POLITECNICO DI TORINO

Master of Science in Aerospace Engineering

Master Thesis

# Aerodynamic Design Optimization of a Rotor Blade-Sleeve Section using a Surrogate-Based Approach


**Supervisors**
Renzo Arina (Politecnico di Torino)
Patrick Pölzlbauer (Technische Universität München)

**Candidate**
Giuseppe Chiapparino

December 2018

**Abstract**

The ever increasing air traffic demands for a reduction of aircraft's emissions: this could be achieved by reducing the fuel consumption, which is highly affected by the drag produced by the aircraft itself. In helicopter aerodynamics, one of the main drag contributors is the rotor-head. As part of the Clean Sky 2 programme, TUM's FURADO Project aims at optimizing the aerodynamic design of a full fairing rotor-head via CFD simulations. The optimization task is performed using a genetic algorithm, a very effective but expensive method. In this thesis, a surrogate-based approach is proposed, so as to perfom the aerodynamic design optimization of a rotor blade-sleeve section belonging to the Airbus' RACER demonstrator. A Bayesian surrogate model, specifically a Gaussian Process, is used to partially replace the CFD solver, thus increasing the computational efficiency of the whole optimization process. The procedure is directly compared to a standard fully CFD-based approach. Results reveal that the proposed method, although showing some drawback in designs constraints handling, has a good potential to obtain a substantial reduction of the computational effort, nonetheless being able to provide promising designs within a few number of optimization's iterations.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Clean Sky 2 Programme and FURADO Project

Over the last decades the aircraft industry has grown constantly, leading to a steady increase of air traffic ([1]), not only over long and medium distances but also over short ranges. This trend is set to continue and strengthen, nearly doubling the number of passengers in the next 20 years ([2],[3]). In parallel, also the emissions and noise pollution produced by aircraft are likely to increase further. Therefore, it is not surprising that recently many organizations decided to tackle this very important problem.

Clean Sky 2 (CS2) is a programme managed by Clean Sky Joint Undertaking (CSJU), a partnership between the European Commission and the European aeronautic industry. Successor of the Clean Sky programme (CS), CS2 was launched in 2014 and aims at the development of state-of-the-art technologies to achieve its major objectives by 2024: a reduction of CO2 and NOX emissions and a decrease in noise produced by aircraft ([4]).


(a) (b)

Figure 1.1. Clean Sky 2 Logo (a) and RACER demonstrator configuration (b), as revealed at the 2017 Paris Air Show ([5]).

Such a technology is developed and tested through Innovative Aircraft Demonstrator Platforms (IADPs) and Integrated Demonstrator Technologies (IDTs), the results of an effort of more than 200 participating entities spread across 24 countries. Among IADPs emerges the Rapid And Cost-Effective Rotorcraft (or simply RACER, [6]) demonstrator developed by Airbus Helicopters. Built upon the predecessor Eurocopter $X^3$ technology demonstrator, the RACER is intended to help in refining the aerodynamic configuration of a high-speed efficient compound helicopter, mixing the

efficiency in forward flight provided by fixed-wings configuration with the energy-efficient VTOL (Vertical Take-Off and Landing) capability of a classic helicopter provided by a dedicated main rotor. The demonstrator's design is optimized for a cruise speed around 400 kph, well beyond the nowadays performance of a civil helicopter.

In the context of the aerodynamic design and optimization of the RACER demonstrator and as a part of the Clean Sky 2 programme, TUM's FURADO Project (Full Fairing Rotor Head Aerodynamic Design Optimization, [7]) plays a remarkable role. The project aims at numerical design optimization and flow analysis of a semi-watertight full fairing rotor head, minimizing drag and maximizing efficiency of a compound helicopter configuration. The 48-months lasting project started in 2016 and is composed of 6 work packages: Project Management, Dissemination and Communication (WP1), Method Development (WP2), Preliminary Aerodynamic Optimization (WP3), Refined Aerodynamic Optimization (WP4), Fairing Mechanical Design (WP5), Data Exchange And Synthesis (WP6).

## 1.2 Objective of the thesis

The main purpose of the thesis is to perform an aerodynamic surrogate-based optimization of a rotor blade-sleeve section belonging to the RACER demonstrator (Fig. 1.2). Specifically, maximizing the efficiency for the advancing blade case and minimizing drag for the retreating blade case are the two objecive functions of the optimization.

The use of a surrogate model, in order to partially replace expensive Computational Fluid Dynamics (CFD) simulations, should shorten the time required to get to optimal solutions, especially in comparison to a traditional optimization approach entirely based on CFD computations.

In particular, the strategy follows two main steps: an initial surrogate-based optimization, which should produce some already nearly optimal solutions, followed by a traditional multi-objective optimization via genetic algorithms directly involving complete CFD simulations.

The idea is that the first surrogate-based step could quickly drive the search for optimal solutions toward promising regions of the design space without spending too much time in this initial search. Thus, the expensive optimization routine can directly start from an advantageous and promising position.

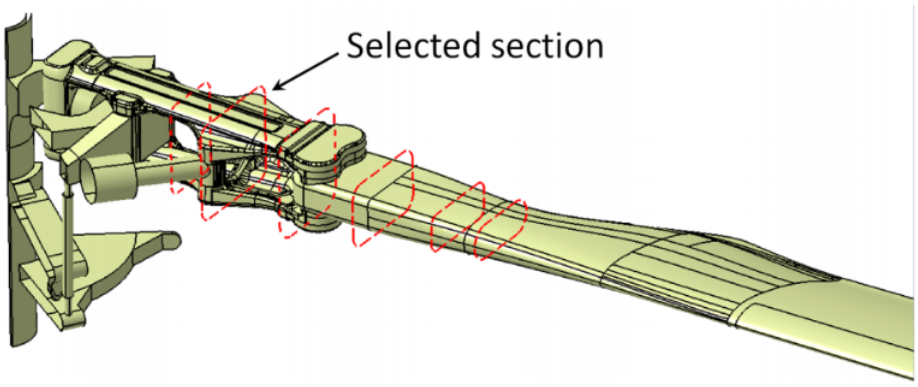



Figure 1.2. RACER's blade-sleeve section under investigation ([26]).

## 1.3 Outline of the thesis

After this brief introduction, the main theoretical aspects of the study are described in Chapter 2, examining the surrogate-based approach for optimization problems, the Bayesian statistics and Gaussian Processes.

Chapter 3 presents the actual problem under investigation, describing the object of the optimization, the procedure leading to the solution, the tools used during the study (software and numerical setup) and some test cases.

In Chapter 4, surrogate-based approach results are analyzed and compared to those of a more traditional procedure, so as to assess whether any benefits can be gained by the use of surrogate models.

Finally, conclusions are drawn and an outlook on further possible developements of the work are proposed in Chapter 5.

# Chapter 2

# Theory

In Chapter 2, the main theoretical aspects of the work are discussed. In particular, they include the surrogate-based optimization method and its implementation through the use of a Bayesian surrogate model, the Gaussian Process emulator.

## 2.1 Surrogate-based optimization

### 2.1.1 General optimization problem

The standard form of a continuous constrained optimization problem is the following ([8], pp. 123-125):

$$\begin{aligned} \text{minimize:} \quad & \boldsymbol{f}(\boldsymbol{x}) \\ \text{subject to:} \quad & \boldsymbol{g_L} \leq \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{g_U} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h_t} \\ & \boldsymbol{a_L} \leq \boldsymbol{A_i}\boldsymbol{x} \leq \boldsymbol{a_U} \\ & \boldsymbol{A_e}\boldsymbol{x} = \boldsymbol{a_t} \\ \text{for:} \quad & \boldsymbol{x_L} \leq \boldsymbol{x} \leq \boldsymbol{x_U}. \end{aligned} \tag{2.1}$$

Here, $\boldsymbol{f}(\boldsymbol{x})$ is the function to be optimized: for historical reasons optimization problems are always treated as minimization problems and if an objective is to be maximized, the solver will minimize its negative value, instead. If $\boldsymbol{f}(\boldsymbol{x})$ is a scalar function, the problem is a single-objective optimization problem (SOP). If, instead, the function $\boldsymbol{f}(\boldsymbol{x})$ is vectorial, then the problem is called multi-objective optimization problem (MOP). Functions $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ define the inequality and equality nonlinear constraints, respectively. The two matrices $\boldsymbol{A_i}$ and $\boldsymbol{A_e}$ are the coefficient matrices for the linear systems describing the inequality and equality linear constraints, respectively. Inequality constraints could be also transformed to make them "one-sided" (for example, $\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}$) instead of "two-sided". Finally, $\boldsymbol{x_L}$ and $\boldsymbol{x_U}$ are the boundaries of the design space.

The presence of constraints allows to tell feasible design points (satisfying all constraints) from infeasible design points. If a problem contains at least one nonlinear objective function or constraint, it is called nolinear programming problem (NLP).

### 2.1.2 Multi-objective optimization: Pareto Front

In general, when considering a multi-objective optimization, there is not just a single solution capable of optimizing all the given objectives at once. Rather, there is a certain number of optimal

solutions defined as Pareto Front. Each optimal solution, called Pareto optimal solution, is such that it is not possible to make any Pareto improvement, which means that it is not possible to improve one of the objective functions without worsening at least one of the others. Inside the design space, the set of all these non-dominated solutions is called Pareto Optimal Set. All the other feasible non-optimal solutions represent, instead, the so-called dominated solutions.

The solution which perfectly optimizes all the objective functions at the same time is called "utopia point" and, as the name suggests, it is (generally) found in the infeasible region of the design space. Normally this condition takes place when the two (or more) objective functions are competing objectives, like performance and cost, so it is not possible to optimize indefinitely both at the same time (Fig. 2.1). Appendix A contains the mathematical formulations of what explained in this subsection.

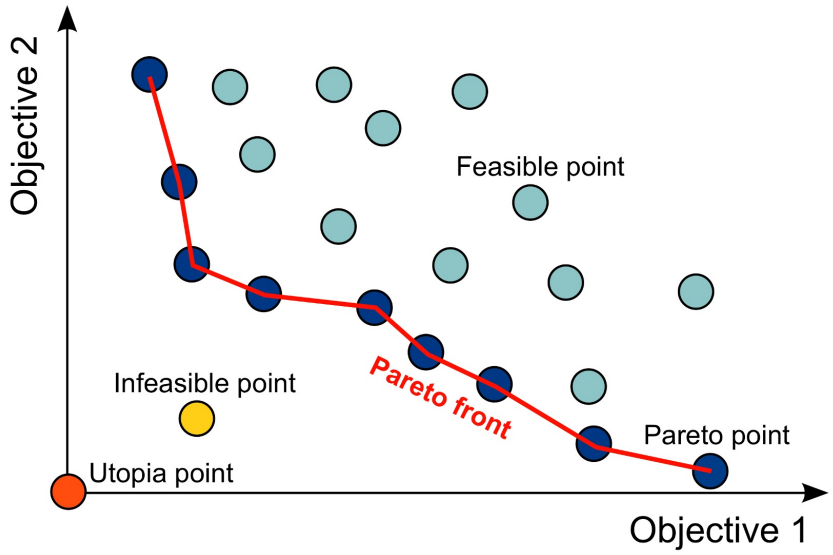



Figure 2.1. Example of a Pareto Front ([9]) for the minimization of 2 competing functions, where minimizing one objective often leads to increase the other; pale blue dots are the dominated solutions.

### 2.1.3 Optimization algorithms

A key point in solving optimization problems is the choice of the algorithm. A general distinction is made between gradient-based and gradient-free algorithms. The formers use information from first order gradients (Jacobian matrix) and/or second order gradients (Hessian matrix) to drive the search for optimal solutions, while the latters do not. However, it is quite common that the information about gradients is not available, as in problems characterized by: non-differentiable functions, discrete or mixed discrete-continuous variables, non-convex or disconnected design space. Moreover, they show limitations also when dealing with multi-modal problems, where more local optima are present ([10]).

In all these cases, the derivative-free approach is the one to follow.

A popular class of algorithms are the Genetic Algorithms. They are based on Darwin's theory of natural selection, where the fittest design points survive and are able to reproduce in successive generations. Each design point is characterized by a combination of the design parameters, just like each biological individual is characterized by its DNA sequence. Starting from a random initial population, the designs resulting in lower objective functions (the fittest ones) are used to produce

the next generations of designs in an iterative process, adopting the mathematical equivalent of phenomena as mutations, breeding and crossover.

These algorithms are easily implemented in automated processes and can be quite effective: for example they were used in the preparation of the NASA's Space Technology mission (ST5), where the shape of an evolved antenna was designed with evolutionary algorithms ([11]). Nevertheless, they often require a huge amount of function evaluations; therefore, the associated computational cost, especially when using complex solvers (as CFD solvers), might be prohibitive.

A possible solution to increase the computational efficiency is represented by the surrogate-based approach.

### 2.1.4 Surrogate-based optimization approach

Surrogate-based optimization is a class of methodologies which adopt a surrogate model (also called emulator or response surface) to find the global or local optimal solutions of a problem.

When solving a surrogate-based optimization problem, the general formulation of Eq. (2.1) can be rewritten as:

$$\begin{aligned} \text{minimize:} \quad & \tilde{\boldsymbol{f}}(\boldsymbol{x}) \\ \text{subject to:} \quad & \boldsymbol{g_L} \leq \tilde{\boldsymbol{g}}(\boldsymbol{x}) \leq \boldsymbol{g_U} \\ & \tilde{\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{h_t} \\ & \boldsymbol{a_L} \leq \boldsymbol{A_i x} \leq \boldsymbol{a_U} \\ & \boldsymbol{A_e x} = \boldsymbol{a_t} \\ \text{for:} \quad & \boldsymbol{x_L} \leq \boldsymbol{x} \leq \boldsymbol{x_U}. \end{aligned} \tag{2.2}$$

In this new formulation, the true functions have been substituted by their corresponding surrogates (denoted by the tilde symbol in Eq. (2.2)). It is interesting to note that also nonlinear constraints have been substitued with emulators.

A surrogate model is an approximated, computationally cheap model with unknown accuracy. It is used to estimate metrics of interests replacing a high-fidelity, costly model like a complete CFD solver (as in this thesis) or to directly model the outcome of a real process like a complex experiment. Surrogate models are often employed in those situations where running the original model takes too much time and computational resources, as in Computational Fluid Dynamics analysis ([12]) and Computational Structural Mechanics, or where performing several time the real experiment is too expensive or even impossible.

Performing the optimization on a more computationally-efficient model represents one of the main advantages of these methods. On the other hand, the emulator might not be accurate enough to find the real optimum. Therefore, it is common to adopt an iterative strategy and train the surrogate model at each iteration, in order to reach convergence to the real model. According to the optimization strategy, however, convergence might not be always guaranteed.

Generally, the model is built from a training set composed of randomly picked data points, which are first evaluated through the high-fidelity model. The design space is probed so as to maximize the amount of information gained while limiting the number of sample points: a common sampling method is the Latin Hypercube Sampling (LHS); however, also other sampling techniques are available. This first step is called Design of Experiment (DoE).

According to the problem under investigation, some emulators might prove more suitable than others. A first distinction is between global surrogate models and local surrogate models.

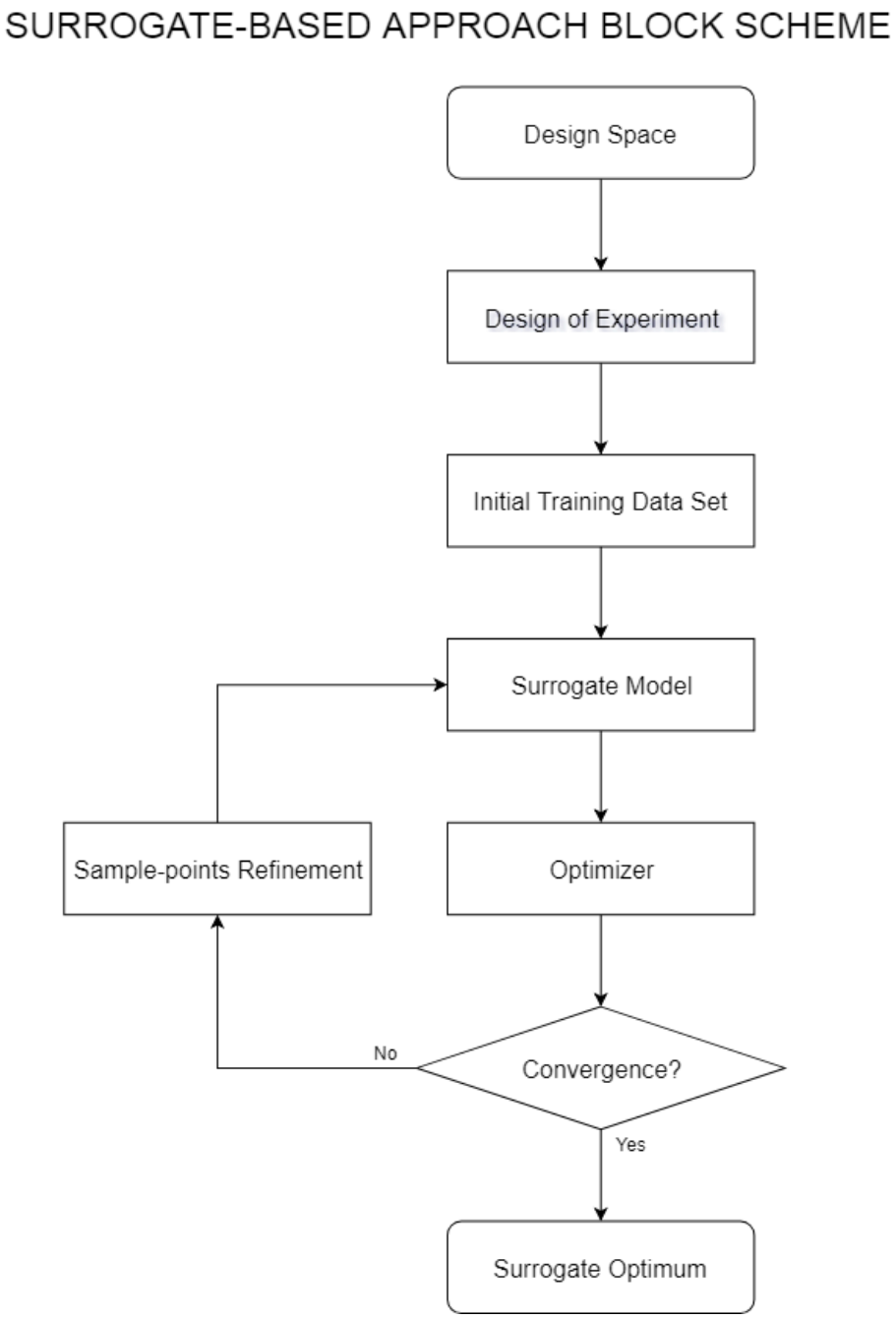



Figure 2.2. General scheme of a Surrogate-based optimization process.

The first category includes models with global support, in which the training data are spread over the complete design space range. These models should guarentee a good global approximation, capturing the global features of the response space; however, they may fail in precisely describing its local behaviour. Models like Gaussian Processes, Neural Network models, Polynomial models and others fall into this category.

The second category, instead, includes models providing a high local accuracy, usually in the proximity of a single response point. Local surrogates often require, together with the function evaluation, the knowledge of the derivatives (of first and, possibly, of second order) at the point of interest; far from that point, the accuracy rapidly decreases. An example of local surrogate model is given by the Taylor Series models.

Aside this major distinction, other characterizing properties of the different emulators are: the smoothness and differentiabilty of the response model, the capability to interpolate all the training data and, possibly, to smooth out noise, the ability to deal with the presence of multiple maxima/minima points, the minimum required number of training points and so on.

As previously mentioned, while performing a surrogate-based optimization task, it is often necessary to update and train iteratively the initial emulator. To do so, some further high-fidelity simulations are run at each iteration and their outcomes are compared with the predictions made by the model. This step is called Sample-points Refinement. There are several ways to determine the new points which are applied to refine the model. For example, once the model has been used to find the surrogate optimal solutions, a subset of these points could be evaluated on the high-fidelity model and used for the update. In case of a multi-objective optimization, these points would be a subset of the surrogate Pareto Front. An alternative is to look for designs that should maximize the amount of information gained by adding those points to the training set (Expected Improvement).

The number of model updates depends on the implemented stopping criteria, the most common of which are: maximum number of iterations, maximum number of true function evaluations and rate of change of optimal solutions between two consecutive iterations.

The way the model is updated with the new information gained during the sample-point refinement depends on the type of machine learning algorithm that has been chosen. A particularly interesting class of learning algorithms are Bayesian methods, which revolve around the application of Bayesian statistics.

Fig. 2.2 shows a simple surrogate-based optimization block scheme ([13]), in which the main steps so far described are visualized.

## 2.2 Bayesian statistics

### 2.2.1 Different definitions of probability

The "classical definition or interpretation of probability" generally refers to the following definition given by Pierre-Simon Laplace: "The probability of an event is the ratio of the number of cases favourable to it, to the number of all cases possible when nothing leads us to expect that any of these cases should occur more than any other, which renders them, for us, equally possible" (Théorie analytique des probabilités, 1812). This interpretation finds its foundation directly in the principle of indifference.

This definition is not the only possible one and several mathematicians and statisticians, well before or after the time of Laplace, had come up with different concepts and interpretations of probability. However, it is important to remark that although they imply quite different interpretations, all the statistical approaches agree on the maths of probability theory. One of the most widely used is the Frequentist definition, while in recent years, especially thanks to the increased computational power of computers, the Bayesian approach regained more and more importance in many fields, like engineering and computer sciences. This interpretation was originally proposed by Thomas Bayes in 1760s and is connected to the general definition by Laplace through the concept of prior probability.

In the Frequentist approach, the concept of probability is related to the relative frequency of occurrence of a certain event out of all the possible outcomes. Therefore, it is directly connected

with the observation of repeatable experiments: as the number of trials increases, the relative frequency is expected to converge to the true frequency of the event, defined as the limit of the relative frequency. This interpretation is objective, but on the other hand is effectively applicable only in those situations where it is possible to perform experiments, which is sometimes hard, if not impossible, to do.

To make an example, suppose to have a deck of 100 cards: these could be either red or blue cards. To asses the probability of drawing a red card from the shuffled deck applying a frequentist approach, one should first go through all the deck, count the red and the blue cards and note down their relative frequency.

In Bayesian statistics, instead, the interpretation of probability is related to the "prior belief" that one has about the occurrence of a certain event: it describes how sure one is about the outcome of the event. Therefore, this approach is subjective and is, in fact, how people usually reason. "In Bayesian statistics, probabilities are in the mind, not in the real world" ([14]). So, two different individuals may give two different probabilities to the same event. A probability of 1 means that someone is completely sure about a certain outcome; with a probability of 0.70 one might not be too much surprised if its conviction turns out to be false. However, as new information becomes available, this prior belief can be updated, being strengthened or weakened: how this happens is found in mathematical terms applying Bayes Theorem, which gives as a result the posterior probability.

For example, consider again the previous 100 cards deck and that, for sake of simplicity, only one of the following three cases is true: there are 50 red and 50 blue card, all cards are red, all cards are blue. In absence of other information, one observer may be reasonably convinced that the deck contains an even number of red and blue cards. Therefore, this observer could give a prior probability of 0.98 to this occurence. The other cases of an all-red or all-blue deck sound remote, so a probability of 0.01 would be assigned to each of them. Then, the draw starts and, after the first 10 cards, only red cards have come out. At this point the belief of the observer has changed: the most probable hypothesis is now the one of an all-red deck, while the all-blue deck hypothesis is defenitely ruled out and it also appears highly improbable to have a red-and-blue deck.

It is important to note that the prior probabilities are subjective. Hence, in the previous example, different observers could assign completely different probabilities to each of the three events. However, with the addition of new information, the posterior probabilities of different observers converge toward the same outcome.

### 2.2.2 Bayes Theorem

Before introducing the Bayes Theorem, it is helpful to recall some useful definitions.

Joint probability: it is the probability that two independent events A and B occur at the same time.

$$P(A, B) = P(A)P(B) \tag{2.3}$$

Marginal probability: it is the probability that an event occurs; if an event A has a joint probability distribution with other events (B, C and D), then the marginal probability is the probability that the event A occurs without regards for the others. It can be found from the joint distribution by summing over all possible values of the other variables.

$$P(A) = P(A, B) + P(A, C) + P(A, D) \tag{2.4}$$

Conditional probability: it is the probability that an event A occurs given that another event B has taken place before. If the two events are independent, the conditional probability of A is

just its marginal probability; otherwise it is defined as the joint probability of the two events over the marginal probability of event B:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{2.5}$$

This last definition, in particular, appears in Bayes Theorem to express the posterior probability and the likelihood function.

Bayes Theorem is an equation for probability theory that allows to update the prior (or a priori) probability according to the newly available information, obtaining the posterior (or a posteriori) probability.

Given two events A and B, the equation reads as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.6}$$

where:

- $P(A|B)$ is the conditional probability for A given B and represents the posterior probability
- $P(A)$, the marginal probability for A, represents the prior probability
- $P(B|A)$ is the conditional probability for B given A and is referred to as likelihood
- $P(B)$, the marginal probability for B, is called marginal likelihood

So, the theorem states that the new belief is proportional to the old one multiplied by the likelihood. The likelihood is a conditional probability of the observations (new information) supposing that the prior belief is actually verified. The marginal likelihood is generally used just as normalization constant and in most of the cases its value is not even important.

