsymbol{\beta}})$	-0.084767	-0.084822	0.000209
$\mathbf{c_{11}}(\vec{\boldsymbol{\beta}})$	-0.008184	-0.008183	0.005690
$\mathbf{f}(\vec{\boldsymbol{\beta}})$	3000.95	3000.95	0

*Table 5.12: **C**, **F** responses to optimal $\vec{\beta}$ found via MAEAs with off-line training*

The deviation between metamodel predicted values and evaluated ones, using the exact PSM of either objective function or some constraint, can be observed in figures 5.15, 5.16, 5.17. The surrogate models are trained on $n_t = n_{doe} = 150$ patterns $\mathbf{X}$ collected via the implementation of LHS scheme that makes use of the ESE algorithm to construct an optimal space-filling design. Each individual $\vec{\beta} \in P_\lambda^0$ selected via RNG1 is used to validate the trained model and calculate the NRMSE.

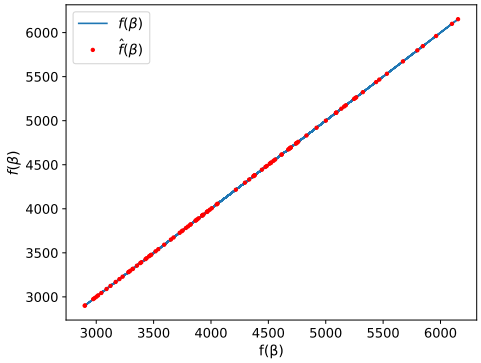
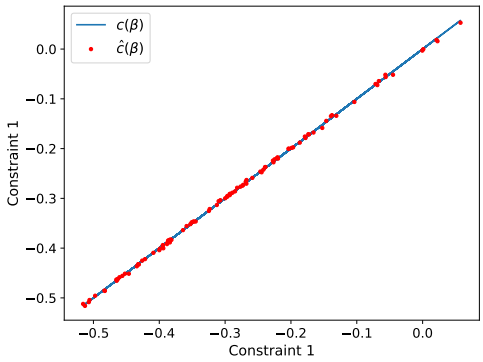
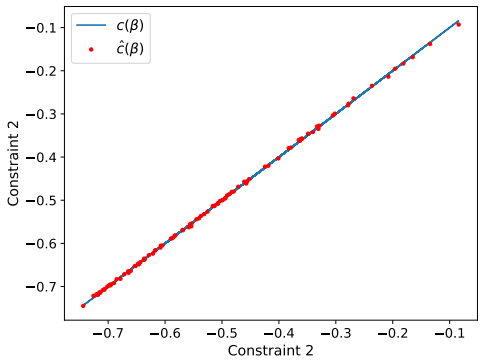


Figure 5.15: Speed reducer case with RNG1. Deviation between exact PSM values of the objective function $\vec{f}(\vec{\beta})$ and KPLS predictions $\hat{\vec{f}}(\vec{\beta})$ with NRMSE = 0.0000847

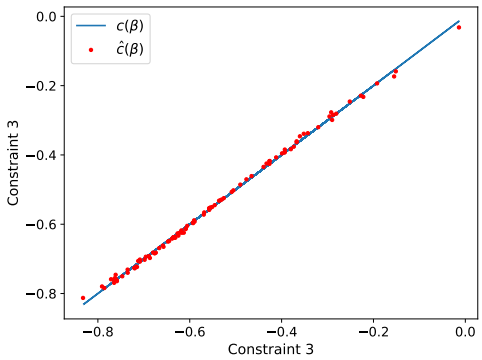
(a) 1st constraint, NRMSE = 0.0082325



*(b) 2nd constraint,*NRMSE = 0.0043814



(c) 3rd constraint, NRMSE = 0.0116700



(d) 4th constraint, NRMSE = 0.0002237

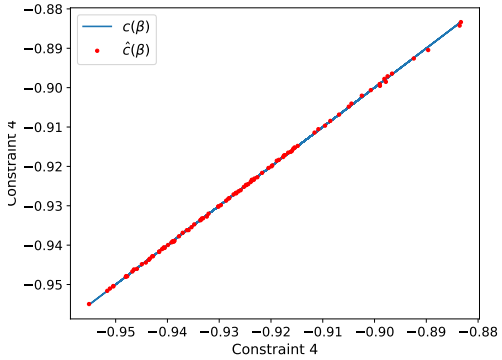
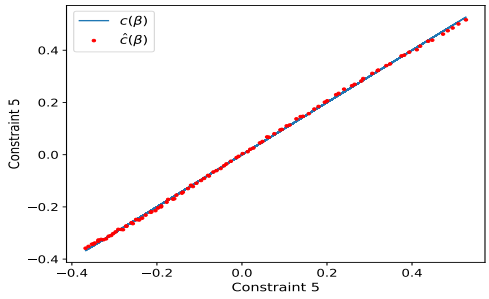
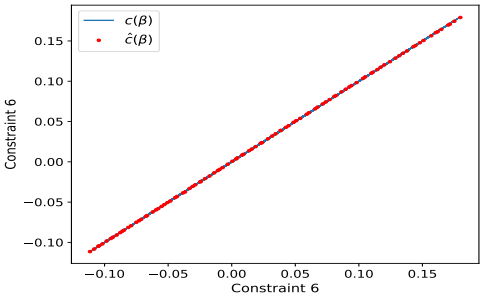


Figure 5.16: Deviation between exact PSM constraint values $\vec{c}(\vec{\beta})$ and RBF predictions $\hat{\vec{c}}(\vec{\beta})$ given via the implementation of SMT. The results refer to the first four constraints in speed reducer case with RNG1.

(a) 5th constraint, NRMSE = 0.02209353

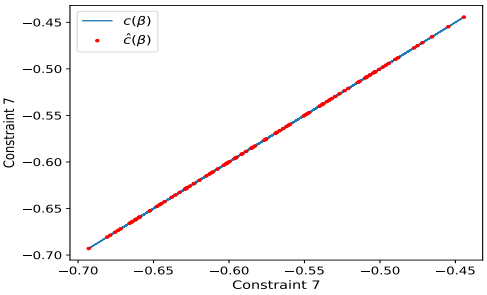


(b) 6th constraint, NRMSE = 0.0031924

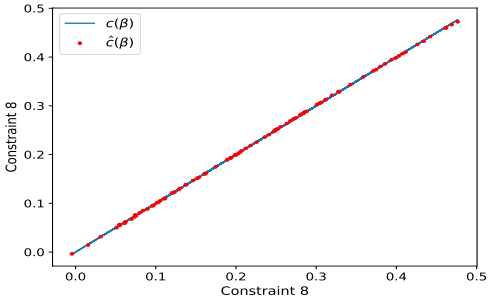


c

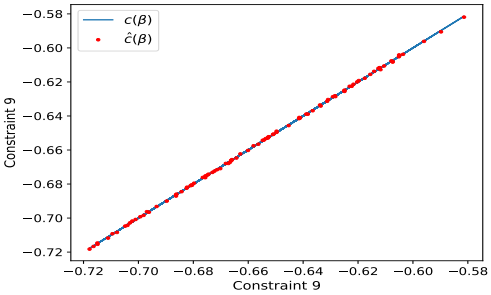
(c) 7th constraint, NRMSE = 0.0000092



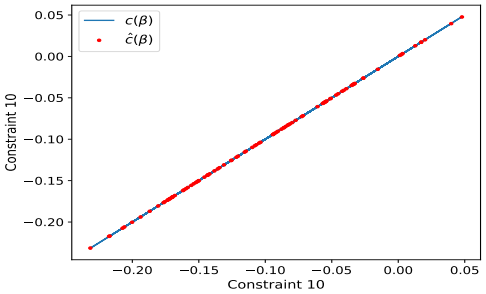
(d) 8th constraint, NRMSE = 0.0042978



(e) 9th constraint, NRMSE = 0.0006251



(f) 10th constraint, NRMSE = 0.0006814



(g) 11th constraint, NRMSE = 0.0011311

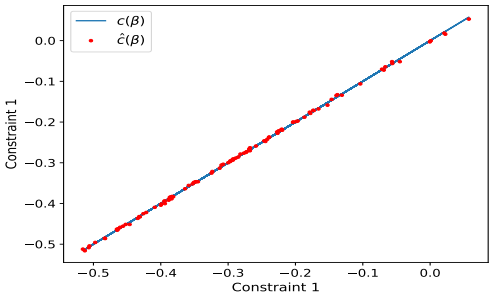


Figure 5.17: Deviation between exact PSM constraint values $\vec{c}(\vec{\beta})$ and RBF predictions $\hat{\vec{c}}(\vec{\beta})$ given via the implementation of SMT. The results refer to the remaining seven constraints in speed reducer case with RNG1.

The NRMSE calculates the mean normalised deviation between the predicted and exactly evaluated on the PSM values on $\lambda = 100$ untried design sites $\mathbf{B}$, where $\vec{\beta} = \vec{\beta}_i = [\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,n_\beta}] \in P_\lambda^0$ is any untried point in the design space contained in the initial offspring population set. NRMSE serves as a metric of model fitting and indicates that all trained metamodels are well-fitted, which explains why the optimization process converges after 1 cycle.

- **Optimization using MAEAs with metamodels trained on-line**

In the MAEA-based optimization of welded beam design using on-line trained metamodels, the LCPE phase is set to initiate once 100 exact PSM evaluations are performed. In LCPE, personalised local metamodels are trained on $15 \leq n_t \leq 30$ training patterns and subsequently $2 \leq \lambda_e \leq 4$ individuals are re-evaluated using the exact evaluation model; a total of 20000 PSM evaluations are performed. Local metamodels are built by utilising either the assisting software SMT or EASY. The former accommodates the use of Kriging, KPLS, KPLSK and RBFs, while the latter relies on Kriging and most often RBFs. In order to identify the most suitable metamodel in SMT for the speed reducer optimization case, a comparison between each respective model is performed w.r.t. the convergence history and the produced outcome; the results of each comparison are presented in table figure 5.18 and table 5.13, respectively.

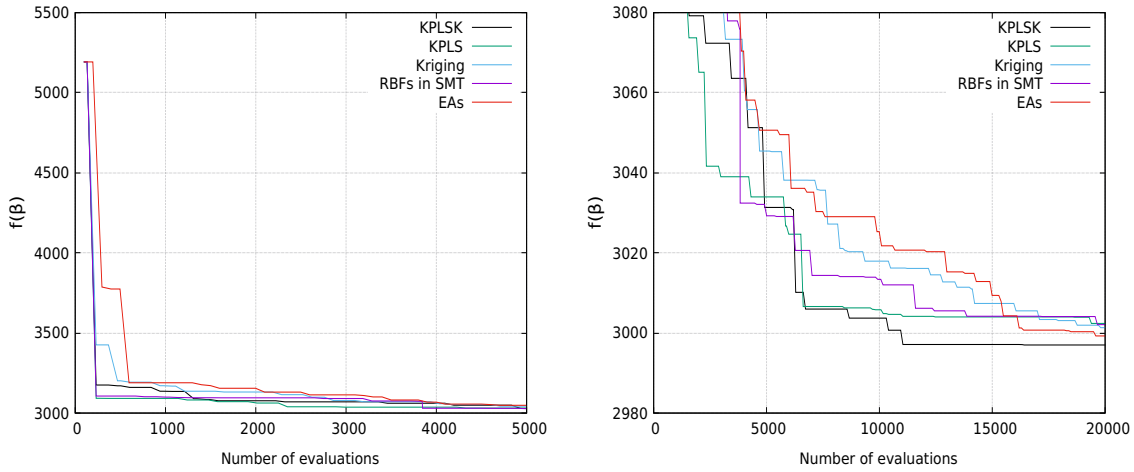


Figure 5.18: Comparison between the convergence histories of speed reducer case using EAs and MAEAs with metamodels trained on-line via SMT

Speed reducer case					
MAEAs, on-line	(μ, λ) **population**	**KPLS**	**KPLSK**	**Kriging**	**RBFs**
SMT	(30, 100)	3002.44	2997.06	3001.38	3002.14

Table 5.13: Speed reducer case with RNG1. Optimal candidate solution found using MAEAs with metamodels trained on-line via SMT

The comparison between convergence histories of the SMT built-in metamodels, depicted in figure 5.18, indicates that KPLSK model is most suitable to facilitate the optimization of the speed reducer case due to its fast convergence to the threshold of both 10000 and 20000 PSM evaluations. In plain EAs, RNG1 yields the best optimization outcome and for that reason MAEAs with on-line training are initialized with the same offspring population P_λ^0.

KPLSK is yet to be compared to EASY built-in RBFs and plain EAs; the results are presented in table 5.14.

Speed reducer case						
MAEAs, on-line	(μ, λ) **population**	**Best**	**Worst**	**Average**	**Average exact eval.**	**Avg. meta-model eval.**
SMT	(30, 100)	2997.06	3005.35	3002.68	20000	22792
EASY	(30, 100)	2998.53	3011.25	3005.46	20000	24775
Plain EAs	(30, 100)	2999.32	3008.35	3004.34	20000	-

Table 5.14: Speed reducer case. Comparison between the outcome of the optimization using MAEAs with on-line training and plain EAs

The design variable vector that minimizes the weight of the speed reducer via the implementation of KLPSK is $\vec{\beta} = [3.501, 0.7, 17, 7.348, 7.769, 3.352, 5.287]$. For MAEAs utilizing EASY built-in RBFs the respective optimal design variable vector is $\vec{\beta} = [3.503, 0.7, 17, 7.499, 7.750, 3.352, 5.287]$.

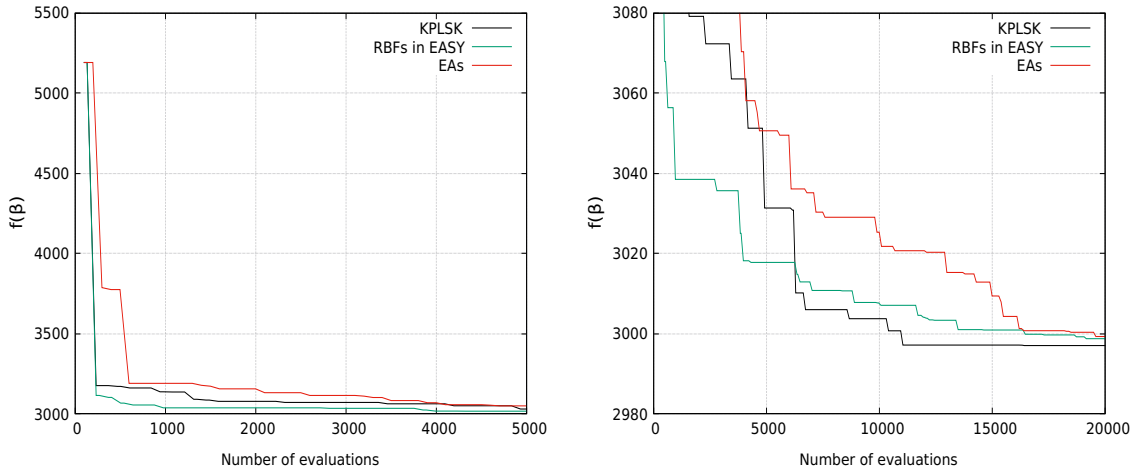


Figure 5.19: Speed reducer case with RNG1. Comparison between the convergence of the optimization using MAEAs with on-line training and plain EAs

In speed reducer optimization case, both MAEA methods outperform EAs and converge to a better optimal solution when compared prior to the threshold of 20000 PSM evaluations, as depicted in figure 5.19. Out of the two on-line trained MAEAs, KPLSK is seemingly the better option in the optimization of the speed reducer case, despite the fast convergence of EASY built-in RBFs for the first 6000 PSM evaluations. The optimization of the speed reducer case using MAEAs with on-line trained KPLSK model resulted on average in a smaller speed reducer weight, in comparison to conventional EAs and MAEAs with EASY built-in, on-line trained RBFs.

5.3 Analysis of the SOO outcome

The results and observations made during the study will subsequently be included in the analysis of the performance of each MAEA method and each utilized metamodel. The performance of each model or method is measured by two conflicting objectives, i.e. model/method efficacy and computational cost. The former can be quantified by observing the convergence of each model, while the latter can be measured via the wall clock time needed to preform each process related to the optimization. The results regarding these two objectives are presented subsequently.

- **Convergence**
  The plot of convergence histories can be used in the comparison of methods that evaluate the evolution individuals on the exact PSM, therefore, MAEAs with off-line training cannot effectively be compared using this method. For this reason, the yielded outcome of each method is compared. In welded beam case, MAEAs with on-line training perform significantly better than the ones with off-line training due to poor fitting of the metamodels approximating the constraint functions. MAEAs with EASY built-in RBFs trained on-line outperform plain EAs and MAEAs using metamodels trained on-line via SMT when compared prior to the threshold of 10000 PSM evaluations.

  In the speed reducer case MAEA-based optimization, off-line trained metamodels well-fitted and yield a similar average outcome compared to plain EAs after 5 runs. MAEAs with on-line training, however, outperform both methods, i.e. EAs and MAEAs with off-line training.

- **Total function calls**
  The total wall clock time depends heavily on the number of exact PSM evaluations performed, so it is necessary to calculate the total amount of PSM evaluations performed by each optimization method. In MAEAs with off-line training, the number of initial sample points selected via DoE techniques is equal to $n_{doe} = 240$ and $n_{doe} = 150$ for the welded beam and the speed reducer design, respectively, while the average number of PSM evaluations is calculated as such:

  $$\overline{n}_{PSM} = n_{doe} + (n_{cycles} - 1)n_{doe}^{'} + n_{elites} \tag{5.25}$$

  In the previous formula, n_{elites} is the total number of elite individuals selected throughout the optimization process, which for SOO problems is equal to the number of cycles performed, since a single elite individual is selected in each cycle. In welded beam optimization case, an average of $\overline{n}_{PSM''} = 240 + 7 \cdot 20 + 8 = 388$ PSM$^{''}$ evaluations were performed, while in speed reducer case the respective PSM evaluations are $\overline{n}_{PSM} = 151$. In comparison, EAs and MAEAs with on-line training used for the minimization of welded beam case perform an average of $\overline{n}_{PSM} = 10000$ evaluations respectively. In speed reducer case, the corresponding exact PSM evaluations performed are $\overline{n}_{PSM} = 20000$. This decrease in exact evaluations, when compared to conventional EAs and MAEAs with on-line training, yields a proportional reduction in computational cost, making MAEAs with off-line training the most cost-efficient optimization method.

In MAEAs with on-line or off-line training, metamodel approximation is used, which contributes in an insignificant increase in the total wall clock time compared to the exact evaluation performed by the PSM. Consequently, the total function calls must account for the number of times the Python or C++ function that is responsible for the metamodel prediction was called; the corresponding number is denoted by $\overline{n}_{meta}$. The total number of function calls, i.e. $\overline{n}_{PSM}$ and $\overline{n}_{meta}$, along with the average outcome of each optimization method, is presented in table 5.15.

