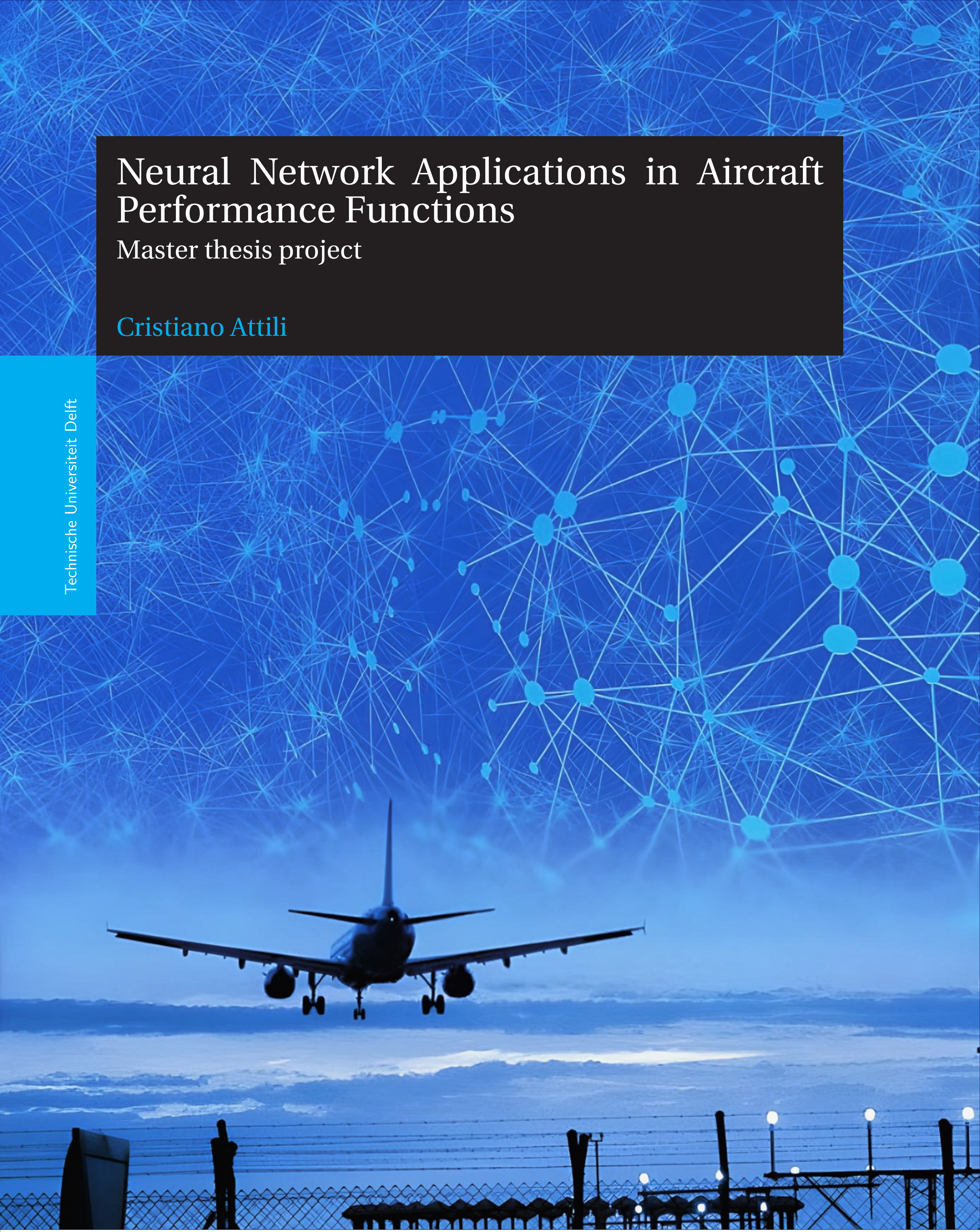


Neural Network Applications in Aircraft Performance Functions

Master thesis project

Cristiano Attili

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Neural Network Applications in Aircraft Performance Functions

Master thesis project

by

Cristiano Attili

in partial fulfilment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology,
to be defended publicly on Friday November 17, 2023 at 01:30 PM.

Student number:	4508122	
Supervisors:	Prof. Dr. ir. G. La Rocca	Delft University of Technology
	Prof. Dr. ir. M. Voskuijl	Netherlands Defence Academy

Equivalent number of words: 29600

Delft University of Technology

Preface

The following report is the culmination of a journey that I have shared with many people. I would like to thank my family and friends, my supervisors Mark and Gianfranco, the TU Delft institution and all the others that helped me along the way. You have my utmost gratitude.

Cristiano Attili

Abstract

One of the trends in the aviation world is to work towards an increasingly more computer-aided approach to flying. Over the course of the last decades, this idea has been fueled by the growth in available computational power and the concurrent decrease in hardware ownership costs. Thanks to modern-day architectures, several types of embedded functions can be integrated in aircraft cockpits, automating and simplifying most of the actions of the pilot.

Despite these improvements, limitations still inevitably exist in terms of power and storage capabilities. To overcome this problem, different solutions have been proposed. A data-driven approach is applied in this work, centered on the implementation of surrogate models. Embedded functions, in fact, typically rely on the use of various mathematical models, which describe the behaviour of some part of reality. As the models often represent a considerable burden in terms of storage and computational requirements, replacing them with surrogate models can represent a viable solution.

The analysis is focused on aircraft performance functions, whose typical scope is the computation of performance indexes such as takeoff and landing distances, rate of climb, range, fuel consumption, cost estimation or noise level. A standard performance model is composed of aerodynamics, propulsion and atmosphere submodels, which allow for the computation of the forces acting on the aircraft. These are coupled with the equations of flight mechanics and additional submodels accounting for further application-specific features. In this work, specific focus is given to landing performance functions, as the literature on the subject is still in the early stages.

A function integrated in the avionics of single-aisle Airbus aircraft is selected as a case study. The objective of the software is to calculate the state of contamination of the runway (e.g. water or soil) from information gained during landing. The idea is to send the results to the airport traffic services, to make the information available to ensuing aircraft approaching the runway. In this way, pilots of incoming aircraft are enabled to appropriately manage the landing and possibly avoid situations of runway overrun. The function is built on multiple levels and relies on various submodels for its internal calculations. The aerodynamics and engine models have been chosen as the target of the surrogate modeling activity, due to their high contribution to the complexity of the function. The reduction is performed at submodel level, though the possibility of creating a common aero-propulsive model is also explored.

As a model structure, feedforward neural networks are chosen for their flexibility, parsimony and capacity of modeling non linear systems. Polynomial models are also implemented as a means for comparison. Neural networks are trained with the Bayesian Regularization algorithm, with training and validation data retrieved from landing simulations. The selection of input features is carried out based on engineering judgement and a correlation study on the available variables. Furthermore, the work aims at enhancing the portability of the function by training models on different instances of aircraft and engines, which are modeled by a set of categorical inputs.

The objective of the activity is to investigate possible compromises between reduction of model complexity and accuracy of the obtained surrogate models. Neural networks with as little as one hidden layer are shown to achieve substantially low memory requirements while retaining satisfactory modeling accuracy, with noteworthy differences between the results on aerodynamics and engine model. It is demonstrated how several compromises can be achieved by choosing different model architectures, with different levels of impact on the chosen performance function. The applicability of each solution, however, remains a case-specific issue, depending on the given requirements in terms of model accuracy and complexity.

Contents

Preface **iii**

Abstract **v**

Nomenclature **ix**

1 Introduction **1**
- 1.1 Problem background . . . 1
- 1.2 Performance functions and models . . . 1
- 1.3 Proposed solution . . . 2
- 1.4 Report structure . . . 2

2 Literature review **3**
- 2.1 Research topic definition . . . 3
- 2.2 State of the art review . . . 4
- 2.3 Considerations and open problems . . . 5
- 2.4 Project features and elements of novelty . . . 6

3 Surrogate modeling and introduction to neural networks **9**
- 3.1 Reducing model complexity . . . 9
- 3.2 Surrogate models . . . 9
  - 3.2.1 Construction of surrogate models . . . 10
- 3.3 Neural networks . . . 12
  - 3.3.1 Formal definition . . . 12
  - 3.3.2 Neural network training . . . 14
- 3.4 Conclusions . . . 15

4 Case study: Aircraft performance and landing **17**
- 4.1 Introduction to landing performance . . . 17
- 4.2 Case study: prototype performance function . . . 18
  - 4.2.1 Working principle . . . 18
  - 4.2.2 Global function structure . . . 19
- 4.3 Conclusions . . . 20

5 Methodology **21**
- 5.1 Preliminary studies . . . 21
  - 5.1.1 Study of the problem features . . . 21
  - 5.1.2 Analysis of problem parameters . . . 23
- 5.2 Dataset building . . . 24
  - 5.2.1 Dataset retrieval . . . 24
  - 5.2.2 Sampling . . . 26
- 5.3 Model building and assessment . . . 28
  - 5.3.1 Selection of input parameters . . . 28
  - 5.3.2 Neural network structures . . . 30
  - 5.3.3 Training algorithms . . . 31
  - 5.3.4 Training settings and stopping criteria . . . 32
  - 5.3.5 Performance evaluation . . . 33
- 5.4 Additional evaluations . . . 34
  - 5.4.1 Polynomial models . . . 34
  - 5.4.2 Feasibility study . . . 34

6 Results **39**
6.1 Baseline model . . . 39
6.1.1 Global results . . . 39
6.1.2 Error analysis . . . 39
6.1.3 Conclusions . . . 43
6.2 Improvements of neural network structure . . . 44
6.2.1 General remarks . . . 44
6.2.2 Aerodynamics model . . . 44
6.2.3 Engine model . . . 47
6.3 Single reduced model . . . 47
6.4 Neural network with selected features . . . 47
6.5 Boolean categorical inputs . . . 48
6.6 Polynomial models . . . 48
6.7 Overview . . . 49
6.7.1 Assessment against accuracy framework . . . 49

7 Conclusion and recommendations **51**
7.1 Evaluation of the results . . . 51
7.2 Limitations and further studies . . . 52

Bibliography **55**

Appendix A: Elements of landing mechanics and performance **61**

Appendix B: Sampling process **65**

Appendix C: Relative error of the baseline model **71**

Appendix D: Polynomial models **77**

Appendix E: Comparison of results across model structure **81**

Nomenclature

Symbol	Definition	Unit
b	Neuron bias	-
$\underline{b}$	Network biases	-
$\hat{\underline{b}}$	Optimal network biases	-
C_D	Drag coefficient	-
C_L	Lift coefficient	-
C_M	Moment coefficient	-
CG	Center of Gravity	%
CONF	High-lift configuration	-
$cov(x,y)$	Covariance of the variables x and y	$[x$ unit$] \cdot [y$ unit$]$
D	Aerodynamic drag force	N
d	Input dimension	-
d_{real}	Actual landing distance	m
d_{ref}	Reference landing distance	m
E	Error function	-
E_D	Output error-based error function	-
E_W	Network weight-based error function	-
e	Euler's number	-
f	Function	-
F_{BRAKE}	Friction force on the braking wheels	N
F_{ROLL}	Rolling resistance force	N
GS	Ground Speed	m/s
g	Gravitational acceleration	m/s^2
	Function (alternative)	-
H	Hessian matrix	-
h_{screen}	Screen height	m
I	Identity matrix	-
i	Index	-
J	Cost function	-
	Jacobian matrix	-
j	Index	-

k	Index	-
	Polynomial degree	-
L	Aerodynamic lift force	N
LOW	Low autobrake setting	-
MAE	Mean Absolute Error	variable unit
MAX	Maximum autobrake setting	-
MED	Medium autobrake setting	-
m	Output dimension	-
	Aircraft mass	kg
N	Number of items	-
N_P	Number of parameters	-
N_T	Number of terms	-
$NMAE$	Normalized mean absolute error	%
$N1$	Engine low pressure spool speed	%
n	Number of samples	-
n_T	Number of training samples	-
R	Reaction force	N
R_{MAIN}	Reaction force on the main landing gear	N
R_{NOSE}	Reaction force on the nose landing gear	N
R^2	Coefficient of determination	%
Rev	Thrust reverser ON/OFF	-
$RMSE$	Root Mean Square Error	variable unit
r_e	Effective wheel radius	m
s	Neuron output	-
T	Tangential force	N
TAS	True Air Speed	m/s
t	Timestep	-
V	Aircraft forward speed	m/s
$var(x)$	Variance of variable x	$[x \text{ unit}]^2$
w	Neuron weight	-
$\underline{w}$	Network weights	-
$\underline{\hat{w}}$	Optimal network weights	-

X_{PROP}	Longitudinal engine thrust	N
x	Input variable	x unit
$\bar{x}$	Average of variable x	x unit
$\underline{x}$	Input vector of a function	x unit
y	Output variable	y unit
$\bar{y}$	Average of variable y	y unit
$\underline{y}$	Output vector of a function	y unit
$\underline{\hat{y}}$	Output vector of a surrogate function	-
Z_{PROP}	Vertical engine thrust	N
α	Angle of attack	$^\circ$
	Regularization parameter	-
β	Regularization parameter	-
γ	Learning rate of gradient descent	-
	Flight path angle	$^\circ$
	Effective number of parameters in Bayesian Regularization	-
ε	Sample error	variable unit
$\underline{\varepsilon}$	Error vector	variable unit
λ_S	Longitudinal slip ratio	-
μ	Friction coefficient	-
	Damping factor of optimization algorithm	-
ρ	Pearson correlation coefficient	-
Σ	Summation	-
	Sigmoid transfer function	-
σ_x	Standard deviation of variable x	x unit
φ	Neuron activation function	-
ω	Rotational speed	1/rad
∇	Gradient	-

1
Introduction

1.1. Problem background

Since the early years of powered flight, the need for information about the environment and the state of the aircraft itself led to the development of various forms of instruments. Fueled by military demands during the 1930's, a wide range of tools were developed for various purposes. It is during this period that the first analog computers made their appearance on boarded systems, mainly as an aid for weapon operation.
The following decades saw a dramatic evolution in accuracy and scope of flight instruments. A few milestones worth mentioning are the introduction of radio navigation and on board electronics (hence the term "avionics"), the deployment of digital computers in the 1970's and of the FMS (Flight Management System) in the 1980's. This process brought a profound change in the architecture of the aircraft, especially to the cockpit, which evolved from a complex series of switches and indicators to the so called "glass cockpit" that we know today. [1] [2]
In this context, the use of software on board has taken a more and more prominent role, with the result of automating and simplifying most of the actions of the pilot, and introducing completely new possibilities. Some examples are: flight controls, engine monitoring, navigation, flight planning, communications and safety. The trend has been even more pronounced over the last 20 years, which saw a dramatic growth in extent and complexity of the implemented software. [3] Such growth is largely ascribable to the concurrent improvements in reliability and affordability of hardware (also in terms of boarded weight), thus allowing for an increase of available computational power on board. Modern aircraft rely on a complex network of computers and buses which elaborate and share information to serve their diverse purposes.
Notwithstanding these improvements in the means, it is clear that limitations still exist in terms of power and data storage capabilities. [4] [5] In fact, not all applications can be implemented and used on board in any given form. Moreover, the question is expected to become more and more central as we approach a natural limit in the miniaturization of transistors as traditionally given by Moore's law. [6] [7]

1.2. Performance functions and models

Among the different pieces of software that are integrated in aircraft cockpits, a relevant part falls under the category of performance functions. The purpose of these tools can be stated as the computation and detection of aircraft performance characteristics. Such characteristics are synthesized by calculation of performance indexes describing, for instance, takeoff and landing distances, rate of climb, range, fuel consumption, cost estimation or noise level.
For the calculation of the mentioned parameters, a number of specific models and algorithms are required, as they often derive from the combination of several aircraft features. Under a theoretical point of view, a performance model (as intended when integrated within a larger function) is made up of other submodels, typically: aerodynamics, engine (propulsion) and atmosphere. These, which allow for the computation of the forces acting on the aircraft, are coupled with (some form of) the equations of flight mechanics for prediction of the trajectory. Additional submodels accounting for other characteristics (e.g. environmental impact) may be featured, according to the specifically sought performance index. Performance models are essential in several moments of the aircraft life-cycle. The implementation starts from conceptual and preliminary

assessments during the design phase, through flight test campaigns, then up to the whole operational phase under the form of embedded functions and customer service investigations.
Performance functions and models, similarly to other types of software, represent a considerable burden for aircraft systems. On one hand, there is an inherent complexity of certain function modules that determines a high computational cost, especially in real-time applications. On the other hand, all the abovementioned models are often based on a high number of given values representing, typically, the peculiar features of each aircraft and engine type the function is integrated into. The more elements the function is intended to include, the higher will be its storage requirements. [4]

1.3. PROPOSED SOLUTION

To overcome the problem of function integration, different procedures have been proposed and in some cases implemented. Notable solutions include: development of standards, integrated system architectures, memory optimization, data compression, parallel and distributed computing. [8] [9] [10] [5] [11]
This work is focused on a fundamentally different approach that, in contrast with those just mentioned, is application-specific and model-oriented. The idea behind this solution is to exploit the potential of surrogate modeling techniques, in order to reduce the complexity of aircraft performance functions. As described more in detail in the next chapters, surrogate modeling has been successfully implemented in several fields over the last decades, to serve the most diverse purposes and applications. The use of surrogate models for the reduction of aircraft cockpit functions, however, is still in its earlier phases and a consistent body of literature on the topic is missing.
The scope of this work is to investigate potential solutions in the reduction of aircraft performance functions, by finding suitable compromises between complexity and accuracy of the surrogate models. The specific technique implemented here is that of artificial neural networks, a versatile and efficient model structure that has been long identified as a suitable solution for this problem. [12] Neural networks will be trained by using different algorithms and architectures, and an additional comparison will be made with multivariate polynomial models.
A specific case study has been selected for the research, a prototype performance function developed by Airbus. The function computes the state of runway contamination during the landing phase, by estimating the aerodynamic, propulsive and ground forces on the aircraft. The reduction is thus performed at submodel level, taking into account the individual aerodynamics and engine model separately. Moreover, in order to enhance the portability and efficiency of the proposed solution, the reduced models will be trained on data from all single-aisle aircraft that implement the function.

1.4. REPORT STRUCTURE

Following the present introductory chapter, a review of the main literature on the topic of interest is presented. This will show what aspects have already been investigated and what points are still lacking a full clarification in the available literature. The chapter will additionally clarify the scope of the proposed research and underline its main aspects of innovation and relevance for the body of science.
A full chapter is then dedicated to the topic of surrogate modeling, due to the relative novelty of the subject. Special attention is given to neural networks, which is the main model structure implemented in this work.
The following chapter will provide theoretical information regarding the specific case study taken into account in the research. Elements of landing performance are provided, to provide the grounding for latter phases of the work. Moreover, given the characteristics of the prototype function under examination, an overview of the main aspects and problems of braking will also be provided. A description of the prototype function is thus outlined, focusing on the scope, structure and working principle of the function.
The Methodology chapter extensively describes how the research has been structured, as well as the rationale behind the choice of the techniques and processes implemented the investigation. The chapter has been organized in sections that follow tightly the different phases of the work, in the order that they were carried out.
The Results chapter is dedicated to the outcomes of the research. This part will first consider the different proposed solutions separately, then provide a final overview to compare the main features of each solution.
The last chapter of the thesis will provide some meaningful conclusions about the outcomes the work. Here an analysis of the results will be made in the light of the objectives set at the beginning of the work. Lastly, a set of recommendations will be made for potential future studies on the topic.
Additional details on various topics are provided in the Appendixes at the end of the document. This includes all aspects that may still be of interest for the reader, though not strictly relevant to the research itself.

2
Literature review

Prior to carrying out the master thesis project presented in the previous chapter, a detailed exploration of the literature available on the topic of interest has been performed. The outcomes of this investigation have been summarised in the high-level analysis presented in the following sections.
The goal of the literature review is twofold. On one hand, it allows for a comprehension of the state of the art in the field. This means that both the content and the methodology of the currently available knowledge are highlighted and understood. On the other hand such review shows, by contrast, what areas have not yet been extensively researched.
Based on these considerations, it is then possible to clearly state a set of objectives for project. Particular stress is laid on the aspects of novelty of the work, which are deemed to expand the currently available body of knowledge on the topic.

2.1. Research topic definition

A crucial aspect to be defined ahead of the literature study, is the exact topic of research. In this case the question is not trivial, as the work is at the crossing point of different disciplines. Moreover, the matter is fundamental in order to have a clear idea of what literature is relevant to the work.
A systematic way to go about this problem is by breaking down the general definition of the project activity, which can be stated as:

Reduction of aircraft performance functions by means of neural networks for integration in aircraft systems

A first consideration can be made on the term "reduction". In the whole document, this will be intended as (equivalent to "surrogate modeling" or "metamodeling") the activity of creating surrogate models of a preexisting model, with the goal of reducing its original complexity. It could be argued that the project be rather revolving around a task of system identification, based on the eminent description of "the determination, on the basis of observation of input and output, of a system within a specified class of systems to which the system under test is equivalent". [13] It is evident from this statement how such activity can be closely compared to that of the present work. For that reason and because of the several common techniques and procedures that are used in system identification applications, works of this category are also included in the literature review.
The review will thus include applications of metamodeling techniques in the aerospace industry. The research on the literature has been mostly limited to such domain, with the notable exception of a set of documents that make the foundations of modern surrogate modeling activities, and hence represent an undeniable reference for the project. Specific stress is given to works that include the implementation of neural networks as a metamodel structure, whose examples are certainly not lacking in the literature.
As a last consideration, the focus of the present work is on the aerodynamics and engine models within an existing landing performance function. It is clear that any work dealing with the reduction of either of the two models, regardless of their specific form, is a valid source of information. These models, as it will be shown in the following paragraphs, can be found as: standalone systems, models inside a function or even submodels of, typically, an aircraft performance model. Hence all works that involve the reduction of aircraft performance functions or models are of interest for this review.

2.2. STATE OF THE ART REVIEW

The use of surrogate modeling in aerospace applications has seen a substantial increment over the last 20 years, which is the time frame where most of the works examined here are found. The relative delay between theoretical conception of the techniques and their large-scale implementation, is mainly due to the recent hardware and software improvements mentioned in 1.1. The objectives targeted by these applications are quite diverse and could be grouped under the categories of: function integration, parameter monitoring, design space exploration, optimization and system identification.

The set of works that most closely deal with the topic of the present research is indeed that of Bondouy et al. [4] [14] [15] In their investigation, in fact, the central subject matter is the development of reduction procedures for integration of aircraft performance models within avionics. Three requirements are identified to assess a surrogate model resulting from a proposed solution: its accuracy, standard computational time and memory size. RS-HDMR (random sampling-high dimensional model reduction) and neural networks are chosen as metamodel structures because of, respectively, their capability of dealing with high-dimensional problems and the reduced number of parameters required to approximate any given function. To overcome the arising curse of dimensionality observed in the construction of HDMR models, an original approach is thus proposed and successfully tested on the real case of the Fuel Consumption Model, which is embedded in the FMS. The idea is to exploit the structure of the model itself, which is made up of a number of submodels and analytical functions organized in a hierarchical multilevel architecture. Each submodel is thus replaced by a corresponding metamodel with no more than 4 input features, either HDMR (up to second order) or a single-layered feedforward neural network. The process is thus formulated as an optimization problem where the objective function is the overall memory size, under the constraints of a set level of accuracy and computational time. Configurations resulting from an optimization process are shown to be hardly ever intuitive, thus justifying the complex approach undertaken.

Similar objectives, though at system level, are found in Ghazi et al. [16] The aim of the work is to build an aero-propulsive model for implementation in the FMS trajectory optimization function [17], which is constrained by the processing capacity of the device and thus needs to be of low complexity. Excellent results were obtained by creating a model based on simulated data and physical laws, rather than via a black box approach. In this case in fact, the fitting is performed by estimating the values of the Oswald factor, $C_{D_{min}}$ and engine thrust in the proposed model, by means of the Nelder-Mead [18] minimization algorithm. Gong and Chan [19], as well as Trani and Wing-Ho [20], later implemented analogous approaches, though using flight manuals as a source of data. An interesting application of deep feedforward neural networks (FFNN) on the reduction of full aircraft functions is given by the work of Julian et al. [21] on aircraft collision avoidance systems. In this study, networks with 6 hidden layers approximate the optimal avoidance decision out of 5 available choices (output scores), based on 7 discretized input states. Julian et al. successfully test different network configurations to dramatically decrease computational time and notably storage requirements of the application by a factor 1000. It is also worth mentioning the work of Pytka et al. [22] on ground performance measurement by means of convolutional neural networks, based on flight test data of a utility aircraft.

A field of research where the use of metamodels has seen an exponential growth in the last years is indeed that of design and optimization. The interest here is in the possibility of expediting functional evaluations, so that efficient explorations of the design space can be performed in reasonable time. Moreover, sometimes a surrogate model can be useful in gaining a deeper insight into the features of a certain system, before effectively establishing a specific design problem. In the context of aircraft performance, significant reduction studies have been performed either at model or (especially) submodel level. The need for such investigations is often due to the complexity in simulating specific systems, notably when considering global aircraft performance, where multiple disciplines come into play and MDO (multidisciplinary design optimization) activities are required. A very powerful example in this sense is the work of Jules and Lin [23], where an innovative MDO tool for aircraft-engine system multi-objective optimization is presented. Metamodeling methods are applied at different levels: Taguchi techniques compute a parameter significance index, neural networks effectively map the single disciplines and fuzzy logic is implemented to manage conflicts between different objectives. A different approach is implemented by Paiva et al. [24], who rather attempt at generating a metamodel of the whole MDO tool via interpolating quadratic response surfaces, regression Kriging and single-layered feedforward neural networks.

Several investigations have been carried out at submodel level for design or monitoring purposes. Due to its complexity and limited accessibility, the engine is a frequently explored system. A work of great interest is that of Corman and German [25] on the reduction of engine cycle models for faster design explorations. Here polynomial regression, FFNN and radial basis functions are chosen to build surrogate models of the cruise

thrust specific fuel consumption (TSFC) with neural networks outperforming the other two techniques by a great margin. Various metrics are implemented to evaluate the results: R^2_{adj} (adjusted coefficient of determination), RMSE (root mean square error) and MAE (mean absolute error), as well as the required number of parameters in the model structure. The excellent capability of neural networks in predicting engine performance is also confirmed by Lee et al. [26] in their computation of engine performance parameters (gross thrust, ram drag, fuel flow, nitrogen oxide emission index) via FFNN. Principal component analysis (PCA) is implemented in Lee's work to perform feature selection, which is a problem that has been dealt with in different complex ways. [27] Additional works worth mentioning are those of Ji [28] on engine fuel flow, Yildirim and Kurt [29] and Zhang and Wang [30] on exhaust gas temperature (EGT) and engine revolutions, Meyer [31] and Gao [32] on engine dynamic performance, Zavoli [33] on hybrid rocket engines.
Among the main works on aerodynamic metamodeling is found that of Nørgaard et al. [34] who employ wind tunnel test data to minimize the number of required wind tunnel measurements during design. Here four FFNN are trained to predict the coefficients C_L, C_D, C_M and aerodynamic efficiency. Wallach et al. [35] instead perform an extensive investigation on aerodynamic coefficient modeling by means of various multi-layer FFNN structures, for both full aircraft configurations and airfoils. The latter, though not being the focus of the present document, is indeed a frequent target of surrogate modeling activities. [36] [37] [38] [39] [40] [41] Lastly, as an example of application in anomaly detection, it is worth mentioning Raol's investigation [42] on lift and moment coefficient calculation by means of FFNN, based on data generated from simple laws of aerodynamics.
Additional implementations of metamodeling techniques have been studied for this project, in order to gain a full knowledge of the available methods and resources in the field. These include, for instance, applications in air traffic control (ATC) [43], aircraft control systems [44] [45] [46], combustion flowfield analysis [47] and avionics fault diagnosis. [48]

2.3. CONSIDERATIONS AND OPEN PROBLEMS

The literature research presented in the previous section has assessed the state of the art in the topic of interest, identifying the methods that could prove useful for a novel research project. One of the main difficulties in building such review was given by the lack of a coherent body of work in the available literature. This is due to a number of factors, including the relative newness of the topic. Moreover, the field of metamodeling is a hybrid area of research, where theories, methods and techniques coming from various sources of knowledge are applied together. In addition to that, the construction of surrogate models is a process that (as seen in the above) can be applied to many different areas of expertise. It doesn't help in this context, that the topic of flight performance is itself the point of synthesis of diverse fields of knowledge. Because of all these aspects, finding a common thread and drawing some meaningful conclusions from the available material is indeed a challenging task.
Nonetheless, several valuable insights have been gained from this activity, including what points are still open in the research. A concise list of the main elements that were identified is given in the following:

1. There is an undeniably growing interest in the field of metamodeling, especially for what concerns aviation. The last decade in particular has seen a steady impulse in the application and research on surrogate models. For the moment, the vast majority of works are focused on applying already well tested solutions while a small part is effectively trying to stretch the current range of known possibilities.

2. The availability of data is a real issue in a domain of research where the central approach is data-driven. Several authors point out the reluctance of manufacturing companies and organizations in sharing proprietary data. [16] [19] [20] The problem is often solved by implementing other means to retrieve raw data, such as simulators and manuals.

3. The specific topic of research has been investigated by a small number of productions so far. This can be due on one hand to the limitations outlined at point 2. and on the other hand to the relative novelty of the concept. The only work that effectively aims at providing a solution to the identified problem is that of Bondouy et al. [4]. Other investigations focused on very similar scopes have however different objectives, which leads to establishing different criteria for assessing the results of the research. In most works, in fact, the baseline objective is to obtain a surrogate model with an acceptable accuracy. In the present project instead, additional constraints are set regarding the integrability of the created models within specific operational frameworks.

4. An understanding of the subject matter, where surrogate models are implemented, is paramount to set up such activity. This reveals necessary for a sound choice of the data to be used for model fitting and to discern whether the results are acceptable for the relevant framework. Moreover, this prevents unnecessary iteration over the steps of the metamodeling process.

5. Concerning aircraft performance problems, the metamodeling activity is mostly carried out at performance model level. To find investigations that deal specifically with performance submodels it is necessary to refer to works that are dedicated specifically to that aspect. The productions of Bondouy et al. are again the only exception to this finding. Moreover, the possibility of creating combined surrogate models of multiple performance submodels seems not to have been widely explored by the research so far.

6. The vast majority of works in this field are focused on specific case studies to test out some new technique or simply apply the current knowledge to solve a problem. The data on which the surrogate models are fitted, are hence relative to precise contexts, which can be for instance a certain type of aircraft or engine model. The possibility of training models that could work in more heterogeneous contexts seems to have not been investigated at full scale by the research in the field, probably because of the higher degree of difficulty bound to such an effort. The main blocking point appears to be the definition of relevant "categorical data", representing different operational settings, as well as their suitable numerical representation for model training. Moreover, this means that the models will have to be fitted on both discrete and continuous inputs, leading to several practical issues. [49]

7. Neural networks are indeed the model structure that has been most widely implemented, with several notable examples of feedforward networks with a single hidden layer. The reason for this trend is found in the great flexibility and relative ease of implementation represented by neural networks. In addition, the possibility of approximating any function with only one hidden layer of neurons is indeed an attractive perspective for any type of metamodeling activity. The accuracy demonstrated by this technique in the works presented here confirms these theoretical intuitions.
   Fuzzy logic models are also gaining recognition thanks to their ability of dealing with uncertainty in the data. This is particularly interesting for real-world applications where data come from several raw sources. It is lastly worth mentioning polynomial models (RSM), which remain a valid alternative to more cutting-edge methods. Though significantly less accurate, polynomials are still being successfully applied in approximation of low-dimensional functions and as a fast means of comparison for performance assessment.

8. The sampling of data to build a suitable dataset for training is mostly carried out via techniques that guarantee a uniform distribution of points (space filling), in particular methods such as Latin Hypercube Sampling and random space filling.

9. For what concerns the validation of models, the most typical approach is to split the main dataset into two different blocks, where one is used for training and the other for validation. The proportions between the two can vary substantially, though it is strongly suggested to provide diverse and complete datasets for both purposes. Several statistical means are available to assess the accuracy of a model. Measurements such as RMSE and other fit coefficients (R^2) have been extensively implemented for this purpose.

10. The choice of input variables is mostly performed via engineering judgement, correlation studies or simply by keeping the same parameters of the original model. The choice of input features is often done in a suboptimal way, presumably due to the high cost in terms of time compared to the expected benefits.

2.4. PROJECT FEATURES AND ELEMENTS OF NOVELTY

Based on the observations drawn in the previous section, it is possible to outline the main characteristics of the proposed thesis project. It will also be highlighted what aspects effectively attempt to overcome the gaps that have been found in the body of research.
As already pointed out in the above considerations, the research on the topic of metamodeling for aircraft function integration is still in the early phases. A notable exception is the work of Bondouy, which also revolves around aircraft performance functions. The present work, however, aims at investigating the problem

under a different light. The main characteristics of the research project can be summarised in the following points:

- A landing performance function is taken as a case study for an investigation on the reduction of aircraft performance functions. The main objective is to find a set of possible solutions for the problem of function integration, by testing various metamodel configurations trained with different algorithms, then assessing their performance. Such assessment is carried out through both accuracy and memory requirement indexes.

- The data used for model training and validation is obtained via a simulator, which is provided by the aircraft manufacturer. This aspect is not to underestimate, in the light of point 2. in the previous section. Moreover, the simulator is implemented along with a sampling process, in order to obtain balanced and complete datasets that fit the specific operational needs of the case study.

- Artificial neural networks are chosen as the main surrogate model structure because of their high flexibility, accuracy in nonlinear modeling and parsimony in the number of parameters. These characteristics are well demonstrated in the practice, as seen with the works examined in 2.2 and will be justified more in depth in the next chapter. Multivariate polynomial models are also implemented for comparison with the main metamodel structure, as done for instance in [25].

- Surrogate modeling is carried out at performance submodel level. Hence the aerodynamics and engine model being reduced are explicitly a part of an aircraft performance framework, which dictates their general characteristics. This is in contrast with the vast majority of the works described in 2.2, including that of Bondouy, which is based on an optimal reorganization of the modules. An additional possibility is also tested in this project, by attempting a reduction of the two performance submodels together in a single metamodel, with common input and output features.

