Politecnico di Torino

Master's Degree in Aerospace Engineering

Master Thesis

# Multi-objective Aeroelastic Analysis and Optimization using Surrogate Models

**Candidate:**
Alessandra Lunghitano, 303902

**Tutors:**
Prof. Frederico José Prata Rente Reis Afonso, Instituto de Engenharia Mecânica (IDMEC), Instituto Superior Técnico, Lisbon, PT

Prof. Afzal Suleman, IDMEC, Instituto Superior Técnico, Lisbon, PT

Prof. Marco Petrolo, Department of Mechanical and Aerospace Engineering (DIMEAS), Politecnico di Torino, Torino, IT

Prof. Enrico Zappino, DIMEAS, Politecnico di Torino, Torino, IT

December 2023

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **DIMEAS** | Department of Mechanical and Aerospace Engineering |
| **IDMEC** | Instituto de Engenharia Mecânica |
| **ANNs** | Artificial Neural Networks |
| **CRV** | Cartesian Rotation Vector |
| **CFD** | Computational Fluid Dynamics |
| **DOF** | Degrees of Freedom |
| **DoE** | Design of Experiments |
| **FoR** | Frame of Reference |
| **FSI** | Fluid-structure interaction |
| **GPR** | Gaussian Process Regression |
| **GEBM** | Geometrically-Exact Composite Beam |
| **HARW** | High aspect-ratio wings |
| **HFMs** | High-Fidelity Models |
| **LHS** | Latin Hypercube sampling |
| **LFMs** | Low-Fidelity Models |
| **MAE** | Mean Absolute Error |
| **MSE** | Mean Squared Error |
| **MDO** | Multidisciplinary Design Optimization |
| **MFMs** | Multi-Fidelity Models |
| **ML** | Machine Learning |
| **MLS** | Moving Least Squares |
| **ODE** | Ordinary Differential Equation |
| **OA** | Orthogonal arrays |
| **PDE** | Partial Differential Equation |
| **RBF** | Radial Basis Function |
| **RANS** | Reynolds Averaged Navier-Stokes |

| | |
|---|---|
| **RMSE** | Root Mean Squared Error |
| **ROMs** | Reduced Order Models |
| **RSMs** | Response Surface Models |
| **SMs** | Surrogate Models |
| **SVR** | Support Vector regression |
| **SVM** | Support Vector Machine |
| **SVMr** | Support Vector Machine for regression |
| **UVLM** | Unsteady Vortex Lattice Method |

**Ringraziamenti**

A mio nonno Basilio, di cui oggi sento la mancanza più di ogni altro giorno. Avrei voluto condividere questo traguardo con te, e portarti a cena come ci eravamo detti tante volte. Non sei qui fisicamente, ma sei accanto a me ogni istante.

A mia nonna Nunzia, grazie per l'amore incondizionato e inquantificabile. Ai miei nonni Maria e Salvatore, vi immagino felici insieme, anche se in posti a noi sconosciuti.

Ai miei genitori, Licia e Alessandro, per l'amore e l'incoraggiamento che mi avete riservato in tutti questi anni, per essermi sempre stati vicini e per la fiducia che avete sempre avuto in me e nelle mie capacità. A mio fratello Flavio, per te ci sarò sempre, anche quando non vorrai giocare con me alla play. Ai miei zii, a mia cugina Giulia, pilastro fondamentale della mia vita, a mia cugina Martina, e alle mie cuginette Marika, Morena, Ginevra e Carlotta. Avrete sempre in me una persona sui cui contare.

A Rosaria e Nicola, grazie per avermi fatto sentire a casa anche qui a Torino, grazie per le infinite quantità di cibo, e per l'immenso supporto che mi avete dato in questi anni.

A Giulia. Grazie per avermi sopportata nei tanti momenti di sclero, grazie per tutti i momenti insieme, quelli di leggerezza, ma soprattutto quelli meno leggeri, in cui mi hai insegnato ad affrontare le cose che mi hanno sempre spaventata di più.

A Rebecca, Teresa, Igor, Gaia e Samuele. Grazie di essere quelle persone con cui ho condiviso tutti i momenti più importanti della mia vita fino ad ora, per essermi sempre stati accanto, e per non avermi mai fatto dubitare che sarà sempre così. Grazie per tutti i consigli che non ho mai ascoltato e la forza che siete sempre in grado di trasmettermi.

Ad Eugenio ed Elia, siete stati la mia seconda casa qui a Torino in questi anni. Per tutti i momenti che abbiamo condiviso insieme, per tutte le storie che abbiamo già da raccontare, e per tutte le nuove che arriveranno. Ci immagino sempre tra 30 anni a guardare il Milan perdere.

A Rosamaria e Andrea, forse non ve l'ho mai detto, ma due anni fa avete totalmente cambiato la mia vita. Grazie di rendere ogni momento insieme un momento felice.

E a tutte le altre persone che sono qui adesso, volevo solo dirvi che non potrei desiderare persone migliori con cui condividere questo momento.

Scusate per l'ansia, vi voglio bene.

## Abstract

Aeroelastic analysis plays a crucial role in the design and evaluation of aircraft structures, ensuring their structural integrity and dynamic stability under aerodynamic loads. However, conducting detailed aeroelastic simulations using high-fidelity computational methods can be computationally expensive. This paper proposes an approach to reduce the computational cost of aeroelastic analyses through the use of surrogate modeling techniques based on machine learning. Surrogate models act as efficient approximations of the complex aeroelastic simulations, providing accurate predictions of critical parameters while significantly reducing the computational cost. The Goland wing is utilized as a the baseline configuration, and aeroelastic simulations are performed using the SHARPy framework, an open-source Python tool developed by Imperial College London, to generate the data-set required for surrogate model training. The trained surrogate regression models are capable of predicting important aeroelastic quantities, including flutter speed, mass, and lift and drag coefficients. Moreover, a multi-objective optimization framework is employed to maximize the lift-to-drag ratio and minimize the structural mass while considering a flutter constraint, improving the effectiveness of the preliminary design process.

**Keywords:** Aeroelasticity; Multidisciplinary Design Optimization; Multi-objective Optimization; Wing Design; Surrogate Models

## Abstract in lingua italiana

Nella progettazione e nella valutazione delle strutture aeromobili, l'analisi aeroelastica garantisce l'integrità strutturale e la stabilità sotto i carichi aerodinamici. L'esecuzione di simulazioni aeroelastiche dettagliate utilizzando metodi di calcolo ad alta fedeltà può tuttavia essere computazionalmente costosa. Un metodo che utilizza modelli surrogati per ridurre il costo computazionale delle analisi aeroelastiche viene proposto nel presente lavoro. I modelli surrogati riducono notevolmente i costi computazionali e forniscono previsioni accurate dei parametri essenziali nelle complesse simulazioni aeroelastiche. Per creare il set di dati necessario per l'addestramento dei modelli surrogati, è stato utilizzato il framework SHARPy, uno strumento Python open source sviluppato dall'Imperial College di Londra. L'ala Goland è stata utilizzata come configurazione di base. La velocità di flutter, la massa e i coefficienti di portanza e resistenza sono tutti previsti dai modelli di regressione surrogati. Inoltre, con l'obiettivo di migliorare l'efficacia del processo di progettazione preliminare, viene utilizzata un'ottimizzazione multiobiettivo per massimizzare l'efficienza aerodinamica e ridurre la massa strutturale tenendo conto di un vincolo di flutter.

**Parole chiave:** Aeroelasticità; Ottimizzazione multidisciplinare; Ottimizzazione multiobiettivo; Progettazione di ali; Modelli surrogati

# Introduction

## 1.1 Motivation

Aeroelasticity is the branch of science that examines how inertial, elastic, and aerodynamic forces interact when they operate on a flexible structure exposed to fluid flow. The interaction of these three forces, depicted in Figure 1.1 can result in a number of unwanted events, including vortex shedding, buffeting, galloping, limit cycle oscillations and, in particular, *divergence* (a static aeroelastic phenomena), and *flutter* (a dynamic aeroelastic phenomena) [1]. The study of certain aeroelastic phenomena, such as flutter,

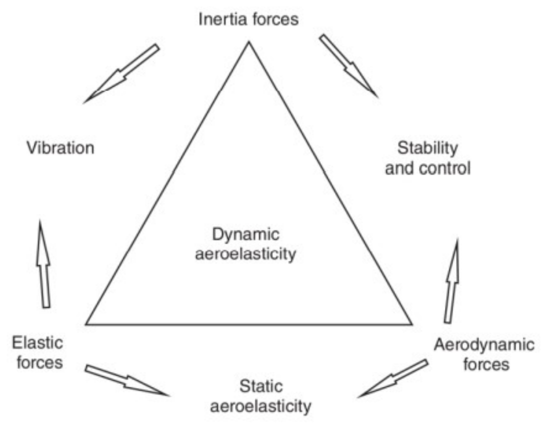


Figure 1.1: Collar aeroelastic triangle [1].

during the preliminary stage of aircraft design can be crucial. Flutter creates divergent oscillations that could cause structural failure, performance and ride comfort degradation, or even loss of control. For these reasons, it definitely represents an undesirable occurrence in airplanes. Costly redesign is necessary if flutter is identified during the aircraft certification process. Adding flutter analysis as a constraint in MDO reduces the danger of expensive changes made late in the design cycle. Moreover, such a study in the early stages of design is becoming critical given the trend to increase wing slenderness to improve aerodynamic performance [11]. The main problem is the high computational cost associated with aeroelastic analysis, especially in the presence of non-linear behaviour (such as large structural deflections and transonic flow conditions) due to the dependence of the flutter point on the equilibrium state. Structural optimization alone, even if including aerostructural analyses for enforcing flutter constraints, yields design solutions with suboptimal performance compared to the optimal designs resulting from MDO, where structural and aerodynamic sizing variables are optimized simultaneously. MDO can minimize structural weight, fuel consumption, or a combination of these two objectives with respect to wing shape, internal structure arrangements, and sizing, while accounting for the interactions between aerodynamics, structures, and other disciplines, and satisfying various constraints [12].

## 1.2 Objectives

In this work, a multi-objective optimization has been performed using SMs to mitigate the computational complexities associated with aeroelastic analyses. The primary goals are to enhance design efficiency and maximize lift-to-drag ratio by integrating flutter constraints within MDO, while also considering the minimization of structural weight as a secondary objective. The research seeks to bridge the gap between computational intensity and design precision, enabling effective aeroelastic optimization. The research navigates a methodical exploration of methodologies and strategies, focusing on SMs for aeroelastic optimization.

## 1.3 Document Outline

This thesis dissertation is organized into six chapters, whose main topics are summarized below. Chapter 1 presents the rationale for the study and a general introduction to the problem. Chapter 2 gives an introduction of the MDO problem and discusses optimization methods in general, with a particular focus on multi-objective optimization techniques. The chapter also provides insights into some of the methods that could be used to successfully include aeroelasticity into the optimization problem. With the goal of minimizing the computational cost in mind, different approaches are being considered, namely Multi-Fidelity Models (MFMs), conventional and machine learning-based SMs, and Reduced Order Models (ROMs). Chapter 3 discusses in more detail conventional and machine learning-based SMs, the methodology chosen to address the problem. It clarifies the theoretical foundations of different SMs and highlights their importance in reducing computing overhead. Chapter 4 revolves around the methodology implemented. Starting with the definition and creation of the dataset, it introduces the selected wing model and the aeroelastic tool used, SHARPy, including its limitations. It also presents the design space and the sampling method used. The chapter then explains the selection and training process of the selected surrogate model, the Extra Trees Regressor. In addition, the validation and testing methodology of the SMs is discussed in depth to ensure the robustness and accuracy of the process. Finally, the heart of the research is addressed: aeroelastic optimization using SMs. The goals, constraints and parameters of optimization are explored. Non-dominated Sorting II (NSGA-II) genetic algorithm, implemented through pygmo, is the chosen optimization technique. Chapter 5 closely examines the results generated by surrogate-based aeroelastic optimization. A rigorous comparative analysis is conducted to examine the optimization results both before and after some adjustments and at different cruising speeds. Finally, Chapter 6, provides a summary of the research path, drawing conclusions from the empirical evidence gathered throughout the study. It conducts a rigorous comparative analysis, comparing the results obtained with conventional aeroelastic simulations. The chapter discusses the advantages, accuracy and efficiency of surrogate-based optimization. The chapter also discusses the broader implications of the results for the aerospace industry and suggests potential future research avenues.

# Multidisciplinary Design Optimization (MDO)

This chapter explores the field of MDO, explaining its basic principles and outlining commonly used optimization techniques. Special attention is given to multi-objective optimization, which is critical to addressing the complex design challenges in modern aerospace engineering.

In this scenario, the importance of considering aeroelastic factors within the optimization framework is emphasized, bringing attention to the problem of high computational cost. With the aim of overcoming this challenge and efficiently integrating aeroelasticity into MDO, a detailed review of some existing strategies is conducted.

## 2.1 Overview of MDO

MDO, a branch of engineering that applies optimization techniques across multiple disciplines, finds its initial applications in aircraft design. This field harnesses the interactions of various disciplines within the design process, enabling the determination of optimal design variables that yield the best objective function value while adhering to defined constraints [13]. The intricate relationship between these disciplines is encapsulated in the MDO scheme depicted in Figure 2.1.

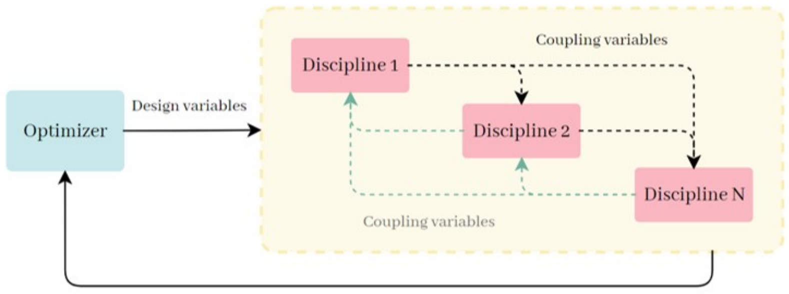


Figure 2.1: MDO scheme.

Modern computational analytic methods can be used to handle MDO issues early in the design process, improving the design and reducing the length and expense of the design cycle. In MDO, it's crucial to specify the architecture, i.e. the organization of the optimization software, the disciplines, and the approximation model analysis. Thus, the distributed and monolithic architectures can be distinguished from one another. In the monolithic approach, only one optimization problem will be tackled, whereas in the distributed approach, several subproblems that have fewer variables and constraints will be solved, dividing the search for the primary problem's optimal solution into smaller tasks [14]. There are numerous approaches to tackle the same optimal design problem, and the choice of architecture affects both the ultimate optimum design and the computing cost. The principles of computational design are based on numerical and analytic techniques that compare the relative merits of a collection of workable solutions. The value of an objective function, which is calculated by numerical simulations, determines the quality of a design. The selection of the objective function is crucial and necessitates thorough understanding of the current multidisciplinary design challenge [15].

## 2.2 Classification of Optimization techniques

Optimization algorithms fall into two primary categories:

- Gradient free: Optimization is based solely on the objective function's value.
- Gradient based: Uses both the objective function value and its gradient with respect to the design parameter.

While gradient free methods offer a broader exploration of the design space, they become computationally expensive with many design variables [14]. Gradient-based methods, conversely, are more efficient for high-dimensional problems but are sensitive to gradient accuracy and may converge to local minima [16].

Given the complexity in areas like aeroelastic optimization, understanding available optimization strategies is essential. Figure 2.2 further divides optimization strategies into classical and meta-heuristic categories.

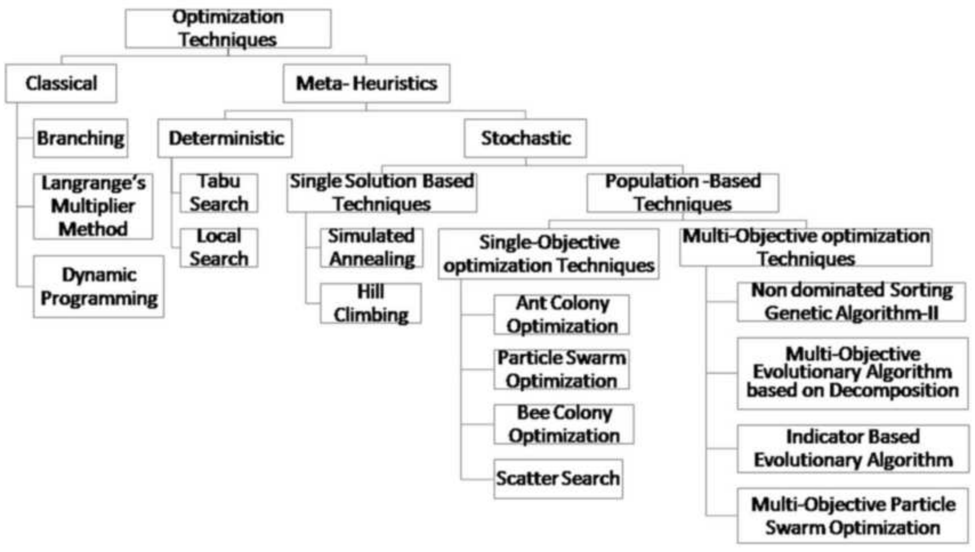

Optimization Techniques
Classical
Meta- Heuristics
Branching
Langrange's Multiplier Method
Dynamic Programming
Deterministic
Stochastic
Tabu Search
Local Search
Single Solution Based Techniques
Population -Based Techniques
Simulated Annealing
Hill Climbing
Single-Objective optimization Techniques
Multi-Objective optimization Techniques
Ant Colony Optimization
Particle Swarm Optimization
Bee Colony Optimization
Scatter Search
Non dominated Sorting Genetic Algorithm-II
Multi-Objective Evolutionary Algorithm based on Decomposition
Indicator Based Evolutionary Algorithm
Multi-Objective Particle Swarm Optimization

Figure 2.2: Categorization of optimization techniques [2].

Classical techniques guarantee optimal solutions but are less used in practical scenarios. Meta-heuristic methods, on the other hand, are versatile and effective for real-world problems, without ensuring perfect solutions. These are split into deterministic and probabilistic techniques [2]. Deterministic methods yield consistent results for identical starting conditions [17], whereas probabilistic ones utilize randomness, offering varied solutions even from identical starting points. These can be further sub-divided into single-solution and population-based strategies, with the latter enhancing search diversity [18]. Moreover, techniques can focus on single or multi-objective problems. Single-objective techniques target one objective function, whereas multi-objective ones address multiple conflicting objectives simultaneously, gauging solution quality by dominance [19].

Numerous optimization algorithms have been discussed in the literature, but none can be considered a universal solution for efficiently solving various problems. The choice of algorithm for a particular problem depends on several factors, such as the number and type of constraints, the types of variables (continuous or discrete), the type of objective function, and the complexity of the problem [2].

In terms of aeroelastic optimization, our choice has fallen on a multi-objective optimization. In the problem we are actively maximizing aerodynamic performance and minimizing mass, subject to adhering to constraints related to aeroelastic stability. In this way, the design seeks to ensure robust aerodynamic efficiency and aircraft wing weight limitation while satisfying the required stability considerations.

## 2.3 Multi-objective optimization

The application of multi-objective optimization allows for a more holistic assessment of the attainable solutions, providing a set of optimal solutions, known as the Pareto frontier, instead of a single optimal solution. This is essential to ensure that the final design is well balanced in terms of different performance goals. [20].

### 2.3.1 Preliminary Concepts

In this section, the basic definitions of multi-objective algorithms are outlined.

Multi-objective techniques are used to optimize two or more competing goals at the same time, while considering certain constraints [21]. Multi-objective problems are defined by an objective function vector $F(x)$ which is minimized or maximized based on a decision variables vector $X$. For every solution $x$ in the decision variable space, there is a corresponding point in the objective function space. The function $f : X \to Y$ evaluates the efficiency of a given solution by assigning the objective function vector $(y_1, y_2, \ldots, y_k)$ in objective space $Y$. Additionally, some multi-objective problems have inequality $g_i(x)$ and equality $h_j(x)$ constraints [2].

#### Objective Function

Objective functions are computable functions used on decision variables to evaluate the quality of a given solution. They can be expressed as:

$$F(x) = (f_1(x), f_2(x), \ldots, f_k(x)) \tag{2.1}$$

Here, $k$ represents the number of objective functions in the problem. In multi-objective problems, the value of $k$ is greater than 1 [2].

#### Decision Variables

In optimization problems, decision variables are independent numerical variables whose values need to be selected [2]. The decision variable vector containing decision variables can be denoted as: $x = [x_1, x_2, \ldots, x_j]^T$, where $j = 1, 2, \ldots, n$.

#### Dominance Relation

A dominance relationship is used to compare multi-objective solutions [2]. A solution $S_1$ is said to dominate solution $S_2$ if:

1. For all objectives, $S_2$ is not clearly superior to $S_1$.
2. In at least one of the objectives, $S_1$ definitely outperforms $S_2$.

#### Pareto Optimality

A point $s^*$ in the feasible search space $S$ is Pareto optimal if no other point $s$ in $S$ improves at least one objective function without worsening another. Mathematically, point $s^* \in S$ is Pareto optimal only if there is no point $s \in S$ such that $f(s) \leq f(s^*)$ with at least one $f_i(s) > f_i(s^*)$. All Pareto optimal solutions, also called non-dominating solutions, in decision variable space, form a Pareto optimal set [21]. There can be an infinite number of non-dominant solutions, all of which are mathematically incomparable. The optimal set depends on the type of objective and is always on the boundary of a feasible region. Several approaches to a multi-objective optimization problem work by approximating the Pareto optimal set [2]. An example is shown in Figure 2.3. The ideal point is achieved by individually minimizing each objective function, disregarding the others. It is uncommon for all the minimizations to converge at a single point in the design space, indicating that it is not possible for one design point to minimize all objective functions simultaneously. Therefore, the ideal point is conceptual and resides solely in the criterion space, typically remaining unattainable in practice [3].

### 2.3.2 Categorization of Multi-objective Optimization Techniques

Multi-objective optimization techniques are broadly classified into three categories: evolutionary, swarm-based, and hybrid techniques [2].

- **Evolutionary Techniques:** Derived from the principles of natural evolution, these techniques are capable of generating a set of trade-off solutions in a single execution while requiring lesser computational resources. Prominent techniques in this category include Pareto Archived Evolution Strategy (PAES), Non-Dominated Sorting Genetic Algorithm-II (NSGA-II), among others. Despite their advantages, they are often criticized for high computational cost and poor constraint management skills.
- **Swarm-based Techniques:** These techniques mimic the collective intelligence found in decentralized biological systems, and are adept at minimizing objective

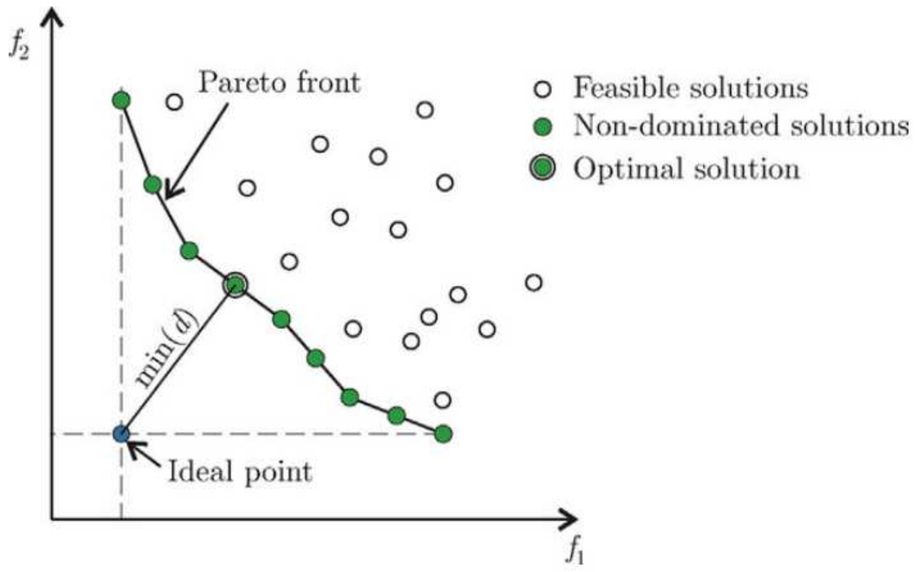


Figure 2.3: Pareto front example [3].

functions without being trapped in local minima. Notable examples include Multi-objective Particle Swarm Optimization (MOPSO) and Multi-objective Ant Colony Optimization (MOACO).