Average outcome						
	Welded beam	$\overline{\mathbf{n}}_{\mathbf{PSM}}$	$\overline{\mathbf{n}}_{\mathbf{meta}}$	**Speed reducer**	$\overline{\mathbf{n}}_{\mathbf{PSM}}$	$\overline{\mathbf{n}}_{\mathbf{meta}}$
MAEAs, on-line training via SMT	2.54	10000	11579	3002.68	20000	22792
MAEAs, on-line training via EASY	2.53	10000	14422	3005.46	20000	24775
EAs	2.59	10000	-	3004.34	20000	-
MAEAs, off-line training via SMT	3.12	388	23064	3006.01	151	18239

Table 5.15: Comparison between the average optimization outcome using plain EAs, MAEAs with on-line and off-line training after 5 runs

MAEAs using metamodels trained on-line via SMT yield the best optimization outcome, increase significantly, however, the computational cost of the process. The high computational cost, also observed in MAEAs trained off-line via SMT, is contributed to the implementation of Python in the optimization process. Python is built dynamically, i.e. the data types are determined at run time, on an interpreter, unlike other coding languages that are pre-compiled, e.g. C++ that EASY is based on. Both these attributes, along with the use of several custom functions, prolong the wall clock time needed for each process to be executed. Another factor in the increase of the computational cost is the evaluation of each offspring individually; if the entirety of the population in a generation was being evaluated, then the cost would decrease λ-fold. Consequently, the implementation of SMT and any other Python-based package that utilizes metamodels is not recommended for pseudo-engineering applications with low-order objective functions. MAEAs with on-line trained RBFs found in EASY are based on C++ and their use does not increase the computational cost of the optimization significantly. They additionally converge faster than conventional EAs and MAEAs using SMT metamodels trained either on-line or off-line, and their use is preferred in simple pseudo-engineering cases.

Chapter 6

Airfoil Shape Optimization

The efficacy of the selected metamodels has been tested on a pair of pseudo-engineering optimization problems, where the exact PSM is cheap in terms of computational cost. The study now focuses on performing a comparison between MAEAs, i.e. both the methods and the metamodels will be evaluated, and plain EAs when implemented in the shape optimization of a 2D airfoil. The parameters of the optimization are computed by solving the steady-state Reynolds-Averaged Navier-Stokes (RANS) equations for compressible flows via the use of the one-equation turbulence model Spalart-Allmaras [50]. Consequently in this case, the exact PSM model is a CFD model, which is solved using PUMA software (Parallel solver, for Unstructured grids, for Multi-blade row computations, including Adjoint) [51] developed by the PCOpt/NTUA. The entirety of the CFD evaluations are performed on Nvidia Tesla K40 12 GB GPUs, using a GPU-enabled variant [52] of PUMA programmed in CUDA.

6.1 Mesh and parametrization

The accuracy and the computational cost of CFD evaluations highly depends on the type and quality of the airfoil mesh. There are three types of grids regarding their structure; structured, unstructured and hybrid. Due to the formation of unstructured grids, unstructured solvers are commonly slower then structured ones. However, the constant increase of computational power, along with the high adaptability to any geometry and the fast construction time, lead to the widespread use of unstructured grids in CFD applications. In this diploma thesis, a structured C-type grid is generated for the purpose of the study, as seen in figure 6.1, which is handled by PUMA as a hybrid grid of tetrahedral cells. C-type grids are preferred due to their shape that matches the trailing edge curvature, thus effectively capturing the wake in viscous flows.

The second parameter that affects the accuracy and the computational cost of CFD evaluations is the quality of the mesh. The higher the resolution of the mesh, the greater the accuracy of the outcome. The construction of a high-resolution mesh would, however, severely increase the number of nodes where the RANS need to be solved. In a steady-state compressible flow, turbulence is developed near the walls, i.e. in the viscous sublayer, and in the wake of the airfoil, where a finer mesh is needed in order to account for the small fluctuations in the values of flow components. Consequently, a mesh of ranging resolution is implemented; coarse in the far field and fine near the walls and in the wake of the airfoil.

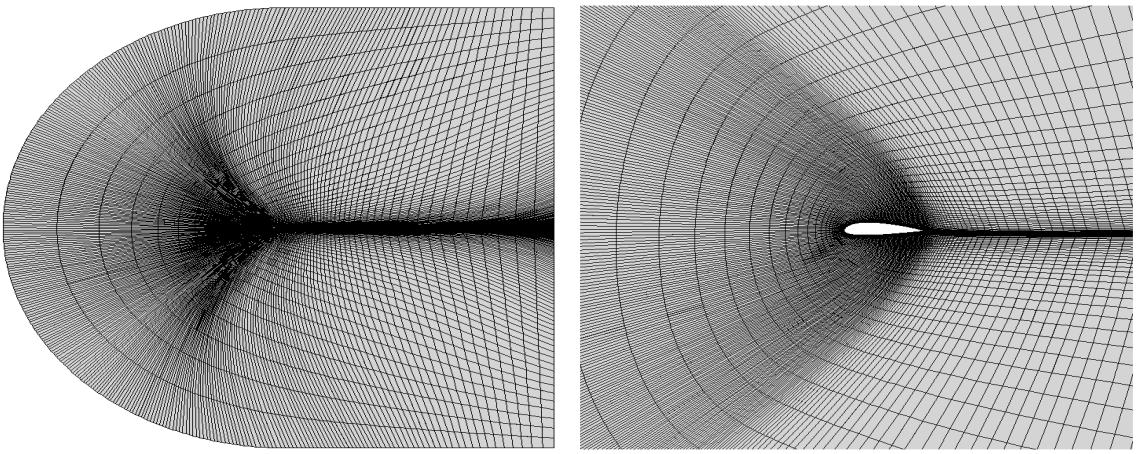


Figure 6.1: 2D C-type structured mesh

flow

The optimization process aims at yielding the airfoil shape that minimizes or maximizes the flow properties, e.g. C_L, C_D, under certain imposed flow conditions and constraints. The modification of the airfoil shape is performed via univariate Non Uniform Rational B-Splines (NURBS). The used NURBS are built via the interpolation of 15 control points in 2D space and produce the curves of the airfoil. Subsequently, the nodes contained in the front patch of the grid are shifted in order to adapt to the new airfoil shape using the spring analogy method, according to which the grid is modelled as net of linear springs with elasticity proportional to the inverse of mesh edge length. The control points of the volumetric NURBS are displaced in the y direction during the optimization and the airfoil shape is modified accordingly. Consequently, the 13 y coordinates of the control points are set as the design variables of the optimization; the y coordinates of the control points corresponding to the leading edge and trailing edge points are kept fixed, as depicted in figure 6.2.

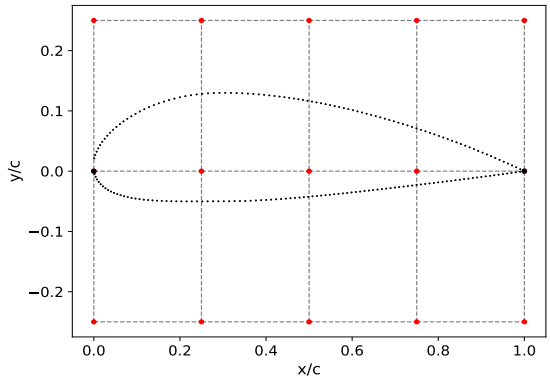


Figure 6.2: Design of NACA 4318. Parametrization of the baseline airfoil geometry using volumetric NURBS with control points that have one (red) or none (black) degree of freedom.

6.2 RANS flow equations

indicates Reynolds averaging. If Favre averaging is applied to all flow quantities, then the continuity, momentum and energy RANS equations can be formulated as such[1]:

$$\begin{aligned}
&\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j) = 0 \\
&\frac{\partial}{\partial t}(\overline{\rho}\tilde{u}_i) + \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_i\tilde{u}_j + \overline{p}\delta_{ij} - \tilde{\tau}_{ij}^{tot}) = 0, \qquad \text{for } i = 1,2 \\
&\frac{\partial}{\partial t}(\overline{\rho}\tilde{E}) + \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j E + \tilde{u}_j\overline{p} - \tilde{q}_j^{tot} - \tilde{u}_i\tilde{\tau}_{ij}^{tot}) = 0
\end{aligned} \tag{6.1}$$

where Reynolds averaging is indicated by the overline. The RANS equations can be also written in conservative form:

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{f}_j^{inv}}{\partial x_j} - \frac{\partial \vec{f}_j^{vis}}{\partial x_j} = 0 \tag{6.2}$$

where in 2D flow $j = 1, 2$ and $\vec{U} = [\overline{\rho}, \overline{\rho}\tilde{u}, \overline{\rho}\tilde{v}, \overline{\rho}\tilde{E}]^T$ is the conservative flow variable vector with components the averaged terms in the continuity, momentum and energy Navier-Stokes equations for compressible flows. The inviscid and viscous fluxes denoted by $\vec{f}_i^{inv}$ and $\vec{f}_j^{vis}$, respectively, and expressed as:

$$\vec{f}_j^{inv} = \begin{bmatrix} \overline{\rho}\tilde{u}_j \\ \overline{\rho}\tilde{u}_1\tilde{u}_j + \overline{p}\delta_{ij} \\ \overline{\rho}\tilde{u}_2\tilde{u}_j + \overline{p}\delta_{ij} \\ \tilde{u}_j(\tilde{E}_t + \overline{p}) \end{bmatrix}, \quad \vec{f}_j^{vis} = \begin{bmatrix} 0 \\ \tilde{\tau}_{1j}^{tot} \\ \tilde{\tau}_{2j}^{tot} \\ \tilde{q}_j^{tot} + \tilde{u}_i\tilde{\tau}_{ij}^{tot} \end{bmatrix} \tag{6.3}$$

where:

$$\begin{aligned}
\tilde{E}_t &= \frac{\overline{p}}{\gamma - 1} + \frac{1}{2}\left(\tilde{u}_i\tilde{u}_i + \frac{\overline{\rho u_i'' u_i''}}{\overline{p}}\right) \\
\tilde{\tau}_{ij}^{tot} &= \frac{\mu + \mu_t}{Re}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) - \frac{1}{3Re}\delta_{ij}\frac{\overline{\rho u_i'' u_i''}}{\overline{p}} \\
\tilde{q}_j^{tot} &= \frac{Cp}{Re}\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial \tilde{T}_s}{\partial x_j}
\end{aligned} \tag{6.4}$$

where μ_t the turbulent or eddy viscosity and Pr, Pr_t the Prandtl and the turbulent Prandtl number, respectively. The stress tensor depends on the molecular or dynamic viscosity of the fluid, denoted by μ, and the Reynolds number Re given by:

$$Re = \frac{\rho u l}{\mu} \tag{6.5}$$

[1]For brevity, the Einstein summation convention for repeated indices is applied in all flow-related equations

where l the characteristic length of the airfoil, which in this case is normalized using the chord length c, so $l \in [0, 1]$. The term q_j that appears in energy equation denotes the j_{th} component of the heat flux and $\tilde{T}_s$ is the static temperature, which for an ideal gas is given by:

$$\tilde{T}_s = \frac{\overline{p}}{\rho R_g} \tag{6.6}$$

where R_g is the specific gas constant, which for air is $R_g = 287\, J/kgK$. The gas specific heat ratio is equal to $\gamma = \dfrac{C_p}{C_v} = 1.4$. The constant parameters C_p, C_v denote the specific heat under constant pressure and constant volume, respectively. The term τ_{ij} that appears in momentum and energy equation, denotes the viscous stress tensor.

The RANS steady-state equations are discretized using finite volume method and intergraded in pseudo-time and can be subsequently solved using a 3rd order Runge-Kutta scheme with residual smoothing, in this case flux Jacobian technique, developed by the Lab of Thermal Turbomachines (LTT), is used to smooth the residuals. The smoothing process requires the implementation of a linear algebra solver and, in this case, Gauss-Seidel method is applied. The process of calculating the RANS residuals is iterative and converges after the residuals have reached a user-defined value or a user-defined number of pseudo-time steps has been reached.

The no-penetration condition is applied, i.e. the normal component of the relative to the wall velocity is set to zero $\vec{\tilde{u}} \cdot \vec{n} = 0$ for stationary walls. The no-slip wall condition is applied in Spalart Allmaras transport equation and $\tilde{\nu}$ is set to zero near the wall. The solid walls of the airfoil are subsequently modelled as adiabatic and the normal component of the relative to the wall heat flux is set to zero $\vec{\tilde{q}}_j^{tot} \cdot \vec{n} = 0$.

6.3 Turbulence model

In order to improve the boundary layer prediction in the presence of adverse pressure gradients various turbulence models are employed. In this thesis, one-equation Spalart-Allmaras turbulence model is implemented, which is based on the observation that on a flat plate in the log-law region of the boundary layer ($y^+ > 30$) the profile of turbulence kinematic viscosity ν_t with y^+ is linear, while in the viscous sublayer ($y^+ < 5$) the profile is quartic. If however we assume a new variable $\tilde{\nu}$, similar to ν_t, with linear profile w.r.t. y^+ in the entirety of the sublayer region, then we can produce more stable results near the solid wall while simultaneously reducing the computational cost that arises from constructing a fine mesh near the solid wall. In order to account for any geometry and possible flow conditions, $\tilde{\nu}$ is calculated by solving the modelled transport equation for compressible flows [55]:

$$\begin{aligned}\frac{\partial}{\partial t}(\rho\tilde{\nu}) + \frac{\partial}{\partial x_j}(\rho\tilde{\nu}u_j) =& \frac{\rho}{\sigma_w Re}\left[\frac{\partial}{\partial x_j}\left((\tilde{\nu}+\nu)\frac{\tilde{\nu}}{\partial x_j}\right) + c_{b2}\frac{\partial\tilde{\nu}}{\partial x_j}\frac{\partial\tilde{\nu}}{\partial x_j}\right] \\ &+ \rho c_{b1}(1-f_{t2})\tilde{S}_h\tilde{\nu} - \frac{\rho}{Re}\left(c_{w_1}f_w - \frac{c_{b_1}}{K^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{d_s}\right)^2\end{aligned} \tag{6.7}$$

where d_s is the distance from the closest solid wall and $\tilde{S}_h$ is the modelled vorticity, which is connected with the vorticity via the following formula:

$$\tilde{S}_h = S_h + \frac{\tilde{\nu}}{(Kd_s)^2}f_{v_2}, \quad f_{v_2} = 1 - \frac{o_s}{1+o_s f_{v_1}} \tag{6.8}$$

where f_{v_1} a function that models damping effects near the solid walls:

$$f_{v_1} = \frac{o_s^3}{o_s^3 + c_{v_1^3}}, \quad o_s = \frac{\tilde{\nu}}{\nu} \tag{6.9}$$

The remaining functions are given by:

$$\begin{aligned} &f_w = g_w\left[\frac{1+c_{w_3}^6}{g_w^6 + c_{w_3}^6}\right]^{1/6}, \quad g_w = r_s + c_{w_2}(r_s^6 + r_s), \quad r_s = min\left(\frac{\tilde{\nu}}{Re\tilde{S}_h K^2 d_s^2}, 10\right) \\ &f_{t_2} = c_{t_3}e^{-c_{t4}o_s^2}, \quad c_{t_3} = 1.2, \quad c_{t_3} = 0.5, \quad K = 0.41, \quad c_{w_2} = 0.3, \quad c_{w_3} = 2.0 \\ &c_{w_1} = \frac{c_{b_1}}{K^2} + \frac{1+c_{b_2}}{\sigma_w}, \quad c_{b_1} = 0.1355, \quad c_{b_2} = 0.622, \quad \sigma_w = 0.6667, \quad c_{v_1} = 7.1 \end{aligned} \tag{6.10}$$

Solving equation 6.7 yields a $\tilde{\nu}$ value, which is then utilized to calculate the turbulent kinematic viscosity via the equation:

$$\nu_t = \tilde{\nu} f_{v_1} \tag{6.11}$$

With ν_t known, the eddy viscosity μ_t can be calculated and used to update the RANS equations.

6.4 Optimization cases

For the purposes of this thesis, the selected metamodels and the respective MAEA methods are tested on the optimization of a NACA 4318 airfoil. The study will focus on optimizing the airfoil shape w.r.t. one or two objectives, namely the aerodynamic lift and drag force. These parameters are commonly contradicting, so each is alternatively used as an objective or an imposed constraint to the optimization problem. The entirety of the CFD evaluations are performed on Nvidia Tesla K40 12 GB GPUs, using a GPU-enabled variant of PUMA running on CUDA language.

6.4.1 MOO optimization at take-off conditions

The first optimization case simulates the conditions governing the take-off stage of an aircraft flight, which are characterized by low free-stream velocity $U = 51\ m/s$, high angle of attack $a = 10^o$ and fluid properties at sea level given in table 6.1. The flow field quantities of the initial baseline geometry at take-off conditions can be seen in figure 6.3.

Fluid properties at sea level h = 0 m			
	ρ [kg/m^3]	**p [bar]**	**T [K]**
Air	1.225	1.01325	288

Table 6.1: Air properties at sea level

(a) Mach field *(b) Pressure field*

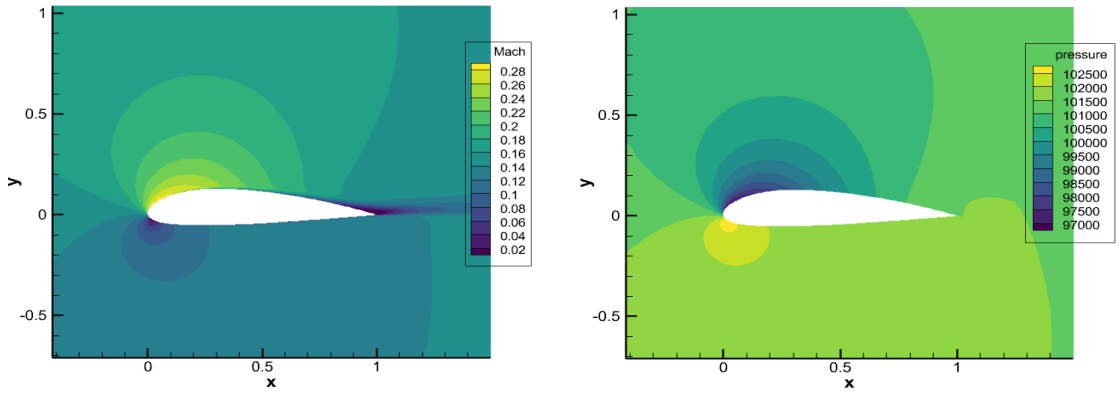


Figure 6.3: Flow field quantities of the baseline airfoil geometry

The airfoil is subsequently optimized w.r.t. two objectives, minimization of the produced drag force and maximization of the produced lift force, while no constraint is imposed. The optimization problem solved is the following:

$$\begin{aligned} max \;\; & f_1(\vec{\beta}) = L \\ min \;\; & f_2(\vec{\beta}) = D \end{aligned} \tag{6.12}$$

where the bounds of the design variables shaping the pressure side of the airfoil are $-0.26 \leq \beta_{1-5} \leq -0.24$, while the design variables shaping the camber line and the suction side are $-0.01 \leq \beta_{6-8} \leq 10.0$ and $0.24 \leq \beta_{9-13} \leq 0.26$, respectively.

