Distributed Approach for Aerodynamic Model Identification of the ICE Aircraft

Thesis Report

M.S.T. van den Aarssen

TU Delft

Distributed Approach for Aerodynamic Model Identification of the ICE Aircraft

Thesis Report

by

M.S.T. van den Aarssen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday April 13, 2018 at 14:00.

Student number:	4223233	
Project duration:	May 12, 2017 – April 13, 2018	
Thesis committee:	Dr. Q.P. Chu,	TU Delft, Chair of the assessment committee
	Dr. ir. C.C. de Visser,	TU Delft, Daily supervisor
	Dr. M.A. Mitici,	TU Delft
	A.E. Brunner,	TU Delft

TU Delft

Summary

Lockheed Martin's Innovative Control Effectors (ICE) aircraft is a high performance fighter designed to have a small radar cross section. The redundant set of thirteen novel control effectors requires control allocation methods to be controllable by a human pilot. The recently developed Incremental Non-Linear Control Allocation (INCA) method requires the control Jacobian in real time to determine the optimal control commands. Because the derivatives were required, at least first order continuity was desired to prevent sudden jumps and sign changes in the Jacobian. The current aerodynamic model is a zeroth order continuous, Simplex B-Spline model. It was limited at zero order continuity because of the computational demands of increasing it.

This thesis presents a distributed approach, using the Alternating Direction Method of Multipliers (ADMM), that estimates the B-Coefficients of a Simplotope B-Spline to construct an aerodynamic model of the ICE-aircraft. Simplotope B-Splines divide the full domain into lower dimensional simplex splines, combined by taking the tensor product. This gives more flexibility in choosing the model structure, as triangulations, degrees, and continuity order can be chosen independently for each individual layer. Moreover, it generally reduces the required number of coefficients and constraints. However, the tensor product introduces high degree cross-coupling terms in the polynomial.

ADMM is a well-known and extensively studied method that optimizes separable cost functions with linear constraints in a distributed fashion. The partitions are split, and coupling coefficients are introduced to ensure global continuity. This results in a fully parallel algorithm that is either limited by the number of partitions or by the availability of parallel pools. The computational complexity is determined by the size of a single partition and infinitely scalable when refining the tessellation of the model.

Results showed that the ADMM algorithm approximates the global solution when comparing the validation results at various noise intensities. Globally continuity was enforced with incidentally a maximum mismatch of 10^{-4}. Higher noise intensity had a worsening effect on the constraint satisfaction, hence requiring more iterations. An adaptive penalty factor was used for robustness against badly chosen values, but the results indicated that the initial choice still affected the validation results.

The distributed approach was used to identify a first order continuous Simplotope B-Spline model for the ICE-aircraft. This model is evaluated in real time by a high performance evaluation function written in `C++`. All implemented models had a validation RMS below 1.40% and all maximum residuals below 28.1%. Assessing the performance to control inputs, the Simplotope B-Spline model was able to follow the original Simulink model with a maximum RMS deviation of 6.54%, which also included extrapolated values. Moreover, the simplotopes responded smoothly, indicating the benefit of having first order continuity, and the outcome of a well functioning distributed approach.

Preface

Dear Reader,

This document is the final report of my Master's Thesis Project, that I have conducted since May of last year. This Thesis finalizes my Master of Science in Aerospace Engineering, which I started in September 2015.

This report is subdivided into three main parts. The main contribution is reported in the shape of a scientific paper, and all work that has been done is concluded, as well as future recommendations are given. My Preliminary Thesis Report, which was handed in September 2017, is included in full as well. At last, the Appendices show all the results I have obtained during the project, which also includes results of approaches I have not chosen to use in the end.

I would like to express my gratitude to Coen de Visser, for supervising me during my thesis project, and providing guidance and advice. I would like to thank Tim Visser as well, for giving me a kick-start into Simplotope B-Splines.

I would like to thank my friends for giving me a fun time during my study period, in lectures and exam weeks, but also with activities, dinners, and parties. I especially like to thank my parents, Ruud, Marjolein, and my grandparents, for making it possible to earn my Master's degree in Aerospace Engineering. And at last I am grateful for having Iris, who supported me the last couple of years, and hopefully will in the next periods of my life.

M.S.T. van den Aarssen
Delft, March 2018

Contents

Nomenclature

Roman Letters

$\boldsymbol{b}$	Simplex barycentric coordinates
B	Simplex basis polynomials
$\boldsymbol{c}$	B-Coefficients of a Simplex or Simplotope B-Polynomial
C	Aerodynamic coefficient
$\hat{d}$	Number of B-Coefficients in a Simplotope
H	Continuity Matrix
$\mathscr{H}$	Indices of Constraints
J	Number of Simplotopes in tessellation
$\mathscr{J}$	Cost Function
ℓ	Number of layers
$\mathscr{L}$	(Augmented) Lagrangian
M	Mach number
n	Number of Dimensions
N	Neighbors
p, q, r	Roll, pitch, and yaw rate in body x, y, z directions
P	Number of Partitions
r	Continuity order
$\boldsymbol{r}, \boldsymbol{s}$	Primal- and Dual residual
R	Number of Continuity Constraints
X	Regression Matrix
Y	Measurements
z	Coupling coefficient

Greek Letters

α	Angle of Attack
β	Angle of Sideslip
$\boldsymbol{\beta}$	Simplotope barycentric coordinates
Γ	Simplotope
δ	Control effector deflection
ϵ	Tolerance
κ	Multi-index for the B-Coefficients
λ	Dual vector
$\boldsymbol{\lambda}$	Simplotope barycentric coordinates
π	Simplotope basis polynomial
ρ	Penalty factor

Subscripts

l, m, n	Aerodynamic moment in body x, y, z directions
n	Neighbor
p	Partition index
X, Y, Z	Aerodynamic Force in body x, y, z directions

Abbreviations

ADMM	Alternating Direction Method of Multipliers
AMI	Aerodynamic Model Identification
AMT	All Moving Tip
AOA	Angle of Attack
DLEF	Differential Leading Edge Flap
ICE	Innovative Control Effectors
INCA	Incremental Non-Linear Control Allocation

LAMT	Left All Moving Tip
LEL	Left Elevon
LIBLEF	Left Inboard Leading Edge Flap
LOBLEF	Left Outboard Leading Edge Flap
LSSD	Left Spoiler Slot Deflector
OLS	Ordinary Least Squares
PF	Pitch Flaps
RAMT	Right All Moving Tip
REL	Right Elevon
RIBLEF	Right Inboard Leading Edge Flap
RMS	Root Mean Square
ROBLEF	Right Outboard Leading Edge Flap
RSSD	Right Spoiler Slot Deflector
SSD	Spoiler Slot Deflector
TV	Thrust Vectoring
VSADMM	Variable Splitting Alternating Direction Method of Multipliers
WFR	Wavefront Reconstruction

List of Figures

List of Tables

I

Master's Thesis

1 Introduction

Various control- and control allocation methods use models in the loop to determine the best or most efficient inputs. This makes an accurate and reliable aerodynamic model essential, and thus Aerodynamic Model Identification (AMI) as a vital step in safe control of aircraft. Especially for highly complex, high performance fighters, such as the Innovative Control Effectors (ICE) aircraft, model in the loop controllers are useful. The redundant set of control effectors requires control allocation methods, as without those, human pilots are unable to fly it.

The aerodynamic dataset of the ICE aircraft is composed of a total of 108 submodels, which are combined to find 6 force- and moment coefficients. The current aerodynamic model, developed by [16], is constructed using Multivariate Simplex B-Splines and showed accurate estimations of the ICE's aerodynamic model. However, the local simplices were only connected by zeroth order continuity: the actual values were continuous, but the first order derivative may show discontinuities. This was visible in the model responses as ripples, where a smooth line was expected. The Incremental Non-linear Control Allocation (INCA) method developed in [10] uses the control Jacobian of the model, and thus the first derivative of the aerodynamic model w.r.t. the control effectors. Discontinuities in the first derivative or sudden sign changes harm the performance of INCA.

Additionally, the size of the models produced computationally expensive estimation routines, some models having over 250,000 B-Coefficients. This computational cost was not only inconvenient, but prevented the model degree of being increased further, and hence limited the approximation power.

Considering the above stated issues, the objective for this Master's thesis was to develop a high fidelity aerodynamic model for the Innovative Control Effectors Aircraft, with at least first order continuity throughout the domain, by taking a distributed system identification approach to construct a multivariate Simplotope B-Spline model.

As explained above, the first order continuity in the model is desired as it is likely to improve the functionality of the control allocation algorithm. However, in order to maintain approximation power, the model degrees will have to be increased by one as well. This will result in more coefficients, and thus even higher computational cost.

This motivated the need for methods reducing the computational complexity of the estimation. Simplotope B-Splines are an extension on Simplex B-Splines, dividing the full domain in multiple layers containing lower dimensional simplex splines, combined by taking the tensor product [17]. Simplotopes have proven to give accurate estimations of aerodynamic models, and also generally require less coefficients and less continuity constraints. This is beneficial in terms of computational costs, which depends directly on the amount coefficients. Moreover, it offers more flexibility when it comes to selecting the model's underlying structure, as it allows to choose different triangulations, polynomial degrees, and continuity orders for every direction independently. The better shaped subdomains also allowed it to be easier filled with measurements.

The B-Coefficients of the simplotopes are estimated by using an innovative, distributed approach for AMI, which fully exploits the local nature of the B-Spline structure, and the sparseness of the global continuity matrix. The Alternating Direction Method of Multipliers (ADMM) is an extensively studied method, that uses the

augmented Lagrangian to optimize separable cost functions with linear equality constraints [3]. By updating the separable parts of the algorithm in alternating fashion, followed by a dual update, the iterations will converge to the global optimal solution.

To reach this thesis's research objective, the following research questions were determined for guiding the project:

1. *How can a multivariate Simplotope B-Spline model of the ICE aircraft be constructed that can be used in modern model-in-the-loop controllers, and what are the advantages and disadvantages compared to the current simplex spline modelling approach?*
   (a) Can the model be constructed for at least 5 dimensions?
   (b) How can this model be constructed with first order continuity between the Simplotopes?
   (c) How much extra measurements needs to be generated using data augmentation methods?
   (d) How can this model be evaluated in real time?
2. *How does using Simplotope B-Splines affect the model estimation?*
   (a) How does it perform compared to Simplex B-splines?
      - Statistically
      - Computationally
      - Accuracy
      - Data Requirements
   (b) How does it perform compared to the look-up tables designed by Lockheed Martin?
      - Statistically
      - Computationally
      - Accuracy
3. *How does the distributed system identification approach affect the model estimation?*
   (a) How much can computational time be reduced compared to global approaches?
   (b) To what extent is the quality of the model affected?
      - Statistically
      - Computationally
      - Accuracy

This thesis report is structured into three parts. The main contribution of this thesis is included in a scientific paper that is given in Chapter 2. This paper will be submitted for publication in the AIAA Journal of Guidance, Control, and Dynamics. The main conclusions and recommendations are given in Chapter 3. This includes a reflection on the research objective and questions as stated above, as well as other main conclusions that could be drawn from all the work done during this thesis project. Recommendations are given about how the presented method can be implemented more efficiently, and extensions that can be explored, among others. On September 1st, after 4 months in the project, a Preliminary Thesis Report was handed in, with a literature study and preliminary findings. This Preliminary Thesis Report has been included in full in Part II.

In the Appendices all results obtained during this thesis project are given. This includes the chosen models in Appendix A, together with comparisons between model structures to justify the choices. As multiple distributed approaches have been evaluated, the derivations and results are given in Appendix B. At last, Appendix C provides documentation of the written software during the project. This includes the estimator, but also a high performance evaluation function in C++, required to provide real-time evaluation of all models.