- **Hybrid Algorithms:** These merge the strengths of evolutionary and swarm-based strategies, enhancing solution diversity and convergence. They mitigate the drawbacks of singular approaches. Examples include multi-objective artificial bee colony and differential evolution (HABC-DE).

These techniques encompass a broad spectrum of tools, each with unique merits, for addressing multi-objective optimization challenges. Among them, the the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [21] stands out for certain advantages, which motivated its selection for the study at hand.

The choice was as our optimization tool was first of all influenced by its proven efficacy in handling multi-objective problems. NSGA-II is an evolutionary technique, derived from the principles of natural evolution. This evolutionary algorithm is particularly adept at identifying a Pareto front, thereby providing a set of non-dominated solutions from which designers can make informed decisions. Moreover, being gradient-free, it ensures a robust exploration of the design space without being overly sensitive to local optima or necessitating gradient computations, which can be challenging for intricate models.

The detailed features of NSGA-II and how it was used to address the optimization challenges in this study will be discussed in full in the in the following methodology chapter (Chapter 4).

Despite the potential of the discussed multi-objective optimization techniques, integrating aeroelasticity into the MDO process presents an additional layer of complexity due to the high computational cost associated with analyzing the aeroelastic behaviors of new aircraft designs. This computational burden necessitates the exploration of strategies to efficiently integrate aeroelasticity into MDO.

## 2.4 Current strategies for efficiently integrating aeroelasticity into MDO

Although the inclusion of aeroelasticity in the optimization process of a new aircraft requires a considerable amount of computational work, the design decision for thin, flexible wings to reduce the aerodynamic induced drag makes them crucial from the very beginning of the conceptual design phase. Therefore, effective computing solutions are explored in order to decrease this computational effort. The type of structural and aerodynamic non-linearities to be captured, the existence of local or global non-linearities, the computational time needed, the accuracy needed, the project stage (preliminary design, detailed design, certification), the nature of the study and the development cost are some of the factors influencing the choice of the models [11].

Different current approaches that could efficiently include aeroelasticity into the optimization problem are presented in this Chapter, including multi-fidelity models (section 2.4.1), surrogate models (section 2.4.2) and reduced-order models (section 2.4.3). An overview of each model and their main features are given.

### 2.4.1 Multi-fidelity Models

In the preliminary or exploratory design stage, low to medium fidelity analytic tools are utilized to do a large number of computations quickly and gather data for high level judgments that are crucial for the long-term cost minimization of the project. In order to accurately capture the physical effects that are relevant and that were previously identified as not being modelled by the low fidelity tools, or as a consequence of design experience, the fidelity of the tools being utilized must be increased [22].

HFMs typically accurately represent system behaviour for the intended purpose. These models are typically expensive, and it is frequently impossible to afford their various realizations. LFMs are less accurate but cheaper. For instance, dimensionality reduction, linearization, less complex physics models, coarser domains, partially converged results, among others as illustred in Figure 2.4 are processes used to obtain them [4].

HFMs and LFMs are combined in MFMs to provide accuracy at a fair price. MFMs offer the conceptual framework to efficiently optimize vehicles by judiciously using a limited number of high-fidelity analyses while leveraging the information provided by low-fidelity methods [23].

In various flutter constrained MDO issues of wings, LFMs [24], HFMs [25,26], and MFMs [27, 28] have all been tested. These models range from LFMs that take into account compressibility and viscous effects, including potential flow theory and panel techniques

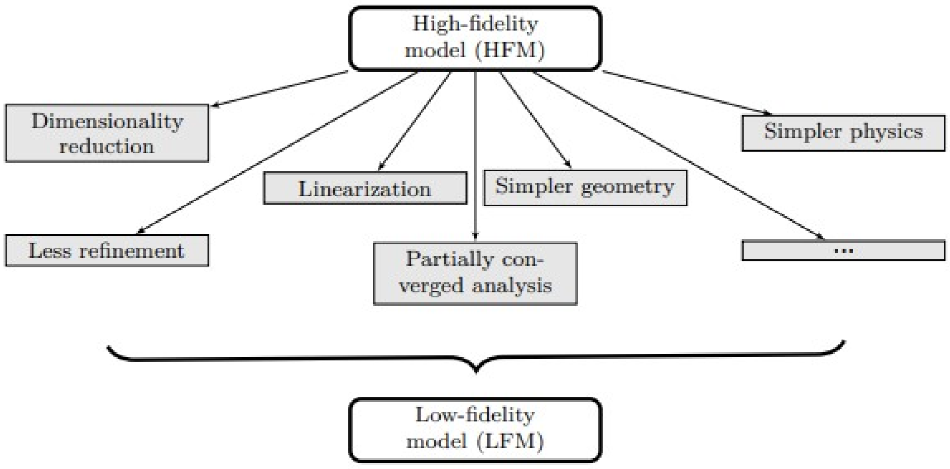


Figure 2.4: From HFMs to LFMs [4].

with corrections [24, 27, 28], to Euler [27] and Reynolds Averaged Navier-Stokes (RANS) solvers [25, 26, 28]. Although fuel burn [28] and structural mass [26] minimization have also been selected, maximizing range [24, 25, 27] was the primary goal of the majority of these optimization challenges. The characteristics of MFMs make them very usable for the study of the aeroelastic problem but they nevertheless present certain limitations that complicate their use for possible optimization. In fact, knowledge of the physical problem of interest is necessary and their reuse between different projects is infrequent. In addition, an aeroelastic optimization would be quite costly due to the need to repeat a large number of iterations. For these reasons MFMs typically entail the creation of SMs [4].

### 2.4.2 Conventional and Machine Learning-based Surrogate Models

In the realm of optimization and analysis, SMs emerge as pivotal tools, bridging the gap between computationally intensive simulations and the need for efficient design exploration. These models provide rapid approximations of complex system behaviors, enabling sensitivity studies, optimization, and decision-making processes. Surrogate models are derived from data collected through simulations or experiments and serve as expedient alternatives to direct evaluations of high-fidelity models. Their applications are particularly significant in MDO due to the intricate interactions among various disciplines involved in the design process [7]. Figure 2.5 illustrates the key steps in surrogate modeling.

The process commences with crafting a sampling plan within the design variable space, referred to as Design of Experiments (DoE). This stage leverages allocation algorithms to distribute samples in the design domain. The preferred technique is influenced by the available number of samples and the intended surrogate modeling approach. Subsequently, a dataset is assembled by executing the computationally expensive model for every input delineated by the DoE. The surrogate model is then chosen and implemented.

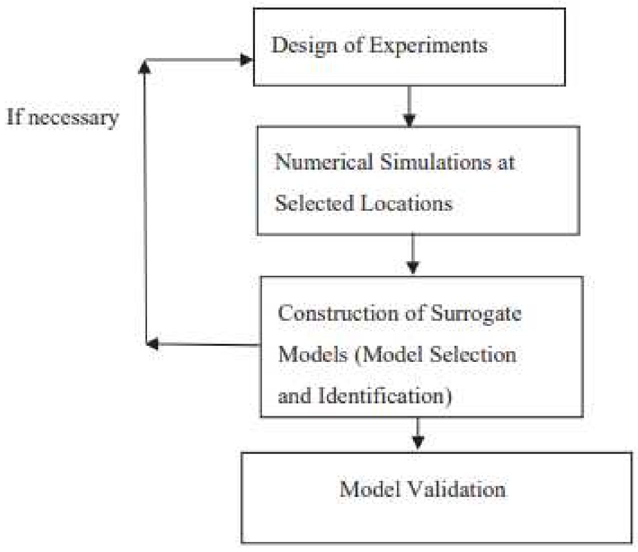

Design of Experiments
If necessary
Numerical Simulations at Selected Locations
Construction of Surrogate Models (Model Selection and Identification)
Model Validation

Figure 2.5: Key stages of the surrogate-based modeling approach [5].

Finally, it is critical to verify the model's accuracy and predictability beyond the available data [7].

In selecting a surrogate modeling strategy, two primary methodologies stand out: the conventional and the machine learning-based techniques.

Conventional methodologies employ Response Surface Models (RSMs), comprising mathematical functions such as linear regression, polynomial regression, and radial basis functions. Though these models are relatively straightforward and cost-effective, they can sometimes falter in capturing intricate nonlinear relationships within the data. Other prevalent methods within this category include Kriging, co-Kriging, Moving Least Squares (MLS), and Support Vector regression (SVR) which leverage spatial correlation to provide accurate approximations [29].

On the other hand, machine learning-based methodologies leverages advanced algorithms to plumb the intricate relationships in the data. Machine Learning (ML) [30] has emerged as one of the most promising technologies in the past decade due to its capability to provide valuable insights into vast amounts of data generated during the Internet era. A distinctive advantage of machine learning-based SMs is the ability to continually learn from a real-time data stream without the necessity of retraining with a new data set.

Models such as Random Forests, Extra Trees Regressor, Gradient Boosting Machines, Artificial Neural Networks (ANNs), Gaussian Process Regression (GPR), and Deep Learning excel in capturing nonlinear and multifaceted behaviors, making them invaluable for complex engineering challenges. ML models can be trained in various ways. Here, the key categories include [31]:

- **Supervised learning:** All observations are labeled, and algorithms are trained using datasets containing inputs.
- **Unsupervised learning:** Observations are unlabeled, and algorithms discern inherent structure from the input data.

- **Semi-supervised learning:** Contains both labeled and unlabeled observations. Typically, a fusion of supervised and unsupervised methods is utilized.
- **Reinforcement learning:** This ML approach targets sequential decision-making problems, aiming to ascertain an optimal policy to guide an agent in a particular environment, often formulated as a Markov decision process.

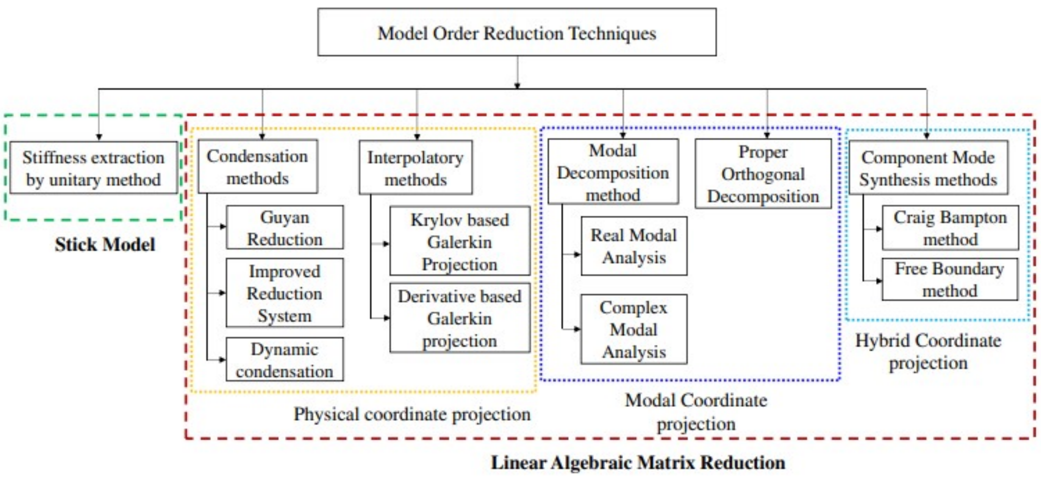
In the open literature, applications based on SMs [32, 33] primarily aim to replace Computational Fluid Dynamics (CFD) analyses and have generated good results, with varying degrees of process acceleration. The same finding applies to those based on machine learning [31], which have broad applications in the optimization of the geometric design space through geometric filtering or modal parametrization. Cea and Palacios [34] state that although SMs have found certain uses in the aeroelastic area, they are rarely used to solve MDO issues that take into account flutter. But research in this field is only just getting started. For example, in order to build strut-braced high aspect ratio wings while taking flutter and stress contraints into consideration, Sohst et al. [35] combined MFMs and SMs. They discovered that even though the flutter limitation had been taken into account during the optimization process, the optimized wing aircraft design had uncalculated instability. Toffol and Ricci [36] have created a way to optimize the structural layout of a conventional airplane so that its mass is kept to a minimum while taking stress and flutter constraints, combining internal codes with SMs.

### 2.4.3 Reduced Order Models

Since they do not take into account the physics of the problem, SMs are severely limited by the fact that they only use one set of parametrized curves to fit the problem. These lend themselves very well to eventual optimization of the aeroelastic problem, but less so to the actual analysis of the problem. ROMs derive from complex physical problems but simplify them in order to shorten the simulation time of the models. To this end, a given problem is often solved several times using a given set of parameters, and a simpler model is then deduced from these results. The central idea of model order reduction is to search for a solution in a specific basis that contains only a few, but sufficient elements to describe the problem [37].

A Partial Differential Equation (PDE) or Ordinary Differential Equation (ODE) described by the higher-fidelity model is transformed into a reduced order model that is essentially another, smaller set of equations with fewer unknowns. This ODE is supposed to be much faster to compute but at the same time captures the same physics as the higher-fidelity model, generating results not much different from the higher-fidelity model [38]. Thomas et al. [6] review various techniques of model order reduction available in the literature including stiffness extraction by unitary loadings, which is commonly used in the aerospace industry, and linear algebraic matrix-based reduction methodologies. These are summarized in Figure 2.6.

The discourse is fairly similar to that of the SMs with regard to the aeroelastic application of these models. They have been used for aeroelasticity problems for a while [39], but their use for MDO problems that take flutter into account is rare in the open literature,



Model Order Reduction Techniques
Stiffness extraction by unitary method
Stick Model
Condensation methods
Guyan Reduction
Improved Reduction System
Dynamic condensation
Interpolatory methods
Krylov based Galerkin Projection
Derivative based Galerkin projection
Physical coordinate projection
Modal Decomposition method
Real Modal Analysis
Complex Modal Analysis
Proper Orthogonal Decomposition
Modal Coordinate projection
Component Mode Synthesis methods
Craig Bampton method
Free Boundary method
Hybrid Coordinate projection
Linear Algebraic Matrix Reduction

Figure 2.6: Model order reduction techniques [6] .

especially for MDO problems that take structural weight and aerodynamic performance into account. Regarding ROMs, certain applications [6, 40, 41] have demonstrated how they can achieve exceptional results in terms of computational cost by greatly lowering the number of Degrees of Freedom (DOF), maintaining the most important variables, and simplifying the physics of the aeroelastic problem. All the studies reported had as their goal to replace CFD analysis. This, in fact, is one of the biggest obstacles when computational savings are desired, as also argued by Li et al. [31].

To create and calculate ROMs there are no readily accessible software packages due to the complexity of the models utilized in the aerospace sector. Finding precise reduced order model error measurements without solving the higher-fidelity model, obtaining reliable ROMs for non-linear parametric problems, and integrating a reduced order model into an automatic optimizer are the main issues in this field that need to be resolved. This is because the data or designs initially used to train a ROM may differ greatly from the actual optimal design [38].

In light of the discussed strategies for efficiently integrating aeroelasticity into MDO, SMs have emerged as a particularly viable solution for our problem due to their ability to provide rapid approximations of complex system behaviors while reducing computational costs. Their capacity to bridge the gap between high-fidelity models and efficient design exploration makes them a compelling choice for further investigation. The following chapter (Chapter 3) delves deeper into surrogate modeling within the realm of MDO, exploring its methodologies and potential benefits in addressing the computational challenges posed by the aeroelasticity integration into the optimization process.

# Surrogate modeling in MDO

The successful full-scale development of modern aerospace systems faces significant challenges in balancing competing goals including higher performance, lower costs, and increased safety. High-fidelity, accurate models are often time- and cost-intensive to create. The so-called surrogate-based strategy for analysis and optimization can be very helpful in this situation. Sensitivity and optimization studies are made possible by the surrogates, which are built using data from high-fidelity models and provide quick approximations of the objectives and constraints at new design points [5].

This chapter focuses on SMs, which aim to integrate aeroelasticity into MDO. Although SMs simply mimic the underlying physics of the systems they represent, they are flexible and can be improved with more data, making them an essential tool. They can integrate seamlessly with MFMs to present a layered approach to systems optimization, demonstrating their utility beyond stand-alone applications. However, the dependence of SMs on physics-based simulations for the development of their data sets must be considered. The demand for these simulations increases as the design space expands, posing new problems. For these reasons this chapter delves deeper into the intricacies of SMs, elucidating on the stages outlined in Figure 2.5.

## 3.1 Design of experiments

It is essential that the design variables have been chosen before conducting the DoE. These stand for the variables that can be manipulated throughout design, in order to see how they affect the results. The selection is not arbitrary and it is based on [42]:

- Relevance: The variable should have a potential significant impact on the system's performance.
- Controllability: The variable should be able to be changed and reliably controlled across studies. It is important to ensure that any adjustments can be practically implemented without excessive complexities or costs.
- Interactivity: Understand and account for the interactions between design variables. A combination of variables might lead to infeasible or undesirable designs even if each variable, in isolation, falls within its feasible range.

DoE provides a structured approach to allocate sample points in a design space for maximum information extraction. Traditional DoE techniques, like factorial designs, are effective for discrete variables, spanning vast areas of the search space. While full factorial designs evaluate every combination of variables, with an increase in variables, fractional factorial designs become practical due to the exponential growth in needed samples [43]. Discretized continuous variables can be integrated into factorial designs. Figure 3.1 showcases various factorial designs for three variables.

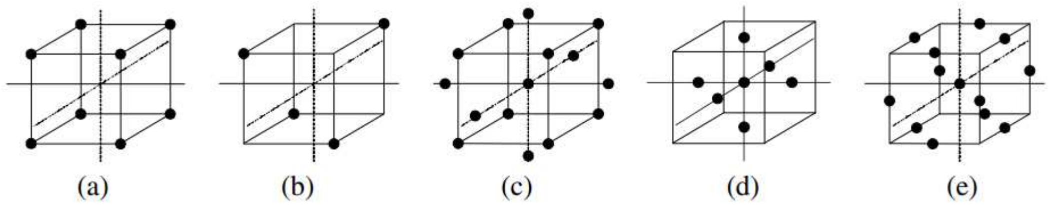


Figure 3.1: Different factorial designs for three design variables [7].

When the target function is unknown, newer DoE strategies like Orthogonal arrays (OA) and Latin Hypercube sampling (LHS) aim to distribute samples uniformly across the design space. OA achieves uniform designs, potentially with replicated points [44], whereas LHS prevents duplicates but might lack uniformity [45]. The latter is a form of stratified sampling. In stratified sampling, the range of a variable is divided into non-overlapping intervals, and one value from each interval is randomly selected. In Figure 3.2 are shown three LHS designs with significant differences in terms of uniformity, where (b) and (c) are more evenly spread throughout the design space.

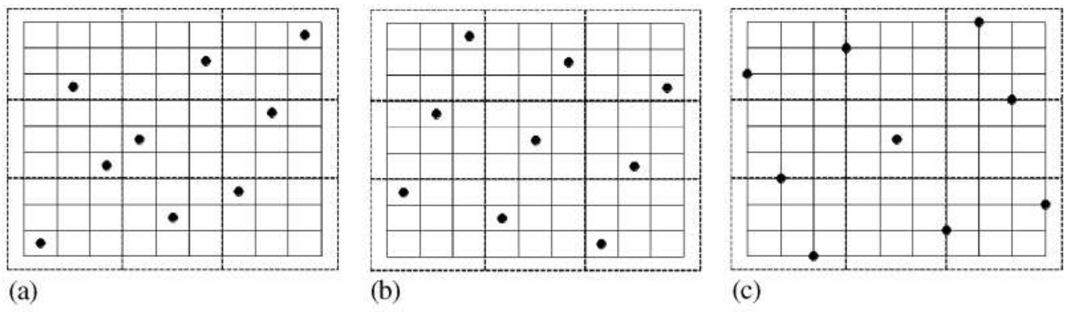


Figure 3.2: Variations of LHS designs [7].

Other common DoE methods encompass quasi-Monte Carlo and Hammersley sampling [43].

## 3.2 Surrogate modeling methods

The selection of the model and the determination of the pertinent parameters comes next, after executing the expensive computational model for each value of the input variable contained in the DoE [5].

### 3.2.1 Linear models

Linear models [46] are foundational regression techniques where predictions are made based on a linear correlation with the features. This relationship can be represented by the Equation (3.1):

$$\hat{y}(\mathbf{w}, \mathbf{x}) = b + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{3.1}$$

where $\hat{y}$ stands for the predicted value, $x_i$ are the dataset's features, $w_i$ denote the coefficients of the regression for $i = 1, 2, \ldots, n$, and $b$ represents the bias term. All variants of linear models adhere to this general structure.

- **Least Squares**: this technique [47] minimizes squared errors between model predictions and actual labels. Its challenge lies in the potential for collinearity among features.
- **Least Absolute Shrinkage and Selection Operator (LASSO)**: LASSO [48] adds regularization to linear models. It effectively performs feature selection, aiding in model simplification.
- **Bayesian ridge**: Bayesian regression [49], instead of assuming a single optimal set of coefficients for the linear relationship, offers a posteriori distribution for the model parameters. This approach allows the incorporation of a priori knowledge about the coefficients, leading to potentially better estimations. The coefficients in this model is represented by a spherical Gaussian.
- **Automatic Relevance Determination (ARD)**: ARD Regression [49] can be considered an extension of Bayesian ridge regression, with key modifications in the priors used. ARD assumes an axis-parallel, elliptical Gaussian distribution. It typically yields sparser coefficients compared to Bayesian ridge.

### 3.2.2 Decision trees

Decision Trees [50] are supervised learning methods capable of tackling both classification and regression tasks. Conceptually, they segment the dataset through a series of conditional statements based on feature values. Additional layers of such conditions can further refine classifications or provide numerical estimations in regression scenarios. A critical parameter for decision trees is their depth, representing the number of successive conditions. Striking the right balance in depth is essential; shallow trees might underfit, whereas overly deep ones risk overfitting. To combat overfitting, model tuning, often focusing on optimal tree depth, is employed.

As shown in Figure 3.3, a decision tree is made up of a root node, child nodes, and leaf nodes. The algorithm starts from the root node and basically selects a split rule and a cut point based on its properties. This process is repeated in each child node until reaching a leaf node.

One approach to amplify decision tree performance and reduce overfitting is through ensemble learning. In this context, Random Forests [51] shine by merging outputs from multiple decision trees. Each tree in the forest is trained on a feature subset from the main dataset, introducing diversity among trees. Post-training, tree outputs are either averaged (regression tasks) or subjected to majority voting (classification tasks).

For even more diversity, Extremely Randomized Trees or Extra Trees [52] can be trained. Unlike random forests where splits are determined by the most discriminating features, extra trees select splits from a pool of random thresholds. This generally reduces variance, albeit with a slight increase in bias.

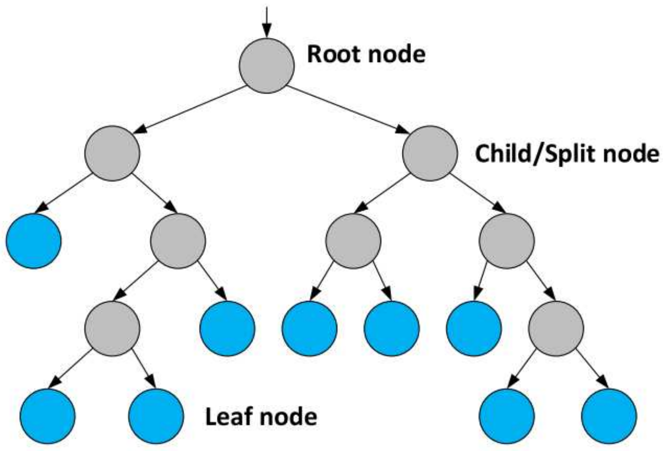


Figure 3.3: Illustration of a decision tree [8].