The optimization is performed via the use EAs and MAEAs, where $\lambda = 40$ offspring are evaluated in each generation and $\mu = 20$ parents are retained, 3 of which are combined to create a new offspring at the start of every new generation. In the optimization of the airfoil using MAEAs with on-line training, the LCPE phase is initialized after 60 CFD evaluations and uses KPLS and RBFs metamodels that are trained via SMT and EASY, respectively. Both methods terminate after 400 CFD evaluations have been performed and are subsequently compared with to MAEAs using KPLS trained off-line on $n_{doe} = 80$ initial training patterns. The evolution in all methods retains 15 prominent solutions that are stored in the temporary elite set P_e, which at the end of the evolution stores the Pareto frontier of non-dominated solutions. Three Pareto fronts are produced via the use of a RNG seed number that corresponds to a different offspring population P_λ^0 and are presented in figure 6.4; the set P_λ^0 contains the baseline airfoil geometry.

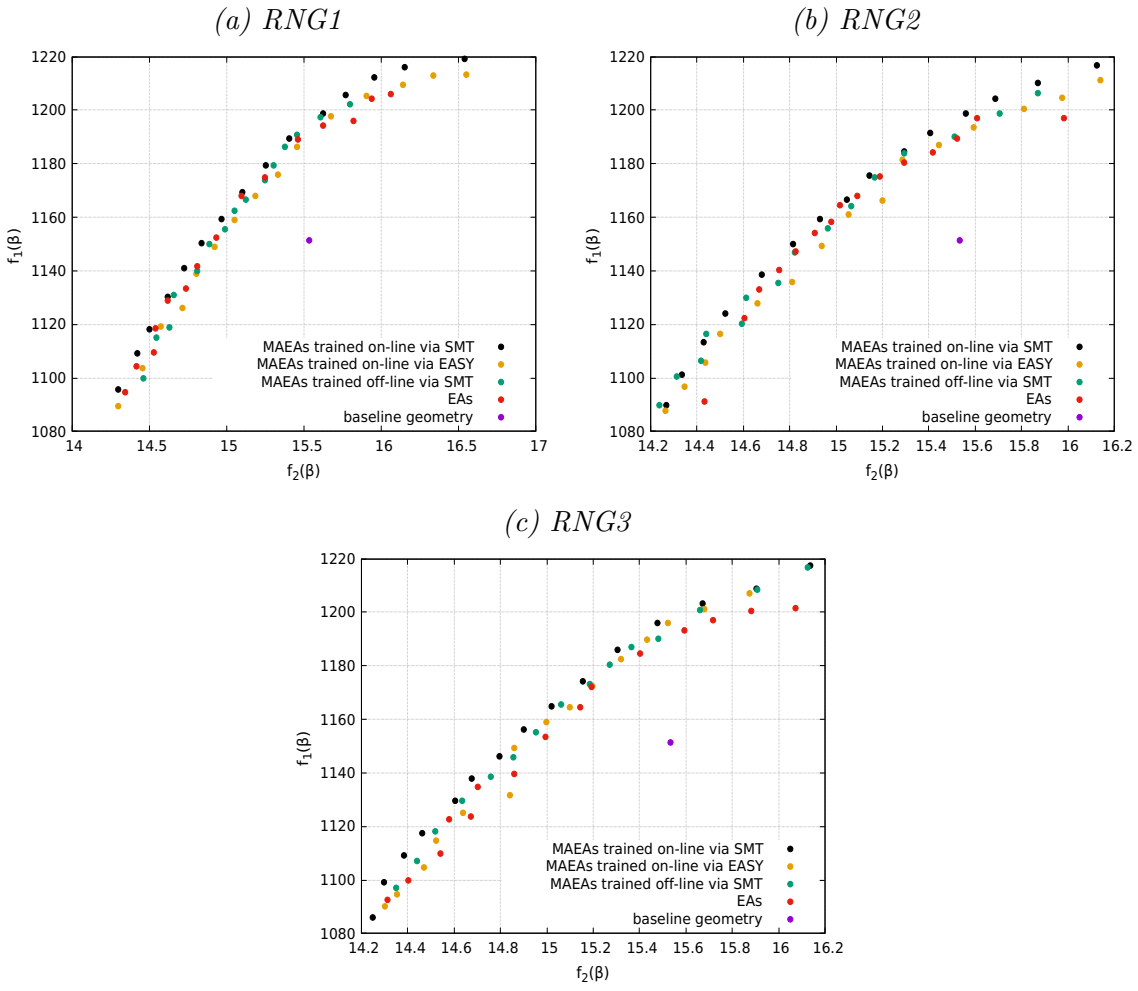


Figure 6.4: NACA 4318 optimization at take-off flow conditions. Comparison of the Pareto fronts of 15 non-dominated solutions computed via the implementation of plain EAs, MAEAs with off-line and on-line trained KPLS and MAEAs with EASY built-in, on-line trained RBFs for 400 CFD evaluations

The optimization initialized with seed number RNG1 is allowed to perform more CFD evaluations and for this reason the corresponding Pareto front contains slightly better non-dominated individuals. In each case, the Pareto fronts formed via the implementation of each optimization method do not reveal which method is dominant and therefore a hypervolume indicator is assigned to each front $\mathcal{F} \subset \mathbb{R}^n$.

In the 2D space, which is the case here, the hypervolume indicator $H(\mathcal{F})$ is equivalent to the area defined by each $\vec{q} \in \mathcal{F}$ and the reference point $\vec{x}_r \in \mathbb{R}^n$ is defined as such:

$$\begin{aligned} x_{r_1} &= \{q_1 \in \mathcal{F}_i : q_1 \geq \xi, \forall \xi \in \mathcal{F}_i, \forall i \in [1, n_f]\} + \xi_1 \\ x_{r_2} &= \{q_2 \in \mathcal{F}_i : q_2 \geq \xi, \forall \xi \in \mathcal{F}_i, \forall i \in [1, n_f]\} + \xi_2 \end{aligned} \tag{6.13}$$

In this case, the parameters assume the values $(\xi_1, \xi_2) = (0.25, 20)$ and the outcome is presented in the table 6.2.

Hypervolume indicator H($\mathcal{F}$)			
	RNG1	**RNG2**	**RNG3**
MAEAs, on-line training via SMT	274.588	222.513	224.786
MAEAs, on-line training via EASY	257.987	206.800	211.937
MAEAs, off-line training via SMT	252.866	213.701	216.461
EAs	257.512	200.114	204.289

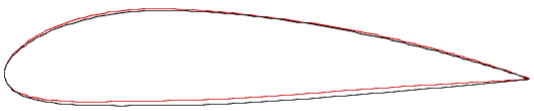
Table 6.2: NACA 4318 optimization at take-off flow conditions. Hypervolume indicator of Pareto fronts formed via the implementation of various optimization methods

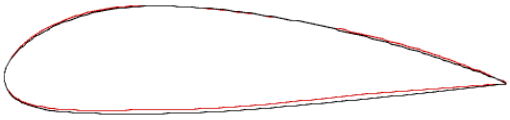
The Pareto front generated by the implementation of MAEAs with metamodels trained on-line via SMT assumes the highest hypervolume indicator value for every RNG. On the contrary, the lowest hypervolume indicator is observed when the optimization is performed via the use plain EAs. MAEAs with off-line training perform better than MAEAs with metamodels trained on-line via the use of EASY for RNG2, RNG3. For those RNG seed numbers, each optimization cycle converges after 1000 evaluations have been performed on the trained metamodel, while for RNG1 only 520 metamodel evaluations are performed. This increase in metamodel evaluations has a cost-efficient impact in the efficacy of the method and is retained in the following optimization cases.

At take-off conditions the main objective is generating an airfoil design that produces the maximum lift force; such designs are compared in figure 6.5.

(a) MAEAs, on-line training via SMT, $L = 1217.45$ *N and* $D = 16.14$ *N*

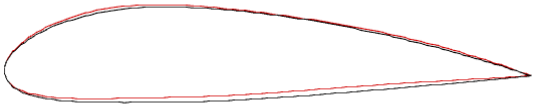
(b) MAEAs, on-line training via EASY, $L = 1207.19$ *N and* $D = 15.87$ *N*





(c) MAEAs, off-line training, $L = 1216.79$ *N and* $D = 16.12$ *N*

(d) EAs, $L = 1201.47$ *N and* $D = 16.07$ *N*



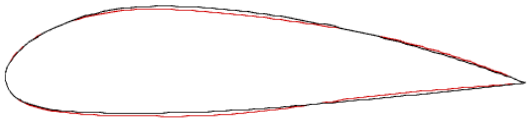


Figure 6.5: NACA 4318 optimization at take-off conditions with RNG3. Comparison between airfoil desings that yield the highest lift force in the Pareto front (red) compared to the baseline design (black).

The use of MAEAs with metamodels trained on-line via SMT result in both the optimal Pareto front and the best optimal solution. In the relative ranking of each method, MAEAs using metamodels trained off-line via SMT finish second when considering their reduced computational cost.

6.4.2 SOO optimization at take-off conditions

The second optimization case focuses on the maximization of the produced lift force when the airfoil operates at take-off conditions. In this case, however, the maximization of the lift force is the only objective of the optimization and the design space solutions is bounded by a user-imposed demand of a less than 8% increase in drag force produced compared to the initial baseline geometry, where $D_{bsl} = 15.53$ N. The constrained SOO can be expressed as such:

$$\begin{aligned} max \;\; & f(\vec{\beta}) = L \\ \text{subject to } & c_1(\vec{\beta}) = D - 1.08 D_{bsl} \leq 0 \end{aligned} \tag{6.14}$$

with bounds of the design variables identical to the ones used in the MOO case.

The optimization is performed via the use EAs and MAEAs, where $\lambda = 40$ offspring are evaluated in each generation and $\mu = 20$ parents are retained, 3 of which are combined to create a new offspring at the start of every new generation. In the optimization of the airfoil using MAEAs with on-line training, the LCPE phase is initialized after 40 CFD evaluations. Both optimization methods terminate after 400 CFD evaluations have been performed. Based on the outcome of the MOO optimization case, the initial offspring population P_λ^0 is produced via the use of a RNG2 and RNG3 seed; the set P_λ^0 contains the baseline airfoil geometry. The convergence history of RNG3 optimization using plain EAs and MAEAs with on-line training is presented in figure 6.6.

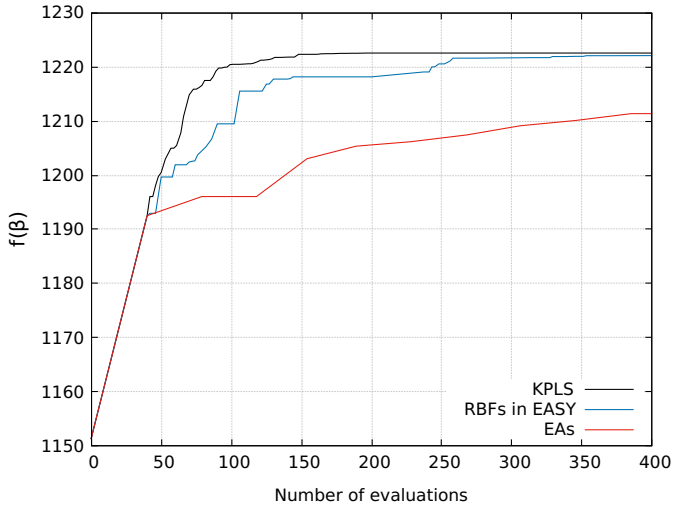


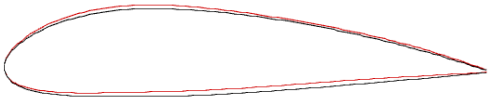
Figure 6.6: Maximizing NACA 4318 lift force at take-off conditions using RNG3. Comparison between the convergence histories of plain EAs and MAEAs with meta-models trained on-line via SMT and EASY

Both MAEA methods outperform conventional EAs in the number of CFD evaluations performed. The optimization facilitated by MAEAs with surrogate models trained on-line via SMT, however, has seemingly the best convergence speed, since it converges to the optimal solution after performing circa 150 CFD evaluations.

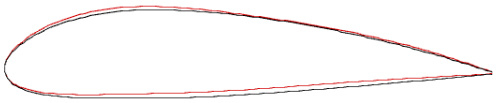
Both methods are subsequently compared with to MAEAs using KPLS trained off-line on $n_{doe}=80$ initial training patterns, in order to determine which method is more efficient. The optimization processin MAEAs with off-line training converges after 1000 evaluations per cycle have been perfomed on the surrogate model, in this case KPLS. At the end of each cycle $n_{new\_doe} = 5$ random training patterns are sampled.

(a) MAEAs, on-line training via SMT, $L=1222.52$ N and $D=16.54$ N

(b) MAEAs, on-line training via EASY, $L=1222.06$ N and $D=16.44$ N



(c) MAEAs, off-line training via SMT, $L=1221.02$ N and $D=16.29$ N



(d) EAs, $L=1211.35$ N and $D=16.25$ N

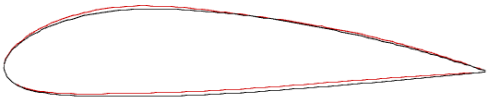


Figure 6.7: NACA 4318 optimization at take-off conditions with RNG3. Comparison between optimal airfoil desings.

All optimized designs in figure 6.7 result in a positive displacement of the camber line. In the suction side, the increase in curvature near the leading edge of the airfoil results in an increase of the favourable pressure gradient $dp/dx < 0$. The adverse pressure gradient $dp/dx > 0$ in the suction side, which is responsible for the turbulence generation, remains relatively unchanged to prevent the flow separation in the boundary layer. In the pressure side, an increase in the pressure gradient is desired and is achieved via an increase in the airfoil curvature. The pressure field of the optimized airfoil is presented next, where the optimized airfoil shows a 6.1% increase in lift force L and a 4.5% increase in drag force D, as shown in figure 6.8.

(a) Baseline

(b) Optimized via MAEAs with metamodels trained off-line

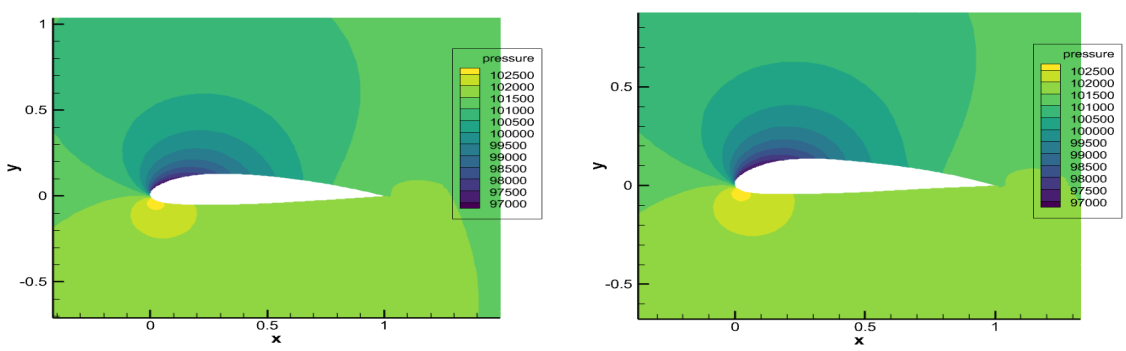


Figure 6.8: Comparison between baseline and optimized airfoil

In the current airfoil shape optimization cases, the PSM is the CFD solver. A single CFD evaluation, which is combined with some subprocess related to the adaptation of the mesh around the new airfoil design, is far more costly than any PSM evaluation performed in previous pseudo-engineering optimization cases and metamodel predictions. For this reason, $\overline{n}_{PSM}$ alone significantly increases the computational cost. For the sake of completeness, however, the total functions calls, i.e. $\overline{n}_{PSM}$ and $\overline{n}_{meta}$, and the average outcome produced via the implementation of each method are presented in table 6.3, in order to ensure that the best possible optimization method is selected.

NACA 4318 optimization at take-off conditions			
	Average $\vec{f}$	**$\overline{n}_{PSM}$**	**$\overline{n}_{meta}$**
MAEAs, on-line training via SMT	1222.33	400	2988
MAEAs, on-line training via EASY	1221.90	400	4000
MAEAs, off-line training via SMT	1221.27	81	1000
EAs	1211.54	400	-

Table 6.3: Comparison between all implemented optimization methods

The total wall clock time of the optimization is significantly decreased when MAEAs with off-line training are implemented and their corresponding outcome is similar to the one obtained via the implementation of MAEAs with on-line training. Consequently, the lift force L of the NACA 4318 at take-off conditions is maximized when MAEAs with off-line training are implemented, followed closely by MAEAs with surrogate models trained on-line via SMT that yield an optimal solution after circa 150 CFD evaluations have been performed.

6.4.3 SOO optimization at cruise conditions

The last optimization case simulates the conditions governing the cruise stage of an aircraft flight, which are characterized by high free-stream velocity $U = 206.64\ m/s$, low angle of attack $a = 2^o$ and fluid properties at sea level given in table 6.4.

Fluid properties at h = 11000 m			
	$\boldsymbol{\rho}$ **[kg/m^3]**	**p [bar]**	**T [K]**
Air	0.364805	0.227	216.8

Table 6.4: Air properties at cruise height

In this case, the minimization of the produced drag force is the only objective of the optimization and the design space is bounded by a user-imposed demand of a less than 8% decrease in lift force compared to the initial baseline geometry, where $L_{bsl} = 1123.81$ N. The constrained SOO can be expressed as such:

$$\begin{aligned} &min \;\; f(\vec{\beta}) = D \\ &\text{subject to} \;\; c_1(\vec{\beta}) = L - 0.92 L_{bsl} \geq 0 \end{aligned} \tag{6.15}$$

with bounds of the design variables identical to the ones used in the MOO case.

The optimization is performed via the use EAs and MAEAs, where $\lambda = 40$ offspring are evaluated in each generation and $\mu = 20$ parents are retained, 3 of which are combined to create a new offspring at the start of every new generation. In the MAEA-based optimization of the airfoil using on-line training, the LCPE phase is initialized after 40 CFD evaluations. Both optimization methods terminate after 400 CFD evaluations have been performed. The initial population set P_λ^0 is produced via the use of a RNG2 and RNG3 seed; the set P_λ^0 contains the baseline airfoil geometry. The convergence history of RNG3 optimization using plain EAs and MAEAs with on-line training is presented in figure 6.9.