2

Scientific Paper

Distributed approach for Aerodynamic Model Identification of the ICE aircraft using the Alternating Direction Method of Multipliers in combination with Simplotope B-Splines

M.S.T. van den Aarssen* and C. C. de Visser†
Delft University of Technology, Delft, 2600GB, The Netherlands

High performance control allocation methods for the Innovative Control Effectors (ICE) aircraft require accurate onboard aerodynamic models, with preferably first order continuity. Simplotope B-Splines, an extension on Simplex B-Splines, have a high approximation power by using local cost functions. However, enforcing global continuity produces computationally expensive optimization problems. This paper presents a distributed approach, using the Alternating Direction Method of Multipliers (ADMM), to reduce the complexity of the B-Coefficients' estimation. ADMM decouples the simplotopes, and introduces coupling coefficients to enforce global continuity, resulting in a parallel estimation algorithm whose complexity is depending solely on the partition size, being independent of refinement of the model tessellation. Results show that for a 3D model, the distributed algorithm converges steadily to the global solution with a good approximation after a couple hundred iterations. Validation results of the distributed approach were similar to those of the global optimal solution for various noise intensities, and the continuity constraints were satisfied with maximum mismatches below 10^{-4}. The distributed approach has been used to construct a first order continuous aerodynamic model for the ICE aircraft, which has been implemented in Simulink, and proven to perform well compared to the original model.

Nomenclature

$\boldsymbol{b}$	=	Simplex barycentric coordinates
B	=	Simplex basis polynomials
C	=	Aerodynamic coefficient
$\boldsymbol{c}$	=	B-Coefficients of a Simplex or Simplotope B-Polynomial

*MSc Student, Control and Simulation Division, Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delft, The Netherlands, marcvdaarssen@gmail.com

†Assistant Professor, Control and Simulation Division, Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delft, The Netherlands, c.c.devisser@tudelft.nl. AIAA Member

H	=	Continuity Matrix
$\mathcal{H}$	=	Indices of constraints
J	=	Number of Simplotopes in tessellation
$\mathcal{J}$	=	Cost Function
$\mathcal{L}$	=	(Augmented) Lagrangian
M	=	Mach number
n	=	Number of Dimensions
$\mathcal{N}$	=	Null space
P	=	Number of Partitions
p, q, r	=	Roll, pitch, and yaw rate in body x, y, z directions
r	=	Continuity order
R	=	Number of Continuity Constraints
$\boldsymbol{r}, \boldsymbol{s}$	=	Primal- and Dual residual
t	=	Simplex
X	=	Regression Matrix
Y	=	Measurements
z	=	Coupling coefficient
α, β	=	Angle of Attack and Angle of Sideslip
$\boldsymbol{\beta}$	=	Simplotope barycentric coordinates
Γ	=	Simplotope Spline
δ	=	Control effector deflection
ϵ	=	Tolerance
κ	=	Multi-index for the B-Coefficients
λ	=	Dual vector
$\boldsymbol{\lambda}$	=	Simplotope barycentric coordinates
π	=	Simplotope basis polynomial
ρ	=	Penalty factor

Subscripts

l, m, n = Aerodynamic moment in body x, y, z directions

p = Partitions index

X, Y, Z = Aerodynamic Force in body x, y, z directions

I. Introduction

Aerodynamic Model Identification (AMI) plays an important role in safe- and high-performance control of aircraft, as various control techniques use an onboard aerodynamic model [1–3]. Inversion methods require e.g. an estimation of the aircraft's aerodynamic coefficients or control Jacobian. Especially a high performance fighter, such as the Innovative Control Effectors (ICE) aircraft, depends upon proper control allocation, as without that, the redundant set of control effectors makes it nearly impossible to be flown by a human pilot.

Numerous techniques for AMI exist. Ordinary polynomials are most common because of their relative simplicity. These are generally used in small models and work well in those cases, but become inaccurate when the aerodynamic model is large or highly non-linear, as is the case with the ICE aircraft [4]. Artificial Neural Networks (ANN) are used in many machine learning cases, because of their high approximation power. However, its nature is considered black box as the internal workings are hard to identify. In order to estimate all the parameters, neural networks require a high amount of training data, while optimizing global cost functions. In addition, ANNs are generally tempting to overfit the identification data [5]. Tensor Product splines combine multiple univariate functions to obtain higher dimensional models, but are not able to fit scattered data [6, 7]. As scattered measurements are most common in the context of AMI, this method is not well suited in general.

Multivariate Simplex B-Splines combine the approximation power of artificial neural networks and the simplicity of the polynomials [8, 9]. Because it divides the full domain in smaller patches with local cost functions, it has a high approximation power. By using the barycentric formulation of the polynomials, the basis is stable and shows easy formulations to enforce any order of continuity between the subdomains [10]. This model structure has been successfully applied to AMI, but also to other fields, such as adaptive optics or partial differential equations [8, 11, 12].

The Simplex B-Spline can be extended into the Simplotope B-Splines. The Simplotope B-Spline subdivides the model in layers with lower dimensional simplex splines, decoupling the underlying structures and offering the flexibility to choose other degrees for different dimensions [13, 14]. The simplotopes are formed by taking the tensor product between these layers, forming new subdomains. This can lead to a reduction in coefficients and continuity constraints, while maintaining or increasing the approximation power.

A beneficial aspect of the B-Spline structure is that the regression problem is linear-in-the-parameters, which means that the B-Coefficients can be estimated using linear methods as Equality Constrained Ordinary Least Squares (ECOLS). This well-known method has multiple solvers, such as directly inverting the Karush-Kuhn-Tucker matrix, null space projection, or a highly efficient iterative solver that has been developed especially considering the B-Spline structure [9, 15, 16]. However, all these methods still solve a global problem, not fully exploiting the local properties of the simplotope splines. Distributed methods have been used as well with B-Splines, using Dual Ascent or the Alternating Direction Method of Multipliers (ADMM) [12, 17]. However, this has only been used for the 2-dimensional wave front reconstruction problem and has not been applied to the higher dimensional AMI case yet.

The current aerodynamic model of the ICE aircraft consists of a multivariate Simplex B-Spline model, linked together with zeroth order continuity [18]. Although it was accurately identifying the data, it showed some wiggles between the datapoints and discontinuities in the control derivatives. Moreover, the coefficients were estimated using the iterative solver described in [16], which was computationally extremely expensive, because some models required over 200,000 coefficients. This was not only inconvenient, but also limiting model's degree and thus approximation power.

In order to reduce the computational cost of the estimation, distributed approaches for AMI in combination with multivariate Simplotope B-Splines have been investigated. Especially a desire to have infinitely scalable methods encouraged research towards fully parallel optimization schemes able to maintain global continuity between the subdomains. This paper presents such a distributed approach, using ADMM, splitting the B-Coefficients per simplotope, and introducing coupling coefficients to ensure continuity. This method's complexity is dependent on the size of the individual units, but is indifferent to refinement of the tessellation because of its parallel nature. The method has been evaluated for several noise intensity levels and compared to its global solution in terms of convergence, validation errors, and global constraint satisfaction.

The distributed approach will be used to identify a model for the ICE-aircraft, which will be compared to the Simulink model that has been developed for research purposes [19].

This paper is structured as follows. First a brief description of the history and the aerodynamic model of the ICE aircraft will be given in Section II, together with the challenges faced with the current aerodynamic model. Next the concept of multivariate B-Splines with Simplices and Simplotopes is introduced in Section III. How this model structure can be used to identify the aerodynamic model, and the introduction of ADMM is presented in Section IV. Results are given in Section V and the findings are concluded in Section VI.

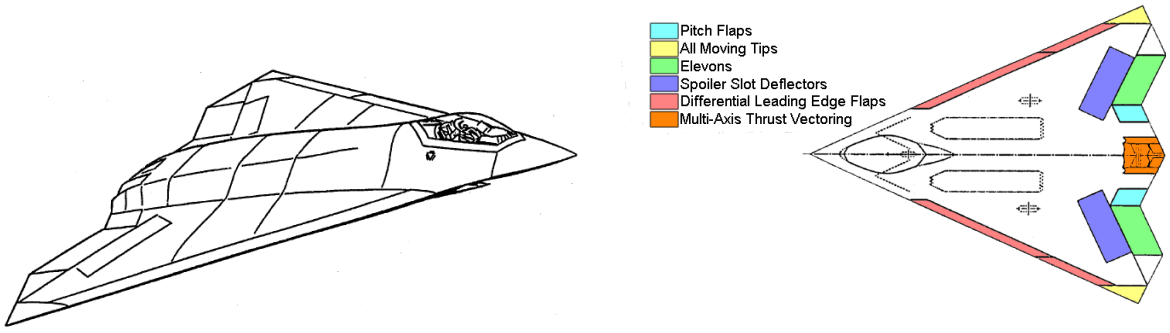
II. Description of the Innovative Control Effectors aircraft

The distributed approach that will be presented in Section IV, will be applied to the aerodynamic model of Lockheed Martin's Innovative Control Effectors (ICE) aircraft. The ICE-aircraft has been designed in the mid 1990s as a high performance fighter jet and includes a redundant set of unconventional control effectors. These control effectors ensure controllability and stability in all regions of the flight envelope. A general description will be given in Section II.A. The aerodynamic model is described in Section II.B.

A. History and Description

The ICE project has been initiated by the United States' Government's desire to have a high performance fighter in all parts of the flight envelope, while reducing the aircraft's radar cross section. This restricted the fighter's design, resulting in a tailless aircraft and integrated controls as shown in Fig. 1a. Stability and controllability issues initiated the research into innovative control effectors. This has been done for two baseline configurations of the fighter. One is land based, which has a 65 degree sweep delta wing; the other is a carrier based version, and has a 45 degree sweep angle, and a canard configuration.

During two phases of researching various novel control concepts, a selection of control effectors had been made. The land based version that will be studied in this paper consists of a total of 13 control effectors [20, 21].



(a) The land based version. **(b) Top view planform**

Fig. 1 The configuration of the ICE aircraft studied in this paper [20, 21]

Except for the more conventional Elevons (2), Pitch Flaps, and Multi-Axis Thrust Vectoring, some novel control surfaces were introduced. The All Moving Tips (AMT) are located at the wing tips and are able to fully rotate down. These were considered overall, the best performing control effectors for directional control in all regions of the envelope.

The Spoiler Slot Deflectors (SSD), located on the wings, combine a conventional spoiler with a deflector on the lower side of the wing, and a hole that enables air flowing through. Although providing good directional control, they were less effective than the AMTs. Moreover, deflecting the SSDs highly affected the trailing edge effectors.

The Differential Leading Edge Flaps (DLEF) are able to provide lateral control. There are inboard- and outboard flaps placed on each side of the wing. This control surface was deemed most useful in power approaches on the one hand, and low speed, high Angle of Attack conditions on the other hand. Moreove, they retained its control effectiveness at high Mach numbers.

A full overview of all the control effectors, their conventions, and the maximum deflections and rates, are shown in Table 2. The convention can either be Trailing Edge (TE) or Leading Edge (LE) Up (U) or Down (D). The exact positions are colored in the planform in Fig. 1b.

Table 2 An overview of all the control effectors of the ICE aircraft, and their conventions, limits, and notations [19].