### 3.2.3 Support Vector Machine for Regression (SVMr)

Support Vector Machine (SVM) is a well-known tool in machine learning introduced by Vladimir Vapnik and his peers in 1992 [53]. In a potentially high- or infinite-dimensional space, SVM creates a hyperplane or set of hyperplanes that can be used for classification, regression, or other tasks. Inferentially, the hyperplane with the greatest distance from the nearest training data points of any class (referred to as the functional margin) achieves a decent separation. Generally speaking, the larger the margin, the smaller the generalization error of the classifier.

Support Vector Machine for regression (SVMr) [54] are effective modeling tools for many regression problems in engineering and are also powerful tools in the field of ML. Kernel theory can be utilized to treat the SVMr as a convex optimization problem, enabling it to handle nonlinear scenarios. The SVMr considers both the model's generalization ability and prediction inaccuracy. Given training vectors $x_1, \ldots, x_n$ in $\mathbb{R}^p$ and a vector $y$ in $\mathbb{R}^n$ of target values, SVMr solves the following primal problem:

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{3.2}$$

subject to:

$$\begin{aligned} y_i - \mathbf{w}^T\phi(x_i) - b &\leq \epsilon + \xi_i \\ -y_i + \mathbf{w}^T\phi(x_i) + b &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, \qquad\qquad i = 1, \ldots, n \end{aligned}$$

Here, $\xi_i$ and $\xi_i^*$ are slack variables allowing for errors greater than $\epsilon$, where $\epsilon$ is the tube width (i.e., the maximum error allowed without any penalty). $C$ is a regularization parameter determining the trade-off between achieving a wide margin and penalizing errors, while $\phi$ is a function mapping the input data to a higher dimensional space (this mapping is performed implicitly by the kernel function).

The dual problem is represented as:

$$\min_{\alpha,\alpha^*} \frac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + \epsilon e^T(\alpha + \alpha^*) - y^T(\alpha - \alpha^*) \tag{3.3}$$

subject to:

$$\begin{aligned} &e^T(\alpha - \alpha^*) = 0 \\ &0 \leq \alpha_i, \alpha_i^* \leq C, \qquad\qquad i = 1, \ldots, n \end{aligned}$$

Here, $Q_{ij} = K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is the kernel function, and $e$ is a vector with all its elements set to one.

The solution is given by:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + b \tag{3.4}$$

Kernel functions [55] play a pivotal role in SVMr, allowing the model to operate in high-dimensional spaces without explicitly computing the coordinates in that space. This mechanism is often called the "kernel trick." The most commonly used kernel functions include linear, polynomial and Radial Basis Function (RBF) (or Gaussian). It is imperative to choose an appropriate kernel function and its hyperparameters based on the nature of the data and through empirical validation methods like cross-validation.

It is essential to emphasize that the spectrum of surrogate modeling is broad and boasts numerous approaches, both traditional and machine learning-based. The choice of a particular technique must be tailored to the requirements of the problem at hand. The methods illustrated here, which span linear, nonlinear, probabilistic and ensemble strategies, have been chosen to provide a comprehensive yet concise overview of this heterogeneous field.

Although they provide a glimpse of the wide range of methods available, their inclusion was strategic: each has gained significant prominence due to its proven effectiveness, recognition, and prevalent application in the field. In addition, each of these methods is supported by robust Python tools, which enable their practical implementation and facilitate direct comparative analysis.

## 3.3 Model selection and validation

The choice of a surrogate model is pivotal. Each model comes with its strengths, weaknesses, and assumptions, making it more or less suited to specific types of aeroelasticity problems. Moreover, the chosen model's efficacy directly impacts the optimization performance, emphasizing the importance of a thorough model selection process.

In the realm of SMs, partitioning the dataset into distinct subsets is a conventional practice. The rationale behind this division is to facilitate distinct stages of the model development process, from training to validation and evaluation. The standard partitioning scheme comprises [56]:

- **Training set:** This is the primary dataset used for the training phase of the model. It is usually the most substantial portion, typically encompassing 60 to 70% of the total data points.
- **Test set:** After training, the model's performance and parameters need validation. The test set serves this purpose, enabling us to choose the best model variant based on unseen data. It typically contains about 15 to 20% of the total samples.
- **Cross-validation set:** This dataset provides an estimate of the model's generalization error. Commonly accounting for 15 to 20% of the total samples, it aids in ensuring that the model isn't overfitting or underfitting.

Once the model is trained and fine-tuned, its performance is quantitatively assessed using specific metrics [57]. Evaluating models using these metrics enables a systematic comparison, guiding the choice for the best-suited model for a given problem. The most frequently used ones are shown below.

- **Mean Absolute Error (MAE):**

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.5}$$

  This metric quantifies the average absolute disparity between observed and predicted outcomes, essentially capturing the mean residual magnitude.

- **Mean Squared Error (MSE):**

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.6}$$

  The MSE computes the mean of squared deviations between observed and model-predicted values, highlighting the residual variance.

- **Root Mean Squared Error (RMSE):**

$$RMSE = \sqrt{MSE} \tag{3.7}$$

  This metric provides insight into the standard deviation of the residuals.

- **Coefficient of Determination:**

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{3.8}$$

  where $\bar{y}$ is the mean of the observed data. This metric denotes the fraction of response variable variance explained by predictors in a regression model. Importantly, it remains scale-invariant, never exceeding one.

MSE and RMSE both penalize prediction errors more severely than MAE. However, RMSE stands out because its values are in the same unit as the dependent variable, making it easier to interpret. Another advantage lies with the differentiability of MSE, which often results in RMSE being the preferred choice for loss functions in many modeling scenarios. In evaluating model performance, lower MAE, MSE, and RMSE values are desired, while a higher R-Squared value suggests a model that fits the data better. It's important to note, though, that R-Squared can be artificially inflated with the addition of more predictors, potentially introducing noise rather than clarity. When it comes to gauging regression models, RMSE often provides deeper insights compared to R-Squared. Together, these metrics streamline the process of choosing the most suitable model for the task at hand [58].

Another pivotal aspect in the model development and selection process is *hyperparameter tuning* [59]. Once a model is chosen based on its structure and assumptions, it is essential to fine-tune its hyperparameters to optimize its performance further. Hyperparameters are parameters that are not learned from the data but are set before the learning process begins. Their setting can significantly influence the model's performance, either enhancing or degrading it. Proper hyperparameter settings can lead to faster convergence during training and improved model generalization on unseen data. Conversely, poorly chosen hyperparameters can render even the best model structures ineffective. Hyperparameter tuning is typically conducted using methods like grid search, random search, or more advanced methods such as Bayesian optimization. *Grid search* is a methodical approach where every possible combination of hyperparameters in a predefined list is evaluated. While it is exhaustive and ensures that the optimal combination within the search space is found, its efficiency diminishes as the dimensionality of the search space grows. In contrast, *random search* involves sampling hyperparameter configurations at random from the same space. Although less exhaustive, random search can be more efficient as it might find good configurations with fewer evaluations.

The goal of these methods is to find the optimal hyperparameters that yield the best performance on the validation set, which can then be confirmed using the test set.

Once the process of adjusting hyperparameters is completed and the model is perceived to be optimal, an essential next step emerges: testing its effectiveness on an independent test set. This step is more than just validation; it is a test that measures the model's resilience and adaptability to data it has not encountered before [60]. Although a model may achieve commendable metrics, such as a low RMSE or high R-square on the training dataset, these successes could be misleading. There could be a situation where the model, having stored the training data too closely (overfitting), fails to generalize effectively to new, unseen data.

On the other hand, a model might be too general, failing to capture the nuances and complexities of the training data. This phenomenon, known as underfitting, can be detected when the model has lackluster performance metrics on both the training and validation datasets. In this case the general shape may be there, but the complexities are lost.

The objective is achieving a balance, ensuring the model identifies inherent trends while making reliable predictions. Tools like *learning curves* (Figure 3.4) prove essential in identifying and remedying these issues. The plot was generated using `scikit-learn`, an open-source Python package.

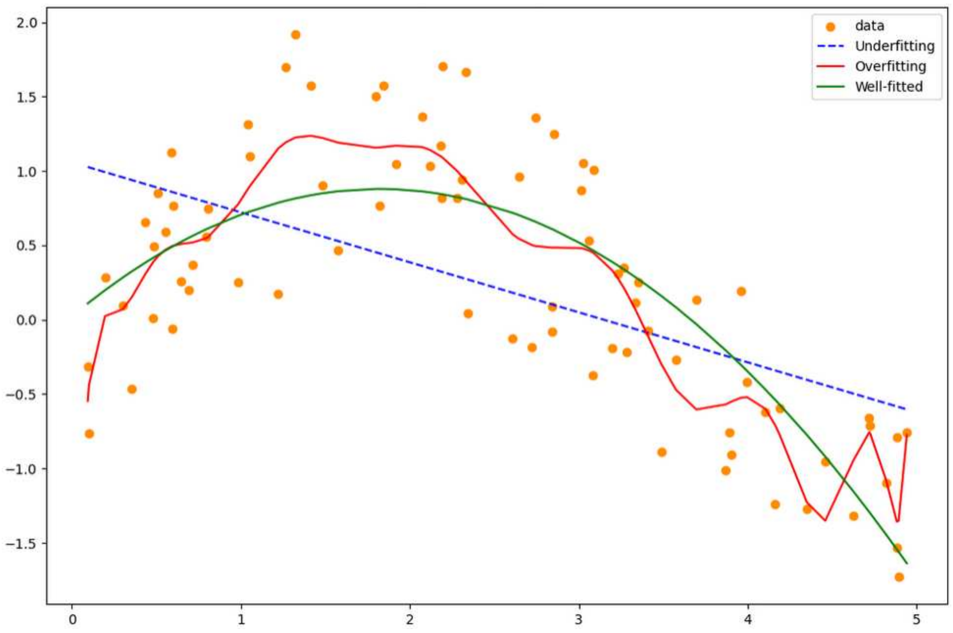
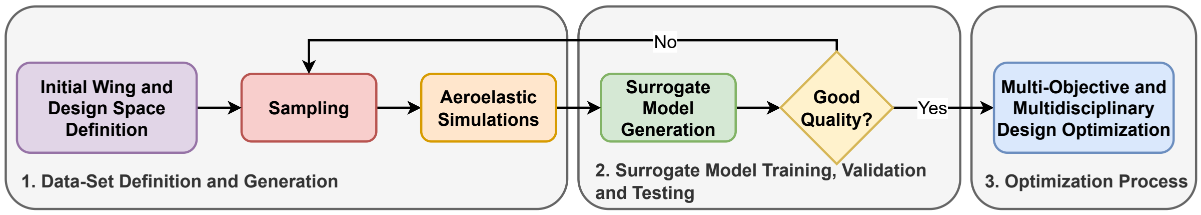


Figure 3.4: Learning curves example.

# Methodology

The methodology employed in this thesis adheres to three fundamental steps commonly found in surrogate-based optimization:

1. Data-set definition and generation
2. Surrogate model training, validation, and testing
3. Optimization process



Initial Wing and Design Space Definition
Sampling
Aeroelastic Simulations
1. Data-Set Definition and Generation
No
Surrogate Model Generation
Good Quality?
Yes
2. Surrogate Model Training, Validation and Testing
Multi-Objective and Multidisciplinary Design Optimization
3. Optimization Process

Figure 4.1: The process flowchart detailing the steps from initial wing and design space definition to multi-objective and multidisciplinary design optimization.

In the following sections, the methodology is elaborated in detail for a cruising speed of 30 m/s, although it should be noted that the same analysis will be conducted at different cruising speeds within this study.

All the aforementioned tasks were accomplished utilizing available open-source Python codes, supplemented by custom Python scripts tailored for this work.

## 4.1 Data-Set Definition and Generation

In this section, we introduce the Goland's Wing as our chosen model and detail the use of SHARPy for aeroelastic analyses. The selection of design space and sampling techniques is also discussed to provide a comprehensive dataset for our study.

### 4.1.1 Wing model: The Goland wing

The *Goland wing* [61], presented in a simplified sketch in Figure 4.2, was chosen as the foundational model for this study. This selection was motivated by:

- The availability of experimental data facilitating the validation of the model.
- Its previous utilization for validating *SHARPy* [62], the elected open-source tool for aeroelastic simulations.

The Goland wing is often considered when discussing aeroelasticity, given its simple rectangular geometry and its wide use in the literature [9, 61, 63, 64]. This is a relatively stiff wing for which Table 4.1 summarises the relevant properties.

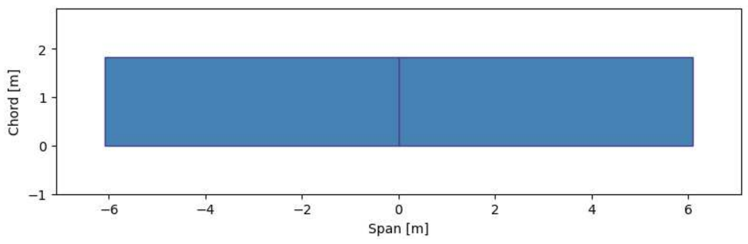


Figure 4.2: Goland wing.

Table 4.1: Goland wing properties.

| Parameter | Value | Units |
|---|---|---|
| Chord | 1.8288 | m |
| Semi-span | 6.096 | m |
| Elastic axis | 33 | % chord from l.e. |
| Centre of gravity | 43 | % chord from l.e. |
| Mass per unit length | 35.71 | kg/m |
| Moment of inertia | 8.64 | kg·m |
| Torsional stiffness | $0.99\times10^6$ | N·m$^2$ |
| Bending stiffness | $9.77\times10^6$ | N·m$^2$ |

### 4.1.2 Aeroelastic tool: SHARPy (Simulation of High Aspect Ratio planes)

SHARPy [62], developed by Imperial College London, is designed for static aeroelastic analyses and nonlinear simulations of flexible aircraft. Using an Unsteady Vortex Lattice Method (UVLM) aerodynamic solver [65], it can model multiple surfaces and their interactions. Simpson et al. [66] describe its unique force evaluation method that accounts for large sideslip angles and induced drag. SHARPy's structure is based on a Geometrically-Exact Composite Beam (GEBM) [67] which supports multibody elements and a mix of mass formulations. Time-integration is done using the Newmark-$\beta$ scheme.

SHARPy was chosen for this research because of its Python framework facilitating automation, and its proven accuracy. For instance, SHARPy's error margin aligns with the developers' claims and is within 5% of experimental results for the Goland's wing.

The subsequent sections outline the fundamental models within SHARPy and their implementation specifics.

#### Aerodynamic Model: Unsteady Vortex-Lattice Method

The UVLM is a computational model used to address 3-D potential-flow problems, particularly for dynamic and deforming lifting surfaces. The approach's foundation is elaborated in Katz and Plotkin [65], while its application on SHARPy is detailed by Murua et al. [9].

At its core, UVLM uses the concept of a *vortex ring.* This is a loop formed by different vortex segments, enclosing to form a structure. Inside this structure, the circulation strength, $\Gamma_k$, remains constant. When the surface moves, it leaves behind an inviscid wake as part of the solution. This wake, represented by vortex rings, forms, sheds, and can even roll based on the prevailing flow speed.

An essential equation in this method is the boundary condition of nonpenetration:

$$A_b\Gamma_b^{n+1} + A_w^{n+1}w = \Gamma_w^{n+1} \ . \tag{4.1}$$

Here, $\Gamma_b$ and $\Gamma_w$ signify the circulation strength of bound and wake vortex rings. $A_b$ and $A_w$ matrices indicate aerodynamic effects, while $X_b$ and $X_w$ are coordinate vectors. Components of these matrices come from projecting velocity, as determined by the Biot-Savart law.

At every time step, vortex rings are added to the wake. This process is aligned with the free-wake model. The wake's influence diminishes over distance, so it's often truncated for computational efficiency. Typically, 20 chord lengths of wake are considered to maintain accuracy in this research.

The shedding and convection of the wake can be summarized as:

$$\mathbf{X}_w^{n+1} = C_b\mathbf{X}_b^{n+1} + C_w\mathbf{X}_w^n + \int_{t^n}^{t^{n+1}} \mathbf{V}(t)\,dt \ . \tag{4.2}$$

This equation dictates how the wake moves. The method usually employs an explicit single-step Euler method, though more advanced methods are available.

Aerodynamic loads can be calculated every time step. UVLM considers loads acting on a local frame, A. The method only retains pressure contributing to local lift, as it's based on thin-wing theory. Drag is aligned with local instantaneous velocity.

The forces are stored in vectors:

$$\mathbf{L}^n = \rho_\infty G_c \left[(U_i\Delta_i + U_j\Delta_j)\Gamma_b^n + \dot{\Gamma} b^n\right], \ \mathbf{D}^n \ = \rho_\infty \left[-U^*\Delta_i\Gamma_b^n + G_s\dot{\Gamma}_b^n\right] \tag{4.3}$$

Where $\Delta_{i(j)}$ are matrices accounting for adjacent panels, and matrices $G_{c(s)}, U_i(j)$, and $U^*$ have specific definitions linked to panel geometry, incidence angles, and weighted velocities as per Katz and Plotkin [65].

### Flexible-Body Dynamics: Displacement-Based Geometrically Exact Composite Beam

Airframe structures are modeled as composite curvilinear beams that undergo large deflections and rotations with the assumption of small local strains. Using a finite-element approach with the Cartesian Rotation Vector (CRV) as primary degrees of freedom, motion equations are deduced. The dynamics of these structures, illustrated in Figure 4.3, involve a moving Frame of Reference (FoR) labeled $a$, relative to an inertial frame $G$ [9].

Hamilton's principle [68] is used to derive equations of motion. The potential and kinetic energy densities per unit length are represented in Equation (4.4).

$$\mathcal{U} = \frac{1}{2}\left\{\gamma^T k^T\right\}\mathcal{S}\begin{Bmatrix}\gamma\\ k\end{Bmatrix}\ ,\ \mathcal{T} = \frac{1}{2}\left\{V_B^T\Omega_B^T\right\}\mathcal{M}\begin{Bmatrix}V_B\\ \Omega_B\end{Bmatrix}\ . \tag{4.4}$$

These energies involve terms such as the inertial velocities $V_B$ and $\Omega_B$, beam strains $\gamma$ and $k$, and the mass and stiffness matrices $\mathcal{M}$ and $\mathcal{S}$ [69]. The orientation of cross sections in the current configuration involves finite rotations denoted by CRV, $\psi(s,t)$.

Strains and velocities are articulated in terms of these rotations and the relative position $R_a(s,t)$. Crucially, the derivation of motion equations considers the position vector in the current state, and the virtual work of applied forces. This depiction of beam dynamics is general and not tied to a specific discretization method. In the study, the motion equations are discretized, involving the tangent mass matrix $\mathbf{M}$ and the use of quaternions $\zeta$ for aircraft orientation. The linearized equation about an equilibrium point involves tangent damping and stiffness matrices $\mathbf{C}$ and $\mathbf{K}$.

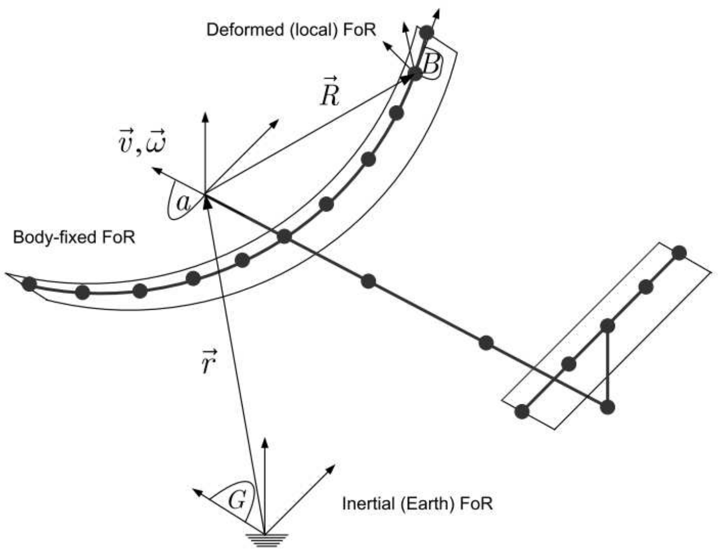


Figure 4.3: Geometrically-exact beam elements [9].

### Coupled Aeroelasticity and Flight Dynamics of Flexible Aircraft

Aeroelastic and flight dynamics for flexible aircrafts are described using combined unsteady aerodynamic and flexibility models. The models necessitate a process to link structural beam nodes and aerodynamic lattices, visualized in Fig. 4.4 (a). The transformation relationship between these two is:

$$(X_b)_a = R_a + C^{aB}(\Psi)\xi_B\ , \tag{4.5}$$
$$(\dot{X}_b)_a = v_a + \tilde{\omega}_a \times R_a + \dot{R}_a + C^{aB}(\Psi)\tilde{\Omega}_B\xi_B\ . \tag{4.6}$$

Where $X_b$ and $\dot{X}_b$ represent aerodynamic lattice grid point deformations and velocities, and $R_a$ and $\Psi$ are beam node displacements and rotations.

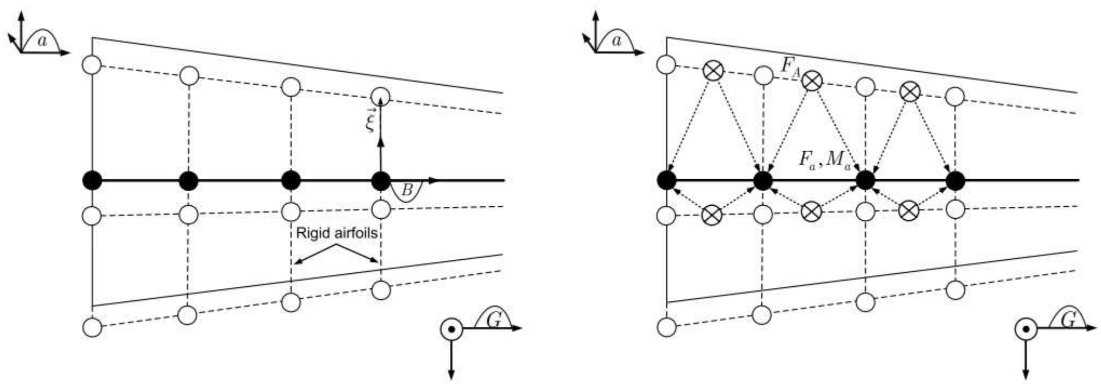


Figure 4.4: Mapping between aerodynamic lattice and structural finite-element discretization [9].

Aerodynamic forces, denoted by:

$$(F_k)^A = \begin{bmatrix} D_k \\ 0 \\ L_k \end{bmatrix} \ , \tag{4.7}$$

are translated to forces at beam nodes. The integration process is summarized by:

$$\begin{Bmatrix} \mathbf{F}_a \\ \mathbf{M}_a \end{Bmatrix} = \lambda \bar{C}^{aA} \mathbf{F}_A \ , \tag{4.8}$$

where specific matrices capture force transformations and integrations.

This mapping enables detailed aeroelastic and flight dynamic analyses. Strategies, such as static aeroelastic and trim analyses, are utilized. Time evaluations use the Newmark-$\beta$ method [70]. Flutter stability is determined using a methodological approach [71] that involves nonlinear solvers, linearization, size reductions of structural and aerodynamic subsystems, and stability analyses at different velocities.

### Linear Stability Analysis of the Goland Wing

Detailed properties of the model are shown in Table 4.1. In this study, the flutter speed is estimated in the time domain with an assumed air density of $\rho = 1.020\,\mathrm{kg/m^3}$. The wing, when clamped at its root, is initiated from a static position with a minimal initial angle of attack, $\alpha = 0.05°$. The coupled aeroelastic model progresses in time through a linear dynamic analysis, excluding rigid-body degrees of freedom. Several freestream velocities are considered until the inception of flutter is observed. The instability originates from a blend of bending and torsion, with the flutter frequency being $\omega_f = 69\,\mathrm{rad/s}$.

Table 4.2 compares the flutter speed and angular frequency deduced with SHARPy against those from prior studies. Both Goland and Luke [72] as well as Goland [61] presented their findings employing an analytical beam model combined with 2-D aerodynamics. It's crucial to highlight that initial outcomes in Goland [61] had errors, later rectified in Goland and Luke [72]. Only the rectified values are displayed in Table 4.2, and there's noticeable alignment with other models rooted in strip theory. Remarkably, while the preliminary (and mistaken) values for the Goland's wing flutter were inadvertently more aligned with the results derived from the UVLM [64], the accurate ones reveal a significant divergence, underscoring the role of 3-D aerodynamic influences.