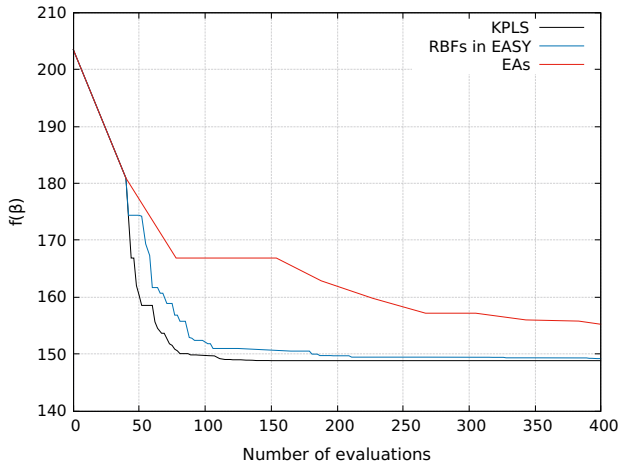


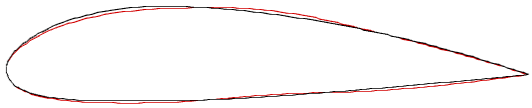
Figure 6.9: Minimizing NACA 4318 drag force at cruise conditions with RNG3. Comparison between the convergence histories of plain EAs and MAEAs with meta-models trained on-line via SMT and EASY

Both MAEA methods outperform conventional EAs in the number of CFD evaluations performed. The optimization facilitated by MAEAs with surrogate models trained on-line via SMT, however, has seemingly the best convergence speed, since it converges to the optimal solution after performing circa 150 CFD evaluations.

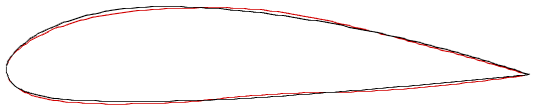
Both methods subsequently compared with to MAEAs using KPLS trained off-line on $n_{doe} = 80$ initial training patterns, in order to determine which method is more efficient. The optimization process in MAEAs with off-line training converges after 1000 evaluations have been perfomed per cycle on the surrogate model, in this case KPLS. At the end of each cycle $n_{new\_doe} = 5$ random training patterns are sampled.

(a) MAEAs, on-line training via SMT, $D = 148.87$ *N and* $D = 1373.12$ *N*

(b) MAEAs, on-line training via EASY, $D = 149.24$ *N and* $D = 1383.62$ *N*



(c) MAEAs, off-line training via SMT, $D = 150.06$ *N and* $L = 1367.56$ *N*



(d) EAs, $D = 155.25$ *N and* $L = 1429.83$*N*

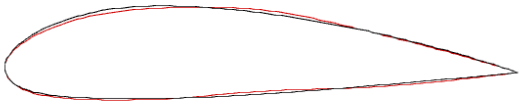


Figure 6.10: NACA 4318 take-off conditions with RNG3. Comparison between airfoil desings that resulted after the implementation of each optimization methods.

The streamline flow of $Mach = 0.7$ is accelerated in the suction side and becomes supersonic, reaching its peak $Mach = 1.3$ for the baseline airfoil as seen in figure 6.11. As a result, a shock wave is formed that interacts with the boundary layer and leads to flow separation. The aim of the optimization is to delay the formation of the shock wave and by extension the flow separation. As depicted in figure 6.10, this can be achieved by decreasing the curvature near the leading edge of the airfoil, and therefore the favourable pressure gradient $dp/dx < 0$. On the other hand, the adverse pressure gradient $dp/dx > 0$ is increased by an increase of curvature. The pressure side of the optimized airfoil forms a slightly concave surface that is formed due to the negative displacement of the camber line and results in high pressure region. Flow separation is mostly responsible for the induced drag force and thus the optimized airfoil designs reduce the flow separation region while simultaneously increasing the pressure coefficient around the airfoil and by extension the produced lift force L. In the baseline geometry, the flow separation initiates at $x/c = 0.33$, while in the optimized designs the flow separation initiates at $x/c = 0.43$ of the normalized characteristic length. As a result the optimized designs show a 26.3% decrease in drag force D, combined with a 21.8% increase in lift force L.

(a) Baseline Mach field

(b) Optimized Mach field via MAEAs with metamodels trained off-line

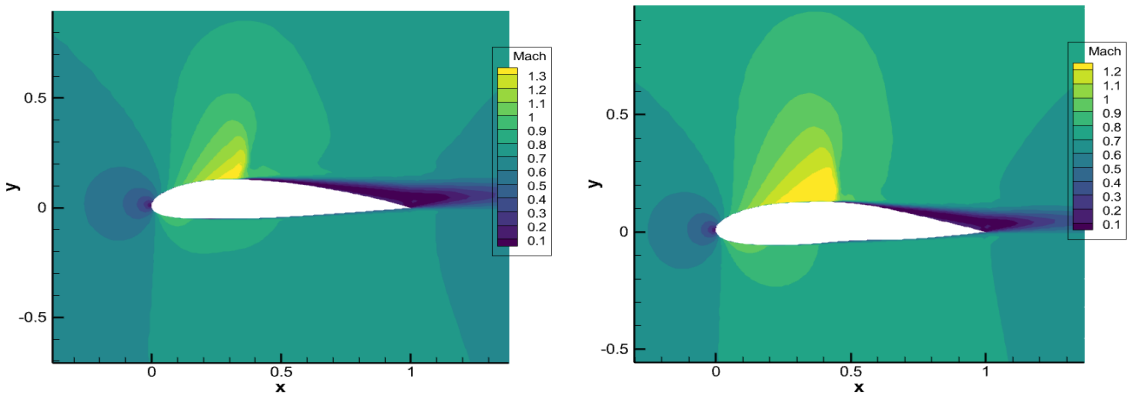


Figure 6.11: Comparison between baseline and optimized airfoil

In the current airfoil shape optimization cases, the PSM is the CFD solver. A single CFD evaluation, which is combined with some subprocess related to the adaptation of the mesh around the new airfoil design, is far more costly than any PSM evaluation performed in previous pseudo-engineering optimization cases and metamodel predictions. For this reason, $\overline{n}_{PSM}$ alone significantly increases the computational cost. For the sake of completeness, however, the total functions calls, i.e. $\overline{n}_{PSM}$ and $\overline{n}_{meta}$, and the average outcome produced via the implementation of each method are presented in table 6.5, in order to ensure that the best possible optimization method is selected.

NACA 4318 optimization at cruise conditions			
	Average $\vec{f}$	**$\overline{n}_{PSM}$**	**$\overline{n}_{meta}$**
MAEAs, on-line training via SMT	148.86	400	2988
MAEAs, on-line training via EASY	149.57	400	4000
MAEAs, off-line training via SMT	149.60	87	2000
EAs	154.63	400	-

Table 6.5: Comparison between all implemented optimization methods

The total wall clock time of the optimization is significantly decreased when MAEAs with off-line training are implemented and their corresponding outcome is similar to the one obtained via the implementation of MAEAs with on-line training. Consequently, the induced drag force D of the NACA 4318 at cruise conditions is minimized when MAEAs with off-line training are implemented, followed closely by MAEAs with surrogate models trained on-line via SMT that yield an optimal solution after circa 150 CFD evaluations have been performed.

Chapter 7

Conclusions and Future Work

7.1 Overview

In this diploma thesis, the implementation of Metamodel-Assisted EAs (MAEAs) in the optimization process of various common engineering cases is tested. MAEAs are introduced due to the increased computational cost observed in CFD applications optimized via the use of conventional EAs. In the entirety of methods utilized, the optimization process is accommodated by EASY. Two main MAEA-based optimization methods are used in this thesis and are based on the approach followed in the training of the metamodels, i.e. on-line and off-line training.

In the former, a global surrogate model is built prior to the evolution on $n_{doe}^{'}$ training patterns $\mathbf{X}$ that are collected via the implementation of a DoE scheme. The most commonly used DoE techniques and the ones used in this thesis are accommodated by a Python package, called PyDOE, and result in the construction of a random, a LH or a factorial design. According to the construction method implemented, LHDs can be further categorized in centered, maximin, maximin centered, maxent or ESE LHDs. A comparison between the attributes of LH and factorial designs led to the conclusion that LHS with ESE construction criterion will be the main sampling method in this thesis. In MAEAs with off-line training, responsible for the training of the metamodels and the DoE construction is a Python- based, open-source software, called SMT. In SMT software, a variety of surrogate models can be found; in this thesis, namely Kriging, its applications in reduced design space using PLS, i.e. KPLS and KPLSK, and RBFs are utilized. The structure of MAEA-based optimization with off-line training and the Python codes written needed to implement it, are subsequently presented.

In optimization using MAEAs with on-line training, an LCPE phase is introduced where metamodels are utilized. The structure of MAEA-based optimization with on-line training and the Python codes written needed to implement it using SMT, are subsequently presented. In this method, EASY built-in RBFs can be utilized in order to perform the optimization and this built-in metamodel is therefore compared to the surrogate models provided by SMT.

Both these methods are compared to plain EAs in terms of efficacy and computational cost. Each optimization method is initially implemented in two simple pseudo-engineering optimization problems, i.e. welded beam and speed reducer case. The first optimization case requires the minimization of the fabrication cost of a welded beam design w.r.t. 4 design variables $\vec{\beta}$ and the design space is bound

by 5 imposed constraints $c(\vec{\beta})$. The second optimization problem dictates the minimization of the weight of a speed reducer w.r.t. 7 design variables $\vec{\beta}$ and the design space is bound by 11 imposed constraints $c(\vec{\beta})$

Once the optimization methods have been tested on these simple pseudo-engineering cases, they are implemented in the shape optimization of a 2D NACA 4318 airfoil. The RANS equations of compressible flows are solved using PUMA CFD solver. The flow field is simulated at take-off and cruise conditions, where the former are used in the MOO optimization and constrained maximization of the lift force L of the airfoil and the latter in the constrained minimization of the drag force D.

7.2 Conclusions

By completing the studies in this diploma thesis, the following conclusions are drawn:

- In pseudo-engineering applications of low computational cost the implementation of MAEAs using metamodels trained off-line or on-line via SMT severely prolongs the wall clock time of the optimization. This is contributed to nature of SMT, which is Python-based, and the communication between EASY and SMT. Consequently, in low fidelity models the use of MAEAs with surrogate models trained via SMT are not recommended. However, MAEA-based optimization using EASY built-in RBFs that are trained on-line is a more cost-effective solution and are generally preferred.

- In the shape optimization of the naca 4318, MAEAs using off-line and on-line training outperform conventional EAs and MAEAs using on-line trained, EASY built-in RBFs. In S00 cases in particular, the use of MAEAs with KPLS trained-on line via SMT results in the convergence of the optimization after circa 150 CFD evaluations have been performed. Consequently, MAEA with off-line and on-line training yield a 80% and 60% decrease in the computational cost of CFD applications, respectively. For this reason, MAEAs with surrogate models trained via SMT are generally preferred in such applications.

- Almost in all tested applications KPLS is seemingly the model with the better fitting and the best overall performance, followed by the KPLSK model. EASY built-in RBFs are placed third in the overall metamodel ranking, followed by Kriging and RBFs found in SMT.

- Overall MAEAs using on-line trained surrogate models perform better than off-line trained ones, since the latter are heavily dependent on the fitting and the accuracy of the metamodel.

- The uncertainty of the results leads to the conclusion that the use of each optimization method and surrogate models heavily depends on the optimization problem. Consequently, a conclusion that applies to every application cannot be extracted, since the user must be the judge of the situation.

7.3 Future Work

Based on the results of this investigation the following can be proposed for future work:

- The efficacy of MAEAs degrades in high-dimensional optimization problems and commonly a plethora of methods are used to facilitate EAs implementation. Most common are distributed search models, e.g. Distributed EAs (DEAs)[56, 57], that distribute the individuals of any population in multiple semi-isolated subpopulations which are called demes. In computationally expensive problems, DEAs are assisted by metamodels (DMAEAs)[58]. It is common to combine DEAs with Hierarchical EAs (HEAs)[57, 59], thus creating HDEAs[60]. If then metamodels are used, HDMAEAs[61] are formed. HEAs have an hierarchical topology structure that resembles a binary tree that expands in two or more rarely three layers. Alternatively, Memetic Algorithms (MAs)[62], which are regarded as hybrid EAs and are similar to HEAs, combine conventional EAs with methods of refining individuals usually in the form of local search heuristics. When combined with metamodels, Metamodel Assisted MAs (MAMAs)[63, 64] are formed. Finally, the synchronization gap during each evolution of HEAs or DEAs led to the creation of Asynchronous EAs (AEAs)[65, 66] and AMAEAs[67]. Some of these methods are available in EASY, e.g. HEAs and DEAs, and can be utilized in future work.

- The prolonged wall clock time that resulted from the implementation of MAEA-based optimization with metamodels trained via SMT is attributed to the use of Python. However, the code responsible for the communication between EASY and SMT could be optimized via the use of Python in-house modules and packages. Alternatively, any Python function can be rewritten in C++ in order to drastically reduce the computational cost of the optimization.

Appendix A

Tests in Metamodel Fitting

The fitting of SMT metamodels is tested on a pair of low-fidelity models that are used as conceptual level estimate of the wing weight of an aircraft. The testing phase initiates by creating a DoE design w.r.t. a semi-arbitrarily imposed lower and upper bound for each design variable. This DoE design consists of n_t training data, which are subsequently evaluated on the problem-specific low-fidelity model, i.e. the exact evaluation model. The resulting n_t $(\vec{\beta}, \vec{f}(\vec{\beta})$ pairs are used for the training of a selected surrogate model. After the training process has been completed, a new set of sample points is selected from the design space via the use of DoE techniques. These are called validation points, are equal in number to training points, i.e. $n_t = n_{val} = 40$ and are used to validate the accuracy of the trained metamodel. Validation points are evaluated using both the exact evaluation model and the trained metamodel resulting in $\mathbf{F}$ and $\hat{\mathbf{F}}$ values, respectively. The deviation between these values for the n_t training data is calculated by the use of NRMSE, which serves as a metric for metamodel accuracy. The computations have been performed on a i7-9750H, 2.60 GHz CPU.

A.1 3 design variable aircraft wing equation

The quality of each metamodel is tested in a simplistic 3 design variable wing weight equation that is used for the conceptual design of light, low performance aircrafts. This equation applies to cantilever wings and includes the weight of wing tip and flight control surfaces, i.e. ailerons, while it excludes fuel tank weight, the effect of sweep angle and spar flight loads from wing and fuselage. The wing weight is determined from the following equation[68]:

$$W_{wing} = 0.0467 W_{TO}^{0.397} S^{0.397} N_z^{0.360} AR^{1.712} \tag{A.1}$$

where W_{TO} is the aircraft take-off weight, N_z the ultimate load factor, S the wing area and AR the wing aspect ratio. The search of a low-performance, light utility aircraft with maximum speed $V_{flight} < 200\ kn$ resulted in the selection of Evektor VUT1OO-131i Cobra[69]. The aforementioned aircraft is used as a template and its design values are set as a baseline for the range of each design variable in the previous equation.

VUT1OO-131i Cobra			
MTOW [lb]	**S** $[ft^2]$	$\mathbf{N_z}$	**AR**
3,197	141.1	3.8	8.4

Table A.1: Design values for VUT100-131i Cobra

By making the assumption that take-off weight is a constant parameter arbitrarily set equal to $W_{TO} = 2,500\ lb$, then the previous equation can be restated as such:

$$W_{wing} = 1.043 S^{0.397} N_z^{0.360} AR^{1.712} \tag{A.2}$$

The design space under study now consists of three independent design variables S, N_z, AR that define the design variable vector $\vec{\beta} = [S, N_z, AR]$. The range of each design variable is set arbitrarily as such:

- $120 < S < 170$
- $2 < N_z < 6$
- $6 < AR < 11$

The deviation between the predicted and the exact model value for n_{val} validation points is subsequently depicted in the following figures:

A.2 8 design variable aircraft wing equation

With the testing of the first objective function now complete, the ability of SMT metamodels to handle high-dimensional problems is still in question. For this reason, the software's capabilities will now be tested in an objective function consisting of 8 design variables. An increase in the number of design variables is expected to increase the complexity of the problem and therefore the computational cost. Consequently, an increase in design variables is a good metric of the software responsiveness to problems of higher dimensionality.

The second weight equation is used for the conceptual design of cargo/transport aircrafts. This equation accounts for the effect of sweep angle and excludes fuel tank weight and spar flight loads from wing and fuselage. The wing weight is determined from the following equation [68]:

$$W_{wing} = 0.0051(W_{dg}N_z)^{0.557}S^{0.649}AR^{0.0035}\left(\frac{t}{c}\right)_{root}^{-0.4}(1+\lambda)^{0.1}(cos\Lambda)^{-1}S_c^{0.1} \quad \text{(A.3)}$$

where W_{dg} is the flight design gross weight, N_z the ultimate load factor, S is the wing area, AR the aspect ratio, λ_r the taper ratio, Λ the quarter-chord sweep, (t/c) the airfoil thickness to cord ratio at the root of the wing and S_c flight control surface area. A search for a cargo aircraft resulted in the selection of Airbus A400M Atlas. Its design values are presented in the following table:

A400M Atlas		
$\boldsymbol{W_{dg}}$	264555	$[lb]$
$\mathbf{N_z}$	-	-
$\mathbf{S}$	2384	$[ft^2]$
$\mathbf{AR}$	8.1	-
$\mathbf{(t/c)_{root}}$	-	-
$\boldsymbol{\lambda_r}$	-	-
$\boldsymbol{\Lambda}$	15	[degrees]
$\mathbf{S_c}$	867	$[ft^2]$

Table A.2: Airbus A400M Atlas design values

where the constrictions for each variable involved are:

- $220000 \leq W_{dg} \leq 280000$
- $2.5 \leq N_z \leq 10$
- $2000 \leq S \leq 3000$
- $6 \leq AR \leq 10$
- $0.08 \leq t_c \leq 0.18$
- $0.5 \leq \lambda \leq 1$
- $-20 \leq \Lambda \leq 20$
- $400 \leq S_c \leq 2000$

A.3 Optimal construction method

The main parameter affecting the quality of an LHD is the construction criterion, denoted as *criterion* in Python. The fitting of the trained metamodel is used as a metric for the quality of the profuced design. The parametric analysis is applied on eq. A.3 that consists of 8 objective variables. The surrogate model utilized here is KPLS with regression model (denoted as *poly*) and correlation function (denoted as *corr*) set to their default values, i.e 'constant' and '*squar_exp*' respectively. The design space is reduced to $n\_comp = 3$ dimensions. The constructed designs consist of $n_{doe} = 25$ sample points.