Control Surface	Convention Positive	Limits	Rate limit	Notation
Spoiler Slot Deflectors (2)	TEU	[0,60]	150	$\delta_{LSSD}, \delta_{RSSD}$
Differential Leading Edge Flaps (4)	LED	[0,40]	40	$\delta_{LIBLEF}, \delta_{RIBLEF}$ (Inboard)
		[-40,40]	40	$\delta_{LOBLEF}, \delta_{ROBLEF}$ (Outboard)
All Moving Tips (2)	TED	[0,60]	150	$\delta_{LAMT}, \delta_{RAMT}$
Elevons (2)	TED	[-30,30]	150	$\delta_{LEL}, \delta_{REL}$
Pitch Flaps (1)	TED	[-30,30]	150	δ_{PF}
Multi-Axis Thrust Vectoring	$\dot{p} > 0, \dot{q} > 0$	[-15,15]	150	δ_{TV}

The redundant set of effectors gives both opportunities and challenges, considering control and stability. With particular effectors more effective in different areas of the flight envelope, it rises the opportunity to fly at a large range of aerodynamic angles; e.g. the Angle of Attack is able to grow up to 90 degrees.

Since it is over-effected, it requires control allocation methods in order to have it properly steered by a human pilot. Only recently an effective Incremental Non-Linear Control Allocation method has been designed and proven to work well in the simulation environment [3]. This method requires an onboard aerodynamic model for the estimation of the control effectiveness Jacobian. This calls for an accurate aerodynamic model whose derivatives can be evaluated during flight time.

B. Aerodynamic Model Description

The aerodynamic model of the ICE aircraft that is currently being used includes non-linear effects of the control effectors, individually as well as coupling effects, dynamic derivatives, aeroelastic effects, and hinge moment derivatives [19].

The data has been collected using subsonic-, transonic-, and supersonic wind tunnel tests on a small scale model of the aircraft and numerical methods, e.g. the DYNAMIC panel method to estimate dynamic derivatives at high mach numbers.

The full aerodynamic model is composed of a total of 108 submodels. Each of the three force coefficients (C_X, C_Y,

C_Z) and moment coefficients (C_l, C_m, C_n) consist of 17-19 submodels. Each submodel represents a contribution of either the bare airframe, an individual control effector, a combination of control effectors, or a dynamic derivative. The main coefficients are being estimated by combining the individual submodels. An example of how a main coefficient is built up is shown in Eq. (1) for the yawing moment coefficient C_n. All the main coefficients have the same subcoefficients in terms of dependency on states, only the dynamic derivatives $C_{n_{18}}$ and $C_{n_{19}}$ are not present for all coefficients. There are submodels that are added to the total, whereas some are subtracted. This combination is different for each main coefficient. The influence of the multi-axis thrust vectoring has not been included in the submodels, but combined with the propulsion model instead.

$$\begin{aligned} C_n \;= & \; C_{n_1}(\alpha, M) + C_{n_2}(\alpha, \beta, M) - C_{n_3}(\alpha, \beta, \delta_{LIBLEF}) - C_{n_4}(\alpha, \beta, \delta_{LIBLEF}, \delta_{LOBLEF}, M) \\ & + C_{n_5}(\alpha, \delta_{LSSD}, \delta_{LEL}, M) + C_{n_6}(\alpha, \delta_{LSSD}, \delta_{RSSD}, \delta_{PF}, M) + C_{n_7}(\alpha, \beta, \delta_{LAMT}) \\ & + C_{n_8}(\alpha, \delta_{LEL}, \delta_{LAMT}) + C_{n_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT}) - C_{n_{10}}(\alpha, \delta_{REL}, \delta_{RAMT}) \\ & - C_{n_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT}) + C_{n_{12}}(\alpha, \beta, \delta_{LSSD}) + C_{n_{13}}(\alpha, \beta, \delta_{RIBLEF}) \\ & + C_{n_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M) - C_{n_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M) - C_{n_{16}}(\alpha, \beta, \delta_{RAMT}) \\ & - C_{n_{17}}(\alpha, \beta, \delta_{RSSD}) + \frac{bp}{2V} C_{n_{18}}(\alpha, M) + \frac{br}{2V} C_{n_{19}}(\alpha, M) \end{aligned} \tag{1}$$

The data has been combined into a Simulink model that interpolates, linearly or by cubic spline, the measurement data in order to find each submodel's contribution. The actuator dynamics are implemented separately as linear actuation models. There is a separation between the leading edge devices, which have a low bandwidth actuator, and all others, which have a high bandwidth [3, 19]. These actuator implementations have been used for the results in Section V.

The currently used aerodynamic model, constructed by [18], and uses multivariate Simplex B-Splines with zeroth order continuity between the simplices. For all submodels, the simplex splines resulted in relative validation RMSs below 1.6%, and relative maximum validation residuals below 21.6%. Because it only contained zeroth order continuity, discontinuities occurred in the first derivative, causing visible ripples in the model responses. These discontinuities also harmed the control allocation method from [3].

Another challenge was evaluating the model in real time. Since the model output is determined by the simplex in which the state falls, this simplex needs to be determined at every time step. This is a costly search, but was implemented with high performance in C++, allowing sufficient possible evaluations per second [18].

III. Preliminaries on Multivariate B-Splines

This section will provide preliminary information of the used model structure: Simplotope B-Splines. First, the theory of Simplex B-Splines is discussed in Section III.A, and how the local patches can be linked together using continuity conditions (Section III.B). Then the theory is extended into Simplotope B-Splines in Section III.C, which

divides the dimensions into multiple layers, containing lower dimensional simplices.

A. Spline Functions and Barycentric Coordinates

Bernstein basis polynomials are formed on simplices, where an n-dimensional simplex is defined as the convex hull of $n+1$ different vertices $\boldsymbol{v}$ [8]. A simplex spline can be created on a Triangulation $\mathcal{T}$, which is the division of the domain in a set of J non-overlapping simplices [9]:

$$\mathcal{T} = \bigcup_{j=1}^{J} t_j, \quad t_i \cap t_j \in \{\emptyset, \tilde{t}\}, \quad \forall\, t_i, t_j \in \mathcal{T} \tag{2}$$

The edge facets consist of lower dimensional simplices $\tilde{t}$. Each edge between two neighboring simplices t_i and t_j can be specified by the out of edge vertices v_* of each simplex. This vertices are the ones of the simplices that are not part of the edge facet. The barycentric coordinates $\boldsymbol{b}$ of a data point are defined locally in the simplex. In this coordinate system all points x get a set of weights $\boldsymbol{b}$ which are multiplied with the vertices $\boldsymbol{v}$ of the particular simplex, as can be seen in Eq. (3) [9].

$$x = \sum_{i=0}^{n} b_i v_i \qquad\qquad \sum_{i=0}^{n} b_i = 1 \tag{3}$$

Every $n-$dimensional data point has $n+1$ barycentric coordinates, corresponding to the amount of vertices. When the chosen point is within the convex hull of the vertices $\boldsymbol{v}$, all barycentric coordinates will be non-negative. Any point outside this hull can be transformed into barycentric coordinates as well, but will have at least one negative coordinate.

The polynomial basis function in Eq. (4) can be used to write any given polynomial $p(\boldsymbol{b})$ of degree d as the weighted sum of these basis functions. This is shown in Eq. (5). By defining the basis function of any point outside the convex hull of the simplex 0, the basis becomes local. The property that all barycentric coordinates add up to one, makes that the basis is also stable.

$$B_\kappa^d(\boldsymbol{b}) = \frac{d!}{\kappa!}\boldsymbol{b}^\kappa \tag{4}$$

$$p(\boldsymbol{b}) = \sum_{|\kappa|=d} c_\kappa B_\kappa^d(\boldsymbol{b}) \tag{5}$$

The weights c_κ in Eq. (5) are the B-Coefficients for this particular simplex and define the shape of the polynomial. Each coefficient corresponds to a single permutation of the multi-index $\kappa = (\kappa_0, \kappa_1, \ldots, \kappa_n)$, provided that $0 \leq \kappa_i \leq d$ and $|\kappa| = d$. The total number of B-Coefficients that is used within a single simplex, $\hat{d}$, can be computed by Eq. (6).

$$\hat{d} = \binom{d+n}{d} = \frac{(d+n)!}{d!n!} \tag{6}$$

B. Continuity

As the Bernstein polynomials are defined locally on the simplices, continuity constraints have to be introduced to ensure that the model has a single surface that is smooth up to a chosen order. For orders of continuity r between two simplices t_i and t_j, the continuity equations can be formed by using Eq. (7) [10].

$$c^{t_i}_{\kappa_0,m,\kappa_1,\ldots,\kappa_{n-1}} = \sum_{|\gamma|=m} c^{t_j}_{(\kappa_0,0,\kappa_1,\ldots,\kappa_{n-1})+\gamma} B^m_\gamma(v^{t_i}_*), \qquad 0 \leq m \leq r \tag{7}$$

For every permutation of the continuity multi-index $\gamma = [\gamma_0, \gamma_1, , \ldots, \gamma_n], 0 \leq \gamma_i \leq m, |\gamma| = m$, a continuity equation is formed based on the out of edge vertex $v^{t_i}_*$ of simplex t_i. The barycentric coordinates have to be computed w.r.t. the vertices of the simplex t_j. Additionally, the condition $|\kappa| + |\gamma| = d$ has to hold for every constraint. With the chosen continuity order r, every edge has R continuity conditions, which can be computed with Eq. (8)

$$R = \sum_{m=0}^{r} \frac{(d-m+n-1)!}{(d-m)!(n-1)!} \tag{8}$$

Because all these constraints are linear, they can be combined for all simplices into a single, global smoothness matrix $\mathbf{H}$. This makes that $\mathbf{H} \cdot \boldsymbol{c} = 0$ can be used as the linear equality constraints of the optimization problem.

C. Simplotope Splines

A simplotope Γ in n dimensions can be subdivided in ℓ layers, in which $1 \leq \ell \leq n$ [13]. The dimensions of the layers are indicated in a multi-index $\nu = (\nu_1, \ldots, \nu_\ell)$, with $|\nu| = n$, and every layer i within a simplotope being a ν_i-dimensional simplex spline. The simplex in each layer will be triangulated individually and combined with the other layers in one tessellation.

A point within the simplotope can be described using the barycentric coordinates $\boldsymbol{b}_i$ for every layer i, combined into $\boldsymbol{\beta} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell)$. Where a point in a simplex has $n + 1$ barycentric coordinates, a point within a simplotope has $n + \ell$ coordinates.

To form basis polynomials in the simplotope, the multi-indices κ for each layer are combined with the other layers. This results in the multi-index $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_\ell)$, in which $\boldsymbol{\lambda}_i$ has the same properties and restrictions as the multi-index κ of layer i. The basis polynomials for Simplotope B-Splines are given by Eq. (9), and this can be combined in a polynomial $\pi(\boldsymbol{\beta})$, shown in Eq. (10).

$$\boldsymbol{B}_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) = \prod_{i=1}^{\ell} B^{d_i}_{\boldsymbol{\lambda}_i}(\boldsymbol{b}_i) \tag{9}$$

$$\pi(\boldsymbol{\beta}) = \sum_{|\boldsymbol{\lambda}_i|=d_i, \forall i \in [1,\ell]} \boldsymbol{c}_{\boldsymbol{\lambda}} \boldsymbol{B}_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) \tag{10}$$

The degree vector $\boldsymbol{d} = (d_1, \ldots, d_\ell)$ indicates each layer's chosen polynomial degree. The local B-Coefficients

$\boldsymbol{c}_{\boldsymbol{\lambda}}$ that shape the polynomial have the same properties as the the coefficients $\boldsymbol{c}_{\kappa}$ for simplices. The total amount of B-Coefficients can be found by taking the product of the number of coefficients $\hat{d}$ of each individual layer. Similar to the simplex basis polynomial, the $\boldsymbol{B}_{\boldsymbol{\lambda}}(\boldsymbol{\beta})$ is only defined when the evaluated point with barycentric coordinates $\boldsymbol{\beta}$ lies within the convex hull of simplotope Γ, and is zero otherwise.