Table 4.2: Flutter speed of the Goland wing.

| Author | Model | $V_f$, m/s | $\omega_f$, rad/s |
|---|---|---|---|
| Goland and Luke [72] | Analytical | 137.2 | 70.7 |
| Patil et al. [63] | Intrinsic beam + strip theory | 135.6 | 70.2 |
| Wang et al. [64] | ZAERO | 174.3 | - |
| Wang et al. [64] | Intrinsic beam + UVLM | 163.8 | - |
| SHARPy [9] | GEBM + UVLM | 165 | 69 |

In SHARPy, flutter is visualized through the analysis of eigenvalues, damping ratios, and natural frequencies. For the previously introduced analysis, the results are shown in Figures 4.5 to 4.7.

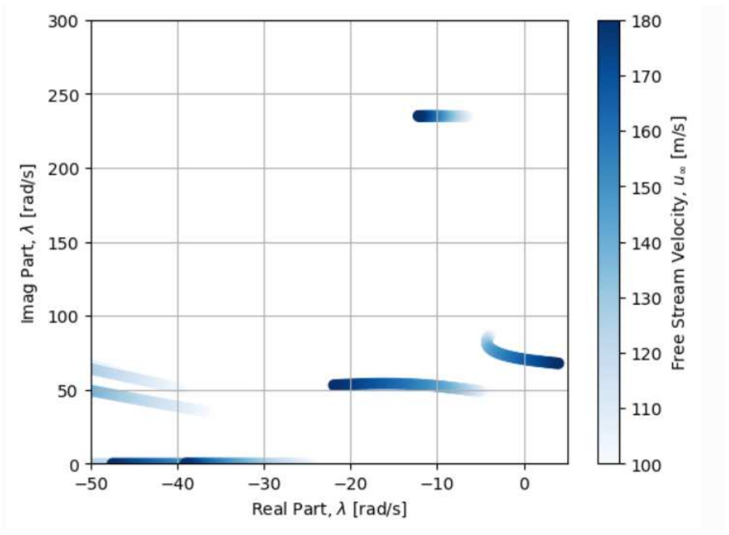


Figure 4.5: Eigenvalues plot.

In the context of aeroelasticity, eigenvalues provide insights into the dynamic characteristics of the aeroelastic system. When the real part of an eigenvalue becomes positive, it indicates that the corresponding mode is unstable, signaling the onset of flutter. Damping ratios offer a measure of the system's resistance to oscillations. When the damping ratio transitions from negative to positive, it indicates potential instability or the onset of flutter. On the other hand, the natural frequency plot illustrates the inherent frequencies at which the system would oscillate in the absence of damping and external forces [1]. It is easy to observe how, precisely, the flutter occurs at the velocity of 165 m/s for the case under consideration.

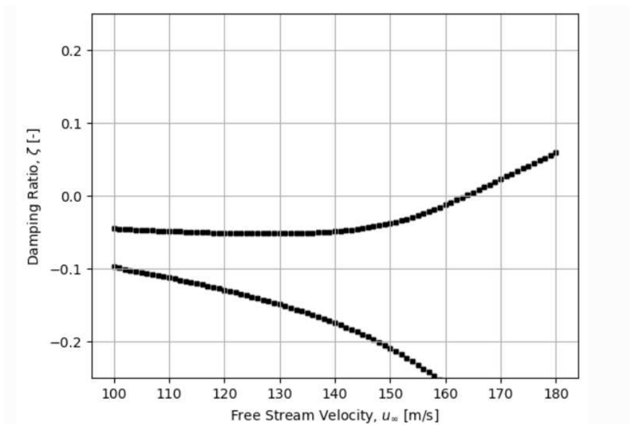


Figure 4.6: Damping ratios plot.

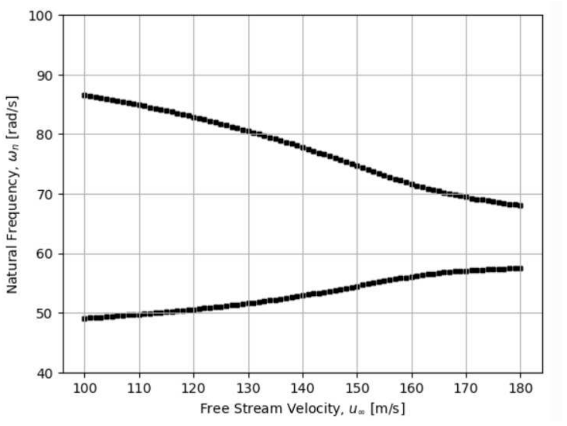


Figure 4.7: Natural frequencies plot.

**Limitations of SHARPy**

Upon conducting the initial aerodynamic simulations of the Goland wing using SHARPy, it was observed that the obtained drag coefficient values, $C_D$, were two orders of magnitude lower than conventional values for several designs. This observation can be primarily attributed to *inviscid model limitations*. In fact, SHARPy's aerodynamic analysis predominantly relies on an inviscid model. As such, viscous effects, which form a significant component of the total drag, especially at low angles of attack, are absent. Moreover, at the cruise speeds evaluated in this study, compressibility effects are minimal and thus not a major contributor to the drag. However, it's essential to note that as cruising speeds approach or exceed transonic velocities, these effects become more significant.

Incorporating viscous factors into the UVLM is normally not very meaningful when the emphasis is on aeroelastic phenomena but to provide a more realistic drag representation a corrective method using *flat plate theory* [73] was employed. The theory, primarily applied to laminar flows, can provide an estimation of skin friction drag, serving as an analog for the missing viscous drag component.

Table 4.3: List of input parameters for the flat plate theory calculations.

| Symbol | Description | Value | Units |
|---|---|---|---|
| $z$ | Altitude | 2000 | m |
| $V_{cr}$ | Cruise speed | 30 | m/s |
| $\rho_{cr}$ | Air density at cruise altitude | 1.02 | kg/m$^3$ |
| $\nu_{cr}$ | Kinematic viscosity at cruise altitude | $1.715 \times 10^{-5}$ | Pa.s |
| $a_{cr}$ | Speed of sound at cruise altitude | 332.5 | m/s |
| $taper$ | Wing taper ratio | 1 | - |
| $c_{mean}$ | Mean aerodynamic chord | 1.8288 | m |
| $tc_{max}$ | Maximum thickness to chord ratio | 0.04 | - |
| $xc_{max}$ | Chordwise location of the max. thickness to chord ratio | 0.5 | - |

The flat plate theory calculations are based on the input parameters showed in Table 4.3.

To perform calculations based on the flat plate theory, a specific Python function was crafted. The main inputs of this function are the aspect ratio ($AR$) and the leading-edge sweep angle ($sweep_{LE}$), that are extracted directly from the data-set since they are treated as design variables. After the computations of the wing span ($b$), the Reynolds number ($Re$) and the Mach number ($M$), the calculations based on the flat plate theory are as follows:

- Effective Mach Number:

$$M_{\text{eff}} = M \times \cos(\text{sweep}_{\text{LE}} \times \frac{\pi}{180}) \tag{4.9}$$

- Friction Coefficient:

$$C_f = \begin{cases} \frac{0.455}{(\log_{10}(Re))^{2.58} \times (1+0.144\times M_{\text{eff}}^2)^{0.65}} & \text{if } Re \geq 3500 \\ \frac{1.328}{\sqrt{Re}} & \text{otherwise} \end{cases} \tag{4.10}$$

- Form Factor:

$$\text{sweep}_{\text{tc\_max}} = \arctan\left(\tan(\text{sweep}_{\text{LE}} \times \frac{\pi}{180}) - \frac{2 \times xc_{\text{max}} \times c_{\text{mean}} \times (1 - \text{taper})}{b}\right) \tag{4.11}$$

$$F = \left(1 + \frac{0.6}{xc_{\text{max}}} \times tc_{\text{max}} + 100 \times tc_{\text{max}}^4\right) \times \left(1.34 \times M_{\text{eff}}^{0.18} \times \cos(\text{sweep}_{\text{tc\_max}} \times \frac{\pi}{180})^{0.28}\right) \tag{4.12}$$

- Wetted Area Ratio:

$$\frac{S_{\text{wet}}}{S_w} = \begin{cases} 1.977 + 0.52 \times tc_{\text{max}} & \text{if } tc_{\text{max}} > 0.05 \\ 2.003 & \text{otherwise} \end{cases} \tag{4.13}$$

- Base Drag Coefficient:

$$C_{D0} = Cf \times F \times Q \times \frac{S_{\text{wet}}}{S_w} \tag{4.14}$$

The interference factor $Q$ is set to 1 because, when considering only the wing, there is no interference with other aircraft's components. After calculating $C_{D0}$ for each set of design variables (aspect ratio and sweep angle) derived from the dataset, it is added to the drag coefficient values obtained from SHARPy. It is important to recognize that this method is an approximation. In the real world, wings deviate from flat plates, and the turbulent boundary layer doesn't cover the entire wing surface due to laminar regions. However, this corrective approach provides a more accurate estimate than the inviscid predictions alone.

### 4.1.3 Design Space and Sampling

Having introduced the selected wing model and the theory behind the code utilized to create the aeroelastic dataset, we now transition to the definition of the design space and the sampling method employed.

The choice of design variables is crucial in an aeroelastic study since they have a significant impact on both aerodynamic and structural properties. The design variables, their boundaries for problems involving the construction of data sets and optimization, and their beginning values that correspond to the Goland's Wing specifications are as listed in Table 4.4.

Table 4.4: Design variables and their boundaries.

| Design Variable | Initial | Lower Boundary | Upper Boundary | Units |
|---|---|---|---|---|
| Aspect-ratio (AR) | 6.67 | 6 | 16 | - |
| Sweep angle ($\Lambda$) | 0 | 0 | 40 | deg |
| Torsional stiffness (GJ) | $0.99 \times 10^6$ | $0.70 \times 10^6$ | $1.70 \times 10^6$ | N.m$^2$ |
| Angle of attack ($\alpha$) | 0.05 | -5 | 15 | deg |

The first parameter chosen was the *aspect ratio (AR)*, which emerged as a key parameter for our aeroelastic study. In light of current economic and environmental constraints, designing more efficient aircraft configurations has become imperative. A noticeable trend in aircraft design is to increase the wing aspect ratio to reduce lift-induced drag, thereby improving fuel efficiency and reducing emissions. However, this design choice presents a challenge from an aeroelastic perspective: a narrower wing is more flexible and is susceptible to greater deformation under the same operating conditions. This effect could potentially alter the aeroelastic response and dynamic behavior of the aircraft, leading to instability [11].

Next, the *sweep angle ($\Lambda$)* was selected as a design variable. This angle is critical in determining the onset of transonic flow phenomena. A larger sweep can delay the onset of such phenomena, allowing aircraft to operate efficiently just below the speed of sound. In addition, the sweep angle has specific implications for stability that designers must manage, especially at high Mach numbers [74].

Structural considerations bring us to the *torsional stiffness (GJ)*. As a structural design parameter, it is inherently linked to how the wing reacts to applied aerodynamic loads. Optimal torsional stiffness ensures that the wing resists torsional movements, which if uncontrolled can lead to catastrophic aeroelastic instabilities, such as flutter [1]. Tuning this parameter is therefore of paramount importance.

Lastly, the *angle of attack ($\alpha$)* plays a significant role in both the lift and drag experienced by the wing. Specifically, $C_L$ rises with increasing $\alpha$ until a critical point, after which it diminishes due to airflow separation, marking the onset of stall. Concurrently, $C_D$ grows, especially past the stalling point, resulting in higher drag. Variations in $\alpha$ directly influence flight stability, with extreme values potentially compromising safe and efficient flight.

The capability of SHARPy to easily modify these design variables was also a key factor in their selection. This feature was essential because it permitted a full and thorough investigation of the design space without becoming bogged down in difficult manual changes.

Regarding sampling, LHS was used to generate a representative data-set for this design space, specifically through the open-source `pyDOE2` package [75]. The decision to employ LHS for sampling is underpinned by several key considerations. LHS, introduced in Section 3.1 is one of the most common sampling techniques, as it ensures a uniform coverage across the design space, avoiding overlaps and guaranteeing that every interval of every variable is equally represented. This is especially crucial for intricate problems like aeroelastic analysis, where random sampling might miss critical regions of the design space. Additionally, LHS is efficient in terms of the number of samples needed to get a good approximation of the design space, an essential advantage given the high computational demand associated with SHARPy.

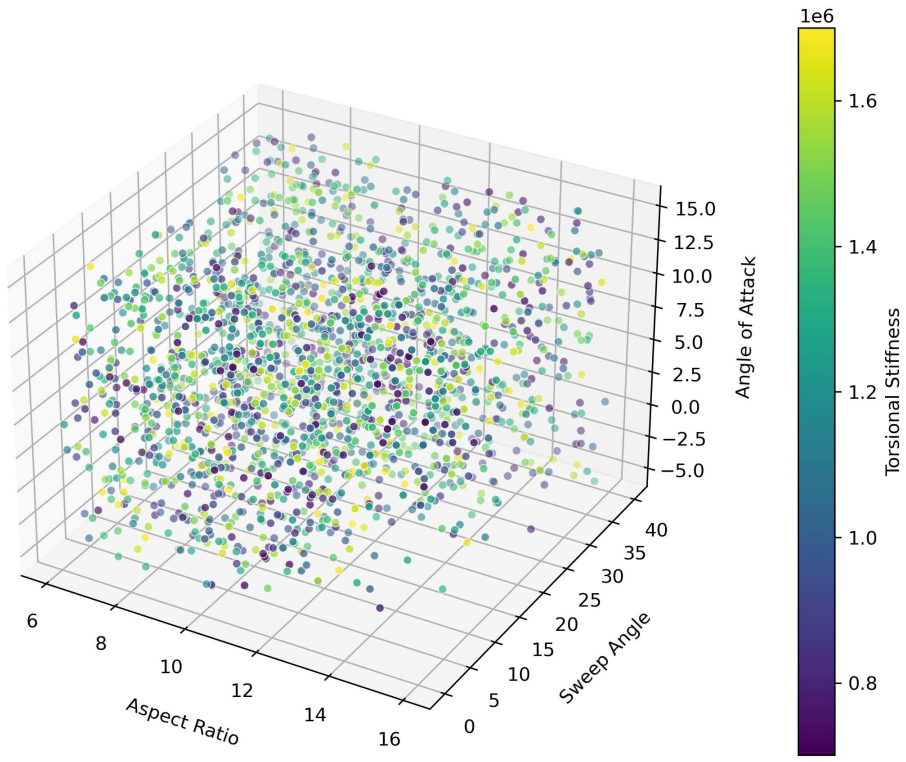


Figure 4.8: 3D distribution of design variables.

At this point, aeroelastic simulations in SHARPy were run for each sample to determine the wing's mass, flutter speed, and the lift ($C_L$) and drag ($C_D$) coefficients under cruising conditions. The first analysis makes the assumption that the cruise speed is 30 m/s. 2000 samples in total were produced for this study. It's crucial to emphasize that, given the complexity of the problem, 2000 samples might seem a modest amount. However, a significant constraint was the high computational cost associated with SHARPy. On average, each iteration took a fairly long time, approximately 93.487 s. Considering the computational intensity of the simulations performed, it is important to delineate the system specifications utilized for the study. The simulations were executed on a host computer that ran a virtual machine with Ubuntu as its operating system. The specifications of the host machine and the virtual environment is presented in Table 4.5.

In order to visualize the distribution of the dataset across the key parameters, the 3D scatter plot shown in Figure 4.8 was constructed using the `matplotlib` library [76]. The plot showcases the relationship and spread of data points in terms of aspect ratio, sweep angle, and angle of attack, with the color gradient representing the torsional stiffness.

Table 4.5: Configuration details of both the physical machine and the virtual environment used for simulations.

| **Parameter** | **Physical Machine** | **Virtual Machine** |
|---|---|---|
| Operating System | Windows 11 | Ubuntu 22.10 |
| Processor | Intel i7-10510U (4 cores, 8 threads) | 4 cores |
| RAM | 16 GB DDR4 | 11 GB |
| Hard Drive | 477 GB SSD | 59.2 GB |
| Graphics | NVIDIA GeForce MX250 | Software Rendering |
| Virtualization Software | N/A | Oracle VM VirtualBox 7.0.6 |

The results generated by SHARPy were then refined using flat plate theory. In addition to the limitations posed by the inviscid model, as discussed previously, the first results obtained in terms of $C_L/C_D$ turned out to be very unrealistic also due to the low angles of attack used for the analysis. In fatc, at these low angles, the induced drag tends to decrease. This is due to the reduced vorticity at the wingtips at lower angles of attack.

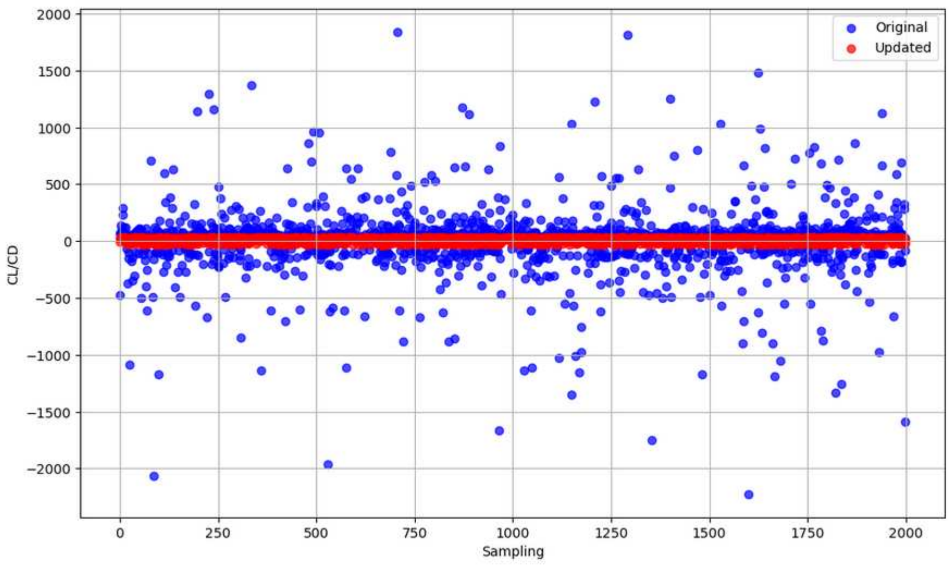


Figure 4.9: Comparison of $C_L/C_D$ values before and after applying the flat plate theory correction.

The adopted strategy improved drag predictions, bringing them closer to real-world values by taking into account missing viscous effects in the UVLM analysis, as can be seen from Figure 4.9 values. The maximum lift-to-drag ratio is now around 40, with a significant improvement compared to the unrealistic initial value of 2000. Consequently, this updated dataset will be used for building surrogate models in the next section.

## 4.2 Comprehensive SMs Analysis: from Selection to Testing

This section delves into a thorough examination of the entire lifecycle of surrogate model development, ranging from the initial phases of selection and evaluation to the nuanced steps of training, validation, and final testing. The ultimate objective of this comprehensive analysis is to identify a surrogate model that not only satisfies our computational requirements but also excels in predictive accuracy. The models discussed in this section will serve as the cornerstone for our subsequent optimization efforts.

### 4.2.1 Selection of SMs methods

The selection of an appropriate surrogate model is paramount to ensure accurate predictions and a streamlined computational effort. Several methodologies exist, each with its advantages and potential limitations. The criteria for selecting our surrogate model included *accuracy*, *efficiency*, and *robustness*.

The process of selecting the most suitable surrogate models for our problem comprised multiple steps, that are illustrated in Figure 4.10.

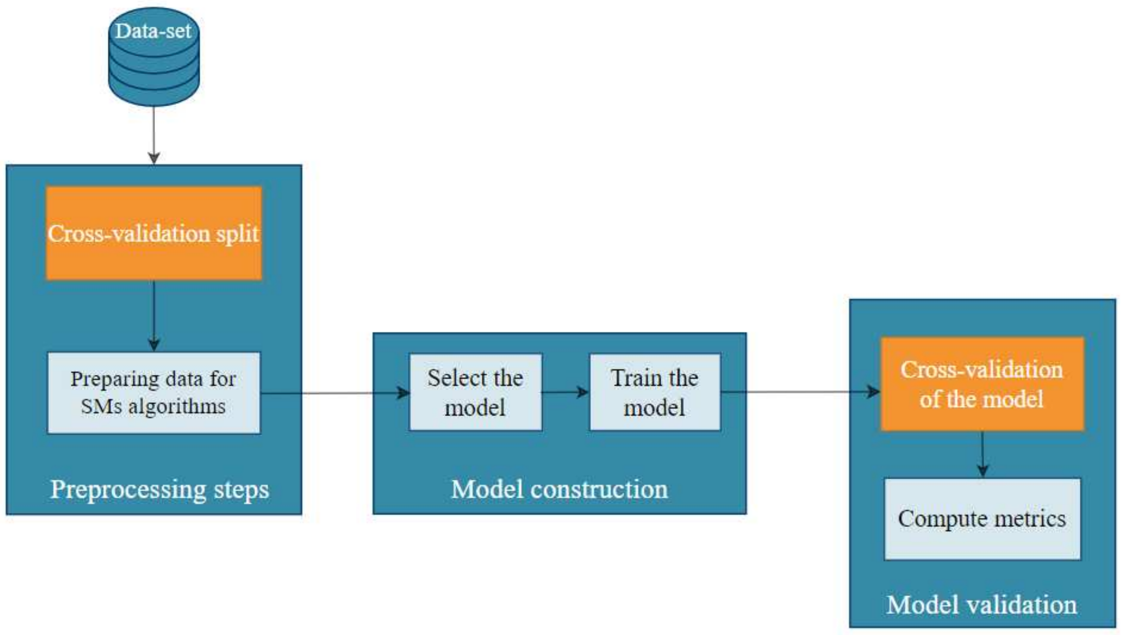


Figure 4.10: Procedure for selection and evaluation of SMs

Various SMs methods, introduced in Section 3.2 and available in scikit-learn Python library [77], were lined up for the comparison, based on both their common application in similar contexts and their intrinsic characteristics conducive for the problem domain.

The evaluated models encompass:

- Bayesian Ridge Regression

- Support Vector Regression (RBF and Polynomial Kernel)
- Decision Tree Regression
- Extra Trees Regression
- Lasso Regression
- ARD Regression
- Linear Regression

In the preliminary phase of surrogate model assessment, a consistent and unbiased approach was taken to ensure that each modeling technique was evaluated under comparable conditions. This was vital to ascertain a fair comparison between the models, eliminating any potential bias introduced by model-specific optimizations.

First, the dataset was divided into training, test, and cross-validation subsets. The division was realized with a proportion of 60%, 30%, and 10% for training, validation, and test sizes respectively, after manually experimenting with various combinations to discern the optimum distribution that resulted in the best-performing model, based on the metrics introduced in Section 3.3. Then, for this exploratory evaluation, model-specific hyperparameters were set in a quasi-random manner, ensuring that they were somewhat consistent across different models. The intent was not to find the absolute best performance for each model but rather to gauge their baseline capabilities. Finally, consistent with metrics presented in Section 3.3, every model was rigorously assessed on the cross-validation subset.

The procedure delineated above was methodically reiterated for each of the four distinct outputs present in the dataset: flutter speed, mass, drag and lift coefficients. All the results are presented in Tables 4.6 to 4.9. The SMs methods in these tables are ordered from the lowest to the highest RMSE value for ease of comparison.

Table 4.6: Flutter speed performance metrics.

| Method | $R^2$ | $MAE$ | $RMSE$ | $MaxError$ | $EVS$ |
|---|---|---|---|---|---|
| ExtraTrees | 0.921 | 2.88 | 8.92 | 88.27 | 0.921 |
| DecisionTree | 0.749 | 4.63 | 15.84 | 141.44 | 0.751 |
| Linear | 0.245 | 17.60 | 27.48 | 126.1 | 0.246 |
| BayesianRidge | 0.245 | 17.67 | 27.48 | 125.8 | 0.246 |
| Lasso | 0.244 | 17.75 | 27.50 | 125.5 | 0.245 |
| ARDR | 0.243 | 17.71 | 27.52 | 126.5 | 0.244 |
| SVR Polynomial | -0.085 | 25.29 | 32.95 | 117.5 | -0.0004 |
| SVR RBF | -0.088 | 25.30 | 32.99 | 117.4 | -0.0004 |

Through a detailed evaluation of the various surrogate modeling techniques, it was evident that the Extra Trees Regressor consistently outperformed the other models for all output parameters. Its superiority was most pronounced in the predictions of flutter velocity, where the disparity in performance between Extra Trees and the other models was evident, as shown in Figure 4.11.