NRMSE				
Criterion	**average**	**maximum**	**minimum**	**Standard deviation**
c	0.03724422	0.06002596	0.01691305	0.00860906
m	0.03731406	0.06309695	0.02201805	0.00823468
cm	0.03661358	0.08129392	0.00645364	0.01377246
corr	0.03634474	0.05715466	0.02144811	0.00869950
ese	0.03177641	0.04580328	0.01379077	0.00736666

Table A.3: Average NRMSE of each construction criterion that resulted from the use of KPLS model for $n_{test} = 50$ times

$$NRMSE = \sqrt{\frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \left(\frac{\hat{f}_i - f_i}{f_i} \right)^2} \tag{A.4}$$

where n_{val} is the number of validation points, in this case $n_{val} = 25$, while the average NRMSE in each case was calculated from the values that were collected from $n_{test} =$ evaluations. The entirety of the analysis is executed using an objective function $\vec{f} : n_\beta \to \mathbb{R}^n$. Consequently, $\vec{f}(\vec{\beta}) = f(\vec{\beta})$ is the scalar value of the objective function with inputs the validation points, while. $\hat{f}(\vec{\beta})$ is the predicted objective function value using the trained surrogate model with inputs, yet again, the validation points. NRMSE is used as a metric for model validation and therefore measures the accuracy of the metamodel. In contrast to regular RMSE, this metric does not depend on the order of magnitude of the observed values and the size of the sample, so it can be used in the comparison of various metamodels when approximating different PSMs and trained on different datasets. However, it is not the only option; the other two types of regression error most often used are:

1. R^2**/Adjusted** R^2 often called the coefficient of determination, measures how much of variability in dependent variable can be explained by the model[70]:

$$R^2 = 1 - \frac{\sum_{i=1}^{n_{val}} (y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n_{val}} (y_i - \overline{y}_i)^2} \tag{A.5}$$

where $y = F(\vec{\beta})$ and $\overline{y}$ is the mean of the observed objective function values:

$$\overline{y} = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \widehat{y} \tag{A.6}$$

R^2 value ranges in the span of $[\infty, 1]$. R^2 is negative when the model does not follow the trend of the data, equal to zero when the calculated model value is constant, disregarding the inputs and closer to 1 when the fit between prediction and actual model value is more accurate. R^2 indicates a goodness of fit and expresses the proportion of variance σ^2 of F_i that has been described by the independent design variables. However, it does not take into consideration overfitting of the problem. Consequently, a regression model that has many independent variables, due to its complexity, may fit well to the sample points but perform poorly for validation points. That is why Adjusted R^2 is introduced because it will penalise additional independent variables added to the model and adjust the metric to prevent overfitting issue. However, variance is dataset dependent and therefore the comparison between R^2 metrics obtained from different datasets is fruitful.

2. **Mean Absolute Error (MAE)** takes the sum of absolute value of error[70]:

$$MAE = \frac{1}{n_{val}} \sum_{i=1}^{N} \left| \hat{f}_i - f_i \right| \tag{A.7}$$

NRMSE is considered to be the optimal metric of displaying and comparing a model's goodness of fit. This metric is chosen by a process of elimination, since adjusted R^2 is not ideal for comparing different models, nor is it directly available in Python, and MAE does not indicate underperformance or overperformance of the model. A NRMSE closer to zero is considered to depict a well-constructed design, while the opposite is indicative of a poor design, which ignores important points in the process and/or includes outliers. The quality of the design is the most decisive factor in the selection of the most suitable LHD construction criterion, since the time needed to construct a design is negligible in MAEAs with off-line training. Consequently, criterion 'ese' is selected in LHS method, which results in ESE LHDs with the lowest NRMSE value among all LHDs. The time needed to construct each LHD is subsequently presented in the following table:

Average DoE implementation time					
	c	**m**	**cm**	**corr**	**ese**
time [s]	0.0003	0.0007	0.0007	0.0016	1.4079

Table A.4: Average LHD construction time, the process is repeated $n_{test} = 50$ times evaluations

A.4 Metamodel comparison

Now that the tests are complete, a vague impression for some of the surrogate models has already been formed. However, the accuracy of each model must be tested more thoroughly. In addition, accuracy is not by itself a decisive factor in the selection of the optimal surrogate model. Unlike the DoE construction phase, the training process is far more costly, especially in MAEA-based optimization with on-line trained surrogaet models, and therefore the required time must be taken into consideration. In order for the selection to be more discernible and indisputable, the analysis is performed on both eq. A.2 and eq. A.3, all the surrogate models are trained and validated on ESE LHDs of $n_{doe} = n_{val} = 25$ sample points; the same number of training patterns is used in MAEAs with on-line training.

Average NRMSE		
	3 design variables	**8 design variables**
Kriging	0.00041113	0.03102203
KPLS	0.00088127	0.03177641
KPLSK	0.00042970	0.02945811
RBF	0.00509959	0.04157616

Table A.5: Average NRMSE, the validation process is repeated $n_{test} = 50$ times

Average training time [s]		
	3 design variables	**8 design variables**
Kriging	0.0812	0.2156
KPLS	0.0536	0.0797
KPLSK	0.1368	0.2995
RBF	0.0007	0.0003

Table A.6: Average training time, the validation process is repeated $n_{test} = 50$ times

Due to the fact that every surrogate model has been trained using the same sampling method, leads to the conclusion that the generally lower NRMSE observed in RBF is a direct correlation to the poorer model fitting. Additionally, KPLSK displays increased training time,which is contributed to the optimization of the parameters θ calculation, combined with a goodness of fit similar to that of KPLS. Kriging has the best fitting in all applications but has the longest training time. KPLS is the most cost-efficient Kriging-based model and displays slightly inferior fitting in comparison.

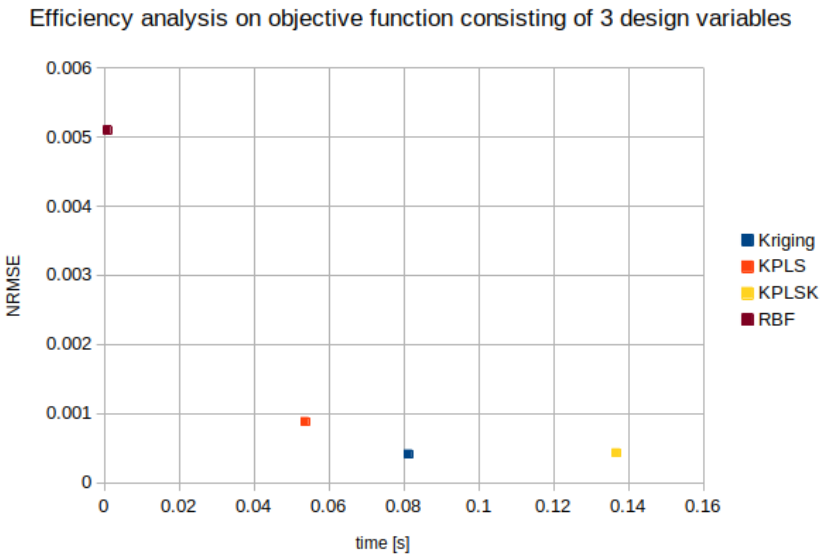


Figure A.1: Efficiency analysis on eq. A.2

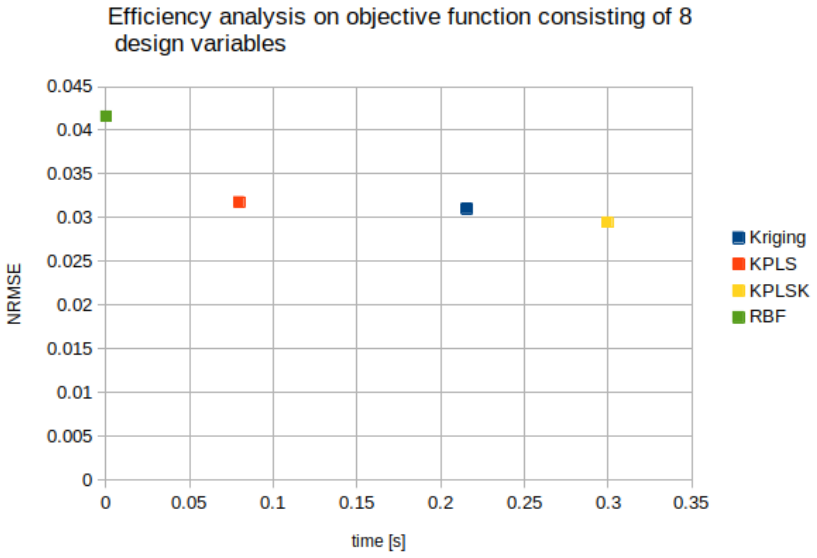


Figure A.2: Efficiency analysis on eq. A.3

Appendix B

SMT

B.1 DoE techniques in SMT

The implementation of the sampling process is facilitated by a Python-based, open-source software, called SMT. The DoE sampling techniques available in SMT are implemented in the 3D design space defined by the bounds of the 3 design variables in equation A.2, i.e. $\vec{\beta} = [\beta_1, \beta_2, \beta_3] = [S, N_z, AR]$, where $120 < S < 170$, $2 < N_z < 6$ and $6 < AR < 11$. The constructed designs consist of $n_{doe} = 40$ samples; the corresponding Python scripts are presented subsequently.

- **Random sampling**
  The creation of a randomly generated design can be accomplished in SMT by executing a Python script with the following form:

```
import numpy as np
from smt.sampling_methods import Random

ndoe = 40 #number of sample points
betalimits = np.array([[120.0,170.0], [2.0,6.0], [6.0,11.0]])
sampling = Random(xlimits = betalimits)
beta = sampling(ndoe)
```

Listing B.1: Implementation of Random sampling

- **Latin Hypercube Sampling (LHS)**
  LHS is a stratified sampling approach with the restriction that each of the input variables has its range well sampled following a probability distribution. In SMT, the creation of a LHD is performed by executing a Python script as such:

```
import numpy as np
from smt.sampling_methods import LHS

ndoe = 40 #number of sample points
betalimits = np.array([[120.0,170.0], [2.0,6.0], [6.0,11.0]])
sampling = LHS(xlimits = betalimits, criterion = 'ese')
beta = sampling(ndoe)
```

Listing B.2: Implementation of LHS

  The Python function LHS assumes various input values depending on the scheme used to construct the LHD. The process of constructing such a design

is facilitated by PyDOE[71], where the construction criterion is denoted by '*c*', '*m*', '*cm*', '*corr*' and '*ese*' to refer to Centered, Maximin, Maximin Centered, Maxent and ESE LHD, respectively, as depicted in the following script:

```
import numpy as np
from smt.sampling_methods import LHS

ndoe = 40 #number of sample points
betalimits = np.array([[120.0,170.0], [2.0,6.0], [6.0,11.0]])
crit = ['c', 'm', 'cm', 'corr', 'ese']
i = int(input('Define criterion between 0 and 4:'))
sampling = LHS(xlimits = betalimits, criterion = crit[i])
beta = sampling(ndoe)
```

Listing B.3: Various construction criteria

In SMT, the LHDs can be further optimized by expanding the initial design by a multiple of the initial number of sample points. The expanded design is implemented by setting the parameter *method* equal to either '*basic*' or '*ese*'. The former applies to centered, maximin, centered maximin and entropy LHDs, while the latter to ESE LHDs. The process of creating expanded LHDs is performed via the use of a Python script of the following format:

```
import numpy as np
from smt.sampling_methods import LHS

#Initial LHD
ndoe = 40 #number of sample points
betalimits=np.array([[120.0,170.0], [2.0,6.0], [6.0,11.0]])
sampling = LHS(xlimits = betalimits, criterion = 'ese', random_state = 1)
beta = sampling(ndoe)
#Expanded LHD
n_newdoe=ndoe #Points to be added
beta_new = sampling.expand_lhs(beta, n_newdoe, method = 'basic')
```

Listing B.4: Expanded LHD

where *random_state* is a keyword argument that is used to control the Mersenne Twister pseudo-random number generator If the input value can be an integer between 0 and $2^{32} - 1$, an array of integer values, or type *None* that is the default setting. As long as the argument assumes the same input value, it produces the same outcome and therefore this method is used for reproducibility.

- **Full Factorial**
  Although this sampling method is referred to as Full Factorial in SMT documentation, it creates either Full or Fractional Factorial Designs and can be used by executing a Python script that resembles the following form:

```
import numpy as np
from smt.sampling_methods import FullFactorial

ndoe = 40 #number of sample points
betalimits = np.array([[120.0,170.0], [2.0,6.0], [6.0,11.0]])
sampling = FullFactorial(xlimits = betalimits)
beta = sampling(ndoe)
```

Listing B.5: Implementation of Full Factorial sampling

B.2 Metamodel Training

- **Kriging**
  The four correlation functions in SMT are declared in the corresponding Python function as an additive parameter *corr*, which is set to 'abs_exp' (exponential), 'squar_exp' (Gaussian), 'matern52' and 'matern32', respectively. Additionally, it is possible to alternate between regression models by defining an extra input parameter in Python; that parameter is *poly* and can be set to 'constant', 'linear' or 'quadratic', which corresponds to a constant, linear or quadratic model as the notation implies. The correlation parameters $\theta \in \mathbb{R}^{n_\beta}$ are computed via the maximum likelihood estimation of equation 4.23, which is minimized via the use of COBYLA algorithm [36], provided by the open-source Python library sciPy[72]. A typical Python script that performs the training of Kriging model has the following form:

```
import numpy as np
from smt.surrogate_models import KPLS, KRG, KPLSK, RBF

....
ndim = 3 #number of problem dimensions
list0 = ['constant', 'linear', 'quadratic']
list1 = ['abs_exp', 'squar_exp', 'matern52', 'matern32']

polynomial = int(input('Define regression model between 0 and 3:'))
correlation = int(input('Define correlation function between 0 and 4:'))
th_range = np.array(([1e-8, 1e+3]))
t = KRG(theta0 = [1e-2]*ndim, theta_bounds = th_range, poly = list0[
    polynomial], corr = list1[correlation], eval_noise = False)
t.set_training_values(beta,F)
t.train()
```

Listing B.6: Training of Kriging model with noise-free observations via SMT

In order for the Python to account for the existing noise the parameter *eval_noise* must be set to 'True'. In addition, the lower bound for nugget must be defined by using the parameter *nugget* and the value of the initial noise parameters must be defined via the parameter *noise0*. The default values of *nugget* and *noise0* are $2.220446049250313 \cdot 10^{-14}$ and 0 for each design variable, respectively.

```
import numpy as np
from smt.surrogate_models import KRG

....
ndim = 3 #number of problem dimensions
th_range = np.array(([1e-8, 1e+3]))
t = KRG(theta0 = [1e-2]*ndim, theta_bounds = th_range, poly = 'constant',
    corr = 'squar_exp', eval_noise = True, noise0 = [1e-2])
t.set_training_values(beta,F)
t.train()
```

Listing B.7: Training of Kriging model with noisy observations via SMT

- **KPLS**
  The number of principal components in SMT is declared in the corresponding Python function via the additive parameter *n_comp*. All the other parameters are defined similarly to Kriging model.

```
import numpy as np
from smt.surrogate_models import KPLS

....
n_com = 2 #number of principal components
th_range = np.array(([1e-8, 1e+3]))
t = KPLS(n_comp = n_com, theta0 = [1e-2]*n_com, theta_bounds = th_range,
    poly = 'constant', corr = 'squar_exp')
t.set_training_values(beta,F)
t.train()
```

Listing B.8: Training of KPLS model with noisy observations via SMT

- **KPLSK**
  All the necessary parameters in KPLSK model are declared in Python script similarly to KPLS and Kriging.

```
import numpy as np
from smt.surrogate_models import KPLSK

....
n_com = 2 #number of principal components
th_range = np.array(([1e-8, 1e+3]))
t = KPLSK(n_comp = n_com, theta0 = [1e-2]*n_com, theta_bounds = th_range,
    poly = 'constant', corr = 'squar_exp')
t.set_training_values(beta,F)
t.train()
```

Listing B.9: Training of KPLSK model with noisy observations via SMT

- **RBF**
  In SMT, the degree and the existence of polynomials can be determined by modifying the value of the Python parameter *poly_degree*. Consequently, a value of -1, 0 or 1 is used to denote the absence of a polynomial trend, a constant trend and a linear trend respectively. If no polynomial trend is used, i.e. $poly\_degree = -1$, multivariate RBF interpolation is performed by solving the linear system described by equations 4.36 and 4.37. If however $poly\_degree = 1$, then RBF interpolation is performed by solving the linear system described by eq. 4.42.

```
import numpy as np
from smt.surrogate_models import RBF

....
t = RBF(d0 = 100, poly_degree = 1)
t.set_training_values(beta,F)
t.train()
```

Listing B.10: Training of KPLSK model with noisy observations via SMT

Appendix C

EASY

In order to gain a better grasp of how the communication channel between SMT and EASY[27] software is established, first, it is important to analyze the way EASY optimization software works.

- **EASY and EAs**

In EASY menu, the parameters of the optimization and both the bounds and the codification of the design variables can be modified. The evolutionary process via the use of EASY is performed as follows:

1. To perform an evaluation EASY writes and saves an ASCII text file *task.dat* that consists of $n_\beta + 1$ lines. The first line corresponds to the selected number of design variables n_β, while the remaining lines contain the values of each design variable and are listed according to the sequence declared by the user in EASY.

2. EASY executes the batch file *task.bat* and then halts all other processes until *task.bat* is finished. In EASY, this batch file is responsible for executing all processes responsible for the evaluation of candidate solutions (EAs-2). A typical *task.bat* has the following structure:

```
@echo off
erase results.dat
preprocessor.exe > nul
evaluation.exe > nul
postprocessor.exe > nul
```

Listing C.1: Structure of task.bat file

   where @echo off in *Windows* OS, prevents the the system terminal from printing any optimization-related results. The > *nul* command uses as output of the respective executable process the temporary file nul.

3. After the results of the previous run have been deleted, *preprocessor.exe* is executed. This executable is responsible for reading *task.bat* file and transforms the collected data in a readable format for the next process.

4. The evaluation of any untried candidate solution $\vec{\beta} \in P_{\lambda}^{g} \subset \mathbb{R}^{n_\beta}$ is performed via the execution of process *evaluation.exe* that contains the problem-specific model, in most cases a CFD tool.

5. The value of each objective and design variable value is then written in a ASCII text file *results.dat*; each line of that file contains a single value.

6. Subsequently, *postprocessor.exe* is executed. This executable reads the output of *evaluation.exe*, i.e. *results.dat* file, and allows the user to denote the necessary objectives and constraints of the optimization via the creation of *task.res* and *task.cns*, respectively.

7. After *task.bat* has finished, EASY expects to read a plain ASCII text file, *task.res*, which contains as many lines as the number of objectives. Each line of that file contains the value of the corresponding objective.

8. Most optimization problems are confronted w.r.t. a set of imposed constraints. In EASY, these constraints are declared in a plain ASCII text file, *task.cns*, which contains as many lines as the number of constraints imposed.

Bibliography

[1] Y.S. Ong, P.B. Nair, and A.J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4): 687–696, 2003.

[2] M.A. Bouhlel, J.T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J.R.R.A. Martins. A Python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135, 2019.

[3] D.C. Montgomery. Design and Analysis of Experiments. John Wiley & Sons, New York, USA, 6th edition, 2005.

[4] Κ. Χ. Γιαννάκογλου. Μέθοδοι Βελτιστοποίησης στην Αεροδυναμική. Πανεπιστημιακές Εκδόσεις Ε.Μ.Π., Αθήνα, σελίδες 125-130, 2006.