Example 1: Consider a 3-dimensional, (2, 1)-simplotope model. The first, 2-dimensional layer has degree 2, and the second, 1-dimensional layer has degree 3. The κ-permutations of the individual layers are:

$$\boldsymbol{\lambda}_1 = \{(2,0,0),(1,1,0),(1,0,1),(0,2,0),(0,1,1),(0,0,2)\}$$
$$\boldsymbol{\lambda}_2 = \{(3,0),(2,1),(1,2),(0,3)\}$$

Taking the tensor product between the layers' basis functions $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ will result in 24 polynomial terms per simplotope. For each of these terms there is a B-Coefficient. The basis functions are computed per layer as in Eq. (4), and combined in Eq. (9). One of the combinations for $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)$ is $((1,0,1),(1,2))$, with the following basis functions:

$$\begin{aligned}\boldsymbol{B}^{(2,3)}_{(1,0,1,1,2)}(\beta) &= B^2_{(1,0,1)}(\boldsymbol{b}_1)\cdot B^3_{(1,2)}(\boldsymbol{b}_2)\\ &= \frac{2!}{1!0!1!}b^1_{10}b^0_{11}b^1_{12}\cdot\frac{3!}{1!2!}b^1_{20}b^2_{21} = 2b_{10}b_{12}b_{20}b^2_{21}\end{aligned}$$

Continuity between simplotopes is defined per layer [14] by imposing the constraints as described in Eq. (7) on each layer's triangulation first. This means that each layer can have its own required order of continuity. The constraints in each layer are duplicated for every copy of the B-net as a result of the tensor product with other layers. All equations are combined in one global matrix $\mathbf{H}$, such that $\mathbf{H}\cdot\boldsymbol{c} = 0$, with $\boldsymbol{c}$ the global vector of all B-Coefficients. The total amount of continuity conditions per edge between two simplotopes, is given in Eq. (11). The R_{ooe} indicates the amount of conditions in the out of edge (ooe) layer, computed by Eq. (8). The second term is the amount that has been added because of the tensor product between the layers.

$$R_{Simplotope} = R_{ooe}\frac{\prod_{i=1}^{\ell}\hat{d}_i}{\hat{d}_{ooe}} \tag{11}$$

Advantages of using simplotopes over simplices are the flexibility it offers to decouple dimensions, such that different degrees of polynomial can be used in separate dimensions. This is beneficial in case highly unequal amounts of data are available in particular dimensions, or when an analytical model indicates coupling between some variables. Moreover, simplotopes are generally better shaped, easier to be filled with data, and often require less coefficients and

less continuity constraints [13]. On the downside, the tensor product between the layers will result in high cross-coupling terms up to a degree as high as $|\boldsymbol{d}| = d_1 + \cdots + d_\ell$, while not having a complete polynomial. In the example above, the shown basis polynomial, and all others, is 5^{th} degree, the sum of the layers' degrees. Although the total amount of coefficients is reduced, the simplotope structure will express every data point in a higher number of polynomial terms and more coefficients as well, increasing the memory requirements of e.g. the regression matrix.

IV. Aerodynamic Model Identification

Because the Bernstein polynomials are linear-in-the-parameters, the estimation can be done by using linear methods such as Ordinary Least Squares (OLS) [9]. The OLS cost function is the sum of the squared residuals, and because of the locality of the polynomials, this results in a summation of the simplotopes' individual cost functions as described in Eq. (12).

$$\mathcal{J}(\boldsymbol{c}) = \sum_{i=1}^{J} ||\mathbf{X}_i \boldsymbol{c}_i - \boldsymbol{Y}_i||_2 \tag{12}$$

Using this cost function, in combination with the continuity constraints as defined earlier, the problem can be set up as in Eq. (13), which is a Equality Constrained Ordinary Least Squares (ECOLS) problem.

$$\begin{aligned} &\text{argmin}_{\boldsymbol{c}} \mathcal{J}(\boldsymbol{c}) \\ &\text{subject to } \mathbf{H}\boldsymbol{c} = 0 \end{aligned} \tag{13}$$

Various solvers have been used for solving this regression problem [9, 15, 16], but all of these solve a global problem, and do not completely profit from the local cost function of the Simplotope B-Spline structure. The Alternating Direction Method of Multipliers (ADMM) is introduced in Section IV.A, with the Variable Splitting ADMM (VSADMM) version explained in Section IV.B. VSADMM is fully parallel and able to estimate the coefficients of the B-Splines at a lower computationl complexity, as described in Section IV.C. Additional options that are used can be found in Section IV.D.

A. Alternating Direction Method of Multipliers

ADMM is able to solve convex optimization problems in a distributed fashion, and has been used in various scientific fields [22]. It has also been applied in combination with multivariate Simplex B-Splines, solving a 2-dimensional wavefront reconstruction problem for adaptive optics purposes [17]. In general, ADMM solves equations of the form in Eq. (14); minimizing the sum of two separable convex functions, subject to linear (equality) constraints.

$$\begin{aligned} &\text{minimize } \; f(x) + g(z) \\ &\text{subject to } \; \mathbf{A}x + \mathbf{B}z = c \end{aligned} \tag{14}$$

It solves the system by forming the augmented Lagrangian in Eq. (15). The penalty factor $\rho > 0$ is chosen to penalize not satisfying the constraints, and thus reaching a feasible solution faster. There are analytical solutions for an optimal penalty factor for quadratic problems [23, 24], but these are not fully applicable to the B-Splines case. For robustness against badly chosen penalty factors, an adaptive approach as explained in Section IV.D.2 has been taken.

$$\mathcal{L}(x, z, \lambda) = f(x) + g(z) + \lambda^T (\mathbf{A}x + \mathbf{B}z - c) + \frac{\rho}{2}||\mathbf{A}x + \mathbf{B}z - c||_2^2 \tag{15}$$

The Lagrangian is minimized by updating the parameters x and z in an alternating fashion, as in Eqs. (16) and (17), combined with the dual update step in Eq. (18). Because of the augmentation term in the Lagrangian, the algorithm has been proven to converge under mild assumptions. On top of that, if there is an optimal, unique solution, ADMM will converge to that value [22].

$$x^{k+1} = \text{argmin}_x \mathcal{L}(x, z^k, \lambda^k) \tag{16}$$

$$z^{k+1} = \text{argmin}_z \mathcal{L}(x^{k+1}, z, \lambda^k) \tag{17}$$

$$\lambda^{k+1} = \lambda^k + \rho \left(\mathbf{A}x^{k+1} + \mathbf{B}z^{k+1} - c\right) \tag{18}$$

The z-update in Eq. (17) uses the updated value of the parameters x^{k+1}, which means that the steps can only be performed sequentially. There are schemes known using the values of the previous iteration, i.e. x^k in Eq. (17), but these do not necessarily have the same convergence properties, and are not even always convergent in general [25].

This algorithm is applicable to Simplotope B-Splines, as there are multiple, separable cost functions to be optimized, with global equality constraints. In that case there would be more than two alternating directions, unless the domain is subdivided into only two partitions. However, the general extension of this algorithm, with three of more alternating directions, is not necessarily convergent [26] and thus may lead to an unstable algorithm. Nevertheless, a general extension has been written and resulting in convergence towards the global optimal solution. However, this still leads to a sequential algorithm, as updates from the current iteration are required for convergence. However, not only a distributed algorithm, but a fully parallel algorithm is preferred, as this can run on multiple cores and thus have better scaling properties. Various options have been presented in literature to parallelize the algorithm, such as adding a proximal term, using a relaxation step, or a combination of both [25, 27, 28]. The Variable Splitting method as described in Section IV.B is used for the distributed aerodynamic model identification approach.

B. Variable Splitting ADMM

By making the algorithm in a distributed manner, the full tessellation is divided into $P \leq J$ partitions. This can either be partitions of a single simplotope, or with multiple simplotopes. The distributed algorithm iteratively optimizes

the coefficients, and enforces the constraints as composed in the continuity matrix $\mathbf{H}$. Using VSADMM, a vector of coupling coefficients z_p is introduced for each partition p, such that $\mathbf{H}_p \boldsymbol{c}_p = z_p$ [28, 29]. Here $\mathbf{H}_p$ are the columns of the continuity matrix $\mathbf{H}$ that are depending on the coefficients $\boldsymbol{c}_p$, i.e. $\mathbf{H} = \begin{bmatrix}\mathbf{H}_1 \dots \mathbf{H}_p \dots \mathbf{H}_P\end{bmatrix}$. To have the continuity conditions satisfied, the additional constraint $\sum_{p=1}^{P} z_p = 0$ is posed on these coefficients. Using this coupling coefficient, the optimization problem can be stated as in Eq. (19).

$$\begin{aligned} &\text{argmin}_{\boldsymbol{c}}\ \mathcal{J}(\boldsymbol{c}) = \textstyle\sum_{p=1}^{P} \frac{1}{2}||\mathbf{X}_p \boldsymbol{c}_p - \mathbf{Y}_p||_2^2 \\ &\text{subject to} \quad \mathbf{H}_p \boldsymbol{c}_p = z_p \\ &\qquad\qquad \textstyle\sum_{p=1}^{P} z_p = 0 \end{aligned} \tag{19}$$

Setting up the augmented Lagrangian in Eq. (20) for this optimization problem, the constraints put on the coupling coefficients are exchanged for an indicator function $\mathcal{I}_{\mathcal{Z}}$, where $\mathcal{Z}$ is the convex set $\left\{[z_1, \dots, z_P] : \sum_{p=1}^{P} z_p = 0\right\}$ [28]. This function will indicate 0 when the solution is part of the set $\mathcal{Z}$, and ∞ otherwise.

$$\mathcal{L}(\boldsymbol{c}, z, \lambda) = \sum_{p=1}^{P} \frac{1}{2}||\mathbf{X}_p \boldsymbol{c}_p - \mathbf{Y}_p||_2^2 + \mathcal{I}_{\mathcal{Z}} + \lambda^T \left(\mathbf{H}_p \boldsymbol{c}_p - z_p\right) + \frac{\rho}{2}||\mathbf{H}_p \boldsymbol{c}_p - z_p||_2^2 \tag{20}$$

The decoupling of the partitions is clear, as the Lagrangian in Eq. (20) is purely a sum of P individual Lagrangians of the partitions. By taking the derivative of this Lagrangian w.r.t. the coefficients $\boldsymbol{c}_p$, no influences from other partitions are present. Therefore, the optimal update as in Eq. (24) is found by setting the derivative to zero and solving for the coefficients $\boldsymbol{c}_p$.

The coupling coefficients are still linked to other partitions, as the sum of all has to be zero. Therefore this update is found in two steps, by first finding the unconstrained optimal update in Eq. (21).

$$z_p = \mathbf{H}_p \boldsymbol{c}_p + \frac{\lambda_p}{\rho} \tag{21}$$

The next step is to correct for the global constraints put on the coupling coefficients. For all constraints, this compensation can be done to subtract the sum of all partitions' unconstrained contributions to constraint j, divided by the number of partitions m_j that appear in that constraint [29]. Knowing that the $\mathbf{H}$ matrix is sparse, the rows are reduced in further computations. Let $\mathcal{H} = 1, 2, \dots, R$ be the indices of all continuity conditions, of which two subsets can be formed: $\mathcal{H}_p$ includes the indices of the internal constraints of partition p, and $\mathcal{H}_{p,n}$ those of the constraints between partition p and its direct neighbors. The rows of $\mathbf{H}_p$ are limited to $\mathcal{H}_p \cup \mathcal{H}_{p,n}$, which means that $z_p \in \mathbb{R}^{R_p}$, with $R_p = |\mathcal{H}_p \cup \mathcal{H}_{p,n}|$. This also makes that $\lambda_p \in \mathbb{R}^{R_p}$. For every partition, only the coupling coefficients of constraints in which it appears should be updated.