Table 4.7: Mass performance metrics.

| Method | $R^2$ | $MAE$ | $RMSE$ | $MaxError$ | $EVS$ |
|---|---|---|---|---|---|
| ExtraTrees | 0.999 | 0.161 | 0.187 | 0.342 | 0.999 |
| ARDR | 0.999 | 0.161 | 0.187 | 0.330 | 0.999 |
| Linear | 0.999 | 0.161 | 0.188 | 0.342 | 0.999 |
| BayesianRidge | 0.999 | 0.161 | 0.188 | 0.343 | 0.999 |
| Lasso | 0.999 | 0.369 | 0.445 | 1.10 | 0.999 |
| DecisionTree | 0.999 | 0.250 | 0.309 | 1.26 | 0.999 |
| SVR RBF | -0.005 | 150.5 | 176.8 | 362.7 | -0.0008 |
| SVR Polynomial | -0.009 | 150.9 | 177.3 | 367.6 | -0.004 |

Table 4.8: $C_D$ performance metrics.

| Method | $R^2$ | $MAE$ | $RMSE$ | $MaxError$ | $EVS$ |
|---|---|---|---|---|---|
| ExtraTrees | 0.997 | 0.0004 | 0.0008 | 0.008 | 0.997 |
| DecisionTree | 0.989 | 0.0009 | 0.0016 | 0.009 | 0.989 |
| BayesianRidge | 0.786 | 0.005 | 0.007 | 0.025 | 0.787 |
| Linear | 0.786 | 0.006 | 0.007 | 0.025 | 0.787 |
| ARDR | -0.002 | 0.0128 | 0.015 | 0.050 | 0 |
| Lasso | -0.003 | 0.0129 | 0.015 | 0.050 | 0 |
| SVR RBF | -0.903 | 0.018 | 0.021 | 0.0357 | 0 |
| SVR Polynomial | -0.903 | 0.018 | 0.021 | 0.0357 | 0 |

While for the flutter model only the Decision Tree method approached the performance of Extra Trees (still remaining inferior), for the other models the difference between the methods was minimal. In fact, with the exception of SVR, all methods showed excellent predictive ability. With these considerations in mind and for consistency and simplicity in the workflow, we chose to use the *Extra Trees Regressor* as the surrogate model method for all output variables.

### 4.2.2 Training and validation of SMs

Once the Extra Trees Regressor was identified as the most proficient surrogate model for our problem domain, we proceeded to train and validate the models.

**Training:** in the training phase, the model learns from the dataset. We focus on tuning hyperparameters [59] to optimize the model's performance, employing both Grid Search and Random Search methods for this purpose. In fact, for models such as the Extra trees, striking the right balance in depth is essential; shallow trees might underfit, whereas overly deep ones risk overfitting.

During training, the model tries to minimize a loss function, effectively learning to make predictions or decisions based on input data. The settings for the hyperparameters are

Table 4.9: $C_L$ performance metrics.

| Method | $R^2$ | $MAE$ | $RMSE$ | $MaxError$ | $EVS$ |
|---|---|---|---|---|---|
| ExtraTrees | 0.998 | 0.009 | 0.014 | 0.078 | 0.998 |
| DecisionTree | 0.993 | 0.025 | 0.035 | 0.138 | 0.993 |
| BayesianRidge | 0.973 | 0.055 | 0.072 | 0.226 | 0.973 |
| Linear | 0.973 | 0.055 | 0.072 | 0.226 | 0.973 |
| ARDR | 0.951 | 0.071 | 0.097 | 0.349 | 0.951 |
| Lasso | 0.792 | 0.165 | 0.200 | 0.606 | 0.792 |
| SVR Polynomial | 0.0010 | 0.378 | 0.440 | 1.010 | 0.001 |
| SVR RBF | 0 | 0.378 | 0.440 | 0.992 | 0 |

optimized to produce a model that not only performs well on the known training data but also generalizes well to new, unseen data.

**Validation:** after training, it is essential to validate the model's performance using a dataset it has not seen before (known as the test set). This allows us to evaluate the model's generalizability and predictive accuracy. In this phase, performance metrics like RMSE and $R^2$ provide insights into the model's efficacy. Moreover, learning curves can provide additional insights into the model's ability to generalize from the training data to unseen data.

The hyperparameters [52] that we focused on for the Extra Trees Regressor are:

- **n_estimators:** this hyperparameter specifies the number of trees in the ensemble. Increasing this number generally improves the model's performance but at the cost of computational time.
- **max_depth:** the depth of a tree is the maximum distance between the root node and a leaf. A tree with a higher maximum depth will capture more information about the data but may also be prone to overfitting.
- **min_samples_leaf:** this hyperparameter specifies the minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least 'min_samples_leaf' training samples in each of the left and right branches. This helps to smoothen the model and is a method to prevent overfitting.
- **min_samples_split:** it dictates the minimum number of samples required for a node to be split. If a node has fewer samples than 'min_samples_split', the algorithm will cease its splitting and declare the node a leaf. Like min_samples_leaf, it also helps in controlling overfitting.

Each hyperparameter uniquely impacts the effectiveness and efficiency of the model. Careful calibration of these parameters not only bolsters the model's predictive accuracy but also aids in preventing overfitting, making the model more generalizable to new data [52].

The optimal hyperparameters, as summarized in Table 4.10, were identified through a

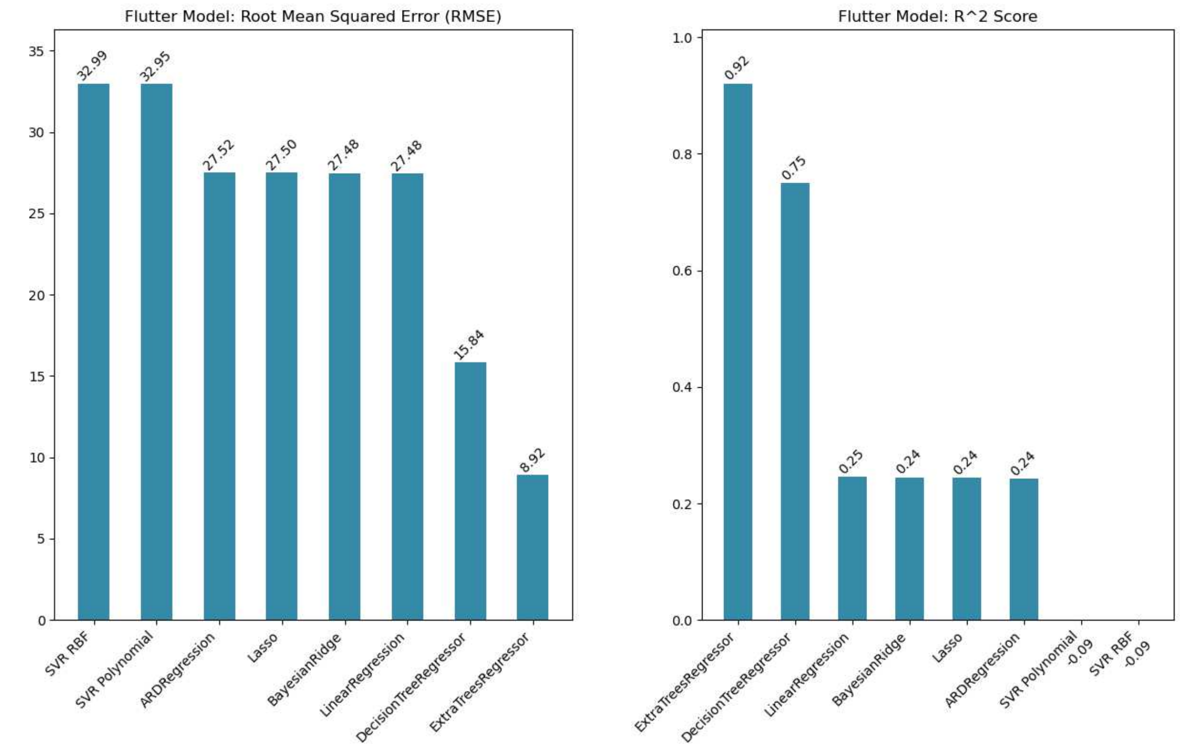


Figure 4.11: Comparison of RMSE and $R^2$ metrics across SMs methods for flutter model.

Grid Search tuning process. Various sets of parameters were tested and evaluated, with the selection criterion being centered around both the Root Mean Square Error (RMSE) and $R^2$ values.

Table 4.10: Optimal hyperparameters for each Extra Trees Regressor model.

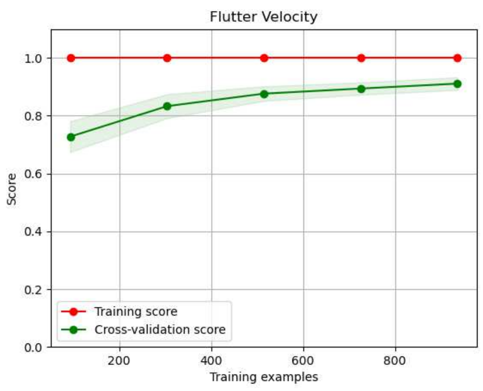
| Model | $n\_estimators$ | $max\_depth$ | $min\_samples\_leaf$ | $min\_samples\_split$ |
|---|---|---|---|---|
| Flutter | 300 | 30 | 1 | 2 |
| Mass | 200 | 20 | 2 | 4 |
| $C_D$ | 300 | 20 | 1 | 2 |
| $C_L$ | 200 | 30 | 2 | 4 |

Upon training the Extra Trees Regressor models with the optimal hyperparameters, we proceeded to evaluate their performance on the test set. The results are summarized in terms of RMSE and $R^2$ metrics in Table 4.11.

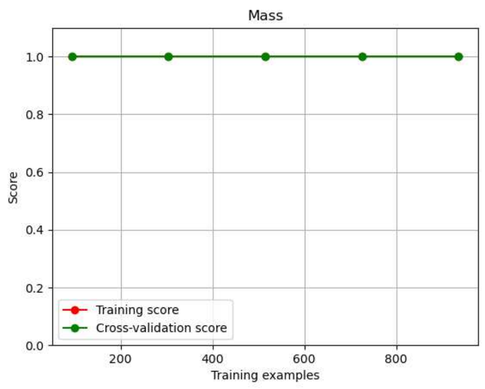
Table 4.11: Final RMSE and $R^2$ metrics for each Extra Trees Regressor model at 30 m/s.

| Model | $RMSE$ | $R^2$ |
|---|---|---|
| Flutter | 8.876 | 0.921 |
| Mass | 0.293 | 0.999 |
| CD | 0.0008 | 0.997 |
| CL | 0.0141 | 0.998 |

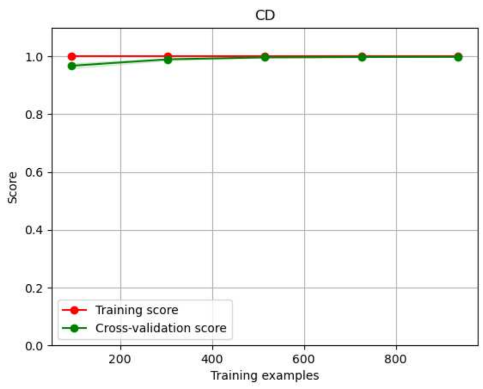
Additionally, the learning curves for each model were plotted to assess their generalization performance. These curves are presented in Figure 4.12.



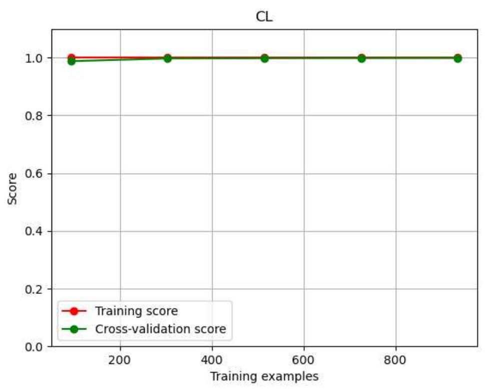
(a) Flutter Velocity



(b) Wing Mass



(c) Drag Coefficient ($C_D$)



(d) Lift Coefficient ($C_L$)

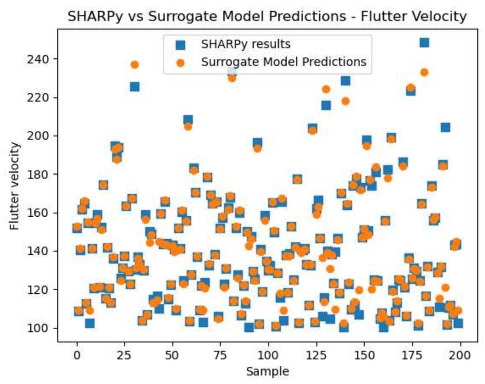
Figure 4.12: Learning curves of SMs.

### 4.2.3 Testing of SMs

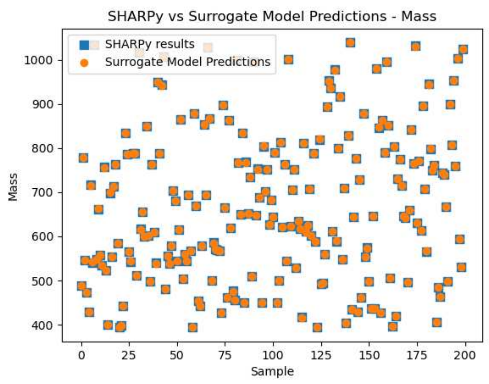
To have a more clear understanding of the suitability of the chosen SMs for aeroelastic analyses throughout the design space, a visualization of parameters used to compose the objective functions ($C_L/C_D$ and mass) and constraint (flutter speed) is provided in Figure 4.13 for the testing dataset.

As one can notice from Table 4.11 and Figure 4.13, the SMs predict well both the mass ($R^2 = 9.99\times10^{-1}$) and lift coefficient ($R^2 = 9.98\times10^{-1}$). Slightly higher discrepancies between surrogate and SHARPy results ($R^2 = 9.97\times10^{-1}$) are observed in the drag coefficient, especially for higher values.

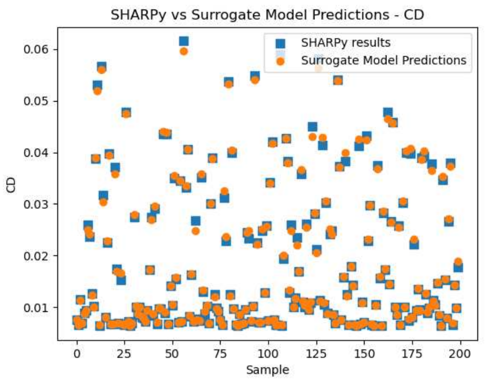
Furthermore, during the validation phase, some notable relative errors were detected for the lowest values of drag coefficient. These errors, while significant in individual cases, were diluted in the computation of the MAE, suggesting that the overall average error remains low. Nonetheless, these errors at low and high drag coefficient values indicate that there might be specific conditions where the surrogate model's accuracy is less reliable.



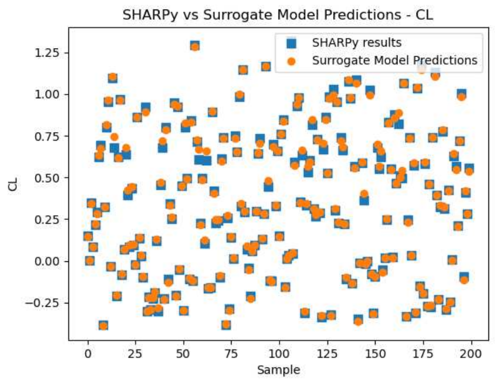
(a) Flutter Velocity



(b) Wing Mass



(c) Drag Coefficient ($C_D$)



(d) Lift Coefficient ($C_L$)

Figure 4.13: Visual comparison of the results obtained using the SHARPy simulations (in blue) and the surrogate model (in orange) in the testing phase of the SMs.

The observed discrepancies in the drag coefficient may be attributed to nonlinearities in aeroelasticity in specific regions of the design space, particularly where some aerodynamic and structural dynamic interactions become particularly complex. This hypothesis is consistent with trends observed in the data for high and low cd values, and suggests a potential area for further investigation to improve the accuracy of aerodynamic modeling and, consequently, surrogate model predictions.

However, it is for the flutter speed that the differences are higher ($R^2 = 9.21 \times 10^{-1}$).

In addition, while for mass, $C_D$ and $C_L$ the stabilized learning curves shown in Figure 3.4 demonstrate excellent generalizability and that additional data would probably not result substantial improvement results, for the flutter model suggests overfitting. This can be seen by the consistently high training score and a significantly lower validation score. The validation score curve doesn't reach a plateau, making it clear that adding new data to the dataset can improve the performance of the surrogate model, also indicated by the fairly high RMSE of 8.876.

It is worth noting that the challenges in optimizing the flutter model are likely due to the inherent complexity in estimating flutter speed using the p-k method, an iterative algorithm. This could be a contributing factor to the model's less-than-ideal performance. These results suggest the need for further refinement of the model, perhaps using a larger dataset, or possibly the exploration of more specialized modeling approaches.

Nevertheless, within the scope of this thesis, the results obtained for all models, including flutter and $C_D$ model, are considered acceptable.

The entirety of this multi-faceted analysis, spanning model training to evaluation, was orchestrated using the aforementioned `scikit-learn` library in Python [77].

## 4.3 Optimization Process

Optimization is a paramount step in the engineering design process, seeking to enhance or maximize specific desired outputs while meeting specified constraints. The SMs constructed and analyzed in the previous section represent the foundation of our optimization problem, thus embodying a "surrogate-based optimization" approach. Among the various optimization techniques discussed in Section 4.2, the choice was made to engage in *multi-objective optimization.* This choice was dictated by the need to address both the aeroelastic aspect through a flutter constraint, as well as to ensure good aerodynamic performance and structural requirements. The adoption of multi-objective optimization provides a deeper examination, revealing the fundamental trade-offs for aerospace design. In addition, this method facilitates more effective exploration of the solution space, helping to identify novel configurations that might remain unknown in a single-objective optimization context. The ability to balance competing objectives is essential in a field where the alignment of performance, cost, and safety is critical to the feasibility and success of a project [20].

In our work, the primary aim was to maximize aerodynamic efficiency, quantified by the lift-to-drag ratio, and minimize structural weight of the wing design. The formal problem is presented as:

$$\begin{array}{ll} \text{minimize} & f\left(-C_L/C_D,\ \text{mass}\right) \\ \text{with respect to} & x = (\text{AR},\ \Lambda,\ GJ,\ \alpha) \\ \text{subject to} & c = V_{cr} - (V_{flutter}/1.5) \leq 0 \end{array} \quad , \tag{4.15}$$

where $f$, $x$, and $c$ denote the classical notation for objective functions, design variable set, and constraints, respectively. In our constraint formulation the factor of 1.5 is a common safety margin employed in the aerospace industry to ensure a sufficient buffer against flutter phenomena, as is standard practice. This safety margin provides an additional level of assurance in meeting the stringent safety and performance standards prevalent in aerospace engineering [78].

The choice of our optimization tool fell on NSGA-II [21], introduced in Section 2.3, which is implemented through Pygmo [79]. Several key features of NSGA-II strongly influenced its use in this research. First, its gradient-free approach assures a thorough and solid exploration of the design space, avoiding the pitfalls of local optima and negating the need for challenging gradient calculations in complex models. Moreover, NSGA-II is praised for its ability to sustain diversity among solution sets, which is vital to ensure a broad search and prevent settling for less-than-optimal solutions. It also prioritizes solutions based on non-domination, guaranteeing a well-rounded and evenly distributed suite of solutions along the Pareto front and offering a varied selection of feasible options for designers during the decision-making process. Furthermore, NSGA-II is frequently utilized in complex multi-objective problems due to its robustness and capability to manage numerous, often conflicting, objectives.

The main features of this algorithm are shown in the following section to better understand its operation.

### 4.3.1 NSGA-II

Before delving into the specifics of NSGA-II, it is pertinent to introduce some fundamental concepts intrinsic to evolutionary algorithms, which form the underpinning of NSGA-II [21].

- **Population Size:** the population size refers to the number of potential solutions (individuals) in a particular generation. A larger population size ensures a diverse search space, though at the cost of increased computational resources.
- **Generation:** a generation is a distinct phase in the evolutionary cycle where a population of solutions is evaluated, selected, and evolved to form a new set of solutions. The process iterates over several generations to refine the solutions towards optimization.
- **Mutation Rate:** mutation is a stochastic process that introduces small random modifications in the individuals' characteristics, promoting diversity in the population and aiding in escaping local optima. The mutation rate dictates the likelihood of these modifications occurring.
- **Crossover Rate:** crossover is a process where traits from two or more parent solutions are combined to form one or more offspring solutions. The crossover rate determines the extent to which this recombination occurs, fostering the exploration and exploitation of the search space.

These parameters are essential in controlling the behavior and performance of the evolutionary algorithm. Furthermore, tuning these parameters can significantly impact the convergence speed and solution quality.

NSGA-II is one of the most popular optimization algorithms, mainly because of its low computation time. It was initially proposed for problems with 2-3 objectives. However, numerous improvements and extensions have been proposed to make the algorithm efficient for solving multi-objective problems with a larger number of objectives. The algorithm has a complexity of $O(MN^2)$, where $M$ is the number of objective functions and $N$ is the size of the population [2].

This complexity is particularly advantageous, especially in cases with a small number of objectives. In our case, the presence of two objective functions allows the algorithm to maintain a limited level of complexity, and consequently of computational cost. Of course, for this to happen, the size of the population must also be kept in check.

NSGA-II operates on the basis of four basic principles: the undominated ordering, the elite preservation operator, the crowding distance and the selection operator, briefly illustrated below [10].

- **Non-Dominated Sorting:** population members are sorted based on Pareto dominance in this stage. The procedure initiates by assigning the first rank to the non-dominated members of the initial population, grouping them into the first front while removing them from the initial population. The sorting continues with the remaining population members, assigning subsequent ranks, and grouping them into respective fronts until all members are ranked and grouped, as depicted in Figure 4.14 (a).
- **Elite-Preserving Operator:** this strategy retains elite solutions of a population by migrating them directly to the succeeding generation, implying that the non-dominated solutions from each generation are carried over to the next until they are dominated by other solutions.
- **Crowding Distance:** this distance measures the density of solutions surrounding a particular solution, computed as the average distance of two solutions on either side of a given solution along each objective. A solution with a larger crowding distance is considered to reside in a less crowded region. The crowding distance for the $i$th solution is the average side-length of the cuboid as illustrated in Figure 4.14 (b), calculated as,

$$cd(i) = \sum_{i=1}^{k} \frac{f_j^{i+1} - f_j^{i-1}}{f_j^{\max} - f_j^{\min}}, \tag{4.16}$$

  where $k$ is the number of objective functions, and $f_j^{\max}$, $f_j^{\min}$ are the maximum and minimum values of the $j$th objective function among all individuals respectively.
- **Selection Operator:** selection for the next generation is performed using a crowded tournament selection operator, which employs the rank and crowding distances of population members. The selection rules are as follows:
  1. If the members belong to different ranks, the one with the superior rank is selected for the next generation.
  2. If both members have identical ranks, the one with the higher crowding distance is selected for the next generation.

### Procedure of NSGA-II

The procedure of the chosen algorithm is summarized below. NSGA-II begins with the generation of an initial population $P_t$ of size $N$. A new population $Q_t$ is subsequently created through crossover and mutation operations on $P_t$. Both populations $P_t$ and $Q_t$ are merged to form a new population $R_t$, which then undergoes the non-dominated sorting procedure. Members of $R_t$ are ranked into different fronts based on their non-domination levels [10].