Let us apply the Bayes Theorem to the previous 100-cards deck example: in doing so, the so-called Bayes Box ([14], pp. 11-14), in Tab 2.1, is a useful tool. The 3 events admitted by the observer are:

- all-red deck, with prior probability $P(A)_R = 0.01$
- 50-50 deck, with prior probability $P(A)_{50} = 0.98$
- all-blue deck, with prior probability $P(A)_B = 0.01$

and the second event (in Eq. 2.6 labelled as $B$) is the draw of 10 red cards in a row from the deck. If the deck were a 50-50 deck, the probability of observing event $10red$ would be:

$$P(10red)_{50} = \frac{50}{100} \cdot \frac{49}{99} \cdot \frac{48}{98} \cdots \frac{41}{91} \simeq 0.00059342, \tag{2.7}$$

that is the likelihood for that hypothesis. Similarly for the other two hypotheses:

$$P(10red)_R = 1 \cdot 1 \cdot 1 \cdots 1 = 1 \tag{2.8}$$

$$P(10red)_B = 0 \cdot 0 \cdot 0 \cdots 0 = 0 \tag{2.9}$$

from which it is possible to get the posterior by simply multiplying each prior by the corresponding likelihood. The marginal likelihood works as a normalization constant and in this example it is given by the sum:

$$P(10red) = P(A)_R \cdot P(10red)_R + P(A)_{50} \cdot P(10red)_{50} + P(A)_B \cdot P(10red)_B \tag{2.10}$$

So, after the event $10red$ has taken place, the belief of the observer is completely turned upside down, as the probability for an all-red deck reaches nearly 95%, while the 50-50 deck probability falls to a mere 5%.

Table 2.1. Bayes Box, 100-cards deck example.

| Event | Prior $P(A)$ | Likelihood $P(10red\|A)$ | Prior × Likelihood $P(A)P(10red\|A)$ | Posterior $P(A\|10red)$ |
|---|---|---|---|---|
| All-red | 0.01 | 1 | 0.01 | 0.945 |
| 50-50 | 0.98 | 0.00059342 | 0.00058155 | 0.055 |
| All-blue | 0.01 | 0 | 0 | 0 |

The Bayes Theorem works also with probability density functions, which are functions describing the relative distribution of frequency of continuous random variables. By integrating these functions, it is possible to compute the probability of the corresponding events.

In the context of machine learning it is common to work with probability density functions rather than actual probabilities: in fact, the prior belief is often the initial hypothesis about the distribution of uncertain parameter values appearing in a model, that has to be corrected imposing the consistency with observed data. The result of this operation is again a probability density function: the posterior distribution. So, labelling A = H for 'hypothesis' and B = D for 'Data', the theorem is rewritten as:

$$p(H|D) = \frac{p(D|H)p(H)}{p(D)} \tag{2.11}$$

where now:

- $p(H)$ describes the initial hypothesis on the probability density function of the values of the uncertain parameters, before any data is observed
- the posterior probability density $p(H|D)$ describes the new distribution, consistent with observations D
- the marginal likelihood function $p(D)$ states the probability to observe the data, regardless the truth or not of H
- the likelihood function $p(D|H)$ indicates how well model's predictions are supported by the data

The likelihood function, often written as $\mathcal{L}(H; D)$, is the probability density function of the observed data D, viewed as a function of the unknown paramters. Those hypotheses that produce higher likelihoods (that is to say, if they are verified, they lead the model to predictions which are more consistent with observations) are the more probable. The most plausible hypothesis is the one which gives the Maximum Likelihood Estimate (MLE). Sometimes, the MLE is not unique, as there could be two or more hypotheses maximizing the likelihood.

The shape of the likelihood function can be chosen considering the properties of the problem under investigation. For example, one may assume that the difference between the model's

predictions, depending on the uncertain parameters, and the real observations has a Gaussian distribution, so that a Gaussian likelihood function can be used:

$$\mathcal{L}(H|D) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma_D}|}} e^{-\frac{1}{2}(\boldsymbol{r}^T [\boldsymbol{\Sigma_D}]^{-1} \boldsymbol{r})} \tag{2.12}$$

where $\boldsymbol{r} \in \mathbb{R}^n$ is the residual vector (difference between model's predictions and observed data) and $\boldsymbol{\Sigma_D} \in \mathbb{R}^{n,n}$ is the covariance matrix of the Gaussian data uncertainty, with $n$ being the number of observations.

Other common choices for the likelihood function are the binomial distribution and the beta distribution.

The marginal likelihood function, instead, plays again the role of a normalization term and does not affect the shape of the posterior distribution, which is the important information to save; moreover it is not always easy to compute mathematically. Hence, the theorem is often written and used in the form:

$$p(H|D) \propto p(D|H)p(H). \tag{2.13}$$

Unless the case of a "vague prior" (uniform prior distribution), the likelihood and the posterior distribution do not have the same shape.

The Bayesian update of a prior to a posterior distribution can be used to improve a mathematical model as soon as new evidence becomes available. Such models are therefore said to be Bayesian methods.

### 2.2.3 Bayesian methods

In machine learning, Bayesian methods are a class of algorithms that do not generate a model which tries to make the best guess prediction for new input points; rather they give a posterior predictive distribution of the new inputs. This means that, given an input, the model will produce the most probable output together with its estimated uncertainty, which depends on the current training data situation. As new information is revealed, the posterior distribution is updated, increasing the confidence in the model.

The next step is to introduce one of such methods.

## 2.3 Gaussian Process and Kriging model

### 2.3.1 Theory of Gaussian Processes

A Gaussian Process is a stochastic process such that every finite collection of its random variables has a multivariate normal distribution (or, equivalently, it is a collection of random variables such that every finite linear distribution of them is normally distributed).

When a Gaussian Process is used to build an emulator, this is often referred to as Kriging model. The name Kriging derives from Danie G. Krige, a mining engineer who pioneered such a method in the field of geostatistics to try to evaluate gold deposits starting from samples of a few boreholes.

The Gaussian Process is a Bayesian regression method for data interpolation able to create smooth surface models, whose structure does not depend on the physics of the phenomenon that it is describing. Thus, it is a non-physical model. Moreover, it is a non-parametric method, which means that it has a finite but unbounded number of parameters that grows with data, in contrast to parametric models that are built with a fixed number of parameters. Therefore, it is often used in engineering and informatics fields to build emulators of highly expensive computer models

or complex experiments (in aerospace context, for example, it was adopted in [15] to model drag coefficient of low Earth orbit satellites). The Gaussian Process allows to interpolate all training points used during the construction of the model. Nevertheless, it is also possible to deal with data affected by noise, for example when modeling outcomes of real experiments, where exactly interpolating all training points would be incorrect.

A Gaussian Process can be regarded as the generalization of Gaussian probability distribution ([16]). A single-valued random variable $x \in \mathbb{R}$ is said to have a Gaussian (or normal) distribution if its probability density function is of the type:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.14}$$

with $\mu$ and $\sigma^2$ being the mean value and the variance of the variable $x$, respectively.

Moving to a more general n-dimensional random variable $\boldsymbol{x} \in \mathbb{R}^n$, this is said to have a multivariate Gaussian distribution if:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}{2}} \tag{2.15}$$

where now $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the $n \times n$ symmetric positive semidefinite covariance matrix. The compact notation is $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

While the Gaussian distribution is a distribution over vectors, the Gaussian Process is a distribution over functions (intuitively seen as infinitely long vectors), defined by a mean function $m(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ and a covariance function $k(\boldsymbol{x}, \boldsymbol{x'}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. This can be written in short as:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x'})) \tag{2.16}$$

This means that any function drawn from a Gaussian Process is an infinitely long collection of random variables, such that its mean value is the mean function $m(\boldsymbol{x})$ and its covariance is given by $k(\boldsymbol{x}, \boldsymbol{x'})$; therefore, both the mean vector and the covariance matrix have infinite elements. This means that a Gaussian Process is of infinite dimensions. As an extension of Gaussian distributions, Gaussian Processes inherit almost all of their properties such as additivity, conditioning and marginalisation. This last property allows to actually fully define a Gaussian Process just considering any finite subset of its random variables, which has the distribution:

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \right) \tag{2.17}$$

with $n$ a finite positive integer.

Fig. 2.3 shows some functions drawn from a zero-mean Gaussian Process built on a Gaussian kernel with a characteristic lengthscale equal to 0.05 and a variance of 1.

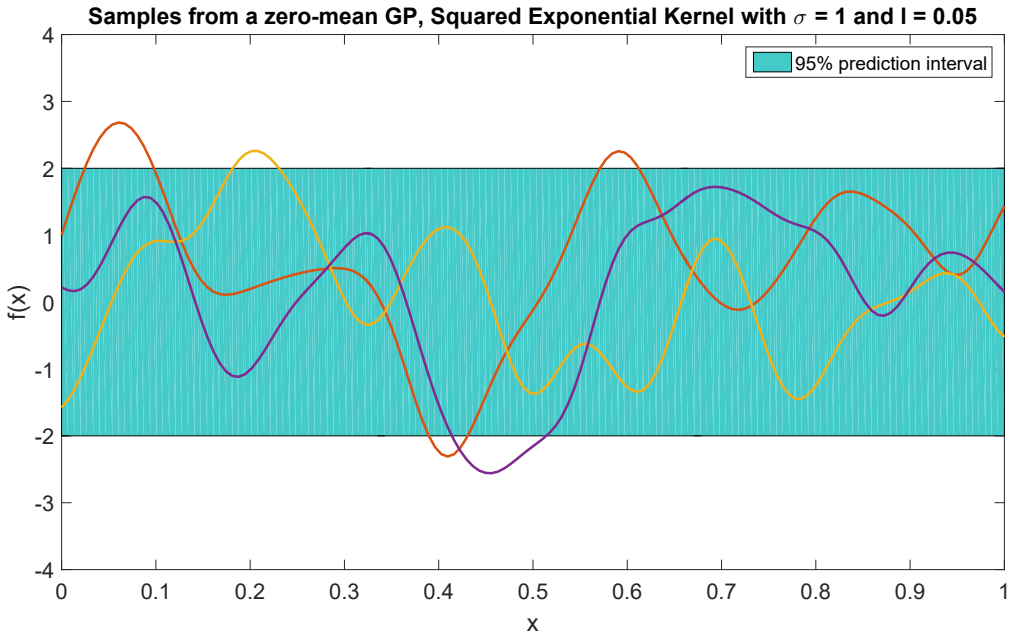



Figure 2.3. Three functions sampled from a zero-mean Gaussian Process.

Regarding the mean function $m(\boldsymbol{x})$, any real-valued function is valid, while the covariance $k(\boldsymbol{x}, \boldsymbol{x'})$ needs to be a function that creates a symmetric positive semidefinite covariance matrix, which means that it has to be a valid Mercer kernel ([17], pp. 96-97).

The choice of the kernel determines the type and the characteristics of the functions composing the Gaussian Process; several types of kernels exists, such as the Cauchy, the Matern or the Powered-Exponential kernel. The latter, when considering an exponent equal to 2, is also called Gaussian kernel (or Squared Exponential kernel), and each entry $(i, j) = (j, i)$ of the covariance matrix takes the form:

$$k(x_i, x_j) = \sigma^2 e^{-\frac{(x_i - x_j)^2}{2l^2}} \tag{2.18}$$

where $\sigma^2$ and $l$ are the variance and the characteristic lengthscale, respectively. They are also called hyperparameter, as they are parameters that live in the kernel and influence the behaviour of the Gaussian Process. The characteristic lengthscale determines how big the influence of one point on another is: the bigger the characteristic lengthscale, the stronger the correlation between farther points (and, graphically, the less oscillating the function). Modifying the variance changes the width of the prediction interval: smaller variance means smaller uncertainty, thus all the sampled functions get closer to the mean function $m(\boldsymbol{x})$.

The choice of a Squared Exponential kernel might be convenient as it is infinitely differentiable, so it produces very smooth curves. However, according to the application, other kernels could be more appropriate (see Appendix B).

Gaussian Processes are popular engineering methods, as they allow to make predictions about new inputs together with their uncertainty. Moreover, by progressively adding new data, their prior predictive distribution is updated to a posterior distribution over functions. These properties characterize Gaussian Processes as Bayesian methods.

### 2.3.2 Example of Kriging model

As an example, consider the function:

$$f(x) = \frac{x}{10} + \frac{sin(x)}{x} \tag{2.19}$$

within the range $x \in [-10,10]$ and suppose to have already observed the four points shown in Fig. 2.4.

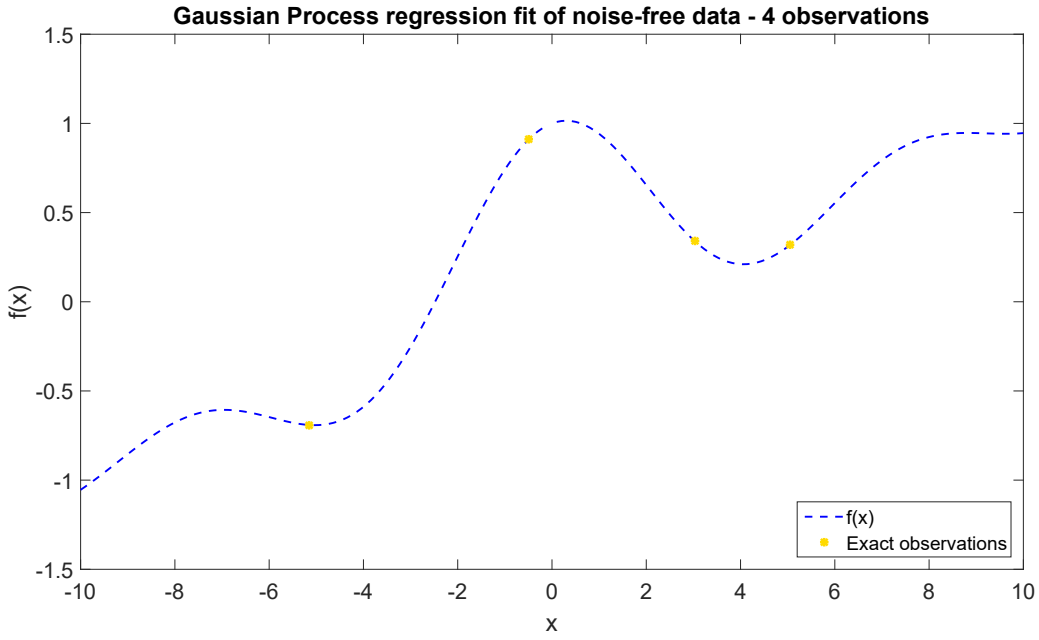



Figure 2.4. Actual function $f(x)$ and four noise-free observations.

The Kriging emulator can be written in the following way:

$$\tilde{f}(x) = g(x)^T\beta + \mathcal{GP}(0, k(x, x')) \tag{2.20}$$

therefore, it requires to define:

- the trend function, here written as the product between the trend basis functions at x and the corresponding coefficients, $g(x)^T\beta$; it captures the general features of the response space and plays the role of the mean function of the Gaussian Process
- the type of covariance function $k(x, x')$, determining the characteristics of generated functions
- the values of the hyperparameters of the kernel.

The trend function is obtained via a simple regression of the available training data and it is often a polynomial of low degree. In general, it does not go higher than the second degree, but for most of the cases even a constant trend function is acceptable. Anyway, the choice of the degree depends on the application under examination.

Then, the zero-mean Gaussian Process is used to correct the trend function, ensuring the interpolation with the provided observations and conferring zero uncertainty to those points. Here, the choice of the kernel is again driven by the specific application the emulator is used for.

Once the kernel is fixed, it is necessary to determine its hyperparameters' values. Generally, this is done by exploiting the information from training data and performing an optimization task to find the optimal values for the hyperparameters, for example via a MLE approach.

Finally, with all the parameters of the model being fixed, it is possible to use the emulator for predictions through statistical inference. The math that allows to do that is discussed in more details in Appendix C.

Back to the example of Eq. 2.19, one could use a linear trend function together with a zero-mean Gaussian Process defined by a squared exponential kernel (Eq. 2.18). The results are shown in Fig. 2.5.

The smooth, red curve is the most probable fit that interpolates the 4 observations. Although not perfectly matched by the emulator, the actual function still falls inside the 95% prediction interval (blue shaded area). This area estimates the region in which 95% of future observations will fall, according to the model and, at each point, its width is given by to the mean function plus and minus two standard deviations. This interval shrinks considerably in the neighborhood of the training points, which act like anchors for the emulator.

Adding one more observation, the Gaussian Process changes, because the posterior distribution has been updated with the new information. As shown in Fig. 2.6, the solid red line of the most probable prediction has slightly changed near the fifth training point. Moreover, the uncertainty in that region has been considerably reduced. The new point has also slightly affected the other parts of the emulator.

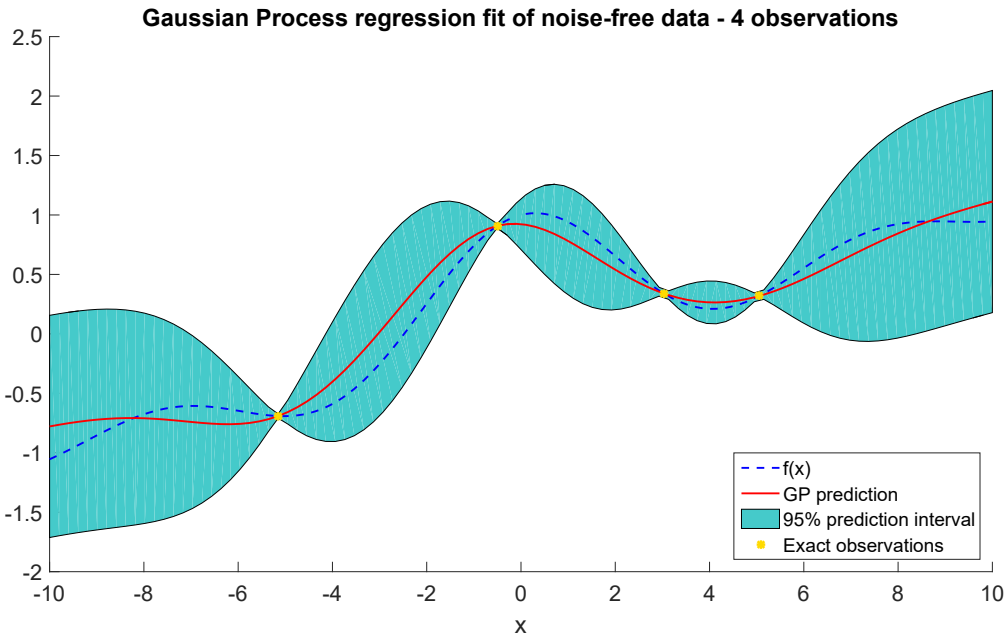



Figure 2.5. Gaussian Process model built with a linear trend function and a Gaussian kernel knowing four exact observations.

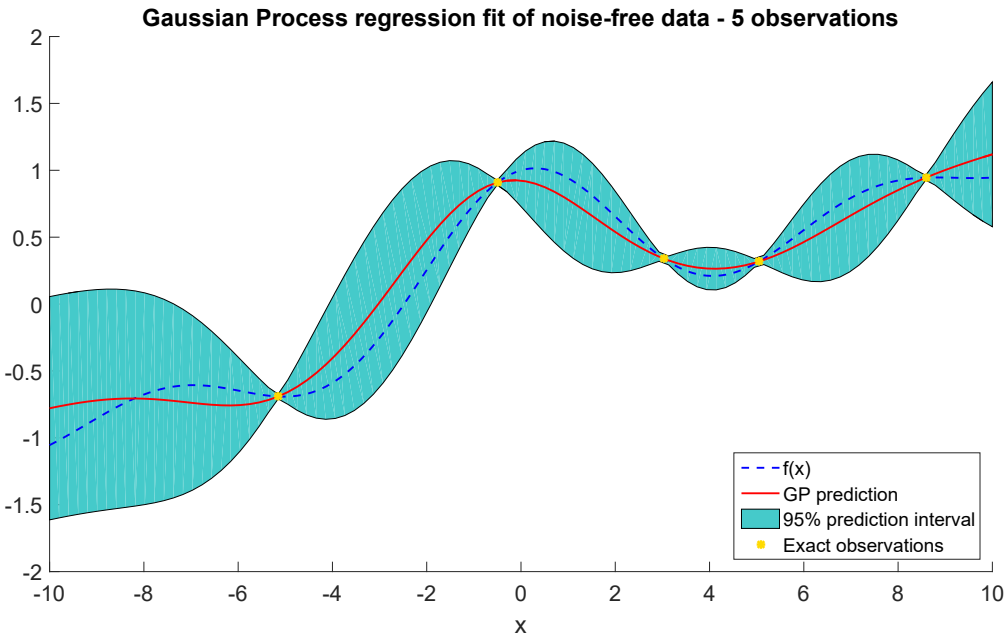



Figure 2.6. Gaussian Process model now updated with a fifth observation; this reduces the uncertainty around the region of the new point.

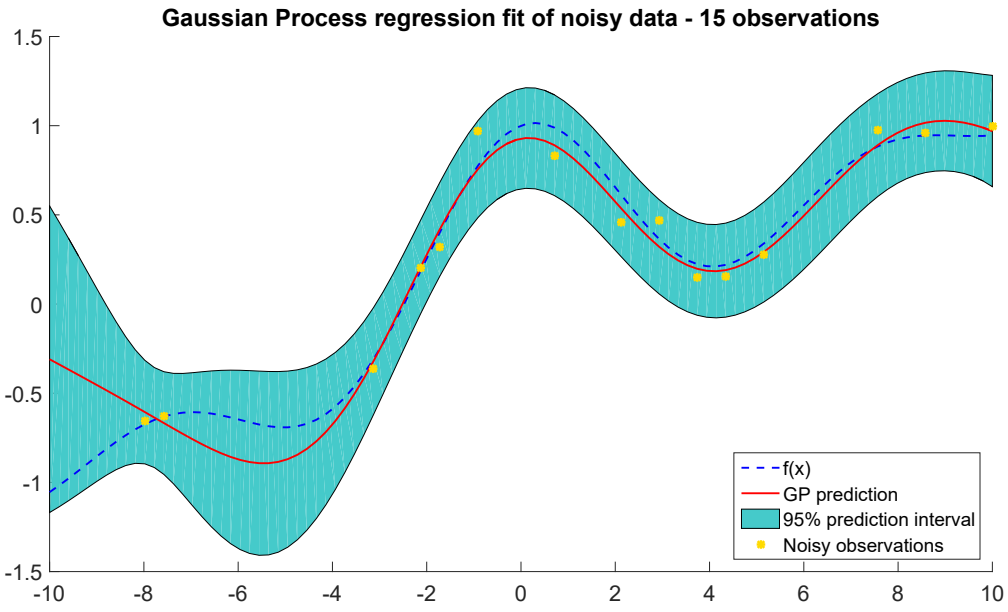



Figure 2.7. Gaussian Process emulating the same function f(x), but considering noisy data; the interpolation is not rigid anymore and the uncertainty does not vanish in correspondence of the observations.

So far, all data have been considered to be noise-free: thus an exact interpolation of all training points is desirable. However, in case of observations disturbed by noise, it is still possible to prevent overfitting.

As Fig. 2.7 shows, the matching between the actual function and the emulator is now looser and the prediction interval does not reduce near the observations; nevertheless, it is still able to enclose $f(x)$.

In the previous example, the design space was simply 1D; however, the whole process can be also applied to higher dimensional input spaces, composed of the n-tuples $\boldsymbol{x} = (x_1, \ldots, x_n)$. In this case, kernel functions change accordingly. For example, the Squared Exponential Kernel takes the form:

$$k(x_i, x_j) = \sigma^2 e^{-\frac{1}{2}\sum_{k=0}^{n} \frac{(x_{i_k} - x_{j_k})^2}{l_k^2}} \tag{2.21}$$

where, now, $n$ different characteristic lengths $l_k$ are defined, one for each direction inside the design space. Fig. 2.8 shows a sample from a Gaussian Process built on a bidimensional input space $\boldsymbol{x} = (x_1, x_2)$, with two different values for $l_1$ and $l_2$ giving a more oscillating behaviour (weaker correlation) along the $x_2$ direction.

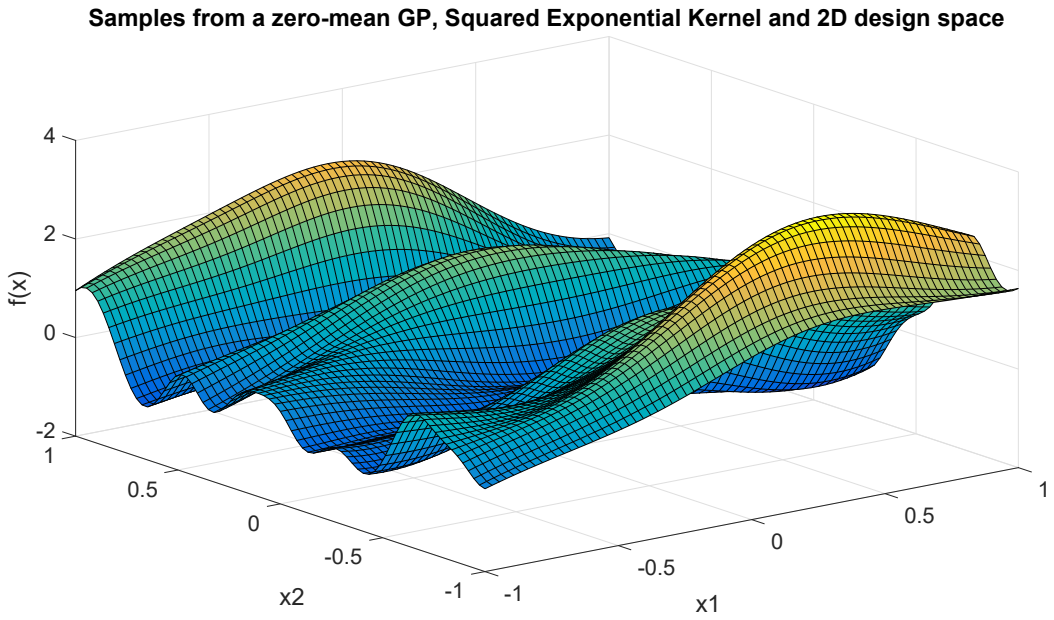



Figure 2.8. Sample from a zero-mean Gaussian Process with bidimensional input space and squared exponential kernel: the characteristic lengthscale along the $x_1$ direction is bigger, giving less oscillations with respect to the $x_2$ direction.

## 2.4 Results' validation methods

### 2.4.1 Method validation

In the next pages, a procedure is described, so as to implement the ideas introduced in this chapter. However, before the application to the actual problem, it was necessary to asses the effectiveness

of the surrogate-based optimization method. Furthermore, the adequacy of a Kriging emulator for the problem under investigation had to be determined.