[5] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3): 221–248, 1995.

[6] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4): 257–271, November 1999.

[7] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature – PPSN VI*, Paris, France, 2000.

[8] E. Zitzler, M. Laumans, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Eurogen 2001, Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, Barcelona, pp. 19-26, 2002.

[9] M. Farina. A neural network based generalized response surface multiobjective evolutionary algorithm. In *2002 Congress on Evolutionary Computation – CEC '02*, Honolulu, HI, USA, May 2002.

[10] J.R. Wagner, E.M. Mount, and H.F. Giles. *Plastics Design Library, Volume: Extrusion*, chapter Design of Experiments, Elsevier Science, 2nd edition, pp. 291-308, 2014.

[11] A. Sethuramiah and R. Kumar. *Modeling of Chemical Wear*, chapter Statistics and Experimental Design in Perspective, Elsevier Science, pages 129-159, 2016.

[12] R. Mead, S. Gilmour, and A. Mead. *Statistical principles for the design of experiments*. Cambridge University Press, 1st ed., pp. 233-271, 2012.

[13] M. Stein. Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics*, 29: 143-151, 1987.

[14] I. Ronald. Latin Hypercube Sampling. *ResearchGate*, January 1999.

[15] M. McKay, R. Beckman, and W. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21: 239–245, 1979.

[16] M. Johnson, L. Moore and D. Ylvisaker. Minimax and maximin distance designs, *Journal of Statistical Planning and Inference*, 26(2): 131-148, 1990.

[17] T. Santner, B. Williams, and W. Notz. *The Design and analysis of computer experiments*. Springer, 2nd edition, pp. 145-200, 2018.

[18] C.E. Shannon, A mathematical theory of communication. *Bell System Technical Journal*, 27(3): 379-423, 1948.

[19] M.C. Shewry and H.P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2): 165-170, 1987.

[20] Y.G. Saab, Y.B. Rao. Combinatorial optimization by stochastic evolution IEEE Trans. *Computer-Aided Design*, pp. 525-535, September 1991,

[21] R. Jin , W. Chen, and A. Sudjianto. An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference*, 134(1): 268-287, 2005.

[22] M.D. Morris and T.J. Mitchell. Exploratory Designs for Computational Experiments. *Journal of statistical planning and inference*, 43: 381-402, 1995.

[23] F.J. Hickernell. A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, 67: 299-322, 1998.

[24] D.C. Montgomery. *Design and Analysis of Experiments*. Wiley, 9th edition, pp. 179-229, 2017.

[25] J. Antony. *Design of Experiments for Engineers and Scientists*, chapter Full Factorial Designs. Elsevier Science, 2nd edition, pp. 63-85, 2014.

[26] D.C. Montgomery. *Design and Analysis of Experiments*. Wiley, 9th edition, pp. 351-384, 2017.

[27] EASY - The Evolutionary Algorithms SYstem Home, 2012. Retrieved from http://velos0.ltt.mech.ntua.gr/EASY

[28] M.K. Karakasis and K.C. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8):941–957, 2006.

[29] M.K. Karakasis and K.C. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8): 941-957, 2006.

[30] A. Keane, A. Forrester, and A. Sóbester. *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 1st edition, 2008.

[31] C.E. Rasmussen and C.K.I. Williams. Gaussian Processes for Machine Learning, the MIT Press, 2006.

[32] G. Giangaspero, D. MacManus, and I. Goulos. Surrogate models for the prediction of the aerodynamic performance of exhaust systems. *Aerospace Science and Technology*, 92: 77-90, 2019.

[33] S.N. Lophaven, H.B. Nielsen, J. Søndergaard. *DACE: a MatLab Kriging Toolbox*. Technical Report IMM-TR-2002-12, 2002.

[34] D.R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4): 345–383, 2001.

[35] J. Sacks, S.B. Schiller, and W.J. Welch. Designs for computer experiments. *Technometrics*, 31(1): 41-47, 1989.

[36] M.J.D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In S. Gomez and J-P Hennart, *Advances in Optimization and Numerical Analysis*, Kluwer Academic (Dordrecht), pp. 51-67, 1994.

[37] A.E. Hoerl. (1962). Application of ridge analysis to regression problems. *Chemical Engineering Progress* 58, 54-59, 1962.

[38] I. Helland. On structure of Partial Least Squares re- gression. *Communication in Statistics - Simulation and Computation* 17: 581–607, 1988.

[39] M.A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier. Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53(5): 935-952, 2016.

[40] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4): 255-282, 1950.

[41] R. Manne. Analysis of two Partial-Least-Squares algorithms for multivariate calibration. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3): 187-197, 1987.

[42] M.A. Bouhlel, N. Bartoli, J. Morlier, and A. Otsmane. An improved approach for estimating the hyperparameters of the kriging model for high-dimensional problems through the partial least squares method. *Mathematical Problems in Engineering*, 2016.

[43] M.J.D. Powell. *The Theory of Radial Basis Function Approximation*. Oxford University Press, pp. 105-210, 1992.

[44] M. N. Oqielat. Scattered data approximation using radial basis function with a cubic polynomial reproduction for modelling leaf surface. *Journal of Taibah University for Science*, 12(3): 331-337, 2018.

[45] V. Bayona. An insight into RBF-FD approximations augmented with polynomials. *Computers & Mathematics with Applications*, 77(9): 2337-2353, 2019.

[46] T. Ray and K. M. Liew. A Swarm Metaphor for Multiobjective Design Optimization. *Engineering Optimization* 34: 141-153, 2002.

[47] J.D. Knowles, D.W. Corne, and M. Fleischer. Bounded Archiving using the Lebesgue Measure. *Congress on Evolutionary Computation*, 4: 2490–2497, 2003.

[48] S.S. Rao. *Engineering Optimization: Theory and Practice*. Wiley, 5th ed., pp. 434-435, 2020.

[49] J. Golinski. Optimal synthesis problems solved by means of nonlinear programming and random methods. *Journal Of Mechanisms*, 5(3), 287-309, 1970.

[50] P.R. Spalart and S.R. Allmaras. A One-Equation Turbulence Model for Aerodynamic Flows. *30th Aerospace Sciences Meeting and Exhibit*, 1992.

[51] K. Tsiakas. Development of shape parameterization techniques, a flow solver and its adjoint, for optimization on GPUs. Turbomachinery and external aerodynamics applications. PhD thesis, Laboratory of Thermal Turbomachines, NTUA, Athens, 2019.

[52] I.C. Kampolis, X.S Trompoukis, V.G. Asouti, K.C. Giannakoglou. CFD-based analysis and two-level aerodynamic optimization on graphics processing units. *Computer Methods in Applied Mechanics and Engineering*, 199(9–12): 712–722, 2010.

[53] A.J.A Favre. Equations des gaz turbulents compressibles. *Journal de Mecanique*, 4, 1965.

[54] G. Kalitzin, G. Medic, G. Iaccarino and P. Durbin. Near-wall behaviour of RANS turbulence models and implications for wall functions. *Journal of Computational Physics*, 204: 265-291, 2005.

[55] S. Allmaras, F. Johnson, and P. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. *7th International Conference on Computational Fluid Dynamics*, 2012.

[56] F. Herrera, M. Lozano, and C. Moraga. Hierarchical distributed genetic algorithms. *International Journal of Intelligent Systems*, 14(11): 1099–1121, 1999.

[57] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34: 286–300, 2015.

[58] W. Annicchiarico. Metamodel-assisted distributed genetic algorithms applied to structural shape optimization problems. *Engineering Optimization*, 39(7): 757-772, 2007.

[59] J.F. Wang and J. Periaux and M. Sefrioui. Parallel evolutionary algorithms for optimization problems in aerospace engineering. *Journal of Computational and Applied Mathematics*, 149(1): 155-169, 2002.

[60] I.C. Kampolis and K.C. Giannakoglou. Distributed evolutionary algorithms with hierarchical evaluation. *Engineering Optimization*, 41(11): 1037–1049, 2009.

[61] M. K. Karakasis, D. G Koubogiannis, and K. C Giannakoglou. Hierarchical distributed metamodel-assisted evolutionary algorithms in shape optimization. *International Journal for Numerical Methods in Fluids*, 53(3): 455–469, 2006.

[62] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5): 474–488, 2005.

[63] C.A. Georgopoulou and K.C. Giannakoglou. Memetic Algorithms. *Springer Series*, 2009.

[64] C.A. Georgopoulou and K.C. Giannakoglou. A multi-objective metamodel-assisted memetic algorithm with strength-based local refinement. *Engineering Optimization*, 41(10): 909–923, 2009.

[65] E.O. Scott and K.A De Jong. Understanding Simple Asynchronous Evolutionary Algorithms. *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. FOGA '15: Foundations of Genetic Algorithms XIII, 2015.

[66] V.G. Asouti and K.C. Giannakoglou. Aerodynamic optimization using a parallel asynchronous evolutionary algorithm controlled by strongly interacting demes. *Engineering Optimization*, 41(3): 241–257, 2009.

[67] V.G. Asouti, I.C. Kampolis, and K.C. Giannakoglou. A grid-enabled asynchronous metamodel-assisted evolutionary algorithm for aerodynamic optimization. *Genetic Programming and Evolvable Machines (SI:Parallel and Distributed Evolutionary Algorithms, Part One)*, 10(3): 373–389, 2009.

[68] D.P. Raymer. Aircraft Design: A Conceptual Approach. AIAA Education Series, 6th edition, pp. 559-584, 2018.

[69] A. Pistek and R. Popela. The VUT 100/200 General Aviation Aircraft Family: Project and Realization. Proceedings of the Institution of Mechanical Engineers, Part G: *Journal of Aerospace Engineering*, 221(2): 193-197, 2007.

[70] *3.3. Metrics and scoring: quantifying the quality of predictions* — scikit-learn 0.24.1 documentation. Retrieved January 17, 2021 from https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics.

[71] pyDOE: The experimental design package for python — pyDOE 0.3.6 documentation. Retrieved February 2021 from https://pythonhosted.org/pyDOE/index.html.

[72] scipy.optimize.fmin_cobyla — SciPy v1.6.1 Reference Guide. Retrieved 17 January, 2021 from https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_cobyla.html.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Περί Βέλτιστης Χρήσης Μεταπροτύπων στους Εξελικτικούς Αλγορίθμους με Εφαρμογές στην Αεροδυναμική

Διπλωματική Εργασία

Μιχάλης Δημήτριος

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2022

ΕΚΤΕΝΗΣ ΠΕΡΙΛΗΨΗ ΣΤΑ ΕΛΛΗΝΙΚΑ

Στο πλαίσιο αυτής της διπλωματικής εργασίας μελετάται η εφαρμογή εξελικτικών αλγορίθμων υποβοηθούμενων από μεταμοντέλα (Metamodel-Assisted EAs MAEAs) σε διάφορες εφαρμογές μηχανολογικού ενδιαφέροντος. Δύο είναι οι κύριες μέθοδοι βελτιστοποίησης με εφαρμογή των MAEAs και σχετίζονται με τον τρόπο εκπαίδευσης των μεταμοντέλων, δηλαδή συνδεδεμένα (on-line) και αποσυνδεδεμένα (off-line) από την εξέλιξη. Και οι δύο αυτοί μέθοδοι εφαρμόζονται με τη βοήθεια ενός εξωτερικού λογισμικού με βάση την Python, που ονομάζεται Surrogate Model Toolbox (SMT) [1], και συγκρίνονται με τους κοινούς EAs με βάση την αποτελεσματικότητα και το υπολογιστικό κόστος που προκύπτει από τη χρήση τους. Η βελτιστοποίηση σε κάθε περίπτωση πραγματοποιείται με τη χρήση του EASY[2] (Evolutionary Algorithm SYstem), ενός λογισμικού που αναπτύχθηκε από τη Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης (ΜΠΥΡΒ) του ΕΜΠ. Από τα διάφορα ενσωματωμένα μεταμοντέλα που υπάρχουν στον EASY, οι συναρτήσεις ακτινικής βάσης [3] (Radial Basis Functions RBFs) χρησιμοποιούνται στην παρούσα διπλωματική εργασία. Ωστόσο, η βελτιστοποίηση μέσω του EASY μπορεί να υποβοηθηθεί από εξωτερικά μεταπρότυπα, τα οποία είναι διαθέσιμα στο SMT. Από αυτά τα εξωτερικά μεταμοντέλα, σε αυτή τη διπλωματική εργασία γίνεται χρήση κυρίως του Kriging[4], των παραλλαγών του για χώρο σχεδιασμού μειωμένων διαστάσεων χάρη στην εφαρμογή της μεθόδου μερικών ελαχίστων τετραγώνων (Partial Least Squares PLS), κυρίως του KPLS[6] και του KPLSK[7], καθώς και των RBFs. Κάθε μια απο τις μεθόδους βελτιστοποίησης εφαρμόζεται σε πρώτο στάδιο σε απλά προβλήματα ψευδο-μηχανικής, κυρίως στην περίπτωση της συγκολλητής δοκού και του μειωτήρα ταχύτητας, ενώ στη συνέχεια στη βελτιστοποίηση μορφής μιας δισδιάστατης αεροτομής. Η επίλυση των εξισώσεων Reynolds Averaged Navier-Stokes (RANS) συμπιεστού ρευστού γύρω από την αεροτομή γίνεται με τη χρήση ενός CFD επιλύτη, που ονομάζεται PUMA[8] και αναπτύχθηκε από τη ΜΠΥΡΒ/ΕΜΠ.

- **Εξελικτικοί Αλγόριθμοι (EA)**

  Οι EAs[9] έχουν τη βάση τους στην εξελικτική θεωρία του Δαρβίνου που ορίζει την κυριαρχία λ_α ατόμων σε κάθε πληθυσμιακή γενιά (μ,λ). Σε κάθε γενιά αξιολογούνται λ παιδιά στο PSM, τα οποία βρίσκονται στο προσωρινό πληθυσμιακό σύνολο P_e, και τους ανατίθεται μια τιμή από μια συνάρτηση κόστους $\Phi(\vec{\beta})$. Με βάση την τιμή που έχει το κάθε υποψήφιο άτομο διαλέγονται οι ελίτ της συγκεκριμένης γενιάς και ανανεώνεται το πληθυσμιακό σύνολο των ελίτ P_α^{g+1} μέσω μιας διαδικασίας ελιτισμού. Στη συνέχεια σχηματίζεται το σύνολο των γονιών της επόμενης γενιάς P_μ^{g+1}, καθώς και των νέων παιδιών μέσω μιας διαδικασίας ανάμειξης των γονιών μ που συνοδεύεται από μια διαδικασία μετάλλαξης.

- **MAEAs με off-line εκπαίδευση**

  Στην βελτιστοποίηση με χρήση MAEAs με off-line εκπαίδευση, ένα καθολικό μεταμοντέλο εκπαιδεύεται αποκομμένα από την εξέλιξη, με βάση n'_{doe} σημείων $\mathbf{X}$ που επιλέχθηκαν χάρη στην εφαρμογή κάποιας DoE μεθόδου δειγματοληψίας. Στη συνέχεια, η εξέλιξη πραγματοποιείται παρόμοια με τη μέθοδο των απλών εξελικτικών, με τη μόνη διαφορά ότι η αξιολόγηση των παιδιών πλέον γίνεται με τη χρήση των εκπαιδευμένων μεταμοντέλων. Η σύγκλιση της μεθόδου βασίζεται στην ακρίβεια του καθολικού μεταμοντέλου. Οι μέθοδοι DoE του SMT με βάση των οποίων πραγματοποιείται η δειγματοληψία είναι εξής:

  - Τυχαία δειγματοληψία (Random sampling)[10]
  - Latin Hypercube Sampling (LHS)[11]
    - Centered
    - Maximin[12]
    - Maximin centered
    - Maxent[13]
    - Enhanched Stochastic Evolutionary (ESE)[14]
  - Παραγοντική δειγματοληψία (πλήρης ή μερική)(Factorial sampling)[15]

- **MAEAs με on-line εκπαίδευση**

  Η εκπαίδευση των μεταμοντέλων σε αυτή τη μέθοδο γίνεται αποτελεί μέρος της εξέλιξης μέσω μιας διαδικασίας προσεγγιστικής προ-αξιολόγησης (Low-Cost Pre-Evaluation LCPE [5]). Στο πλαίσιο της διαδικασίας αυτής γίνεται η επιλογή κατάλληλων μοτίβων εκπαίδευσης στην γειτονιά του κάθε υπο-αξιολόγησή $\vec{\beta}$ βάση των οποίων εκπαιδεύονται τοπικά, προσωποποιημένα μεταμοντέλα.

Η σύζευξη SMT και EASY απαιτεί τη σύνταξη των εξής κωδίκων στην Python που εκτελούν τις παρακάτω ενέργειες:

1. Αξιολόγηση των δειγμάτων με το PSM
2. Εκπαίδευση του μεταμοντέλου
3. Πρόβλεψη με βάση το εκπαιδευμένο μεταμοντέλο

Η ιδιαιτερότητα και η καινοτομία του EASY βρίσκεται στην on-line εκπαίδευση, όπου η επέμβαση του χρήστη στην περίπτωση των εξωτερικών μεταμοντέλων γίνεται με την κλήση των ανωτέρω κωδίκων. Οι κώδικες αυτοί καλούνται στην συνέχεια από τον EASY μέσω κατάλληλα διαμορφωμένων batch αρχείων. Η χρήση εσωτερικών μεταμοντέλων απαιτεί τη σύνταξη μόνο ενός κώδικα που είναι υπεύθυνος για την αξιολόγηση των υποψήφιων λύσεων.

Στους MAEAs με off-line εκπαίδευση όλες οι διεργασίες της βελτιστοποίησης καλούνται μέσα από ένα καθολικό κώδικα της Python και επιπρόσθετα συντάσσεται ένας κώδικας που εκτελεί τη δειγματοληψία, καθώς και ένας για την επαναξιολόγηση της 'βέλτιστης' λύσης με το PSM. Η off-line εκπαίδευση είναι δυνατό να πραγματοποιηθεί από εσωτερικά μεταμοντέλα του EASY, ωστόσο δεν ενδείκνυται.