Using these reduced sets, the update rule for the coupling coefficients can be found in Eq. (22). For all constraints, the satisfaction of the global constraints as in $\mathbf{H}$ can be confirmed by summing all the contributions $z_{p,j}$.

$$z_{p,j} = \left(\mathbf{H}_{p,j}\boldsymbol{c}_p + \frac{\lambda_{p,j}}{\rho}\right) - \frac{1}{m_j}\left(\sum_{i=1}^{P}\mathbf{H}_{i,j}\boldsymbol{c}_i + \frac{\lambda_{i,j}}{\rho}\right), \quad \forall j \in \left(\mathcal{H}_p \cup \mathcal{H}_{p,n}\right) \tag{22}$$

In the case of Simplotope B-Splines, there are only two possibilities for m_j: either a constraint is internal, part of $\mathcal{H}_p$, which means that $m_j = 1$; or a constraint is part of $\mathcal{H}_{p,n}$, and then $m_j = 2$, because a continuity constraint is always between exactly two simplotopes. If $m_j = 1$, the coupling update will simply reduce to 0. The update steps for iteration $k+1$, that have to be carried out per partition, are shown in Eqs. (23) to (25).

$$z_{p,j}^{k+1} = \begin{cases} \left(\mathbf{H}_{p,j}\boldsymbol{c}_p^k + \frac{\lambda_{p,j}^k}{\rho}\right) - \frac{1}{2}\left(\sum_{i=1}^{P}\mathbf{H}_{i,j}c_i^k + \frac{\lambda_{i,j}^k}{\rho}\right), & \forall j \in \mathcal{H}_{p,n} \\ 0, & \forall j \in \mathcal{H}_p \end{cases} \tag{23}$$

$$\boldsymbol{c}_p^{k+1} = \left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\left(\mathbf{X}_p^T\boldsymbol{Y}_p + \rho\mathbf{H}_p^T\boldsymbol{z}_p^{k+1} - \mathbf{H}_p^T\lambda_p^k\right) \tag{24}$$

$$\lambda_p^{k+1} = \lambda_p^k + \rho\left(\mathbf{H}_p\boldsymbol{c}_p^{k+1} - \boldsymbol{z}_p^{k+1}\right) \tag{25}$$

The iterative updates of the coefficients and the duals in Eqs. (24) and (25) respectively do not depend in any case on updated iterations of other partitions. This means that each iteration can run fully in parallel, limited by either the number of partitions, or the amount of parallel pools that are available.

C. Complexity Analysis

The complexity of the VSADMM is analyzed and compared to a global estimation. A completely parallel implementation is assumed, with the necessary amount of parallel pools available and perfect communication between the partitions.

The coupling coefficient step in Eq. (23) can be updated with $O\left(R_p\right)$, with R_p the number of continuity conditions affecting partition p, as the matrix $\mathbf{H}_{p,j}$ is sparse. The same holds for the dual update step in Eq. (25). The amount of continuity conditions in a single partition is $O\left(\hat{d}\right)$ for a simplotope, and thus scales linearly with increasing partition size, but does not depend on the total number of partitions. Combining all the operations in the two update steps results in a $O\left(ck\hat{d}\right)$ complexity, where the constant c depends on the number of non-zeroes in the local constraint matrix and the multiplication and additions for Eqs. (23) and (25). Generally, $c < \hat{d}$, so the worst case is assumed as $c = \hat{d}$.

The most expensive operations are the inversion and vector-matrix multiplication in Eq. (24). Finding this results in a single $O(\hat{d}^3)$ operation, followed by k iterations with $O(2\hat{d}^2)$ operations, making the total complexity $O\left(\hat{d}^3 + 2k\hat{d}^2 + ck\hat{d}\right)$.

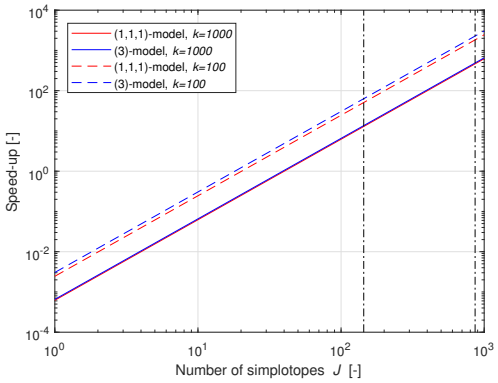


Fig. 2 Speed-up by using VSADMM, with vertical lines at 144 and 864 Simplotopes

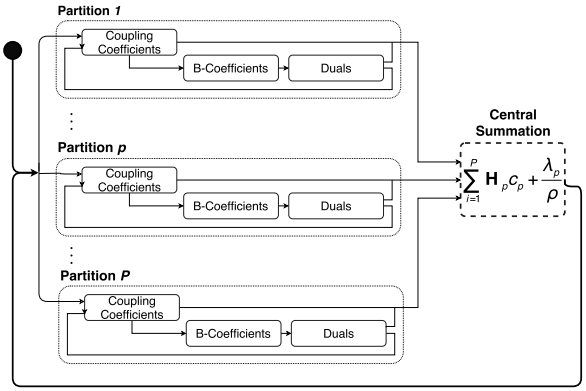


Fig. 3 Parallel flow of VSADMM

The algorithm gives the possibility of infinite scalability in terms of refinement of the tessellations, as this only increases the number of partitions, but not the individual size. Therefore, if a tessellation is refined from J to $2J$ simplices, adding J parallel pools will result in no increment in the computational cost of estimating the parameters.

The complexity is compared to the commonly used global method, the iterative solver in [16]. The solver typically converges in 2-3 iterations, requiring $O\left(J^2\hat{d}^2\right)$ FLOPs per iteration because of the sparse, global matrix that is solved not needing a $J^3\hat{d}^3$ operation first because of the sparseness of the matrix. For the distributed approach, assuming two iterations for the global solver, this will result in the speed-up in Eq. (26) when VSADMM works fully in parallel.

$$O\left(\frac{2J^2\hat{d}^2}{\hat{d}^3 + 2k\hat{d}^2 + ck\hat{d}}\right) = O\left(\frac{2J^2}{\hat{d} + 3k}\right) \tag{26}$$

Equation (26) breaks even at $J = \sqrt{\frac{1}{2}(\hat{d} + 3k)}$. The speed-up is shown in Fig. 2 for two models that will be introduced in Table 3 in Section V: a $(1, 1, 1)$-model with 216 coefficients per simplotope and a (3)-model with 56 coefficients. Both have $k = 1,000$ iterations, which is generally sufficient to obtain a good approximation of the aerodynamic model. Note that in these cases the $\hat{d}$ is small compared to the amount of iterations. For reference, two lines for 100 iterations are added as well.

For both models the distributed method is beneficial from approximately 45 simplotopes. The two vertical lines indicate the 144 partitions that are being used in the $(1, 1, 1)$-model, which has a speed-up of approximately 10, and the 864 simplotopes in the (3)-model, that can achieve a speed-up as high as 500. This only improves when e.g. 100 iterations are used. It clearly indicates that the distributed approach achieves increasingly beneficial speed-ups with increasing number of partitions. Reducing the number of iterations will have a major effect on this speed-up, with some suggestions provided in Section IV.D.

This parallel workflow is depicted in Fig. 3. The communication overhead between different partitions is limited by the unconstrained, optimal coupling update in Eq. (21). These are collected and summed at a central point, after which the sum $\frac{1}{2}\sum_{i=1}^{P}\mathbf{H}_i\boldsymbol{c}_i + \frac{\lambda_i}{\rho}$, as used in Eq. (22), needs to be communicated back to the partitions. This centralized operation will receive $\sum_{p=1}^{P}|\mathcal{H}_{p,n}|$ scalars from the partitions, and will have to communicate as many back after they have been summed. The complexity analysis, and thus the speed-up, does not take any transport latency into account that may occur when these values are communicated back and forth.

D. ADMM Variations and Improvements

As a lot of research has been done into ADMM algorithms, numerous improvements and speed-ups are found [22]. First, a stopping criterion has been defined Section IV.D.1, which sets feasibility tolerances on the dual- and the primal function for accepting suboptimal, but sufficiently optimal, solutions. While exploring, it occurred that the choice of the penalty factor ρ had a major effect on the convergence properties and the quality of the solution. In order to make the algorithm more robust against this initial choice of ρ, it is made adaptive as explained in Section IV.D.2. More options and variations are available for ADMM, such as over-relaxation and fast-ADMM [22, 30], but these are not implemented.

1. Stopping Criterion

In order to assess the VSADMM's solution during the iterations, two residuals are used [22]: the primal residual $\boldsymbol{r}^{k+1}$, indicating the feasibility of the primal function, and the dual residual $\boldsymbol{s}^{k+1}$, indicating the feasibility of the dual function and thus the optimality of the primal function. These two residuals for iteration $k+1$ are given by Eqs. (27) and (28). The constraints on the coupling coefficients, $\sum_{p=1}^{P}\boldsymbol{z}_p = 0$, have not been included here, as the update step for $\boldsymbol{z}_p$, in Eq. (23), per definition enforces these constraints.

$$\boldsymbol{r}_p^{k+1} = \mathbf{H}_p\boldsymbol{c}_p^{k+1} - \boldsymbol{z}_p^{k+1} \tag{27}$$

$$\boldsymbol{s}_p^{k+1} = -\rho\mathbf{H}_p\left(\boldsymbol{c}_p^{k+1} - \boldsymbol{c}_p^{k}\right) \tag{28}$$

The stopping criteria can be defined as $||\boldsymbol{r}_p^{k+1}||_2 \leq \epsilon^{pri}$ and $||\boldsymbol{s}_p^{k+1}||_2 \leq \epsilon^{dual}$, with the tolerances ϵ as defined in Eqs. (29) and (30). Here $\epsilon^{abs} \geq 0$ and $\epsilon^{rel} \geq 0$ are the absolute and relative tolerance respectively, and can be chosen to set the desired accuracy. Here, the accuracies are set to be $\epsilon^{abs} = 10^{-8}$ and $\epsilon^{rel} = 10^{-6}$ for the absolute and relative accuracy respectively. The stopping criterion is also defined per partition, such that each partition will have a primal- and dual

feasible solution.

$$\epsilon^{pri} = \sqrt{\hat{d}_p}\epsilon^{abs} + \epsilon^{rel} \max\left\{||\mathbf{H}_p \boldsymbol{c}_p^{k+1}||_2, || - \boldsymbol{z}_p^{k+1}||_2\right\} \tag{29}$$

$$\epsilon^{dual} = \sqrt{R_p}\epsilon^{abs} + \epsilon^{rel}||\mathbf{H}_p^T \lambda_p^{k+1}||_2 \tag{30}$$

Additionally to the stopping criterion, a maximum amount of iterations should be set in case the criterion will not be reached. A sufficiently high number, such as 1,000, can be chosen to obtain a suboptimal solution. In this case the solution should be treated and analyzed carefully, as it might not be close to an optimal or feasible solution yet.

2. Adaptive Penalty Factor

To make the algorithm more robust against the choice for penalty factor ρ, a method that changes this parameter per iteration has proven to be effective in practice [31, 32]. Since it is best to let the dual and the primal converge to feasibility at the same pace, the two residuals are compared as in Eq. (31).

$$\rho_{k+1} = \begin{cases} \rho_k \cdot (1 + \tau_k) & \text{if } ||\boldsymbol{r}^k||_2 > \mu \cdot ||\boldsymbol{s}^k||_2 \\ \rho_k \cdot \frac{1}{1+\tau_k} & \text{if } \mu \cdot ||\boldsymbol{r}^k||_2 < ||\boldsymbol{s}^k||_2 \\ \rho_k & \text{otherwise} \end{cases} \tag{31}$$

This adaptive scheme increases ρ when the primal residual is more than a factor μ higher than the dual residual, and decreases ρ when it is μ times lower. In this way it changes the focus from primal to dual feasibility, if the two are converging at a different pace. In this case, τ_k has been chosen as 1, and $\mu = 5$. At every update of the penalty factor, the new iteration is initiated with the coefficients and duals from the previous iteration.