- One of the most interesting traits of novelty is indeed that of training metamodels that are valid for different operational contexts. In the present case, the models will be trained on data coming from multiple (single-aisle) aircraft versions and engine manufacturers. For this type of investigation, specific "categorical inputs" will have to be wisely chosen in order to describe, in mathematical terms, the specific configuration on which the models are operated. This feature meets the needs of portability and efficiency that may be demanded by aircraft manufacturers.

- The choice of input variables is based on specific studies that are explained more in detail in Chapter 5. Furthermore, a feature selection process is implemented, to achieve a further degree of variation in the range of proposed metamodeling solutions.

- The outcomes of this project are assessed in the light of the set objectives and limitations of the work. This is done by comparing all the different solutions, but also taking into account the specific context in which the models are meant to be deployed.

3
Surrogate modeling and introduction to neural networks

The current chapter aims to provide the reader with the essential background for an understanding of the project. This regards the concept of surrogate modeling, which springs from the more general idea of system simplification. Artificial neural networks are then introduced, drawing from an intuitive comprehension of the technique, to then give a more rigorous mathematical description. The chapter will also present a high-level classification of neural network architectures, along with the main aspects of model training.

3.1. Reducing model complexity

All models that are embedded in aircraft functions are the result of a systematic process. Typically, the more steps the model has to go through, the less accurate it will be in the end with respect to the original physical model. This procedure is however necessary both to create a model that is closer to the machine language, but also to obtain a piece of software with limited burden for the aircraft systems.
The need for model simplification can be dated back to the early 19th century with Fourier's studies on trigonometric series [50]. A formal, comprehensive study is given in this regard by Chwif et al. in [51] and [52] where the subject of model complexity is discussed for reduction of simulation models.
Computational complexity is a concept that can be directly related to the amount of resources required to run a certain algorithm. The exact definition of "resources" is also a subject of debate but typically three main measures are considered: computational time, required memory and number of arithmetic operations (hence the common definition of "arithmetic complexity"). These features are at times considered separately or together, according to the specific application. [53] [54] [55] For the problem tackled in this work, the time component is clearly a remarkable feature, though not the most prominent. On the other hand, the aspect of memory requirements is indeed a crucial one, as the computational memory of avionics has been shown to be overloaded by a great amount of highly demanding applications. If volatile memory is central, not less important is the aspect of secondary memory, which is deemed critical in aircraft systems. Both aspects place a considerable burden both in the operation of cockpit systems and in the possibility of introducing additional applications on board.
In the work of Brooks and Tobias [56] a distinction is made between three categories of reduction methods: rearrangement of the code, techniques maintaining the original model output and techniques that approximate the original output. With the third category it is possible to associate the concept of metamodeling.

3.2. Surrogate models

A metamodel, or surrogate model, can be simply defined as "model of a model" [57], that is, a mathematical entity approximating the input/output behaviour of another model.
The reasons why the implementation of a surrogate model may be preferred to the original one can be quite diverse. Intuitively, as it is impractical to recreate a physical phenomenon each time a simulation is needed for research or operational purpose, in the same way a metamodel can be advantageous with respect to its original version. For engineering applications, metamodels are typically built when operation of the original model presents a high computational cost. In this sense, surrogate models are constructed to reduce the

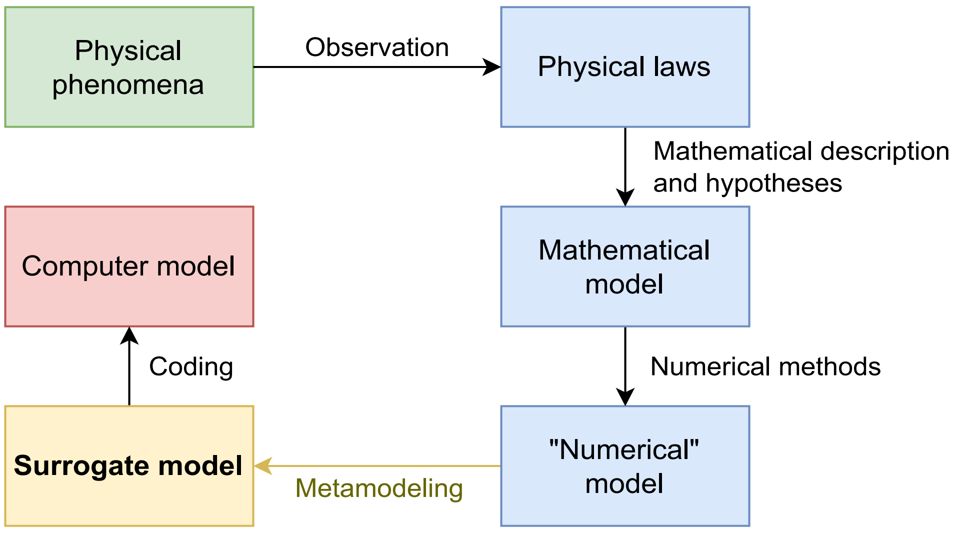


Figure 3.1: Workflow for construction of an embedded model, including a metamodeling phase.

computational complexity of the original model. This can be very valuable in engineering research, where faster (and rougher) design space explorations are eased by the use of metamodels. [58] [59] Several further examples can be found in the literature review of Chapter 2. In the context of aircraft embedded functions, an essential aspect of interest in metamodels is the possibility of reducing the storage requirements of a model. These models are essentially based on functional evaluations and interpolation of data tables. Performing a reduction through metamodeling can in some way be interpreted as the exploitation of the latter principle. The storage cost of a metamodel, in fact, is essentially the integration of a set of parameters that characterize the specific features of the model. The application of a metamodel then only consists in its evaluation on a set of inputs. Hence, an additional cost is given by integration of a model structure under the form of an embedded function, which is evaluated on the stored parameters and the inputs provided by the aircraft systems upon call of the model.
This procedure thus presents a twofold advantage with respect to a standard integration method. On one hand the computational effort is minimized, since it consists in only one functional evaluation. On the other hand the storage cost is also dramatically reduced, as only a (small) set of coefficients and one function need to be integrated in the cockpit. These advantages, however, have to be judged in the light of a reduced accuracy in the metamodel, which is the result of an approximation process. A trade-off is therefore always sought in the implementation of metamodels, between the relative benefits of simplification and approximation power. [25]
The scope of the reduction activity within a broader modeling process can be seen in Figure 3.1. The well-known model life cycle starts from the observation of physical phenomena and subsequent formulation of conceptual (physical) laws [60]. A mathematical description of the laws, along with contingent simplifying hypotheses, results in a mathematical model. At this stage, the original system is described through a series of more or less complex equations in a certain number of variables. A so-called "numerical model" (which will still be indicated here as "mathematical") of the original phenomenon is finally obtained by translating the mathematical model into a form that can be understood by a computer and can be solved within acceptable time. To achieve this, numerical methods such as discretization of variable domains and functional approximation are implemented. Metamodeling is included as an additional step before the final integration procedure (e.g. writing the model in a programming language).

3.2.1. CONSTRUCTION OF SURROGATE MODELS

Having introduced the concept of surrogate modeling and its relevance for this work, it is worthwhile to briefly elaborate on the procedure of constructing a metamodel. Following the notation proposed by Simpson et al. [61] (where vectors are here represented by underlined symbols), if the "true" functional relationship between inputs and outputs ($\underline{x}$ and $\underline{y}$) in a given model is

$$\underline{y} = f(\underline{x}) \tag{3.1}$$

then a surrogate model can be built in the form

$$\underline{\hat{y}} = g(\underline{x}) \tag{3.2}$$

such that between original and approximated output variables holds the relationship

$$\underline{y} = \underline{\hat{y}} + \underline{\varepsilon} \tag{3.3}$$

where the deviation $\underline{\varepsilon}$ is due to both approximation and measurement errors.
The core of the metamodeling activity is thus the construction of a suitable function $g(\underline{x})$. The way this is achieved depends on the choices made at several steps of the process, whose structure is generally agreed upon in the literature. In this sense, a very detailed analysis is given by Kleijnen [62], who proposes the following standard procedure:

1. Determine the goal of the metamodel
2. Identify the inputs and their characteristics
3. Specify the domain of applicability (experimental region)
4. Identify the output variables and their characteristics
5. Specify the accuracy required of the metamodel
6. Specify the metamodel's validity measures and their required values
7. Specify the metamodel, and review this specification
8. Specify a design including tactical issues, and review the DOE
9. Fit the metamodel
10. Determine the validity of the fitted metamodel

While most of the above points are self-explanatory and can be related to the discussion of previous sections, more attention is required for numbers 2., 7. and 8. As for the first, the choice of the inputs to a certain metamodel is not a trivial task. Feature selection and extraction are the processes by means of which a set of relevant parameters are chosen as inputs for a surrogate model. Several techniques are included under this definition and the topic is still an open point for research. Methods range in complexity from sensitivity studies and engineering judgement to iterative procedures based on sequential model training. [63]
Point 8. essentially refers to the definition of a suitable set of input-output data to be collected by running the original model. These are the data on which the surrogate model is built at point 9. The theory of design of experiments (DOE) thus plays a crucial role in determining a statistically optimal set of simulations (experiments) from which the data is extracted. Several techniques are available for an optimal design of experiments and the choice of the best method is contingent on the specific requirements of the problem. It is not guaranteed, for instance, that the choice of a uniform spacing will always yield the best results [64]. Moreover, the optimal selection of data must also be pondered based on the demanded accuracy and operating conditions of the implemented surrogate model. This is particularly critical for applications on hierarchical model structures, where cross-level error propagation can dramatically impact reliability. [65]
Similar considerations can be made on point 7. with respect to the selection of convenient surrogate model structures. This aspect is not trivial and in most cases it is inherent to the specific nature and requirements of the problem under examination, as well as the way the first 6 points of the process have been carried out. [58] [61] [62] [66] Furthermore, it can be argued that this point also include the selection of a fitting method (interpolation or minimization of a residual), which is in turn subject to the choice of the metamodel.
Several model structures are available for the activity, whose relevance for the present application has been outlined with examples in 2.2. Among the main surrogate structures it is worth mentioning neural networks (which will be examined in detail in the remainder of the chapter), multivariate polynomial functions [25] [36] [61], splines [58] [67] [68] [69], Kriging [59] [70] [71] [72], High dimensional model representation (HDMR) [15] [73], support vector machines [37] [74], fuzzy regression [75] [76] [77] [78] and hybrid model structures. [4] [79] [80] [81] [82]

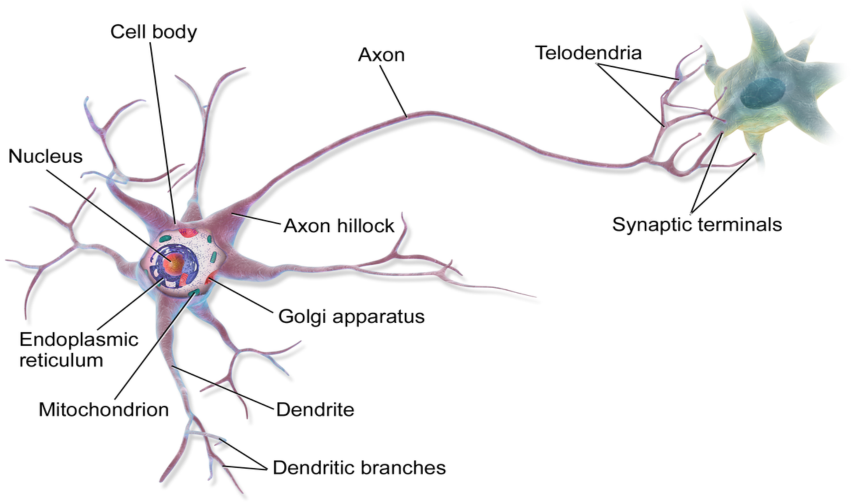


Figure 3.2: Representation of a biological neuron with synaptic connections. [83]

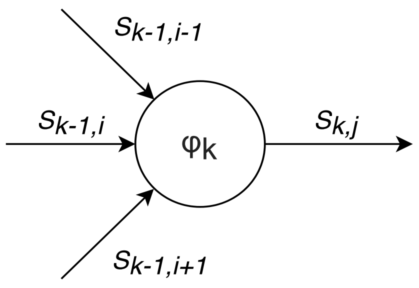


Figure 3.3: Formal representation of a neuron, with inputs $s_{k-1,i}$ and output $s_{k,j}$.

3.3. NEURAL NETWORKS

Of all metamodel structures presented above, artificial neural networks (here often shortened as NN) clearly deserve special attention, being the focus of the reduction activity in the project. The technique, as the name says, draws inspiration from its direct biological counterpart in the human brain (Figure 3.2). Just as with neurons and synapses exchanging information through electrical impulses, artificial networks are made up of nodes (or neurons) that are connected by logical relationships. Neurons are typically organized in layers and the simplest form of network consists in an input layer, an output layer and one hidden layer between the first two. This is called a shallow neural network. When more than one hidden layer is present between the input and output ones, this is generally defined as a deep network. There is no limitation to the quantity of nodes and layers in a network, as well as the number of inputs and outputs.
The first conception of neural network models dates as back as the 1940s, with the work of McCulloch and Pitts on the mathematical modeling of neurons. [84] Since then, theoretical investigation and practical application have grown together in response to technological advancement and need for new modeling methods. Neural networks are in fact currently implemented in a number of approximation and classification problems. The range of applications is nowadays very broad and concerns the most diverse fields, such as automotive, banking, finance, entertainment, manufacturing, medical and telecommunications. Several examples of implementation in the aerospace sector have also been provided in the previous chapter.

3.3.1. FORMAL DEFINITION

For a more rigorous introduction to the idea of neural network modeling, it is useful to first define its fundamental unit: the neuron. Such element is modeled such that, as shown in Figure 3.3, it accepts numerical inputs from one or more sources and forwards an output to one or more targets. In a structured network, these sources and targets are typically other neurons. The only exception holds for neurons in external layers (input and output layer), whose function is to accept inputs from or forward outputs to the outside environ-

ment. Still with reference to Figure 3.3, it can be seen how the indexes k, j and i iterate respectively on the network layers, the nodes in the currently considered layer and those of the previous layer.
A fundamental property of a neuron is the logical relationship that links inputs and outputs. As a general rule, the following mathematical expression can be given for the output of a node:

$$s_{kj} = \varphi_k\Big(b_{kj} + \sum_{i=1}^{N_{k-1}} w_{k_{ji}} s_{(k-1)i}\Big) \tag{3.4}$$

In other words, the output of a neuron is given by a linear combination of its N_{k-1} inputs $s_{k-1,i}$, weighted by a set of coefficients $w_{k_{ji}}$. A bias term b_{kj} is typically added to the expression as well.
A crucial element is the operator φ_k, which is called activation (or transfer) function and is usually chosen as the same in every neuron of a layer. The role of the activation function is to provide a neural network with nonlinear characteristics (and hence justify the use of multiple layers), which is the reason why hidden layer functions are typically chosen as nonlinear. In fact, if a neural network were implemented with only linear transfer functions in its nodes, that would be equivalent to a single linear expression of the outputs in terms of the inputs. Several types of activation functions are available depending on the sought behaviour, with popular choices being radial basis functions, rectified linear unit (ReLU) and notably sigmoids. [85] The output value of a transfer function is usually scaled to produce results between 0 and 1. [86]
Having defined the characteristics of a neuron, it is thus possible to combine these elements to create virtually endless types of architectures. In Figure 3.4, an example of feed forward network (better defined further below) has been provided. The notation used in the definition 3.4 strictly applies to this type of structure, though it can be easily generalized to other types of network, which may integrate more complex elements in their architectures. Moreover, the choice of a linear transfer function in the input layer is motivated: this means that the layer is simply accepting input values from the environment and passing them on to the next layer without further elaboration. [87] [88]
Lastly, it can be noted how neural networks are an extremely flexible model structure, presenting a number of features that can be adjusted in quantity and characteristics. It is therefore possible to identify several different categories of networks based on various (sometimes overlapping) criteria, where the following are some of the most common:

- *Feed forward neural networks* (FFNN) have already been mentioned multiple times without being formally defined, as they represent by all means the most popular type of network. The naming comes from the way information propagates within their structure. Information can only flow "from left to right", that is, from the input layer towards the output layer. This can be clearly seen in Figure 3.4, showing how connections between nodes cannot occur in loops.

- *Radial basis function neural networks* (RBFNN) are feed forward networks that implement radial basis functions, typically Gaussian, as transfer functions. They usually have one hidden layer and are implemented in problems requiring good local approximation. The performance of RBFNN is in fact high in regression tasks even with noisy data, but tends to degrade for larger problems. Training RBFNN can be significantly more expensive than sigmoid-based FFNN and may require the support of additional techniques to simplify the problem.

- *Autoencoders* are a specific class of networks that aim to learn representations of input data, typically to reduce the complexity of a problem. The simplest form of autoencoder is a FFNN, where data are coded and then decoded by passing through one or more hidden layers. The training of an autoencoder is unsupervised: data are first encoded and then reconstructed with a degree of approximation, which is the variable to be minimized. The process of decoding, after encoding, ensures the validity of the trained representation. Different types of encodings can be chosen, depending on the characteristics sought to retain from the input data (e.g. removing noise during image processing).

- *Convolutional neural networks* (CNN) are feed forward networks that use the linear convolution operation within at least one of their layers. These network structures implement the concept of receptive field, which effectively acts as a means to regularize the model (i.e. assigning a certain weight to some chosen nodes). This behaviour emulates the biological processes taking place within the visual cortex. CNN are in fact ideal candidates for applications such as image, video and language processing.

- *Recurrent neural networks* (RNN) essentially break the main rule of feed forward networks, allowing for loops and free flow of information among nodes. These networks are capable of exhibiting a dynamic

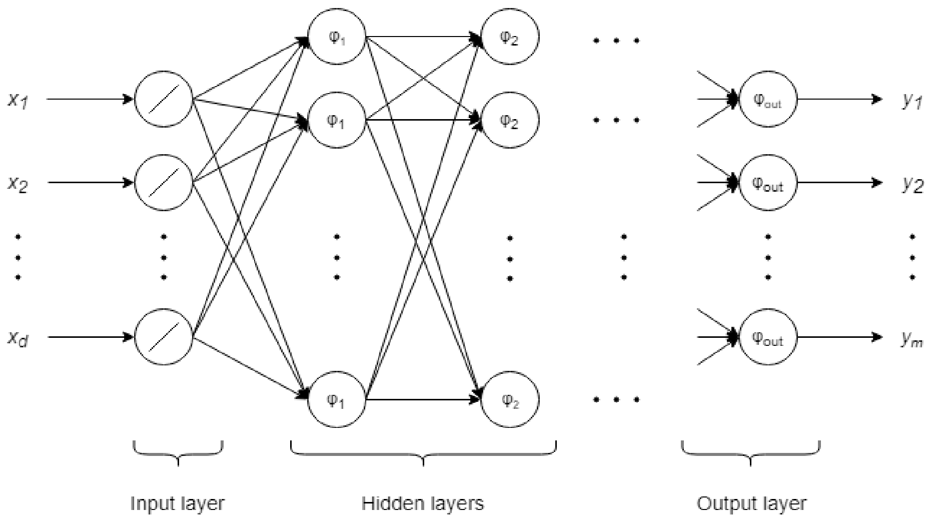


Figure 3.4: Deep feed forward network with d input features and m output variables. Note that a linear activation function has been set for the input layer.

behaviour through a mechanism of internal memory, allowing for different sets of computations to influence each other over time. These characteristics make RNN suitable for problems involving sequential data, such as timeseries.

3.3.2. Neural network training

In a regression problem the training of a model is supervised, which means that the model is presented with sets of input-output data that are known to correspond to each other (i.e. the data are "labeled"). As shown in the previous paragraphs, neural networks are based on a considerable number of adaptable features: number of layers, number of nodes, transfer functions, working principles, weights and biases. Once the other parameters are fixed, training a neural network is the process of solving an optimization problem where weights and biases are the parameters to be tuned. The goal of a training, just like any other surrogate model fitting, is to obtain a network that approximates a certain (unknown) function with arbitrary accuracy. The function that has to be approximated is represented by the set of input-output samples, called "training dataset", which can be obtained in different ways according to the specific application. In more formal terms, a training process can be formulated as:

$$(\underline{\hat{w}}, \underline{\hat{b}}) = argmin(J(\underline{\varepsilon})) \tag{3.5}$$

where the optimal network parameters are obtained from the minimization of a cost function J, which is in turn dependent on the model residual $\underline{\varepsilon}$ (the cumulative error on all training samples). Out of the various types of cost functions that can be implemented, quadratic forms are a common choice, thanks to the capacity of highlighting small deviations (outliers are however also given more weight).
Training algorithms can widely vary in terms of capabilities and complexity. The results, however, can greatly depend on the settings chosen for training and the characteristics of the problem at hand.
One of the simplest training algorithms, which also shows very clearly the principles behind network fitting, is gradient descent. As shown in equation 3.5, the training process aims at finding the optimal combination of weights and biases that minimize a cost function J. In other words, the minimum of J should be found in a domain where the independent variables are $\underline{w}$ and $\underline{b}$. With gradient descent, the algorithm starts its search in an arbitrary point of the domain (usually random). At every iteration the algorithm will take a step in the direction of decrease of the cost function J. In mathematical terms, the gradient descent is based on the following update rule for weights and biases:

$$(\underline{w}, \underline{b})_{t+1} = (\underline{w}, \underline{b})_t - \gamma \nabla J\big((\underline{w}, \underline{b})_t\big) \tag{3.6}$$

According to the above formula, the network's weights and biases are thus updated at each iteration t by a term that is based on the gradient of the cost function ∇J. The factor γ is called step size or learning rate and, in principle, it can be adjusted during training according to the current confidence in the direction of

descent. Smaller steps will typically be taken as the algorithm moves closer to a (local) minimum.
Gradient descent is indicated as a first-order method, in the sense that it employs only first-order derivatives in its functioning. Computation of the gradient, which is in turn the set of partial derivatives of J, is always a crucial point in every training implementation and can be based on a number of methods. One of these is the well-known backpropagation algorithm or, in simple terms, the application of the chain rule to trace back partial derivatives with respect to weights and biases in each layer of a neural network. This algorithm is based on the knowledge of the network's structure and essentially considers the model as a set of nested functions where weights, biases and input parameters are the independent variables.
In the present study, two training algorithms have been implemented: Levenberg-Marquardt and Bayesian regularization. These can effectively be regarded as more complex forms of the gradient descent algorithm and are better described in section 5.3.3.

3.4. Conclusions

In the present chapter, the main theoretical elements about surrogate modeling have been provided, with specific focus on neural networks. The latter model structures have been proven to be universal approximators [89] [90]: a neural network with at least one hidden layer and arbitrary activation functions can approximate a nonlinear function with any degree of accuracy, provided a suitable number of neurons. Moreover, it has been demonstrated that the properties of FFNN can be further exploited by increasing the number of layers. [91] Because of their flexibility, tolerance to error, ability to handle high-dimensional problems and parsimony in the number of parameters [92], neural networks have found application in many problems of function approximation and classification. Based on these characteristics and the evaluation of the problem at hand, neural networks have been selected as the main surrogate model structure for the present investigation. On the other hand, it is not simple to determine a priori the best network architecture for a certain problem. Moreover, training a network on a very complex problem can require substantial time and memory resources. The following chapters will provide details on how these problems have been dealt with, from the choice of network architecture until the evaluation of the trained metamodels. [4] [61] [67] [87]

4

Case study: Aircraft performance and landing

As already stated in the previous chapters, the area of investigation of the project is the application of surrogate modeling on aircraft functions for integration within avionics. The specific field of interest, among the many that are associated with the implementation of embedded functions and models, is that of aircraft performance. Having previously introduced the context of the work and the fundamental theory of metamodeling, the present chapter thus aims to specify the topic of interest by providing information on the chosen case study. This is done by introducing the topic of landing performance and presenting a prototype function, which will be the focus of the reduction study.

4.1. Introduction to landing performance

Landing is the last phase of the flight mission, which aims to bring the aircraft from a steady descent towards the airport until a (close to) complete halt on the runway. The process itself can be divided into various parts, as shown in Figure 4.1, which are characterised by different speeds, configurations and flight attitudes.
The landing phase takes place after the one of approach, which is usually intended as the segment of flight following the descent from cruise altitude. An aircraft conventionally starts its landing at screen height h_{screen}, when crossing an ideal 50 ft obstacle at the runway threshold. After this point, the aircraft is supposed to follow an almost straight descending path, with the engines in idle power.
This program is kept until the pilot eventually pulls up the nose of the aircraft to prepare for touch down (TD). Such manoeuvre is named flare and it serves the function of letting the main landing gear wheels touch the ground first. Moreover, the new aircraft attitude increases the lift and slows down the aircraft, thus alleviating the impact on the runway. Flare is a crucial part of the landing which, if not correctly executed, can lead to cases of hard landing, tail strike or runway overrun.
Once the main landing gear has touched the ground, the aircraft enters the segment of rotation (or derotation) where pitch decreases until the nose gear also touches the ground (nose down, ND). It is in this time frame that the braking actions are initiated, with different delays from TD according to the braking configuration. Spoilers are normally deployed as soon as the main landing gear hits the ground. They contribute to braking by increasing form drag and reducing lift, hence improving the adherence of the wheels to the ground and enhancing their friction characteristics.
The remainder of the landing procedure, called ground or braking roll, takes place between the moment of ND and the point where the aircraft achieves full stop or a conveniently low speed. The sole objective of this phase is to decelerate the aircraft, which is achieved by a combination of mechanical braking, spoilers and thrust reverser action. Special attention is given to when the latter two are used together. Depending on aircraft geometry and attitude, the flow created from thrust reversers around the wing can in fact hinder the efficiency of spoilers. [93] Aircraft landing performance is directly measured by the distance effectively covered during landing. A breakdown of its components is given in Figure 4.1. A shorter landing distance is preferable, as it both cuts down time for airline operations and prevents any circumstance of runway overrun. Further context on this topic can be found in Appendix A, which also includes an introduction to the main governing equations.

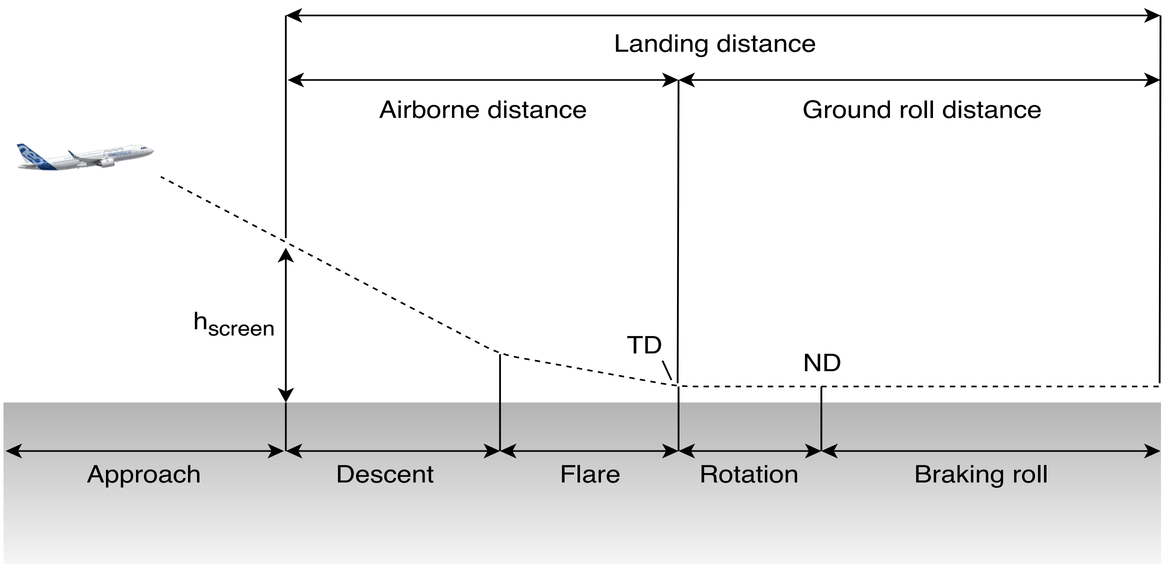


Figure 4.1: Scheme of the landing phase, with breakdown of its components and landing distances.

4.2. CASE STUDY: PROTOTYPE PERFORMANCE FUNCTION

In order to investigate the reduction of aircraft performance functions, a specific case study has been selected and analyzed for the purpose. The target of the study is a prototype function developed by Airbus which is embedded in the ATSU (Air Traffic Services Unit) [94] [95] of single-aisle aircraft. The objective of the function is to calculate the state of contamination of the runway (e.g. water or soil) from the information gained during landing. The idea is to then send the output to the airport traffic services, so that they could forward it to the next aircraft approaching the runway. In this way, pilots of incoming aircraft are enabled to appropriately manage the landing and possibly avoid a situation of runway overrun. The approach offers a reliable, physics-based, objective, real-time and automatically-updated solution to the problem of runway contamination assessment. These characteristics make the application of such a function significantly preferable to an evaluation by simple visual inspection of the runway state.
The function relies on Airbus' aircraft performance models for its internal calculations and is built on multiple levels of models and submodels. Of notable interest for this project, are the aerodynamics and engine performance models, which will be the object of metamodeling in the next chapters.

4.2.1. WORKING PRINCIPLE

In the prototype function, the runway state identification is based on comparison of the landing distance actually covered by the aircraft with the simulated reference distances that would be covered in each of the 6 TALPA (Take-Off And Landing Performance Assessment) runway states: [96]

- TALPA6: DRY
- TALPA5: GOOD/WET
- TALPA4: GOODTOMED/COMPACTED SNOW
- TALPA3: MEDIUM/SNOW
- TALPA2: MEDTOPOOR/WATER SLUSH
- TALPA1: POOR/ICE

The whole process is run in differed time, as all information is gathered from the data recorded in the avionics during the landing phase. By implementing such landing data, it is hence possible to compute the traveled distance d_{real} (one single value) and d_{ref_i} (one value for each of the 6 TALPA states). The runway state is thus conservatively identified as the one corresponding to the worst of the two d_{ref_i} between which the value of d_{real} is estimated. It should be noted how this results from a comparison of distances rather than friction

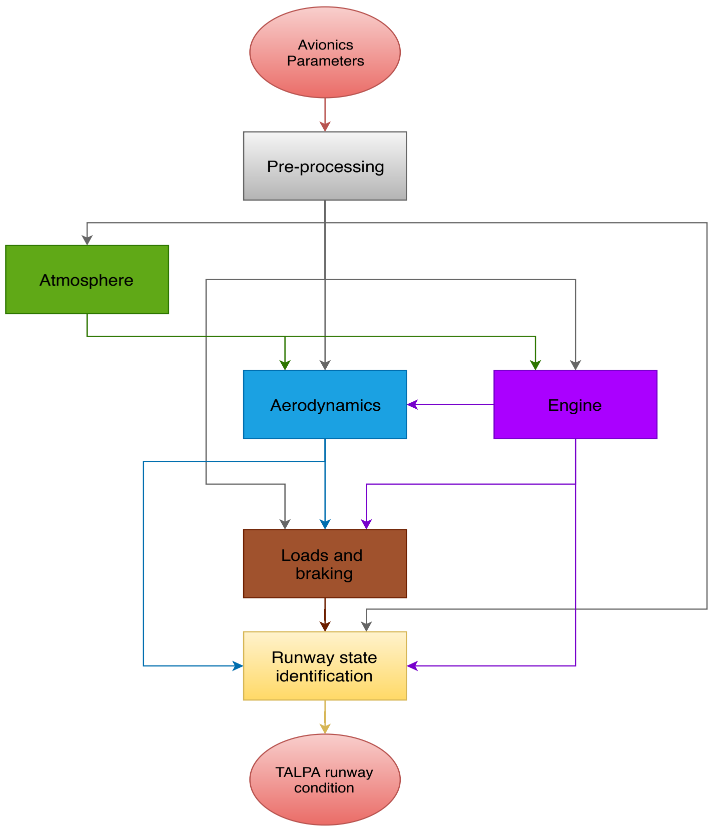


Figure 4.2: Conceptual scheme of the runway condition assessment. The arrows between different modules are meant to indicate a data flow, with different colours according to the module of origin.

characteristics, as the latter remain an unknown of the problem.
An essential aspect of the runway state identification is that the calculation is only active within suitable regions of the landing envelope. This is the case with friction-limited braking phases, which in most cases correspond to when the anti-skid is active and controlling the deceleration. This represents a critical requirement in the computation, as it makes possible to establish a direct physical relationship between friction and normal force. Including non friction-limited braking segments would introduce a physical bias in the calculation, as the braking force would be representative of the delivered torque, rather than the friction between runway and tyres.
The logics to detect such time frames are fairly complex and heavily depend on the specific braking settings and high-lift configuration. As a consequence of the above, the algorithm will not be available when the braking level is not high enough. That is the case, for instance, in an Autobrake MED setting with thrust reversers on (we are not friction-limited). For different reasons the computation won't run in a full-pedal deceleration manoeuvre on dry surface, as in this situation the braking will be limited by torque rather than friction.

4.2.2. GLOBAL FUNCTION STRUCTURE

The calculation of d_{real} and d_{ref_i} requires a number of parameters, which are retrieved within the ATSU environment. While some of these are directly available, others have to be computed from available quantities. For this reason, the core part of the function includes the use of physical models which enable the computation of all necessary parameters.
A conceptual scheme of the function is provided in Figure 4.2. Here the main processing components implemented by the function are provided, along with the data flows exchanged between them. From a global standpoint, the inputs of the function essentially consist in avionics data, while the output (as explained in the previous section) is a runway state expressed according to TALPA standards. Between these two points, a set of physical models and processing units are implemented, including a pre-processing module which is meant to provide a set of coherent and clean data.
The physical computations, as previously mentioned, are largely based on Airbus performance models and

the overall structure is akin to that of a standard aircraft performance model. This includes the modeling of atmospheric, loads, braking, aerodynamics and engine parameters. Out of all of these modules, the two latter ones present a more complex set of features and are of special interest for this study.
The aerodynamics block has its inputs coming from the atmospheric and engine model, as well as additional data retrieved from the avionics. The objective of the model is to compute the global aerodynamic characteristics of the landing, that is the total drag and lift forces acting on the aircraft, along with their corresponding normalized aerodynamic coefficients. On the other hand, the engine model computes the outputs of total thrust, expressed in net and gross values, as well as the projection of the latter on the air-path axes. An additional important aspect that the propulsion module takes care of is the transient state occurring during the activation (or deactivation) of thrust reversers, which cannot be modeled with simple, direct laws due to the dynamic nature of the phenomenon. Both the aerodynamics and the engine model show a degree of complexity higher than the rest of the modules of the function, in that they rely on a number of input parameters, aircraft tables and internal calculations.