The efficacy of the surrogate-based approach is shown in a number of test problems provided by the software DAKOTA: one of them is analysed in Sec. 3.6 of the next chapter.

The adequacy of the Gaussian Process to model the response functions of interest was instead tested on already available data, derived from a previous application of the classical optimization approach (no surrogate models). This result is analysed in Sec. 3.7.

### 2.4.2 Results' analysis procedure

As mentioned, the main scope of the thesis is to adopt a surrogate-based optimization approach so as to reduce the computational effort required by the whole optimization process in comparison to a more standard approach.

Therefore, the best way to compare the two procedures is to let them compete directly on the field: starting from the same initial population, both methods were run and their results examinated considering the number of real CFD simulations required and the number of generations needed to get to similar outcomes.

Finally, some of the most promising designs proposed by the surrogate-based approach are analysed. In particular, the distributions of the pressure ($C_p$) and skin friction ($C_f$) coefficients are compared to a reference case. The two coefficients are defined as:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{2.22}$$

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{2.23}$$

with

$$\tau_w = \mu_\infty \left(\frac{\partial u}{\partial y}\right)_w \tag{2.24}$$

The quantities denoted by "$\infty$" refer to the freestream conditions. The pressure coefficient is related to the difference between the static pressure at the considered point and the freestream pressure, normalized over the dynamic pressure. The skin friction coefficient is the normalization of the wall (or skin) shear stress ($\tau_w$), related to the velocity gradient at the wall and the dynamic viscosity ($\mu_\infty$).

Furthermore, the flow fields of the selected candidates are examined as well.

# Chapter 3

# The RACER Optimization Problem

In Chapter 3, the details of the study are explained: the first sections show the aim of the work and the geometry of the problem. Thereafter, the practical implementation of what described in Chapter 2 is presented: the construction of the response surface and the optimization strategy chosen, together with the softwares employed to put it in act and the test cases to assess its suitability, in the light of the issues emerged during the work.

## 3.1 Scope of the study

The problem under examination is the aerodynamic design optimization of the 2D profile of the rotor blade-sleeve fairing section belonging to the RACER demostrator. According to numerical investigations performed on similar helicopter configurations ([27]), blade-sleeves are the major contributors for rotor-head drag, accounting for around 16% of the parasite helicopter drag. Therefore, the blade-sleeve region is an interesting area where to look for drag reduction. The examinated section of the study is located close to root of the blade at a normalized radius $r/R = 0.78$.

In particular, the aim is to optimize the shape of the section in order to satisfy the following objectives:

- increase efficiency ($E = \frac{C_l}{C_d}$) of the advancing blade (blade azimuthal angle $\Psi = 90°$) during cruise flight
- reduce drag coefficient ($C_d$) of the retreating blade (blade azimuthal angle $\Psi = 270°$) during cruise flight.

Therefore, the applied objective functions are characterized by two different flow conditions, which are not directly related to each other. The other requirement is to implement a surrogate-based optimization procedure in order to save computational resources necessary for the study.

Each candidate aerodynamic profile is created with an automated process and is described by four Bézier curves and two connection segments. The design variables are given by selected coordinates of the Bézier curves' control points, which can span within fixed ranges. Moreover, the shape of the section has to fall inside a region limited by an inner and an outer boundary.

## 3.2 Geometry

### 3.2.1 Geometry design

Each design is built with four Bézier curves, two modelling the front and two shaping the back of the section, connected by two horizontal straight segments.

A Bézier curve is a parametric polynomial curve that links two points in $\mathbb{R}^2$ or $\mathbb{R}^3$ usually, and is in general written as:

$$P(t) = (1-t)^n P_0 + nt(1-t)^{n-1} P_1 + \cdots + nt^{n-1}(1-t)P_{n-1} + t^n P_n = \\ = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} t^i (1-i)^{n-i} P_i = \sum_{i=0}^{n} B_{i,n}(t) P_i \tag{3.1}$$

where: $t \in [0,1]$ is the parameter, $n$ is the degree of the Bezier curve, $P_i \in \mathbb{R}^2$(or $\mathbb{R}^3$) are the $n+1$ building points of the curve and $B_{i,n}(t)$ are the Bernstein polynomials, which form a basis of the space of all polynomials of degree lower or equal to $n$. Fig. 3.1 shows the polynomials setting $n = 4$.

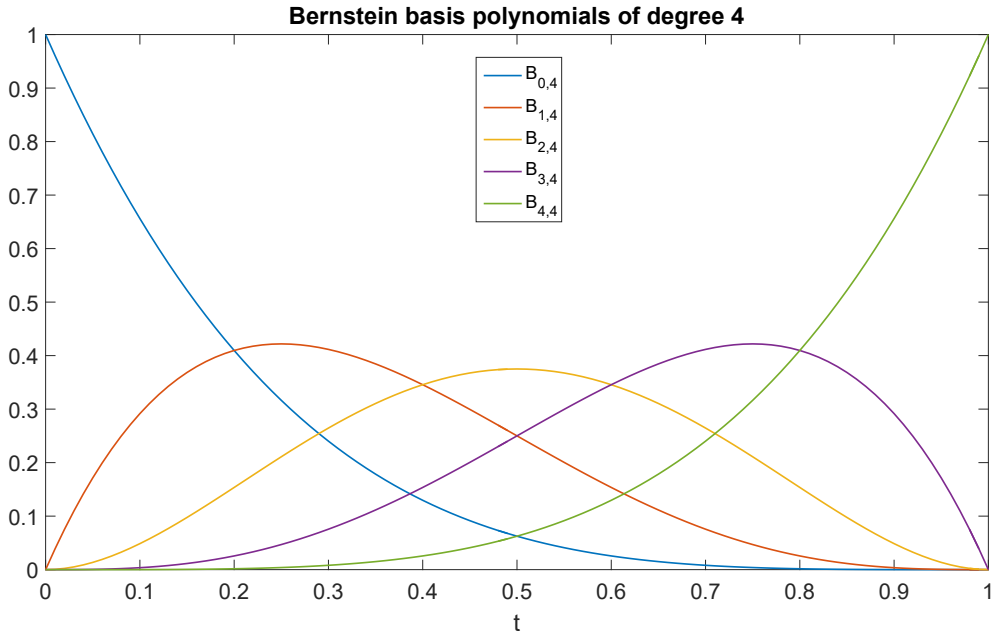



Figure 3.1. Example of Bernstein basis polynomials of degree 4: within the interval [0,1] all functions are positive and form a partition of unity.

A linear Bézier curve is the straight segment connecting the points $P_0$ and $P_1$. A quadratic curve is a segment of parabola connecting the extreme points $P_0$ and $P_2$, with the position of point $P_1$ governing the shape of the segment. A cubic curve is a segment of cubic polynomial linking the extreme points ($P_0$ and $P_3$) and shaped by the position of the two inner points ($P_1$ and $P_2$), and so on as $n$ gets bigger. The curve tends to mimic the shape of the control polygon with vertices $P_0,\ldots,P_n$, which are said to be the control points of the curve. The only two control points that belong to the curve are the first and the last one, as in Fig. 3.2 and Fig. 3.3.

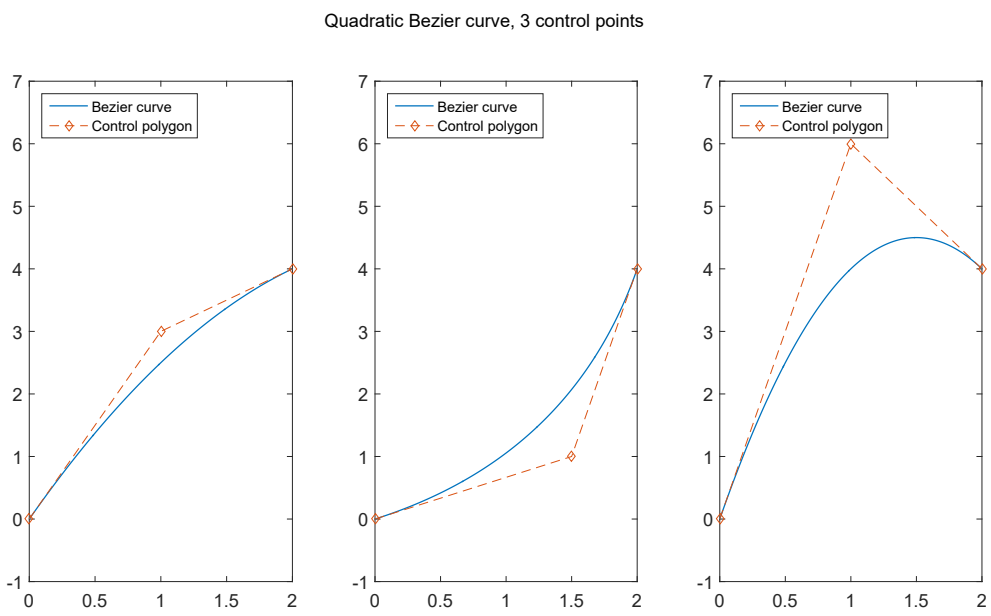



Figure 3.2. In a quadratic Bézier curve (blue solid line) the central control point of the control polygon (red dashed line) determines the shape of the segment of parabola.

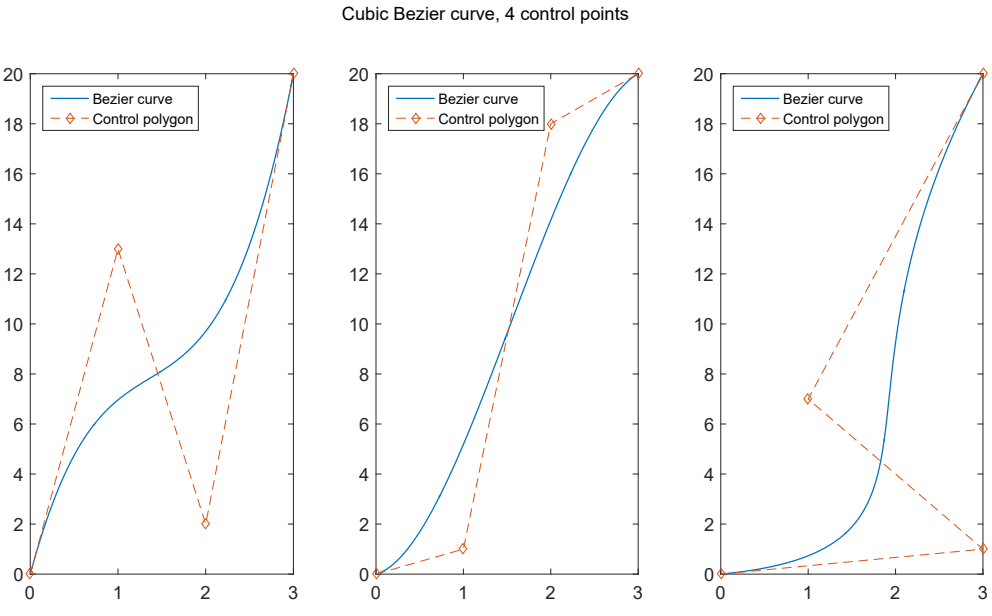



Figure 3.3. In a cubic Bézier curve (blue solid line) the two central control points of the control polygon (red dashed line) drive the shape of the cubic curve.

To draw the shape of the blade-sleeve section, four third-order Bézier curves are used. This totals 16 control points, each of them labelled $B_iP_j$, with $i$ and $j$ indicating the curve number and the control point of the specific curve, respectively.

The first curve is the upper left one, which is connected to the second one (upper right) thanks to the first straight segment. The third Bèzier curve is the lower right curve, connected to the fourth one (lower left) through the second horizontal segment. Each couple of side-curves (2-3 and 4-1) has an extreme control point in common, as can be seen in Fig. 3.4.

### 3.2.2 Constraints

The main geometrical constraints are represented by an inner and an outer boundary limiting the region for feasible designs, which are prescribed by the project partner Airbus Helicopters.

The inner boundary has a rectangular shape with rounded corners. The sizing of this boundary is based on the mechanical movement of the rotor blade and cooling requirements. The outer one has elliptical shape with a small offset in both the horizontal and vertical directions. It defines the maximum permissible size of the fairing and ensures that no contact between different mechanical components occurs.

Moreover, further linear inequality constraints exclude those designs that have peaky points and sharp edges:

$$\begin{aligned} B_1P_0z - B_1P_1z < 0 \\ B_2P_3z - B_2P_2z < 0 \\ B_3P_1z - B_2P_3z < 0 \\ B_4P_2z - B_1P_0z < 0 \end{aligned} \tag{3.2}$$

Lastly, some design coordinates are fixed for construction reasons and some points are forced to be aligned to ensure tangential transition between the curves, limiting the total number of degrees of freedom shaping the profile.

### 3.2.3 Degrees of Freedom

Each design is drawn with a total of 16 control points on the $xz$ plane. However, because of the previously mentioned matching conditions among the different segments generating the profile, the degrees of freedom to fully specify a single design are reduced to 14.

With "...$x$" and "...$z$" indicating the x and z coordinate of the point, respectively, the degrees of freedom are:

$$\begin{aligned} \boldsymbol{x} = \{&B_1P_0x,\ B_1P_0z,\ B_1P_1z,\ B_1P_2x,\ B_1P_2z,\ B_2P_1x,\ B_2P_2z, \\ &B_2P_3x,\ B_2P_3z,\ B_3P_1z,\ B_3P_2x,\ B_3P_2z,\ B_4P_1x,\ B_4P_2z\}. \end{aligned} \tag{3.3}$$

Each of the 14 design variables has its own defined continuous range of variation. An example of valid design meeting all the imposed constraints is shown in Fig. 3.4.

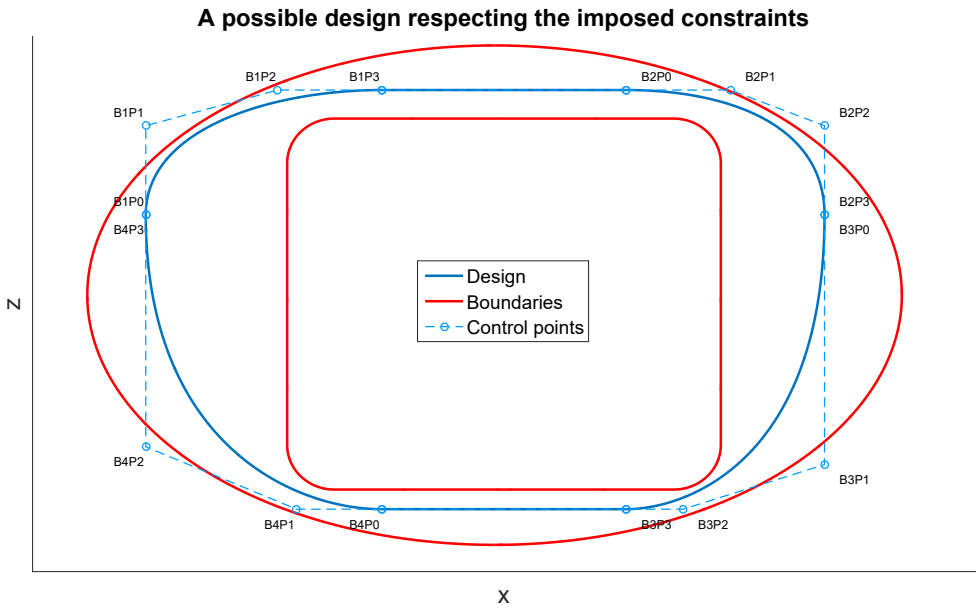



Figure 3.4. Example of valid design (blue solid line) enclosed inside the two boundaries (red solid lines).

## 3.3 Surrogate-based optimization process

### 3.3.1 Overview of the process

As written in Sec. 2.1, the process implies the construction of a suitable surrogate model starting from a set of training points sampled from the parameter space. The model is used to carry out the optimization and it is iteratively updated with some true function evaluations, which are derived from the expensive flow simulations. This approch should allow to reach an acceptable accuracy of the surrogate model within a reasonable amount of time.

However, because of some difficulties in dealing with the design constraints, it was necessary to slightly modify the approach, which is denoted "mixed" optimization strategy throughout the present work.

### 3.3.2 Building the surrogate model

Overall, 200 training points were sampled from the design space to build the initial Gaussian Process. These samples were evaluated by CFD simulations.

The selection of the training points was conducted by means of Latin Hypercube Sampling (LHS), a stratified sampling technique ([18] and [8], pp. 74-75), which means that the sampling population is divided in homogenous subsets before the sampling operation. The stratification is achieved by dividing the range of each variable to be sampled in $N_{samples}$ distinct segments of equal probability. According to the probability distribution of the variable over its whole range, the width of the segments could vary. For a uniform probability distribution the length would be the same for each part, for a normal distribution it would be smaller near the mean value of the variable, and so on. From each segment a sample is randomly selected. This operation is done for

all the variables in the design space. Then, all the samples obtained are shuffled and combined to create a set of $N_{samples}$ parameters vectors. Having picked only one value from each segment, every row and column of the hypercube partitions contains exactly one sample. The LHS technique ensures to completely cover the range of all the variables, thus providing complete information for each one of them.

As an example, consider a draw of 25 samples from a design space with two variables ($x$ and $y$) uniformly distributed between 0 and 1; each range is first divided in 25 parts of equal length: the resulting 2D grid has $25 \times 25 = 625$ tiles. Then, a random value is picked from each segment (so, in total, 25 draws for $x$ and 25 for $y$): these draws are then combined so as to obtain 25 couples $(x_i, y_i)$, so that, on the 2D grid, each row and each column contains just one marked tile. This can be seen in Fig. 3.5.

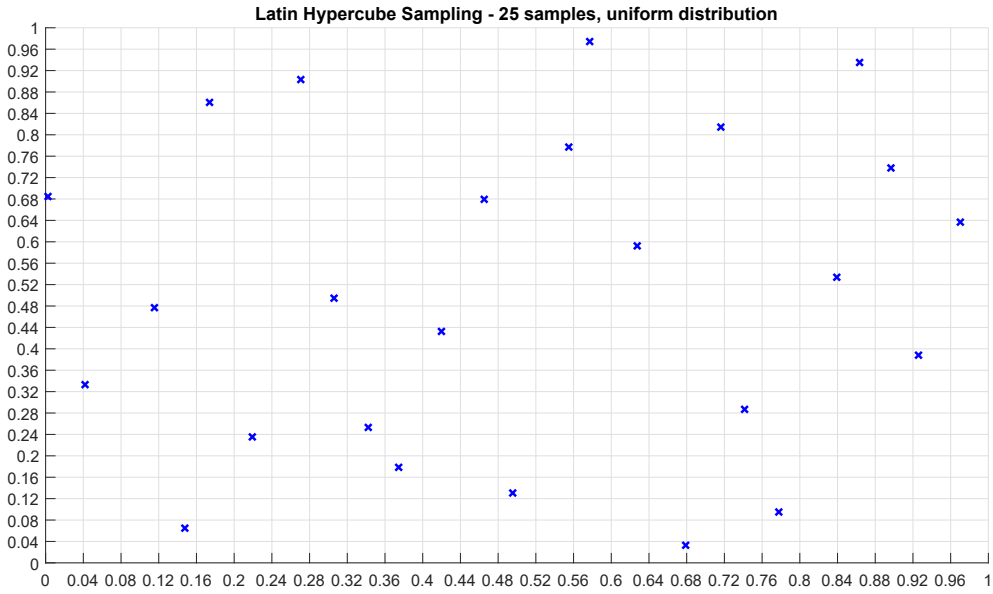



Figure 3.5. Example of LHS from a 2D design space with uniform distribution, each row and column of the design space contains only one sampled point.

In case of a normal distribution of the variables, what changes is just the length of the segments in which each variable range is divided. In the region which is closer to the mean value of the Gaussian, where the variable's probability distribution is higher, the length of the segments is smaller. Hence, more segments and, consequently, more sampled points cluster around the peak of the Gaussian.

The same approach can be adopted for higher dimensional design spaces, like the actually considered problem with its 14 design variables. In this case, all 14 variables were supposed to have a uniform probability distribution. Thus, each range was divided in 200 parts and a sampled was taken from each segment obtaining 200 parameter vectors.

Then, these 200 design were evaluated by CFD simulationsl to get the initial training set.

### 3.3.3 Choice of the optimization strategy

The optimization toolbox DAKOTA offers several surrogate-based optimization methods, such as surrogate-based local minimization, surrogate-based global minimization and hybrid minimization methods.

Within the present work, the choice fell on a surrogate-based global minimization method. It consists in building a global surrogate model (in this case a Kriging model) on which the optimizer directly operates, and then in updating it, again globally, at each iteration.

This method is implemented through the following steps ([8], pp. 263-265). At the beginning, the surrogate model is built using the initial training points. Then, through a multi-objective optimization on the surrogate, the best solutions (surrogate Pareto Front) are found and a selected subset of these is evaluated with the true model. These new true evaluations are then appended to the initial set of training points and the resulting new set is used to update the surrogate model. Actually, what DAKOTA really does at this stage, is re-building each time the emulator from scratch. Thereafter, the emulator is used in the next optimization run, and so on, until a stopping criterium is met.

In principle, using this method, the model should get more and more accurate in the region of interest at each iteration: this should lead fairly quickly to optimality, without losing to much time in exploring unnecessary regions of the design space. However, there is no guarantee of convergence.

This method, as implemented in DAKOTA, was designed to be specifically used for multi-objective genetic algorithm optimization: the idea is to use the surrogate Pareto Front points to refine the model while limiting the number of points necessary to this operation. Moreover, it allows to update the emulator globally in the frame of an iterative process, possibly correcting the general behaviour over the entire design space.

A further aspect to be taken into account is that this method allows to evaluate several design points at the same time by invoking the CFD simulations in parallel and then providing a large number of new points for the update of the surrogate model. This is made possible by the fact that, during each iteration, the optimizer determines the whole set of surrogate Pareto Front points at first, and then calls the CFD solver to validate the solutions that have just been found.

### 3.3.4 Discarded optimization techniques

Several other optimization strategies were also considered among those allowed by DAKOTA, but they were then discarded for different reasons.

One of the main limits is represented by the number of real CFD evaluations. A local approach, such as a trust-region approach, would allow a detailed exploration of restricted regions of the design space by building a very accurate emulator in areas of interest and marching to local optima via small optimization steps.

The general procedure is the following ([8], pp. 259-263): one design point is chosen, for example from the initial set of training data, and a small trust-region of the design space is "drawn" around it. From this region, the needed number of points is sampled to build the emulator, possibly using data already available from the starting set. Once the model is ready, it is used to explore the region to find in which direction the function has its local optimum: this step can be done in several ways. As soon as a suitable new point is found, this has to be validated running the real model, which could confirm or deny the result. If the step is accepted, the new point becomes the center of a new trust-region and the process repeats; if the step is rejected, the situation could be handled in different ways, like considering the other best directions, taking just half of the step or modifying the dimension of the trust-region. An advantage of this method is that it is convergent ([19], pp. 83-89). However, its major drawback lies in the need of evaluating several

regions to reach an optimal design: even with a simple linear approximation, a high number of training points is necessary in each region to build a surrogate. For example, given the 14 design variables of the problem under examination, at least 15 training points in each trust region would be needed. Furthermore, in presence of several local optima (as it could often happen in multi-objective optimizations), it is inevitable to start the search from different initial positions, each with its own trust-region and emulator. The higher the number of these initial designs is, the larger is the amount of true model evaluations required. Moreover, a linear model is, in general, not accurate enough and, therefore, the amount of real function evaluations further increases.

Hence, it turned out that this approach is not easily applicable to the problem under investigation, because it would take much more time than the standard genetic algorithm approach, even when using a multi-start method to analyse more paths in parallel.

The same drawback affects those methods that rely on the information about the first and/or second derivatives to update the model. The accuracy improvement would be local and too many real CFD would be needed just to build the model and updating it. Hence, methods like those based on Taylor series approximation could not be used.

A possible expedient could be the use of a hybrid optimization strategy, which combines a global and a local approach. The idea is to use a global method to find a first surrogate Pareto Front, which is then refined with a trust-region method starting from the Pareto designs. However, the local part still represented the weakest link in the chain, because it needed again a considerable amount of time-consuming true evaluations to progress.

Finally it is to bear in mind that, in trust-region approaches, the validation of the single point at the center of each region represents a sort of misuse of the computational resources of the SuperMUC. Indeed, only one simulation at a time can be run in this particular situation, and in general is a step that slows the whole process down.

## 3.4 AASD Optimization tool chain

### 3.4.1 General description

The complete design and optimization process was performed with the Automated Aerodynamic Shape Development tool chain (AASD, [20], [26]). This tool chain was developed during the early stages of the FURADO project at TUM and allows for aerodynamic design optimization using CFD simulations.

The tool chain is composed of five main modules (Fig. 3.6) which are put in communication via Python scripts: the optimization module (A), which relies on state-of-the-art algorithms provided by the software DAKOTA; a geometry generator module (B) that creates the shape of the design using CATIA; a mesh generator (C), which can work with either ANSYS ICEM ([21]) or CENTAUR from CentaurSoft; a flow solver (D), ANSYS FLUENT ([22]) or the DLR-TAU code ([23]); and an analysis module, which consists of a Python script.

The optimization task is performed by DAKOTA, whose output is a set of design variables that represents the control points of the candidate design to be evaluated. Before the generation of the corresponding geometry, a Python script checks the candidate against the constraints previously described. If those are not met, the analysis module immediately returns bad objective functions to DAKOTA and the current loop for that design is considered complete, while in case of positive outcome, the shape is automatically generated using CATIA and, once ready, the mesh is set up. The user can choose between ICEM and CENTAUR. Then the design is evaluated through the CFD software, optionally FLUENT or TAU. Subsequently, a Python script allows for the analysis of the flow simulations and the evaluation of the objective functions. These are passed to DAKOTA closing the loop.

During the process it is possible to run more CFD simulation at the same time, which allows to progress faster in the optimization procedure; however, this requires the access to a remote HPC network, in this case the SuperMUC Petascale System at Leibniz Rechenzentrum ([24]).

Within this thesis, the ANSYS softwares ICEM and FLUENT were used in the tool chain. A maximum of 10 candidates were evaluated in parallel at the same time on SuperMUC.