To make the estimation stable, $\sum_{k=1}^{\infty} \tau_k$ should be finite [31, 32]. For this reason, the penalty factor is only allowed to change 5 times before it has to remain constant and $\tau = 0$.

V. Simulations and Results

The results are divided into two main parts. First, in Section V.A the performance of the distributed approach is analyzed by comparing the distributed solutions with the global solution. This has been done with the adaptive penalty factor, and two different starting values for ρ: 1 and 0.01. The tests are performed on a 3-dimensional model, to which noise has been added of various noise intensities. The models with first order continuity, that have been used in the estimation of the ICE model, are implemented in Section V.B and compared to the ICE's Simulink model. For all results, the dataset and hypercube partitioning from [18] have been used. These are triangulated using Kuhn's triangulation. This triangulation method is chosen over others, such as Delaunay, because it gives conforming simplices, required for

good edges, and simplices of a single congruence class, providing practical benefits [33]. The same, extended datasets have been used as well, using single imputed datapoints, allowing an increase of model degree.

A. Evaluation of the Distributed Approach

This section assesses the performance of the VSADMM algorithm compared to the global solution. The datasets are artificially contaminated with noise by using a method described in [8], in order to show the performance in different noisy environments. For a noise intensity of k%, the noise can be generated by using a uniform distribution $\nu = U(-0.5, 0.5)$ and Eq. (32).

$$\nu(k) = 0.01 \cdot k \cdot (\max \boldsymbol{Y} - \min \boldsymbol{Y}) \cdot \nu \tag{32}$$

Before the noise is added to the data, a distinction is made between the identification data and the validation data. Noise has only been added to the identification data, and the methods are validated by using clean data. The validation set consists of 20% of the full set and is chosen randomly.

Four different tessellations of the model have been compared, as described in Table 3. The combinations are chosen in order to show performance at several layer divisions, from fully decoupled to fully coupled dimensions. The $(2, 1)$-simplotope contains α and β together in one layer and M in its own layer. The (3)-simplotope model has all three dimensions in a single layer and is essentially a 3-dimensional simplex spline. In order to assess the adaptability of the distributed method in a bigger partition size, the simplex model has also been partitioned per hypercube, each consisting of 6 simplices.

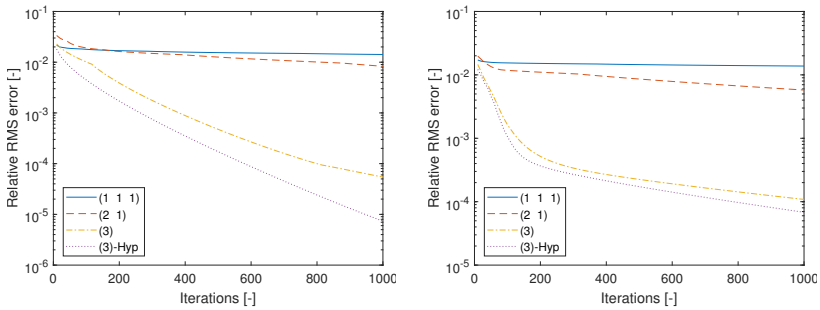
Table 3 Layer compositions of $C_{X_2}(\alpha, \beta, M)$ used for noise comparisons.

Layer Division	Dimensions	Continuity	Simplotopes	Partitions
$(1, 1, 1)$	$(5, 5, 5)$	$(1, 1, 1)$	144	144
$(2, 1)$	$(5, 5)$	$(1, 1)$	296	296
(3)	(5)	(1)	864	864
(3)	(5)	(1)	864	144

Because of the tensor product between the layers, the $(1, 1, 1)$ model consists of terms with a total degree of 15. These terms might result in unrealistic model terms that help fitting a good model through the identification data, but result in high validation errors.

The maximum amount of iterations that has been set is 1,000 for the distributed approach. The global solution is found using the iterative solver described in [16], which has been limited to a maximum of 250 iterations. The distributed solver does generally not reach the stopping criterion and is thus limited the maximum amount of iterations, while the global iterative solver typically reaches convergence before that.

Figure 4 shows convergence properties of the distributed algorithm for the several tessellations. It computes the relative RMS between the intermediate solutions of the VSADMM and the global solutions. The $(1, 1, 1)$- and $(2, 1)$-simplotopes advance to the the global solution slowly after approximately 200 iterations, but reaches within a few percent. On the other hand, the (3)-models approximate the global solution better and keep on improving until 1,000 iterations. The hypercube partitioned model performs slightly better, but is also computationally more complex. The simplotope models benefit slightly from reducing the initial penalty factor ρ to 0.01, while this has a worsening effect on the simplex models. Possible explanations for the difference between simplotope and simplices are that degrees of freedom it contains. Simplices have considerably more constraints than simplotopes, giving the simplotope spline more coefficients to be optimized. To conclude, even though the adaptive penalty factor reduces the effect of a bad initial choice, it still has an influence and thus should be considered at initialization of the algorithm.

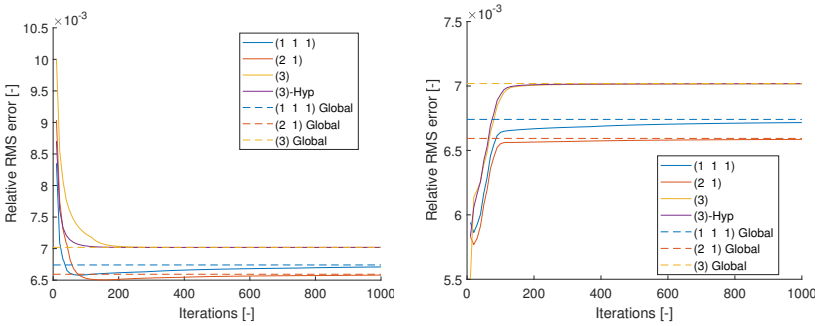


(a) $\rho = 1.0$ **(b)** $\rho = 0.01$

Fig. 4 Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity.

Similarly, at 0% noise, Fig. 5 shows that the validation performance of the distributed algorithm, in terms of relative RMS validation error, converges to the global results as well. Comparing Figs. 5a and 5b shows a difference between how the choice of penalty factor affects the approach to the global solution. A high ρ (Fig. 5a) puts more weight on satisfying the constraints and is less optimal at first than starting with a low ρ in Fig. 5b. When iterating and thus enforcing the constraints more, will make the results advance to the global solution. Because these figures are made at 0% noise, local estimations are likely to be better than the ones with continuity more strictly enforced.

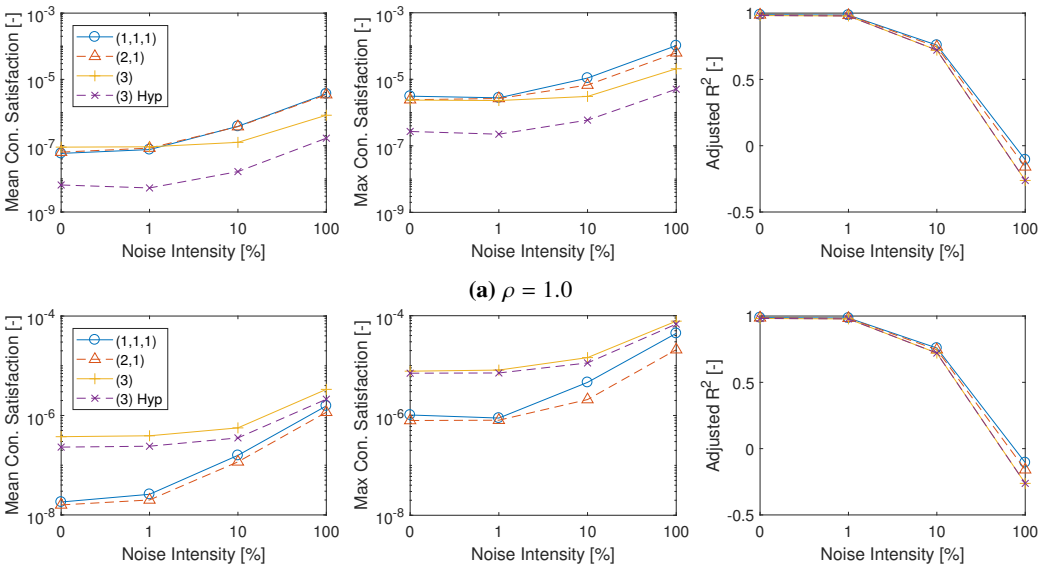
To assess constraint satisfaction, two metrics are used: the mean constraint mismatch, $||\mathbf{H}\boldsymbol{c}||_1/R$, and the maximum constraint mismatch, $||\mathbf{H}\boldsymbol{c}||_\infty$. Figure 6 shows that in 1,000 iterations, in all cases the constraints are approximately satisfied for both settings of the penalty factor. It is not completely (numerically) zero yet, as would be the case with global methods, but is getting close with a maximum mismatch at 100% noise in the order of 10^{-4}. The common trend is that the constraint satisfaction worsens with increasing noise levels, indicating that noisy environments would require more iterations in case the same quality of continuity is desired.



(a) $\rho = 1.0$ **(b)** $\rho = 0.01$

Fig. 5 Convergence of the validation relative RMS error of the distributed algorithm towards the global optimum, at 0% noise intensity.

In Fig. 6a the simplex models show better continuity than the simplotopes, while Fig. 6b show the opposite. In both cases the coefficients of determination show equal values, with simplotopes that have more decoupled dimensions, performing better in fitting to the identification data. The fact that the simplex models show the same coefficient of determination in both partition sizes, verifies these have the same solutions.



(a) $\rho = 1.0$

(b) $\rho = 0.01$

Fig. 6 Mean and Maximum constraint satisfaction and the coefficient of determination for the 3-dimensional test cases.

Comparisons in Fig. 7 indicate that for all layer combinations the fitness to the identification data and the validation results are close or equal to that of the globally estimated solution. A slight difference in the maximum relative residual can be found in Fig. 7a for the $(1, 1, 1)$-simplotope spline. However, it is remarkable that in this case the distributed,

suboptimal solution performs better than the global solution.