Στην βελτιστοποίηση μέσω του EASY με MAEAs με off-line και on-line εκπαίδευση γίνεται χρήση εξωτερικών μεταμοντέλων που είναι διαθέσιμα στο SMT. Αυτά είναι τα ακόλουθα:

- **Kriging**

  Στο Kriging η προσέγγιση της επιθυμητής λύσης γίνεται μέσω της συνάρτησης:

  $$\hat{\mathrm{f}}(\vec{\beta}) = \vec{p}^{\,T}(\vec{\beta})\hat{\vec{\mathrm{w}}} + \vec{r}_{X\beta}\mathbf{R}^{-1}\left(\mathbf{F} - \mathbf{P}\hat{\vec{\mathrm{w}}}\right) \tag{C.1}$$

  όπου $\mu_K = \vec{p}^{\,T}(\vec{\beta})\hat{\vec{\mathrm{w}}}$ είναι ένας ντετερμινιστικός όρος που μπορεί να εκφραστεί ως σταθερό, γραμμικό ή τετραγωνικό μοντέλο παλινδρόμησης. Ο 2ος όρος είναι πραγματοποίηση μιας Gaussian διεργασίας $z(\vec{\beta}) \backsim N(0, C)$ με μηδενική μέση τιμή και συνδιακύμανση $C(\vec{\beta}) = \sigma^2 R(\vec{\chi}_i, \vec{\chi}_j)$ των παρατηρούμενων μεγεθών. Η συσχέτιση μεταξύ των μεγεθών δίνεται στο SMT από την εκθετική συνάρτηση:

  $$R\left(\vec{\chi}_i, \vec{\chi}_j\right) = \prod_{l=1}^{n_\beta} exp\left(-\theta_l \left|\chi_{i,l} - \chi_{j,l}\right|\right) \tag{C.2}$$

  ενώ διατίθενται και οι συναρτήσεις Gaussian, Matérn 5/2 και Matérn 3/2. Στο Kriging δίνεται η δυνατότητα δυνατότητα να υπολογιστεί και η σχετική αβεβαιότητα της πρόβλεψης από τη σχέση:

  $$MSE(\vec{\beta}) = \hat{\sigma}^2(1 - \vec{r}_{X\beta}^{\,T}\mathbf{R}^{-1}\vec{r}_{X\beta}) \tag{C.3}$$

  Η επίλυση των παραπάνω εξισώσεων απαιτεί τον υπολογισμό των παραμέτρων συσχέτισης $\theta \in \mathbb{R}^{n_\beta}$ που προκύπτουν από τη μεγιστοποίηση της συνάρτησης πιθανοφάνειας που δίνεται από την εξίσωση:

  $$\begin{aligned} ln(l_F(\vec{\theta}|\mathbf{F})) = &-\frac{n_t}{2} ln\left[\frac{1}{n_t}\left(\mathbf{F} - \mathbf{P}(\mathbf{P}^T\mathbf{R}^{-1}\mathbf{P})^{-1}\mathbf{P}^T\mathbf{R}^{-1}\mathbf{F}\right)^T\right. \\ &\left.\times\, \mathbf{R}^{-1}\left(\mathbf{F} - \mathbf{P}\left(\mathbf{P}^T\mathbf{R}^{-1}\mathbf{P}\right)^{-1}\mathbf{P}^T\mathbf{R}^{-1}\mathbf{F}\right)\right] + ln(det\mathbf{R}) \end{aligned} \tag{C.4}$$

- **KPLS**

  Σε μια προσπάθεια να ελαττωθεί ο χρόνος κατασκευής του Kriging σε χώρους σχεδιασμού υψηλής διαστατικότητας γίνεται εφαρμογή της μεθόδου PLS, που οδηγεί σε αναγωγή του χώρου σχεδιασμού σε $h < n_\beta$ διαστάσεις. Η χρήση του KPLS μοντέλου οδηγεί σε προσαρμογή της εκθετικής (και Gaussian) συσχέτισης ως εξής:

$$R\left(\vec{\chi_i}, \vec{\chi_j}\right) = \prod_{l=1}^{h} \prod_{k=1}^{n_\beta} exp\left(-\theta_l \left| D_{*k}^{(l)} \chi_{i,k} - D_{*k}^{(l)} \chi_{j,k} \right|\right) \tag{C.5}$$

  όπου το μητρώο $\mathbf{D}_* = [\vec{D}_*^{(1)}, \vec{D}_*^{(2)}, \ldots, \vec{D}_*^{(h)}] \in \mathbb{R}^{n_\beta \times h}$ εκφράζει την επιρροή κάθε διάστασης $k \in [1, n_\beta]$ στην κατασκευή της l-ης μετασχηματισμένης διάστασης, έτσι ώστε το $\theta \in \mathbb{R}^h$.

- **KPLSK**

  Οι μετασχηματισμένες συναρτήσεις συσχέτισης του KPLS είναι ορισμένες σε υποσύνολο του n_β-διάστατου χώρου. Η ιδέα πίσω από το KPLSK είναι να εκφραστούν οι συναρτήσεις αυτές στην ολότητα του n_β-διάστατου χώρου όπου είναι εκφρασμένο το Kriging. Αυτό επιτυγχάνεται στη Gaussian συνάρτηση, η οποία γράφεται στη μορφή:

$$R\left(\vec{\chi_i}, \vec{\chi_j}\right) = \prod_{l=1}^{h} \prod_{k=1}^{n_\beta} exp\left(-\theta_l \left(D_{*k}^{(l)} \chi_{i,k} - D_{*k}^{(l)} \chi_{j,k} \right)^2\right) = \prod_{k=1}^{n_\beta} exp\left(-\eta_k \left(\chi_{i,k} - \chi_{j,k}\right)^2\right) \tag{C.6}$$

  όπου ο όρος $\eta_k = \sum_{l=1}^{h} \theta_l D_{*k}^{(l)^2}$ για $k = 1, 2, \ldots, n_\beta$ δρα ως αρχικό σημείο για την τοπική βελτιστοποίηση της συνάρτησης πιθανοφάνειας του Kriging με βάση τις τιμές των παραμέτρων $\theta^{(l)}$ που προκύπτουν από την εφαρμογή του KPLS για $l = 1, 2 \ldots, h$.

- **RBFs**

  Το μεταμοντέλο των RBFs, χρησιμοποιεί n_t παρατηρούμενα ζεύγη $(\vec{\chi}, f(\vec{\chi}))$ για να εκφράσει τη συνάρτηση προσέγγισης σε κάποιο σημείο $\beta \in \mathbb{R}^{n_\beta}$ ως γραμμικό συνδυασμό συναρτήσεων βάσης με κέντρα το εκάστοτε σημείο $(\vec{\chi}_\xi$, που εκφράζεται μαθηματικά ως:

$$\begin{aligned} s(\vec{\beta}) &= \sum_{j=1}^{n_t} \mathrm{w}_{t_j} g\left(\|\vec{\beta} - \vec{\chi}_j\|\right) = \sum_{j=1}^{n_t} \mathrm{w}_{t_j} g(r_j) \\ &\text{έτσι ώστε} \quad s(\vec{\chi}_i) = F(\vec{\chi}_i) = F_i \quad \text{, for } i = 1, n_t \end{aligned} \tag{C.7}$$

  Οι μέθοδος αυτή είναι δυνατόν να συνδυαστεί και με ένα γραμμικό μοντέλο παλινδρόμησης:

$$s(\vec{\beta}) = \sum_{j=1}^{n_t} \mathrm{w}_{t_j} g\left(\|\vec{\beta} - \vec{\chi}_j\|\right) + \sum_{i=1}^{k} \mathrm{w}_i p_i(\vec{\beta}) \tag{C.8}$$

- **Εφαρμογή σε προβλήματα ψευδο-βελτιστοποίησης**

Η αποτελεσματικότητα των μεταμοντέλων δοκιμάζεται αρχικά σε δυο απλές εφαρμογές ψευδο-μηχανικής χαμηλού υπολογιστικού κόστους. Η πρώτη εφαρμογή αφορά τη μείωση του κόστους κατασκευής μια συγκολλητής ράβδου σε [$][16] και η δεύτερη τη μείωση του βάρους ενός μειωτήρα ταχύτητας σε [g][17]. Η μελέτη σε αυτή τη διπλωματική επικεντρώθηκε στη σύγκριση μεταξύ των αποτελεσμάτων που προέκυψαν από τη βελτιστοποίηση με χρήση απλών εξελικτικών, MAEAs με off-line εκπαίδευση και MAEA με on-line εκπαιδευμένα μεταμοντέλα τα οποία εκαπιδεύονται μέσω του SMT και του EASY. Τα αποτελέσματα της μελέτης παρουσιάζονται στη συνέχεια:

1. ***Περίπτωση συγκολλητής δοκού ενός στόχου***
   Πρόβλημα 4 μεταβλητών σχεδιασμού $\vec{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4) = (h, l, t, b) \in \mathbb{R}^4$ με 5 περιορισμούς:

$$
\begin{aligned}
& min \;\; f(\vec{\beta}) = 1.10471\beta_1^2\beta_2 + 0.04811\beta_3\beta_4(14.0 + \beta_2) \\
\text{subject to} \;\; & c_1(\vec{\beta}) = \tau(\vec{\beta}) - \tau_{max} \leq 0 \\
& c_2(\vec{\beta}) = \sigma(\vec{\beta}) - \sigma_{max} \leq 0 \\
& c_3(\vec{\beta}) = \beta_1 - \beta_4 \leq 0 \\
& c_4(\vec{\beta}) = \delta(\vec{\beta}) - \delta_{max} \leq 0 \\
& c_5(\vec{\beta}) = P - P_c(\vec{\beta}) \leq 0
\end{aligned}
\tag{C.9}
$$

   Ακολουθεί η σύγκριση μεταξύ των διαθέσιμων μεταμοντέλων του SMT 5 αρχικοποιήσεις RNG και τερματισμό στις 10000 PSM αξιολογήσεις:

Περίπτωση συγκολλητής δοκού					
MAEAs, on-line	(μ, λ) **population**	**KPLS**	**KPLSK**	**Kriging**	**RBFs**
SMT	(30, 100)	2.45	2.74	2.72	2.62

Table C.1: Περίπτωση συγκολλητής δοκού με RNG1. Η βέλτιστη λύση βρέθηκε από MAEA με on-line εκπαιδευμένων μεταμοντέλων μέσω του SMT

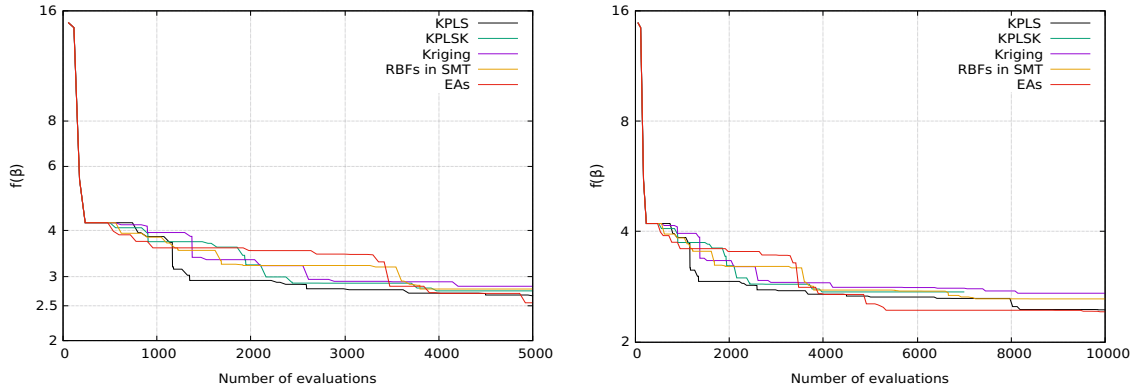


Figure C.1: Περίπτωση συγκολλητής δοκού με RNG1. Σύγκριση μεταξύ της σύγκλισης EAs and MAEAs με on-line εκπαιδευμένων μεταμοντέλων μέσω SMT

Το πόρισμα της μελέτης είναι ότι η χρήση του KPLS οδηγεί σε βέλτιστη σύγκλιση του προβλήματος της συγκολλητής δοκού, επομένως το KPLS συγκρίνεται στη συνέχεια με τις on-line εκπαιδευμένες RBFs που διαθέτει ο EASY.

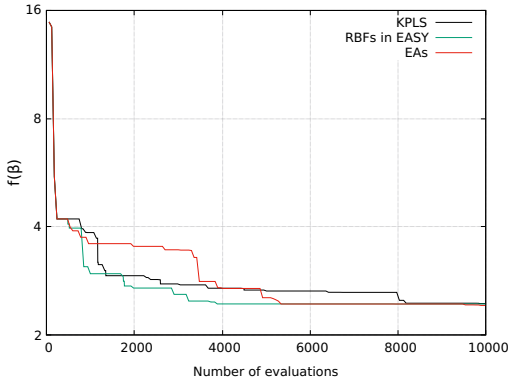


Figure C.2: Περίπτωση συγκολλητής δοκού με RNG1. Σύγκριση μεταξύ του ιστορικού σύγκλισης απλών EAs and MAEAs με χρήση on-line εκπαιδευμένων μεταμοντέλων μέσω SMT και του EASY

Το ιστορικό σύγκλισης φανερώνει την υπεροχή MAEAs με χρήση των on-line εκπαιδευμένων και ενσωματωμένων στον EASY RBFs έναντι των υπόλοιπων μεθόδων βελτιστοποίησης.

2. ***Περίπτωση συγκολλητής δοκού δύο στόχων***
   Σε αυτή την περίπτωση τίθεται ως δεύτερος στόχος η μείωση της παραμόρφωσης δ στο άκρο της δοκού και η βελτιστοποίηση πραγματοποιείται με MAEA με on-line εκπαίδευση για 1000 PSM αξιολογήσεις:

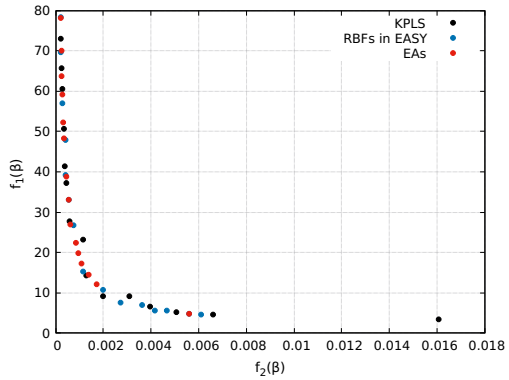


Figure C.3: Μέτωπο Pareto των 15 μη-κυριαρχούμενων υποψήφιων λύσεων που βρέθηκαν μετά την πραγματοποίηση 1000 PSM αξιολογήσεων

Δείκτης υπερόγκου για RNG1 H($\mathcal{F}$)	
MAEAs, on-line, SMT	1.6222
MAEAs, on-line, EASY	1.6215
EAs	1.6061

Table C.2: Περίπτωση συγκολλητής δοκού. Δείκτης υπερόγκου των μετώπων Pareto που προέκυψαν από την εφαρμογή των διαφόρων μεθόδων βελτιστοποίησης

Σε αυτή την περίπτωση φαίνεται πως τα MAEA με χρήση on-line εκπαιδευμένων μεταμοντέλων μέσω του SMT οδηγούν στη δημιουργία του καλύτερου μετώπου Pareto συγκριτικά με τις υπόλοιπες μεθόδους.

3. ***Περίπτωση μειωτήρα ταχύτητας ενός στόχου***
Πρόβλημα ελαχιστοποίησης του βάρους σε [g] ενός μειωτήρα ταχύτητας με 7 μεταβλητές σχεδιασμού και 11 κατασκευαστικούς περιορισμούς:

$$min \;\; f(\vec{\beta}) = 0.7854bm^2\left(3.3333N_{teeth}^2 + 14.9334N_{teeth} - 43.0934\right) - 1.508\left(d_1^2 + d_2^2\right) \\ 7.4777\left(d_1^3 + d_2^3\right) + 0.7854\left(L_1d_1^2 + L_2d_2^2\right) \tag{C.10}$$

Αρχικά έγινε σύγκριση μεταξύ της σύγκλισης MAEA με on-line εκπαιδευμένων μεταμοντέλων μέσω του SMT. Η βελτιστοποίηση με EAs και MAEAs έχει σε κάθε περίπτωση ως κριτήριο τερματισμού τις 20000 PSM αξιολογήσεις και επαναλήφθηκε με 5 RNG τιμές.

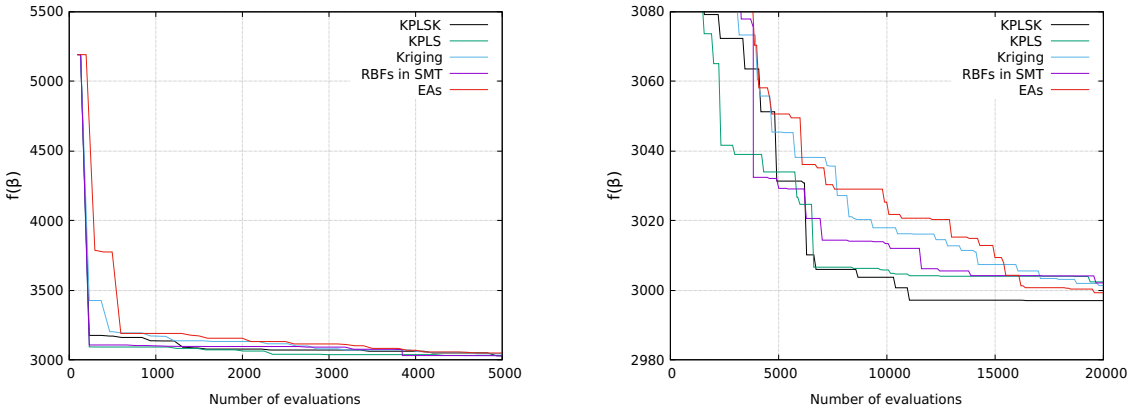


Figure C.4: Περίπτωση μειωτήρα ταχύτητας με RNG1. Σύγκριση μεταξύ του ιστορικού σύγκλισης EA και MAEA με on-line εκπαίδευση μέσω του SMT

Η σύγκλιση του KPLSK συγκρίνεται στη συνέχεια με τις on-line εκπαιδευμένες RBFs που διαθέτει ο EASY.

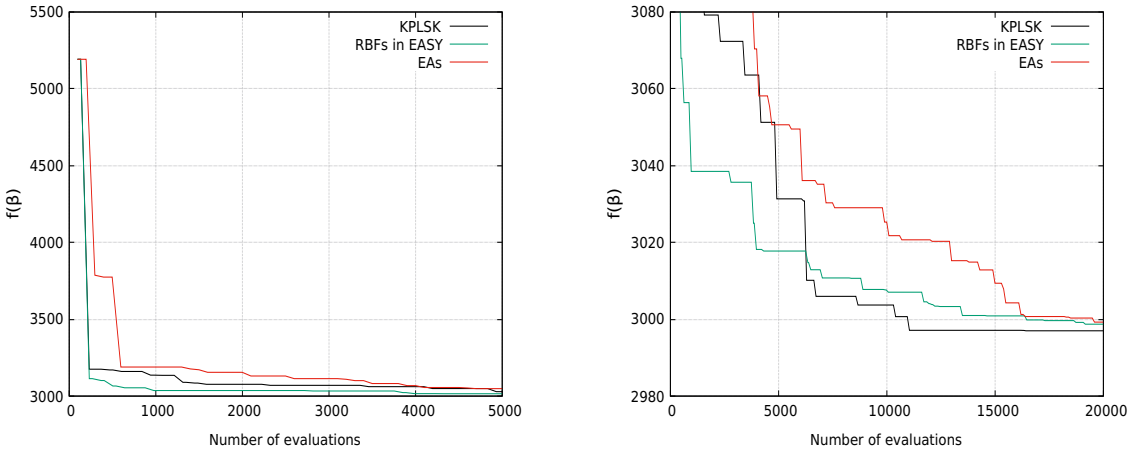


Figure C.5: Περίπτωση μειωτήρα ταχύτητας με RNG1. Σύγκριση μεταξύ του ιστορικού σύγκλισης EA και MAEA με on-line εκπαιδευμένων μεταμοντέλων

4. ***Περαιτέρω ανάλυση των δύο εφαρμογών ενός στόχου***
   Για την λήψη ορθότερης απόφασης στην επιλογή της καταλληλότερης μεθόδου βελτιστοποίησης συμπεριλήφθηκε στην ανάλυση ο συνολικός αριθμός κλήσεων του PSM και των μεταμοντέλων, όπου το υπολογιστικό κόστος του δεύτερου είναι αμελητέο σε σχέση με το πρώτο.