The subsequent step involves selecting $N$ members from $R_t$ to form the next population $P_{t+1}$. If the first front size is greater than or equal to $N$, only $N$ members from the least

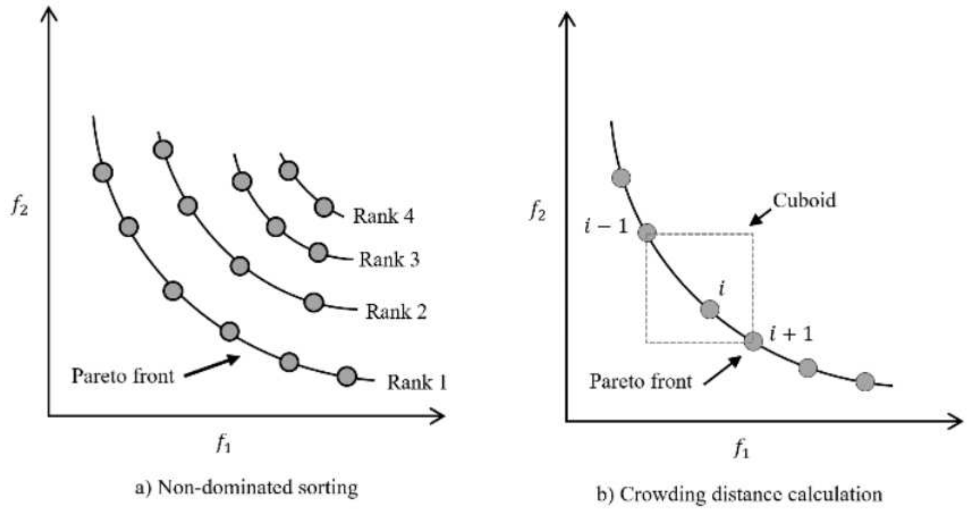


Figure 4.14: Non-dominated sorting procedure and Crowding distance calculation [10].

crowded region of the first front are selected to form $P_{t+1}$. However, if the first front size is smaller than $N$, all members of the first front are transferred to the next generation, and the remaining members are selected from the least crowded region of the second front to complete $P_{t+1}$. This process continues with subsequent fronts until $P_{t+1}$ has $N$ members. Populations $P_{t+2}$, $P_{t+3}$, $P_{t+4}$, ..., for the next generations are assembled using the same procedure, until the stopping criteria are met, as visualized in Figure 4.15 [10].

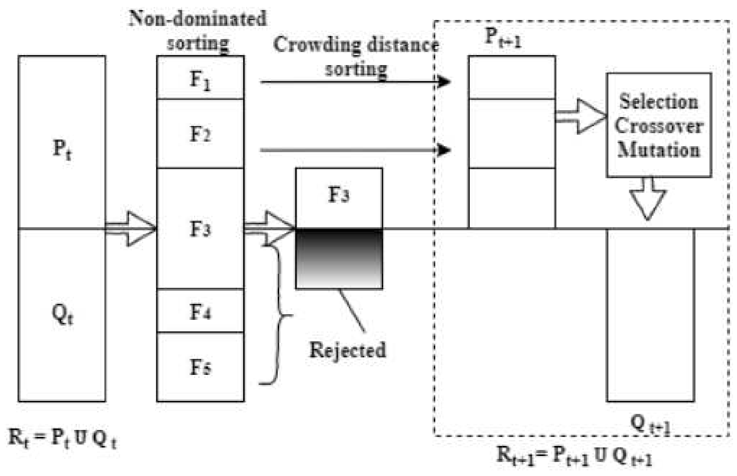


Figure 4.15: Procedure of NSGA-II [10].

## 4.3.2 Optimization methodology

Transitioning from the core mechanics of the NSGA-II algorithm, the practical implementation of this optimization framework in the current study follows a methodical approach outlined below. This methodology is based on the surrogate models created before. These models, generated from the data procured through SHARPy simulations and some corrections (such as the flat plat theory application), endeavor to mimic the system behavior within targeted regions, with a significant reduction in computational cost.

The role of key parameters such as population size, generation, mutation rate, and crossover rate, as previously elucidated, comes to the forefront in the hyperparameter tuning phase. An exhaustive tuning exercise is carried out to balance the trade-off between computational cost, hypervolume, and ranking. *Hypervolume* measures the space covered by a set of solutions in the objective space, effectively capturing both the convergence and the diversity of the solution set. A larger hypervolume indicates a better spread of solutions along the Pareto front and closer convergence to the true Pareto front. Instead, *ranking* is performed based on non-dominance levels, where a lower rank (i.e., rank 1) indicates better performance. It helps in differentiating between solutions and selecting the better ones for creating subsequent generations [10].

Mirroring the SHARPy simulations, boundaries for the optimization design variables are established: aspect ratio, sweep angle, torsional stiffness, and angle of attack. An initial population is crafted using LHS, ensuring a well-dispersed set of starting points for the optimization process. The optimization variables are interfaced with the simulation environment, setting the stage for the ensuing aeroelastic analyses within the defined boundary conditions.

In keeping with the iterative nature of generations in NSGA-II, an optimization driver iteratively refines design variables, aiming to maximize the lift-to-drag ratio ($C_L/C_D$) and minimize structural weight. The integration of flutter constraints through a penalty method ensures the aeroelastic integrity of evolving designs. If flutter constraints are violated, a penalty is applied that moves the optimization away from undesirable design territories. Specifically, a penalty equal to $10^4 \cdot |V_{flutter} - V_{cruise} \cdot 1.5|$.

Optimization continues until a predefined iteration limit is reached, which marks the culmination of the optimization process according to the stopping criteria outlined in the NSGA-II procedural scheme, as described earlier in its procedure. In our endeavor, the NSGA-II algorithm treats each objective with equal importance, offering a Pareto front of solutions that provide a balanced trade-off between aerodynamic efficiency and structural weight of the wing design [21].

Upon completion of optimization, the resulting solutions are examined to evaluate their aerodynamic, structural, and aeroelastic merits, providing a consistent interpretation of the proposed wing designs. The "best" solutions are then selected, specifically the one with the lowest mass and the one with the highest lift-to-drag ratio. The associated decision vectors are then entered into the code used to create the dataset, initiating a comparison to assess the accuracy of the optimization.

The generated Pareto front, a manifestation of NSGA-II's undominated sorting mechanism, shows the tradeoffs between lift-to-drag ratio and structural weight. This front represents a number of trade-offs between the considered objectives, and the selection of a specific solution among those in the Pareto front will require further evaluation based on project-specific preferences or requirements. This aspect underscores the importance and practicality of the multi-objective approach, which allows for better exploration and understanding of the inherent trade-offs between conflicting project objectives.

### 4.3.3 Refinement of Design Space

After obtaining the initial Pareto fronts, a verification was conducted by selecting the "best" solutions, specifically the one with the lowest mass and the one with the highest lift-to-drag ratio. The associated decision vectors were then entered into the code used to create the dataset, initiating a comparison to assess the accuracy of the optimization.

From the analysis, a high percentage discrepancy for some results emerged. It was therefore proceeded to examine how the optimization solutions were distributed within the design variable boundaries. It was observed that the majority of solutions were concentrated near the lower boundary for sweep angle and torsional stiffness, as can be observed in Figure 4.16, suggesting that extending these boundaries might lead to better solutions. This observation guided the decision to extend the design variable boundaries and add new values to the dataset, in an attempt to further explore the design space and potentially obtain better optimization results.

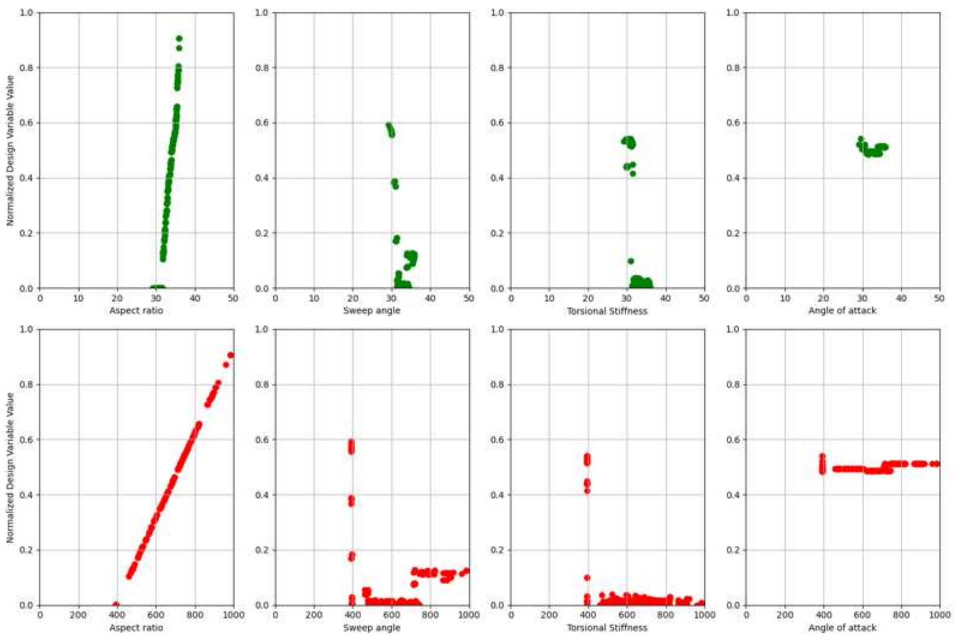


Figure 4.16: Distribution of optimization solutions along design variables boundaries for both objectives: max CL/CD (above) and min mass (below).

Furthermore, the testing of SMs showing a high prediction error at high aspect ratios represented a significant finding in the analysis process. The design variables associated with points with an error greater or equal to 5% in the flutter velocity model clearly showed that the high aspect ratio is a critical area for model accuracy, as shown in Table 4.12.

Consequently, in introducing new data with extended lower boundaries for sweep angle and torsional stiffness, the range for aspect ratio was narrowed down to between 13 and 16. This change aims to densify the dataset in the identified critical area, allowing for a more accurate representation and potentially improving the surrogate model's prediction capability in this aspect ratio range.

Table 4.12: Design variables for high error points ($\geq 5\%$) in flutter model.

| Aspect Ratio | Sweep Angle (deg) | Torsional Stiffness (N.m$^2$) | Angle of Attack (deg) |
|---|---|---|---|
| 15.83 | 34.8 | $7.67 \times 10^5$ | 10.97 |
| 15.42 | 13.76 | $9.06 \times 10^5$ | 3.58 |
| 15.74 | 35.78 | $1.34 \times 10^6$ | 6.56 |
| 15.23 | 26.74 | $7.79 \times 10^5$ | 9.79 |
| 15.32 | 34.34 | $8.67 \times 10^5$ | 9.27 |
| 13.67 | 24.99 | $7.89 \times 10^5$ | 13.78 |
| 14.96 | 27.53 | $8.36 \times 10^5$ | 10.07 |
| 14.02 | 2.24 | $1.31 \times 10^6$ | -1.36 |
| 13.45 | 24.49 | $9.54 \times 10^5$ | 13.63 |
| 15.0 | 15.35 | $1.17 \times 10^6$ | -0.98 |
| 13.03 | 8.21 | $9.26 \times 10^5$ | 9.64 |
| 14.46 | 1.14 | $1.27 \times 10^6$ | 11.88 |
| 13.77 | 5.31 | $1.66 \times 10^6$ | 5.96 |
| 15.67 | 34.84 | $1.60 \times 10^6$ | 8.74 |

This iterative process of analysis and adjustment of the dataset and design variable boundaries demonstrates a methodical approach to solving the issues emerged during optimization. The ability to identify areas of weakness in the models and make informed adjustments to address these weaknesses is fundamental for the success of the research project.

# Results

This chapter details the outcomes of optimization procedures created for various cruising speeds, starting from 30 m/s and going through 60 m/s and 130 m/s. The optimization process, which was previously described in Chapter 4, was used with a set of parameters and was based on SMs in order to get the best possible design solutions. These answers were then contrasted with reference results, which were primarily produced from SHARPy simulations.

## 5.1 Optimization Results at 30 m/s Cruise Speed

The optimization process, whose methodology was described in detail in the previous chapter, was first executed for a cruise velocity of 30 m/s. The SMs used for such optimization are those introduced in Table 4.11.

A combination of a population size of 180 and a generation count of 50, with a crossover rate of 0.8 and a mutation rate of 0.5, was selected for the genetic algorithm, post hyperparameter tuning. The algorithm converged to an optimal set of solutions within a computational duration of 328.2 seconds.

The solutions derived from the optimization process were compared with the benchmark results obtained with SHARPy[1] to evaluate their accuracy and effectiveness. Table 5.1 provides a comparative analysis of the optimized solutions against the reference values, considering the results for the solutions that maximize CL/CD and minimize mass.

Table 5.1: Comparative analysis between optimized solutions and SHARPy results at 30 m/s.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 39.35 | 39.41 | 0.14 |
| | Mass (kg) | 1025.85 | 1025.62 | 0.02 |
| | Flutter Speed (m/s) | 123.12 | 143.86 | 16.84 |
| min Mass | CL/CD | 23.73 | 27.04 | 13.96 |
| | Mass (kg) | 391.83 | 392.21 | 0.09 |
| | Flutter Speed (m/s) | 186.25 | 185.02 | 0.66 |

The table shows minimal deviation between the optimized solutions and the benchmark results in terms of CL/CD and mass, indicating high accuracy of the optimization results. However, a significant discrepancy is observed in the flutter rate, especially in the CL/CD max objective. On the other hand, as for the solutions related to the minimum mass value, it turns out that the CL/CD error is high, while the other two are acceptable.

[1]Throughout the document, SHARPy also incorporates the implemented flat plate theory correction.

Consequently, a further analysis was undertaken to enhance the optimization results, adopting the methodology introduced in Section 4.3.3.

In response to those results, the boundaries of the design variables were expanded in terms of sweep angle and torsional stiffness, and additional data points were incorporated into the dataset, respecting aspect ratios of 13 to 16. The aim was to enhance the dataset in pivotal areas, enhancing predictive accuracy within the specified aspect ratio range, and conferring the optimization algorithm greater liberty to traverse between boundaries.

The extended boundaries utilized for generating the additional 500 data points, subsequently combined with the preceding dataset, are reported in Table 5.2. The values changed from the initial ones, shown in Table 4.4, are displayed in red.

Table 5.2: New design variables boundaries.

| Design Variable | Lower Boundary | Upper Boundary | Units |
|---|---|---|---|
| Aspect-ratio (AR) | 13 | 16 | - |
| Sweep angle ($\Lambda$) | -10 | 40 | deg |
| Torsional stiffness (GJ) | $0.50 \times 10^5$ | $1.70 \times 10^6$ | N.m$^2$ |
| Angle of attack ($\alpha$) | -5 | 15 | deg |

The new optimization, using the same NSGA-II hyperparameters as the first one, took a computational time of *343.32 seconds*, and was based on the SMs trained on the new dataset, whose metrics are reported in Table 5.3. There is a visible improvement in flutter speed estimation capability compared to the previous model (Table 4.11).

Table 5.3: Final RMSE and $R^2$ metrics at 30 m/s.

| Model | $RMSE$ | $R^2$ |
|---|---|---|
| Flutter | 7.276 | 0.952 |
| Mass | 0.293 | 0.999 |
| CD | 0.0006 | 0.998 |
| CL | 0.0059 | 0.999 |

Table 5.4 elucidates a comparative analysis between the optimized solutions and benchmark results, reflecting the impacts of the boundary adjustments.

The revised results demonstrate a notable improvement, with reduced percentage errors across most parameters in comparison to the reference results obtained through SHARPy. However, it is pertinent to note that the flutter speed, especially for the solution aiming at maximizing CL/CD, still maintains a percentage error above 5%.

Although changes to the boundaries of the design variables and expansion of the dataset produced significant improvements, further analyzes were then performed by increasing the number of generations and the size of the NSGA-II population. It is hypothesized that giving the algorithm more time and resources to explore the solution space could lead to even better results, particularly in reducing the error in estimating the flutter speed.

Table 5.4: Comparative analysis between optimized solutions and SHARPy results with adjusted boundaries at 30 m/s.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 39.53 | 39.72 | 0.48 |
| | Mass (kg) | 1021.16 | 1021.26 | 0.01 |
| | Flutter Speed (m/s) | 110.13 | 119.84 | 8.81 |
| min Mass | CL/CD | 27.54 | 28.45 | 3.30 |
| | Mass (kg) | 393.84 | 392.23 | 0.10 |
| | Flutter Speed (m/s) | 175.57 | 173.35 | 1.26 |

The population size was increased to 280, and the generation count was augmented to 200, while crossover and mutation rates were maintained.

Table 5.5: Comparative analysis between optimized solutions using the new parameters and SHARPy at 30 m/s.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 39.83 | 39.79 | 0.09 |
| | Mass (kg) | 1043.57 | 1043.46 | 0.01 |
| | Flutter Speed (m/s) | 133.65 | 127.89 | 4.30 |
| min Mass | CL/CD | 26.37 | 27.13 | 2.85 |
| | Mass (kg) | 313.84 | 392.17 | 0.08 |
| | Flutter Speed (m/s) | 188.41 | 188.10 | 0.16 |

The new results, as delineated in Table 5.5, demonstrate that all metric deviations were effectively confined within a 5% error margin, underscoring the efficacy of the optimization strategy employed. While the results improved, there was an increase in computational time. The new computational duration stands at *1801.28 seconds*, compared to the initial time of 343.32 seconds, indicating a trade-off between result accuracy and computational efficiency. In Figure 5.1 the Pareto fronts obtained for both optimization are showed.

Analyzing the Pareto fronts depicted in Figure 5.1, it's evident that the graph illustrates the inherent trade-offs between the objectives under consideration. As one metric improves, there's a certain level of compromise in another, which is typical of multi-objective optimization problems.

The results from the expanded configuration (in orange) seem to cover a wider range of the Pareto front than the initial configuration (in blue). This suggests that the increased number of generations and a larger population size allowed the algorithm to explore a more comprehensive solution space and thus achieve better results, especially towards higher CL/CD values.

Both configurations have dense regions, where multiple solutions are closely packed. This is particularly noticeable in the middle section of the CL/CD axis. However, there are

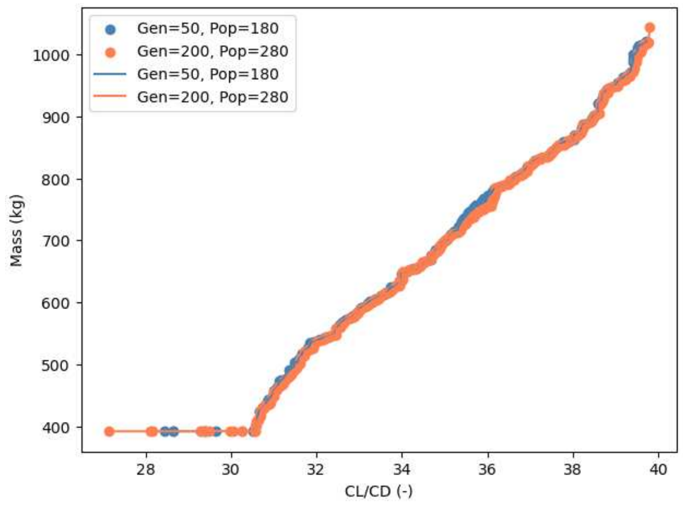


Figure 5.1: Pareto fronts at 30 m/s.

also some outlier points, especially noticeable for the blue configuration, which might represent specific unique solutions the algorithm identified.

Noteworthy is the observation that throughout the optimization process, the flutter constraint was never active. This led us to study the behavior of the Goland wing at higher cruise speeds.

## 5.2 Optimization Results at 60 m/s Cruise Speed

The optimization process conducted at cruise speeds of 60 m/s replicates the methodology delineated in Chapter 4 for 30 m/s, adhering to the initial design space and selection method of SMs. The latter, as supported by the results, again underscored the *Extra Trees Regressor* as the prevailing model, showcasing superior predictive precision in the examined scenarios. The models performance, after an hyperparameter tuning, are reported in Table 5.6.

Table 5.6: Final RMSE and $R^2$ metrics for each surrogate model at 60 m/s.

| Model | $RMSE$ | $R^2$ |
|---|---|---|
| Flutter | 7.360 | 0.913 |
| Mass | 0.377 | 0.999 |
| CD | 0.0034 | 0.947 |
| CL | 0.013 | 0.999 |

The optimization procedure was executed utilizing NSGA-II, wherein the algorithmic parameters, namely the population size, generation count, crossover rate, and mutation rate, were kept consistent with the 30 m/s scenario to maintain a coherent comparative framework across different cruise speeds.

In this analysis, after an initial optimization with the initial design space that had presented excessive percent errors, the same approach as in the 30 m/s analysis was applied. The boundaries of the design space have been extended to negative sweep angles and the dataset density has been intensified between for higher aspect ratios, which was identified as the most critical part for the surrogate model.

In optimization using NSGA-II, parameters were retained from the initial analysis to ensure consistency between different scenarios while simultaneously aiming to reduce computational costs. Indeed, this remained within an acceptable range, for a total of *309.89 seconds*. In particular, the results presented here were then subjected to a design space refinement process which, such as the 30 m/s scenario, proved effective.

Table 5.7 reveals that the percentage errors remain relatively small, except for the flutter speed in the solution at maximum CL/CD. Again, SHARPy results include those of the flat plate theory correction, and considering a Mach number of 0.18, the effect of compressibility is excluded.

Table 5.7: Comparative analysis between optimized solutions and SHARPy results with adjusted boundaries at 60 m/s.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 36.49 | 36.00 | 1.33 |
| | Mass (kg) | 1008.61 | 1008.55 | 0.01 |
| | Flutter Speed (m/s) | 89.17 | 96.22 | 7.90 |
| min Mass | CL/CD | 24.86 | 26.13 | 5.10 |
| | Mass (kg) | 392.35 | 392.50 | 0.04 |
| | Flutter Speed (m/s) | 178.55 | 175.45 | 1.73 |

Unlike the 30 m/s optimization scenario, where the flutter constraint was not present for any of the 9000 possible combinations analyzed by NSGA-II, for a cruise speed of 60 m/s the situation changed. This scenario, which include 9000 combinations as well, had the flutter constraint active for 0.322% of the time. The associated design variables remained predominantly within a specific range, as shown in Table 5.8.

Table 5.8: Range of design variables within which the flutter constraint was active at 60 m/s.

| Design Variable | Minimum Value | Maximum Value | Units |
|---|---|---|---|
| Aspect-ratio (AR) | 15.2 | 15.4 | - |
| Sweep angle ($\Lambda$) | 10.4 | 11.3 | deg |
| Torsional stiffness (GJ) | $1.62 \times 10^6$ | $1.70 \times 10^6$ | N.m$^2$ |
| Angle of attack ($\alpha$) | 6.86 | 6.95 | deg |

However, the algorithm identified a Pareto front comprising 180 optimal solutions, in which the flutter phenomenon remained perpetually inactive. In fact, the results, in terms of design variables, differ from the previous design space reported in Table 5.8. The optimization adequately bypassed problematic regions within the design space, identifying and validating optimal solutions where flutter did not materialize.

Moving forward, an extended analysis was undertaken, modifying the optimization parameters as done for 30 m/s. The purpose of this computational effort was to see if it was possible to reduce the error rate below the 5% threshold for all results.

Table 5.9: Comparative analysis between optimized solutions using the new parameters and SHARPy at 60 m/s.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 36.08 | 36.16 | 0.21 |
| | Mass (kg) | 927.40 | 927.06 | 0.04 |
| | Flutter Speed (m/s) | 91.94 | 92.23 | 0.32 |
| min Mass | CL/CD | 21.28 | 21.56 | 1.32 |
| | Mass (kg) | 392.62 | 392.32 | 0.08 |
| | Flutter Speed (m/s) | 194.92 | 188.92 | 3.08 |

Results show in Table 5.9 clearly indicate that delving deeper into the design space produced significant improvements, with all the percentage errors lower than 5%.

However, it is critical to emphasize the balance between computational resources and solution quality. The increase in population and generations led to better results, but also to a 490.66% increase in computational time compared to the first optimization. Nevertheless, the potential gains in accuracy seem to justify the increase in computational time. In Figure 5.2 both the optimization Pareto fronts are reported, where it is noticeable an improvement in the Pareto front for higher values of CL/CD.