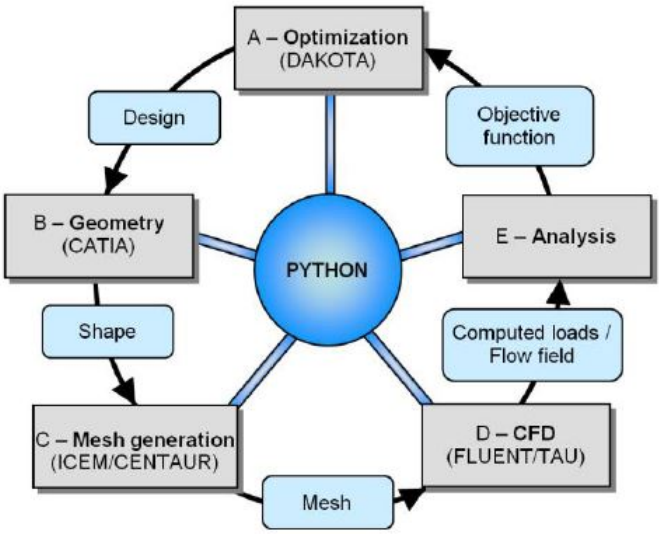



Figure 3.6. AASD tool chain ([20], p. 3).

### 3.4.2 DAKOTA optimization software

The "Design Analysis Kit for Optimization and Terascale Applications" (in short, DAKOTA) is a software developed at Sandia National Laboratories. It contains a vast quantity of optimization algorithms and provides methods to flexibly apply and combine them.

As pictured above, it represents the starting and finishing point of each iteration of the tool chain and it is the brain controlling the optimization process.

The surrogate-based global optimization approach implemented in DAKOTA was specifically designed for MOGA, which is the multi-objective genetic algorithm used to perform the optimization. The software offers a certain number of global emulators. Among these, the user can choose to build a Gaussian Process specifying a set of training points, hence creating a dedicated Kriging model for each objective function. Up to now (version 6.8, released in May 2018), the only correlation function type available in the program is the Squared Exponential Kernel, whose hyperparameters are determined via the Maximum Likelihood Estimation (MLE). In DAKOTA, the characteristic lengthscale is not directly estimated; rather, the software calculates the parameters defined as:

$$\theta_k = \frac{1}{2l_k^2}, \tag{3.4}$$

where $k = 1, \ldots, n$ and $n$ is the number of variables. Together with the value of the variance, they are estimated from the training data using the global optimization method DIRECT (DIvision of RECTangles, [8], p. 131). Alternatively, the user can provide tha values for the characteristic lengthscale, if specific correlations between variables need to be fixed ([8], pp. 162-163).

As of version 6.8, the program is still not provided with a completely problem-free Graphical User Interface (which has been introduced only in version 6.5). Therefore, the user has to communicate with the software through text input files, that are read by the program and then executed with no possibility to intervene until the process ends. This feature could create some head-scratchers, especially if coupled with the issues described in Sec. 3.8.

## 3.5 Numerical setup

### 3.5.1 Flow conditions

The numerical flow simulations are conducted for two flow conditions. These are the advancing and the retreating rotor blade, as these are the azimuthal rotor positions ($\Psi = 90°$ and $\Psi = 270°$, respectively) generating the highest drag values ([25]).

The two objective functions of the study correspond to the aerodynamic efficiency coefficient (given by the lift-to-drag ratio $E = \frac{C_l}{C_d}$) for the advancing blade and the drag coefficient ($C_d$) for the retreating one.

The simulations are performed under the ICAO standard conditions at sea level (Tab. 3.1). The velocity facing the blade-sleeve section is given by the combination of the cruise flight velocity of the RACER, $V_{Cruise} = 220\ kts$, and the rotational speed of the rotor at a dimensionless radial coordinate $r/R = 0.078$, with $R$ being the total blade's radius. The collective pitch, the longitudinal-cyclic pitch and the flow deflection caused by the fusoleage contribute in determining the angle of attack of the advancing and retreating blades.

Table 3.1. Flow conditions for sea-level cruise of the RACER.

| | | |
|---|---|---|
| Temperature $T_\infty$ | 15 | $[°C]$ |
| Pressure $p_\infty$ | 101325 | $[Pa]$ |
| Density $\rho_\infty$ | 1.225 | $[kg/m^3]$ |
| Cruise speed $V_{Cruise}$ | 220 | $[kts]$ |

### 3.5.2 Computational domain

The CAD program CATIA is used in batch mode to generate the geometry of each candidate design. Thereafter, the mesh generator ANSYS ICEM automatically creates the computational mesh. A block structured grid with hexahedral elements is used to discretize the spatial domain. The domain is quasi two-dimensional as it is extruded along the blade radial direction by one hexahedral element. For each candidate the final mesh is composed of 276000 elements: this number was selected in a previous stage of the FURADO Project as it proved to be a good compromise between computational effort and accuracy [26].

The boundary conditions of the flow problem are given by no-slip wall conditions on the profile and farfield conditions on the outer boundary of the computational domain. The farfield is considered to be a circumference of a 40-chord length diameter centered around the profile, and it can be seen in Fig. 3.7.

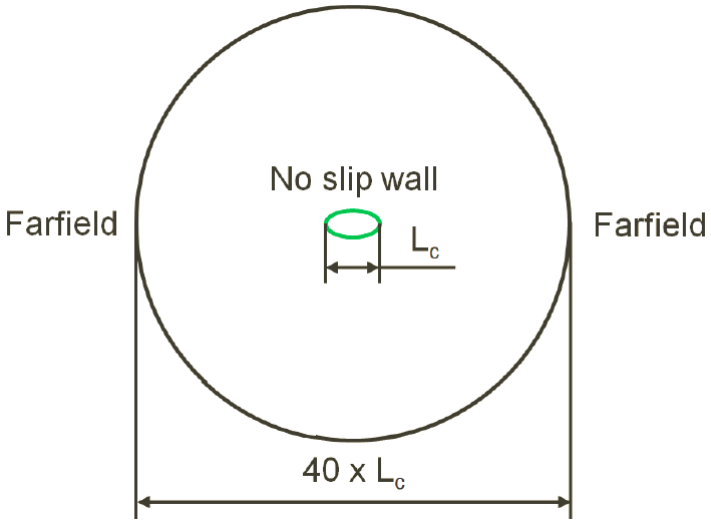



Figure 3.7. Boundary conditions of the computational domain.

### 3.5.3 CFD solver setup

The CFD solver ANSYS FLUENT is used to solve the compressible URANS (Unsteady Reynolds-Averaged Navier-Stokes equations), being the $M > 0.3$ under the considered assumptions for the advancing blade. A steady state solution helps in initializing the flow conditions of the transient simulation. Turbolences are modelled by the $k-\omega$ SST model ([28]) and the turbolence properties at the farfield are set so as to provide a turbolence intensity of $Tu = 0.3\%$ at the front of the profile. The coupling between pressure and velocity is handled by the Semi-Implicit Method for Pressure Linked Equations Consistent (SIMPLEC) algorithm, with a standard pressure scheme for pressure interpolation and a second-order upwind scheme for spatial discretization of density, momentum, turbolent kinetic energy, specific dissipation rate and energy. The time discretization is operated via a bounded second-order implicit scheme, with a time step $\Delta t = 10^{-4}s$. Air is considered as an ideal gas to perform the compressible flow simulation and its properties for standard pressure and temperature are listed in Tab. 3.2.

Table 3.2. Ideal gas air properties.

| | | |
|---|---|---|
| Dynamic viscosity $\mu_\infty$ | $1.789 \cdot 10^{-5}$ | $[kg/ms]$ |
| Specific heat capacity $c_p$ | 1006.43 | $[J/kgK]$ |
| Thermal conductivity $\lambda$ | 0.0242 | $[W/mK]$ |

## 3.6 Test case

Before proceeding with the application of the surrogate-based optimization to the actual problem, a test case was run to assess the suitability of the method.

Within the next pages, a test problem from DAKOTA is shown so as to demonstrate the capabilities of surrogate-based optimization in solving multi-objective optimization problems using

genetic algorithms. This test case is also more extensively analysed in [29] and as a part of a test suite to investigate Multi-Objective Evolutionary Algorithms in [30] (pp. 175-182).

The problem is formulated as follows:

$$\begin{aligned} \boldsymbol{F} &= (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \\ f_1(\boldsymbol{x}) &= 1 - e^{-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2} \\ f_2(\boldsymbol{x}) &= 1 - e^{-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2} \\ \text{for } x_L \leq x_i \leq x_U &\text{ with } i = 1, \ldots, n \end{aligned} \tag{3.5}$$

The problem produces a n-dimensional symmetric and simply connected Pareto Optimal Set and a single 2D concave Pareto Front, with $n$ being a positive integer number at choice.

This test case has a known analytical solution for the Pareto Front: the two functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ have minima at $(x_1, \ldots, x_n) = (\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}})$ and $(x_1, \ldots, x_n) = (-\frac{1}{\sqrt{n}}, \ldots, -\frac{1}{\sqrt{n}})$, respectively. Therefore, the Pareto Optimal Set is given by the points defined by:

$$x_1 = x_2 = \cdots = x_n \quad \wedge \quad -\frac{1}{\sqrt{n}} \leq x_1 \leq \frac{1}{\sqrt{n}} \tag{3.6}$$

As implemented in DAKOTA, the test problem shows the case with $n = 3$ and constraints $x_L = -4$, $x_U = 4$. Thus, the Pareto Optimal Set is a 3D straight line intercepting the triplets $(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})$ and $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$, as shown in Fig. 3.8.

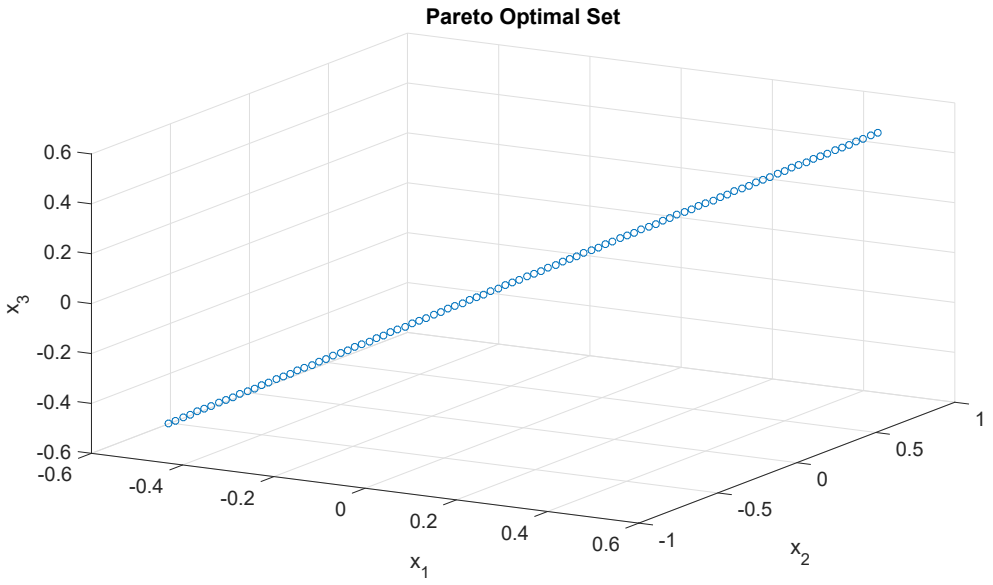



Figure 3.8. Test case: the Pareto Optimal Set is a symmetric and simply connected set.

Fig. 3.9 shows the comparison among three Pareto Fronts: the analytical one (as in Eq. 3.6), the one computed implementing a multi-objective genetic algorithm directly on the true model (Eq. 3.5) and the surrogate Pareto Front from the application of the genetic algorithm on a surrogate model, in this case a Gaussian Process generated using an initial set of 50 training points sampled via LHS.

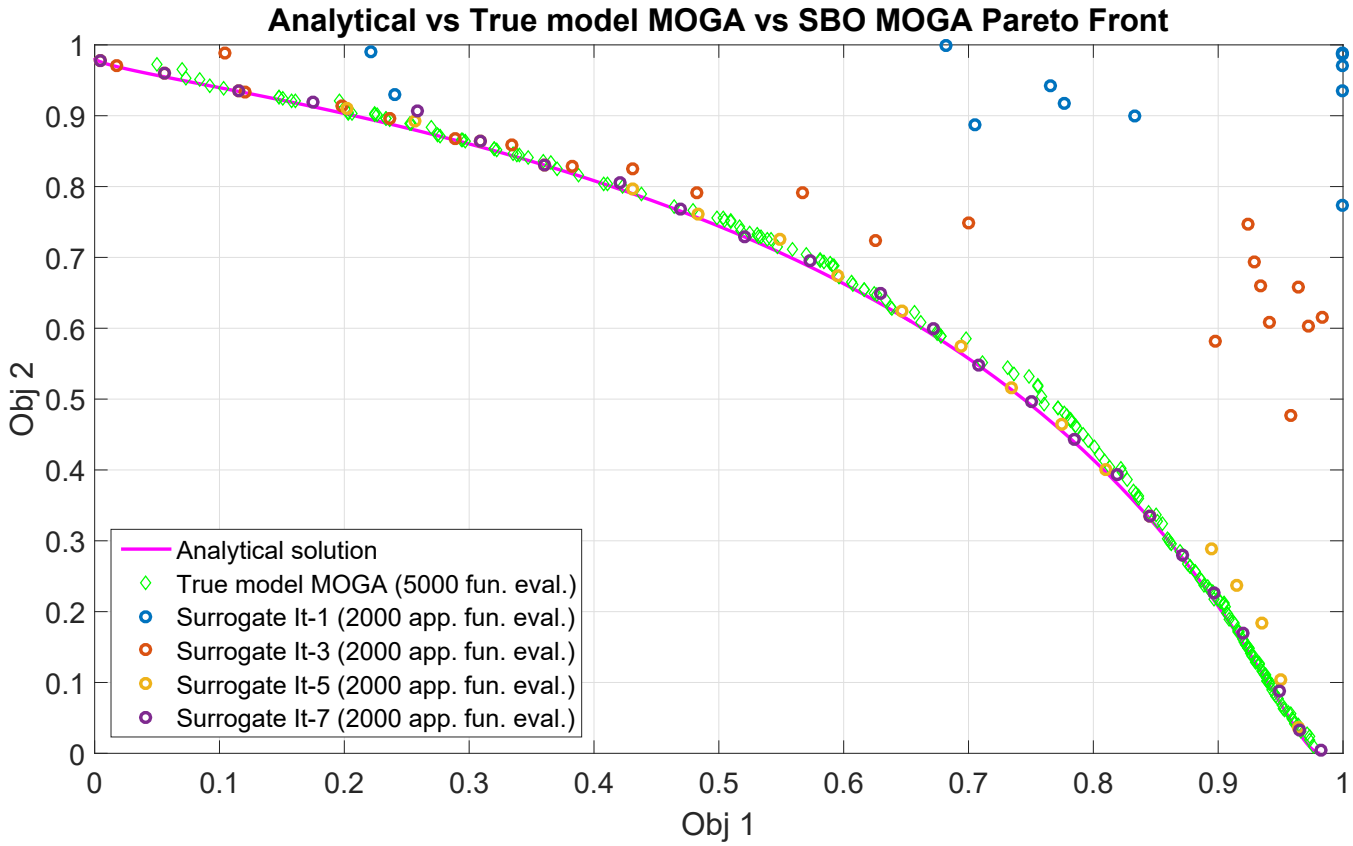



Figure 3.9. Test case comparison: analytical solution vs. true model MOGA solution vs. surrogate-based MOGA solution.

The solid line represents the analytical solution. The green diamonds are Pareto Front points obtained after applying the MOGA with 5000 evaluations on the true model. The coloured circles indicate the actual values (that is, after the validation on the real model) of the surrogate Pareto Front points computed at selected iterations 1, 3, 5 and 7 of the surrogate-based optimization. In each iteration, the emulator was used to run MOGA for a total of 2000 approximate function evaluations; the surrogate optimal points at the end of each iteration formed the surrogate Pareto Front. These points were evaluated on the true model before being added to the training set and run the next iteration. As it is shown, at first the evaluation of the surrogate Pareto Optimal Set does not match the real Pareto Front. However, as the model is updated, it quickly converges.

In this example, building and updating the emulator up to the shown results requires around 200 true function evaluations and 14000 approximate function evaluations computed on the surrogate model. The true function evaluations come from 50 initial points plus around 20 at each new iteration, while the approximate ones come from 2000 per iteration. These numbers can be modified by tuning some input parameters in DAKOTA. On the other hand, 5000 true function evaluations are needed to get a comparable result when performing the optimization directly on the real model.

Although in this test the true model is not particularly expensive, the surrogate approach allows to save roughly 95% of (potentially) time-consuming function evaluations. In general, more complicated models (as CFD outputs) require a higher number of true function evaluations to build a sufficiently accurate emulator; however, this number is still small when compared to the amount of function evaluations used in a traditional genetic algorithm approach.

Therefore, such a method offers the possibility to reduce the computational effort regarding the implementation of optimization tasks with genetic algorithms.

## 3.7 Gaussian Process model tested on a similar problem

A test was run on a set of available data in order to verify whether a Gaussian Process emulator could suitably model the response space of the problem. Furthermore, it was necessary to prove whether the method is reliable enough in making reasonable predictions.

The dataset was composed of 412 valid designs, which originated from a previous optimization process with a genetic algorithm using CFD simulations for the function evaluations.

The set was divided in two groups: training data to build the Kriging model and test data to evaluate the accuracy of the predictions made by the model. Three models with different training/test point ratios were studied: the test set was fixed and composed of 22 randomly picked points, while the training set was different for the three models. The first one counted 210 training points; in the second, 90 more training points were added, for a total of 300 supporting points; in the last model, all 390 supporting points were used.

Running three cases was intended to observe whether increasing the number of training points (thus simulating the update of the model during a surrogate-based optimization) could actually improve the accuracy of the emulator in predicting new results.

For each predicted point, the error with respect to its true value was computed as:

$$e_q(x_i) = q^{pred}(x_i) - q^{true}(x_i) \tag{3.7}$$

with $q$ being either $E$ or $C_d$.

In particular, the mean absolute error (MAE) was the metric considered to compare the three different models. This metric is computed as the average of all the absolute residuals of the quantity of interest:

$$MAE = \frac{\sum_{i=1}^{n} |e_q(x_i)|}{n} \tag{3.8}$$

where $n$ is the number of tested points.

Moreover, the combination of the two residuals for $E$ and $C_d$ allows to obtain the euclidean distance error between predicted and actual values for each test point.

All the results that appear in the next figures and tables have been normalized with respect to a symmetric reference case. The applied normalization formulas are given by:

$$E_{norm} = \frac{E_{real} - E_{ref}}{|E_{ref}|} \tag{3.9}$$

$$C_{d,norm} = \frac{C_{d,real} - C_{d,ref}}{|C_{d,ref}|}. \tag{3.10}$$

Note that, when normalizing $E_{ref}$ and $C_{d,ref}$ for the reference case using Eq. 3.9 and Eq. 3.10, the normalized values are equal to 0. This means that, on a normalized coefficient-drag plane, the point corresponding to the reference profile occupies the origin of the axes.

Fig. 3.10 shows the results of the first case: Kriging's predictions are represented by the coloured dots and each of them is linked to its corresponding true value. The efficiency coefficient for the advancing blade case is plotted on the horizontal axis, while the drag coefficient for the retreating one is given on the vertical axis. The colour bar indicates the total distance error (the euclidean distance) between a prediction and its actual value. When looking at the plot, it has to be beared in mind the difference in the lengthscale of the two axes. Similarly, Fig. 3.11 shows the same kind of results for the second case, while Fig. 3.12 shows those of the third case.

Tab. 3.3, instead, compares the values of the two metrics considered for the three different surrogate models.

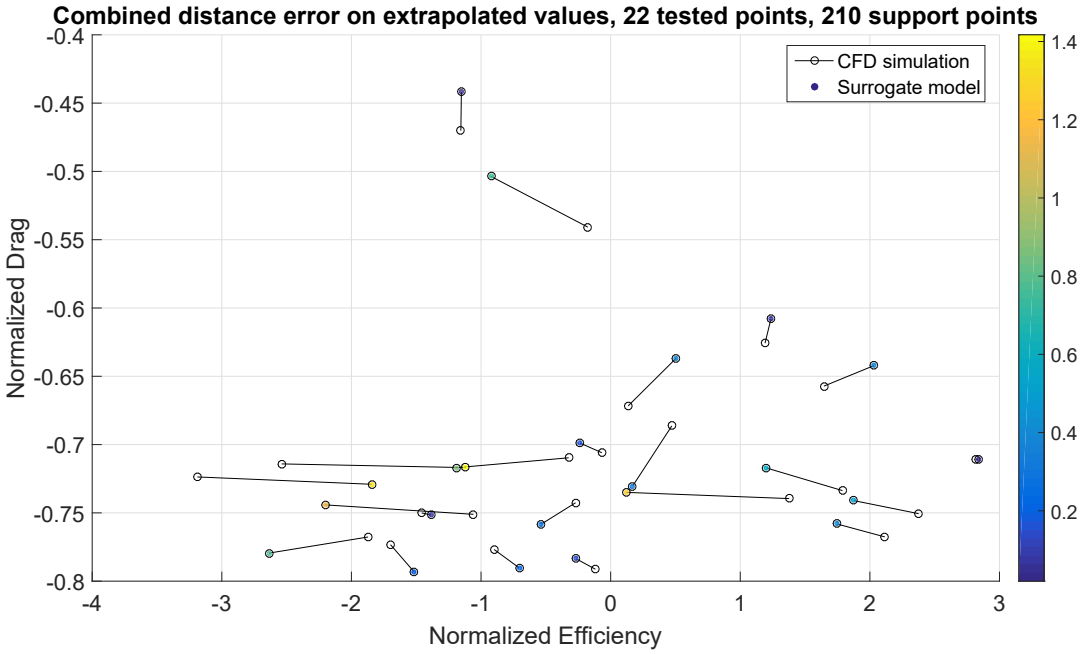



Figure 3.10. Error plot of predictions by the first Kriging model, built with 210 training points and checked against 22 test points.

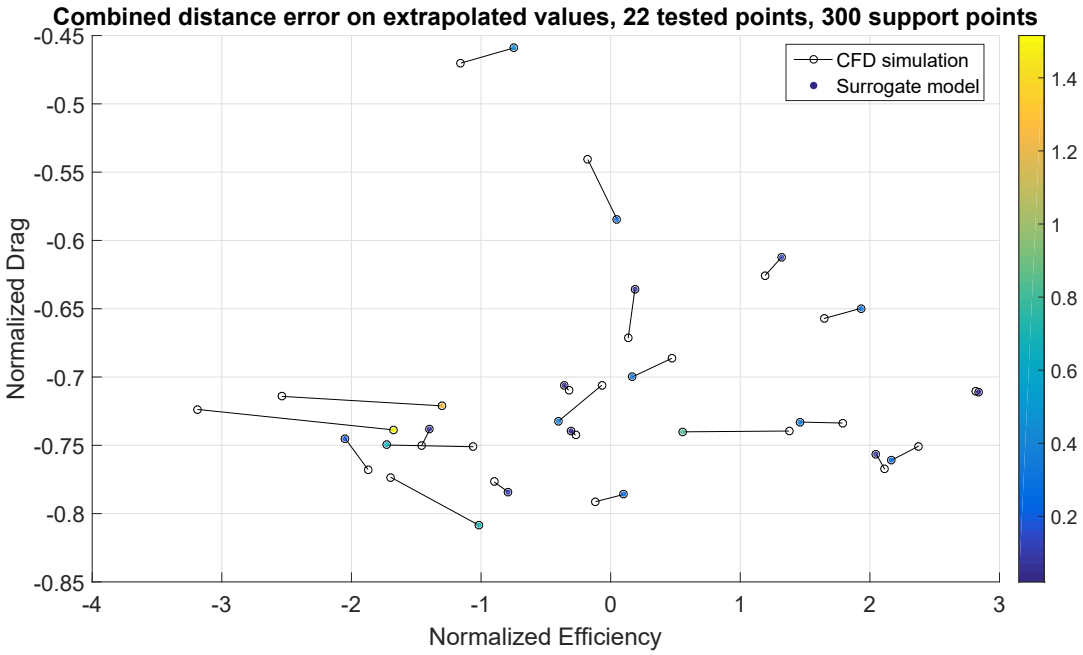



Figure 3.11. Error plot of predictions by the second Kriging model, built with 300 training points and checked against 22 test points.

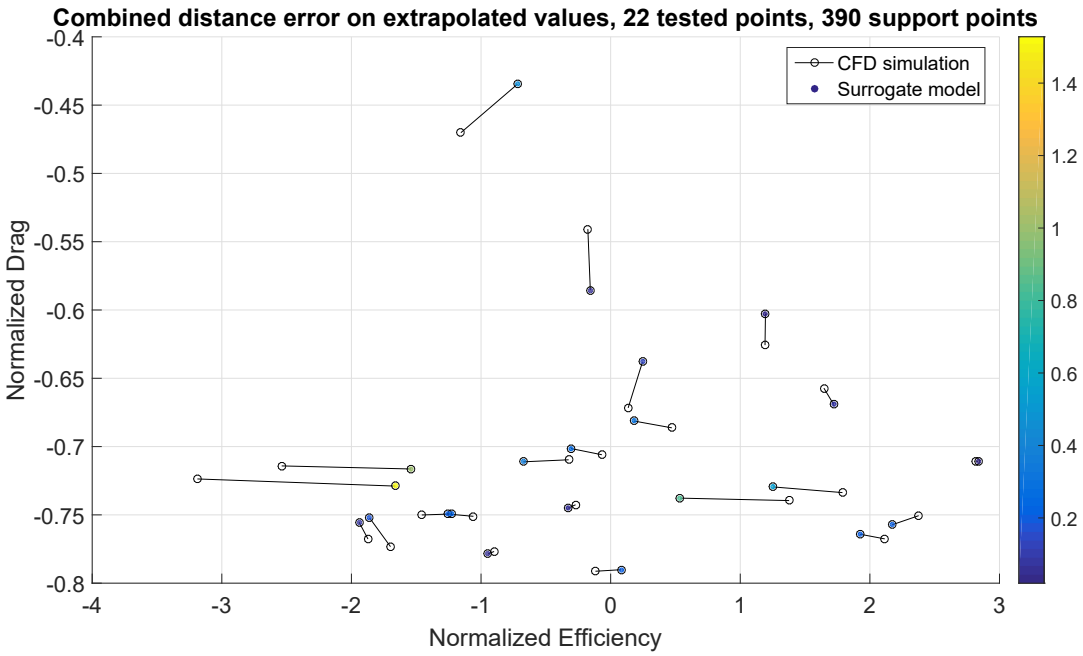



Figure 3.12. Error plot of predictions by the third Kriging model, built with 390 training points and checked against 22 test points.