4.3. CONCLUSIONS

The aim of the master thesis project presented in this document is to investigate the feasibility of reducing flight performance models by means of neural networks. Within this context, a specific case study has been examined in this chapter. A prototype function has been introduced as a potential candidate for the project, as it presents the essential features of a landing performance function.
Based on the information gathered about the function, an investigation on the chosen topic of neural network metamodeling is thus proposed for this case study. The analysis will tackle the reduction of two specific physical models (or performance submodels) within the function: aerodynamics and engine. The interest in reducing these two models is driven by their considerable degree of complexity, which makes them ideal candidates for a reduction process.
The aim of the analysis will be that of investigating whether a reduction in terms of model complexity, in the light of various proposed solutions, can be reached with acceptable compromise on the approximation capacity of a surrogate model.

5
Methodology

The present chapter outlines the procedure implemented to investigate the chosen research topic. A detailed explanation of the various phases of the work is provided, along with a theoretical and practical justification of the methods and means applied in the process.
The project workflow has been structured in four major phases in order to carry out the set research objectives. Each phase is in turn divided into multiple steps that tackle the lower-level research tasks. An overview of this organization is formally provided in Figure 5.1.
In the first phase, a preliminary analysis is performed to set up the work. A thorough study of the characteristics of the models and their parameters is done, so as to be able to perform the next steps of the work. This process is crucial to get acquainted with the problem and its main features, as well as to point out which aspects need to be taken under special consideration.
The second phase makes use of the results of the latter step, to build the datasets used for training and validation. The data are first retrieved by means of a simulator, then they are sampled to acquire a specified experimental design.
The following phase is the core of the whole work. This includes all training and validation activities to obtain the sought reduced models. The proposed approach consists in building a set of baseline models, which are then compared to another set of models with different architectures, to assess the research objectives of the project.
The final phase consists in assessing the limitations of the work performed. This is done in two different ways: first by comparing the obtained networks with a set of polynomial models, then by analyzing their framework of operability.
The following sections will describe the main body of the work by examining each of the phases presented above. The aim is to elaborate on the specific choices and the methods implemented, along with the rationale behind them.

5.1. Preliminary studies

The first fundamental phase in the research process consists in identifying the critical features of the problem in question. Before performing any investigation on model reduction, in fact, it is paramount to have a clear view of the subject matter, so as to give a proper direction to the metamodeling process.

5.1.1. Study of the problem features

The first part of the project encompasses the acquaintance with the problem and its main aspects of interest. This includes the means implemented in the work, whose features and limitations play a non negligible role in the whole project deployment.

Experimental set-up

The whole set of tools necessary for the activity, which is fully software-based, have been provided by the host company Altran Technologies. Various specific means have been implemented for the different tasks set in the research.
A considerable part of the work, including the initial exploratory study, has been carried out on a Unix system.

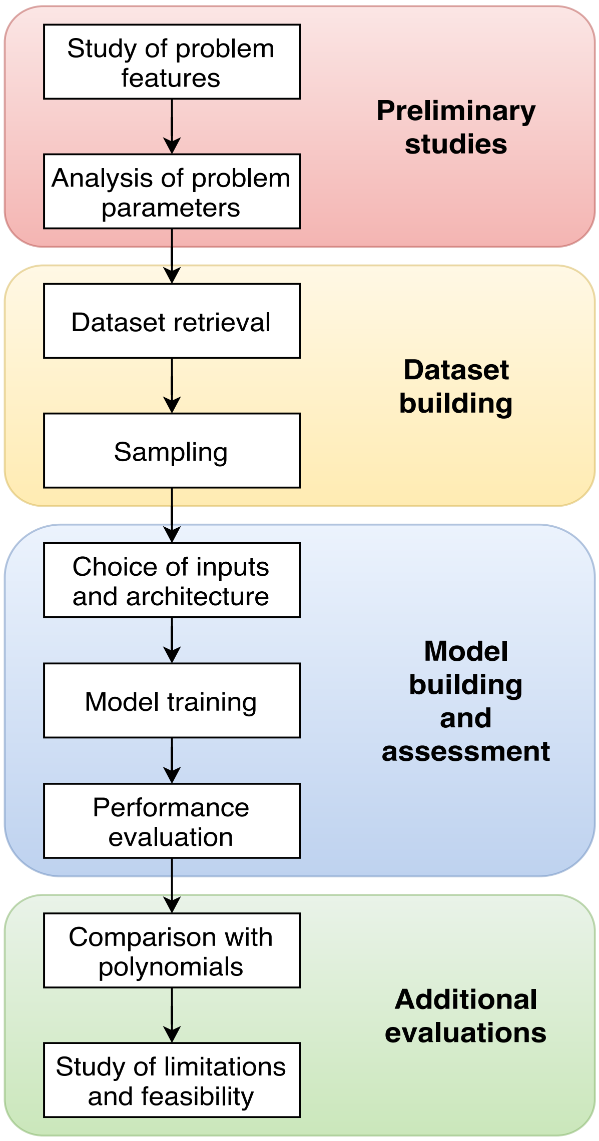

Study of problem features
Analysis of problem parameters
Preliminary studies
Dataset retrieval
Sampling
Dataset building
Choice of inputs and architecture
Model training
Performance evaluation
Model building and assessment
Comparison with polynomials
Study of limitations and feasibility
Additional evaluations

Figure 5.1: Scheme of the research workflow.

In this environment, an Airbus certified software has been implemented to perform landing simulations for generation of training and validation datasets. The working principle is quite straightforward: the user defines all the environmental characteristics (e.g. temperature, Mach, runway condition) for a certain initial point and the software performs a simulation of the full landing in compliance with airworthiness regulations. The output of a calculation is thus a number of time series computed over the course of a landing simulation, one for each relevant parameter of the problem. The software's broad capabilities allow to simulate essentially any situation, even component failures and adverse weather conditions.
Transient effects are accounted for in the simulations, as is the case with the reverse transition in the engine model. Since the modeling of transients is out of the scope of the work, samples including these effects have been cut out from the obtained datasets before the sampling phase.
Matlab has been applied to carry out all remaining tasks of the work. This concerns all processing of the simulator outputs, sampling of the data, feature selection and training and validation of the metamodels. For these objectives, several Matlab default tools including statistical and plotting functions have been used. As for the training of neural networks in particular, the Neural Network Toolbox of Matlab has been selected for the task. Indeed, the same activity could have been carried out by means of other (open source) software packages. This choice has been made because of the reliability, stability and completeness of the toolbox. Matlab is also proposed as the main means for visualization and analysis of the results.
Lastly, it is worth highlighting that industrial data has been implemented in the project, representing a considerable benefit with respect to other works in the field. In fact, as pointed out in the literature review, access to proprietary, consolidated data for research purposes is seldom allowed by aircraft manufacturers.

5.1.2. ANALYSIS OF PROBLEM PARAMETERS

The simulation software is run on a number of landing cases, with the objectives of:

- Gaining further insight into the physics of the problem
- Verifying consistency with the known theory
- Identifying potentially crucial aspects for the creation of datasets
- Analyzing the effect of input and non-input parameters (for the simulator) on the variables of the problem

The study is performed by first identifying a representative landing test case, which is used as a baseline configuration. A set of input parameters are then varied individually from the baseline, to determine the effects of the single variables. The set of inputs whose effects have been investigated in this analysis included:

- Aircraft type
- Engine type
- Runway condition
- Pressure altitude (PA)
- Runway slope
- Outside air temperature (OAT)
- Wind speed
- Center of gravity position (CG)
- Weight
- Calibrated air speed (CAS) at touch down
- Pitch attitude at touch down
- Thrust reverser status

For the most part, the findings of this study are in close agreement with the theory and what could be inferred by an a priori investigation on the problem. Some aspects are however less trivial and are worth mentioning. One point that stands out is the analogy between the effects brought by a change in the runway condition and those due to a change in touch down speed or runway slope. This means that comparable landing conditions can be created by acting on these three input parameters, which will hence be relevant for the next phase of the work.
The type of aircraft and engine, as expected, heavily affect the results of the simulation. After nose down, the aircraft settles to a constant angle of attack for the remainder of the ground roll. Such value depends on the combination of aircraft and engine type set in the simulation. In this context, by aircraft type it is meant a generic model such as A321, while engine type is a certain model produced by an engine manufacturer with a specific power rating. These details are better explained in Figure 5.2, which specifies the common designation of Airbus aircraft models.
The aircraft performs a steady de-rotation as soon as it touches the ground with the main landing gear, with a rate of pitch down that is not affected by the aircraft speed. Concerning the aerodynamic forces, drag and lift start decreasing at TD. As the spoilers are deployed, the value of drag continues to grow, while the lift falls considerably. After ND both variables start to steadily decrease, due to the corresponding deceleration of the aircraft on the runway. Engine thrust forces, on the other hand, are essentially determined by the controlling variables N1 (low pressure spool speed) and EPR (pressure ratio).

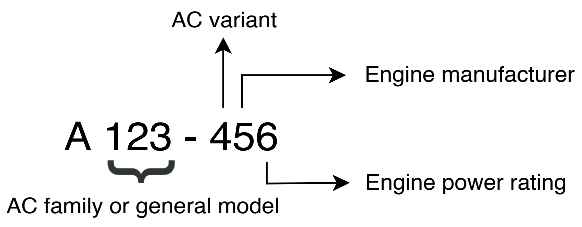


Figure 5.2: Explanation of Airbus aircraft designation, where AC stands for Aircraft.

5.2. DATASET BUILDING

The results of the previous investigations form the foundations of the crucial phase of dataset building. At this point of the project, some important directions are given to the analysis by building a specific dataset from simulations and post-processing. The whole process can be roughly divided into two steps: first the production of a comprehensive dataset from simulator runs, followed by a refinement towards a certain experimental design.

5.2.1. DATASET RETRIEVAL

In order to gather a first, coarse dataset, the simulation software is run on a number of specific landing cases, where each case is identified by different values of the input parameters to the simulation. The simulator is run in sequence on each distinct case by means of a software sovrastructure within the Unix system, which is also responsible for producing the result files.

Data pipeline

At this stage of the work, the problem is defining a suitable set of values for a chosen group of input parameters, so as to obtain the sought dataset. It is not yet essential to have defined a specific design, but rather a sensible choice on the range and number of values provided for each input to the simulator. Since the whole simulation process is cloud-based, the only real constraint here is on the time required for the computation. This time can easily escalate, as the number of landing cases grows with the amount of given input parameters and values for each of them.
The problem that is faced here is explained more intuitively in Figure 5.3. The diagram illustrates a simplified view of the main elements coming into play over the whole course of the reduction process, highlighting the data of interest in each specific point. Four stages have been identified in particular and have been marked in the figure with grey circles. At stage 1, which is the instance analyzed in the present paragraph, a set of inputs and their values are chosen and fed to the simulator. In this way an initial coarse dataset is obtained (stage 2) which is made up of all time series produced via simulation for all landings. The available variables

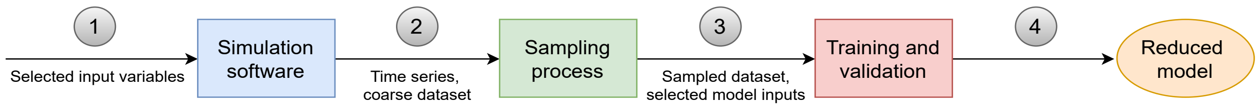


Figure 5.3: Key elements and availability of data throughout the reduction process.
The purpose of the scheme is to show the state and scope of the data at each stage of the workflow, pointing out at the transformations it goes through.

here are those that can be obtained as output of the simulator. Stage 3 is where a refined dataset has been produced, thanks to a sampling process on the coarse dataset aimed at obtaining a specified design of experiments. Data are thus ready to enter a training process to obtain the reduced models. For the purpose, a set of features is selected as the input variables to the trained model. The logic of sampling is thus linked to the characteristics sought for the training and validation datasets and will have an influence on the computational accuracy of the obtained model (stage 4).
As it stands out from Figure 5.3, the characteristics of the data at each point of the process are always affected by the choices made at previous stages. The problem at stake in this paragraph can be hence formulated as: choosing a good set of inputs and values at stage 1, so as to retrieve a dataset at stage 2 that can be suitably sampled to obtain a training dataset (stage 3) with satisfactory characteristics. Solving the problem within the abovementioned time constraints is not trivial and requires a high amount of foreplanning.

Choice of simulation inputs

The approach taken in this work is based on the consideration that the simulator always produces time series results for all output variables. Therefore, any selection on the parameters that will be sampled, can be directly made at stage 2. Moreover, in the view of a later sampling, the only general requirement on the dataset at stage 2 is set with respect to its completeness. As a consequence, the objective of the simulation process will only be that of filling an adequate operation envelope with arbitrary refinement.
Such envelope should reflect the whole set of landing situations that could be presented to a reduced model during its deployment on board. It should be kept in mind however that the targeted envelope corresponds to (a first version of) the domain of training of the reduced models. Hence, the choices made in its creation should not be simply aiming at covering every landing instance, but rather all conditions that could be relevant to those specific models during operation.
In the choice of the input set, the study on the problem parameters described in 5.1 comes useful. Thanks to that, it is also possible to exclude variations that would have resulted in the same effects, thus saving significant computing time both at simulation and postprocessing level. For what concerns the ranges of each input variable, the target of a landing envelope for standard single-aisle aircraft has been followed. The process is performed in an iterative manner to adjust the dataset to the sought characteristics, especially concerning those parameters that are outputs but not inputs to the simulator, and hence require a specific calibration. The resulting setting is thus a compromise between considerations of completeness, refinement and constraints on the problem.
Table 5.1 reports the values set for each selected input to the simulations. Where only one value is given, it means that it was kept fixed through all simulations, hence representing a baseline standard condition that is complemented by the variation of other variables. This is the case with the runway slope and condition, as well as pitch angle at touch down, whose variations can be fairly well covered by those on TD speed. The aircraft model here is intended as a specific combination of aircraft and engine type. The parameter CONF stands for high-lift configuration, qualitatively indicating the position of such devices on the wing. The only configurations allowed for landing are 3 and Full, where the latter effectively corresponds to both flaps and slats being completely extended. For what concerns landing mass and CG position, specific values have been set in order to cover the complete range of values allowed in each situation. The maximum and minimum values of mass are in fact dependent on the maximum landing weight and operating empty weight of each aircraft model. Half of the values have been chosen as overweight cases, thus including emergency situations in the analysis. The allowed CG positions are instead dependent on both the aircraft mass and the high-lift configuration, since these are related to requirements on longitudinal stability and controllability during landing.
It is worth specifying that the Runway Condition parameter is set according to the runway states recognized by TALPA (Dry, Wet, Compacted Snow, Snow, Water Slush and Ice) which are the ones that the prototype function attempts at detecting. For what concerns the Brake mode variable, which defines the intensity of

Input parameter	Values
Aircraft model	10 Airbus single-aisle models
Runway condition	Dry
PA [ft]	-2000, 1000, 4000, 7000, 10000
Runway slope [%]	0
OAT [°C]	-25, -5, 15, 35
Wind [kt]	0
CONF	Full, 3
Gross mass [t]	6 values per aircraft model
CG [%]	3 values per each value of mass
Brake mode	Med
GS at TD [kt]	100, 125, 150, 175, 200
Pitch attitude at TD [°]	2
Thrust reverser status	Inoperative, Max reverse, Idle

Table 5.1: Input parameters and values used in the landing simulations.

the braking action and the way it is distributed over the manoeuvre, this was set to the average condition MED.
While the choices on most values reasonably cover the operational range of the problem, others represent a limitation. The exclusion of a wind component falls in this category, as well as the choice of not including system failure cases. Because of the latter choice, the engine model will be trained on values of half the whole thrust, so as to model only one engine system. In this way, when the model is implemented in the cockpit, the output will be doubled in normal conditions, while it will be taken only once in case of engine failure.
Both limitations have been introduced in order not to excessively complicate the problem but rather work towards a solid baseline dataset, retaining representativity over the whole domain.

5.2.2. SAMPLING

The output of the study described in the previous paragraph is a total of 108000 landing simulations starting from touch down until full stop on the runway. Each simulation bears the whole set of output parameters given in the form of time series sampled at 10 Hz. It is clear that a sampling process is needed to reduce the millions of available points to a dataset of reasonable size.
Matlab has a limited capacity in handling large datasets, thus some preliminary tests have been made to check the training efficiency of the software against varying dataset size. Based on that, a target of around 50000 training points has been set as a reasonable compromise between completeness and memory limitations. An equivalent validation dataset for independent testing is also set as an objective, thus raising the amount of required samples to approximately 100000.
The sampling procedure has multiple goals, which can be summarized as:

1. Remove points that will not be modeled by the reduced models
2. Reduce the size of the dataset
3. Maintain the representativeness of each simulation
4. Obtain a uniform distribution of values

Removal of non-modeled points

The first task of those identified above effectively corresponds to the first step made in this phase of the work. Notably, points lying in three (possibly overlapping) zones have to be removed from each time series:

- Before nose down
- Low speed timezones
- Engine transition from direct to reverse thrust

The first two segments are out of the scope of the function, since the physics of the problem does not allow a suitable assessment of the runway condition. Removing these samples is required since all simulations have been performed with non-zero pitch attitudes at touch down and are recorded until full stop on the runway. The third point concerns all cases of Max or Idle Reverse, where there is a transition of the engine from a direct to reverse law. During this transient, the thrust does not follow the usual relationships that link engine operation and power output. That is why specific laws are used in these phases, which are modeled individually for each engine model. These samples are removed here since feedforward networks are not capable of modeling transients and are trained on static samples rather than time series.

First dataset reduction

Before implementing an effective sampling process towards a specific experimental design, a further reduction on the dataset is performed. This is necessary, since the number of points is still very high and may make the sampling algorithm practically inoperable due to computing time. The reduction is made by retaining 10 samples from each simulation, so as to guarantee their representativeness (point 2 of the sampling goals). These samples are selected uniformly according to the values of GS (ground speed). The choice is not made on specific values of GS, but rather by dividing the (remaining parts of) time series in 10 sections (uniform on GS) and taking one random point from each section. This is done to avoid redundancies of points from time series that have similar inputs or happen to result in the same effects.
The choice of GS as a reference rather than the time variable is due to the fact that, for the problem in question, time does not have a particular meaning. The unfolding of the landing manoeuvre does not follow regular timely steps, but rather it changes according to how fast the aircraft manages to decelerate on the ground.

Sampling procedure

The main sampling phase aims to obtain a dataset as complete and uniform as possible, so as to enable a training and validation of metamodels that are representative of their corresponding originals. A stratified sampling approach [97] based on equal allocations on the discrete variables has been implemented. The choice is deemed to serve the following purposes:

- Provided that previous steps of the process have retained full coverage of the relevant domain, this sampling approach retains the full observed range of the variables.
- Areas of the domain with high redundancy of sample points are reduced in weight. The presence of these areas (e.g. low speed) is due to the commonalities between simulations with similar starting parameters.
- A space filling design is sought to produce uniformity in an N-dimensional sense, in order to implement a balanced dataset in the training process. N-dimensional uniformity is much harder to achieve than 1-dimensional uniformity. Moreover, some variables of the problem cannot be controlled during the simulations and their effect is only mitigated.
- The natural tendency of the physical phenomenon, represented by the patterns of the continuous variables, is retained. This ensures higher metamodel performance in those parts of the operation envelope where the model is likely to work more often.

As a result of the sampling process, the dataset distributions are essentially unaffected, except being scaled down by a factor 10. A constraint was introduced on some discrete variables to make sure that the different aircraft types are equally represented in the dataset, in order to enhance the portability of the surrogate models. Additional details on the rationale behind the chosen sampling procedure and an overview of the results is provided in Appendix B.

Training and validation datasets

The sampling process was performed separately on four pieces of the dataset retrieved from simulations, in order not to overload the postprocessing phase. Once the procedure is completed, the four parts are rejoined together and then randomly split into two subsets: one for training and the other for independent validation. The two sets possess analogous characteristics and the distributions are conform to the ones they have been sampled from. The number of points in each dataset is 54000, hence achieving the goal set earlier to perform efficient model trainings with Matlab.

To ensure that a good randomization had been obtained, a set of model trainings was performed with different draws for the two datasets. Having obtained no remarkable difference in the results of such trainings, it can be concluded that a representative splitting of the datasets has been attained.

5.3. MODEL BUILDING AND ASSESSMENT

Once the necessary data has been collected and reduced to an acceptable amount, it can be eventually implemented in a model building process. The present section deals with the choice of various elements that are relevant to this procedure: the input features, model structures, training algorithms and the criteria to assess the quality of the trained models.

5.3.1. SELECTION OF INPUT PARAMETERS

According to the literature on the subject, various approaches can be taken in the choice of a set of input variables to a reduced model. Most of the works in the field seem to rely on engineering judgement, based on the knowledge of the problem. However, more structured and complex methods are available and are generally identified under the category of feature selection and extraction techniques. [63]
The issue plays a central role in this project, as the number of input features has a direct influence on the complexity of the model. The problem is thus finding a minimal set of inputs that would still sufficiently describe the essential features of the model being reduced, concurrently avoiding to excessively hinder its accuracy. The only hard constraint on the process is given by the availability of data within the aircraft avionics environment, where the models are implemented. Moreover, the data retrieved via the simulator in 5.2.1 already include all potentially relevant parameters and hence only require to be selected before model training.

Correlation study

A statistical analysis has been performed to help in this selection process. A Pearson correlation coefficient is computed for each possible pair of parameters available at this stage of the work. The data obtained from the processes described in 5.2.2 has been implemented for the purpose. The Pearson coefficient for two generic variables x and y is defined as:

$$\rho_{xy} = \frac{cov(x,y)}{\sigma_x \sigma_y} \tag{5.1}$$

where the covariance and standard deviation of the variable x can be expressed as

$$cov(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{5.2}$$

$$\sigma_x = \sqrt{var(x)} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{5.3}$$

These definitions are valid for a population of n samples, iterated on by the index i, with mean values indicated as $\bar{x}$ and $\bar{y}$. In a feature selection process, such correlation coefficient can be used for two different purposes. On one hand, redundant inputs can be screened out based on high mutual correlation, as they would provide similar information to the model. On the other hand, the correlation between potential input variables and the (defined) output parameters can provide important information on the relevance of each input feature. The two aspects are thus taken into account when analysing the correlation matrix displayed in Table 5.2.
With respect to the latter point, it is notably useful to inspect the values in the last four columns, where the coefficients are referred to the output variables of the models. Coefficients are normalized between -1 and 1, thus a value close to the two extremes hints a strong correlation or anticorrelation between the parameters. A value close to zero should be interpreted as a weak interdependence, thus the parameter may not represent an added value for the reduced model.
Overall, the study does not produce substantial discoveries, but rather proves what the known physics and features of the problem already suggest. Some results might however not be intuitive, as is the case with the correlation indexes found for the angle of attack. This is especially true for values of correlation with the aerodynamic forces, which are normally expected to be substantially affected by the angle of attack. The explanation for this behaviour is found in 5.1.2, where the angle of attack was shown to follow a linear trend between TD and ND and then set to a constant value for the remainder of the roll. Such value only depends on structural features of the aircraft, which are here represented by the variables of aircraft and engine type. As a

	Alpha	GS	Mach	N1	Weight	CG	Pressure	Conf	OAT	Reverse	AC type	Eng type	Drag	Lift	Xprop	Zprop
Alpha	1	0.0211	0.0211	-0.0683	-0.1763	0.1039	-2.88E-06	-9.89E-19	2.91E-19	0.000803	-0.189	0.2571	-0.0612	-0.1386	-0.0535	-6.69E-05
GS	0.0211	1	0.9951	0.2472	0.0338	0.0083	-0.0299	0.0113	0.0163	-0.2582	0.0069	-0.0463	0.9163	0.3663	-0.0487	0.384
Mach	0.0211	0.9951	1	0.2408	0.034	0.0082	-0.0301	0.0115	-0.0736	-0.2572	0.0069	-0.046	0.9215	0.3673	-0.052	0.3831
N1	-0.0683	0.2472	0.2408	1	0.0091	0.0036	-0.0533	-0.0031	0.055	0.3813	-0.0162	-0.1472	0.134	0.1306	-0.7181	-0.4418
Weight	-0.1763	0.0338	0.034	0.0091	1	0.1379	2.77E-08	0	1.23E-19	-0.0027	0.3782	0.0953	0.0555	0.1445	-0.0235	0.0096
CG	0.1039	0.0083	0.0082	0.0036	0.1379	1	9.64E-06	-0.0336	-1.57E-18	0.0026	-0.0993	-0.0384	0.0063	0.0893	0.0053	0.0069
Pressure	-2.88E-06	-0.0299	-0.0301	-0.0533	2.77E-08	9.64E-06	1	-2.98E-07	0.000233	0.0057	-5.12E-06	-1.18E-05	0.128	0.0317	-0.104	-0.0275
Conf	-9.89E-19	0.0113	0.0115	-0.0031	0	-0.0336	-2.98E-07	1	0	-0.0015	1.55E-17	-1.38E-18	-0.095	-0.2357	-0.0055	-0.0021
OAT	2.91E-19	0.0163	-0.0736	0.055	1.23E-19	-1.57E-18	0.000233	0	1	-0.0016	-1.31E-18	3.53E-19	-0.0651	-0.0183	0.0338	-0.000555
Reverse	0.000803	-0.2582	-0.2572	0.3813	-0.0027	0.0026	0.0057	-0.0015	-0.0016	1	-0.0085	-0.0132	-0.2924	-0.1319	-0.7101	-0.8215
AC type	-0.189	0.0069	0.0069	-0.0162	0.3782	-0.0993	-5.12E-06	1.55E-17	-1.31E-18	-0.0085	1	0.2916	0.0198	0.2153	-0.0705	-0.0299
Eng type	0.2571	-0.0463	-0.046	-0.1472	0.0953	-0.0384	-1.18E-05	-1.38E-18	3.53E-19	-0.0132	0.2916	1	-0.1008	0.0259	-0.1664	-0.0779
Drag	-0.0612	0.9163	0.9215	0.134	0.0555	0.0063	0.128	-0.095	-0.0651	-0.2924	0.0198	-0.1008	1	0.4344	0.06	0.4546
Lift	-0.1386	0.3663	0.3673	0.1306	0.1445	0.0893	0.0317	-0.2357	-0.0183	-0.1319	0.2153	0.0259	0.4344	1	-0.0361	0.2048
Xprop	-0.0535	-0.0487	-0.052	-0.7181	-0.0235	0.0053	-0.104	-0.0055	0.0338	-0.7101	-0.0705	-0.1664	0.06	-0.0361	1	0.8486
Zprop	-6.69E-05	0.384	0.3831	-0.4418	0.0096	0.0069	-0.0275	-0.0021	-0.000555	-0.8215	-0.0299	-0.0779	0.4546	0.2048	0.8486	1

Table 5.2: Table of Pearson correlation coefficients computed for all parameter pairs available after the sampling phase. Different levels of correlation and anticorrelation are highlighted with varying intensities of green and red respectively.

Input parameter	Baseline set	Aerodynamics set	Engine set
Angle of attack α [°]	✓		
Mach [-]	✓	✓	✓
N1 [%]	✓	✓	✓
Gross mass m [kg]	✓	✓	
CG position [%]	✓		
OAT [°C]	✓	✓	✓
Air pressure [Pa]	✓	✓	✓
CONF [-]	✓	✓	
Rev [boolean]	✓	✓	✓
Aircraft type [-]	✓	✓	✓
Engine type [-]	✓	✓	✓

Table 5.3: Input features chosen for the work: a common baseline set and two reduced sets specifically selected for each model.

consequence, lower correlation is found with the aerodynamic forces, since these are not primarily affected (in a single flight) by changes in angle of attack but rather by other variables.

Input selection

The approach taken in the selection of input features relies on one hand on engineering judgement coming from knowledge of the problem, and on the other hand on the results of the above correlation study. Two input sets are defined for each model: a baseline set of variables, that is implemented in most of the analyses, and a reduced secondary one whose performance is compared to the baseline. In the first case, the inputs are essentially taken as a subset of those of the original models. Since the aerodynamics and engine model are coupled by the effect of spoilers, a common set of inputs can be taken, exhaustively describing the problem characteristics in both cases. For what concerns the secondary sets, these are selected separately for each model, taking into account their specific features. A summary of the input variables chosen in either case is provided in Table 5.3. An important clarification must be made here with reference to Figure 5.2, on the categorical parameters of aircraft and engine type. Here and in the remainder of the document, an aircraft type will be intended as a precise aircraft variant within a general family or model of aircraft. This is necessary, as different variants often have very different geometric and mass characteristics. As for the engine type, this will refer to the manufacturer of the specific engine model.
As a last consideration, it should be noted that the engine performance quantities of N1 and Reverse have been considered for one engine only. This follows the choices explained in 5.2.1, where it was stated that cases of system failure have not been taken into account in the analysis. Because of that, the problem is studied in ideally symmetrical conditions, and it is assumed that the same aerodynamic and propulsive phenomena are taking place on both sides of the aircraft.

5.3.2. NEURAL NETWORK STRUCTURES

The main metamodel structure implemented in this project is feedforward neural networks. These, however, can be built in a variety of specific architectures that adapt at best to the characteristics of each specific problem. In order to give a coherent organisation to the work, it was chosen to establish a baseline network structure on which variations are then applied in the various studies of interest.

Baseline model structure

Based on their property of universal approximators [89] [90] single hidden layer networks can be safely chosen as a starting point for the analysis. Linear transfer functions are set in the input and output layers, while a hyperbolic tangent function is chosen for the hidden layer. Such sigmoid function is essentially a logistic function scaled and shifted to output values between -1 and 1. As for bias coefficients, these are set both in the hidden and output layers. A last aspect is that of the number of neurons, which is effectively dictated for input and output layers by the number of parameters. The choice is less trivial in the case of hidden layers, for which various rules of thumb exist in the literature on the topic. A number of 20 neurons is established as a baseline for the hidden layer, reflecting the standard structural requirements of aircraft embedded models. Clearly, since some of the choices specified above are purely arbitrary, the selected baseline model only represents a starting point for the analysis. As it will be described in the next chapter, variations on the model

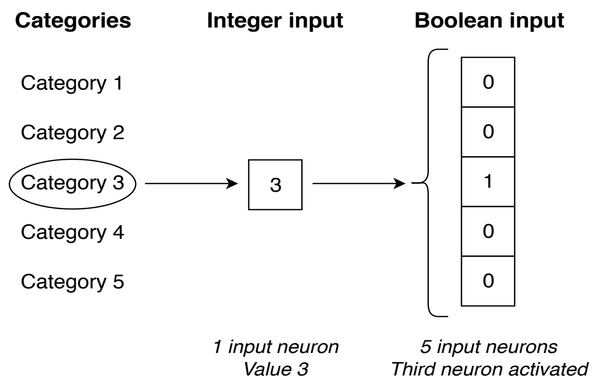


Figure 5.4: Two approaches to dealing with categorical inputs.
In this example, a variable is provided with 5 different categories. A corresponding input can be fed to a network via 1) a single neuron with integer value 3 or 2) a set of 5 boolean neurons where the third one is activated.

structure (number of layers and neurons) will be tested to assess the optimal characteristics in terms of memory and accuracy performance.

Categorical input variables

A further aspect deserving particular attention concerns the way input features are provided to a reduced model. In the present analysis, it is possible to distinguish two kinds of input features: numerical and categorical. The former refers to actual data such as aircraft mass or speed, which are normally related to a unit of measurement and thus express a quantity that can be directly provided to a model. The latter kind of inputs, instead, consists in all parameters that rather express non-quantifiable information (a quality) and hence require a specific treatment. [88]
Examples of the latter, are the four last variables in Table 5.3, although there are important differences between them. It is clear nonetheless, how neither of these parameters refer to a scale, as they represent different situations instead of quantities. In this case the high-lift configuration and reverse variable can only represent two states each, thus they are simply provided to the model in the form of a single binary input. The situation is more complex for the parameters of aircraft and engine type, as they are supposed to represent 3 and 4 categories respectively.
Two solutions, of which an example is provided in Figure 5.4, are available to solve the problem. On the same lines of what has been done with binary variables, a first approach consists in providing the model with as many boolean inputs as the number of categories that the parameter should represent. In this way each category can be expressed by activating single boolean inputs with a 1, keeping the remaining inputs (of that specific categorical variable) to a null value. This is often referred to as one-hot encoding.
The second proposed solution only requires one model input per categorical feature. In this case each category is assigned to a specific numerical value (usually ordered integers), which is then provided to the model via the single input feature. This approach is clearly less accurate than the previous one as it implies a numerical meaning in the scale and order of the chosen values, which may provide misleading information to the model. This might in turn make it harder to fit on the data or simply lead to learning fictitious patterns.
Despite this flaw, the latter solution is still preferred for its efficiency in the amount of inputs, which is an aspect of strong interest with regards to the objectives of the project. Arbitrarily ordered integers for single categorical input features are thus implemented throughout the investigation. However, considering the expected benefits in terms of accuracy in the use of multiple binary inputs, this solution is also explored in the next chapter.