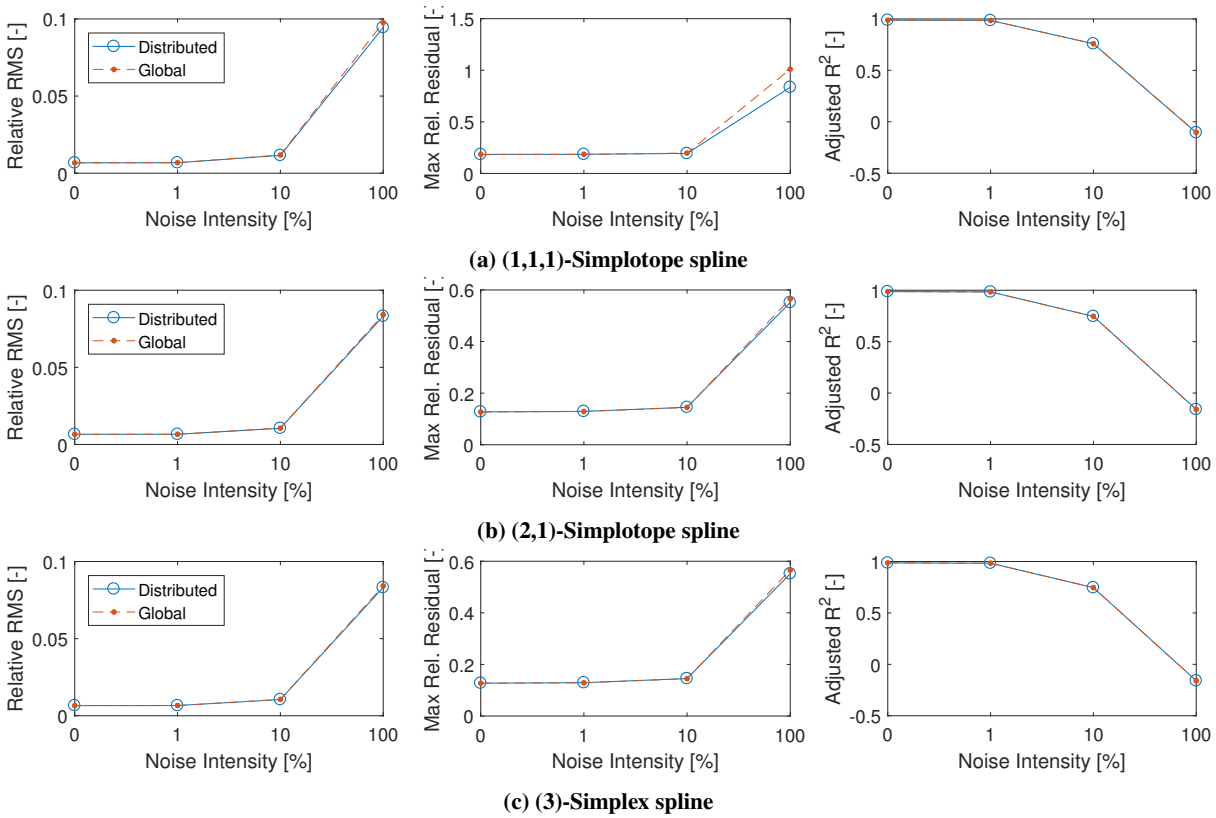
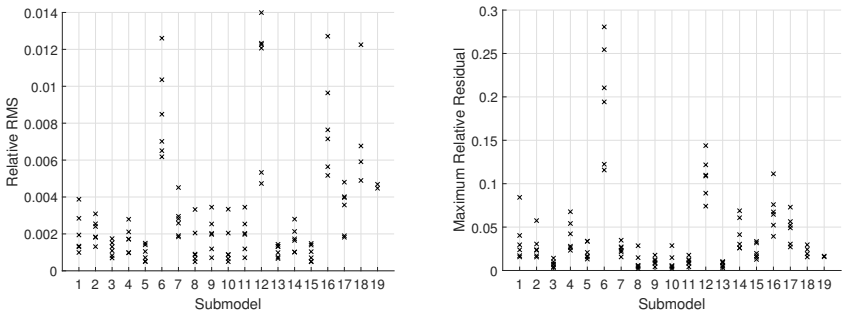


Fig. 7 Comparison between the results of the distributed- and global approaches in terms of: relative RMS validation error, relative maximum validation residual, and adjusted coefficient of determination. ($\rho = 1.0$.)

When comparing the results between the different model structures in Fig. 7, the RMS and the coefficients of determination are comparable. The (1, 1, 1)-simplotope structure shows slightly better performance in the case of the fitness, as can be seen in Fig. 6 as well. On the other hand, the RMS shows slightly worse performance when more dimensions are decoupled at higher noise intensities. The maximum residual of the (1, 1, 1)-simplotope, is considerably worse than those of the other two model structures, with residuals up to 100%. The Simplotope B-Spline theory in Section III already indicated that high cross-coupling terms occur in simplotope polynomials. In the (1, 1, 1)-simplotope model with degrees (5, 5, 5) consists of polynomial terms with a combined degree of 15. Combining these high degree terms with the higher coefficient of determination and bad maximum residual, this indicates an overfitting regression. This is especially harmful at high noise intensities, because the unnatural terms might have a statistical significant effect in fitting the model, while in reality there is none.

B. Implementation in the Innovative Control Effectors model

Various model combinations for all the submodels, in terms of layers' divisions and degrees, are tested in order to find the best performancing models. The combinations that were found to perform best were implemented in the final ICE Simplotope model. The details of the models can be found in Table 5 in the Appendix. The validation results of the chosen models can be seen in Fig. 8, showing the relative RMS in Fig. 8a and the maximum relative residual in Fig. 8b. The maximum RMS of any submodel is 1.40%, while the maximum relative error that is found is 28.1%. Of all 108 submodels, the median of the RMS is at 0.192% and that of the maximum residual at 2.33%. These results are comparable to the validation results of the zeroth order continuity model developed in [18].



(a) RMS error **(b) Maximum residual**

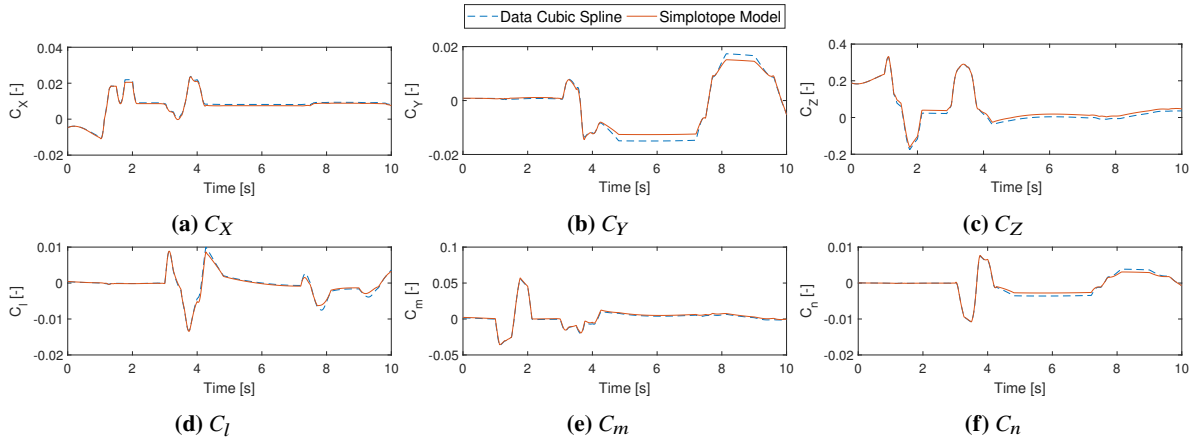
Fig. 8 Relative validation results of all the implemented simplotope models

The performance of the Simplotope model was compared with the model as implemented in Simulink. In order to make the Simplotope model able to be evaluated in real time, an optimized evaluation function had to be written. For ICE's Simplex model a `MEX` function was written in C++ [18], which has been extended to work with Simplotopes as well and is able to perform over 2,000 evaluations every second*. The `MEX` function was placed in Simulink as an embedded `Matlab` block. The version of ICE's Simulink model used for comparison is the one that uses a cubic spline to interpolate the measurements and clips all values as extrapolation mode. Moreover, the inputs are combined with the linear models for the actuators, in order to provide a more realistic input. [3, 18, 19].

The first input is a doublet on both elevons symmetrically, followed by a block input on the left AMT and a block input on the right AMT. The resulting model responses are given in Fig. 9.

It can be seen that the model generally does not have any problems with following the coefficients as determined by the Simulink model. However, after approximately 4 seconds, the estimations start to differ slightly. When comparing this point in time with the states' responses in Fig. 10, it corresponds to the moment the Angle of Attack reaches out of bounds. Although both the models have clipping extrapolation modes, yet a difference occurs.

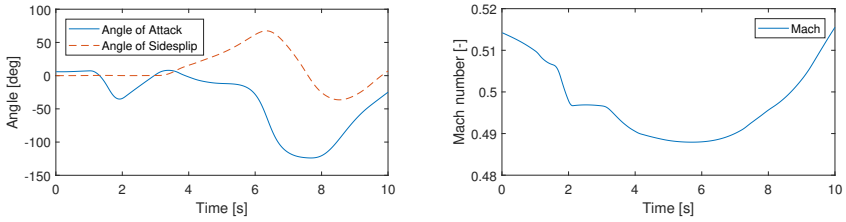
*With Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz



(a) C_X **(b)** C_Y **(c)** C_Z

(d) C_l **(e)** C_m **(f)** C_n

Fig. 9 Simplotope Models' Results compared with ICE's Simulink Model after elevons and AMT inputs



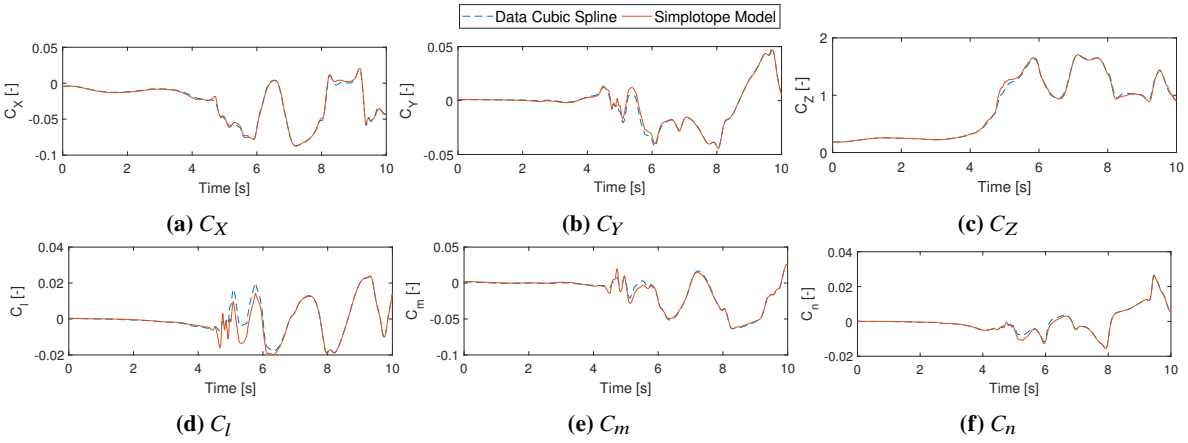
(a) Angle of Attack and Angle of Sideslip **(b) Mach number**

Fig. 10 Aerodynamic states' responses to the elevons and AMT inputs.

The second input is a combined input of the Differential Leading Edge Flaps and the Spoiler Slot Deflectors, with the responses shown in Fig. 11. Although the difference between the two responses are bigger than above, the simplotopes still provide a good approximation of the Simulink model.

The aerodynamic states resulting from the inputs are shown in Fig. 12. Here the aerodynamic angles stay within the range of the simplotope models, but the Mach number in Fig. 12b reduces below 0.3, which is outside the boundaries. However, the inaccuracies in Fig. 11 are not due to this behavior, as this occurs before the mach number reaches out of bounds. The inaccuracies at 5-6 seconds occur becuase that is the moment the Spoiler Slot Deflector are extended, with several models reacting badly at high inputs of those.

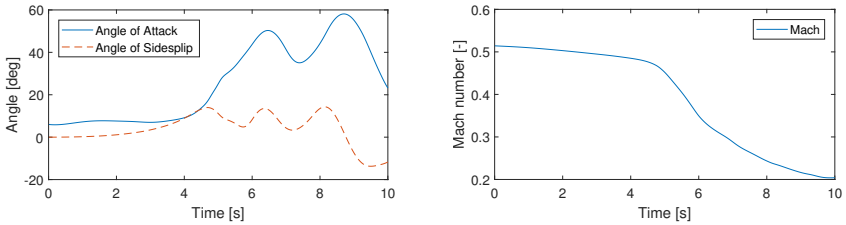
An overview of these and two other inputs is given in Table 4. Two additional scenarios are: no inputs at all during 10 seconds and sequentially deflecting all 11 control effectors. It can be seen that the maximum RMS that is seen in this case is 6.54% between the original baseline model and the estimation of all 108 simplotope methods. There is no coefficient that performs generally better or worse than the others. The difference between the total results and the individual submodel's results in Fig. 8 can be explained by the fact that many models get out of bounds during operation, while the error is cumulated through all the models.