Table 3.3. Comparison of metrics values for the three surrogate models.

| | 210 support points | 300 support points | 390 support points |
|---|---|---|---|
| Efficiency: mean absolute error | 0.507124 | 0.360348 | 0.309615 |
| Drag: mean absolute error | 0.014405 | 0.013066 | 0.010248 |

Comparing the three models, the prediction over efficiency improves substantially as the number of training points increases: the normalized mean absolute error reduces from roughly 0.50 of the first case to 0.36 in the second and to 0.31 in the third.

Additionally, the drag prediction improves as well by adding new training points. However, starting from an already very low normalized error, its progression is not as strong as the one for efficiency. Still it improves from 0.014 to 0.013 and finally to 0.010.

In general, for all three cases, the predicted results can be considered acceptable, especially when considering drag coefficient predictions. The prediction over efficiency appears less reliable, even though most of the time it is close to the actual value. Such a discrepacy in the behaviour of the two objective functions could be explained by the fact that the efficiency is given by the ratio of two coefficients (lift over drag). Because both of them have nonlinear trends, they are perhaps more difficult to be modeled than a single nonlinear coefficient, as the case of the drag.

This simple test proves that a Gaussian Process surrogate is able to model and predict the outcome of a CFD simulation, with acceptable results, and that it is possible to improve the accuracy of the model by increasing the number of training points: this is particularly true for the drag coefficient, while the efficiency prediction remains less reliable.

The results of the test need to be considered also in the light of two important aspects. First, the functions that are approximated have nonlinear behaviour in the real world, even for the simplest configurations, and they are almost always modeled with more or less complicated surrogate models. Second, the problem under investigation is of considerable dimension, being the emulator built on 14 variables. Even though this number is often small compared to the majority of machine learning algorithms applications, it is still a remarkable amount of parameters to be used for a surrogate of an aerodynamic model. These 14 variables have direct impact on airfoil parameters such as: chord length, thickness and camber line of the profile, curvature at leading and trailing edges and angle of attack. So, the model here built is trying to cover a large number of different configurations at once.

## 3.8 Implementation of nonlinear constraints

As explained in Sec. 2.1, linear and nonlinear constraints are treated separately in optimization problems.

In DAKOTA, linear constraints can be directly applied to the variables, before these are selected for a candidate design point. Thus, in principle, no design violating linear constraints is considered as feasible. However, this is not always true because of the way DAKOTA handles this type of constraints. Nonlinear constraints, instead, can not be explicitely imposed: when using a surrogate-based approach, they are also modeled as if they were further objective functions (Eq. 2.2). This poses the problem of training the optimizer so that it can tell in advance the good candidates from the bad ones. This should allow that only the designs respecting the constraints would be proposed as a solution for the surrogate model.

One possibility for the selection process is to associate to each design vector a flag value indicating whether the point is valid or not. For example, the value 0 is associated to feasible candidates

and the value 1 to infeasible ones. So, with this information, a nonlinear constraint function can be built: this, being a Kriging emulator itself, produces a smooth response curve. Then the optimizer can use this function to try to predict which of the candidates generated by the genetic algorithm are actually meeting the constraints, hence keeping them while filtering the others out. The valid designs should be those whose constraint function value is closer to 0 (for example within a selected range like $[-0.01,0.01]$).

This simple solution works fine. However, if the set of training data is composed of only valid design points (as for the initial set sampled with LHS), then the information about nonlinear constraints is not really specified: if all the designs are valid, the resulting nonlinear constraint emulator creates a flat response curve of value 0, which fails in filtering future designs. Thus, the first generation of surrogate Pareto designs could contain a high number of non-valid candidates, which are correctly recognized as so only once they enter the tool chain process: there, the analysis module returns bad objective functions for those candidates. Nevertheless, these points are still used, together with the valid ones, to update the emulator. On one side this helps to train the nonlinear constraint part, but on the other side it ruins the objective functions' surrogate models, which ultimately leads to a failure of the whole process. This is particularly true when the amount of infeasible designs used to update the model is around 15% of the total training points or more. At that point, the only possibility to continue the process without losing the valid designs found so far is to manually remove infeasible points from the training list and restart another optimization loop. But this operation is defenitely time-consuming and, if the number of constraint-violating designs generated at each iteration is too high, it slows the whole process down.

Hence, to mitigate this issue, it was necessary to partially limit the ranges of variation of the design parameters, so as to ensure that, while running MOGA, the final solutions proposed as surrogate Pareto Front were, for the biggest part, feasible designs. This solution led to the possibility of succesfully computing up to four or five consecutive surrogate Pareto Front generations without producing too many infeasible designs, before it was necessary to clean the set of training points.

## 3.9 Mixed strategy

To work around the previously mentioned problems, it was decided to follow a mixed strategy. This consists in combining a surrogate-based approach, to rapidly kick-start the search for Pareto solutions, with a classical genetic algorithm optimization on the true model, to refine the result.

The first step was implemented as explained in Sec. 3.3: a set of 200 training points was selected via LHS, consequently a global Kriging emulator was built and used to perform the multi-objective optimization. Each variable's range of variation was limited to around 65% of the original extent, mitigating the problems exposed in Sec. 3.8 and enabling to calculate, in avarage, five generations before any problem showed up forcing to clean the training set. During each iteration, the MOGA optimizer ran a total of 4000 approximate function evaluations on the surrogate model to find the surrogate Pareto Front. Then, a set of maximum 25 of these solutions were passed on to the CFD solver to be validated. Finally, those points were appended to the training set before the following iteration. The scheme of this first phase ("Phase 1") is shown in Fig. 3.13.

Once this approach had exhausted its potential and no more new Pareto optimal solutions were generated, the second step started.

In the second stage ("Phase 2"), the best 200 candidates among all the valid designs evaluated so far (training set plus Pareto solutions) were used as the initial population for the optimization with the MOGA optimizer using CFD simulations for the function evaluations. During this second phase, all variables could span over their original ranges and only linear constraints were implemented: those designs violating nonlinear constraints simply received bad objective functions, to ensure they could not progress to further generations. To assess the candidates' fitness, all the

designs from the current generation were ranked by the number of other designs they were dominated by: those who were dominated by no more than six other designs were considered eligible to progress to the next iteration. However, if that number had been too little, DAKOTA would have selected more valid candidates, forcing at least 95% of the current population to become the new parent population for the subsequent iteration. In each iteration two children designs were created by two parent designs setting a crossover rate of 0.8 and a mutation rate of 0.05: this last number multiplied by the population size of the current generation gave the total amount of mutations in each iteration. Two stopping criteria were set: the maximum number of true function evaluations (3000) and the maximum number of generations (20).

The outcome of this procedure was compared to the results obtained from the standard MOGA method applying CFD simulations for the objective function evaluations. This standard process started with the initial 200 training points that were sampled through LHS. Moreover, the same settings and stopping criteria regarding the genetic algorithm were considered.

The original plan was to run up to 20 generations for each tool chain. Unfortunately, there were several technical problems at LRZ which affected the SuperMUC Petascale System. This did not allow the SuperMUC to run properly and even put it out of order for several weeks. For this reason, it was not possible to run all the programmed simulations. Therefore, the chain with the standard MOGA algorithm could complete 19 generations out of 20, while the mixed approach chain could only complete 5 out of 20 populations.

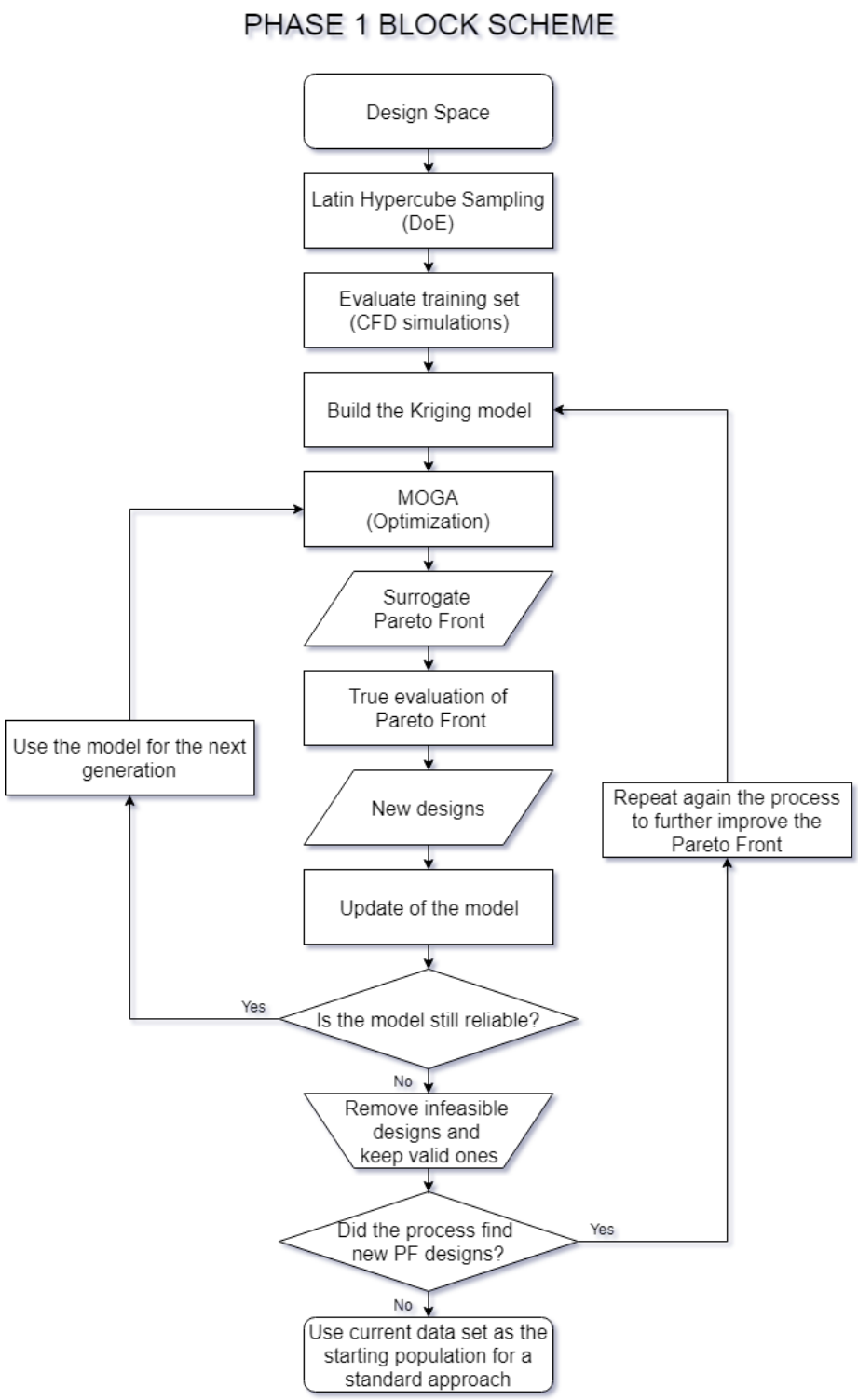



Figure 3.13. Mixed strategy: first step.

# Chapter 4

# Results

The first two sections of chapter 4 analyze the results of the surrogate-based step (Phase 1) and the comparison between the mixed approach and the standard genetic algorithm optimization (Phase 2). Thereafter, it is explained why the mixed approach can be considered a step forward with respect to the classical approach. In the final section, the three most promising designs are analyzed more in detail, considering the chordwise pressure and skin friction distributions and the flow field surrounding the optimized designs.

## 4.1 Reference geometry

All the results shown in this chapter are normalized with respect to those obtained from the symmetric reference geometry of Fig. 4.1.

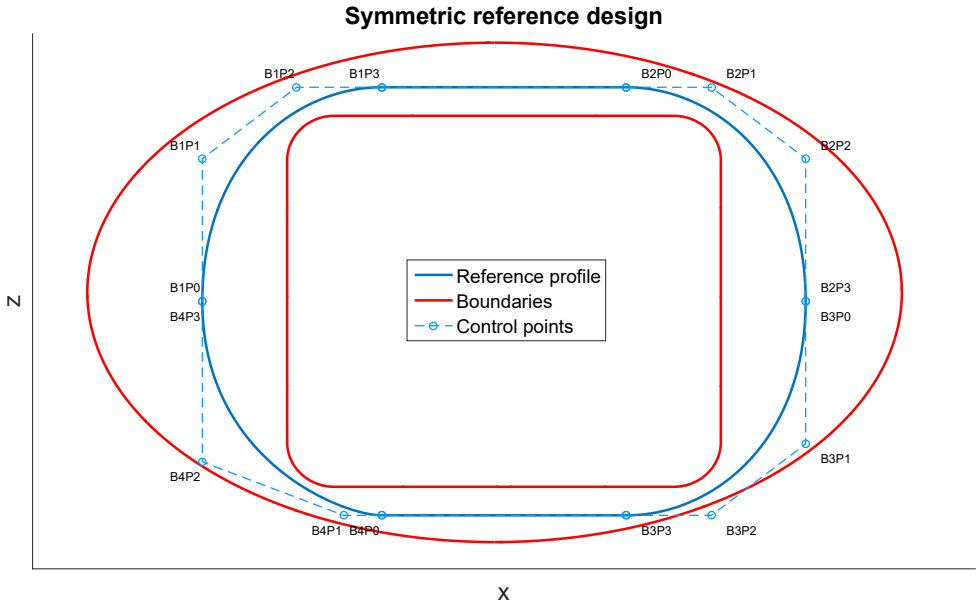



Figure 4.1. Reference design.

## 4.2 Phase 1 results

Phase 1 was the proper surrogate-based optimization step, in which the emulator was used instead of the complete CFD solver to perform the optimization task.

Fig. 4.2 pictures the initial population of 200 designs on the normalized efficiency-drag plane: this represents the starting point of the process. All values are normalized according to Eq. 3.9 and Eq. 3.10, which means that the point corresponding to the reference case has coordinates $(E_{ref,norm}, C_{d,ref,norm}) = (0,0)$.

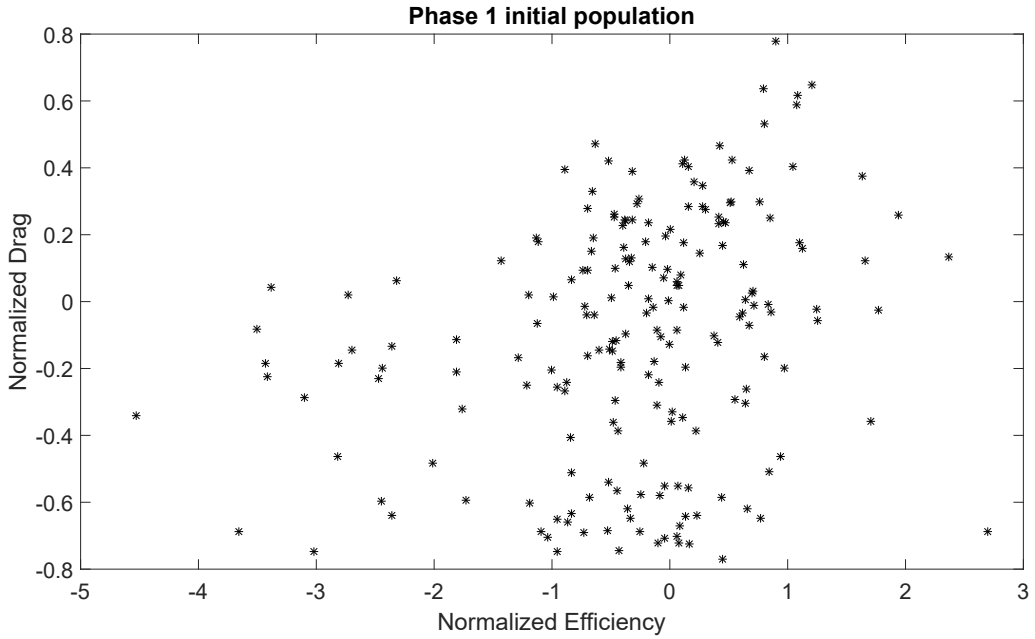



Figure 4.2. Initial population of 200 designs sampled with LHS from the design space.

During Phase 1, a total of ten generations were computed on the surrogate model. However, two iterations of five populations each had to be run. Indeed, after the first five generations, it was necessary to remove infeasible points from the training set, in order to make the model effective again. After the 10th generation, instead, no more interesting points were added. Fig. 4.3 and Fig. 4.4 show the new feasible points found applying this process.

The first three generations discovered points in unexplored regions of the response space: in particular the first two found some potential Pareto solutions. Nevertheless, already after the third one, the model could not produce further valid solutions and generations four and five only added one more feasible design to the tally.

Also the 6th generation of surrogate Pareto solutions explored an interesting region of the response space. The other four, while still adding more feasible points (except for the 7th), did not produce relevant results. Therefore, it was decided to end this first phase after the 10th generation. Up to this point the surrogate-based optimization step added a total of 110 new feasible designs.

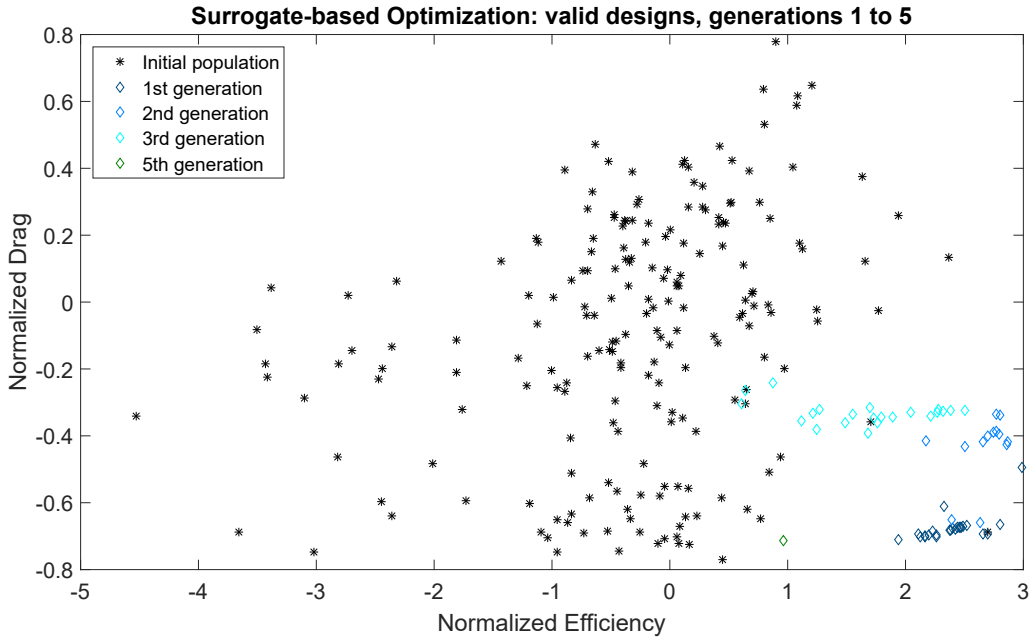



Figure 4.3. Surrogate-based optimization: valid designs proposed in generations 1 to 5. Almost all new proposed designs explored novel regions of the response space. The 4th and 5th generations, however, found only one feasible solution.

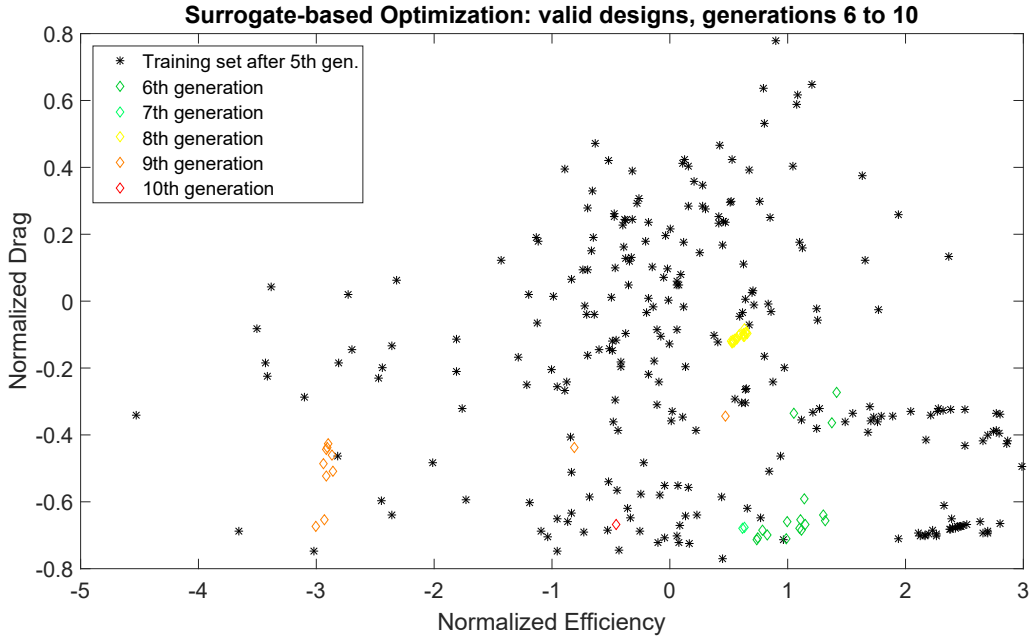



Figure 4.4. Surrogate-based optimization: valid designs proposed in generations 6 to 10. During this second run, only the 6th generation was capable to propose new interesting solutions.

To assess the cost of this first step, it is possible to consider two different parameters: the total number of new true CFD evaluations performed and the number of generations produced to get the final result. The first parameter is an indicator of computational resources used, in particular computational time and operations required to reach the solution. Instead, the second can be regarded as an indicator of the real time needed to complete the process. The difference between these two metrics could be explained with the following example: assuming that it is possible to run in parallel all the simulations of a single generation (very plausible event), then performing 1 or 25 CFD simulations per iteration would require the same real time but a defenitely different amount of computational resources. Of course, if there is a limitation on the number of concurrent simulations at a time, then also the number of true evaluations per generation directly affects the real time spent. However, this setting depends on external factors which can not be easily generalized.

So, during Phase 1, a total of 110 CFD simulations were run, distributed over 9 generations (the 4th of 10 actually did not produce any valid design, therefore no simulations were run there), as summarized in Tab.Fig. 4.1.

Table 4.1. Phase 1 summary of CFD evaluations number.

| Generation | CFD evaluations |
|---|---|
| Gen. 1 | 24 |
| Gen. 2 | 13 |
| Gen. 3 | 22 |
| Gen. 4 | 0 |
| Gen. 5 | 1 |
| Gen. 6 | 16 |
| Gen. 7 | 2 |
| Gen. 8 | 20 |
| Gen. 9 | 11 |
| Gen. 10 | 1 |
| Total | 110 |

Considering that the process was divided in 2 iterations and that the last generation of each of them only added 1 single valid design, probably the total number of generations could have been reduced to 7 (4 and 4, with one having 0 valid designs), with 108 CFD evaluations, or to even just 6 generations (3 and 3) and 97 CFD simulations.

## 4.3 Phase 2 results

During Phase 2, two separate tool chains were run. The first, labelled "Mix", carried on the mixed strategy by using as initial population the 200 fittest designs of the 310 from the end of Phase 1; the worst 110 dominated candidates were left out. The second chain, labelled "Baseline", started its optimization from the original 200 designs initial population. The starting generations are sketched in Fig. 4.5. The green diamonds indicate Pareto solutions of the current data set.

As shown, the initial population of the Mix chain is obviously characterized by a higher number

of optimal designs, having already undergone an optimization process in Phase 1. It is, indeed, already possible to sketch a first Pareto Front in the lower right region of the response space, where designs characterized by high efficiency and low drag are found. In the Baseline starting population, instead, only two points are Pareto solutions.

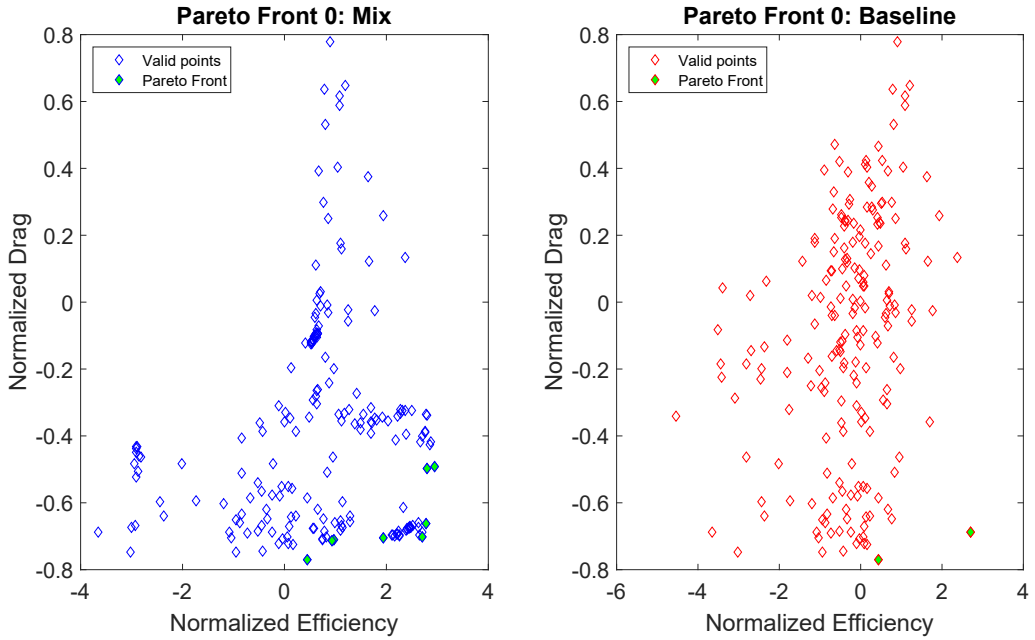



Figure 4.5. Generations 0 at comparison: the Mix chain relies on the advantage given by the optimization performed in Phase 1.