5.3.3. TRAINING ALGORITHMS

Following a set of preliminary tests on various training algorithms, two of them were found suitable for the problem. The first method is the well-known Levenberg-Marquardt optimization algorithm [98] which is based on the update rule:

$$\underline{w}_{t+1} = \underline{w}_t - (J^T J + \mu I)^{-1} J^T \underline{\varepsilon} \tag{5.4}$$

where $\underline{w}_t$ is the vector of the network weights and biases at time step t, $\underline{\varepsilon}$ is the output error vector, J its Jacobian with respect to all the weights (normally computed by standard backpropagation) and I the identity

matrix. The quantity μ in the formula is called damping factor. This coefficient is adapted during training according to the current level of confidence in the step direction, adjusting the algorithm between a Gauss-Newton method (small μ) and a first order gradient descent approach (large values of μ). This is considered as a pseudo-second-order algorithm, as the Hessian matrix H (which holds the second derivatives of the objective function) is approximated by the term $J^T J$.
The method attempts at minimizing an objective function assumed to be a combination of squared deviations in the form:

$$E = \underline{\varepsilon}^T \underline{\varepsilon} \tag{5.5}$$

Levenberg-Marquardt has been proven as a fast and reliable method for most small to medium sized problems. Despite the present work not making exception, the second training method implemented still resulted in consistently better results.
Bayesian Regularization [99][100] applies the same rules as Levenberg-Marquardt in the update of the network parameters. The fundamental difference, however, is that the objective function is now a linear combination of errors and weights:

$$E = \beta E_D + \alpha E_W = \beta(\underline{\varepsilon}^T \underline{\varepsilon}) + \alpha(\underline{w}^T \underline{w}) \tag{5.6}$$

Here the regularization parameters α and β are computed by applying Bayes' theorem on conditional probability and assuming Gaussian distributions for all parameters in the problem:

$$\alpha = \frac{\gamma}{2E_W} \tag{5.7}$$

$$\beta = \frac{n_T - \gamma}{2E_D} \tag{5.8}$$

with n_T the number of training samples and γ the "effective number of parameters" defined as

$$\gamma = N_P - 2\alpha(H^{-1})^T \tag{5.9}$$

The latter coefficient, defined against the number N_P of parameters (weights and biases) in the model and the Hessian matrix H of the error function, is a measure of the current efficiency in the use of the chosen parameters in the model. In essence, the algorithm seeks to push the values of unnecessary parameters towards zero, in what is commonly called a "pruning" mechanism. In this way the model is prevented from overfitting on the data, without the need of extensively cross-checking training performance with a validation set. To stop the process, in fact, the convergence of the cost function E is normally taken as a reference point.
As mentioned before, Bayesian Regularization provided better results in the vast majority of network trainings. For this reason the method was implemented in nearly all studies as a benchmark, comparing the results with those of Levenberg-Marquardt only in very specific instances. Indeed this improved performance came with a cost, as Bayesian Regularization would take on average approximately three times the training time required with Levenberg-Marquardt.

5.3.4. TRAINING SETTINGS AND STOPPING CRITERIA

Having defined the surrogate model structures and fitting algorithms, there are several parameters that have to be set in order to perform each training. These have been adjusted through repeated tests to find consistency and good accuracy of results and have been kept constant throughout the whole analysis.
Trainings were repeated 10 times per single model architecture, retaining the best performance in terms of mean absolute error on the validation dataset. This was done to mitigate the elements of randomness introduced in the every training: the starting point of the optimization and the extraction of a validation and test subsets from the training dataset built in 5.2.2. The validation dataset (which is clearly independent of that obtained in 5.2.2) is used in the Levenberg-Marquardt algorithm as one of the stopping criteria, by cross-checking performance during training in order to detect overfitting. The test dataset, instead, provides an independent measure of the model generalization achieved during and after training with both algorithms.
An essential point of the process is the normalization of inputs, since those selected in 5.3.1 range over very different magnitudes. The purpose here is served by a mapminmax function, scaling all inputs between -1 and 1 before training.
More numerical information about the specific training settings is reported in Table 5.4. These focus in particular on partitioning of the training set, stopping criteria and handling of the damping factor μ. Each setting has a correspondence in the Matlab environment where the models were built, which provides reliable tools to manage each specific aspect of the process. In the table, MSE stands for Mean Squared Error.

Setting	Levenberg-Marquardt	Bayesian Regularization
Validation subset size	20%	-
Test subset size	10%	20%
Training performance function	MSE	MSE
Maximum training epochs	1000	1000
Performance goal	0	0
Maximum validation failures	6	-
Minimum performance gradient	1e-7	1e-7
Initial damping factor μ	0.001	0.005
μ increase factor	10	10
μ decrease factor	0.1	0.1
Maximum allowed μ	1e10	1e10

Table 5.4: Training settings implemented with the Levenberg-Marquardt and Bayesian Regularization algorithms.

5.3.5. PERFORMANCE EVALUATION

One essential feature of the study is the method of assessment of the obtained models. As previously discussed, this has to take into account both the accuracy of the metamodel and the degree of storage requirements achieved.
For what concerns the accuracy, various indexes can be computed on the training and validation datasets, such as mean absolute error (MAE) and root mean square error (RMSE). These are defined as:

$$MAE = \frac{\sum_{i=1}^{n} |\varepsilon_i|}{n} \tag{5.10}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} {\varepsilon_i}^2}{n}} \tag{5.11}$$

where ε_i is the error on sample i and n the number of given samples. Both indexes are provided in the same unit as the output variables, thus representing a convenient way to understand the physical nature of the results. The value of MAE is sometimes chosen as the main accuracy index due to its easier interpretation and reduced sensitivity to outliers, compared to other coefficients such as RMSE. This is due to the intrinsic nature of MAE, in which all errors have a proportionally weighted impact on the end result. [101]
The benefit of having performance indexes in the same unit as the output, comes unfortunately with the disadvantage that results from different models and datasets cannot be directly compared. In fact, absolute measurements of the error on engine data have a different meaning than those computed on aerodynamic data. Even more importantly, the issue is present when comparing results for the same model but with different training or validation datasets. A relative quantity is hence introduced as the global accuracy index for this study, the normalized MAE:

$$NMAE = \frac{MAE}{\bar{y}} \tag{5.12}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} |y_i|}{n} \tag{5.13}$$

which is merely the mean absolute error over a normalization coefficient $\bar{y}$, here chosen as the average of absolute target values in the output dataset. In the present report, this index will be provided in percentage terms.
A last point worth mentioning is that the evaluation of model accuracy on the validation dataset is preferred over that on the training dataset. The former, in fact, provides an independent measurement of the model accuracy, which is not given by the latter.
The second aspect in the evaluation of the surrogate models is that of reduction of model complexity. In the work of Bondouy [15] on the selection of surrogates for multilevel structures, both memory size and computational time are implemented as performance measurements. Since the current project is a feasibility study focusing on several aspects of the reduction of aircraft models, it has been chosen to only retain the memory size as a second performance index, provided in terms of numerical coefficients within the model. Such

approach is also implemented by Corman [25] in the assessment of metamodel complexity. This is deemed to be a good compromise, as the computational time is expected to follow a comparable reduction to that observed in memory size.

5.4. ADDITIONAL EVALUATIONS

The last phase of the project consists in establishing the validity of the work done through further evaluations. As a first means of assessment, it was chosen to compare the performance of the obtained networks to those of alternative metamodel structures. This is common practice in the field and various types of surrogate model are available for the job. In this case, polynomial models have been selected for comparison.
Moreover, in order to provide a practical criterion to evaluate the results of the work, an additional study was carried out to assess the impact of the integration of the trained model architectures on the performance of aircraft functions.

5.4.1. POLYNOMIAL MODELS

Multivariate polynomial models of low degree (response surfaces [25]) have been implemented for their ease of application and accuracy in local approximation. Some tests were run on Matlab to assess the practical limitations of the process, which led to the decision of training models of up to degree 3. Both training and validation dataset obtained in 5.2.2 were scaled down by a factor of approximately 2.5, following the sampling method described in the same section. Each dataset thus featured slightly less than 21000 samples before fitting.
The polynomial models were built for specific key cases, as a benchmark for comparison with the neural network models. The specific rationale behind the construction of these models is provided in Appendix D. Results in terms of accuracy and required coefficients will be provided in the next chapter in parallel to those of the network models.

5.4.2. FEASIBILITY STUDY

As explained in the beginning of the document, the aim of the project is to investigate the possibility of introducing neural network-reduced models within aircraft functions. The methods and tools presented so far are all aimed at producing a set of surrogate models that could represent a valid alternative to the original functional modules. What is still missing, in order to be able to answer this project's research question, is a framework against which the obtained models can be assessed by means of the indexes presented in 5.3.5.
With this objective in mind, a study is performed on the prototype function to determine how the accuracy of a reduced model could impact the final assessment of runway contamination. As shown in Chapter 4, the computation of runway condition is achieved by comparing a d_{real} with 6 reference distances d_{ref} that correspond to the TALPA levels of contamination. The critical aspect in this method is that the 6 d_{ref}, which will eventually be computed by implementing the reduced models, should be accurate enough not to result in the d_{real} being on the wrong side of a d_{ref_i}.
This concept is better explained by looking at Figure 5.5, where the hypothetical result of a runway condition analysis is represented in an intuitive way. As shown, the function would normally compute a set of d_{ref} (on the left) and assess the contamination level as the lower of the two closest to d_{real}. In this case, however, a set of estimates d_{ref}^{est} is introduced to represent the values that would be computed (with lower accuracy) by implementing surrogate models. A positive or negative difference between the two values, Δd_{ref_i}, is therefore normally expected. In this example, the contamination assessment would lead to a wrong result, as $d_{ref_i}^{est}$ is higher than both d_{ref_i} and d_{real}, hence resulting in a contamination level corresponding to $d_{ref_{i-1}}$ instead of d_{ref_i}.
Moreover, a worse case could be thought of, where $d_{ref_i}^{est}$ and d_{real} would be symmetrically placed on the other side of d_{ref_i}. In such situation the assessment would not only be wrong, but also result in an underestimation of the runway contamination level.
Based on the above, the requirement on the accuracy of runway condition assessment can thus be stated:

$$\Delta d_{ref_i} < \Delta d_{real_i} \tag{5.14}$$

where the two quantities are given as

$$\Delta d_{ref_i} = \left| d_{ref_i}^{est} - d_{ref_i} \right| \tag{5.15}$$

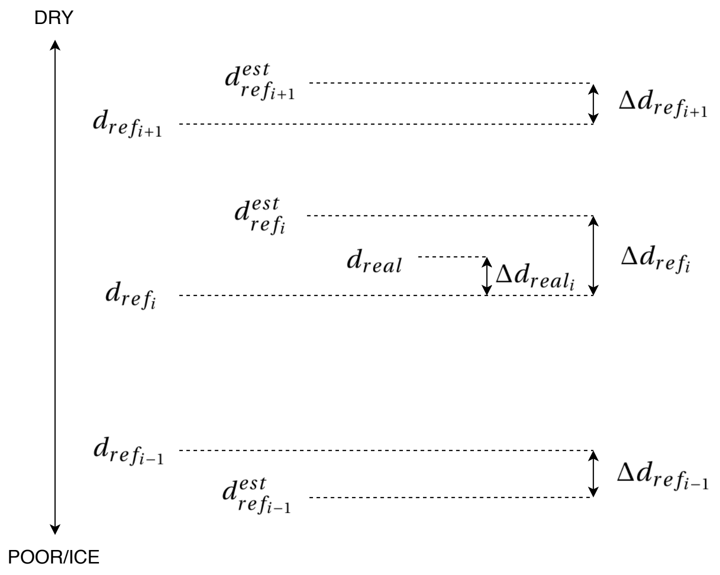


Figure 5.5: Hypothetical result of a runway condition analysis. The reported variables are the actual reference distances d_{ref_i}, the same distances as computed with reduced models d^{est}_{ref} and the actual traveled distance d_{real}. The absolute deviations Δd_{ref_i} and Δd_{real_i} are also shown. In this example the assessment leads to a wrong result, since $d^{est}_{ref_i}$ is higher than d_{ref_i} and d_{real}.

$$\Delta d_{real_i} = \left| d_{real} - d_{ref_i} \right| \tag{5.16}$$

This expression provides a good indication on what characteristics should be sought in the reduced models under examination. It is evident that the parameter on which the analysis needs to be focused is Δd_{ref_i}, which should be kept as low as possible.
A target relative accuracy can thus be set here, with a tentative value of 0.5%. This figure will be referred to a normalized measurement of the error on d_{ref_i}, defined as:

$$\Delta d^{\%}_{ref_i} = 100 \cdot \frac{\Delta d_{ref_i}}{d_{ref_i}} \tag{5.17}$$

Having set this objective, the study is focused on finding the relationship between the above error and the accuracy of the reduced aerodynamic and engine models. In this context, the variables of interest will be aircraft drag and X_{PROP}, as these have a direct impact on the calculation of the reference landing distances.

Numerical assessment

The approach taken here consists in a numerical assessment of the errors made on d_{ref_i} within the working domain of the surrogate models. In order to give a clear idea of the process, the sequence of steps is shown in Figure 5.6. A further breakdown of the algorithm, which was implemented in Matlab, is given in the following:

1. Data for the analysis are retrieved from a set of landings which is analogous to those employed in 5.1. The essential difference with the study of the previous chapter is that data will be considered as full time series, rather than single sample points.

2. The algorithm starts by considering each landing timeseries separately. The first step in this sense is to compute d_{ref_i} with the available data, which are considered "correct" as they're output of simulations.

3. Having set a specific landing and computed the landing distance, steps 3 to 6 aim to compute an equivalent distance but with "real" data. In order to do that, an artificial error distribution is introduced in the values of D and X_{PROP}, as if surrogate models were used to produce them. Modeling errors, however, can have very different distributions (which could be generally described by their average and standard deviation) hence the need to establish a process to simulate this aspect.
   The approach taken here will assume normal error distributions, which is deemed as a good compromise based on the initial study performed in 5.1.2. A specific combination of NMAE values for D and

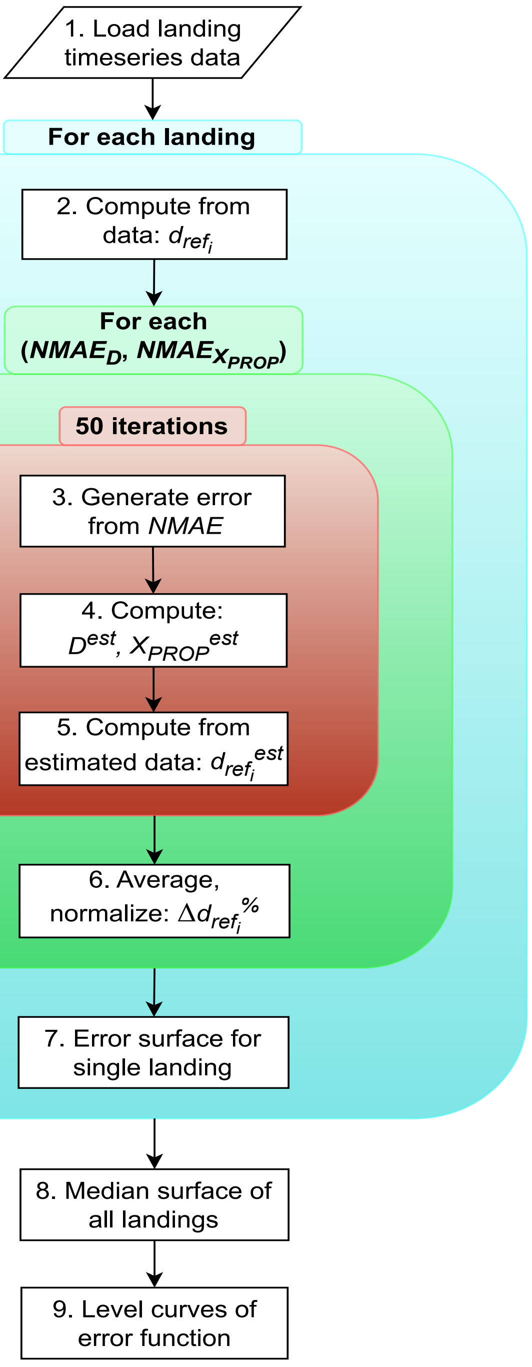

1. Load landing timeseries data
For each landing
2. Compute from data: d_{ref_i}
For each ($NMAE_D$, $NMAE_{X_{PROP}}$)
50 iterations
3. Generate error from $NMAE$
4. Compute: D^{est}, $X_{PROP}{}^{est}$
5. Compute from estimated data: $d_{ref_i}{}^{est}$
6. Average, normalize: $\Delta d_{ref_i}{}^{\%}$
7. Error surface for single landing
8. Median surface of all landings
9. Level curves of error function

Figure 5.6: Workflow of the numerical feasibility study.

X_{PROP} is set at this step of the algorithm. This accuracy measure is chosen as it allows to compare results from different instances and is extensively used for this goal in the remainder of the work. An error distribution with zero average is thus randomly generated for each variable, having set the standard deviation from the relationship [102] for Gaussian distributions:

$$\sigma = \sqrt{\frac{\pi}{2}} MAE \tag{5.18}$$

4. The error distribution generated for a specific combination of NMAE on D and X_{PROP} is applied on the timeseries of both variables. Thus the estimated data D^{est} and X^{est}_{PROP} are obtained.

5. The computation of step 2 is executed again, but this time on the perturbed timeseries, resulting in an estimated reference distance $d^{est}_{ref_i}$. Steps 3-5 are repeated 50 times, computing an equivalent number of approximated reference distances.

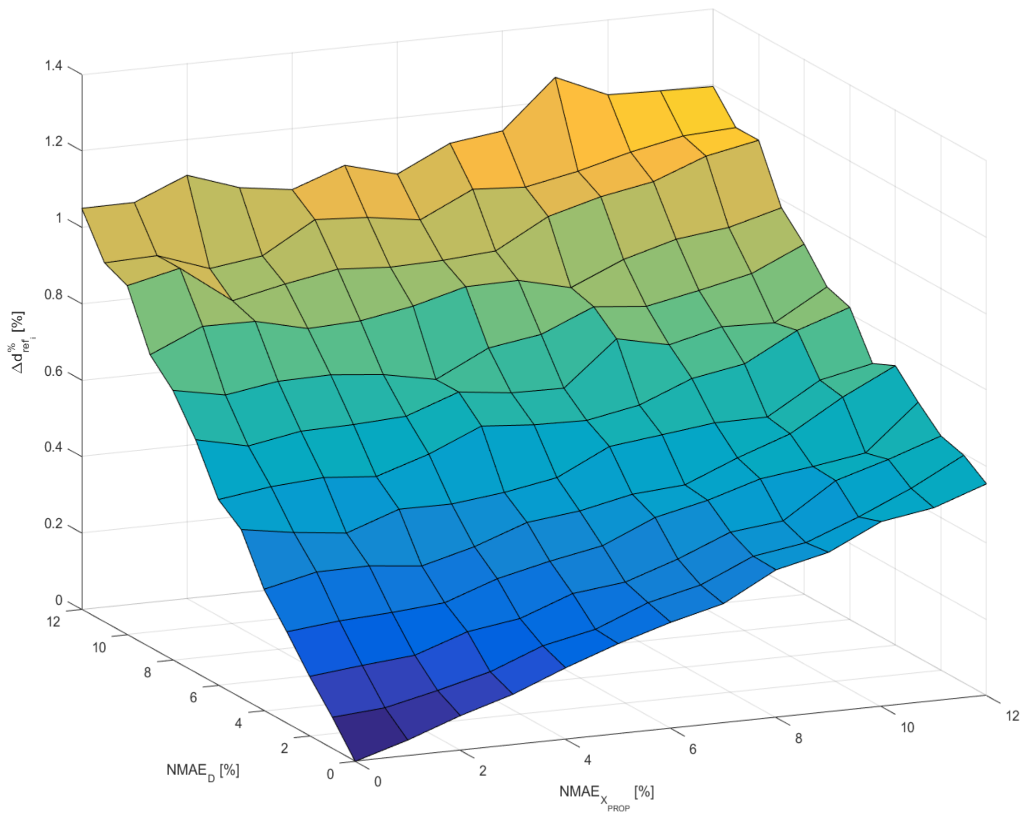
6. Having obtained an acceptable level of randomization with 50 iterations (the amount has been adjusted through several tests), the average reference distance can be calculated. An absolute deviation is computed with the unbiased value obtained at step 2, then normalized to obtain a $\Delta d^{\%}_{ref_i}$ as in (5.17), which is valid for the set values of NMAE on D and X_{PROP}.

7. Steps 3-6 are repeated over a domain of NMAE combinations, in order to cover all values relevant to the study with sufficient granularity. In this way, a complete error surface of $\Delta d^{\%}_{ref_i}$ can be built for a single landing instance.

8. All error surfaces obtained by iteration of steps 2-7 are reduced to a unique median surface, in order to provide a universal tool for the evaluation of metamodels, independent of the specific landing characteristics. An error volume could have been retained, rather than an average surface, however this would have made the final assessment more complex and not as easily interpretable. This choice has the additional benefit of reducing the influence of outliers and highlighting the main underlying trend, on the same lines of the sampling philosophy discussed in 5.2.2.

9. From the global error surface obtained, it is possible to trace heatmaps or (as done here) a set of level curves for constant values of $\Delta d^{\%}_{ref_i}$. Based on these results, we can evaluate the performance of surrogate models for any combination of NMAE on the variables D and X_{PROP}.

The resulting error surface mentioned at point 8. of the above process is visualized in the top of Figure 5.7. This shape has been obtained consistently throughout several repetitions of the process and is therefore not affected by the randomization elements introduced. In the bottom part of the same figure, a set of level curves of the error surface are reported (as hinted at step 9.). Such levels have been taken arbitrarily in this case, only to give an idea of the trend of the error function.
It certainly stands out how the relative error on drag has a higher effect than the one on longitudinal thrust. In fact, after a comparable initial increase, the error on d_{ref_i} starts to grow more rapidly along the $NMAE_D$ dimension. For reference, a relative error of around 11% on X_{PROP} corresponds to a 0.5% difference in landing distance, if the error on thrust is the only contributor. For the same 11% error, if only caused by drag, the error on d_{ref_i} is doubled to approximately 1%. Having said that, however, these values already show how the present application has a good level of resilience with respect to the error on these variables.
In the context of this work, the obtained global surface and especially a corresponding set of level curves, can be used to evaluate any surrogate model with outputs D and X_{PROP}. Based on the tentative target accuracy of 0.5% proposed in the previous paragraphs, any combination of modeling errors $NMAE_D$ and $NMAE_{X_{PROP}}$ falling below the 0.5% level curve in Figure 5.7 (bottom) would be considered accurate. This framework will be applied, with appropriate adjustments, to the trained models that are presented in the next chapter.



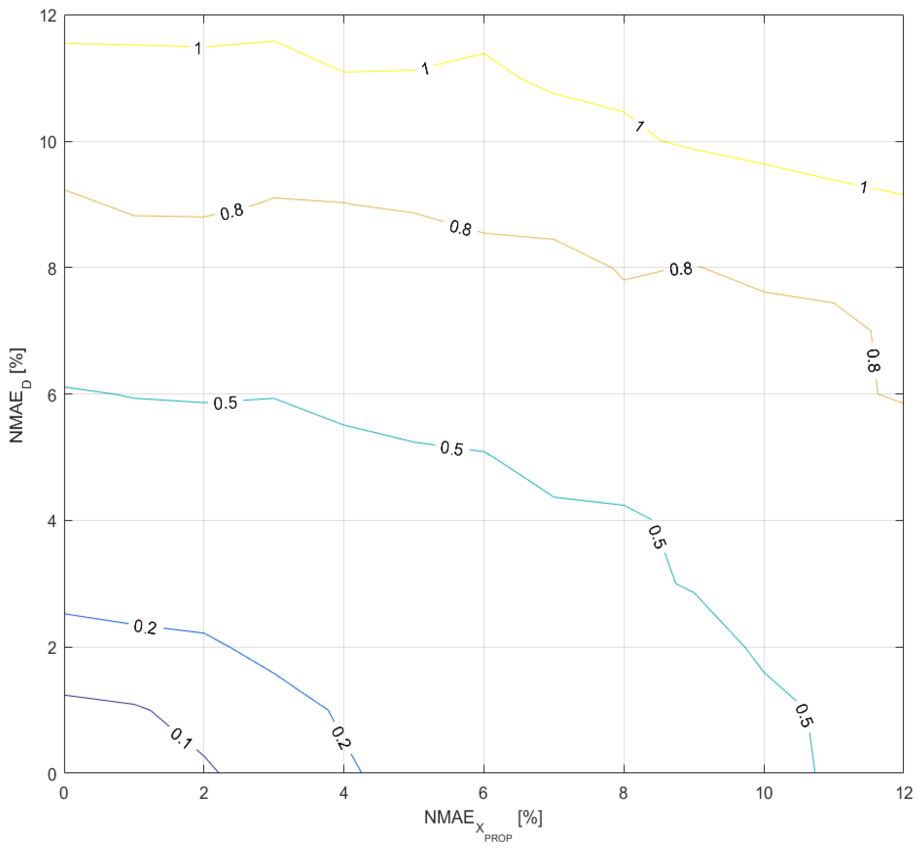


Figure 5.7: Global error surface resulting from the numerical feasibility study (top) and level curves of the same surface taken at relevant constant values (bottom).

6
Results

The current chapter reports the outcomes of the machine learning process introduced in 5.3. The overall structure of the chapter is established on the comparison between a baseline surrogate model and several variations on this main architecture. A further assessment is performed on comparable polynomial models for a selected number of neural networks. The evaluation of all obtained models is accomplished by means of the performance indexes presented in 5.3.5, and in the light of the impact on the function of interest.

6.1. Baseline model

The first results presented in this chapter concern the baseline network described in 5.3.2, which is used as a standard for comparison with several other metamodel structures. An intuitive representation of the model is provided in Figure 6.1. The image shows the number of nodes in each layer, the chosen activation functions and the input and output variables of the network.

6.1.1. Global results

The baseline network structure has been applied to reduce both aerodynamics and engine model, resulting in the global performance indexes reported in Table 6.1.
The reduction achieved in memory requirements (282 coefficients for each model) comes with an inevitable decrease in accuracy of the models with respect to all output variables. An evident trait is that the accuracy is always worse when computed on the validation dataset. This is a reasonable outcome that will be verified throughout all results, since a metamodel is expected to perform better on the data it has been trained with. Moreover, since this deviation between datasets is not high in comparison with the values presented, it can be concluded that the two models have reached a good generalization on the available data.
The data shown in Table 6.1 are referred to neural networks that have been trained with the Bayesian Regularization algorithm [99][100], which will be held as the reference training method in the remainder of the document. As already pointed out in the previous chapter, Levenberg-Marquardt [98] has also been implemented throughout this project, producing overall consistently poorer results. To prove this fact, the performance of the baseline models trained with Levenberg-Marquardt has also been reported in Table 6.2.

6.1.2. Error analysis

In order for the results of Table 6.1 to be fully comprehensible, a comparison of the errors with the scale of the measurements involved is required. The plots provided in Figures 6.2 and 6.3 can help in this task. Here the error computed on each sample in the training and validation dataset is traced against the corresponding target value. Such representation is given for both pairs of output variables in the two models. Ideally, the points in the graphs would remain within certain error limits along the whole domain of the output variable. It is reasonable to expect however, for a still acceptable surrogate model, to have the errors grow in proportion with the target value, hence maintaining a roughly constant relative error. Additional details and plots on the analysis of the relative error are provided in Appendix C.

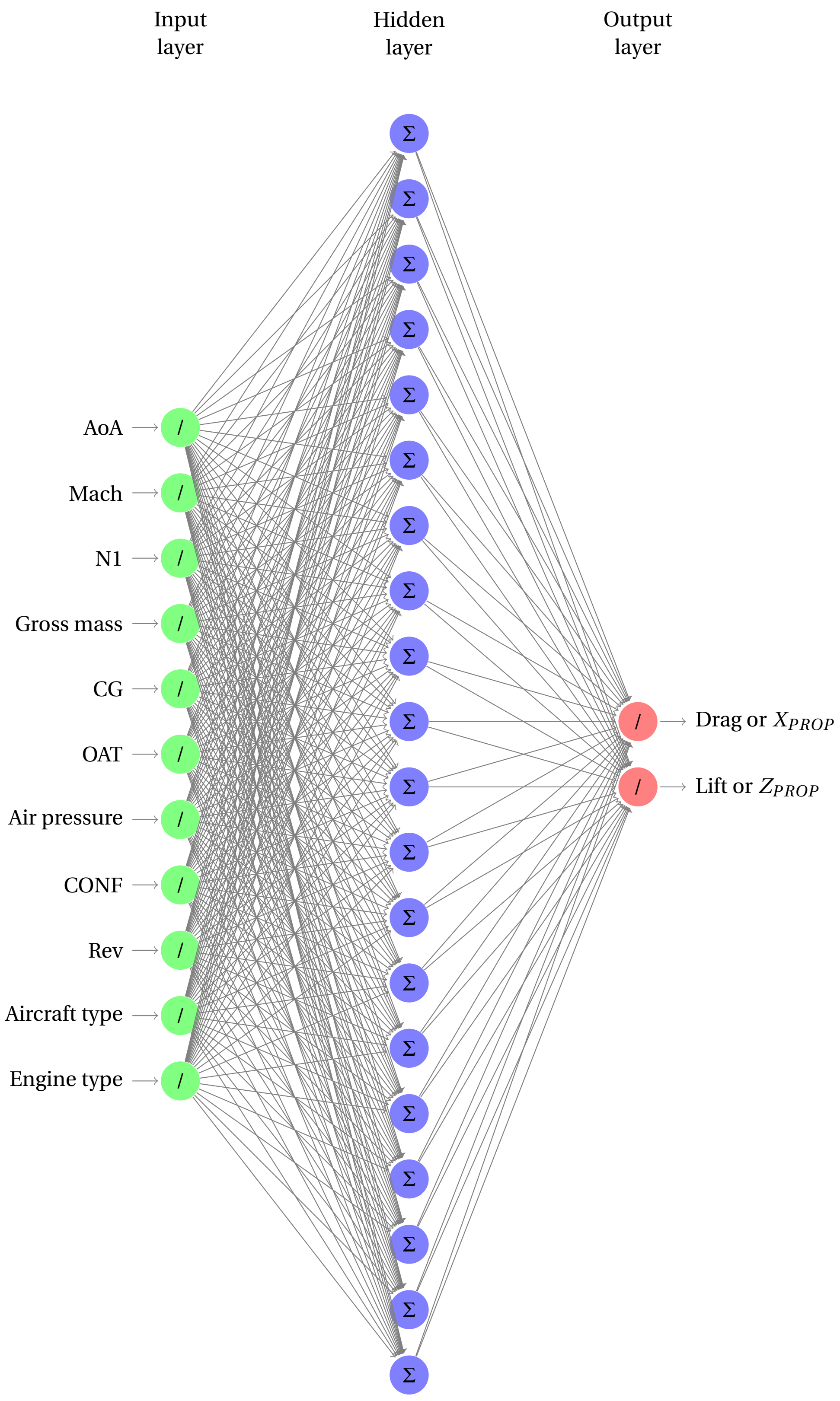


Figure 6.1: Baseline neural network structure: 11 input neurons, 20 in the hidden layer and 2 in the output layer. Input and output nodes follow a linear law, while the hidden neurons are given a sigmoid transfer function.

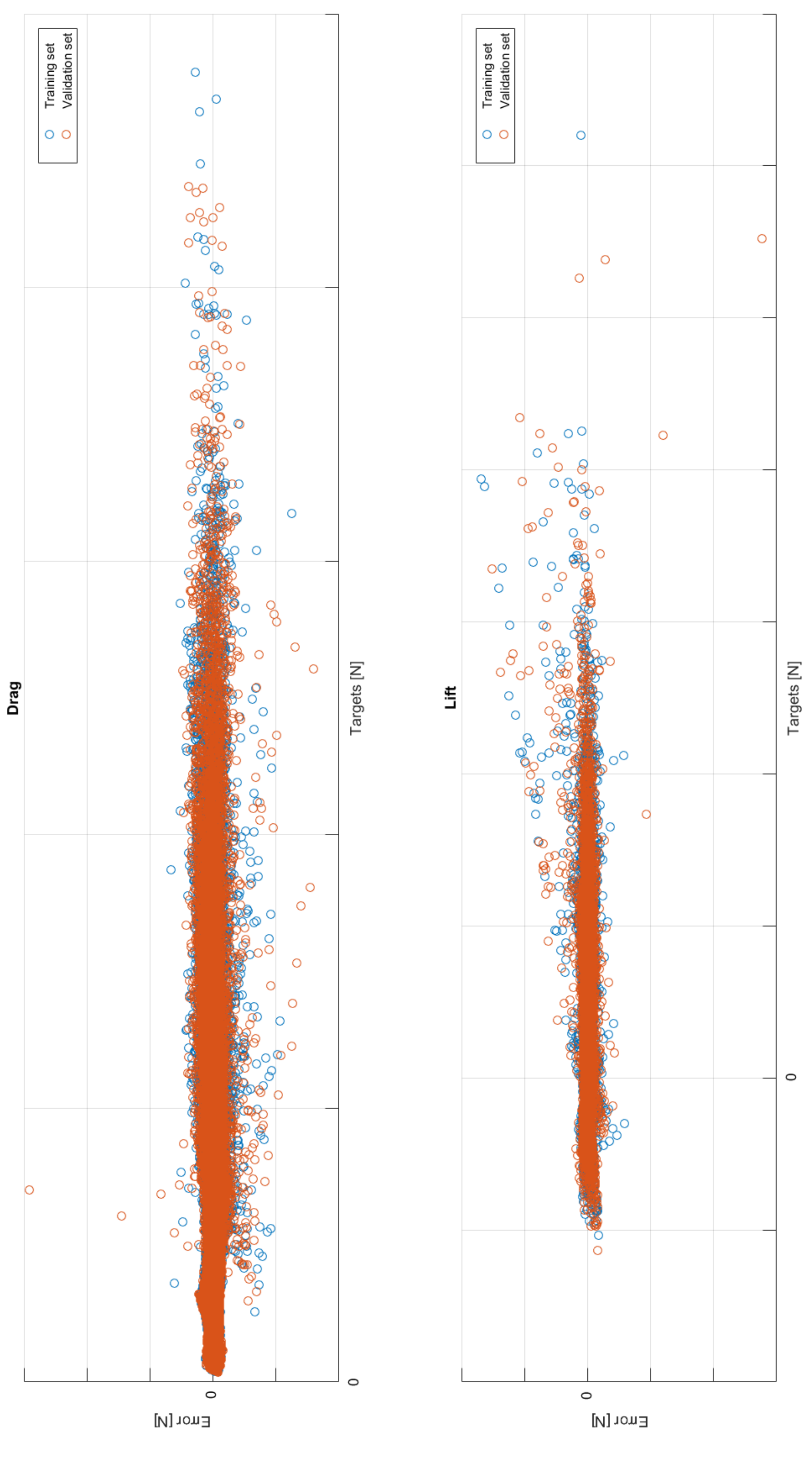


Figure 6.2: Baseline aerodynamics network error against target values in the training and validation datasets. Data are plotted for both output variables Drag and Lift.