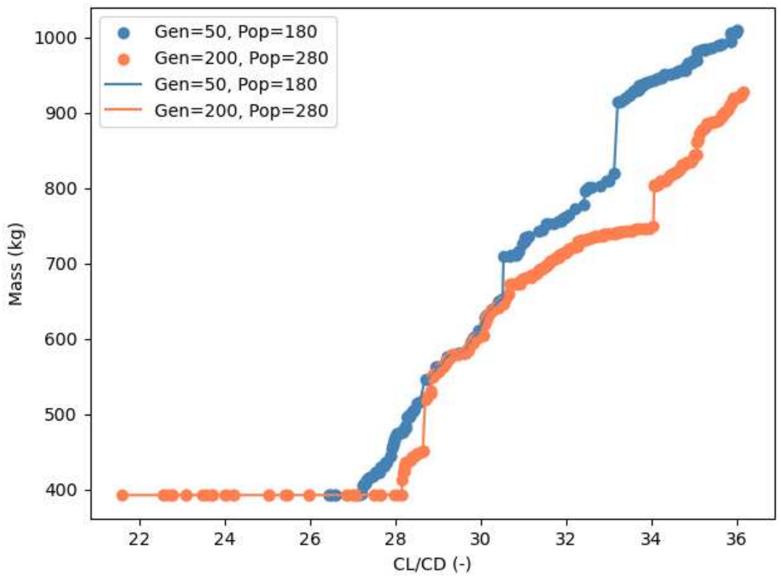


Figure 5.2: Pareto fronts at 60 m/s.

For the high CL/CD values, the orange set is distinctly better than the blue one. This means that with extended optimization (Generation = 200, Population size = 280), the solutions achieved a higher aerodynamic efficiency with a smaller increase in mass compared to the earlier iteration.

In the middle section of the graph, instead, there is a significant overlap between the two datasets. This overlap suggests that for certain design choices, increasing the generations and population doesn't significantly change the outcomes.

## 5.3 Optimization Results at 130 m/s Cruise Speed

Given the low percentage of solutions with active flutter at 60 m/s, an additional analysis was conducted at 130 m/s to explore more complex aeroelastic behaviors.

This scenario was also evaluated using the methodology described in Chapter 4. The associated metrics of the *Extra Trees Regressor* models, used to make the surrogate-based optimizazion, are reported in Table 5.10.

Table 5.10: Final RMSE and $R^2$ metrics for each surrogate model at 130 m/s.

| Model | $RMSE$ | $R^2$ |
|---|---|---|
| Flutter | 7.981 | 0.907 |
| Mass | 0.263 | 0.999 |
| CD | 0.0017 | 0.997 |
| CL | 0.011 | 0.999 |

The design space, after a first analysis, was expanded as well. Despite this approach, even with the new dataset of 2500 data, the percentage errors remained very high compared to SHARPy. Moreover, the values were very far from reality, particularly in terms of CL/CD and flutter speed. For this reason these results are not reported.

Analyzing in detail the optimization, it was found that in contrast to the 60 m/s scenarios of the optimization, the 130 m/s scenarios—encompassing 9000 combinations—showed the flutter constraint as active in 7452 cases.

The robust occurrence of flutter, evident in 82.2% of the design possibilities, imposed a significant barrier in identifying optimal solutions that also adhered to the flutter constraint. In the Pareto front constituting 180 optimal solutions, the 24.4% were marred by active flutter, casting an anomalous appearance and subsequently skewing the percentual error during the initial comparison against benchmark models.

Given the significant presence of flutter and the resulting obstacle to obtaining valid optimal solutions, the optimization approach was reevaluated by increasing both the population size and the number of generations, as done for the previous analyses. Specifically, the population was increased from 180 to 280 and the generations were increased from 50 to 200. The attempt was to test whether this intensification, by exploring a larger solution space, would lead to more accurate solutions or greater convergence to the global optimum.

This approach produced a substantial improvement, with only 7.5% of the 280 solutions affected by flutter, compared with the previous 24.4%. However, this came at a heavy computational price, with a time of *2153.31 seconds* and an increase of the 690% compared to the first optimization. Furthermore, the percentage errors were still very high.

These results are presented in Table 5.11, where SHARPy also includes the correction made by flate plate theory. Also for this cruise speed, since the Mach number is 0.391, the effect of compressibility was neglected.

Table 5.11: Comparative analysis between optimized solutions using the new parameters and SHARPy results at 130 m/s.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 21.08 | 14.15 | 32.87 |
| | Mass (kg) | 960.72 | 961.60 | 0.09 |
| | Flutter Speed (m/s) | 76.18 | 92.27 | 21.12 |
| min Mass | CL/CD | 14.93 | 9.99 | 33.09 |
| | Mass (kg) | 393.27 | 393.01 | 0.03 |
| | Flutter Speed (m/s) | 169.41 | 144.13 | 14.92 |

As can be observed, the percentage errors are still very large when compared to the analyses performed at 30 and 60 m/s, and several findings, such a negative CL/CD ratio, would suggest that the optimization in this case is not appropriate to represent the situation.

Actually, this was because penalties had an significant impact on the outcomes. This aspect will be explored in depth by carrying out a detailed analysis of only the subset of solutions for which the flutter constraint, and consequently the penalty values, were not present. The Pareto fronts produced by the two different optimizations accomplished so far are shown in Figure 5.3.

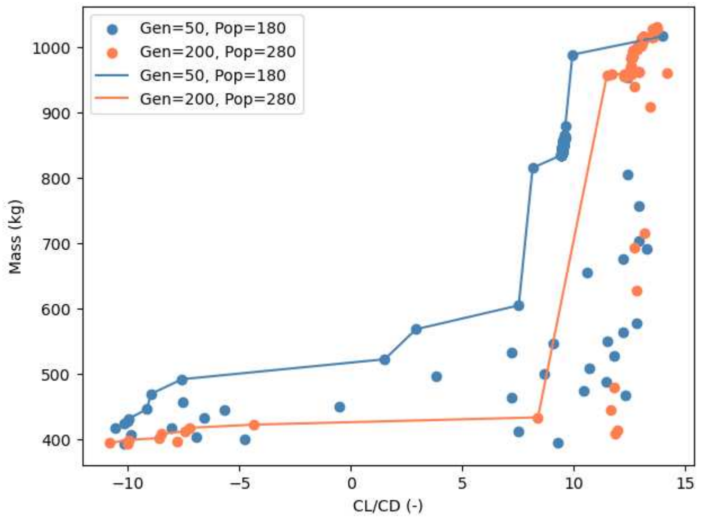


Figure 5.3: Pareto fronts at 130 m/s.

It is evident that several sites along the Pareto fronts display negative CL/CD values. These are the unfeasible solutions brought on by the active flutter issue and associated penalties attached to the outcomes. The solutions of a rank different from 1, which classify them as suboptimal in relation to the main Pareto frontier, are indicated by the points presented without a continuous line.

As can be observed, the quantity of these points is substantially higher in the optimization scenario that employs fewer generations and smaller population sizes.

Following, a second analysis including only the Pareto results not affected by flutter is carried out to demonstrate what was said previously regarding penalties. The new Pareto are shown in Figure 5.4.

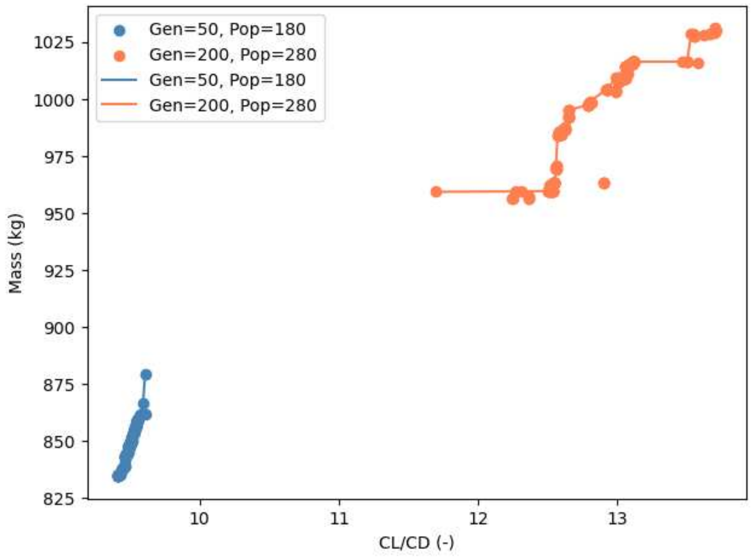


Figure 5.4: Pareto fronts at 130 m/s with only the solutions without the flutter constraint active.

Two distinct groups of solutions can be immediately identified from the graph. In both cases, the optimized design space is significantly reduced, but the second offers a more diverse set of solutions. This implies that with more generations and a larger population, optimization is able to explore a wider solution space and potentially find more optimal solutions.

After isolating the study on these no-flutter solutions, the two extreme were identified: both the one with the greatest CL/CD and the least mass. The motive behind this selection was to compare them with the SHARPy results, as done for the complete Pareto analysis.

Tables 5.12 and 5.13 encapsulate the quantitative comparative analysis, juxtaposing SHARPy results against optimization outputs for the two distinct optimization runs.

Notably, while the broader Pareto front exhibited substantial percentage errors, this refined analysis showcases good accuracy. The errors for the 200-generation, 280-population optimization were consistently less than 5%. The results, even in the initial, less exhaustive optimization, can be considered accurate because the relative errors are less than 10%.

Table 5.12: Comparative analysis: only no-flutter solutions with 50 generation and 180 population.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 9.24 | 9.61 | 4.04 |
| | Mass (kg) | 862.38 | 862.04 | 0.04 |
| | Flutter Speed (m/s) | 184.21 | 195.06 | 8.09 |
| min Mass | CL/CD | 8.99 | 9.41 | 3.99 |
| | Mass (kg) | 834.56 | 834.64 | 0.01 |
| | Flutter Speed (m/s) | 187.82 | 195.97 | 4.24 |

Table 5.13: Comparative analysis: only no-flutter solutions with 200 generation and 280 population.

| Objective | Metric | SHARPy | Optimization | Percentage Error (%) |
|---|---|---|---|---|
| max CL/CD | CL/CD | 13.12 | 13.71 | 4.47 |
| | Mass (kg) | 1030.99 | 1030.06 | 0.03 |
| | Flutter Speed (m/s) | 189.21 | 194.87 | 2.96 |
| min Mass | CL/CD | 11.99 | 12.37 | 3.14 |
| | Mass (kg) | 955.82 | 956.41 | 0.06 |
| | Flutter Speed (m/s) | 188.82 | 195.03 | 3.29 |

These results, in line with those obtained for the cruising speeds previously examined, clearly reveal that the high penalties were the main cause of the high percentage errors previously observed in Table 5.11. Therefore, considering the division between the optimization prediction and the penalty values added when the flutter constraint is active, we obtain results that are perfectly in line with the previous analyses.

This further strengthens the validity of the optimization approach and that, despite the obstacles represented by penalties, the algorithm is able to offer excellent quality solutions.

# Conclusions and Future Work

## 6.1 Conclusions

The issue of integrating aeroelastic analysis in the context of MDO was addressed in this thesis. The computational challenges inherent to this integration prompted the exploration of SMs, providing a pathway for computational efficiency.

The Goland wing was chosen, taking advantage of the availability of experimental data that facilitate the validation of the model and its consolidated implementation within the SHARPy tool. The dataset was constructed using the LHS, selecting aspect ratio, sweep angle, torsional stiffness, and angle of attack as key design variables. A rigorous evaluation process led to the adoption of the Extra Trees Regressor as the surrogate model of choice. This decision was driven by its performance metrics and information gleaned from its learning curves.

The surrogate models had noteworthy prediction ability at the fundamental cruise speed of 30 m/s, as shown by RMSE and $R^2$ values for mass and lift coefficient reported in Table 4.11. However, certain variances were observed, particularly in drag coefficient predictions and flutter speed. These discrepancies are potentially attributable to inherent nonlinearities in aeroelasticity in specific regions of the design space, where some aerodynamic and structural dynamic interactions become more complex. This hypothesis is consistent with trends observed in the data for high and low CD values. For the flutter model, the high complexity in estimating flutter speed using an iterative algorithm, the p-k method, and the overfitting showed by its learning curve, led to less accurate performance. As cruise speeds were increased to 60 m/s and 130 m/s, the complexities of flutter became even more pronounced.

Despite these challenges, within the scope of this thesis, the results obtained for all the SMs, including flutter and CD model, were considered acceptable. While an error of up to 5% was initially targeted for the testing of the SMs, predictions exhibiting a percentage error slightly higher than this but always below 10% were also deemed acceptable.

Moreover, the adjustments to the design space after a first optimization, explained in Section 4.3.3, contributed significantly to enhancing accuracy. In fact, it is possible to observe an improvement in the metrics for the new SMs at 30 m/s, showed in Table 5.3.

A surrogate-based optimization strategy was then developed, leveraging open-source Python codes. The overarching goal was to harmonize performance with aeroelasticity metrics. The multi-objective optimization was specifically designed to maximize CL/CD and minimize mass, keeping in view the flutter constraint across three distinct cruise speeds. The Non-dominated Sorting Genetic Algorithm II (NSGA-II) as implemented in Pygmo was used to further explore the design space, considering the multi-objective nature of the problem and the cheap run time of the surrogate model.

The optimization process was employed at different cruising speeds. For the initial 30 m/s cruise speed, the optimization was backed by the associated SMs, elucidating the most efficient design solutions. While the results produced were initially off in certain metrics, notably in flutter speed, the revision of the design space and expansion of the design variable boundaries resulted in substantial enhancements. This is evident through the decreased percentage errors when juxtaposed against the SHARPy results.

Nevertheless, a persistent deviation in flutter speed indicates potential areas for further investigation and refinement. An extended analysis was then conducted using the augmented dataset, wherein the optimization parameters were also increased. This advanced analysis ensured that the percentage differences between the optimization outcomes and SHARPy were brought under the 5% threshold.

The extended configuration of the optimization algorithm, with a generation of 200 and a population size of 280, seems to provide a more comprehensive exploration of the trade-offs between the two objectives, especially at higher CL/CD values.

Furthermore, the inactivity of the flutter constraint during optimization alludes to the importance of analyzing the behavior of the Goland wing at increased cruise speeds.

At a cruise speed of 60 m/s, the Extra Trees Regressor model stood out for its predictive accuracy. The optimization, carried out using the NSGA-II algorithm, remained consistent with the 30 m/s scenario parameters to ensure comparative coherence. Adjustments to the design space were made to address significant percent errors, akin to the approach at 30 m/s. The optimized solutions were in close agreement with SHARPy results, especially post adjustments. Notably, the flutter constraint was active 0.322% of the time, a deviation from the 30 m/s scenario. An extended optimization effort successfully reduced all relative errors below 5%.

The provided Pareto fronts illustrated the effectiveness of using a more intensive optimization process, with a greater number of generations and population sizes. Better design options have been made possible by the evolving findings, particularly for those striving for high aerodynamic efficiency. This, however, resulted in longer computation times.

Optimization at the cruise speed of 130 m/s revealed more complex aeroelastic behaviors than the 60 m/s analysis. For the new cruise speed the percentage of such events was markedly higher than those observed at 60 m/s, leading to a less smooth navigation of the landscape of possible solutions. Flutter played a dominant role in constraining solutions, affecting 82.2% of design possibilities. This shifted optimal solutions, with 24.4% initially showing active flutter.

Refining the optimization by increasing both population size and generations showed improvements. This intensified exploration somewhat overcame the challenges posed by flutter, reducing affected solutions from 24.4% to 7.5%. However, these gains in accuracy and flutter reduction came at a significant computational cost, with a 690% increase in computational time.

Examining only the solutions without flutter constraints, the optimization was particularly accurate, especially in the intensified optimization scenarios. The error margins were in line with those of the previous velocity analyses. It was thus evident that the significant errors initially seen in the full Pareto front stemmed from the penalties associated with activating the flutter constraint. Indeed, the visualizations provided had shown numerous solutions that were not feasible and many others that were suboptimal due to ranks different from 1.

Upon examination of the Pareto fronts for the solutions not affected by the flutter, it was clear that the possibilities for optimal design were still significantly limited for this cruising speed. This suggests that the flutter constraint may be overly restrictive for the design space initially considered. Despite this, the optimization was found to be accurate in predictive terms even in this scenario.

A final comparison is presented in Table 6.1, detailing the results obtained using consistent parameters and dataset sizes across the three analysed cruise speeds.

Table 6.1: Optimization results across analyzed cruise speeds.

| $V_c$ (m/s) | Objective | CL/CD (-) | Mass (kg) | AR (-) | $\Lambda$ (deg) | GJ (N.m$^2$) | $\alpha$ (deg) | $V_{\text{flutter}}$ (m/s) |
|---|---|---|---|---|---|---|---|---|
| 30 | 1 | 39.79 | 1043.46 | 15.97 | -2.10 | 1.54e+06 | 4.27 | 127.89 |
| | 2 | 27.13 | 392.17 | 6.00 | 28.7 | 1.36e+06 | 5.13 | 188.10 |
| 60 | 1 | 36.16 | 927.06 | 14.20 | 0.72 | 1.61e+06 | 6.14 | 92.23 |
| | 2 | 21.56 | 392.32 | 6.00 | 32.92 | 7.38e+05 | 7.03 | 188.92 |
| 130 | 1 | 13.71 | 1030.06 | 15.78 | 6.66 | 9.37e+05 | 9.12 | 194.87 |
| | 2 | 12.37 | 956.41 | 14.65 | 7.24 | 8.80e+05 | 10.39 | 195.03 |

The optimization parameters were set at: population size = 280, number of generations = 200, mutation rate = 0.5, and crossover rate = 0.8. The results in terms of CL/CD, mass and flutter speed ($V_{\text{flutter}}$) associated respectively with the maximum value of CL/CD, the objective 1, and the minimum mass value, objective 2, are reported. The corresponding design variables are also presented: aspect ratio (AR), sweep angle ($\Lambda$), torsional stiffness (GJ), and angle of attack ($\alpha$). For the last analysis, at 130 m/s, the results considered are only the ones without the flutter active.

In Figure 6.1, 6.2, 6.3 the simplified representations of the optimized wing designs are shown. For each cruise speed analysed, two configurations are shown: the one that maximizes CL/CD and the one that minimizes mass.

The first thing that can be observed is that, for all the speeds analysed, by improving one parameter the other worsens accordingly, according to the nature of multi-objective optimization. High values of CL/CD consequently correspond to higher mass values.

It is possible to observe that the aspect ratio (AR) is consistently higher when maximizing the CL/CD compared to minimizing the mass, across all cruising velocities. This observation aligns with established aerodynamic principles: wings with a higher AR have longer

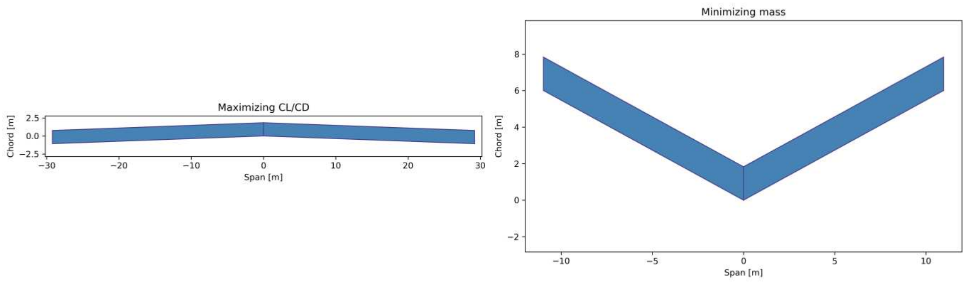


Figure 6.1: Optimized wing design at 30 m/s.

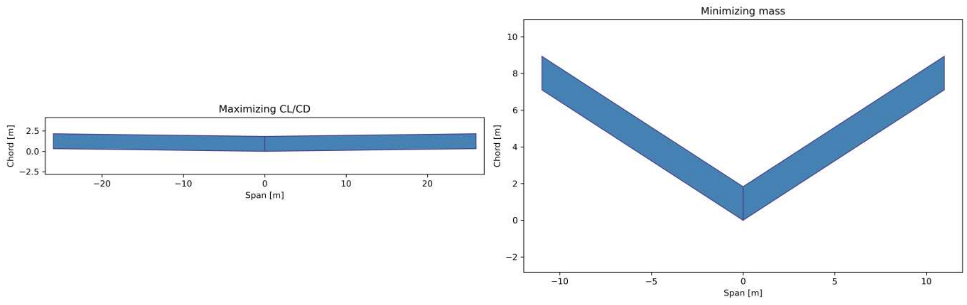


Figure 6.2: Optimized wing design at 60 m/s.

spans relative to their chord, which reduces induced drag and consequently improves the lift-to-drag ratio. However, the flip side to this advantage is that longer, slender wings tend to be more flexible. As a result, they are more susceptible to aeroelastic instabilities, notably flutter. This is evident in the data as well: as the AR increases for better CL/CD, the flutter speed ($V_{flutter}$) tends to decrease, indicating a reduced stability margin against flutter. This trade-off underlines the inherent challenges in aircraft design, where the quest for aerodynamic efficiency often comes at the cost of structural and aeroelastic considerations.

Moreover, as the cruising speed increases, the flutter speed ($V_{flutter}$) decreases, indicating that flutter tends to manifest at lower speeds relative to the cruising speed. An increase in cruising speed results in a corresponding rise in dynamic pressure acting on the aircraft's structure. This augmented dynamic pressure amplifies the aerodynamic forces, which can lead to intensified interactions between the structure's inertial, elastic, and aerodynamic forces. As a result, the structure, especially the wings, becomes more susceptible to aeroelastic instabilities such as flutter. The data support this understanding. This highlights the importance of careful aeroelastic considerations, especially for aircraft designed to operate at higher speeds.

At lower cruise speeds of 30 m/s and 60 m/s, when the primary objective is to minimize mass, a positive and more pronounced sweep angle is observed. This might be associated with a structural benefit, redistributing the lift towards the root, which can result in a lighter and more efficient wing structure. However, when maximizing the CL/CD, a lower sweep may favor aerodynamic efficiency at these speeds.

As the cruise speed increases to 130 m/s, approaching transonic speeds, sweep becomes

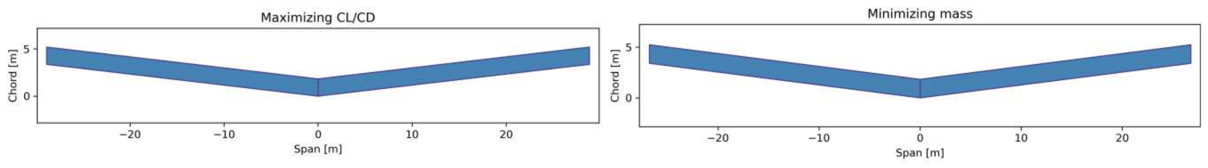


Figure 6.3: Optimized wing design at 130 m/s.

crucial to delay the onset of transonic drag rise, and this is evident in the positive sweep angles seen for both objectives.

Variations in torsional stiffness and angle of attack across the different objectives and velocities are subtler compared to other parameters. However, it's noteworthy that higher CL/CD values are associated with lower angles of attack. This trend aligns with the expectation that a reduced angle of attack would lead to decreased aerodynamic drag. Lowering the angle of attack, while still achieving adequate lift, is a strategy to enhance the aerodynamic efficiency of the wing.

Another important observation about the aspect ratio is that for both 30 m/s and 60 m/s cruising speeds, the optimal solutions approach the boundaries of the defined design space, which was set between 6 and 16. This is significant as it indicates that the design space might not fully encompass the optimal regions for these objectives at these speeds. The solutions gravitating towards the edge of the design space suggest that there may exist more optimal designs just beyond the predefined limits. This lays a foundation for considering an expansion of the design space boundaries in future investigations.