The first couple of generations are not really able to introduce more optimal designs, as still a lot of dominated designs are used to produce the new children (and, in particular, the very first generation is just a subset of the original population 0). Therefore, during these first steps, the population of fit designs just decreases in size without showing particular progresses. Nevertheless, as valid designs are evaluated with the CFD solver, this stage is still time-consuming.

The Baseline progression can be seen in Fig. 4.6, in which generations 0, 5, 10, 15 and 19 are sketched: with the obvious exception of the first one, the four remaining generations only show designs that are new with respect to previous populations. The designs introduced up to the 5th iteration are still far from being the ideal ones, while consistent improvements appear from the 10th generation on. The points belonging to the 19th generation are all close to the Pareto Front, thus are all nearly optimal solutions. It is also possible to note that the number of designs in each generation decreases progressively, leading to a final population of just 76 points.

A similar explanation works also for Fig. 4.7, where generations 0, 2, 3, 4 and 5 of the Mix chain are compared: only designs introduced from the 3rd iteration are starting to explore new interesting regions of the response space. Also the 4th and 5th generation further continue this trend.

In both plots, the legend shows also the total number of true function evaluations performed on the CFD solver during Phase 2 up to the corresponding generation.

As can be noticed in Tab. 4.2, the number of CFD evaluations is exactly the same for generations 11 and 12 of the Baseline: this tool chain crashed and was forced to start again from populations 11. Thus generation 12 is just subset of the previous one, without introducing new designs.

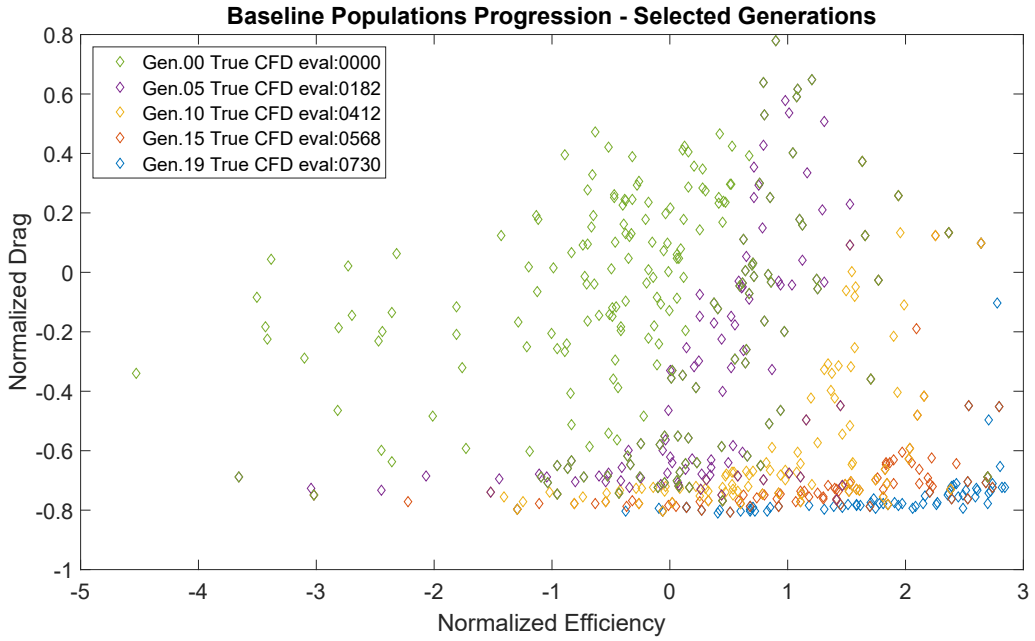



Figure 4.6. Baseline chain progression: generations 0, 5, 10, 15 and 19 at comparison. As the optimization continues, the populations reduce in size but get closer to the optimal solutions, forming a well defined Pareto front.

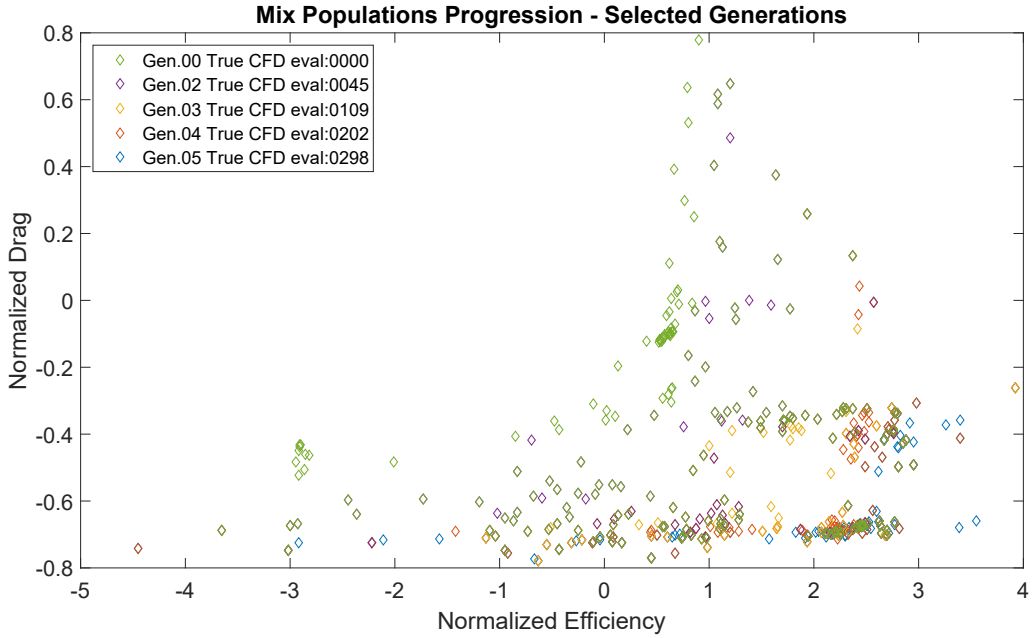



Figure 4.7. Mix chain progression: generations 0, 2, 3, 4 and 5 at comparison. As the optimization continues, the populations reduce in size but get closer to the optimal solutions, forming a well defined Pareto front.

As for population 0, also all the following generations show that the Mix chain has an advantage with respect to the Baseline, when generations with the same number are considered.

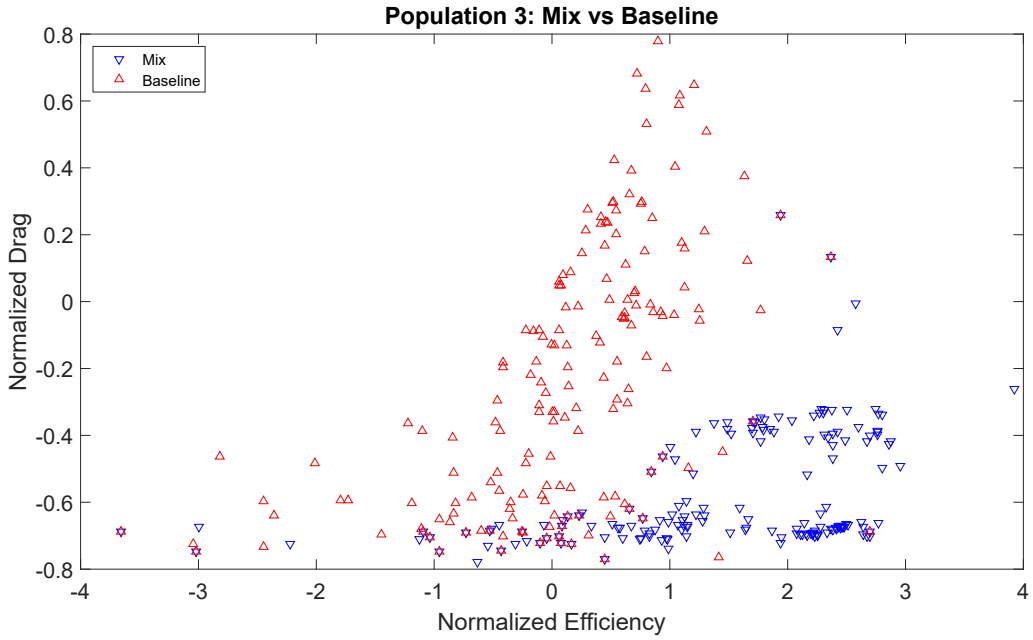



Figure 4.8. Comparing generations 3: the Mix chain outperforms the Baseline chain.

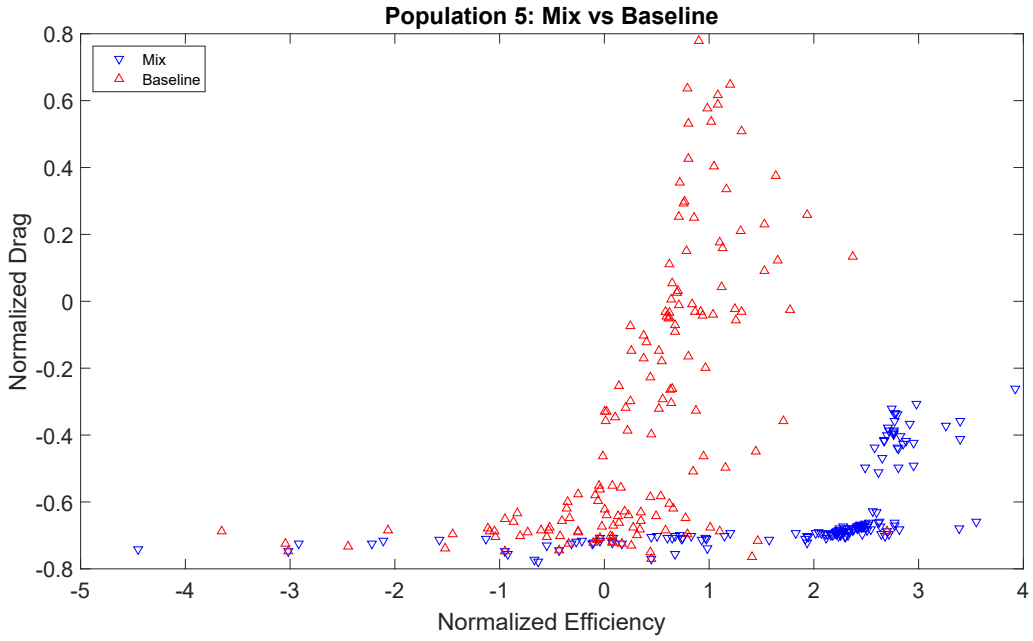



Figure 4.9. Generations 5 at comparison: a Pareto Front starts forming in the Mix chain.

In Fig. 4.8, for example, the 3rd generations are directly compared. Both have now a reduced total number of designs (172) because of the shrinking percentage set in the options of the genetic algorithm. Apart for very rare cases, the Mix population outperforms the Baseline. In this latter, most of the designs are still far from the ideal Pareto Front.

Similarly, the 5th Mix (now with 151 total designs) and Baseline (155 designs) generations are put in comparison in Fig. 4.9. The designs from the Mix chain start forming a well defined Pareto Front and have better objective functions than those of the Baseline. That is especially true when considering those designs with higher lift-to-drag ratio.

Regarding the Baseline population, still a high amount of designs is characterized by a drag coefficient bigger than the reference value, that is with a positive normalized drag.

To be able to find better results in the Baseline than in the Mix chain, it is necessary to compare staggered generations. For example, Fig. 4.10 shows the 1st Mix generation together with the 7th population from the Baseline. At this point, the Baseline is able to propose designs with lower drag coefficient values; however, these are not as competitive when it goes down to efficiency.

The difference in generation numbers also implies different number of true CFD evaluations: up to the 7th generation, the Baseline chain totals 268 new evaluations during Phase 2, while still no new simulation was run at this stage for the Mix chain.

As mentioned, because of the way DAKOTA implements the MOGA method, the 1st generation is just a shrinked subset of the initial one. Therefore, the results obtained at the end of Phase 1 with the surrogate-based approach are comparable to those achieved after seven generations with the standard fully CFD-based method.

Similarly, Fig. 4.11 shows the 3rd Mix generation (109 CFD simulations in Phase 2) compared with the 12th Baseline (450 simulations). Again, some of the Baseline solutions show a slightly lower drag coefficient value, but those of the Mix chain have still higher efficiency. Nevertheless, it is now possible to spot the formation of a defined Pareto Front also in the Baseline chain.

The 5th Mix generation (298 true function evaluations during Phase 2) is compared to the 14th Baseline (534 evaluations) in Fig. 4.12. In this case, the separation between the two populations is stronger from the drag coefficient point of view. Moreover, the Baseline becomes competitive regarding the lift-to-drag ratio as well.

Finally, Fig. 4.13 compares the very last populations reached by both the chains: the 5th Mix generation (for a total of 298 simulations) against the 19th Baseline (total of 730 simulations). The original plan was to run 20 generations on each chain. However, because of technical problems which affected the SuperMUC, it was possible to reach only the 5th Mix and the 19th Baseline population.

From this plot, it is evident that the Baseline solutions are now much better than those of the Mixed approach, even though there is still room for improvement from the efficiency point of view. To get to these results, however, the Baseline chain required 14 generations and 432 evaluations more than the Mix chain during Phase 2.

Regarding the computational effort, Tab. 4.2 summarizes the number of CFD evaluations up to each generation for both the chains. Coloured numbers indicate generations with comparable Pareto Fronts across the Mix and the Baseline chains.

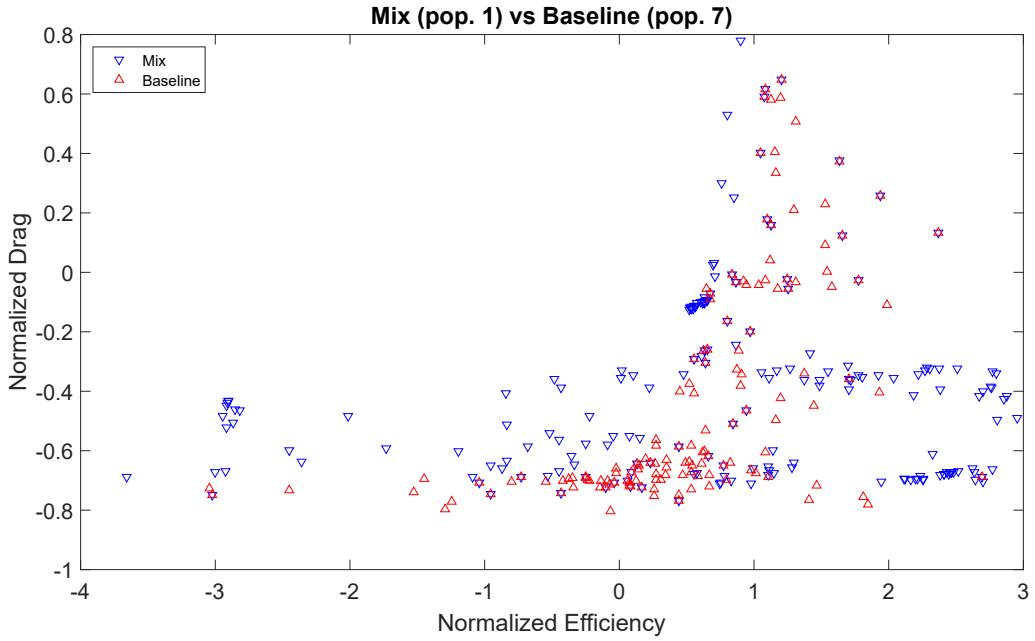



Figure 4.10. Mix generation 1 vs. Baseline generation 7. The desigs from the Mix one have in general higher efficiency and comparable drag.

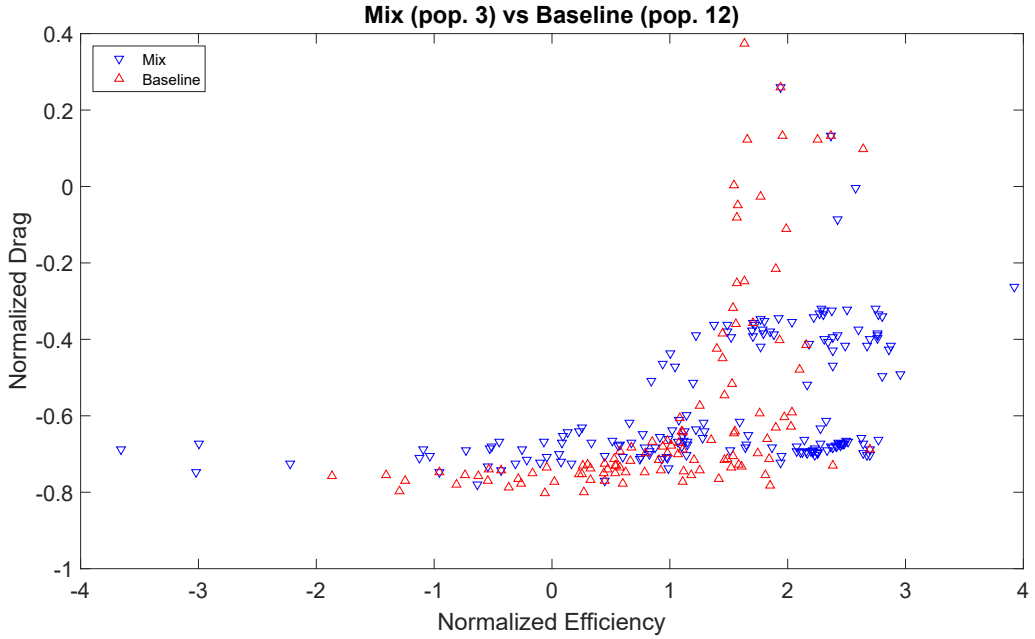



Figure 4.11. Mix generation 3 vs. Baseline generation 12. The Baseline optimal designs are characterized by low drag; however, their efficiency is, in general, lower than the Mix chain case.

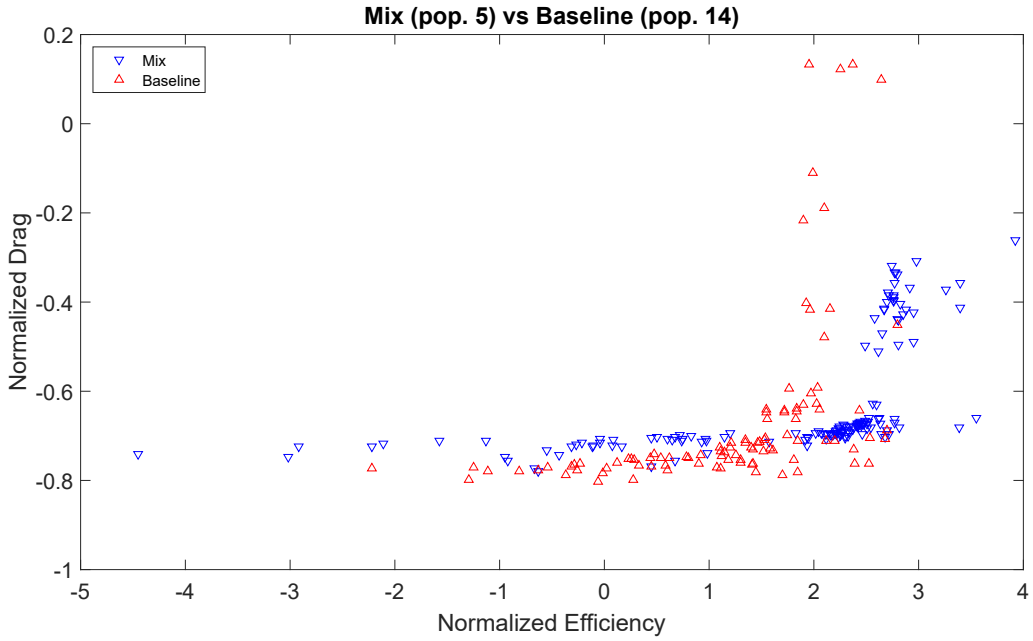



Figure 4.12. Mix generation 5 vs. Baseline generation 14. At this point, the Baseline chain is almost more competitive than the Mix one.

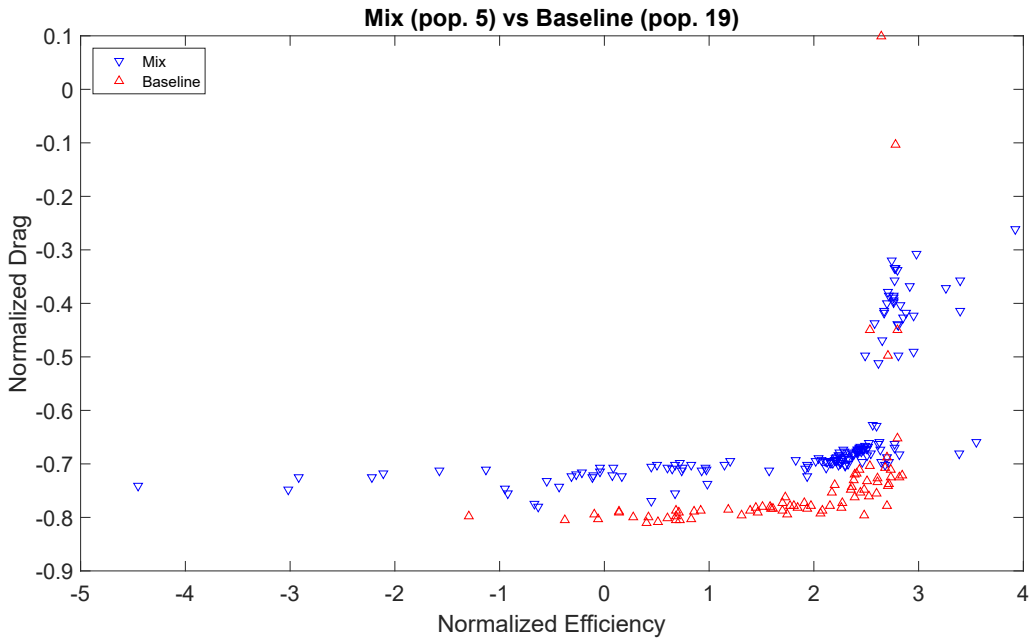



Figure 4.13. Mix generation 5 vs. Baseline generation 19. The last Baseline population generally outperforms the last Mix generation, from both the drag and the efficiency point of view, except for a few isolated cases.

Table 4.2. Phase 2 summary of CFD evaluations number.

| Generation | CFD Mix | CFD Baseline |
|---|---|---|
| Gen. 1 | 0 | 0 |
| Gen. 2 | 45 | 60 |
| Gen. 3 | 109 | 105 |
| Gen. 4 | 202 | 144 |
| Gen. 5 | 298 | 182 |
| Gen. 6 | - | 222 |
| Gen. 7 | - | 268 |
| Gen. 8 | - | 310 |
| Gen. 9 | - | 359 |
| Gen. 10 | - | 412 |
| Gen. 11 | - | 450 |
| Gen. 12 | - | 450 |
| Gen. 13 | - | 489 |
| Gen. 14 | - | 534 |
| Gen. 15 | - | 568 |
| Gen. 16 | - | 610 |
| Gen. 17 | - | 647 |
| Gen. 18 | - | 694 |
| Gen. 19 | - | 730 |
| Gen. 20 | - | - |
| Total | 298 | 730 |

## 4.4 Comments on results from Phase 1 and Phase 2

The comparison between Mix and Baseline results shows that the Baseline chain needs a higher number of iterations to reach similar results to those of the Mix chain. This means that a standard method, in which genetic algorithm optimization is directly applied to the CFD solver, requires a higher number of true function evaluations with respect to the mixed approach proposed in this work.

However, as written in Sec. 4.2, it has to be taken into account also the computational effort needed by the mixed approach to go through Phase 1. Therefore, in Tab. 4.3, the number of true CFD simulations performed in Phase 1 (110) has been added to the amount of evaluations for Phase 2. Moreover, the 200 initial training designs have been added to both the tallies.

Table 4.3. Final summary of CFD evaluations number.

| Generation | CFD Mix | CFD Baseline |
|---|---|---|
| Gen. 1 | 310 | 200 |
| Gen. 2 | 355 | 260 |
| Gen. 3 | 419 | 305 |
| Gen. 4 | 512 | 344 |
| Gen. 5 | 608 | 382 |
| Gen. 6 | - | 422 |
| Gen. 7 | - | 468 |
| Gen. 8 | - | 510 |
| Gen. 9 | - | 559 |
| Gen. 10 | - | 612 |
| Gen. 11 | - | 650 |
| Gen. 12 | - | 650 |
| Gen. 13 | - | 689 |
| Gen. 14 | - | 734 |
| Gen. 15 | - | 768 |
| Gen. 16 | - | 810 |
| Gen. 17 | - | 847 |
| Gen. 18 | - | 894 |
| Gen. 19 | - | 930 |
| Gen. 20 | - | - |
| Total | 608 | 930 |

From the Tab. 4.3, it is possible to observe that, even when considering the first stage of the surrogate-based approach, the mixed method requires a lower number of true function evaluations. Regarding the number of generations, in order to have comparable results, the two methods need the same amount of populations: 15 for the mixed approach (including those from Phase 1) and 14 for the standard method. However, as mentioned at the end of Sec. 4.2, the duration of Phase 1 could have been shortened. This could have further reduced both the number of generations and CFD evaluations of the mixed approach.

It is also interesting to note that the number of valid candidates evaluated at each iteration is higher in the Mix chain than in the Baseline. This could hint that, when combining the characteristics of already nearly optimal designs, it is more probable to obtaing feasible children designs. In the Baseline chain, as long as the members of the current population are spread across the response space, the valid candidates are just a small fraction of those proposed. However, toward the last generations (that have a reduced population), the number of feasible children designs is comparable to, and sometimes bigger than, the number of those violating the constraints. This phenomenon could somehow penalize the mixed approach as the number of generations increases.