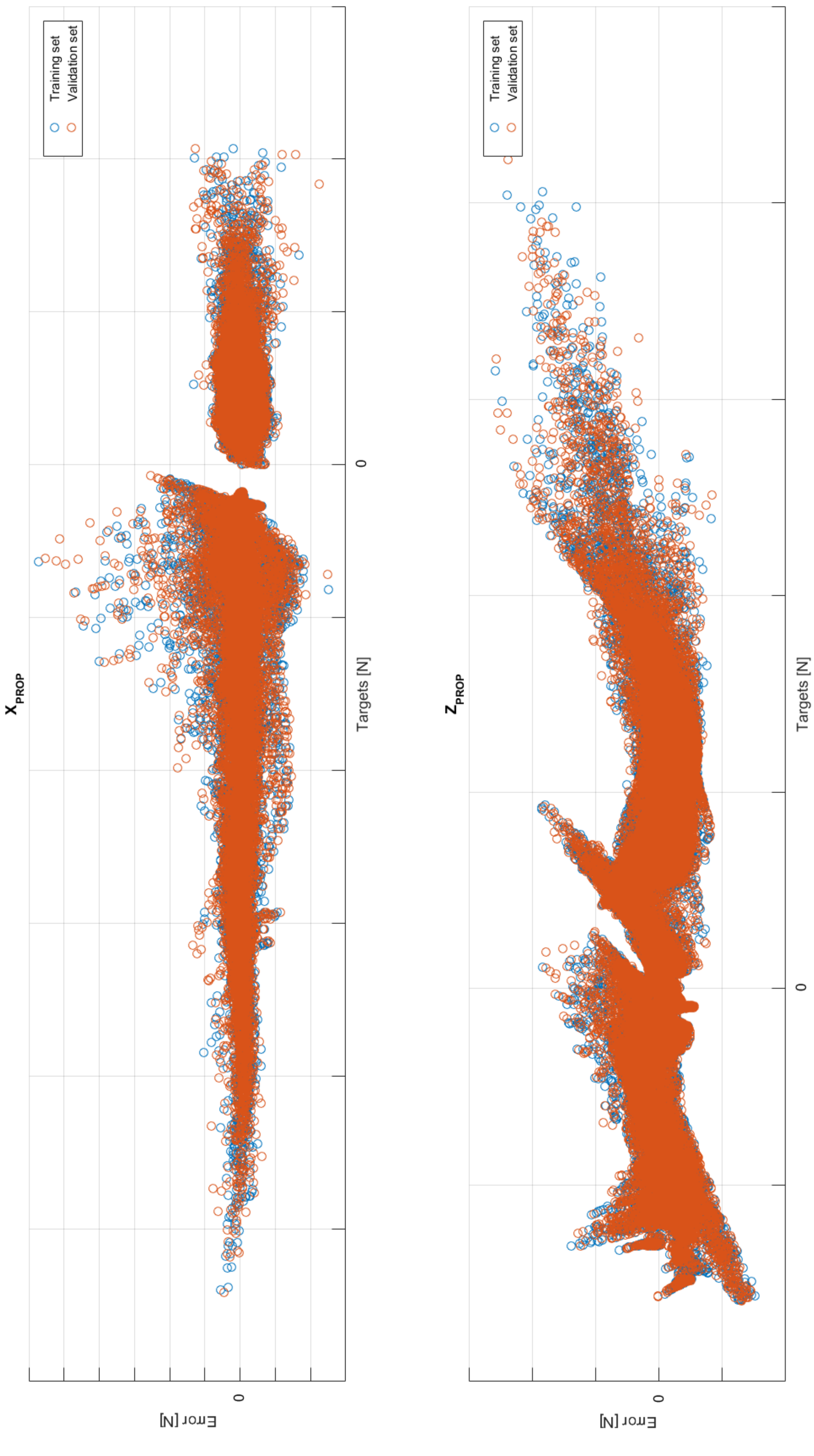


Figure 6.3: Baseline engine network error against target values in the training and validation datasets. Data are plotted for both output variables X_{PROP} and Z_{PROP}.

	Aerodynamics model		Engine model	
Index	Drag	Lift	X_{PROP}	Z_{PROP}
Number of inputs [-]	11		11	
Number of coefficients [-]	282		282	
$NMAE_{train}$ [%]	2.99	12.91	2.06	15.62
$NMAE_{val}$ [%]	3.00	12.87	2.05	15.63

Table 6.1: Performance indexes computed on the results of the baseline neural network models trained with Bayesian Regularization.

	Aerodynamics model		Engine model	
Index	Drag	Lift	X_{PROP}	Z_{PROP}
Number of inputs [-]	11		11	
Number of coefficients [-]	282		282	
$NMAE_{train}$ [%]	3.38	13.00	2.01	13.34
$NMAE_{val}$ [%]	3.37	13.24	2.03	13.33

Table 6.2: Performance indexes computed on the results of the baseline neural network models trained with Levenberg-Marquardt.

Aerodynamics model

The expected trends are found, to a certain extent, for the results of the aerodynamics model in Figure 6.2. For what concerns drag, the error is limited in nearly every part of the domain. A notable exception, other than some inevitable outliers, is represented by the area of lower target values, where the errors grow considerably in a relative sense. However, this aspect is of minor importance for the present study, as the model is less likely to work in these conditions.
Similar considerations can be made with respect to the results of lift, which is however characterized by the presence of several outliers. An interesting detail is the linear growth of the error for a subset of samples, which is a behaviour found at lower scale also in the drag plot. This characteristic (and in general any evident pattern in the error representation) is a classic signal for the neural network not being able to recognize some trait in the original model. In other words, the data provided to train and evaluate the metamodel contain some features that were not effectively taken into account during training. The issue is therefore ascribable to either a non ideal choice of metamodel architecture (in which case the network is simply not able to model some phenomena because it lacks structural flexibility) or an inadequately performed training (which can be excluded here given the carefully-thought measures described in 5.3.4).

Engine model

The issue of unrecognized patterns is found at a noticeably higher level in the results of the engine model, given in Figure 6.3 for both variables. This is particularly evident in the plot of the Z_{PROP} variable where, since the values are of a low order compared to the horizontal component of thrust, local phenomena appear more clearly. Concerning the error on X_{PROP} (which is the variable of real interest for the engine model) the values obtained are globally not as contained as what is achieved in the aerodynamics model. High errors are observed for low values of thrust, both in absolute and relative sense. Contrarily to the aerodynamics model, this aspect represents a problem here, since the engine is often operated in low-thrust regimes during the landing manoeuvre.
A last interesting feature that stands out in the plot is the absence of samples for slightly negative values of thrust. Such gap is readily explained, as it corresponds to the area of transient operation of the engine, which was removed from the analysis as described in 5.2.2.

6.1.3. CONCLUSIONS

Based on all considerations made above, it can be concluded that the baseline neural network represents a promising starting point for the reduction of both aerodynamics and engine model. Overall, the aerodynamics model seems more reliable for standard aircraft operation, with satisfying results achieved also in terms of storage requirements. The latter consideration can also be made with respect to the engine model, although its relatively poor accuracy prompts the need for further investigations before making it an actual candidate for deployment. The main direction in which to conduct the study, as already pointed out in the above, is towards possible improvements of surrogate model structure. The next sections will examine various options that have been considered in this project, trying to find acceptable compromises between accuracy and memory requirements.

6.2. IMPROVEMENTS OF NEURAL NETWORK STRUCTURE

As already pointed out in this work, one of the most interesting features of feedforward neural network models is the ability of fitting any nonlinear function with arbitrary degree of accuracy. This can be achieved with as few as one hidden layer, provided that enough neurons are given in the model. Moreover, a network with more than one hidden layer is expected to achieve better performance, given the same total number of neurons. The presence of multiple layers in fact, allows the network to process the input information in a more complex way. [91]

Based on these theoretical considerations, a study has been performed on the effect of modifying both the number of neurons and layers of the baseline model, while maintaining unaltered its remaining characteristics. Network architectures featuring at most three hidden layers were considered. In the single-layered case, the number of neurons in the hidden layer was varied between 5 and 30. With multiple layers instead, combinations of 3, 5 and 10 neurons were tested. All these choices were made by considering the range of characteristics sought for an accurate and well-reduced metamodel. The aim of the analysis is in fact that of studying potential trends in the results and identifying the most promising traits in the surrogate models.

The global outcomes of the investigation are given in Figures 6.4 and 6.5, where the accuracy in terms of NMAE of every trained metamodel is plotted against the required number of coefficients. The graphs are provided separately for each variable in the two models and the values are computed on the validation dataset. Moreover, the points corresponding to the baseline model structure of the previous section have been highlighted, showing no significant difference with the results of the analogous 20 neuron single-layered model of this study.

6.2.1. GENERAL REMARKS

Overall, a set of trends resembling an exponential decay is observed in almost every case. This result is in good agreement with the literature [25] and is also a reasonable one: the more complex a metamodel is, the better its approximation of the original model, at the cost of a higher memory requirement. However, this aspect seems to be strictly valid only for what concerns the total number of neurons in the networks, as different numbers of layers instead introduce more complex features in the results. A higher number of layers, even for the same number of coefficients, does not necessarily correspond to better performance (as also confirmed by the literature) and the way neurons are distributed among the layers has a heavy effect on the accuracy. To highlight this fact, the lines in each graph were not only rendered with different colors based on the number of layers, but also different lines were traced according to the number of neurons in the first layer. This latter characteristic seems to play an important role in defining different levels of reduction: for the same number of layers, a higher number of neurons in the first layer pushes the exponential-like trend closer to the axes of the graph. The same effect is observed, though to a lower extent and coherence, for what concerns the number of layers. Such aspect is paramount for the present analysis, as curves that approach the bottom left of the graphs identify better compromises of accuracy and storage requirements.

From a general point of view, the analysis provides valuable hints as to the type of models that are suitable for the problem at hand. These have been investigated separately for the results of the two models.

6.2.2. AERODYNAMICS MODEL

Concerning the aerodynamics model, shown in Figure 6.4, good reductions are obtained even with one hidden layer. A number of 15 and 20 neurons (where the latter corresponds to the baseline of 6.1) may represent an interesting compromise depending on the specific requirements involved. However, the real power of the neural network technique is clearly better exploited when more hidden layers are considered. Quite surprisingly, the set of 2-layered models appears to offer a consistently wider range of candidates compared to those with 3 layers. Taking into account the results on both Drag and Lift calculation, the models identified as (10,5) and (10,5,5) represent the best compromises for the aerodynamics model. These in fact show substantially better approximation characteristics than the baseline model, at the same time achieving good levels of storage requirements. Of the two, the former model can be regarded as the preferred choice, given its higher reduction, structural simplicity and marginal accuracy penalization compared to the 3-layered network.

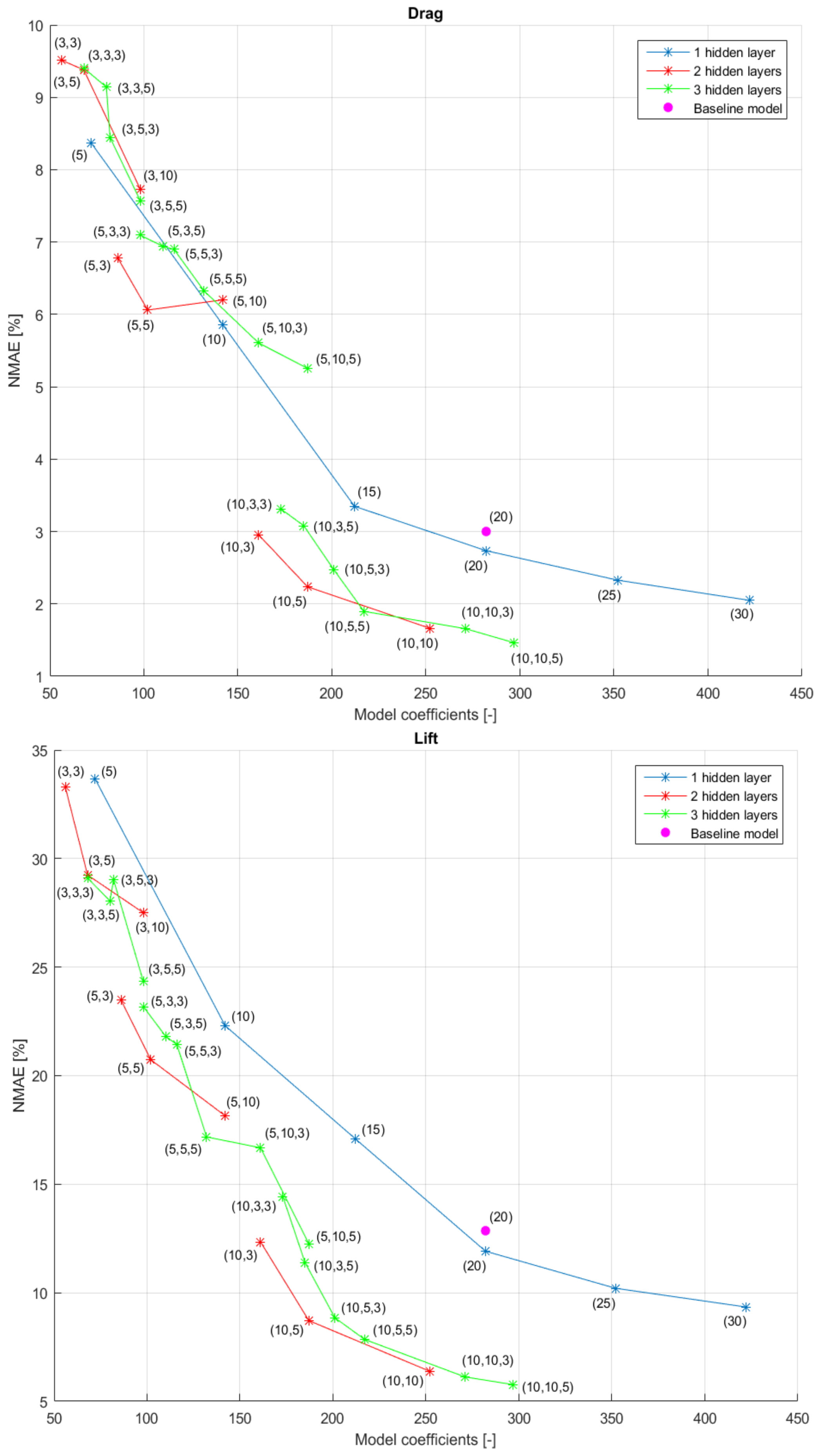


Figure 6.4: Study of neural network structure modification for the aerodynamics model, with values of NMAE computed on the validation dataset. The number of neurons in the hidden layers is reported between brackets next to each point in the graph.

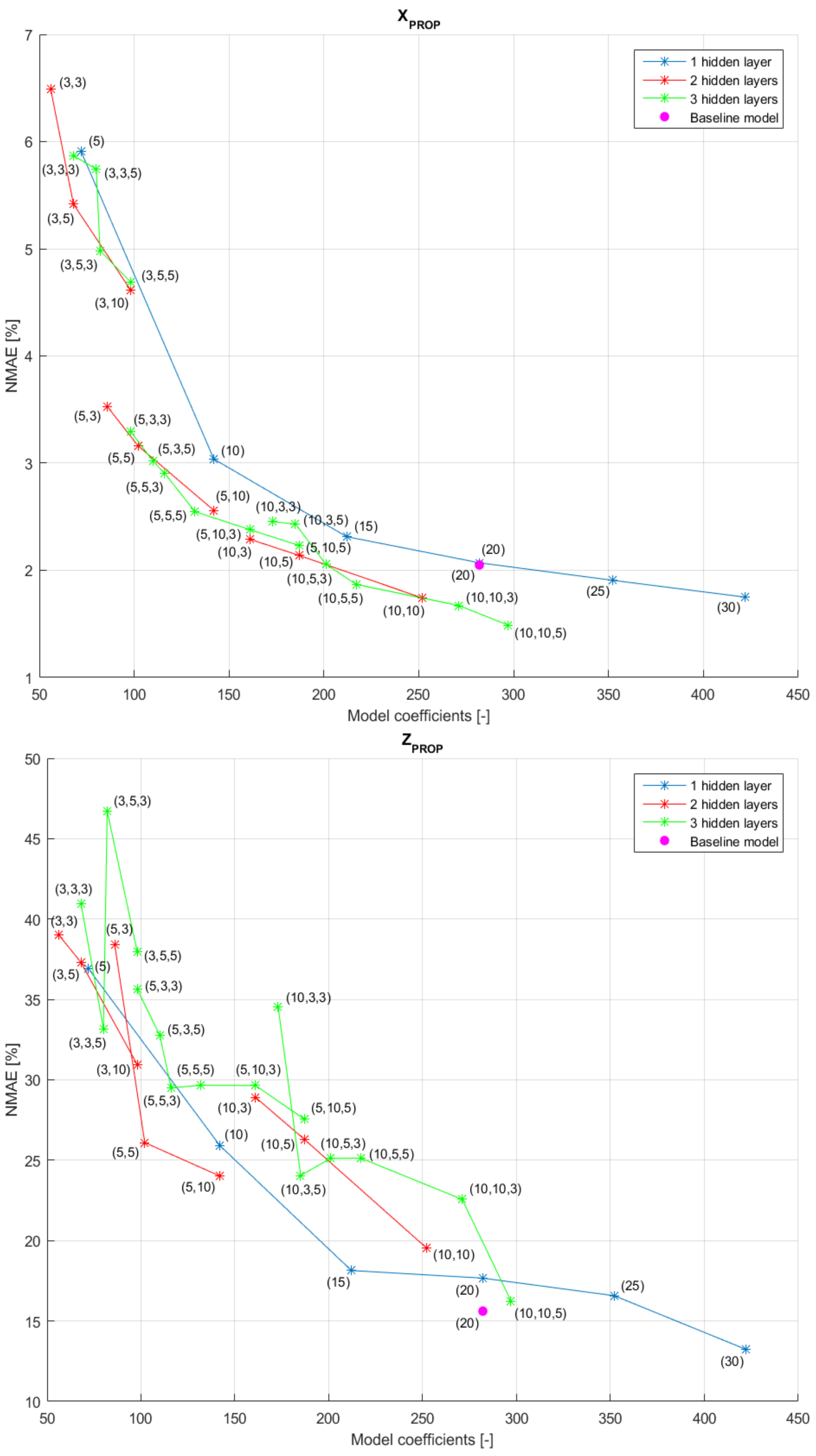


Figure 6.5: Study of neural network structure modification for the engine model, with values of NMAE computed on the validation dataset. The number of neurons in the hidden layers is reported between brackets next to each point in the graph.

Index	Single model			
	Drag	Lift	X_{PROP}	Z_{PROP}
Number of inputs [-]	11			
Number of coefficients [-]	324			
$NMAE_{train}$ [%]	3.31	13.85	4.65	19.94
$NMAE_{val}$ [%]	3.28	13.84	4.64	19.94

Table 6.3: Performance indexes computed on the results of the single neural network for both aerodynamics and engine model.

6.2.3. ENGINE MODEL

Similar considerations are made with respect to the results of the engine model in Figure 6.5. In this case the baseline model could indeed benefit from an additional reduction, as shown by the positive performance of the 15 neuron model. Concerning networks with multiple hidden layers, the situation is more difficult to judge compared to the aerodynamics model. Here the results for the X_{PROP} variable should be considered the more reliable source of information. The variable Z_{PROP} in fact, due to its lower magnitude, generates more erratic patterns that are relatively reliable for the analysis.
In the choice of the best candidate model, here the requirements of the specific application play a fundamental role. Several options are available according to whether the accuracy or the size reduction of the network is given more importance. For the sake of consistency, the models (10,5) and (10,5,5) are chosen also in this case as the preferred compromises. For the same ideas seen with the aerodynamics model, the 2-layered network can be picked as the overall best choice in the set of results.

6.3. SINGLE REDUCED MODEL

The feasibility of a profoundly different approach to the problem has also been tested in the project. In this investigation the aerodynamics and engine model are reduced together in one unique neural network, with common inputs and outputs. The major advantage with this solution, is that the storage requirements of such global metamodel is lower than that of the two separate reduced models summed together. This consideration obviously holds if the same network architecture is implemented.
Based on the above idea, a neural network model with a single 20-neuron hidden layer has been implemented, reducing both the aerodynamics and engine model together. The input parameters are the same as the baseline model structure shown in 6.1, while the four outputs are the combination of those found in the two models. The performance achieved by the best trained network is reported in Table 6.3. Given the model structured considered here, only one value is provided for the number of coefficients in the network.
A few considerations can be made by comparing the obtained results with those of the baseline models in Table 6.1. Concerning the accuracy, it stands out clearly how the separate baseline models outperform the single model, notably for the engine variables. In terms of memory reduction, however, the single model achieves a significant improvement compared to the equivalent aggregate index of the baseline models.
If a choice were to be made based uniquely on the above results, the solution of a single model would be discarded. This is an inevitable consequence of the excessively poor accuracy obtained in the computation of engine thrust, especially when compared to the fairly good performance on the aerodynamic outputs. It is expected that smoothing out such disparity would require heavy adjustments of model structure, conceivably leading to a substantially higher number of coefficients. A more complete study, which is out of the scope of this project, would focus on testing different architectures for the single model against combinations of separate models having likewise different architectures.

6.4. NEURAL NETWORK WITH SELECTED FEATURES

An additional investigation has been performed, based on the already described idea of feature selection of 5.3.1. In this case the aerodynamics and engine model have been approximated by means of baseline neural networks, only with reduced sets of input variables. The two input sets are reported in Table 5.3 and were chosen according to a correlation study between input and output parameters in the original models. The analysis is particularly interesting for the topic in question, as it directly tackles the problem of memory load. The benefit however, comes at an unavoidable cost in terms of accuracy, as the reduced model is provided with less information about the phenomena of the system.

	Aerodynamics model		Engine model	
Index	Drag	Lift	X_{PROP}	Z_{PROP}
Number of inputs [-]	9		7	
Number of coefficients [-]	242		202	
$NMAE_{train}$ [%]	3.39	24.90	2.17	12.99
$NMAE_{val}$ [%]	3.37	24.74	2.16	12.96

Table 6.4: Performance indexes computed on the results of neural network models with selected features.

	Aerodynamics model		Engine model	
Index	Drag	Lift	X_{PROP}	Z_{PROP}
Number of inputs [-]	16		16	
Number of coefficients [-]	382		382	
$NMAE_{train}$ [%]	2.96	15.23	2.10	12.80
$NMAE_{val}$ [%]	2.88	15.17	2.12	12.75

Table 6.5: Performance indexes computed on the results of neural network models with boolean categorical inputs.

The results of the investigation, provided in Table 6.4, show some remarkable characteristics when compared to those of the baseline. On one hand, the reduction ratings achieved for both models are indeed substantial and outperform the baseline models in both cases. On the other hand, the loss of accuracy occurs with very different extent in the two models. The aerodynamics model shows a significant decrease in approximation power, with poor performance especially in the computation of lift. Conversely, despite being built with a lower number of input features (7 against 9), the engine model attains a considerably better performance than the aerodynamics one. This aspect suggests how the input set of the baseline model structure is more redundant for the description of the propulsion problem, while the aerodynamic phenomena indeed require more variables to be properly modeled.

6.5. BOOLEAN CATEGORICAL INPUTS

The problem of providing a reduced model with categorical information has been thoroughly treated in 5.3.2. The preferred solution was reported as that of single inputs with integer values, which is the one implemented in all the models presented so far in the chapter. For the models under examination, this aspect concerns the variables of aircraft and engine type (high-lift configuration and reverse are binary inputs).
The present section is focused on the alternative approach, consisting in sets of binary variables where only one is active at a time. This possibility has been applied in the construction of the baseline neural network structure, where the only difference is thus a higher number of input features.
The overall results of the study on the aerodynamics and engine model are given in Table 6.5. In both cases, no significant improvement in performance is observed with respect to the baseline model of section 6.1. Moreover, this outcome is obtained at a substantial cost in terms of memory requirements, making it an unfavourable solution for the study. A larger or deeper network would be required to fully take advantage of the more accurate categorical description, however that would also come at an additional cost in terms of storage. These results lead to the conclusion that, for the problem at hand, the choice of integer values for categorical inputs represents a satisfactory compromise which does not hinder the performance of the surrogate models.

6.6. POLYNOMIAL MODELS

As already mentioned in the previous chapter, it is common practice in metamodeling studies to compare different surrogate model structures. In this project, multivariate polynomial models have been chosen as a means for comparison with the results achieved by artificial neural networks.
In this case, the surrogate models are always fitted with ordinary least squares and the number of available samples is reduced by a factor 2.5 compared to the neural network training datasets. A further important difference is that polynomial models can only output one variable, hence two separate metamodels have to

	Aerodynamics model		Engine model	
Index	Drag	Lift	X_{PROP}	Z_{PROP}
Polynomial degree [-]	2	2	3	3
Number of inputs [-]	11	11	7	7
Number of coefficients [-]	78	78	120	120
$NMAE_{train}$ [%]	7.01	60.65	11.74	19.30
$NMAE_{val}$ [%]	6.98	60.74	11.70	19.40

Table 6.6: Performance indexes computed on the results of the polynomial models.

be fitted for both aerodynamics and engine model.
Polynomial models of degree up to 4 have been trained, where however the highest degree has only been implemented in the case of selected features for the engine variables. Additional models would have required an excessive number of coefficients, with no added benefit to the study.
Both baseline and selected input sets (cf. Table 5.3) have been tested and the overall results are reported in Table 6.6. Based on the observed performance across different polynomial structures, second degree polynomials are chosen for the aerodynamic variables, while third degree polynomials are retained for the engine model as the best solutions. Despite the satisfactory reduction levels obtained, the accuracy provided by polynomial models is still far from acceptable for real-world application in aircraft systems.

6.7. OVERVIEW

The last section of this chapter aims to summarise the essential aspects of the results of the different analyses previously presented, in order to identify the most promising compromises between accuracy and storage requirements.
On a general level it can be pointed out that all metamodeling solutions presented in this chapter achieve considerable reduction ratings based on the required number of coefficients. The only notable exception here are the models trained with boolean categorical inputs, whose memory requirements are far greater than all other solutions.
The baseline neural network has been shown to be a satisfactory starting point for the further refinement performed in the work. The extension of the model to multiple layers led to the best overall compromises between accuracy and memory requirements, confirming the intuition that an investigation on the model structure should be sought for an optimal construction of an embedded model.
Lastly, among the main underperforming solutions are polynomial models and feature-selected models, where the latter only being a viable option for the engine variables.

6.7.1. ASSESSMENT AGAINST ACCURACY FRAMEWORK

The last step in the analysis, as mentioned in 5.4.2, is to evaluate the performance of the obtained models against the accuracy framework developed in the same section of this work. In this case, the accuracy is assessed on the final calculation of the function under consideration (the landing distance), hence evaluating the effect of introducing surrogate models in the function.
In Figure 6.6, the NMAE validation results of the previously selected most relevant models are placed within the abovementioned framework, given in terms of error level curves. It is worth noting that, as the graph is given in terms of NMAE on Drag and Xprop, combinations of separately-trained models are here considered only with same model structure. Obviously more combinations are available by implementing different model structures to compute the two variables, with varying results.
The neural network structure with 3 hidden layers marks the only point below the 0.2% threshold, even though this clearly comes at a cost in terms of memory requirements. Nonetheless, for all selected models, the resulting effect on the chosen index $\Delta d^{\%}_{ref_i}$ appears to be limited, with values well below the tentative threshold of 0.5% previously selected in 5.4.2.

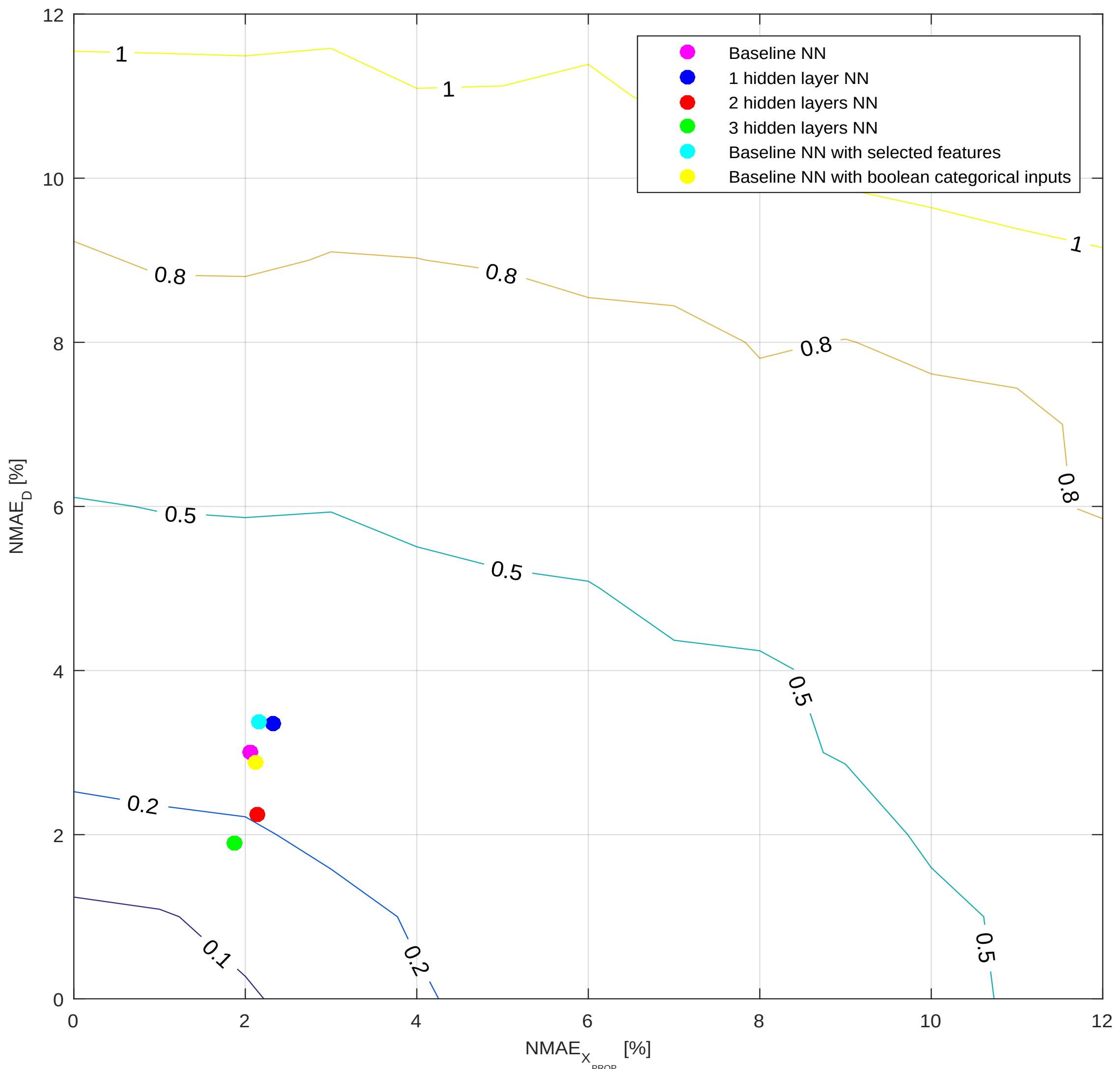


Figure 6.6: Level curves obtained in section 5.4.2 for landing distance computation error, compared with the estimated performance of the best trained models in the study.

7
CONCLUSION AND RECOMMENDATIONS

The present document has reported on the work of a master thesis project whose subject matter is the implementation of neural networks within aircraft performance function. An outline of the fundamental theory behind the topic, as well as a literature review on the state of the art in the field have been provided. Based on that, the general aims of the project have been formulated, which can be stated as the investigation of different solutions for the reduction of aircraft performance submodels within the case study of a landing performance function. A detailed overview of the methodology implemented in the work has been given, justifying the choices made at each step of the process. The overall procedure has been summarised as: a study of the problem, retrieval and sampling of datasets from a simulator, training and validation of surrogate models on such data, assessment of the various proposed solutions. The models are evaluated on an accuracy index against a reduction index, accounting for the degree of model simplification. Moreover, the most promising solutions are assessed against their estimated performance within the target function. An additional comparison is made with polynomial models, which act as a benchmark for the assessment of the neural network models.

7.1. EVALUATION OF THE RESULTS

Overall, the investigation produced very promising outcomes, confirming the potential of the various techniques tested on the case study. This concerns the dataset construction, including choices made on the definition of a training and validation domain, but also the training algorithms implemented. Moreover, the analysis has given important hints as to which directions hold validity in the reduction process examined here. This aspect is especially important for what concerns the architectural choices made in the various neural network structures proposed.

One of the main results that stands out, is the substantial advantage in the use of multilayer networks, which leads to interesting compromises between accuracy and reduction. This is a clear conclusion from the analyses presented in 6.2. What is not obvious, however, is whether an even higher number of layers would necessarily produce better performance. To support this statement, it is observed how the results of the Bayesian Regularization algorithm have outperformed those of the Levenberg-Marquardt algorithm. This might be due to, among the various reasons, the notable "pruning" process of the algorithm, which can be seen as a rudimentary optimization of the network. Moreover, since only a set of fixed number of neurons (3, 5, 10 per layer) have been tested, it is possible that smarter choices in this sense could lead to better performance even with only 3 hidden layers. With respect to the network architectures, the 20 neuron single-layer baseline models have been proven as an interesting starting point, representing a robust means for benchmarking. Its validity in the assessment of different solutions is, however, purely indicative and is used as a reference for the whole body of results. This is especially true in the light of the huge potential proven by other design solutions.