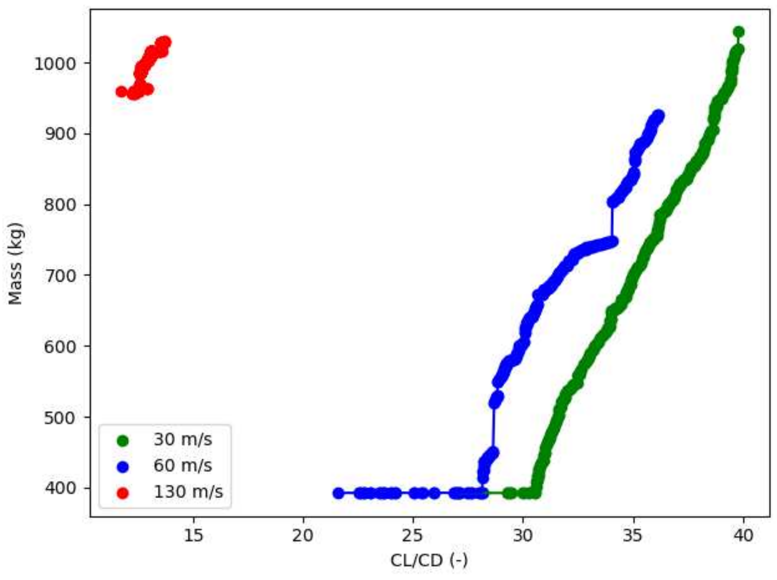


Figure 6.4: Pareto fronts across analysed cruise speeds.

The plot presented in Figure 6.4 delineates the Pareto fronts for the three analysed cruise speeds: 30 m/s, 60 m/s, and 130 m/s, derived using the same optimization parameters.

It's important to note that the results at 130 m/s represent only the feasible solutions.

As the cruise speed augments, the Pareto front tends to shift towards lower CL/CD values. This could be attributed to the increased aerodynamic and structural challenges at higher speeds.

The results at 130 m/s present an evident distinction. Unlike the relatively distributed Pareto fronts at 30 m/s and 60 m/s, the solutions at 130 m/s are remarkably clustered. This localized concentration indicates that, at this higher cruising speed, design solutions that avoid flutter are limited, especially within the constraints of our design space.

Moreover, these solutions at 130 m/s are characterized by relatively higher masses and moderate CL/CD values. This could imply that, to avoid flutter at this speed, the structures would have to be heavier or more robust, potentially compromising aerodynamic efficiency.

Pareto fronts for 30 m/s and 60 m/s show a wider range of trade-offs between mass and CL/CD, indicating a greater variety of feasible design solutions at these speeds.

In essence, the graph accentuates the intricate challenges faced in aerostructural design. Especially at higher speeds, such as 130 m/s, the overly stringent constraint of flutter within our design space narrows the optimal solutions, emphasizing the tension between aerodynamic performance and structural robustness.

## 6.2 Computational Cost Analysis

In this section, we compare the computational efficiency of direct simulations using SHARPy and predictions made with our surrogate models. All the computational results discussed in this section were obtained using a specific workstation, the details of which are provided in Table 4.5.

A direct simulation with SHARPy takes approximately 93.49 seconds per run. In contrast, our surrogate models predict the outcomes in less than a hundredth of a second.

However, the real advantage becomes evident when considering optimization procedures. In Figure 6.5 the computational time for each optimization made is reported:

- An optimization run with 50 generations and a population size of 180 takes an average of 321.76 seconds. This implies a total of $50 \times 180 = 9000$ evaluations.
- Another optimization scenario with 200 generations and a population size of 280 takes about 1825.03 seconds, equivalent to $200 \times 280 = 56000$ evaluations.

For the first set of optimizations conducted with SHARPy, the computational time required would have been approximately $9000 \times 93.49$ seconds $= 841410$ seconds, equivalent to nearly 10 days.

Using direct SHARPy simulations for the subsequent scenario would have necessitated around $56000 \times 93.49$ seconds $= 5235440$ seconds, which amounts to roughly 60 days.

The data underscore that the adoption of SMs provides a considerable reduction in computational time, especially in optimization contexts where multiple evaluations are required.

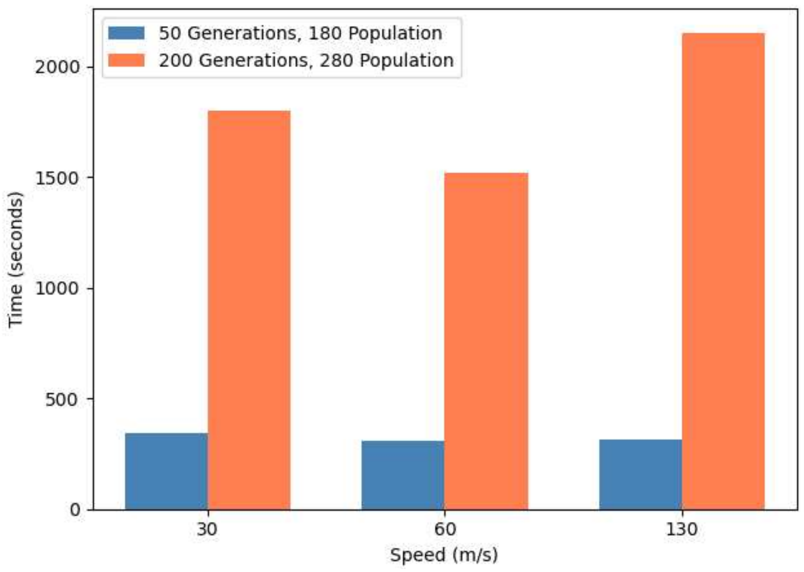


Figure 6.5: Computational costs for each optimization scenario.

## 6.3 Future Work

The work presented in this thesis has opened several avenues for future research:

- Increasing the dataset: Learning curves related to flutter velocity indicate potential for further refinement. Unlike metrics such as mass, CD and CL, where learning curves stabilize indicating optimal generalization of the model, the flutter model appears to suffer from overfitting. This is evidenced by the disparity between the training and validation scores and the absence of a plateau in the validation score curve. Expanding the dataset in this domain, especially in critical regions, could improve the performance and reliability of the surrogate model.
- Adaptive sampling during optimization: One of the main findings of this thesis is the efficacy of surrogate models in substantially reducing computational time. Building on this, an interesting approach would be the implementation of adaptive sampling techniques during the optimization process to enhance the accuracy and robustness of the surrogate model online. Such an approach, where the surrogate model evolves and improves as the optimization progresses, can ensure better convergence and solution quality.
- High-Fidelity RANS simulations: For better accuracy in the aerodynamic objectives, a multi-fidelity approach integrating High-Fidelity RANS simulations can be explored.
- Analysis across a spectrum of cruise speeds: This study was based primarily on a few fixed cruise speeds. Expanding this analysis over a wider range of cruise speeds

will not only provide a richer dataset, but also pave the way for the integration of cruise speed as a design variable. Such a comprehensive analysis can provide a more holistic view of the aeroelastic performance landscape.

- Further refinement of the design space: An interesting observation made during the analysis is that the optimal solutions for cruising speeds of 30 m/s and 60 m/s gravitate toward the boundary of the predefined design space, particularly with regard to aspect ratio. This suggests that the current design space may not fully capture the optimal regions for these objectives. It becomes imperative to consider expansion and refinement of design space boundaries in subsequent studies to ascertain and exploit more optimal design configurations.
- Exploration of alternative wing models: The Goland wing, while useful for many preliminary studies, provides a simplified representation of real-world wing structures. The choice of wing model can significantly influence the results of an optimization analysis. Although the Goland wing model has been effectively used for the purposes of this thesis, exploration of alternative and potentially more realistic wing models could shed light on new insights and produce better performance metrics.

Incorporating these strategies can strengthen the robustness and accuracy of the optimization framework based on the surrogate model, ensuring both computational efficiency and reliability of design results.

# Bibliography

[1] J. Wright and J. Cooper, *Introduction to Aircraft Aeroelasticity and Loads.* Aerospace Series, Wiley, 2015.

[2] S. Sharma and V. Chahar, "A Comprehensive Review on Multi-objective Optimization Techniques: Past, Present and Future," *Archives of Computational Methods in Engineering*, vol. 29, p. 3, 07 2022.

[3] F. Bre and V. Fachinotti, "A computational multi-objective optimization method to improve energy efficiency and thermal comfort in dwellings," *Energy and Buildings*, vol. 154, pp. 283–294, 2017.

[4] M. G. Fernández-Godino, C. Park, N.-H. Kim, and R. T. Haftka, "Review of multi-fidelity models," *arXiv preprint arXiv:1609.07196*, 2016.

[5] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[6] P. V. Thomas, M. S. ElSayed, and D. Walch, "Review of Model Order Reduction Methods and Their Applications in Aeroelasticity Loads Analysis for Design Optimization of Complex Airframes," *Journal of Aerospace Engineering*, vol. 32, no. 2, p. 04018156, 2019.

[7] S. Koziel, D. E. Ciaurri, and L. Leifsson, "Surrogate-based methods," in *Computational Optimization, Methods and Algorithms*, pp. 33–59, Springer, 2011.

[8] M. Camana, S. Ahmed, C. García, and I. Koo, "Extremely Randomized Trees-Based Scheme for Stealthy Cyber-Attack Detection in Smart Grid Networks," *IEEE Access*, vol. 8, pp. 19921–19933, 2020.

[9] J. Murua, R. Palacios, and J. Graham, "Assessment of Wake-Tail Interference Effects on the Dynamics of Flexible Aircraft," *AIAA Journal*, vol. 50, pp. 1575–1585, 07 2012.

[10] S. Verma, M. Pant, and V. Snasel, "A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems," *IEEE Access*, vol. 9, pp. 57757–57791, 2021.

[11] F. Afonso, J. Vale, Éder Oliveira, F. Lau, and A. Suleman, "A review on non-linear aeroelasticity of high aspect-ratio wings," *Progress in Aerospace Sciences*, vol. 89, pp. 40–57, 2017.

[12] E. Jonsson, C. Riso, C. A. Lupp, C. E. Cesnik, J. R. Martins, and B. I. Epureanu, "Flutter and post-flutter constraints in aircraft design optimization," *Progress in Aerospace Sciences*, 2019.

[13] I. Sadrehaghighi, "Aircraft Multi-Disciplinary Optimization (MDO)," 2022. Series CFD Open.

[14] J. R. Martins and A. B. Lambe, "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA journal*, vol. 51, no. 9, pp. 2049–2075, 2013.

[15] L. Brevault, M. Balesdent, N. Bérend, and R. Le Riche, "Decoupled Multidisciplinary Design Optimization Formulation for Interdisciplinary Coupling Satisfaction Under Uncertainty," *AIAA Journal*, vol. 54, no. 1, pp. 186–205, 2016.

[16] J. R. R. A. Martins, *A coupled-adjoint method for high-fidelity aero-structural optimization.* PhD thesis, Stanford University, 2003.

[17] D. E. Kvasov and M. S. Mukhametzhanov, "Metaheuristic vs. deterministic global optimization algorithms: The univariate case," *Applied Mathematics and Computation*, vol. 318, pp. 245–259, 2018.

[18] J. O. Agushaka and A. E. Ezugwu, "Initialisation Approaches for Population-Based Metaheuristic Algorithms: A Comprehensive Review," *Applied Sciences*, vol. 12, no. 2, p. 896, 2022.

[19] S. Bandyopadhyay and S. Saha, *Some Single- and Multiobjective Optimization Techniques*, pp. 17–58. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[20] M. Elyasi and A. Roudbari, "Multi-objective robust design optimization (mordo) of an aeroelastic high-aspect-ratio wing," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, p. 560, 10 2020.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization," *Siam Review*, vol. 60, no. 3, pp. 550–591, 2018.

[23] P. Beran, D. Bryson, A. Thelen, M. Diez, A. Serani, and L. Mainini, "Framework for Comparison of Multi-Fidelity Approaches for Military Vehicle Design," in *AVT-354 Research Workshop on Multi-Fidelity Methods for Military Vehicle Design*, (Varna, Bulgaria), September 2022.

[24] C. Riso and C. E. S. Cesnik, "Impact of Low-Order Modeling on Aeroelastic Predictions for Very Flexible Wings," *Journal of Aircraft*, vol. 60, no. 3, pp. 662–687, 2023.

[25] A. C. Gray, C. Riso, E. Jonsson, J. R. R. A. Martins, and C. E. S. Cesnik, "High-Fidelity Aerostructural Optimization with a Geometrically Nonlinear Flutter Constraint," *AIAA Journal*, vol. 61, no. 6, pp. 2430–2443, 2023.

[26] E. Jonsson, C. Riso, B. B. Monteiro, A. C. Gray, J. R. R. A. Martins, and C. E. S. Cesnik, "High-Fidelity Gradient-Based Wing Structural Optimization Including Geometrically Nonlinear Flutter Constraint," *AIAA Journal*, vol. 61, no. 7, pp. 3045–3061, 2023.

[27] D. Bryson, M. Rumpfkeil, and R. Durscher, "Framework for Multifidelity Aeroelastic Vehicle Design Optimization," in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, (Denver, Colorado), 2017.

[28] A. S. Thelen, D. E. Bryson, B. K. Stanford, and P. S. Beran, "Multi-Fidelity Gradient-Based Optimization for High-Dimensional Aeroelastic Configurations," *Algorithms*, vol. 15, no. 4, p. 131, 2022.

[29] G. Singh and R. V. Grandhi, “Mixed-Variable Optimization Strategy Employing Multi-fidelity Simulation and Surrogate Models,” *AIAA journal*, vol. 48, no. 1, pp. 215–223, 2010.

[30] P. Kumar, K. Sinha, N. K. Nere, Y. Shin, R. Ho, L. B. Mlinar, and A. Y. Sheikh, “A machine learning framework for computationally expensive transient models,” *Scientific reports*, vol. 10, p. 11492, 2020.

[31] J. Li, X. Du, and J. R. Martins, “Machine learning in aerodynamic shape optimization,” *Progress in Aerospace Sciences*, vol. 134, p. 100849, 2022.

[32] A. I. Forrester, A. Sóbester, and A. J. Keane, “Multi-fidelity optimization via surrogate modelling,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.

[33] K. R. Brouwer and J. J. McNamara, “Surrogate-based aeroelastic loads prediction in the presence of shock-induced separation,” *Journal of Fluids and Structures*, vol. 93, p. 102838, 2020.

[34] A. Cea and R. Palacios, “Geometrically Nonlinear Effects on the Aeroelastic Response of a Transport Aircraft Configuration,” *Journal of Aircraft*, vol. 60, no. 1, pp. 205–220, 2023.

[35] M. Sohst, J. Lobo do Vale, F. Afonso, and A. Suleman, “Optimization and comparison of strut-braced and high aspect ratio wing aircraft configurations including flutter analysis with geometric non-linearities,” *Aerospace Science and Technology*, vol. 124, p. 107531, 2022.

[36] F. Toffol and S. Ricci, “Preliminary Aero-Elastic Optimization of a Twin-Aisle Long-Haul Aircraft with Increased Aspect Ratio,” *Aerospace*, vol. 10, no. 4, p. 374, 2023.

[37] S. L. Brunton and J. N. Kutz, *Reduced Order Models*, p. 373–374. Cambridge University Press, 2019.

[38] G. Mendonça, F. Afonso, and F. Lau, “Model order reduction in aerodynamics: Review and applications,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 15, pp. 5816–5836, 2019.

[39] D. J. Lucia, P. S. Beran, and W. A. Silva, “Reduced-order modeling: new approaches for computational physics,” *Progress in Aerospace Sciences*, vol. 40, no. 1, pp. 51–117, 2004.

[40] T. Lieu, C. Farhat, and M. Lesoinne, “Reduced-order fluid/structure modeling of a complete aircraft configuration,” *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 41-43, pp. 5730–5742, 2006.

[41] O. Stodieck, J. Cooper, S. Neild, M. Lowenberg, and L. Iorga, “Slender-Wing Beam Reduction Method for Gradient-Based Aeroelastic Design Optimization,” *AIAA Journal*, vol. 56, no. 11, pp. 4529–4545, 2018.

[42] F. A. C. Viana, C. Gogu, and T. Goel, “Surrogate modeling: tricks that endured the test of time and some recent developments,” *Structural and Multidisciplinary Optimization*, vol. 64, pp. 2881 – 2908, 2021.

[43] A. A. Giunta, S. F. Wojtkiewicz, and M. Eldred, “Overview of Modern Design of Experiments Methods for Computational Simulations,” in *41st Aerospace Sciences Meeting and Exhibit*, (Reno, Nevada, USA), January 2003.

[44] A. Hedayat, N. Sloane, and J. Stufken, *Orthogonal Arrays: Theory and Applications.* Springer Series in Statistics, Springer New York, 1999.

[45] M. D. Mckay, R. J. Beckman, and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[46] A. Gelman and J. L. Hill, *3 - Linear regression: the basics.* 2012.

[47] G. Hutcheson, *Ordinary Least-Squares Regression.* SAGE Publications, Ltd., 1999.

[48] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, pp. 267–288, 1996.

[49] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[50] T. J. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning.* NY, USA: Springer New York, 2009.

[51] T. K. Ho, "A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors," *Pattern Analysis & Applications*, vol. 5, pp. 102–112, 2002.

[52] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, 2006.

[53] V. Vapnik, *The Nature of Statistical Learning Theory.* Statistics for Engineering and Information Science, New York, NY, USA: Springer, 2000.

[54] A. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[55] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," in *NIPS*, 2007.

[56] Y. Xu and R. Goodacre, "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning," *Journal of Analysis and Testing*, vol. 2, 10 2018.

[57] N. Amor, M. Noman, and M. Petru, "Prediction of functional properties of nano TiO2 coated cotton composites by artificial neural network," *Scientific Reports*, vol. 11, p. 12235, 06 2021.

[58] A. Kassambara, *Machine Learning Essentials: Practical Guide in R.* STHDA, 2018.

[59] B. Panda, "Hyperparameter tuning," 10 2019.

[60] S. Pothuganti, "Review on over-fitting and under-fitting problems in Machine Learning and solutions," *International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering*, vol. 7, pp. 3692–3695, 09 2018.

[61] M. Goland, "The Flutter of a Uniorm Cantilever Wing," *Journal of Applied Mechanics*, vol. 12, no. 4, pp. 197–208, 1945.

[62] A. Carre, A. Muñoz, N. Goizueta, and R. Palacios, "SHARPy: A dynamic aeroelastic simulation toolbox for very flexible aircraft and wind turbines," *The Journal of Open Source Software*, vol. 4, p. 1885, 12 2019.

[63] M. J. Patil, D. H. Hodges, and C. E. S. Cesnik, "Nonlinear Aeroelastic Analysis of Complete Aircraft in Subsonic Flow," *Journal of Aircraft*, vol. 37, pp. 753–760, 2000.

[64] W. Zhicun, P. Chen, D. Liu, D. Mook, and M. Patil, "Time Domain Nonlinear Aeroelastic Analysis for HALE Wings," in *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, (Newport, Rhode Island, USA), May 2006.

[65] J. Katz and A. Plotkin, *Low-Speed Aerodynamics.* Cambridge Aerospace Series, Cambridge University Press.

[66] R. J. S. Simpson, R. Palacios, and J. Murua, "Induced-drag calculations in the unsteady vortex lattice method," *AIAA Journal*, vol. 51, no. 7, pp. 1775–1779, 2013.

[67] M. Géradin and A. Cardona, *Flexible Multibody Dynamics: A Finite Element Approach.* John Wiley, 2001.

[68] R. Palacios, J. Murua, and R. Cook, "Structural and Aerodynamic Models in Nonlinear Flight Dynamics of Very Flexible Aircraft," *AIAA Journal*, vol. 48, no. 11, pp. 2648–2659, 2010.

[69] R. Palacios and C. E. S. Cesnik, "Cross-Sectional Analysis of Nonhomogeneous Anisotropic Active Slender Structures," *AIAA Journal*, vol. 43, no. 12, pp. 2624–2638, 2005.

[70] M. Géradin and A. Cardona, *Flexible Multibody Dynamics: A Finite Element Approach*, vol. 4. 01 2001.

[71] S. Maraniello and R. Palacios, "State-Space Realizations and Internal Balancing in Potential-Flow Aerodynamics with Arbitrary Kinematics," *AIAA Journal*, vol. 57, no. 6, pp. 2308–2321, 2019.

[72] M. Goland and Y. L. Luke, "The Flutter of a Uniform Wing With Tip Weights," *Journal of Applied Mechanics*, vol. 15, pp. 13–20, 03 2021.

[73] T. Corke, *Design of Aircraft.* Prentice Hall, 2003.

[74] R. Vos and S. Farokhi, *Aerodynamics of Swept Wings*, pp. 427–511. Dordrecht: Springer Netherlands, 2015.

[75] M. Baudin, M. Christopoulou, Y. Collette, J.-M. Martinez, A. D. Lee, R. Sjögren, and D. Svensson, "pyDOE2: An experimental design package for python." `https://pythonhosted.org/pyDOE/#`, January 2020.

[76] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[78] Office of Primary Responsibility ACE-100, Small Airplane Directorate, "System Safety Analysis and Assessment for Part 23 Airplanes," Tech. Rep. 23.1309-1E, Federal Aviation Administration, 11 2011.

[79] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: pagmo," *Journal of Open Source Software*, vol. 5, no. 53, p. 2338, 2020.

[80] A. del Carre, A. Muñoz-Simón, N. Goizueta, and R. Palacios, "SHARPy: A dynamic aeroelastic simulation toolbox for very flexible aircraft and wind turbines," *Journal of Open Source Software*, vol. 4, no. 44, p. 1885, 2019.

[81] K. Jovanov and R. De Breuker, "Accelerated convergence of high-fidelity aeroelasticity using low-fidelity aerodynamics," in *Proceedings of the 16th International Forum on Aeroelasticity and Structural Dynamics, St. Petersburg, Russia*, vol. 28, 2015.

[82] C. A. Lupp, C. E. Cesnik, P. Beran, J. Deaton, and D. Easterling, "Including geometrically nonlinear flutter constraints in high fidelity aircraft optimization," in *International Forum on Aeroelasticity and Structural Dynamics (IFASD 2019)*, 2019.

[83] Y. Gavrilova, "How to choose a machine learning technique." Last visited in 2023/1/15.

[84] A. B. Lambe and J. R. Martins, "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes," *Structural and Multidisciplinary Optimization*, vol. 46, no. 2, pp. 273–284, 2012.

[85] J. R. Martins, J. J. Alonso, and J. J. Reuther, "A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design," *Optimization and Engineering*, vol. 6, no. 1, pp. 33–62, 2005.

[86] S. Chakraverty, N. Mahato, P. Karunakar, and T. D. Rao, *Advanced numerical and semi-analytical methods for differential equations.* John Wiley & Sons, 2019.

[87] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization," *SIAM Review*, vol. 60, no. 3, pp. 550–591, 2018.

[88] R. Yondo, E. Andrés, and E. Valero, "A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses," *Progress in Aerospace Sciences*, vol. 96, pp. 23–61, 2018.

[89] G. Mendonça, F. Afonso, and F. Lau, "Model order reduction in aerodynamics: Review and applications," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 15, pp. 5816–5836, 2019.

[90] M. Giselle Fernández-Gonino, C. Park, N. H. Kim, and R. T. Haftka, "Issues in Deciding Whether to Use Multifidelity Surrogates," *AIAA Journal*, vol. 57, no. 5, pp. 2039–2054, 2019.

[91] R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, and M. Steinbrecher, "Computational intelligence: A methodological introduction," *Computational Intelligence*, 2015.

[92] G. V. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[93] H. J. HASSIG, “An approximate true damping solution of the flutter equation by determinant iteration.,” *Journal of Aircraft*, vol. 8, no. 11, pp. 885–889, 1971.

[94] S. A. Dovgii and A. V. Shekhovtsov, “An Improved Vortex Lattice Method for Nonstationary Problems,” *Journal of Mathematical Sciences*, vol. 104, pp. 1615–1627, 2001.

[95] S. Y. Wie, S. Lee, and D. Lee, “Potential Panel and Time-Marching Free-Wake Coupling Analysis for Helicopter Rotor,” *Journal of Aircraft - J AIRCRAFT*, vol. 46, pp. 1030–1041, 05 2009.

[96] V. T. T. Nguyen, *Introduction to Optimum design*. 01 2011.

[97] M. Nikbay, A. Yanangonul, L. Oncu, and M. Kocas, “Multi-objective and gradient based structural design optimization of an aircraft wing,” 09 2008.

[98] M. Nikbay, A. Yanangonul, L. Oncu, and M. Kocas, “Multi-objective and gradient based structural design optimization of an aircraft wing,” 09 2008.