Nevertheless, the previous results show the potential of using a surrogate model, not only in a

full surrogate-based approach, as in Phase 1, but also in a mixed strategy.

## 4.5 Flow field evaluations

In the next pages, the most promising designs from the Mix chain are analysed. In particular, three candidates belonging to the final Pareto Front have been chosen: the one with lowest drag ("Minimum drag design", D1, Fig. 4.15), the one with highest efficiency ("Maximum efficiency design", D2, Fig. 4.16) and one characterized by a good compromise between the two coefficients ("Trade-off design", D3, Fig. 4.17). The position of the three candidates on the Pareto Front can be seen in Fig. 4.14.

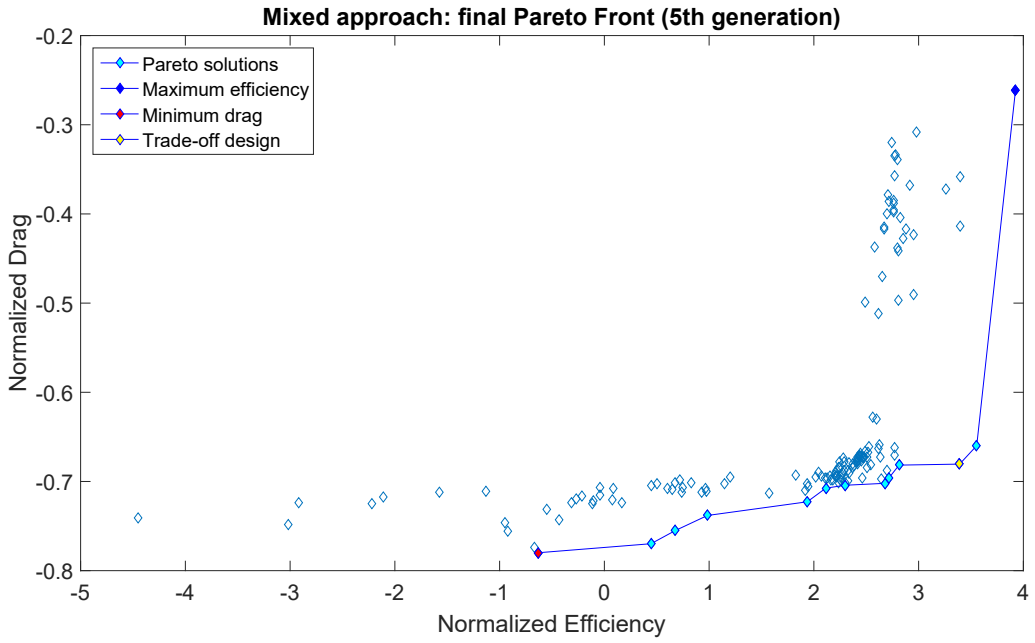



Figure 4.14. Final Pareto Front of the Mix chain: three promising designs are selected.

Tab. 4.4 reports the normalized values of efficiency and drag coefficients with respect to the reference values. All three candidates achieved a reduction in drag coefficient with respect to the reference design. In particular, the minimum drag design reduces drag by 78%; however, its efficiency is actually lower than the reference value. The maximum efficiency candidate increases lift-to-drag ratio by 392%, but achieves a small reduction in drag. The trade-off design, instead, is in between the performance of the other two.

Table 4.4. Performance of the three designs selected from final Pareto Front.

| | $E_{norm}$ | $C_{d,norm}$ |
|---|---|---|
| Minimum drag design (D1) | -0.6333 | -0.7799 |
| Maximum efficiency design (D2) | 3.9245 | -0.2617 |
| Trade-off design (D3) | 3.3915 | -0.6805 |

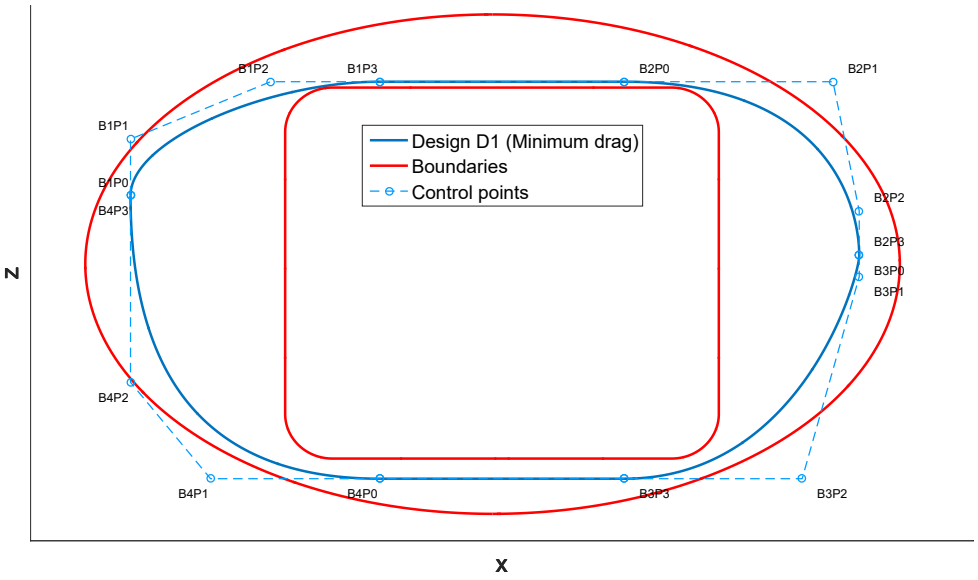



Figure 4.15. Minimum drag candidate D1.

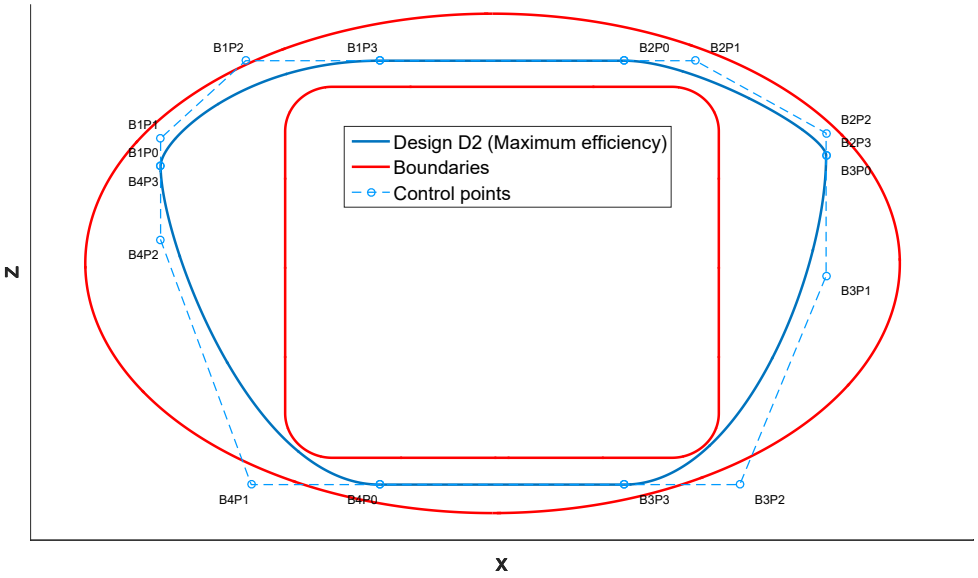



Figure 4.16. Maximum efficiency candidate D2.

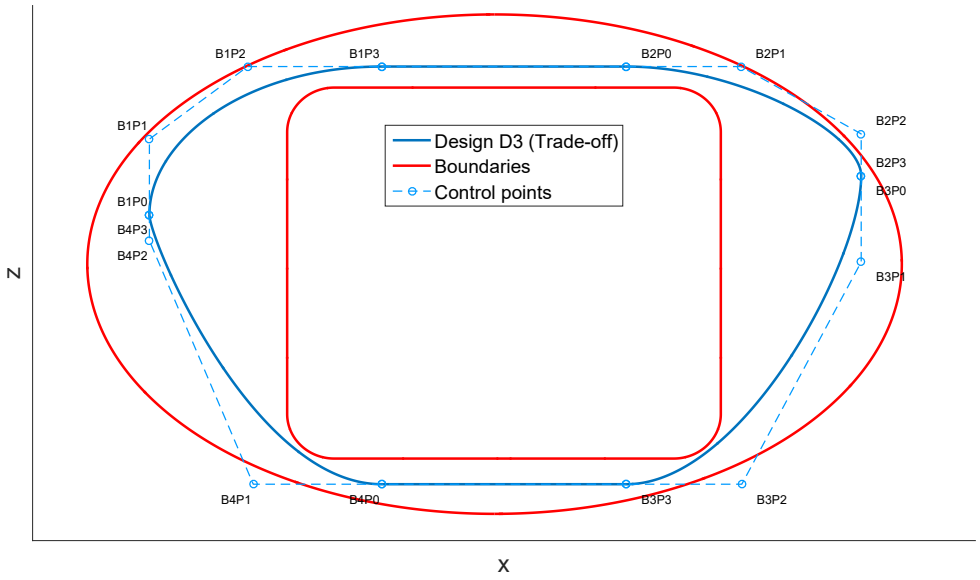



Figure 4.17. Trade-off candidate D3.

Fig. 4.18 shows the comparison of the pressure coefficient ($C_p$) chordwise distribution between the selected designs (red) and the reference profile (blue). Both the advancing and the retreating blade cases are plotted. The dashed lines represent the distribution on the top of the section, while the solid lines map the $C_p$ distribution on the bottom part of the profile. In the retreating blade case, the flow is reversed, thus the pressure distribution looks mirrored with respect to the advancing blade condition.

Similarly, Fig. 4.19 shows the skin friction coefficient ($C_f$) distribution along the normalized chord, comparing the optimized designs to the symmetric reference geometry.

Concerning the minimum drag design (D1) in the advancing blade case, the suction peaks on both the upper and lower sides of the section are smaller than those of the reference geometry. This is caused by the shape of the optimized design, which allows for a reduced flow deflection, hence keeping the flow static pressure higher than in the reference case, for both the fore and aft regions of the profile. It is possible to detect the region of the flow separation in correspondence to the plateau of the $C_p$ distribution at the rear end of the chord: in the optimized design, the flow separates earlier on the bottom side. The separation point can be also assessed by looking at the skin friction coefficient plot: where the value for $C_f$ goes to zero, there the separation takes place. This point is consistent with the position of the flat line in the pressure distribution.

Similarly, the minimum drag design (D1) shows lower pressure coefficient peaks also in the retreating blade case, which is characterized by a reversed flow condition. In this case, the flow separation takes place closer to the leading edge (which is now at $x/c = 1$), compared to the reference profile. However, the minimum drag design shows a stronger pressure recovery, which can be associated to a reduction of the drag generated by the profile. This is also valid for the advancing blade case.

For the maximum efficiency design (D2), the frontal suction peaks are comparable to those of the reference design for the advancing blade case. However, at the rear end, the downforce caused by the bottom part of the profile is smaller than for the reference case. Moreover, it is also much

smaller than the lift produced by the top part. This leads to a high lift-to-drag ratio for candidate D2. The pressure recovery is comparable to the one for the reference case, which means that the gain in efficiency is almost all due to the higher lift produced by the candidate.

A comparable pressure recovery can be seen in the retreating blade case as well, where also the flow separation on the bottom part takes place more toward the leading edge than for the reference case. Hence, the reduction in drag is not as strong as for the minimum drag design and candidate D2 is characterized by a drag coefficient which is very close to the reference value.

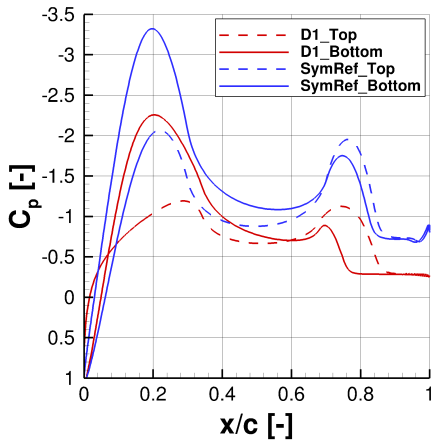
The trade-off design (D3) places itself in between the two previously analysed candidates. As can be seen from the Pareto Front plot, the drag coefficient is not so far from to the minimum drag design value. This reflects in very similar $C_p$ distribution plots between candidates D1 and D3 for the retreating blade case. Furthermore, the lift-to-drag ratio is very close to the maximum efficiency profile value for the advancing blade case. When comparing the two $C_p$ distributions of designs D2 and D3, the trade-off profile shows a better pressure recovery, but a lower lift with respect to design D2. Hence, the high efficiency is mainly due to the low drag produced by design D3.

In Fig. 4.20, the pressure fields and the streamlines for the three designs are sketched. Fig. 4.21, instead, shows the corresponding velocity fields, in which the velocity magnitude given by the x and z components is normalized over the freestream velocities $V_{adv}$ and $V_{ret}$ for the advancing and retreating blade case, respectively. Both the pressure and the velocity fields indicate the presence of a Von Kàrmàn vortex street behind design D2, in both the advancing and retreating blade cases. The vortical flow field in the wake regions of the maximum efficiency design can be also observed in Fig. 4.21, where the light blue regions indicate smaller flow velocities and yellow zones are characterized by higher flow velocity.

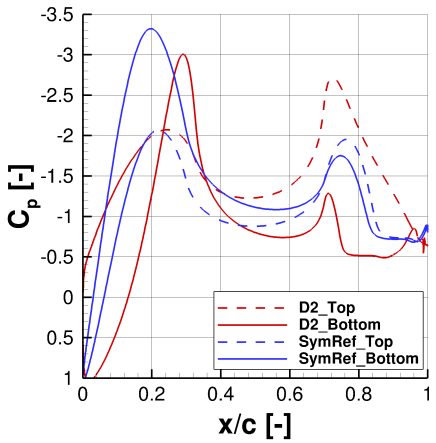
The presence of strong vortices behind the profile creates high suction peaks increasing the drag on the aft part of the body. Moreover, the big distance along the z direction between vortices implies a thicker wake behind the profile, which is related to higher drag ([31]). However, for the advancing blade case, the flow on the upper part of the profile is overall faster than the flow on the bottom part. This generates high lift on the profile, which is responsible of the high value of lift-to-drag ratio of this design.

For the advancing blade case, design D1 shows a very stretched dead water region behind the trailing edge and no vortex shedding takes place. However, the flow is faster on the bottom part of the profile, so the downforce is bigger than the lift from the top part, as also suggested from the $C_p$ plot. This gives a negative efficiency to this profile. In the retreating blade case, the dead water region is less elongated and it is possible to note some small perturbations in the wake about two chord lengths downstream. However, this does not affect the drag generated by this profile.

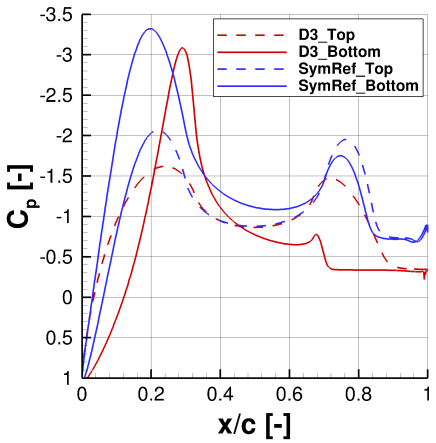
Finally, the trade-off design is showing again intermediate characteristics between the other two designs. The low drag coefficient for the retreating blade case is guaranteed by the absence of the downstream vortex shedding, as in the minimum drag profile D1. In the advancing blade case, instead, the lift due to the fast flow on the top part of the profile, together with a low drag from a vortex-free wake, gives the high value of lift-to-drag ratio that characterizes design D3. This makes the trade-off design a very compelling candidate.



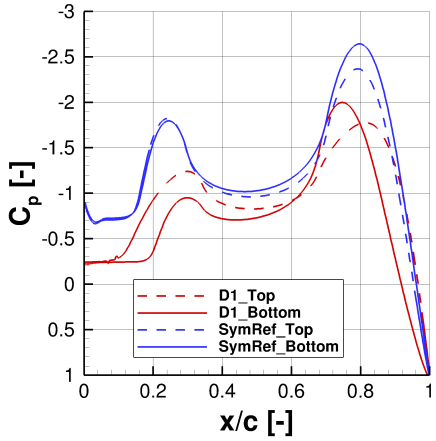

(a) Minimum drag design, advancing blade.



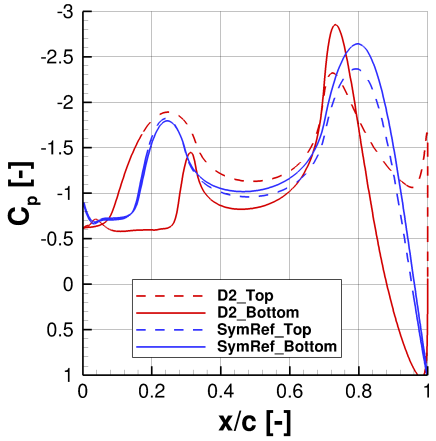

(b) Maximum efficiency design, advancing blade.



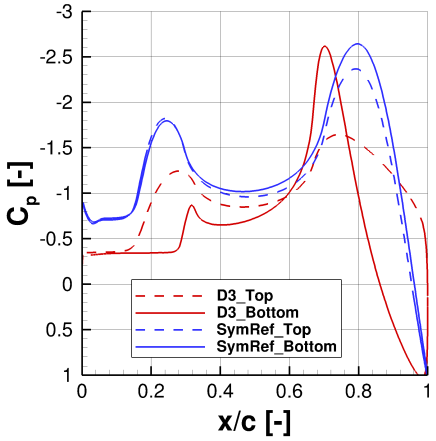

(c) Trade-off design, advancing blade.




(d) Minimum drag design, retreating blade.

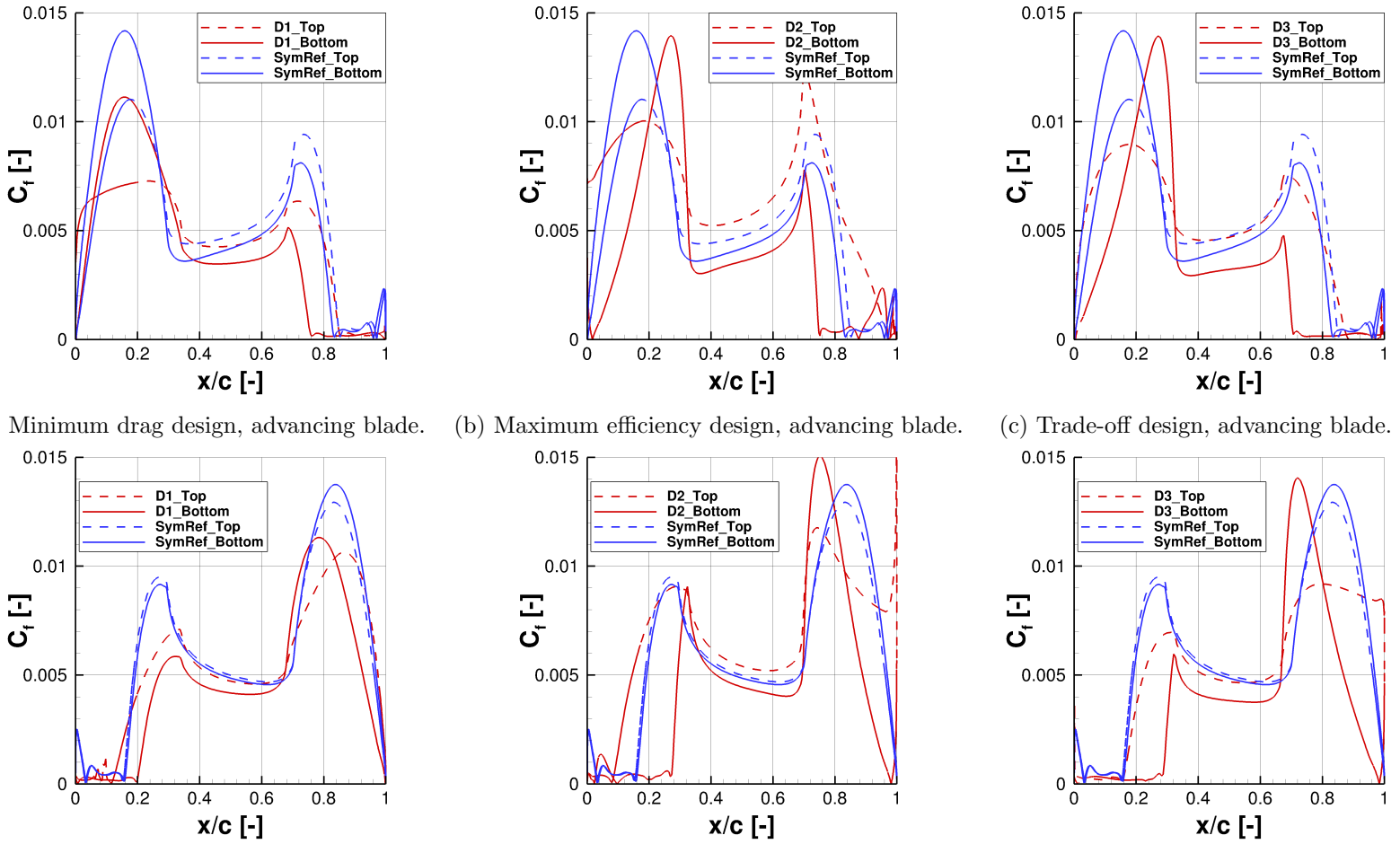



(e) Maximum efficiency design, retreating blade.




(f) Trade-off design, retreating blade.

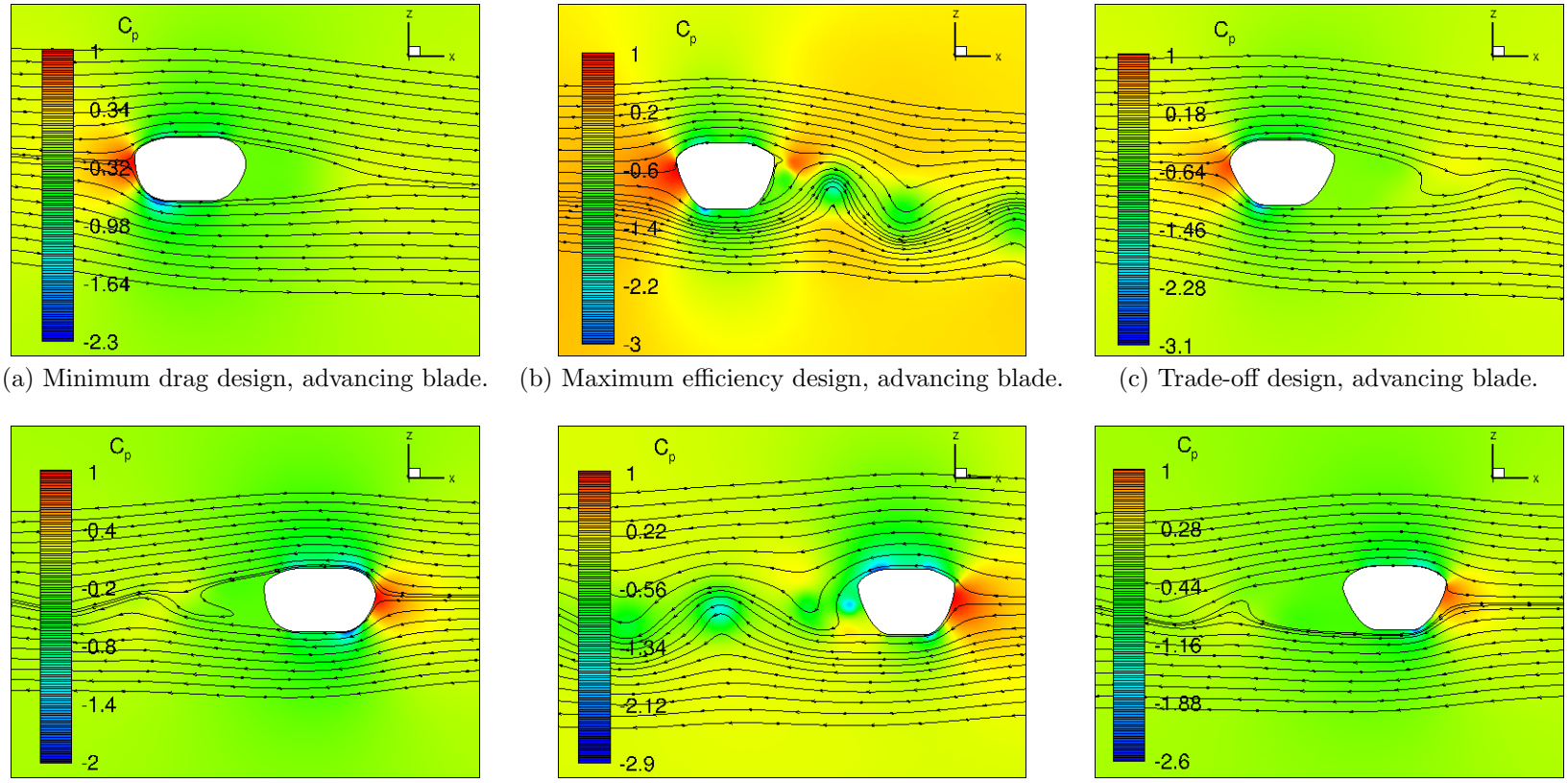
Figure 4.18. Comparison of the chordwise $C_p$ distribution between the three selected designs and the symmetric reference design. The upper row shows the advancing blade case, while the retreating blade case is sketched on the lower row.




(a) Minimum drag design, advancing blade. (b) Maximum efficiency design, advancing blade. (c) Trade-off design, advancing blade.

(d) Minimum drag design, retreating blade. (e) Maximum efficiency design, retreating blade. (f) Trade-off design, retreating blade.