Concerning the aspect of feature selection, the study has demonstrated a considerable difference in sensitivity between the aerodynamics and engine submodels, with the former being an evidently more complex problem to approximate. Conversely, the latter model has definitely shown some wider margins in terms of input refinement. It has to be noted, however, that the choice of input parameters made in this project is not

an optimal one, being only based on correlation indexes and engineering judgement.
The idea of reducing both performance submodels in a single surrogate model has led to very poor outcomes. This has been observed especially for what concerns the engine outputs. Moreover, this conclusion is made in comparison with the baseline model structure, taking into account the relative proportions in reduction ratings. Even with such approach, however, the resulting accuracy does not justify the construction of a single model. It is paramount to note, nonetheless, that only one network structure has been tested for this solution. It is not to exclude that different architectures with more layers and nodes could lead to better performance and thus prove the validity of the solution.
An aspect that is deemed as one of the strongest points in the project, is that of including multiple instances of aircraft and engine types within the training process. The generalization of the obtained metamodels thus holds a special value in terms of portability of the solutions. In this light, the results shown in this document are particularly interesting, especially for aircraft systems that do not allow the integration of case-specific functions. The problem of categorical inputs has been solved in two ways. Interestingly, the simpler solution of providing the models with integer values for different situations has produced the best outcomes. The other solution of providing sets of boolean inputs to the networks instead, resulted in sensibly poorer accuracy, especially when considered its higher requirements in terms of input nodes (and hence number of network coefficients).
The additional study based on comparison with multivariate polynomial structures has led to the expected results. Polynomial models are not able to follow the complex patterns of the aerodynamics and engine phenomena, leading to low accuracy and high storage requirements. This conclusion can be confidently made, given that the models have been tested in a range of requirements that is comparable to that of the neural networks.
In terms of feasibility of the proposed solution, the retained models have been assessed against a framework to estimate the impact on the calculation of the objective function. The results show how the approximation introduced by the surrogate models has a low impact, measured in terms of relative error on the landing performance calculation, well contained within the 0.5% threshold set as a baseline. This result confirms the overall feasibility of a neural network-based approach to the problem of flight performance function submodel reduction.
A last element worth mentioning, is that none of the solutions proposed above is "the best" in an absolute sense. The whole point of the project has been that to explore and compare different solutions for the integration of reduced models within aircraft systems. The activity does not account for the peculiar limitations and requirements that should be considered for the specific application, as that needs to be evaluated on a case-by-case basis. Optimal compromises cannot be given a priori, but are rather dependent on the objectives of the application and the nature of the chosen solution.

7.2. LIMITATIONS AND FURTHER STUDIES

The previous section has provided an outline of the quality and most remarkable trends within the results of the project. As an additional, last step, it is useful to describe the main limitations that these results present in terms of extent and applicability in different contexts. Based on these observations, some ideas can thus be suggested for the development of further investigations on the topic of interest.
In this regard, the following points have been identified:

- One of the main limitations in the thesis project can be found in its general approach to the task. As already said in various contexts, the work has been structured as an exploration of the different possibilities that are available in the reduction of performance models. It is not the purpose of this activity to examine one specific solution in depth, but rather compare the efficacy of various approaches to the same problem. The idea is then to pave the way for detailed investigations in the directions that have been open, based on the demonstrated potential of some solutions. This idea mainly concerns the implementation of training algorithms and metamodel structures, while steps of dataset retrieval and sampling are more heavily related to the specific context of application.

- An approach that could build on this project would be focused on a specific subset of modeling techniques, given that some characteristics have been fixed thanks to existing results. An investigation would thus be conducted in order to make a set of optimal choices and get to results that are specific for the defined scope. An example of that, as a consequence of the observations made in 7.1, could be an optimization study on the number of layers and neurons in a multilayer network. Even more

innovative approaches could be applied, such as evolutionary algorithms (as already mentioned with [37]) which are a smarter way to optimize a model with various parameters. A similar reasoning can be made with respect to the number of input parameters, which have been the target of a suboptimal feature selection process. This could be conducted in various ways, by applying one of the available algorithms with an arbitrary degree of complexity. [63]
It must be noticed however, that for problems of integration within aircraft systems, specific software or hardware structures are typically required to embed any model or function. Therefore, the impact that reprogramming or rebuilding of these structures may have on the overall integration costs is not a negligible aspect. Despite being a purely operational observation, this is however a point that, in some cases, might lead to the decision of retaining non-optimal characteristics within a certain model or function.

- The assessment of the proposed metamodels has been based on the pure research for compromises between approximation accuracy and storage requirements. The latter parameter, quantified by the number of coefficients in a model, is held as an index for its inner complexity. A third variable, such as standard computational time (see [15]) could also be implemented in further investigations. This might be an essential indication, according to the specific application at hand.

- The present case study has indeed been simplified in order to have a manageable yet exhaustively explored problem. Among the simplifications introduced, the problem has been studied as symmetrical. This has led to a sensibly reduced set of variables in the problem, including angles of drift and roll and any type of system failure. In addition to that, it was chosen not to include the effect of wind. Such features are the result of compromises on the size and level of detail of the analysis. These would definitely have to be included in a future complementary study before effective implementation in real operational applications.
  Concerning the missing inclusion of engine transient data in the study, as already observed, this was a necessary choice since the type of models implemented do not allow for time series processing. With respect to the specific case, this does not represent a real problem, as this phase can be approximated with simpler functions and is thus not worth of a reduction study. For more complex applications with engine transition, recurrent neural networks may be taken into account as a possible modeling solution. [103]

- The whole training and validation process has been carried out on data obtained from simulations. Moreover, the procedures implemented by the simulator comply with the requirements set by airworthiness regulations for a "correct" landing. Before effective deployment of any tool based on the results this project, it would be appropriate to perform an additional validation on real-life data. As a further idea, a possible future study would test the same methods implemented here by performing the whole training of reduced models with only this type of data.

- An important limitation to the project, is that the activity has been performed on the specific case study of a landing function. Hence, even though the formulation of the aerodynamics and engine model is fairly generic, the problem has been indeed tackled in a heavily contextualized setting. The results shown in the previous chapters are therefore to be interpreted, in a possible further study on performance functions, in the light of such limitation. The concern is especially valid for high-speed applications (out of the takeoff and landing envelope) where the same physical laws are clearly not applicable. It is essential to keep in mind, moreover, that the reduction has been carried out at performance submodel level. Any possible coupling with the equations of motion has not been considered and this would represent a whole additional level of complexity to take into account.

Bibliography

[1] T. Group, *Learn more about cockpit: history, how it works and evolution,* .

[2] S. Miller, *Contribution of flight systems to performance-based navigation,* .

[3] B. Dougherty, J. White, *et al.*, *Deployment optimization for embedded flight avionics systems,* CrossTalk **24**, 31 (2011).

[4] M. Bondouy, *Construction de modèles réduits pour le calcul des performances des avions,* Ph.D. thesis, Université Toulouse 3 Paul Sabatier (2016).

[5] G. Igna, L. Dieudonne, S. Voss, and B. Schatz, *Model-based deployment generation for safety-critical avionics systems,* in *Industrial Embedded Systems (SIES), 2017 12th IEEE International Symposium on* (IEEE, 2017) pp. 1–8.

[6] S. Kumar, *Fundamental limits to Moore's Law,* (2015).

[7] G. Moore, *Cramming more components onto integrated circuits,* Electronics **38** (1965).

[8] P. Bieber, F. Boniol, M. Boyer, E. Noulard, and C. Pagetti, *New challenges for future avionic architectures,* AerospaceLab , p (2012).

[9] H. Butz, *Open integrated modular avionic (ima): State of the art and future development road map at airbus deutschland,* Signal **10**, 1000 (2010).

[10] U. Drepper, *What every programmer should know about memory*, Red Hat, Inc (2007).

[11] P. Weisberg and Y. Wiseman, *Efficient memory control for avionics and embedded systems,* International Journal of Embedded Systems **5**, 225 (2013).

[12] C. L. Bowman, M. R. DeYong, and T. C. Eskridge, *Role of neural networks for avionics,* in *Advanced Imaging Technologies and Commercial Applications*, Vol. 2566 (SPIE, 1995) pp. 96–106.

[13] V. Klein and E. Morelli, *Aircraft system identification: Theory and Practice* (AIAA Inc., Reston, Virginia, 2006).

[14] M. Bondouy, S. Jan, S. Laporte, and C. Bes, *On the choice of surrogates for multilevel aircraft perfomance models,* in *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences* (Springer, 2015) pp. 95–109.

[15] M. Bondouy, S. Jan, S. Laporte, and C. Bes, *Optimal surrogates selection for embedded, hierarchical multilevel aircraft models,* IEEE Transactions on Aerospace and Electronic Systems **51**, 3415 (2015).

[16] G. Ghazi, R. M. Botez, and M. Tudor, *Performance database creation for Cessna Citation X aircraft in climb regime using an aero-propulsive model developed from flight tests,* AHS Sustainability (2015).

[17] L. Jensen, R. J. Hansman, J. Venuti, and T. Reynolds, *Commercial airline altitude optimization strategies for reduced cruise fuel consumption,* in *14th AIAA Aviation Technology, Integration, and Operations Conference* (2014) p. 3006.

[18] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, *Convergence properties of the nelder–mead simplex method in low dimensions,* SIAM Journal on optimization **9**, 112 (1998).

[19] C. Gong and W. Chan, *Using flight manual data to derive aero-propulsive models for predicting aircraft trajectories,* in *AIAA's Aircraft Technology, Integration, and Operations (ATIO) 2002 Technical Forum* (2002) p. 5844.

[20] A. A. Trani and F. Wing-Ho, *Enhancements to SIMMOD: a neural network post-processor to estimate aircraft fuel consumption,* (1997).

[21] K. D. Julian, M. J. Kochenderfer, and M. P. Owen, *Deep neural network compression for aircraft collision avoidance systems,* Journal of Guidance, Control, and Dynamics **42**, 598 (2019).

[22] J. A. Pytka, P. Budzyński, P. Tomiło, J. Michałowska, E. Gnapowski, D. Błażejczak, and A. Łukaszewicz, *Imumeter—a convolution neural network-based sensor for measurement of aircraft ground performance,* Sensors **21**, 4726 (2021).

[23] K. Jules and P. P. Lin, *Artificial neural networks applications: from aircraft design optimization to orbiting spacecraft on-board environment monitoring,* (2002).

[24] R. M. Paiva, A. R. D. Carvalho, C. Crawford, and A. Suleman, *Comparison of surrogate models in a multidisciplinary optimization framework for wing design,* AIAA journal **48**, 995 (2010).

[25] J. Corman and B. German, *A comparison of metamodeling techniques for engine cycle design data,* in *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference* (2010) p. 9352.

[26] K. Lee, T. Nam, C. Perullo, and D. N. Mavris, *Reduced-order modeling of a high-fidelity propulsion system simulation,* AIAA journal **49**, 1665 (2011).

[27] M. Lazzara, M. Chevalier, M. Colombo, J. G. Garcia, C. Lapeyre, and O. Teste, *Surrogate modelling for an aircraft dynamic landing loads simulation using an lstm autoencoder-based dimensionality reduction approach,* Aerospace Science and Technology **126**, 107629 (2022).

[28] Z. H. Ji and S. Jiang, *Monitoring aero-engine fuel flow based on neural net,* in *Advanced Materials Research*, Vol. 490 (Trans Tech Publ, 2012) pp. 979–984.

[29] M. T. Yildirim and B. Kurt, *Engine health monitoring in an aircraft by using Levenberg-Marquardt feedforward neural network and radial basis function network,* in *INnovations in Intelligent SysTems and Applications (INISTA), 2016 International Symposium on* (IEEE, 2016) pp. 1–5.

[30] C. Zhang and N. Wang, *Aero-engine condition monitoring based on support vector machine,* Physics Procedia **24**, 1546 (2012).

[31] C. Meyer and W. Maul, *The application of neural networks to the ssme startup transient,* in *27th Joint Propulsion Conference* (1991) p. 2530.

[32] Y. Gao and Y. Shen, *Prediction of aero-engine exhaust gas temperature based on chaotic neural network model,* in *Advances in Mechanical and Electronic Engineering* (Springer, 2012) pp. 145–150.

[33] A. Zavoli, P. M. Zolla, L. Federici, M. T. Migliorino, and D. Bianchi, *Surrogate neural network for rapid flight performance evaluation of hybrid rocket engines,* Journal of Spacecraft and Rockets **59**, 2003 (2022).

[34] M. Nørgaard, C. C. Jorgensen, and J. C. Ross, *Neural network prediction of new aircraft design coefficients,* (1997).

[35] R. Wallach, B. Mattos, R. Girardi, and M. Curvo, *Aerodynamic coefficient prediction of transport aircraft using neural network,* in *44th AIAA Aerospace Sciences Meeting and Exhibit* (2006).

[36] R. H. Myers, D. C. Montgomery, *et al.*, *Response surface methodology: process and product optimization using designed experiments*, Vol. 3 (Wiley New York, 1995).

[37] E. Andrés, S. Salcedo-Sanz, F. Monge, and A. M. Pérez-Bellido, *Efficient aerodynamic design through evolutionary programming and support vector regression algorithms,* Expert Systems with applications **39**, 10700 (2012).

[38] M. dos Santos, B. de Mattos, and R. Girardi, *Aerodynamic coefficient prediction of airfoils using neural networks,* in *46th AIAA Aerospace Sciences Meeting and Exhibit* (2008) p. 887.

[39] C. Jie, S. Gang, and J. Xin, *Intelligent aerodynamic design for airfoil based on artificial neural network method,* in *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on,* Vol. 5 (IEEE, 2010) pp. 289–293.

[40] M. A. Naqvi, *Prediction of circulation control performance characteristics for super STOL and STOL applications,* Ph.D. thesis, Georgia Institute of Technology (2006).

[41] L. Chen, B. A. Cakal, X. Hu, and N. Thuerey, *Numerical investigation of minimum drag profiles in laminar flow using deep learning surrogates,* Journal of Fluid Mechanics **919**, A34 (2021).

[42] J. R. Raol, *Feed forward neural networks for aerodynamic modelling and sensor failure detection,* (1996).

[43] O. N. Diallo, *A predictive aircraft landing speed model using neural network,* in *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st* (IEEE, 2012) pp. 3D2–1.

[44] T. Awad, S. F. Rezeka, A. Saafan, and M. El-Gohary, *Control of aircraft maneuvers using neural network,* Alexandria Engineering Journal **45**, 509 (2006).

[45] J. M. Biannic, G. Hardier, C. Roos, C. Seren, and L. Verdier, *Surrogate models for aircraft flight control: some off-line and embedded applications,* Aerospace Lab , pages (2016).

[46] J. Kang and K. Choi, *Development of an artificial neural network control allocation algorithm for small tailless aircraft based on dynamic allocation method,* International Journal of Aeronautical and Space Sciences **23**, 363 (2022).

[47] M. Krügener, J. F. Zapata Usandivaras, M. Bauerheim, and A. Urbano, *Coaxial-injector surrogate modeling based on reynolds-averaged navier–stokes simulations using deep learning,* Journal of Propulsion and Power **38**, 783 (2022).

[48] S. Chen, H. Ge, J. Li, and M. Pecht, *Progressive improved convolutional neural network for avionics fault diagnosis,* IEEE Access **7**, 177362 (2019).

[49] J. Pelamatti, L. Brevault, M. Balesdent, E. Talbi, and Y. Guerin, *Overview and comparison of gaussian process-based surrogate models for mixed continuous and discrete,* High-Performance Simulation-Based Optimization **833**, 189 (2019).

[50] J. Fourier, *Mémoire sur la propagation de la chaleur dans les corps solides,* Nouveau Bulletin des Sciences de la Société Philomathique de Paris **6**, 112 (1808).

[51] L. Chwif, M. R. P. Barretto, and R. J. Paul, *On simulation model complexity,* in *Proceedings of the 32nd conference on Winter simulation* (Society for Computer Simulation International, 2000) pp. 449–455.

[52] L. Chwif, R. J. Paul, and M. R. P. Barretto, *Discrete event simulation model reduction: A causal approach,* Simulation Modelling Practice and Theory **14**, 930 (2006).

[53] S. Arora and B. Barak, *Computational complexity: a modern approach* (Cambridge University Press, 2009).

[54] R. J. Brooks and A. M. Tobias, *Choosing the best model: Level of detail, complexity, and model performance,* Mathematical and computer modelling **24**, 1 (1996).

[55] C. H. Papadimitriou, *Computational complexity* (John Wiley and Sons Ltd., 2003).

[56] R. J. Brooks and A. M. Tobias, *Simplification in the simulation of manufacturing systems,* International Journal of Production Research **38**, 1009 (2000).

[57] J. P. C. Kleijnen, *Statistical tools for simulation practitioners* (Marcel Dekker, Inc., 1986).

[58] R. R. Barton, *Metamodels for simulation input-output relations,* in *Proceedings of the 24th conference on Winter simulation* (ACM, 1992) pp. 289–299.

[59] T. W. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree, *Kriging models for global approximation in simulation-based multidisciplinary design optimization,* AIAA journal **39**, 2233 (2001).

[60] G. E. P. Box and N. R. Draper, *Empirical model-building and response surfaces.* (John Wiley & Sons, 1987).

[61] T. W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen, *Metamodels for computer-based engineering design: survey and recommendations,* Engineering with computers **17**, 129 (2001).

[62] J. P. C. Kleijnen and R. G. Sargent, *A methodology for fitting and validating metamodels in simulation,* European Journal of Operational Research **120**, 14 (2000).

[63] I. Guyon and A. Elisseeff, *An introduction to variable and feature selection,* Journal of machine learning research **3**, 1157 (2003).

[64] L. Pronzato and W. G. Müller, *Design of computer experiments: space filling and beyond,* Statistics and Computing **22**, 681 (2012).

[65] Y. Liu, Y. Shi, Q. Zhou, and R. Xiu, *A sequential sampling strategy to improve the global fidelity of metamodels in multi-level system design,* Structural and Multidisciplinary Optimization **53**, 1295 (2016).

[66] F. A. C. Viana, C. Gogu, and R. T. Haftka, *Making the most out of surrogate models: tricks of the trade,* in *ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (American Society of Mechanical Engineers, 2010) pp. 587–598.

[67] R. R. Barton, *Simulation metamodels,* in *Proceedings of the 30th conference on Winter simulation* (IEEE Computer Society Press, 1998) pp. 167–176.

[68] C. De Boor, K. Höllig, and S. Riemenschneider, *Box splines*, Vol. 98 (Springer Science & Business Media, 2013).

[69] J. H. Friedman, *Multivariate adaptive regression splines,* The annals of statistics , 1 (1991).

[70] R. Jin, W. Chen, and T. W. Simpson, *Comparative studies of metamodelling techniques under multiple modelling criteria,* Structural and multidisciplinary optimization **23**, 1 (2001).

[71] J. R. Koehler and A. B. Owen, *9 computer experiments,* Handbook of statistics **13**, 261 (1996).

[72] G. Matheron, *Krigeage d'un panneau rectangulaire par sa périphérie,* Note géostatistique **28** (1960).

[73] G. Li, S. Wang, H. Rabitz, S. Wang, and P. Jaffé, *Global uncertainty assessments by high dimensional model representations (HDMR),* Chemical Engineering Science **57**, 4445 (2002).

[74] S. M. Clarke, J. H. Griebsch, and T. W. Simpson, *Analysis of support vector regression for approximation of complex engineering analyses,* Journal of mechanical design **127**, 1077 (2005).

[75] C. N. Madu, *A fuzzy theoretic approach to simulation metamodeling,* Applied mathematics letters **8**, 35 (1995).

[76] D. A. Savic and W. Pedrycz, *Evaluation of fuzzy linear regression models,* Fuzzy sets and systems **39**, 51 (1991).

[77] H. Tanaka, S. Uejima, and K. Asai, *Linear regression analysis with fuzzy model,* IEEE Transactions on Systems Man and Cybernetics **12**, 903 (1982).

[78] L. A. Zadeh, *Fuzzy sets,* Information and control **8**, 338 (1965).

[79] S. Oh, W. Kim, W. Pedrycz, and B. Park, *Polynomial-based radial basis function neural networks (p-rbf nns) realized with the aid of particle swarm optimization,* Fuzzy Sets and Systems **163**, 54 (2011).

[80] M. P. Perrone and L. N. Cooper, *When networks disagree: Ensemble methods for hybrid neural networks,* Tech. Rep. (Institute for Brain and Neural Systems, Brown University, Providence (RI), 1992).

[81] B. Tunga and M. Demiralp, *A novel hybrid high dimensional model representation (hhdmr) based on the combination of plain and logarithmic high dimensional model representations,* in *Proceedings of the 12th WSEAS International Conference on Applied Mathematics* (Citeseer, 2007) pp. 157–161.

[82] J. Zhang, S. Chowdhury, and A. Messac, *An adaptive hybrid surrogate model,* Structural and Multidisciplinary Optimization **46**, 223 (2012).

[83] BruceBlaus, *Multipolar neuron,* .

[84] W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity,* The bulletin of mathematical biophysics **5**, 115 (1943).

[85] D. Hughes and N. Correll, *Distributed machine learning in materials that couple sensing, actuation, computation and communication,* arXiv preprint arXiv:1606.03508 (2016).

[86] D. S. Broomhead and D. Lowe, *Multivariable functional interpolation and adaptive networks,* Complex systems **2**, 321 (1988).

[87] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural network design* (Martin Hagan, 2014).

[88] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction,* (2009).

[89] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators,* Neural networks **2**, 359 (1989).

[90] A. N. Kolmogorov, *On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition,* in *Doklady Akademii Nauk*, Vol. 114 (Russian Academy of Sciences, 1957) pp. 953–956.

[91] R. Eldan and O. Shamir, *The power of depth for feedforward neural networks,* in *Conference on learning theory* (PMLR, 2016) pp. 907–940.

[92] A. R. Barron, *Universal approximation bounds for superpositions of a sigmoidal function,* IEEE Transactions on Information theory **39**, 930 (1993).

[93] L. Trapp and G. Oliveira, *Aircraft thrust reverser cascade configuration evaluation through CFD,* in *41st Aerospace Sciences Meeting and Exhibit* (2003) p. 723.

[94] S. de Cuendas, *The Future Air Navigation System FANS B,* Safety First (2007).

[95] A. F. O. Support and Services, *Getting to grips with FANS,* (2007).

[96] F. A. Administration, *SAFO: Runway Assessment and Condition Reporting* (2016).

[97] W. G. Cochran, *Sampling techniques* (John Wiley & Sons, 2007).

[98] D. W. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters,* Journal of the society for Industrial and Applied Mathematics **11**, 431 (1963).

[99] F. D. Foresee and M. T. Hagan, *Gauss-newton approximation to bayesian learning,* in *1997 IEEE international conference on neural networks* (1997) pp. 1930–1935.

[100] D. Livingstone, *Artificial neural networks: methods and applications* (Springer, 2008).

[101] R. G. Pontius, O. Thontteh, and H. Chen, *Components of information for multiple resolution comparison between maps that share a real variable,* Environmental and Ecological Statistics **15**, 111 (2008).

[102] R. C. Geary, *The ratio of the mean deviation to the standard deviation as a test of normality,* Biometrika **27**, 310 (1935).

[103] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,* (2001).

[104] A. Agrawal, P. D. Deshpande, *et al.*, *Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters,* Integrating Materials and Manufacturing Innovation **3** (2014).

[105] A. Airbus, *Getting to grips with aircraft performance,* Cust. Serv. Blagnac (2002).

[106] W. Blake, *Jet transport performance methods,* Boeing Commercial Airplanes , 6 (2009).

[107] J. Roskam and C. T. E. Lan, *Airplane aerodynamics and performance* (DARcorporation, 1997).

[108] A. A. Alroqi and W. Wang, *Comparison of aircraft tire wear with initial wheel rotational speed,* International Journal of Aviation, Aeronautics, and Aerospace **2**, 2 (2015).

[109] B. Ewers, J. Bordeneuve-Guibé, *et al.*, *Expert supervision of an anti-skid control system of a commercial aircraft,* in *Proceedings of the 1996 IEEE International Symposium on Intelligent Control* (IEEE, 1996) pp. 420–425.

[110] H. Song, B. Fang, and P. Wang, *Research and applications of immune pid adaptive controller in anti-skid braking system for aircraft,* in *2009 International Conference on Information Engineering and Computer Science* (IEEE, 2009) pp. 1–5.

[111] J. A. Tanner and S. M. Stubbs, *Behavior of aircraft antiskid breaking systems on dry and wet runway surfaces: A slip-ratio-controlled system with ground speed reference from unbraked nose wheel,* (1977).

[112] S. Li and T. Kawabe, *Slip suppression of electric vehicles using sliding mode control method,* Intelligent Control and Automation **4**, 327 (2013).

[113] 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 **42**, 55 (2000).

Appendix A
Elements of landing mechanics and performance

Definitions of landing distance

Aircraft landing performance is directly measured by the distance effectively covered during landing. This parameter, as seen in Figure A.1, is made up of two components. It is in fact the sum of an airborne distance, covered between screen height and touch down, and a ground roll distance, which refers to the whole part of landing where the aircraft is on ground. A shorter landing distance is clearly preferable, as it both cuts down time for airline operations and prevents any circumstance of runway overrun.
The ideal landing distance as given in flight manuals for each aircraft model, is calculated in ideal conditions, with the aircraft at the correct speed at screen height. This is called the actual landing distance (ALD) or calculated landing distance. Depending on external conditions, a required landing distance (RLD) is computed from the ALD by taking into account a suitable safety margin. This measurement must be known before landing and it is essential to ensure it to be smaller than the landing distance available (LDA), which basically corresponds to the runway length. [105] [106] [107]
Several factors can have an impact on the effective landing distance:

- Aircraft weight increases stall speed and the inertia of the aircraft during braking.
- Head wind can be beneficial within certain limits, as it increases true airspeed. The opposite goes for tail wind.
- Landing surface contamination decreases the efficacy of mechanical braking.
- A positive runway slope is beneficial, as there is an additional gravity component during ground braking. Negative runway slope instead delays touch down and adds a harmful gravitational component during ground roll.
- High-lift devices can greatly affect the landing procedure according to the chosen configuration.
- System failures can have several negative effects, including the loss of symmetry in the landing path.

Governing equations

In the study of landing mechanics, it is convenient to make a breakdown of the forces along the air-path axis and its perpendicular direction. With this idea in mind, diagrams of the forces acting on the aircraft during landing can be identified in Figures A.2 and A.3. From these, the governing equations of flight mechanics can be obtained and implemented in the calculation of aircraft landing performance.
The first diagram is referred to the airborne phase of landing. It is valid for both descent and flare, with the only difference of a continuously varying flight path angle γ in the latter segment. Here the only forces coming into play are those due to aerodynamics, engine thrust and gravity. The sum of forces along both axes can be thus expressed:

$$\sum F_X = X_{PROP} - D - m \cdot g \cdot sin(\gamma) \tag{1}$$

$$\sum F_Z = Z_{PROP} + L - m \cdot g \cdot cos(\gamma) \tag{2}$$

where X_{PROP} and Z_{PROP} are the longitudinal and vertical components of engine thrust, D and L are the aerodynamic drag and lift, while the last term in each equation is the relative contribution of the gravitational force on each axis.

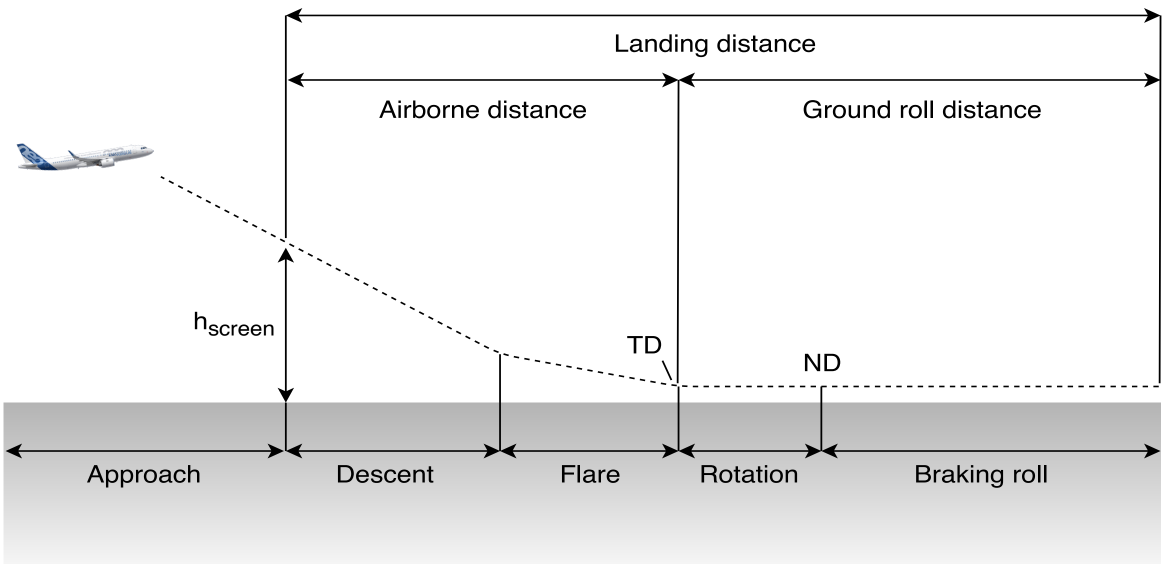


Figure A.1: Scheme of the landing phase, with breakdown of its components and landing distances.

During the ground phase, identified in Figure A.3, additional forces due to the interaction of the landing gear with the runway surface have to be taken into account. The sums of forces in this case are given as:

$$\sum F_X = X_{PROP} - D - F_{BRAKE} - F_{ROLL} - m \cdot g \cdot sin(\gamma) \tag{3}$$

$$\sum F_Z = Z_{PROP} + L + R_{MAIN} + R_{NOSE} - m \cdot g \cdot cos(\gamma) \tag{4}$$

where F_{BRAKE} is the friction force between the runway and the braking wheels of the landing gear, while F_{ROLL} is the rolling resistance on the non-braking wheels. A gravity component is usually required in both equations as, having assumed that the aircraft is moving parallel to the ground, it accounts for a possible slope in the runway shape. The choice of splitting the ground forces between main landing gear and nose wheel has been made in view of the application presented in the next section.
Friction forces can be calculated by the well-known Coulomb model of friction, generally expressed as

$$T = \mu \cdot R \tag{5}$$

with the tangential force T being given as the product of a friction coefficient μ and a reaction force R that is perpendicular to the surfaces in contact. This law hence provides an useful relationship between the ground forces in equations (3) and (4), which can be differently exploited according to the specific case study.

Braking performance

Under a physical point of view, as seen in the previous paragraph, the landing problem is relatively straightforward. However, the features of the braking action and hence the performance of such manoeuvre are heavily dependent on (auto) pilot choices and state of the runway.
Concerning the first aspect, the main parameters that the pilot can control are landing speed, aircraft attitude, high-lift device configuration and intensity of the braking action. The first two are bound by constraints on stall characteristics and structural requirements. Furthermore, only high-lift configurations 3 and Full, corresponding to the two most extended positions of flaps and slats, are allowed during landing.
With respect to the braking action, there are normally three increasing levels of intensity that can be selected during landing: LOW, MED and MAX. These correspond to effectively increasing braking levels, as well as different activation delays from deployment of the spoilers. With LOW and MED, braking intensity is increased by steps towards a maximum value, where in the latter case a check on pitch angle is also performed. Conversely, the MAX setting directly increases the braking action to the maximum level available, as soon as spoilers are out. Whether the braking action is performed by mechanical torque on the landing gear wheels, by reverse thrust or a combination of both, it is usually left as an automatic choice of the aircraft systems.
The state of the runway is a crucial parameter that heavily affects the performance of a landing. Runway

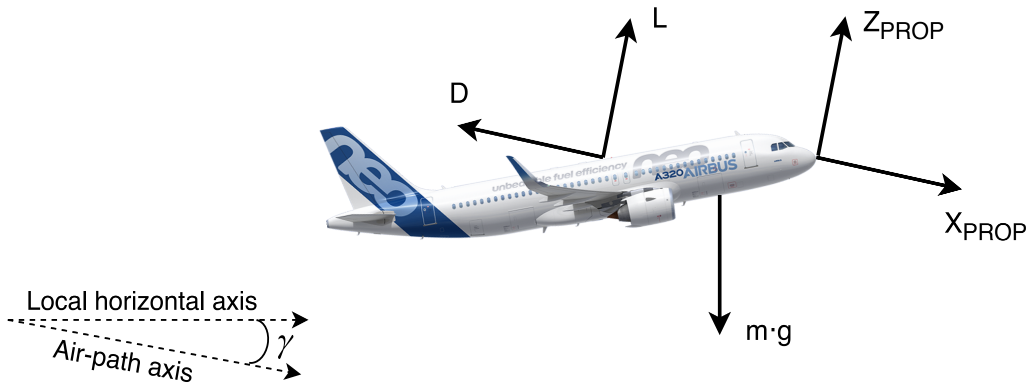


Figure A.2: Diagram of the forces acting on an aircraft during the airborne phase of landing.