Μέσο αποτέλεσμα						
	Συγκολλητή δοκός	$\overline{\mathbf{n}}_{\mathbf{PSM}}$	$\overline{\mathbf{n}}_{\mathbf{meta}}$	**Μειωτήρας ταχύτητας**	$\overline{\mathbf{n}}_{\mathbf{PSM}}$	$\overline{\mathbf{n}}_{\mathbf{meta}}$
MAEAs, on-line, SMT	2.54	10000	11579	3002.68	20000	22792
MAEAs, on-line, EASY	2.53	10000	14422	3005.46	20000	24775
EAs	2.59	10000	-	3004.34	20000	-
MAEAs, off-line, SMT	3.12	388	23064	3006.01	151	18239

Table C.3: Σύγκριση μεταξύ των τελικών αποτελεσμάτων

Από τον ανώτερο πίνακα διαφαίνεται πως η βελτιστοποίηση με MAEAs με χρήση on-line εκπαιδευμένων RBFs μέσω του EASY είναι η πιο αποδοτική μέθοδος σε προβλήματα ψευδο-μηχανικής χαμηλού υπολογιστικού κόστους.

- **Βελτιστοποίηση μορφής στη NACA 4318**

Η ίδια διαδικασία επαναλαμβάνεται, αυτή τη φορά στη βελτιστοποίηση μορφής μιας δισδιάστατης NACA 4318. Αυτό το πρόβλημα βελτιστοποίησης είναι υψηλού υπολογιστικού κόστους, αφού απαιτείται η επίλυση των εξισώσεων RANS για μόνιμη και συμπιεστή ροή. Για την επίλυση γίνεται χρήση του CFD επιλύτη PUMA που αναπτύχθηκε από τη ΜΠΥΡΒ/ΕΜΠ. Η μελέτη επικεντρώνεται στη βελτιστοποίηση της γεωμετρίας της εν λόγω αεροτομής με αεροδυναμικά κριτήρια. Συγκεκριμένα, μελετάται η βελτιστοποίηση της αεροτομής σε συνθήκες απογείωσης και ευθείας πτήσης, από τις οποίες προέκυψαν τα ακόλουθα αποτελέσματα.

1. ***Βελτιστοποίηση δύο στόχων σε συνθήκες απογείωσης***

   Στόχοι τις βελτιστοποίησης είναι η μείωση της παραγόμενης οπισθέλκουσας D και αύξηση της παραγόμενης άνωσης σε συνθήκες απογείωσης με ταχύτητα $U = 51m/s$, γωνία πρόσπτωσης $\alpha = 10^o$ και συνθήκες αέρα:

Ιδιότητες ρευστού σε υψόμετρο h = 0 m			
	$\boldsymbol{\rho}$ **[kg/m^3]**	**p [bar]**	**T [K]**
Αέρας	1.225	1.01325	288

Table C.4: Ιδιότητες ρευστού σε μηδενικό υψόμετρο

Η βελτιστοποίηση επαναλήφθηκε για 3 τιμές RNG. EAs και MAEAs με on-line εκπαίδευση πραγματοποίησαν 400 CFD αξιολογήσεις, ενώ η βελτιστοποίηση με MAEAs off-line εκπαίδευση συνέκλινε μετά από 1000 εκτιμήσεις ανά κύκλο.

(a) RNG2 *(b) RNG3*

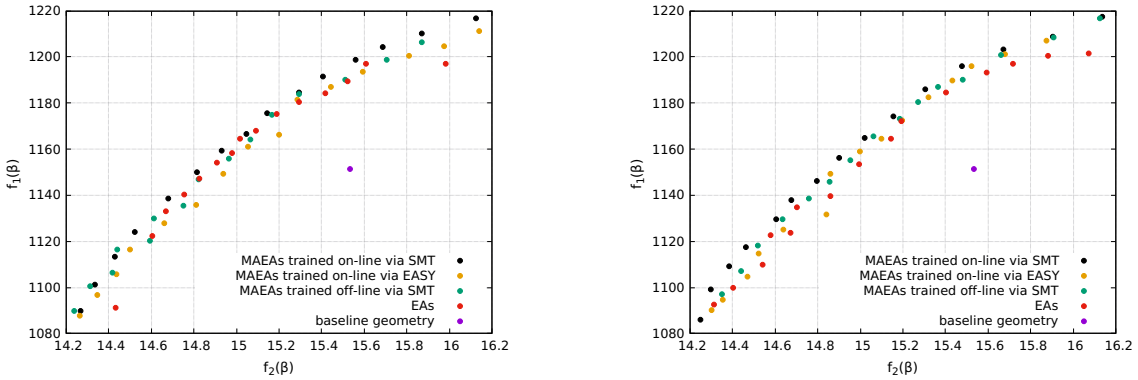


Figure C.6: Βελτιστοποίηση της NACA 4318 σε συνθήκες απογείωσης. Σύγκριση μεταξύ των μετώπων Pareto 15 μη-κυριαρχούμενων λύσεων που προέκυψαν από την εφαρμογή EAs, MAEA με off-line και on-line εκπαιδευμένου KPLS και MAEA με on-line εκπαιδευμένων RBFs μέσω του EASY για 400 CFD αξιολογήσεις

Ακολουθεί χρήση της μεθόδου δείκτη υπερόγκου για το προσδιορισμό του κυρίαρχου μετώπου Pareto.

Δείκτης υπερόγκου $\mathbf{H}(\mathcal{F})$			
	RNG1	**RNG2**	**RNG3**
MAEAs, on-line, SMT	274.588	222.513	224.786
MAEAs, on-line, EASY	257.987	206.800	211.937
MAEAs, off-line, SMT	252.866	213.701	216.461
EAs	257.512	200.114	204.289

Table C.5: Βελτιστοποίηση της NACA 4318 σε συνθήκες απογείωσης. Σύγκριση των δεικτών υπερόγκου των μετώπων Pareto που σχηματίστηκαν από τη χρήση διάφορων μεθόδων βελτιστοποίησης

2. ***Βελτιστοποίηση ενός στόχου σε συνθήκες απογείωσης***

Στη δεύτερη περίπτωση ως στόχος της βελτιστοποίησης τίθεται η μεγιστοποίηση της άνωσης σε συνθήκες απογείωσης με ένα περιορισμό που αφορά την αύξηση της οπισθέλκουσας της αρχικής αεροτομής $D_{bsl}=15.53$ N:

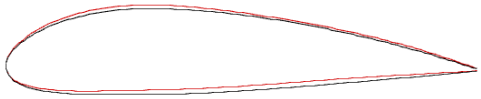
$$\begin{aligned} max \;\; & f(\vec{\beta}) = L \\ \text{subject to} \;\; & c_1(\vec{\beta}) = D - 1.08 D_{bsl} \leq 0 \end{aligned} \tag{C.11}$$

Βελτιστοποίηση της NACA 4318 σε συνθήκες απογείωσης			
	Μέσο $\vec{\mathbf{f}}$	$\bar{\mathbf{n}}_{\mathbf{PSM}}$	$\bar{\mathbf{n}}_{\mathbf{meta}}$
MAEAs, on-line, SMT	1222.33	400	2988
MAEAs, on-line, EASY	1221.90	400	4000
MAEAs, off-line, SMT	11221.27	81	1000
EAs	1211.54	400	-

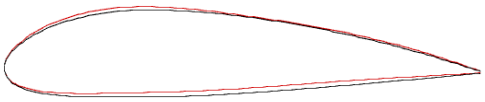
Table C.6: Σύγκριση μεταξύ των εφαρμοσμένων μεθόδων βελτιστοποίησης

(a) MAEAs, on-line, SMT, $L = 1222.52\ N$ *and* $D = 16.54\ N$

(b) MAEAs, on-line, EASY, $L = 1222.06\ N$ *and* $D = 16.44\ N$



(c) MAEAs, off-line, SMT, $L = 1221.02\ N$ *and* $D = 16.29\ N$



(d) EAs, $L = 1211.35\ N$ *and* $D = 16.25\ N$

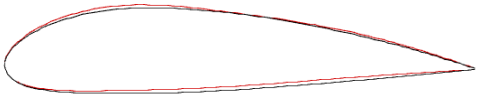
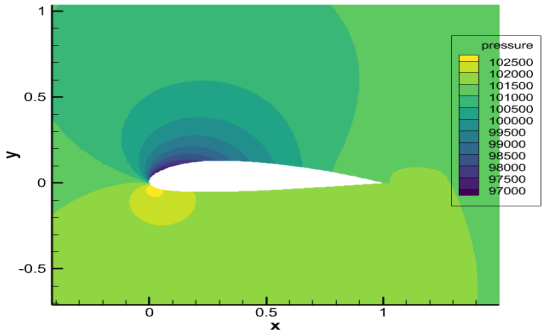


Figure C.7: Βελτιστοποίηση της NACA 4318 σε συνθήκες απογείωσης με RNG3. Σύγκριση μεταξύ των βέλτιστων αποτελεσμάτων.

(a) Αρχική γεωμετρία



(b) Βελτιστοποιημένη γεωμετρία

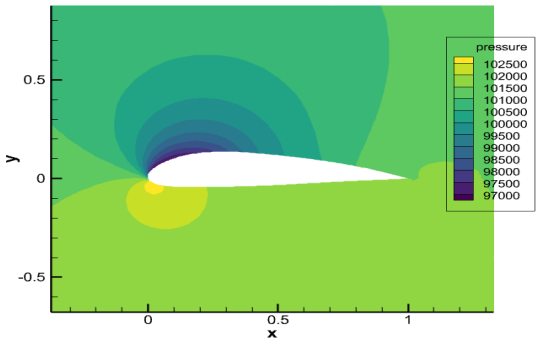


Figure C.8: Σύγκριση μεταξύ του πεδίου πίεσης αρχικής και της βελτιστοποιημένης γεωμετρίας με χρήση MAEA με off-line εκπαίδευση

3. ***Βελτιστοποίηση ενός στόχου σε συνθήκες ευθείας πτήσης***

Στη τελευταία περίπτωση ως στόχος της βελτιστοποίησης τίθεται η ελαχιστοποίηση της οπισθέλκουσας σε συνθήκες ευθείας πτήσης με ένα περιορισμό που αφορά την μείωση της άνωσης της αρχικής αεροτομής $L_{bsl} = 1123.81$ N:

$$\begin{aligned} min\ \ & f(\vec{\beta}) = D \\ \text{subject to}\ \ & c_1(\vec{\beta}) = L - 0.92 L_{bsl} \geq 0 \end{aligned} \tag{C.12}$$

Η βελτιστοποίηση πραγματοποιείται σε συνθήκες ευθείας πτήσης με ταχύτητα $U = 206.64 m/s$, γωνία πρόσπτωσης $\alpha = 2^o$ και συνθήκες αέρα:

Fluid properties at h = 11000 m			
	ρ [kg/m³]	**p [bar]**	**T [K]**
Αέρας	0.364805	0.227	216.8

Table C.7: Ιδιότητες ρευστού σε ύψος ευθείας πτήσης

Βελτιστοποίηση της NACA 4318 σε συνθήκες ευθείας πτήσης			
	Μέσο $\vec{f}$	$\bar{n}_{PSM}$	$\bar{n}_{meta}$
MAEAs, on-line, SMT	148.86	400	2988
MAEAs, on-line, EASY	149.57	400	4000
MAEAs, off-line, SMT	149.60	87	2000
EAs	154.63	400	-

Table C.8: Σύγκριση μεταξύ των εφαρμοσμένων μεθόδων βελτιστοποίησης

(a) MAEAs, on-line, SMT, $D = 148.87$ *N and* $D = 1373.12$ *N*

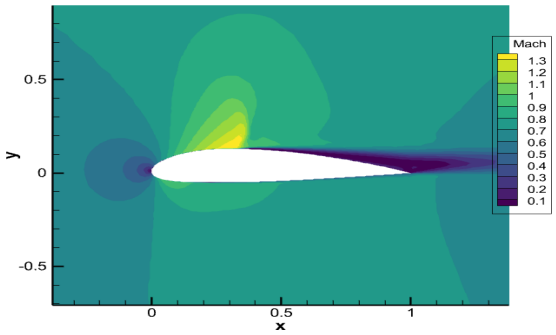
(b) MAEAs, on-line, EASY, $D = 149.24$ *N and* $D = 1383.62$ *N*

(c) MAEAs, off-line, SMT, $D = 150.06$ *N and* $L = 1367.56$ *N*

(d) EAs, $D = 155.25$ *N and* $L = 1429.83N$

Figure C.9: Βελτιστοποίηση της NACA 4318 σε συνθήκες ευθείας πτήσης με RNG3. Σύγκριση μεταξύ των βέλτιστων αποτελεσμάτων.

(a) Αρχική γεωμετρία



(b) Βελτιστοποιημένη γεωμετρία

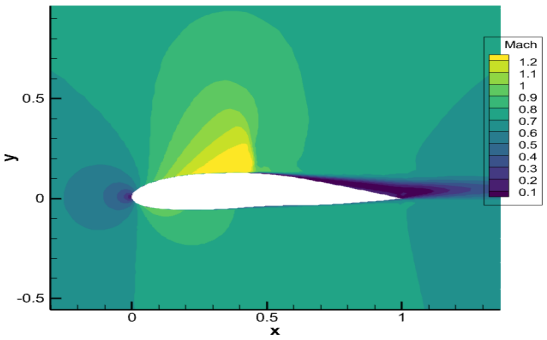


Figure C.10: Σύγκριση μεταξύ του πεδίου Mach αρχικής και της βελτιστοποιημένης γεωμετρίας με χρήση MAEA με off-line εκπαίδευση

- **Σύνοψη-Συμπεράσματα**

Σε αυτή διπλωματική εργασία μελετήθηκε η βέλτιστη χρήση εξελικτικών αλγορίθμων υποβοηθούμενων από μεταμοντέλα σε εφαρμογές χαμηλού (εφαρμογές ψευδοβελτιστοποίησης) και υψηλού (βελτιστοποίηση αεροτομής με αεροδυναμικά χαρακτηριστικά) υπολογιστικού κόστους. Βρέθηκε ότι στην πρώτη περίπτωση, η χρήση MAEAs με on-line εκπαιδευμένων RBFs που διαθέτει ο EASY έχει το μικρότερο αντίκτυπο στο υπολογιστικό κόστος σε συνδυασμό με γρήγορη σύγκλιση της βελτιστοποίησης. Η επιλογή αυτή βασίζεται τόσο στην κακή εκπαίδευση (off-line) που συναντάται σε προβλήματα πολλών στόχων, όσο και στην επιβάρυνση της βελτιστοποίησης λόγω της χρήσης της αργής Python.

Στην δεύτερη περίπτωση που απαιτείται η επίλυση των RANS εξισώσεων και το υπολογιστικό κόστος είναι υψηλό, η εφαρμογή των MAEA με χρήση off-line και on-line εκπαιδευμένων μεταμοντέλων μέσω του SMT οδηγεί σε $60 - 80\%$ μείωση του υπολογιστικού κόστους. Συνεπώς, σε αεροδυναμικές εφαρμογές υψηλού υπολογιστικού κόστους και λίγων περιορισμών ενδείκνυται η χρήση εξωτερικών μεταμοντέλων.

Με βάση τις εφαρμογές που εξετάστηκαν σε αυτή τη διπλωματική εργασία, προκύπτει η διαφαινόμενη υπεροχή του KPLS μεταξύ τόσο των εξωτερικών όσο και των εσωτερικών μεταμοντέλων που χρησιμοποιήθηκαν. Ακόμα, η βελτιστοποίηση με χρήση MAEAs με on-line εκπαίδευση διαφαίνεται ως η επικρατέστερη μέθοδος βελτιστοποίησης.

Βιβλιογραφία

1. M.A. Bouhlel, J.T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J.R.R.A. Martins. A Python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135, 2019.

2. EASY - The Evolutionary Algorithms SYstem Home, 2012. Retrieved from http://velos0.ltt.mech.ntua.gr/EASY

3. M.J.D. Powell. *The Theory of Radial Basis Function Approximation*. Oxford University Press, pp. 105-210, 1992.

4. A. Keane, A. Forrester, and A. Sóbester. *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 1st edition, 2008.

5. M.K. Karakasis and K.C. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8):941–957, 2006.

6. M.A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier. Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53(5): 935-952, 2016.

7. M.A. Bouhlel, N. Bartoli, J. Morlier, and A. Otsmane. An improved approach for estimating the hyperparameters of the kriging model for high-dimensional problems through the partial least squares method. *Mathematical Problems in Engineering*, 2016.

8. K. Tsiakas. Development of shape parameterization techniques, a flow solver and its adjoint, for optimization on GPUs. Turbomachinery and external aerodynamics applications. PhD thesis, Laboratory of Thermal Turbomachines, NTUA, Athens, 2019.

9. Κ. Χ. Γιαννάκογλου. Μέθοδοι Βελτιστοποίησης στην Αεροδυναμική. Πανεπιστημιακές Εκδόσεις Ε.Μ.Π., Αθήνα, σελίδες 125-130, 2006.

10. R. Mead, S. Gilmour, and A. Mead. *Statistical principles for the design of experiments*. Cambridge University Press, 1st ed., pp. 233-271, 2012.

11. I. Ronald. Latin Hypercube Sampling. *ResearchGate*, January 1999.

12. M. Johnson, L. Moore and D. Ylvisaker. Minimax and maximin distance designs, *Journal of Statistical Planning and Inference*, 26(2): 131-148, 1990.

13. M.C. Shewry and H.P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2): 165-170, 1987.

14. R. Jin , W. Chen, and A. Sudjianto. An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference*, 134(1): 268-287, 2005.

15. D.C. Montgomery. *Design and Analysis of Experiments*. Wiley, 9th edition, pp. 179-229, 2017.

16. T. Ray and K. M. Liew. A Swarm Metaphor for Multiobjective Design Optimization. *Engineering Optimization* 34: 141-153, 2002.

17. S.S. Rao. *Engineering Optimization: Theory and Practice*. Wiley, 5th ed., pp. 434-435, 2020.