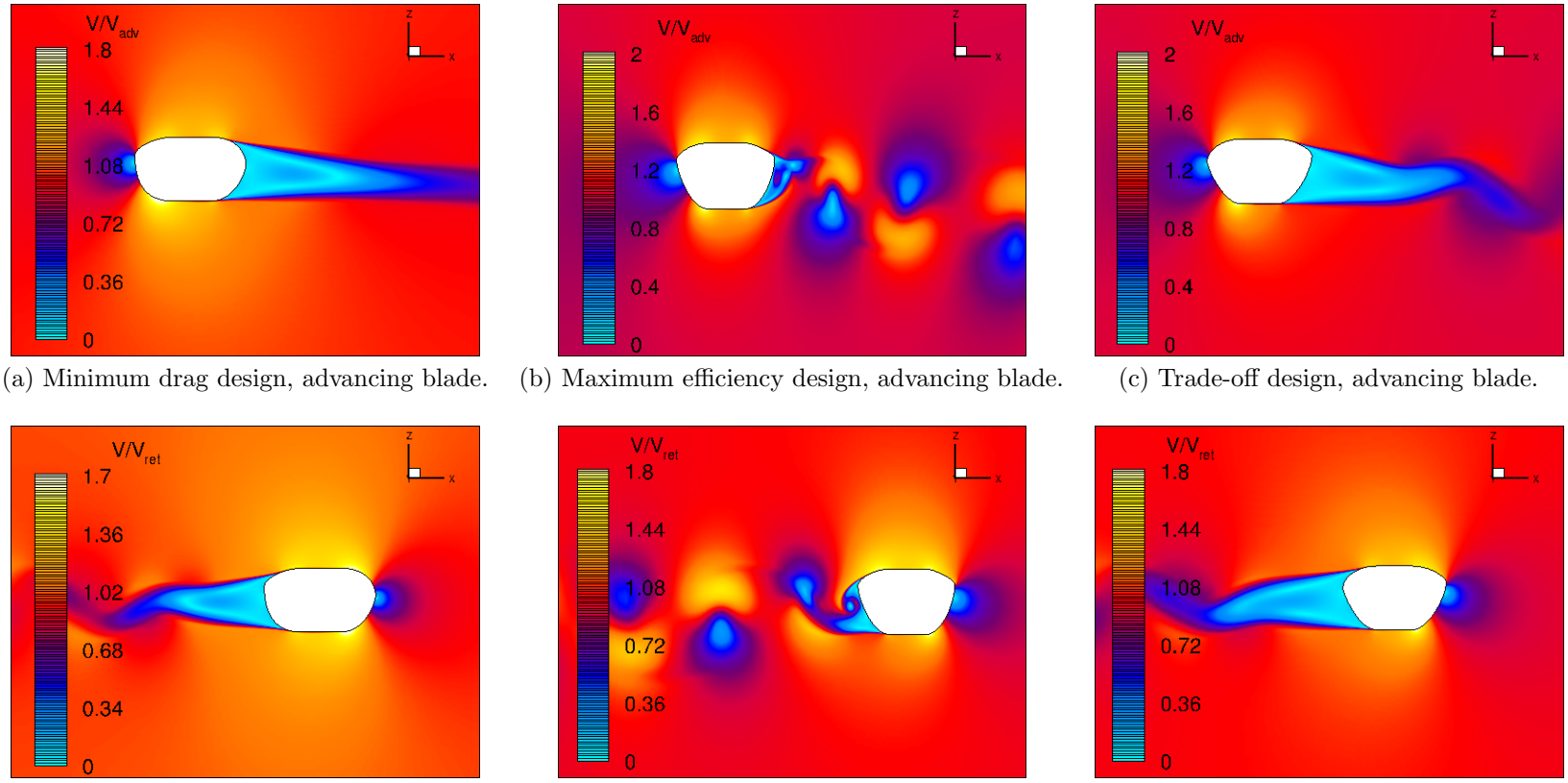
Figure 4.19. Comparison of the chordwise $C_f$ distribution between the three selected designs and the symmetric reference design. The upper row shows the advancing blade case, while the retreating blade case is sketched on the lower row.




(a) Minimum drag design, advancing blade. (b) Maximum efficiency design, advancing blade. (c) Trade-off design, advancing blade.

(d) Minimum drag design, retreating blade. (e) Maximum efficiency design, retreating blade. (f) Trade-off design, retreating blade.

Figure 4.20. Pressure field and streamlines around the optimized designs. High pressure areas are represented in red, while the blue colour denote low pressure regions.




(a) Minimum drag design, advancing blade. (b) Maximum efficiency design, advancing blade. (c) Trade-off design, advancing blade.

(d) Minimum drag design, retreating blade. (e) Maximum efficiency design, retreating blade. (f) Trade-off design, retreating blade.

Figure 4.21. Normalized velocity field around the optimized designs. Light blue regions are characterized by the lowest velocities, while, in the yellow zones, the flow moves faster.

# Chapter 5

# Conclusions

This master thesis aimed at performing an aerodynamic design optimization of a rotor blade-sleeve section, which belongs to the Airbus' RACER demostrator, by adopting a surrogate-based approach. This method was used in order to reduce the time and computational effort of the whole procedure with respect to a standard process which performs all the function evaluations on the CFD solver. The two objective functions were the maximization of the efficiency coefficient and the minimization of the drag coefficient of the advancing and retreating blade, respectively. Therefore, to perform the optimization, a multi-objective genetic algorithm was used.

The surrogate-based optimization approach relies on the use of a cheap emulator to approximate a complex computational model or experiment, so as to progress faster through the optimization task. To get an appropriate accuracy of the surrogate model, this has to be updated iteratively performing a reduced number of targeted CFD evaluations. The update of the model is carried out in a Bayesian framework, where a Gaussian Process emulator (also called Kriging model) gives the most probable prediction of the output together with its uncertainty, rather than identifying the best-fit of the response.

Before the actual implementation, some test cases were run to assess the validity of the surrogate-based approach and the suitability of the Gaussian Process to model the problem under investigation. Thereafter, because of some issues with the optimization software, it was decided to implement a mixed strategy. The first step consisted in a proper surrogate-based optimization, while, in the second, a standard genetic algorithm optimization applied to the CFD solver was employed. The first step (Phase 1) provided a foundation of promising solutions from which the Phase 2 genetic optimization could rapidly build up towards further optimal designs.

During Phase 2, the obtained results showed that the mixed approach allowed to find valuable optimal designs within a limited number of generations and performing a smaller number of CFD simulations. This, in the economy of a genetic algorithm optimization, could lead to a sensible reduction in computational effort required for the process. However, also the cost of all the Phase 1 operations has to be considered when comparing the surrogate-based approach to the standard one. In this context, the advantages of the mixed strategy are partially reduced. Nevertheless, this method has shown a good potential.

In the last section of the work, the most promising optimized designs were analysed evaluating their flow field and aerodynamic coefficients distributions. They all showed substantial improvements with respect to the symmetric geometry used as a benchmark during the study.

To conclude, the surrogate-based optimization approach is a flexible and widely applicable method, which has the potential to significantly reduce the computational effort in optimization tasks. This is particularly true when considering problems that involve a large number of expensive computer simulations. However, using a surrogate model, as it is implemented in DAKOTA, shows

its critical issues in the handling of non-linear constraints, especially when the considered problem is of high dimensionality like in this work. Such an issue could be solved by allowing the user to fully specify the non-linear constraints equations, so as to consider only feasible designs during the optimization. A similar solution would have made Phase 1 much more efficient and, ideally, would have avoided the need of Phase 2.

Concerning the use of a genetic algorithm for an aerodynamic design optimization, the method has proved to be very effective in quickly finding promising designs, which can be used as a base for further and more refined optimizations.

# Bibliography


[1] World Bank Open Data, Air transport, passengers carried "http://data.worldbank.org/indicators/IS.AIR:PSGR" [Online]

[2] IATA, 2036 Forecast Reveals Air Passengers Will Nearly Double to 7.8 Billion "http://www.iata.org/pressroom/pr/Pages/2017-10-24-01.aspx" [Online]

[3] European Aviation Environmental Report 2016, ISBN: 978-92-9210-197-8, doi: 10.2822/385503. Chapter 1

[4] CLEAN SKY 2 JOINT UNDERTAKING FIRST AMENDED BI-ANNUAL PLAN and BUDGET 2016-2017, page 16

[5] Eurocopter $X^3$, Wikipedia "http://en.wikipedia.org/wiki/Eurocopter_$X^3$" [Online]

[6] Airbus, Clean Sky 2, "http://www.airbus.com/innovation/clean-sky-2.html" [Online]

[7] Clean Sky Projects: FURADO Project, "http://www.cleansky-projects.tum.de/en/furado/l" [Online]

[8] DAKOTA, A Multilevel Paralle Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.8 User's Manual, generated on May 8, 2018. Sandia National Laboratories, Albuquerque, NM, USA.

[9] Klammer, Martin; Nikolaj Dybowski, J.; Hoffmann, Daniel; Schaab, Christoph (2015): Pareto Optimization Identifies Diverse Set of Phosphorylation Signatures Predicting Response to Treatment with Dasatinib. PLOS ONE. Figure. https://doi.org/10.1371/journal.pone.0128542

[10] Lecture notes from "adl.stanford.edu/aa222/lecture_notes_files/chapter6_gradfree.pdf" [Online]

[11] Hornby, G. S., Globus, A., Linden, D. S., Lohn, J. D., Automated Antenna Design with Evolutionary Algorithms. American Institue of Aeronautics and Astronautics, 2006

[12] C. Smith, Surrogate Models for Aerodynamic Performance Prediction. University of Surrey, 2015

[13] Zhong-Hua Han and Ke-Shi Zhang (2012). Surrogate-Based Optimization, Real-World Applications of Genetic Algorithms, Dr. Olympia Roeva (Ed.), ISBN: 978-953-51-0146-8, InTech, Available from: http://www.intechopen.com/books/real-world-applications-of-genetic-algorithms/surrogate-based-optimization

[14] B. J. Brewer. STATS 331, Introduction to Bayesian Statistics

[15] Metha, P. M., Walker, A., Lawrence, R., Higdon, D., Koller, J., Modelling Satellite Drag Coefficients With Response Surfaces, Advances in Space Research (2014), doi: http://dx.doi.org/10.1016/j.asr.2014.06.033

[16] Chuong B. Do (updated by Honglak Lee), Gaussian Processes, November 22, 2008, "cs229.stanford.edu/section/cs229-gaussian_processes.pdf" [Online]

[17] C. E. Rasmussen, and C. K. I. Williams. Gaussian Processes for Machine Learning. The MIT Press, 2006, ISBN 026218253X.

[18] R. L. Iman. Latin Hypercube Sampling. DOI: 10.1002/9780470061596.risk0299

[19] DAKOTA, A Multilevel Paralle Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.8 Theory

Manual, generated on May 8, 2018. Sandia National Laboratories, Albuquerque, NM, USA.
[20] P. Pölzlbauer. AASD - Automated Aerodynamic Shape Development, User Manual
[21] ANSYS Inc.: Ansys icem cfd user's manual. User Guide v17.0, Ansys Inc., Canonsburg, PA, USA (2016)
[22] ANSYS Inc.: Fluent. User Guide v17.0, Ansys Inc., Canonsburg, PA, USA (2016)
[23] Gerhold, T.: Overview of the hybrid rans code tau. MEGAFLOW - Numerical Flow Simulation for Aircraft Design 89 of Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer Verlag, 81-92 (2005)
[24] Leibniz Supercomputing Centre, "https://www.lrz.de/services/compute/supermuc" [Online]
[25] Grawunder, M., Reß, R., Stein, V., Breitsamter, C., Adams, N.: Flow Simulation of a Five-Bladed Rotor Head, pp. 235–243. Springer International Publishing, Cham (2014). DOI 10.1007/978-3-319-03158-3 24. URL https://doi.org/10.1007/978-3-319-03158-3 24
[26] Pölzlbauer, P., Desvigne, D. & Breitsamter, C. CAES Aeronaut J (2018). http://doi.org/10.1007/s13272-018-0341-0
[27] Desvigne, D., Alfano, D.: Rotor-head/fuselage interactional effects on helicopter drag: Influence of the complexification of the rotor-head geometry. In: European Rotorcraft Forum. Moscow, Russia (2013)
[28] Menter, F., Two-equation eddy-viscosity turbulence models for engineering applications. AIAA Journal 32(8), 1598–1605 (1994). DOI 10.2514/3.12149. URL https://doi.org/10.2514/3.12149
[29] C. M. Fonseca, and P. J. Fleming. Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction. University of Sheffield, UK, 1995
[30] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. Evolutionary Algorithms for Solving Multi-Objective Problems, second edition. Springer, ISBN 978-0-387-33254-3
[31] G. Buresti. Vortex shedding from bluff bodies. Chapter 4 in "Wind Effects on Buildings and Structures", Riera, J.D., Davenport, A.G., Eds, Balkema, Rotterdam, 1998, pp. 61-95

# Appendix A

# Pareto Terminology

This appendix contains the mathematical definitions of the notions introduced in Subsec. 2.1.2 about Pareto optimality, according to [30], pp. 10-11.

***Pareto Optimality***: A solution $\boldsymbol{x} \in \Omega$ is said to be Pareto Optimal with respect to $\Omega$ if and only if there is no $\boldsymbol{x}' \in \Omega$ for which $\boldsymbol{v} = F(\boldsymbol{x}') = (f_1(\boldsymbol{x}'), \ldots, f_k(\boldsymbol{x}'))$ dominates $\boldsymbol{u} = F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$. The phrase Pareto Optimal is taken to mean with respect to the entire decision variable space unless otherwise specified.

***Pareto Dominance***: A vector $\boldsymbol{u} = (u_1, \ldots, u_k)$ is said to dominate another vector $\boldsymbol{v} = (v_1, \ldots, v_k)$ (denoted by $\boldsymbol{u} \preceq \boldsymbol{v}$) if and only if $\boldsymbol{u}$ is partially less than $\boldsymbol{v}$, i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i$.

***Pareto Optimal Set***: For a given Multi-Objective optimization Problem, $F(\boldsymbol{x})$, the Pareto Optimal Set, $P^*$, is defined as:

$$P^* = \{\boldsymbol{x} \in \Omega \mid \neg\exists \boldsymbol{x}' \in \Omega : \ F(\boldsymbol{x}') \preceq F(\boldsymbol{x})\}. \tag{A.1}$$

***Pareto Front***: For a given Multi-Objective optimization Problem, $F(\boldsymbol{x})$, and Pareto Optimal Set, $P^*$, the Pareto Front $PF^*$ is defined as:

$$PF^* = \{\boldsymbol{u} = F(\boldsymbol{x}) \mid \boldsymbol{x} \in P^*\}. \tag{A.2}$$

***Weak Pareto Optimality***: A point $\boldsymbol{x}^* \in \Omega$ ia s weakly Pareto optimal if there is no $\boldsymbol{x} \in \Omega$ such that $f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*)$, for $i = 1, \ldots, k$.

***Strict Pareto Optimality***: A point $\boldsymbol{x}^* \in \Omega$ ia s strictly Pareto optimal if there is no $\boldsymbol{x} \in \Omega$, $\boldsymbol{x} \neq \boldsymbol{x}^*$ such that $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*)$, for $i = 1, \ldots, k$.

# Appendix B

# Effect of kernel type on Gaussian Processes

As written in section 2.3, a variety of kernels could be use in describing a Gaussian Process according to the specific application. One of the more often used is the Powered Exponential Kernel, which takes the general form of:

$$k(x_i, x_j) = \sigma^2 e^{-\frac{1}{2}(\frac{||x_i - x_j||}{l})^\gamma} \tag{B.1}$$

in which $0 < \gamma \leq 2$, $||x_i - x_j||$ is the Euclidean norm, $\sigma^2$ is the variance and $l$ is the characteristic lengthscale (these last two are called hyperparameters). The case of $\gamma = 1$ takes the name of Exponential Kernel, while the case of $\gamma = 2$ is the Squared Exponential or Gaussian Kernel.

Another important class of covariance functions are the Matern functions, given by:

$$k(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)}(\frac{\sqrt{2\nu}||x_i - x_j||}{l})^\nu K_\nu(\frac{\sqrt{2\nu}||x_i - x_j||}{l}) \tag{B.2}$$

where $\nu$ and $l$ are positive hyperparameters, $\Gamma(\nu)$ is the Gamma function and $K_\nu$ is a modified Bessel function. For machine learning purposes, the most interesting cases are those with $\nu$ half-integer, which considerably simplifies the covariance function. In particular, $\nu = 3/2$ and $\nu = 5/2$ have appeared to be the most useful ([17], p. 85):

$$k_{\nu=3/2}(x_i, x_j) = (1 + \frac{\sqrt{3}||x_i - x_j||}{l})e^{-\frac{\sqrt{3}||x_i - x_j||}{l}} \tag{B.3}$$

$$k_{\nu=5/2}(x_i, x_j) = (1 + \frac{\sqrt{5}||x_i - x_j||}{l} + \frac{5||x_i - x_j||^2}{3l^2})e^{-\frac{\sqrt{5}||x_i - x_j||}{l}}. \tag{B.4}$$

The Matern Kernel is obtained by multiplying the previous functions by the variance $\sigma^2$.

Other covariance functions may appear way simpler, like the Brownian Kernel:

$$k(x_i, x_j) = min(x_i, x_j), \tag{B.5}$$

and many others are also available (see [17], pp. 79-102).

However, the type of kernel directly affects the characteristics of the functions composing the Gaussian Process. For example, Fig. B.1 compares four zero-mean Gaussian Processes with fixed hyperparameters (variance and lengthscale) but different kernels: Gaussian (blue), Exponential (red), Brownian (yellow) and Matern with $\nu = 3/2$ (purple) kernel.

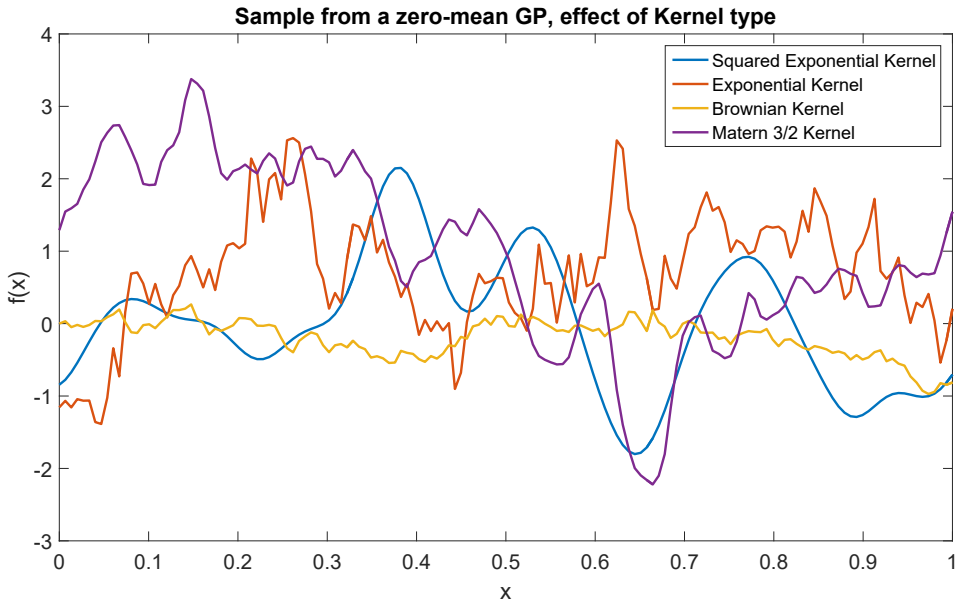



Figure B.1. Effect of different kernel choices (fixed hyperparameters).

As the figure shows, the Gaussian Kernel createa a very smooth curve, which is infinitely differentiable. Thus, this type of kernels is of great interest in many applications. Gaussian kernels are shaped by two hyperparameters: $\sigma^2$ (variance) and $l$ (characteristic lengthscale). The first affects the width of the response space spanned by the Gaussian Process: bigger variance implies wider functions (Fig. B.2).

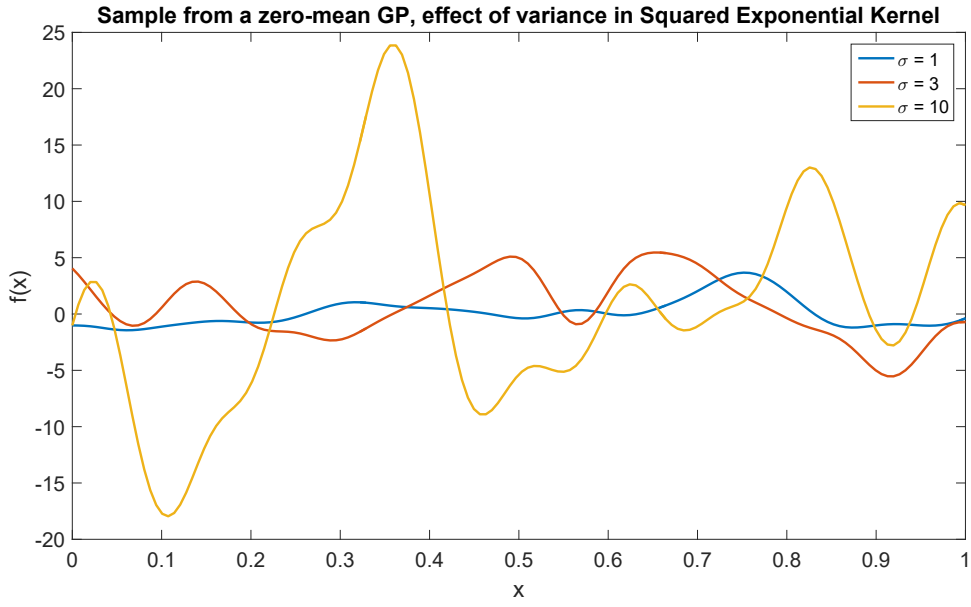



Figure B.2. Effect of different variances (fixed lengthscale).

The second influences the correlation between two points of the response curve: a shorter lengthscale means a weaker correlation between points far from each other and, hence, a stronger zig-zag pattern of the curve (Fig. B.3).


Figure B.3. Effect of different lengthscales (fixed variance).

# Appendix C

# Prediction of noise-free observations in Gaussian Processes

Section 2.3 describes the theory behind Guassian Processes and Kriging models. In the following appendix, the math to make predictions with a Kriging model is shown ([17], pp. 15-16 and 200-201).

In general, the Kriging model is used in order to make predictions, with their corresponding uncertainty, given some known data. These $n$ training points have inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ and give as an output the vector:

$$\boldsymbol{f}(\boldsymbol{x}) = (f(x_1), \ldots, f(x_n)), \tag{C.1}$$

while the $n^*$ predictions of the test points will form the vector:

$$\boldsymbol{f}^*(\boldsymbol{x}^*) = (f(x_1^*), \ldots, f(x_{n^*}^*)), \tag{C.2}$$

given the inputs $\boldsymbol{x}^* = (x_1^*, \ldots, x_{n^*}^*)$.

For sake of simplicity, consider a zero-mean Gaussian Process. Both the training and the test vectors have a Gaussian distribution (recall that this comes from the very definition of Gaussian Process). Thus, their joint distribution is again Gaussian and it is given by:

$$\begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}) \\ \boldsymbol{f}^*(\boldsymbol{x}^*) \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) & \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^*) \\ \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}) & \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix} \right) \tag{C.3}$$

where the matrices $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) \in \mathbb{R}^{n,n}$, $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^*) \in \mathbb{R}^{n,n^*}$, $\boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}) \in \mathbb{R}^{n^*,n}$ and $\boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}^*) \in \mathbb{R}^{n^*,n^*}$ are the covariance matrices evaluated at all pair of training and test points, and whose entries depend on the choice of the kernel. Note that the matrix $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^*)$ is the transpose of $\boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x})$ and that they are not squared matrices, as in general $n \neq n^*$. The distribution of Eq. C.3 represents the prior distribution.

In order to obtain the posterior distribution, it is necessary to condition the joint Gaussian prior distribution on the training data. This corresponds to applying the Bayes Theorem, Eq. 2.11.

When considering two jointly Gaussian random vectors like:

$$\begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{bmatrix}, \begin{bmatrix} \boldsymbol{A} & \boldsymbol{C} \\ \boldsymbol{C^T} & \boldsymbol{B} \end{bmatrix} \right), \tag{C.4}$$

their marginal distributions are given by $\boldsymbol{v_1} \sim \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{A})$ and $\boldsymbol{v_2} \sim \mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{B})$, while the conditional distribution of $\boldsymbol{v_1}$ given $\boldsymbol{v_2}$ is:

$$\boldsymbol{v_1}|\boldsymbol{v_2} \sim \mathcal{N}\left(\boldsymbol{\mu_1} + \boldsymbol{C}\boldsymbol{B^{-1}}(\boldsymbol{v_2} - \boldsymbol{\mu_2}), \boldsymbol{A} - \boldsymbol{C}\boldsymbol{B^{-1}}\boldsymbol{C^T}\right) \tag{C.5}$$

and the conditional distribution of $\boldsymbol{v_2}$ given $\boldsymbol{v_1}$ is:

$$\boldsymbol{v_2}|\boldsymbol{v_1} \sim \mathcal{N}\left(\boldsymbol{\mu_2} + \boldsymbol{C^T}\boldsymbol{A^{-1}}(\boldsymbol{v_1} - \boldsymbol{\mu_1}), \boldsymbol{B} - \boldsymbol{C^T}\boldsymbol{A^{-1}}\boldsymbol{C}\right). \tag{C.6}$$

Hence, the conditional probability of the test points on the training data is given by:

$$\begin{aligned}\boldsymbol{f}^*(\boldsymbol{x}^*)|\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{N}\,(&\boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x})\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x})^{-1}\boldsymbol{f}(\boldsymbol{x}),\\ &\boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{K}(\boldsymbol{x}^*, \boldsymbol{x})\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x})^{-1}\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^*)).\end{aligned} \tag{C.7}$$

As expected, the posterior distribution is still Gaussian. However, the mean and the covariance functions now incorporate the information from the training set. In order to obtain the predictions $\boldsymbol{f}^*(\boldsymbol{x}^*)$, it is simply necessary to sample from the probability distribution of Eq. C.7.