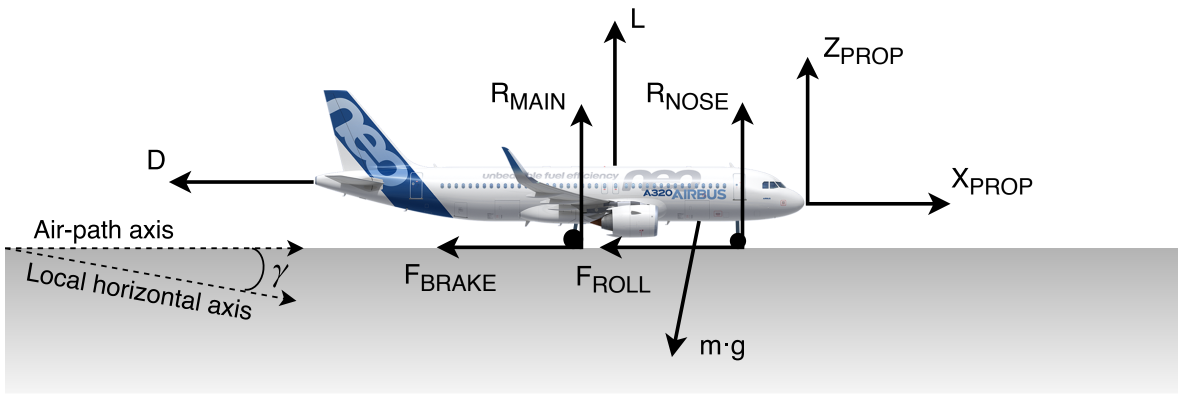


Figure A.3: Diagram of the forces acting on an aircraft during landing roll after the moment of nose down, with a runway slope γ.

surfaces can be contaminated by a number of environmental agents such as soil, mud, water and ice whose specific characteristics are not always easy to identify before a landing takes place. From a physical point of view, this translates into a varying value of friction coefficient between the runway surface and the landing gear wheels, in turn affecting the aircraft braking performance.
The identification of the friction coefficient, however, is a complex problem. Additional parameters such as aircraft speed and tire pressure lead to strongly nonlinear dependencies whose investigation is out of the scope of this work. One variable that needs to be mentioned, however, is indeed slip ratio. Such parameter quantifies the relative sliding occurring between a rotating wheel and the surface it rolls on. Though no universal definition exists, an intuitive description of longitudinal slip ratio is given by [108]:

$$\lambda_S = \frac{V - r_e\omega}{V} \tag{6}$$

where V is the aircraft forward speed and ω the wheel rotational speed. The radius r_e is the effective wheel radius, which takes into account for deformation of the pneumatic, due to both the rolling motion and the weight of the aircraft itself. From the above definition it is clear that a total absence of slipping could only be attained in the case of a pure rolling motion (where forward speed equals the external tangential speed) without any wheel deformation.
Slip ratio is always kept under control, in order to achieve an optimal value of friction coefficient (see Figure A.4) and hence braking distance. For that reason, anti-skid systems are implemented to monitor the value of slip ratio and consequently adjust the braking torque on landing gear wheels. This action not only optimizes the rolling phase, especially on slippery surfaces, but also prevents the brakes from overheating.
In the context of braking performance, there are different definitions of braking action according to the factors that determine its efficiency. It is referred to torque-limited braking when, for a set pressure in the hydraulic system, the braking torque generated is not sufficient to bring the wheels to a halt. A more frequent scenario is when the supplied torque is effectively high enough to spin down the wheels, but the obtained braking force between runway surface and tyre is lower than the one targeted by the braking system. This

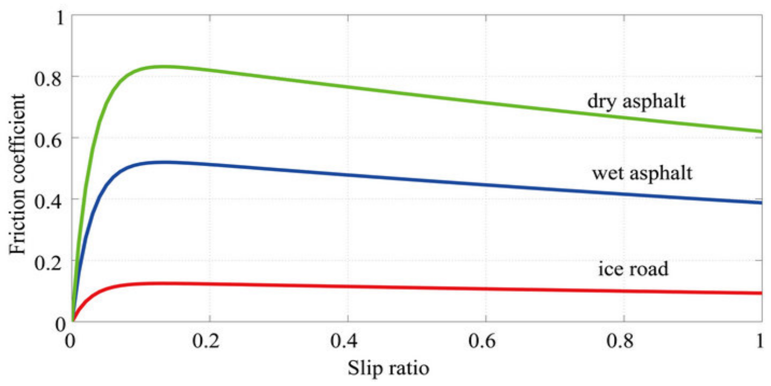


Figure A.4: Relationship between slip ratio and tire friction coefficient, as found in [112] for different types of surfaces.

case can be referred to as a friction-limited braking, since the limiting factor of the manoeuvre is the friction force. In these situations, which are typical of contaminated runways, an anti-skid system is thus activated to efficiently manage the roll phase (hence the alternative naming of anti-skid-limited braking). [109] [110] [111]

Appendix B
Sampling process

This Appendix expands the content of Section 5.2.2, which reported the main aspects of the sampling process. Here more details are provided on the link between the observed sampling domain and the chosen methodology, as well as on the results of the process itself.

Analysis of the sampling domain

The objective of the sampling phase is to obtain a dataset as complete and uniform as possible, so as to enable a training and validation of metamodels that are representative of their corresponding originals. Completeness of the data has been provided in previous steps with sound choices on each step of the process, hence the sampling should simply make sure that the value ranges covered by each variable be sufficiently represented. For what concerns the uniformity, several approaches are possible according to the type of design desired for the dataset. In principle, it is reasonable to seek a space filling design that would produce uniformity in an N-dimensional sense, in order to implement a balanced dataset in the training process.

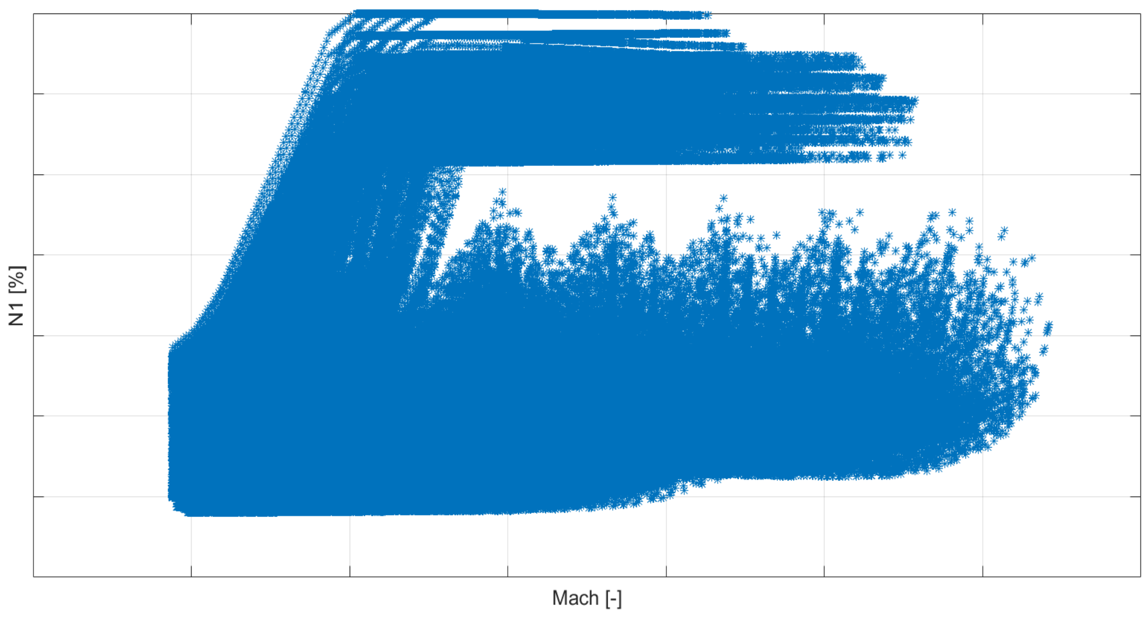


Figure B.1: Distribution of Mach number and engine rating values before final sampling.

There are two separate aspects to take into account in the choice of a sampling method. The first is the high redundancy of points in certain zones of the N-dimensional space of samples. This does not come as a surprise, since all simulations are performed between the same two moments of a landing and sometimes only slightly differ by their initial conditions. For instance, it is expected to find a high number of points corresponding to a low speed rather than those at high speeds, where the initial values have a higher variability. A visual clue of this fact is provided by Figure B.1, where all points available at this stage of sampling are plotted for the two variables Mach and N1. Evident variations in density of samples can be spotted across the 2-dimensional domain, highlighting zones where different amounts of data are available. This is due to the nature of the problem itself, which leads to uneven patterns throughout the domain. In particular it is possible to follow, in the middle of the graph, the drops in engine rating corresponding to different initial speeds

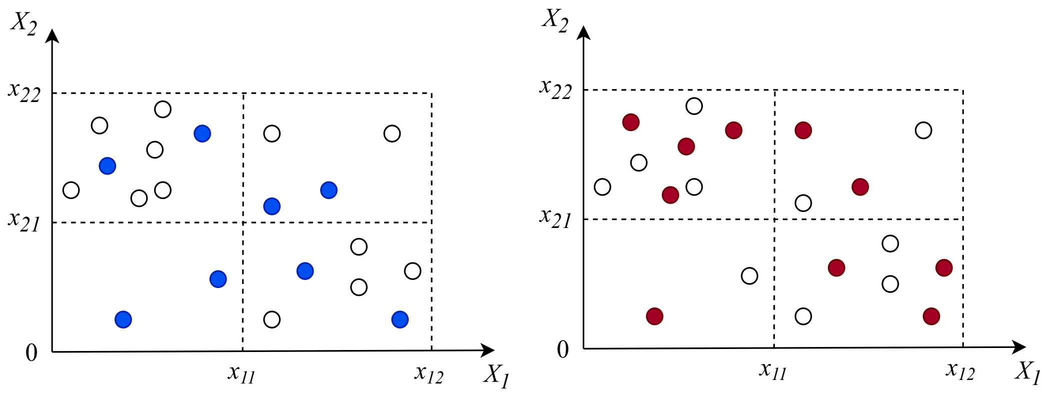


Figure B.2: Comparison of equal (left) and proportional (right) allocation in stratified sampling on a 2-dimensional domain.

at ND. Above that, instead, it can be observed how the trends corresponding to Max Reverse are constant in N1 for most of the pattern, to then drop at a lower speed.
The second aspect to take into account is the difference between one-dimensional and N-dimensional uniformity. An ideally balanced training dataset would cover the whole range of values that might be input to the models throughout their operation, and the samples would be given at a step that guarantees the sought modeling accuracy. A 1-dimensional uniformity would mean that each variable in the dataset has an uniform distribution on its range of values. Multidimensional uniform distributions instead, should be understood as groups of N-dimensional points that are equally spaced in an N-dimensional space. The problem is that uniformity in dimension 1 (on each variable) does not guarantee N-dimensional uniformity in the same dataset (on all variables). This is the case with the current work, due to the uncontrollable nature of some variables of the problem. These are the ”continuous” parameters of speed (GS, TAS, Mach) and those of engine rating (N1, EPR) that could not be discretely set when launching landing simulations. This aspect is clarified by Figure B.1, which shows some significant gaps in the domain formed by Mach and N1. Therefore, even obtaining a uniform distribution on the two single variables (which is still not the case here) would never result in an overall uniform domain with the two variables combined.

Sampling methodology

Several sampling techniques are available in the literature and the selection of a suitable method depends on both the characteristics of the problem and the features sought in the sampled dataset. Techniques such as random, latin hypercube and orthogonal sampling [113] were discarded, to avoid the risk of a scattered dataset and allow for a tighter control over its characteristics. The procedure implemented here is rather based on the concept of stratified sampling [97], which is often applied in order to reduce variance in the dataset, compared to a simple random sampling. According to this method, an N-dimensional space is divided into an arbitrary number of N-dimensional cells (strata) and a certain number of samples is taken from every cell. As explained more intuitively in Figure B.2, different strategies can be adopted for the number of points that are selected in each cell. With an equal allocation strategy, a fixed number of samples is taken per cell. Proportional allocation instead aims at selecting a number of samples proportional to the amount of points that is found in the cell. After several tests, a hybrid stratified approach was implemented. The process was performed by only sampling the domains of the discrete variables with equal allocations, effectively excluding those representing speed (GS, TAS, Mach) and engine rating (N1, EPR).
The approach serves two main objectives. On one hand uniformity is maintained in the dataset for the single discrete variables, but also for the continuous ones whose features already displayed in Figure B.1 make it very complex to achieve a better distribution. On the other hand, such complex dispersion of the continuous variables is the expression of a natural tendency of the physical phenomenon, which was decided to be maintained in the process. Indeed the choice is partially arbitrary: it would have been equally reasonable to demand a perfectly balanced dataset, in order to produce a model with (ideally) constant performance in the whole domain of definition. Here an engineering choice was rather made to ensure a higher number of points (hence better metamodel performance) in those parts of the operation envelope where the model is likely to work more often.
The sampling process was performed on four separate batches of data, which were retrieved from simulations and sampled individually. These were split according to high-lift configuration (3 and Full) and whether

a case of overweight landing, in order not to overload the postprocessing phase.

Results

The histograms of dataset distributions for the relevant parameters are reported in Figure B.3 to B.5, for both before and after sampling. It can be observed how the shape of the distributions is essentially unaffected, except being scaled down by a factor 10.
The evident non uniformities obtained on some parameters are only apparent and are due to different reasons. The parameter Rev, which indicates whether the engine is in operating in reverse, is explained by the choice on the variable Rev type, which instead tells whether it is a case of direct thrust, maximum reverse or idle reverse. It has been chosen, in fact, to give an equal distribution of the three cases to the dataset, which in turn explains the distribution on Rev.
The other clear disconformities are in Weight and CG. In both cases this is justified by the choices on the two variables, which are given specific values for each aircraft and engine type. The last two features are summarized by the variable AC model, which refers to the aircraft model used in the simulation software (hence a specific combination of aircraft type and engine type). This parameter has also been kept uniformly distributed not to give a specific (and unjustified) importance to certain aircraft models during training. As a matter of fact, this objective was a driving feature of the sampling process.
Lastly, the distributions of GS and N1 remain unchanged. The natural trend of the problem is essentially left untouched and the models will be trained more in depth in the areas of more frequent operation.

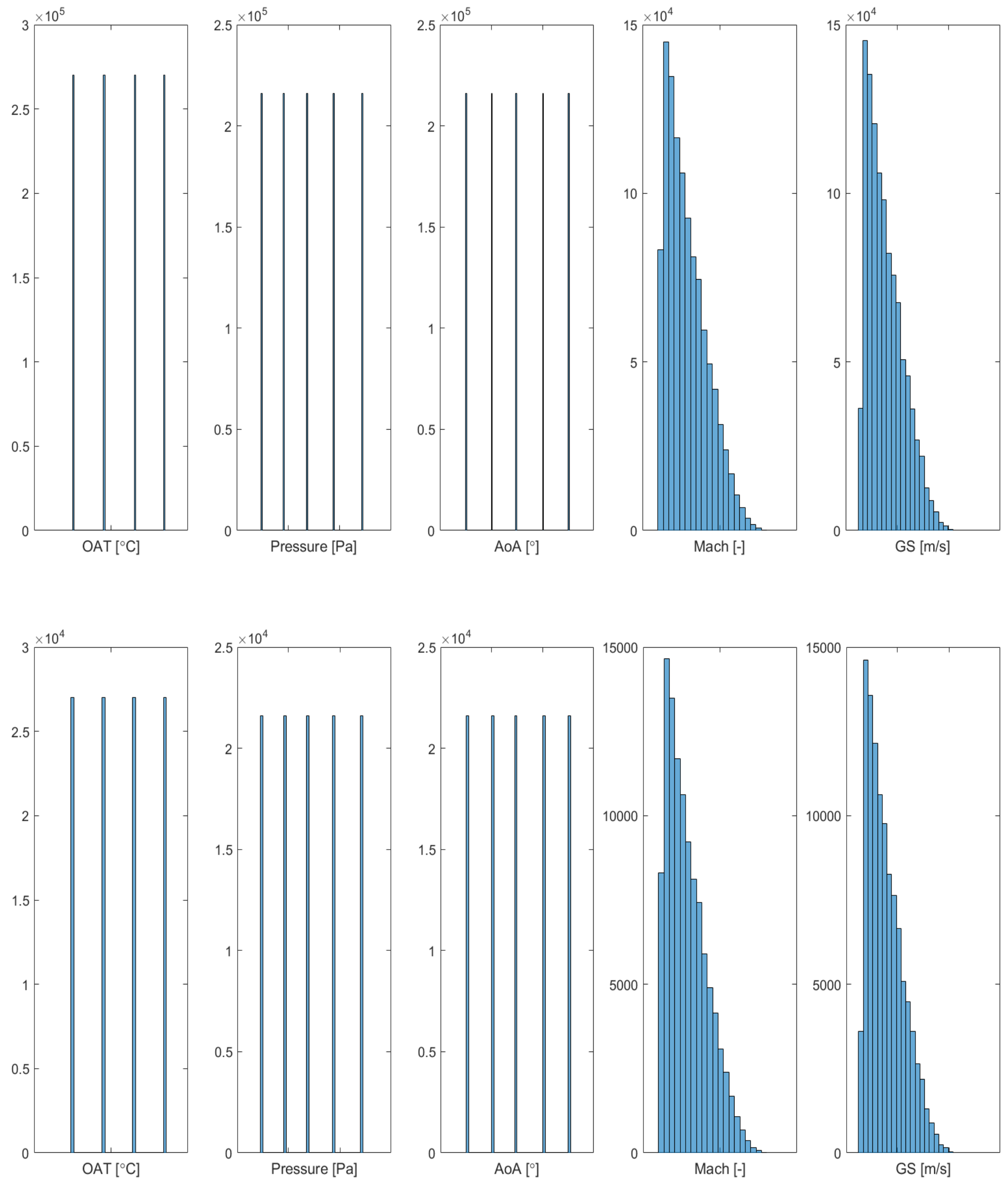


Figure B.3: Distribution before (top) and after (bottom) sampling: outside air temperature, pressure, angle of attack, Mach and GS.

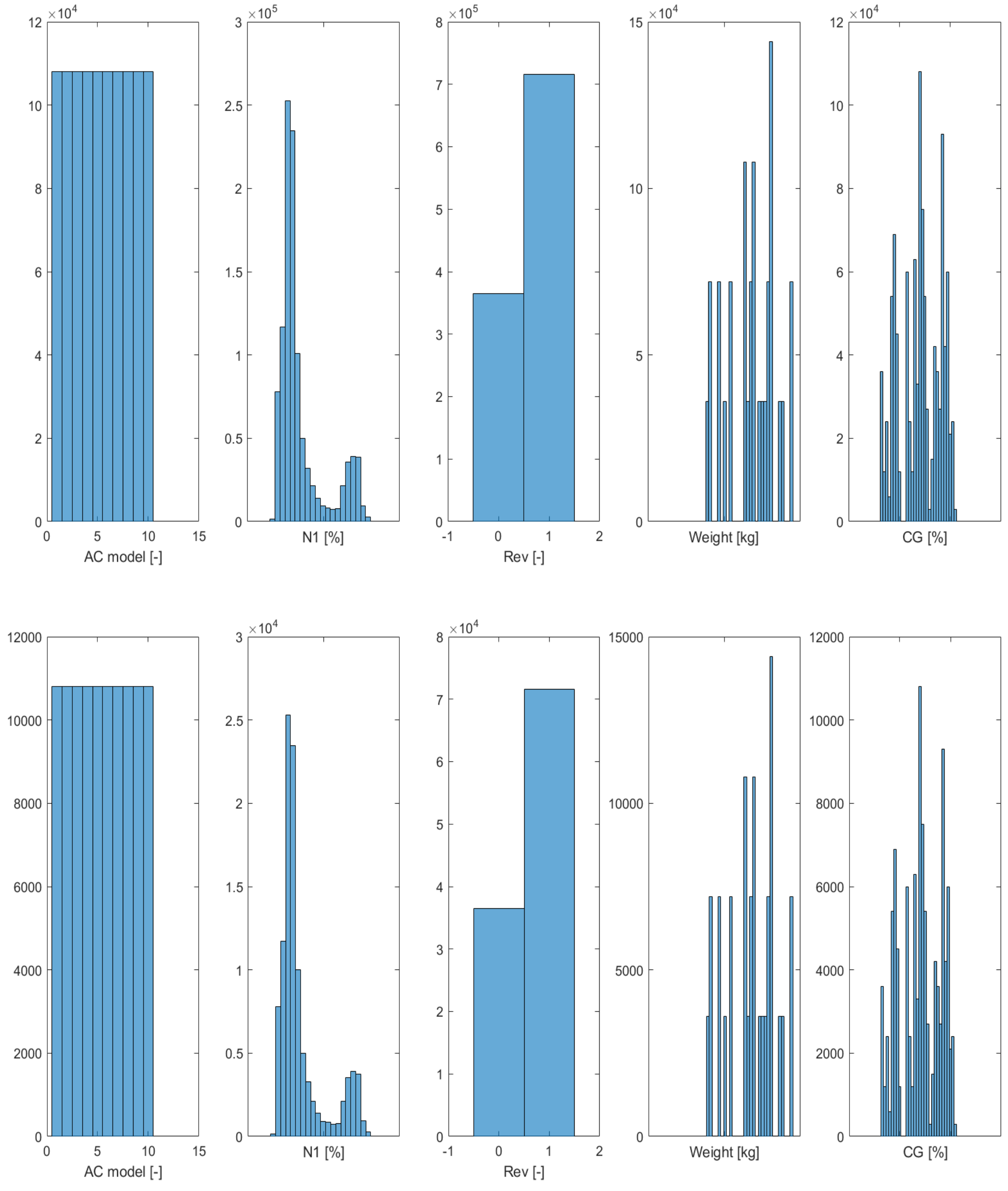


Figure B.4: Distribution before (top) and after (bottom) sampling: AC model, engine rating, reverser activation, mass and CG.

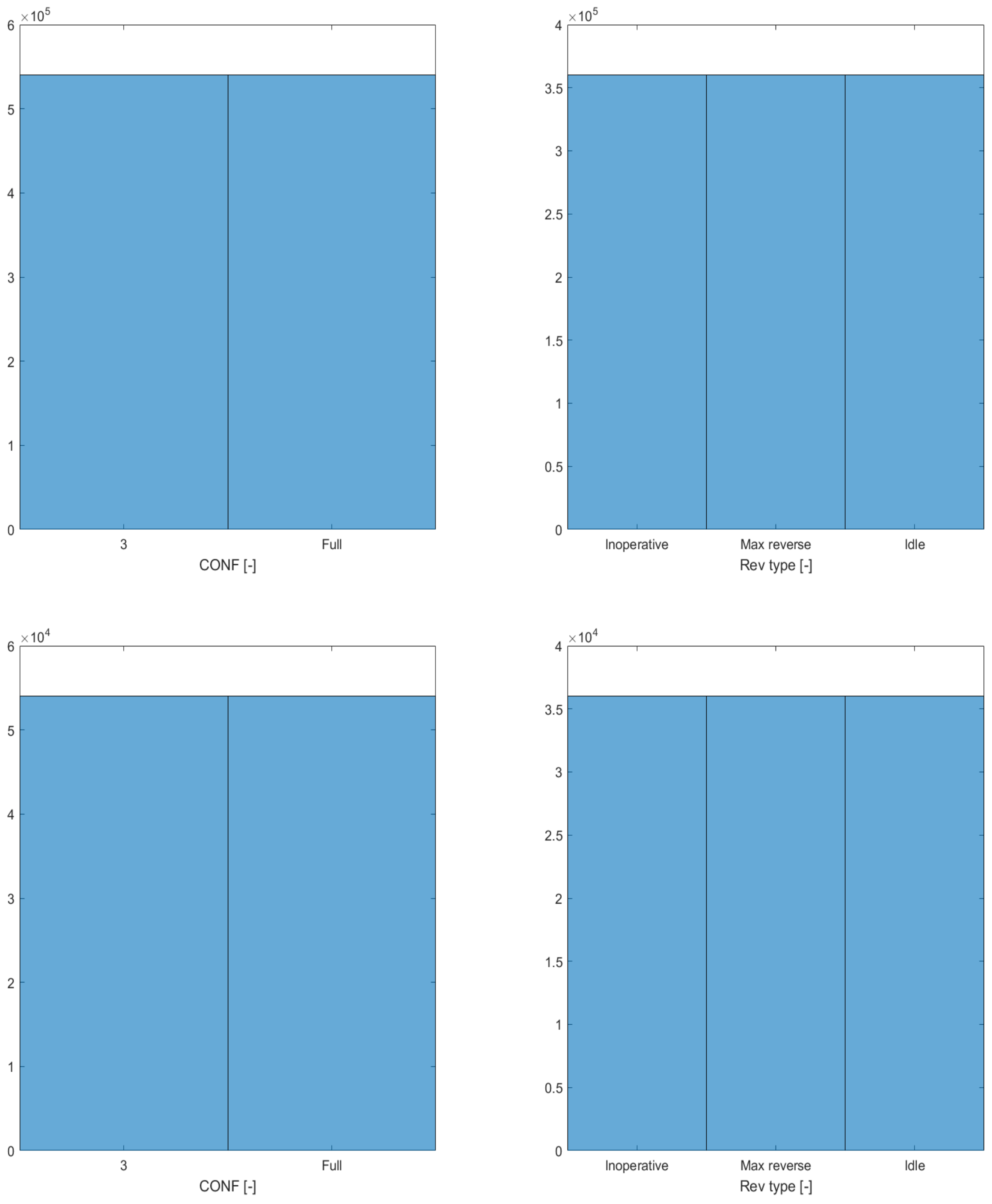


Figure B.5: Distribution before (top) and after (bottom) sampling: high-lift configuration and thrust reverser status.

Appendix C
Relative error of the baseline model

The results shown in 6.1 on the performance of the baseline neural networks demonstrate that the model structure is already capable of achieving acceptable approximation behaviour across the domain of interest, with some differences between the output variables. The results are here expanded by providing the plots of relative error for the four variables.

Outliers represent an issue for the interpretation of the results, as it can be seen from the scale of the plot of relative error in Figures C.1 and C.3. To mitigate this problem, the same plots have been traced for a subset of the training and validation datasets, which have been selected with the interquartile range rule. According to this method, outliers can be detected as any points lying outside of the range [Q1-1.5*IQR, Q3+1.5*IQR], with Q1 and Q3 the lower and upper quartiles and IQR the difference between the two (interquartile range). The 1.5 multiplier, which would allow to include approximately 99% of points for a normal distribution, has been adjusted here to show a meaningful portion of the error distributions.

A 3.5 IQR multiplier has been implemented to obtain the results shown in Figure C.2 (for reference, a multiplier of 3 is normally considered to identify extreme outliers). As already pointed out in 6.1, the error on drag is observed to be contained within 10% in most of the domain. Moreover, a better interpretation can now be given on the results of lift, where the very long tails of the distribution have been limited to values of around 125%. It is worth highlighting that the current choice of IQR multiplier leads to a coverage of approximately 98% and 91% of the Drag and Lift distributions respectively.

In the same way as for the aerodynamic model, the plots shown in Figure C.4 have been obtained via the application of the IQR rule to limit the presence of long tails in the error distributions of the engine model. A multiplier of 4.5 has been implemented here, covering approximately 97.5% and 92% of the X_{PROP} and Z_{PROP} distributions respectively, thus allowing for a more meaningful interpretation of the results.

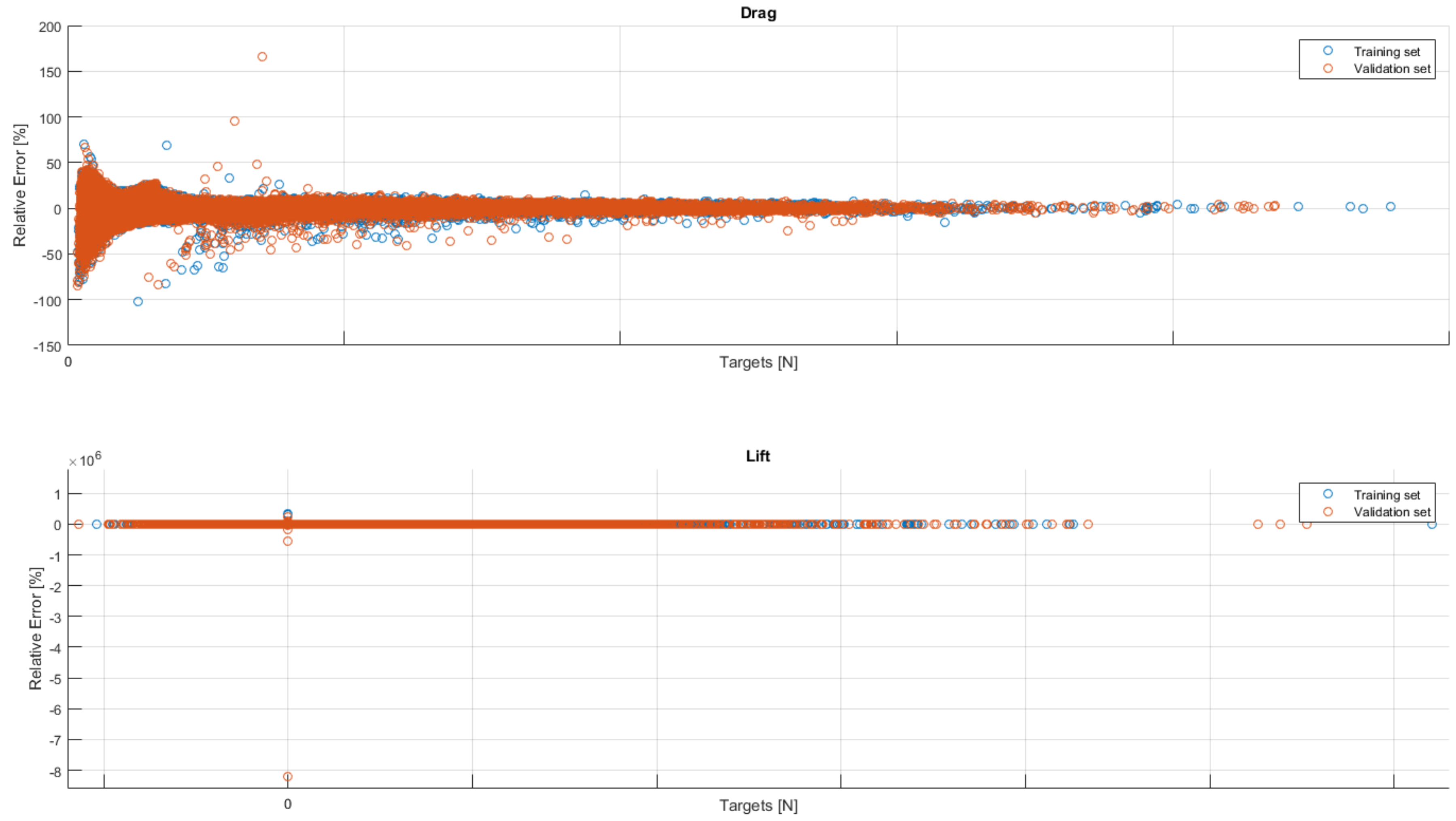


Figure C.1: Baseline aerodynamics network relative error against target values in the training and validation datasets. Data are plotted for both output variables Drag and Lift.

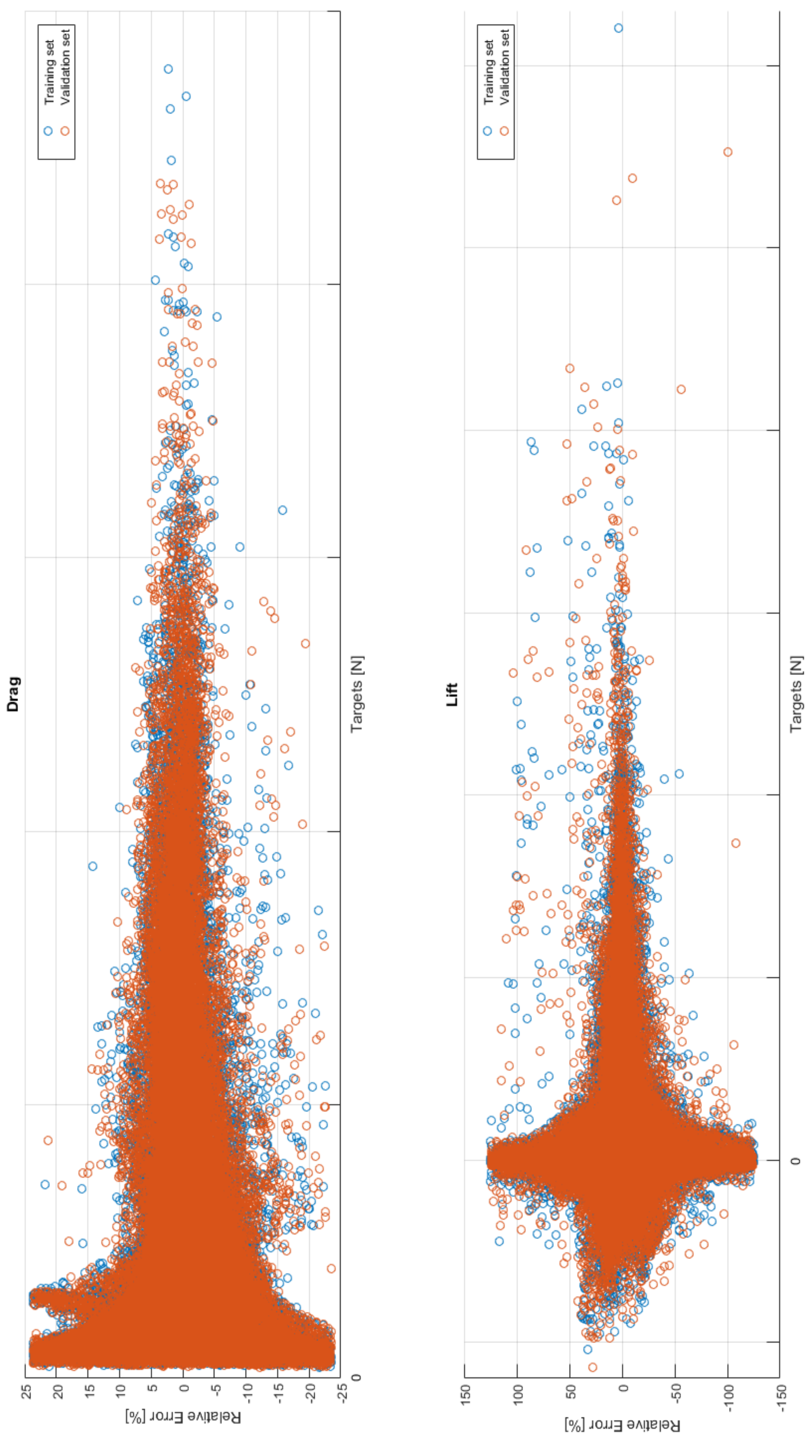


Figure C.2: Baseline aerodynamics network relative error against target values in the training and validation datasets. Outliers have been removed by means of the interquartile range rule.

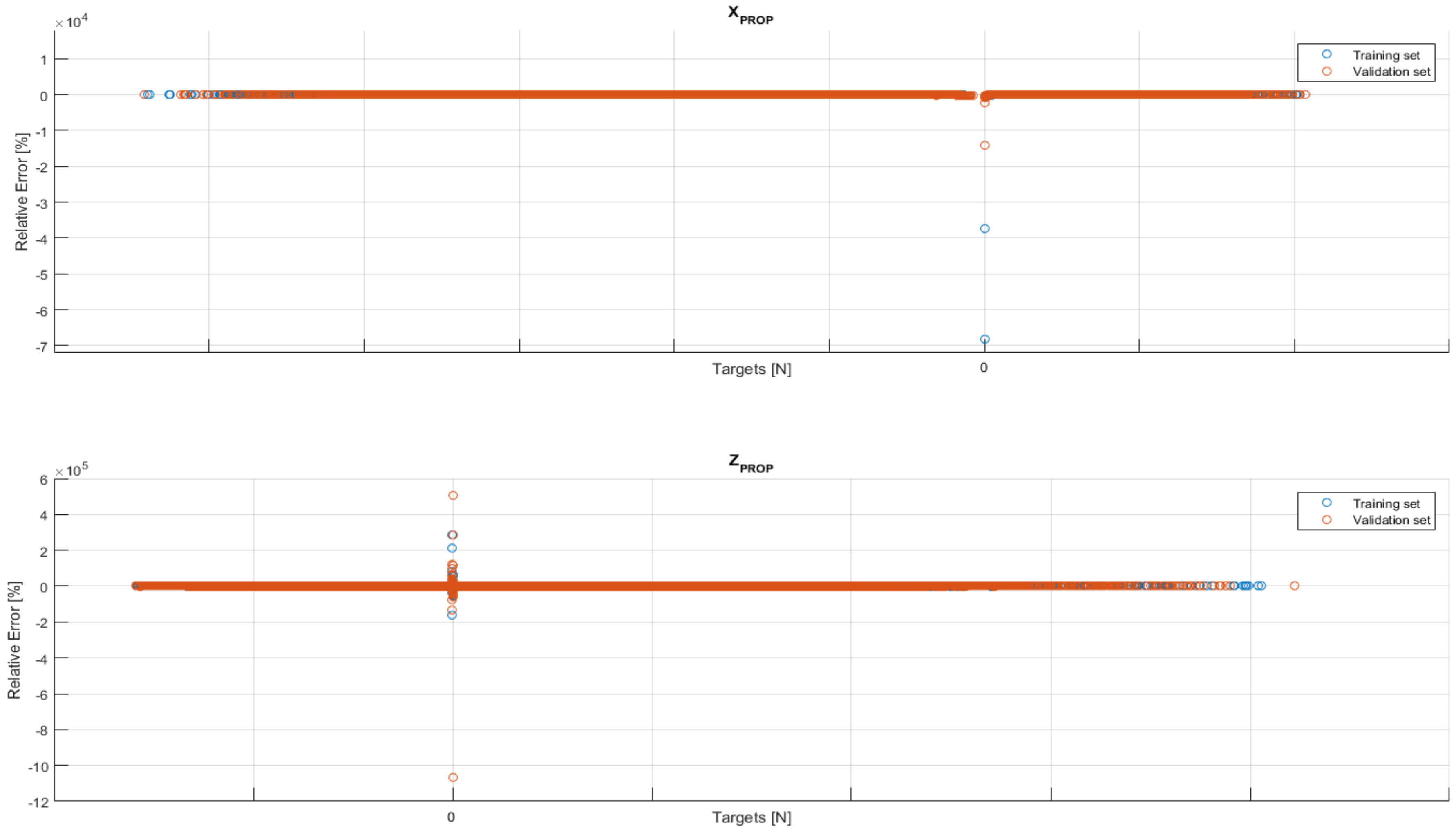


Figure C.3: Baseline engine network relative error against target values in the training and validation datasets. Data are plotted for both output variables X_{PROP} and Z_{PROP}.

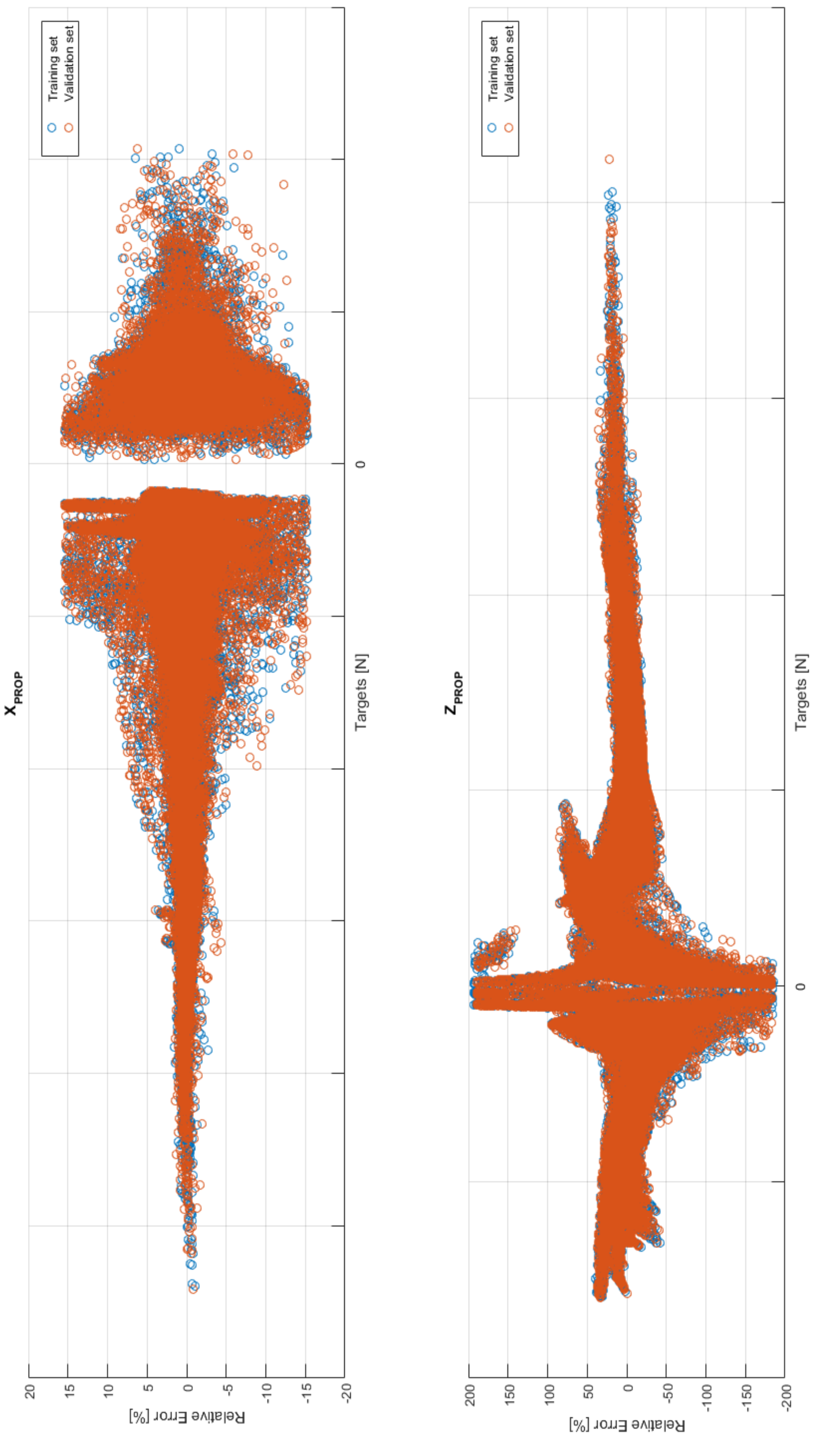


Figure C.4: Baseline engine network relative error against target values in the training and validation datasets. Outliers have been removed by means of the interquartile range rule.

Appendix D
Polynomial models

Background

Given an input vector $\underline{x}$ of dimensionality d and a set of (unknown) real coefficients c_α, a polynomial of degree k and monomials x^α is defined as:

$$\hat{y} = \sum_{\|\alpha\|_1 \leq k} c_\alpha x^\alpha \tag{7}$$

with

$$\|\alpha\|_1 = \sum_{i=1}^{d} \alpha_i \tag{8}$$

$$x^\alpha = \prod_{i=1}^{d} {x_i}^{\alpha_i} \tag{9}$$

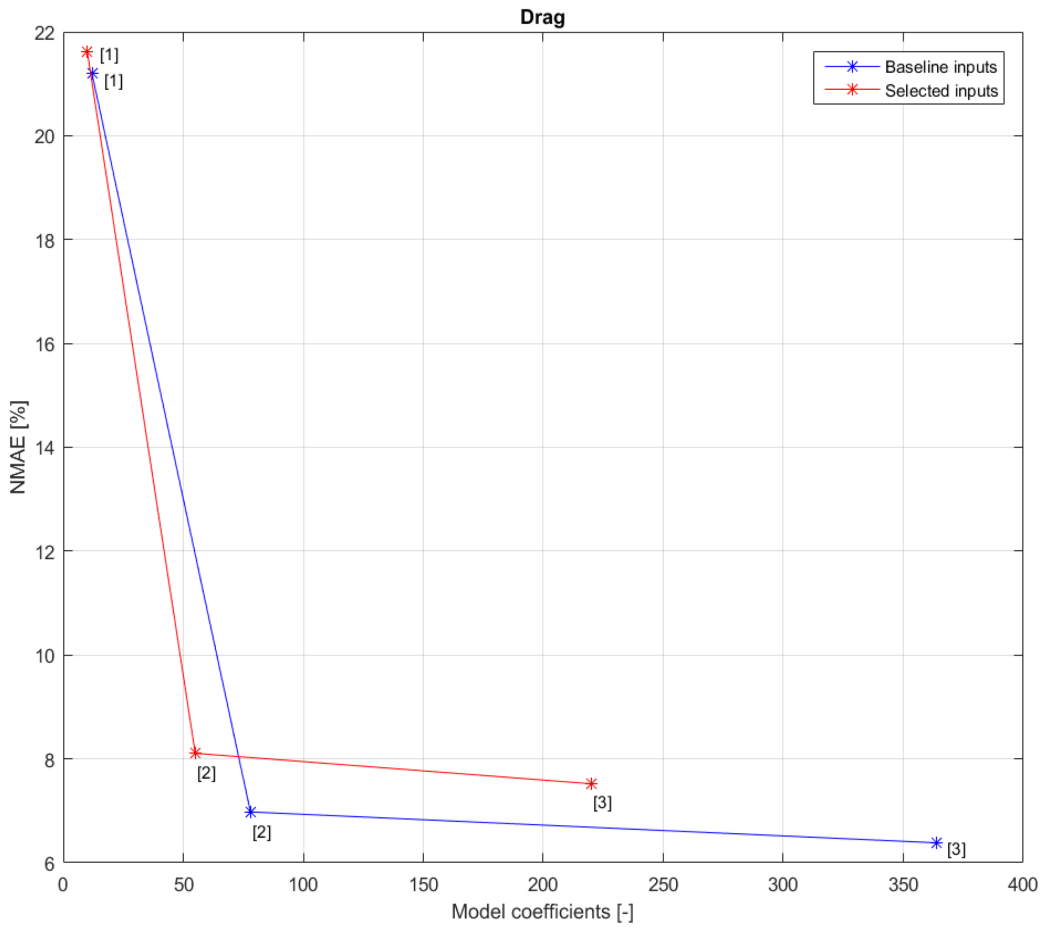
One of the main disadvantages of such models is that the number of terms grows rapidly with the number of inputs and degree of the polynomial, following to the relationship

$$N_T = \frac{(k+d)!}{k!d!} \tag{10}$$

An excessive increase of polynomial degree to overcome accuracy issues is generally discouraged, as local oscillations (Runge's phenomenon) are likely to occur at higher order. Moreover, according to Simpson [61], polynomial models should not be used for applications with more than around 10 input variables. The present study is hence a limit case in these terms.
Polynomial models are linear in the parameters (model coefficients) and therefore allow for linear regression techniques. An ordinary least squares method has been implemented based on the consolidated work of Cecen, mentioned in [104]. As the technique has the property of scale invariance, meaning that the constant coefficients of the model can act as scaling factors, normalization of inputs was not required for training.
The advantages and limitations mentioned so far, as well as the memory constraints faced with Matlab, thus led to the need of using polynomials of maximum degree 3. Only specific cases, with a selected number of input features, also allowed the fitting of polynomials of degree 4. Additional models of higher degree could have been obtained, however the number of coefficients required (equation (10)) would considerably exceed the memory targets expected in the work. This aspect is especially true when considering that a polynomial model can only output one variable, in contrast with neural networks. In fact, two different polynomials have to be fitted for each Aerodynamics and Engine model.

Performance of the trained models

The results of the trained polynomial models in terms of validation NMAE against required number of coefficients (or polynomial terms) are shown in Figures D.1 and D.2.
With respect to the results of neural network models, a substantially poorer performance is obtained in all cases, with the accuracy of first degree models standing out negatively over the others. Second and third degree polynomials, on the other hand, achieve similar approximation levels, especially in the case of aerodynamic modeling. Stronger deviations are observed for the engine variables when selected input features are implemented. It is interesting to notice how the same aerodynamic modeling difficulties found in 6.4 are encountered here with the selected input cases. The baseline input set is, in fact, sensibly outperformed only in the engine modeling. Lastly it can be seen how the fourth degree polynomials, despite a considerable number of terms, lead to worse results than those of degree 3. This can be ascribed to the phenomenon of unwanted oscillations in high-degree fitting functions.
As explained in the main body of the report, these results led to the conclusion that the above models only stand as a means for comparison with the trained neural networks, without any practical viability in aircraft applications.



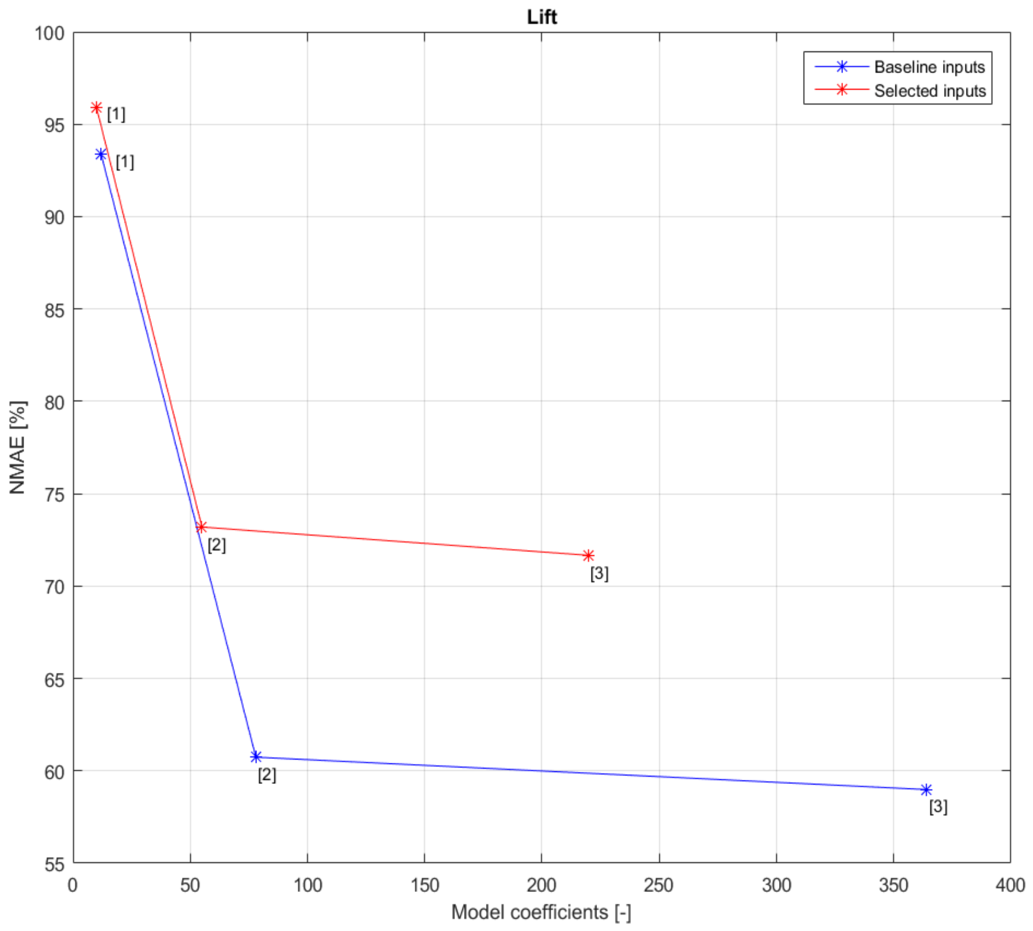


Figure D.1: Study of polynomial models for the aerodynamic model, with polynomial degrees reported between brackets.

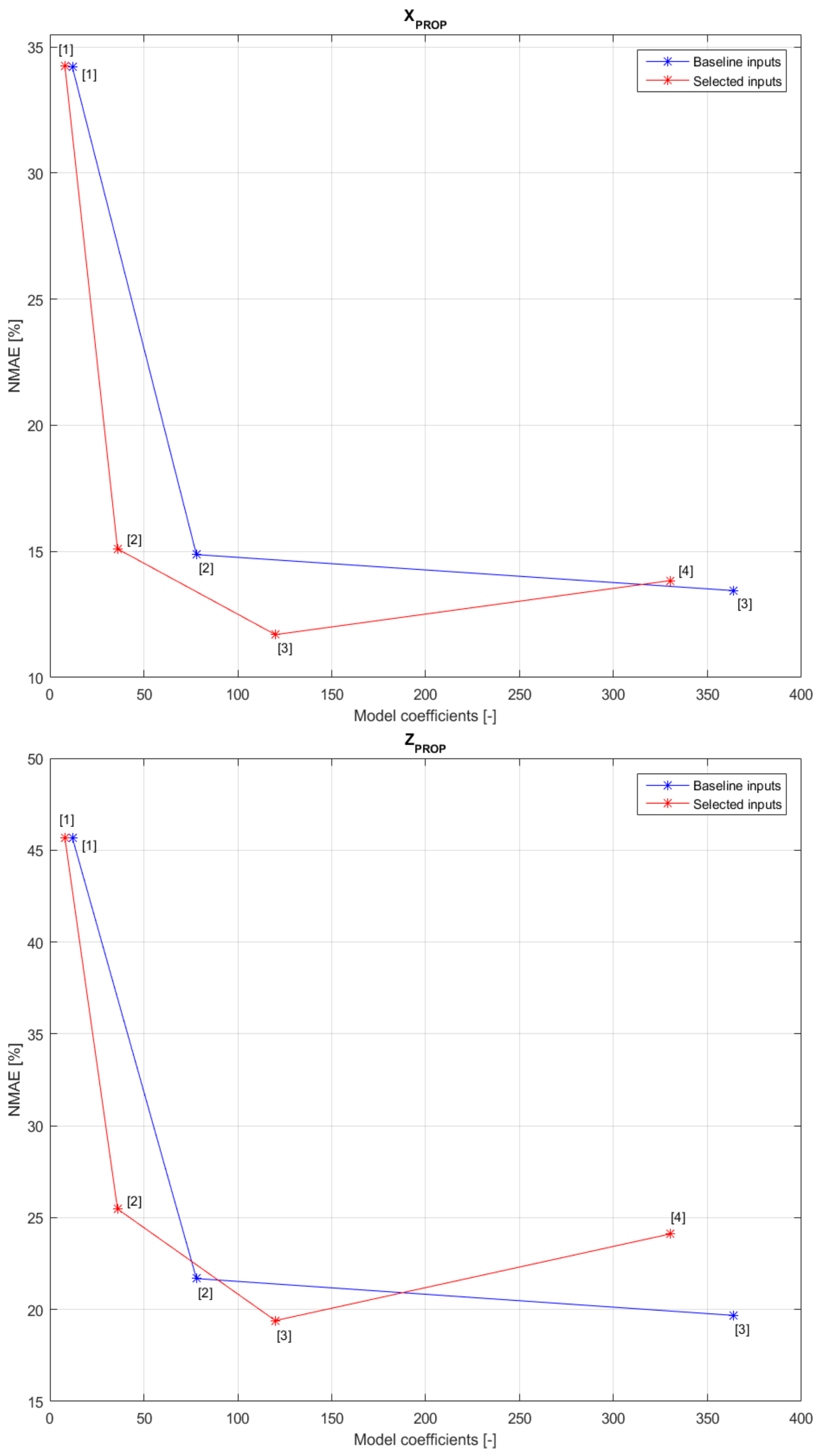


Figure D.2: Study of polynomial models for the engine model, with polynomial degrees reported between brackets.

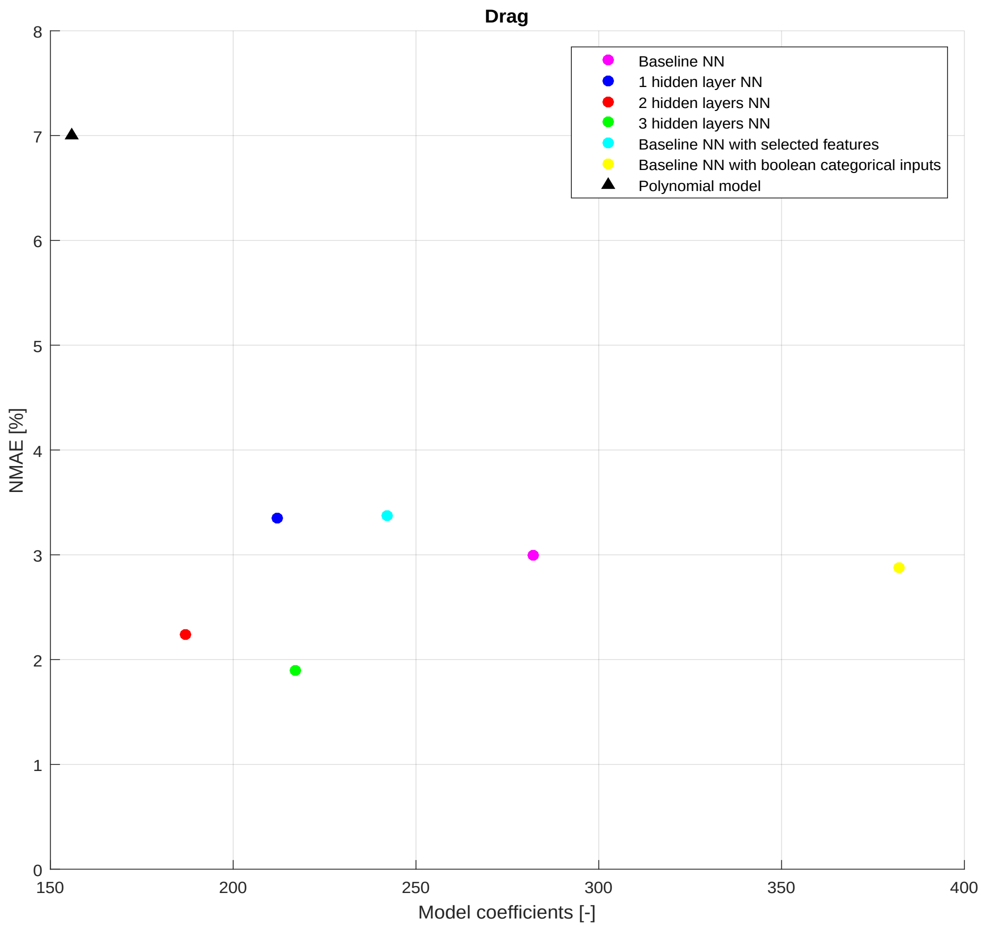
Appendix E
Comparison of results across model structure

In this Appendix, more details are provided on the comparison between the different model structures trained in the work, as already treated in 6.7. In Figures E.1 and E.2 the results in terms of validation NMAE and number of required coefficients are reported for a chosen set of surrogate models. The set includes, other than the baseline model, the most relevant results of each study presented in the report. It is reminded here that all neural networks featured in this overview have been trained with Bayesian Regularization, while the fitting of polynomial models is carried out with ordinary least squares regression.

The polynomial models reported in the graphs are the ones identified as best compromises among those tested in section 6.6. However, the intrinsic characteristics of such model structure represent a limitation for the current study. Despite the remarkable reduction capacity, in fact, the very poor performance in terms of accuracy makes the technique an unfeasible approach to build aerodynamics or engine metamodels.

The process of feature selection produces interesting results in the computation of the engine model outputs. Contrarily to what might be expected, the substantial decrease in memory requirements corresponds to a negligible deterioration of accuracy. Obviously, this consideration holds when comparing two models with the same fundamental network structure. Conversely, the approach does not lead to a gain over the baseline structure in the case of the aerodynamic modeling, which clearly requires more information to attain an accuracy comparable to other solutions.

The approach that effectively marks a strong improvement from the baseline metamodels is that of exploring multilayered neural networks. Both the presented best solutions with 2 and 3 hidden layers undoubtedly outperform the results of the other reported models. The baseline network model nonetheless represents a satisfactory starting compromise with substantial room for improvement.



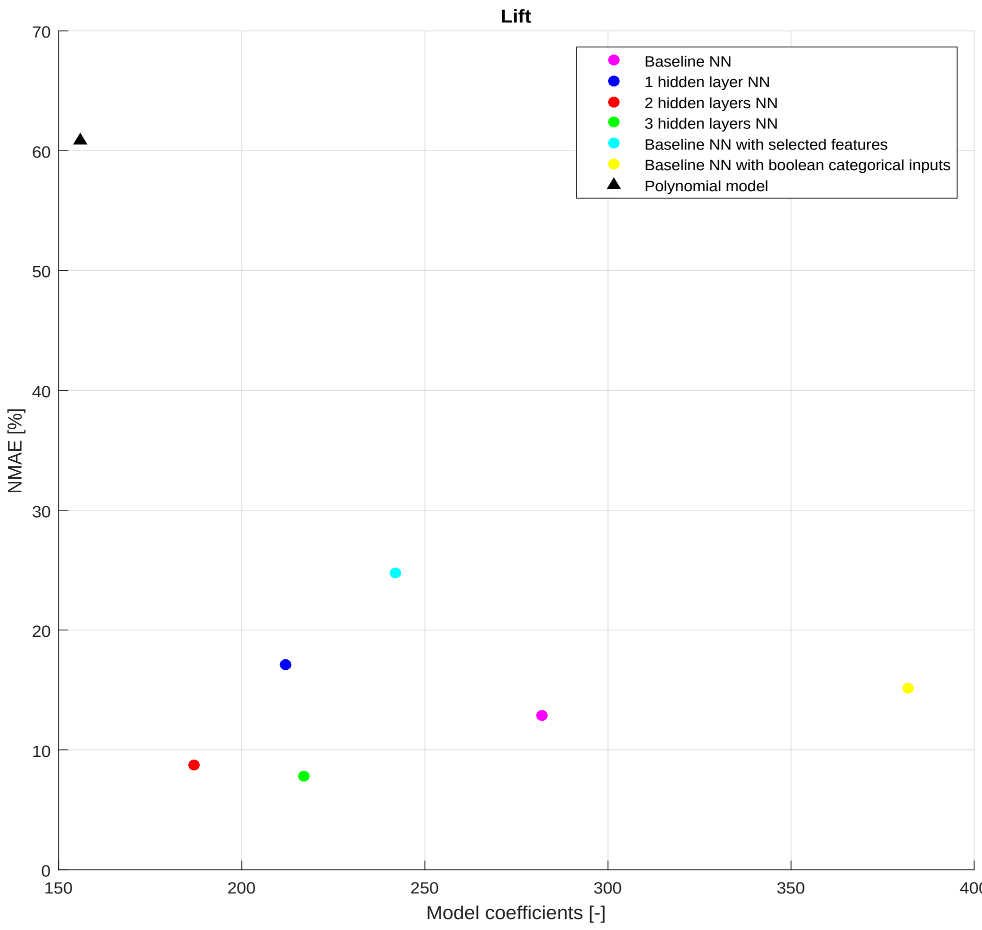
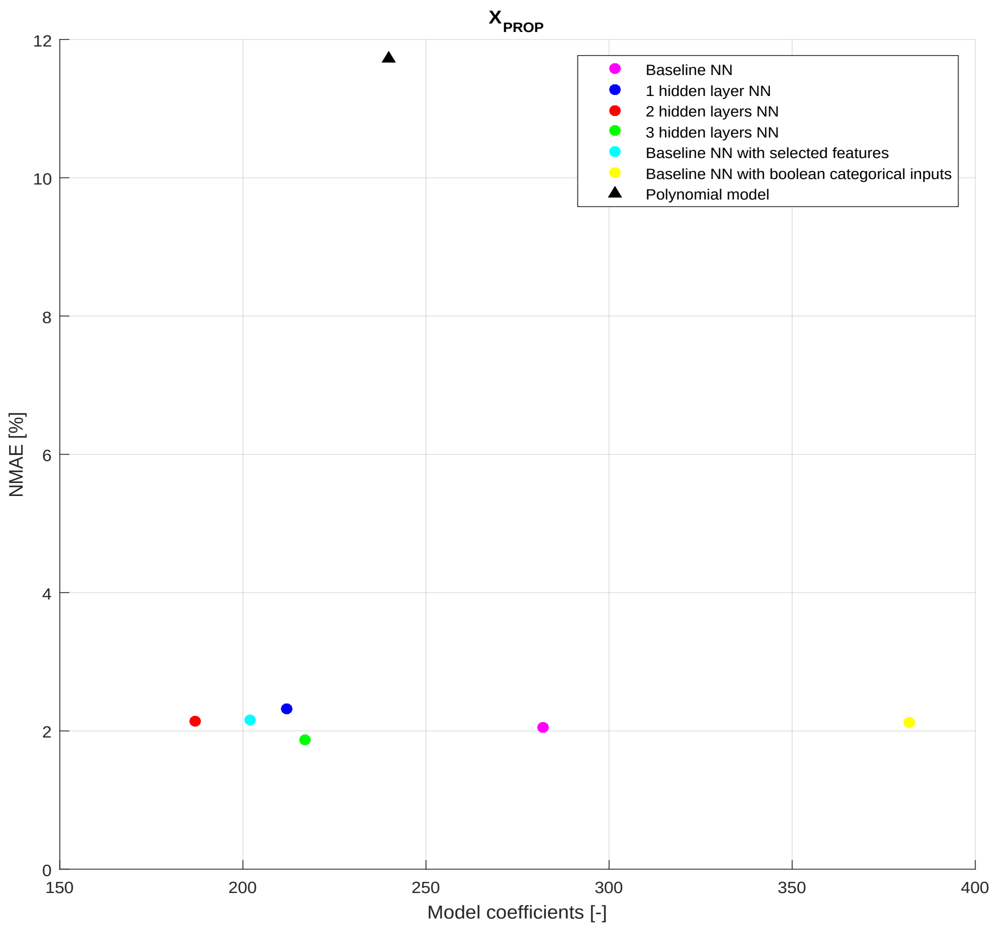


Figure E.1: Overview of the validation NMAE performance obtained with different types of metamodels for the aerodynamic variables.



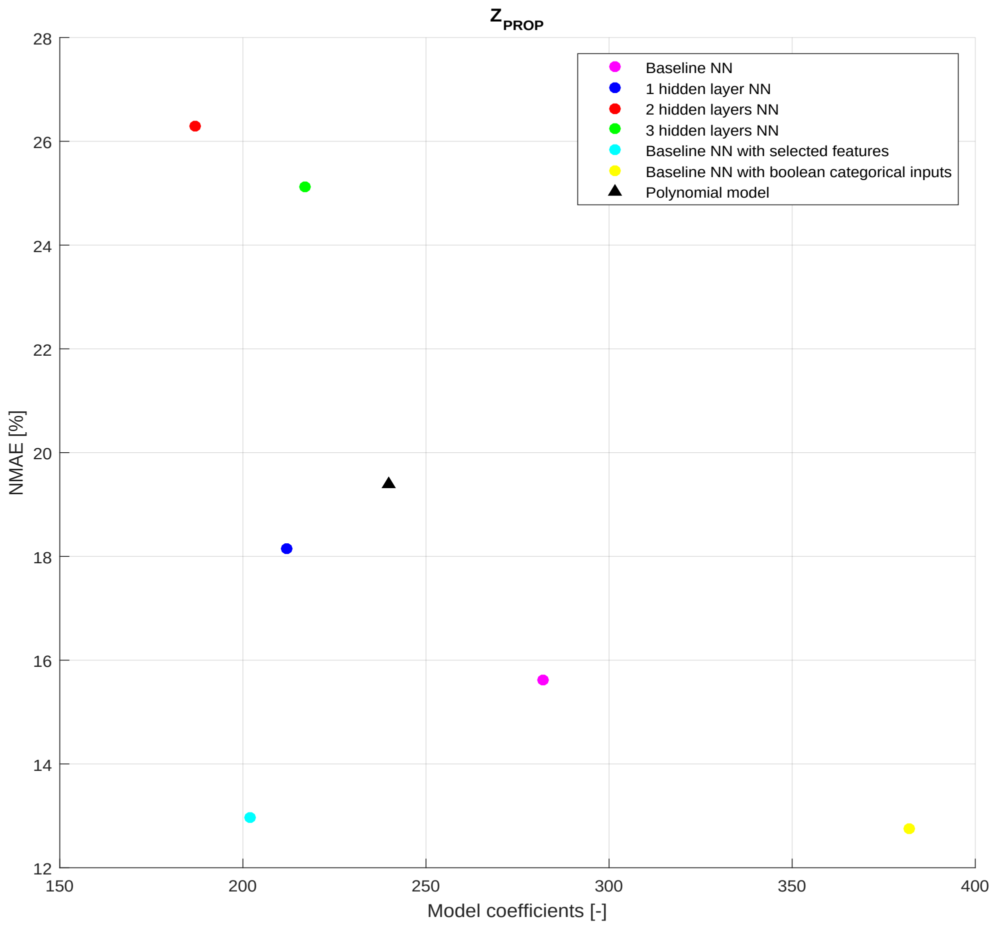


Figure E.2: Overview of the validation NMAE performance obtained with different types of metamodels for the engine variables.