(a) C_X **(b)** C_Y **(c)** C_Z

(d) C_l **(e)** C_m **(f)** C_n

Fig. 11 Simplotope Models' Results compared with ICE's Simulink Model after DLEF and SSD inputs



(a) Angle of Attack and Angle of Sideslip **(b) Mach number**

Fig. 12 Aerodynamic states' responses to the elevons and AMT inputs.

Table 4 Scaled RMS between the Simulink data and the Simplotope Model during 10 seconds.

Input	C_X	C_Y	C_Z	C_l	C_m	C_n
None	2.14%	1.90%	2.82%	3.36%	2.75%	2.18%
Elevons / AMTs	1.75%	4.39%	2.50%	1.99%	1.23%	2.78%
DLEFs / SSDs	1.54%	2.10%	1.71%	6.54%	2.76%	1.68%
All	1.20%	1.56%	1.55%	6.41%	2.49%	3.03%

Another observation from the simplotope model responses in Figs. 9 and 11 is that it contains no visible ripples or discontinuities, even though the distributed approach has been taken. This justifies the reason why first order continuity has been enforced, as the line is smoothly following the original model. Moreover, it is a confirmation of the performance of the distributed approach in terms of constraint satisfaction.

VI. Conclusion

A novel distributed approach for Aerodynamic Model Identification has been presented, using the Alternating Direction Method of Multipliers (ADMM) in combination with multivariate Simplotope B-Splines. The simplotope splines have the possibility to decouple dimensions into separate layers, containing lower dimensional simplex splines;

each of these have their own individual properties as triangulation, model degree, and continuity order. These layers are combined by taking the tensor product, resulting in simplotopes with an own set of barycentric coordinates and B-Coefficients. Using simplotopes gives more flexibility in choosing a model structure, gives better shaped domains, and reduces the number of coefficients and constraints. However, it may also introduce high degree cross coupling terms that may result in unwanted overfitting.

The distributed approach divides the full tessellation into a number of partitions. The Variable Splitting ADMM (VSADMM) method fully decouples the estimation of the coefficients of the individual partitions. The global continuity matrix is split up by introducing coupling coefficients for each constraint. Global continuity is enforced, by requiring the sum of the coupling coefficients, over all partitions, to be zero. Because VSADMM is able to work fully in parallel, the complexity is determined solely by the partition size, and refining the grid will not increase complexity as parallel pools can be added. The only centralized operation is gathering the coupling coefficients, summing them, and return the sum to the partitions. This limits the communication between the partitions to twice as many scalars as there are continuity equations.

The performance of the distributed approach has been evaluated by adding noise at various intensities to a 3-dimensional submodel of the Innovative Control Effectors (ICE) aircraft. This model has been evaluated for several combinations of layers of the dimensions and different partition sizes, for initial penalty factors ρ of 1.0 and 0.01.

The simplotope models did converge to a value within a few percents of the global optimal solution, but the pure simplex models kept on advancing beyond an RMS of 10^{-4} compared to the global solution. Bigger sized partition performed better and are beneficial for the quality, but is also computationally more expensive.

In all cases the global continuity constraints were enforced well, with maximum mismatches in the order of 10^{-4} after 1,000 iterations, only for the highest noise levels. In general, higher noise intensities resulted in worsened enforcement of the constraints, and thus require more iterations to get equal continuity quality. A higher penalty factor was beneficial for the pure simplex model, while the simplotopes performed better with lower penalty factor. Even though an adaptive penalty factor has been used, the initial choice still affects the final solution of the ADMM algorithm.

For all combinations, the validation performance, in terms of relative RMS error, and maximum relative error, were similar to the global results, as well as the fitness to the identification data in terms of adjusted coefficient of determination. In case of the $(1, 1, 1)$-simplotope model, the global solution was a little worse than the distributed one. This model showed considerably worse maximum residuals, when the noise intensity reached 100%. Because the coefficient of determination is still high, this is a result of overfitting, caused by the high degree polynomial terms formed by the tensor product between the layers.

The distributed approach is used to estimate the aerodynamic model of the ICE-aircraft with first order continuity. All 108 submodels were estimated with several layer combinations, and the best in terms of validation RMS error and

maximum residual, were used in the final model. All models had a relative RMS below 1.4%, and the maximum relative error was 28.1%. These values are similar to the results of the zeroth order continuous model.

For real time simulation in Simulink, all the models were implemented in an optimized `MEX` evaluation function, able to perform over 2,000 evaluations every second. The model followed the original model with RMSs of 7% maximum, which is mainly because of the aerodynamic states getting out of bounds of the model domain. Contrary to the zeroth order continuity model, it showed no ripples due to discontinuities in the first derivative. This justifies the higher order continuity, and confirms the functionality of the distributed estimator when it comes to constraint satisfaction.

Appendix

Table 5 Overview of the simplotope models of the ICE aircraft and their properties.

Coefficient	Parameters	Data Range	Datapoints	Cubes	Layers	Degrees	Simplotopes	Coefficients
C_{*_1}	α	[-5.0, 90.0]	48	12	(1, 1)	(7, 6)	120	6,720
	Mach	[0.3, 2.16]	10	10				
C_{*_2}	α	[-5, 90]	30	6	(1, 1, 1)	(7, 6, 6)	144	56,448
	β	[-30, 30]	15	6				
	Mach	[0.3, 2.16]	8	4				
C_{*_3}	α	[-2.5, 45]	20	10	(1, 1, 1)	(3, 3, 3)	10	640
	β	[-10, 10]	3	1				
	δ_{LIBLEF}	[0, 40]	2	1				
C_{*_4}	α	[-2.5, 45]	20	5	(1, 1, 1, 1)	(5, 5, 5, 5)	10	45,360
	β	[-10, 10]	3	1				
	δ_{LIBLEF}	[0, 40]	2	1				
	δ_{LOBLEF}	[-40, 40]	5	1				
	Mach	[0.3, 1.2]	4	1				
C_{*_5}	α	[-2.5, 90]	28	10	(1, 1, 1, 1)	(5, 4, 4, 5)	160	144,000
	δ_{LSSD}	[0, 60]	4	2				
	δ_{LEL}	[-30, 30]	5	4				
	Mach	[0.3, 2.16]	7	2				
C_{*_6}	α	[-2.5, 90]	29	10	(2, 1, 1, 1)	(3, 1, 1, 3)	80	12,800
	δ_{LSSD}	[0, 60]	2	1				
	δ_{RSSD}	[0, 60]	2	1				
	δ_{PF}	[-30, 30]	5	2				
	Mach	[0.3, 2.16]	7	2				
C_{*_7}	α	[-2.5, 90]	29	10	(1, 1, 1)	(5, 5, 4)	40	7,200
	β	[-30, 30]	7	2				
	δ_{LAMT}	[0, 60]	5	2				
C_{*_8}	α	[-2.5, 42.5]	19	9	(1, 1, 1)	(4, 3, 3)	9	720
	δ_{LEL}	[-30, 30]	3	1				
	δ_{LAMT}	[0, 60]	3	1				
C_{*_9}	α	[-2.5, 42.5]	19	9	(1, 1, 1)	(4, 3, 3)	9	720
	δ_{LOBLEF}	[0, 40]	3	1				
	δ_{LAMT}	[0, 60]	3	1				
$C_{*_{10}}$	α	[-2.5, 42.5]	19	9	(1, 1, 1)	(4, 3, 3)	9	720
	δ_{REL}	[-30, 30]	3	1				
	δ_{RAMT}	[0, 60]	3	1				

$C_{*_{11}}$	α	[-2.5, 42.5]	19	9	(1, 1, 1)	(4, 3, 3)	9	720
	δ_{ROBLEF}	[0, 40]	3	1				
	δ_{RAMT}	[0, 60]	3	1				
$C_{*_{12}}$	α	[-2.5, 90]	29	15	(1, 1, 1)	(4, 3, 3)	90	7,200
	β	[-30, 30]	7	3				
	δ_{LSSD}	[0, 60]	4	2				
$C_{*_{13}}$	α	[-2.5, 45]	20	10	(1, 1, 1)	(3, 3, 3)	10	640
	β	[-10, 10]	3	1				
	δ_{RIBLEF}	[0, 40]	2	1				
$C_{*_{14}}$	α	[-2.5, 45]	20	5	(1, 1, 1, 1)	(5, 5, 5, 5)	10	45,360
	β	[-10, 10]	3	1				
	δ_{RIBLEF}	[0, 40]	2	1				
	δ_{ROBLEF}	[-40, 40]	5	1				
	Mach	[0.3, 1.2]	4	1				
$C_{*_{15}}$	α	[-2.5, 90]	28	10	(1, 1, 1, 1)	(5, 4, 4, 5)	160	144,000
	δ_{RSSD}	[0, 60]	4	4				
	δ_{REL}	[-30, 30]	5	2				
	Mach	[0.3, 2.16]	7	2				
$C_{*_{16}}$	α	[-2.5, 90]	29	10	(1, 1, 1)	(4, 4, 3)	40	4,000
	β	[-30, 30]	7	2				
	δ_{RAMT}	[0, 60]	5	2				
$C_{*_{17}}$	α	[-2.5, 90]	29	15	(1, 1, 1)	(4, 4, 3)	90	9,000
	β	[-30, 30]	7	3				
	δ_{RSSD}	[0, 60]	4	2				
$C_{*_{18,19}}$	α	[0, 30]	13	3	(1, 1)	(5, 5)	12	432
	Mach	[0.6, 2.2]	6	4				

References

[1] Tol, H. J., De Visser, C. C., van Kampen, E., and Chu, Q. P., "Nonlinear Multivariate Spline-Based Control Allocation for High-Performance Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1840–1862. doi:10.2514/1.G000065, URL http://arc.aiaa.org/doi/10.2514/1.G000065.

[2] Tol, H. J., De Visser, C. C., Sun, L. G., van Kampen, E., and Chu, Q. P., "Multivariate Spline-Based Adaptive Control of High-Performance Aircraft with Aerodynamic Uncertainties," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 4, 2016, pp. 781–800. doi:10.2514/1.G001079, URL http://arc.aiaa.org/doi/10.2514/1.G001079.

[3] Matamoros, I., and De Visser, C. C., "Incremental Nonlinear Control Allocation for a Tailless Aircraft with Innovative Control Effectors," *2018 AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, 2018, p. 220. doi:doi:10.2514/6.2018-1116, URL https://doi.org/10.2514/6.2018-1116.

[4] Klein, V., and Morelli, E. A., *Aircraft System Identification: Theory and Practice*, American Institute of Aeronautics and Astronautics, Inc., 2006.

[5] Hong, X., Mitchell, R. J., Chen, S., Harris, C. J., Li, K., and Irwin, G. W., "Model selection approaches for non-linear system identification: A review," *International Journal of Systems Science*, Vol. 39, No. 10, 2008, pp. 925–946. doi: 10.1080/00207720802083018.

[6] Farin, G., "A History of Curves and Surfaces in CAGD," *Handbook of computer aided geometric design*, Elsevier, 1984, Chap. 1, pp. 1–23.

[7] Anderson, I. J., Cox, M. G., and Mason, J. C., "Tensor-product spline interpolation to data on or near a family of lines," *Numerical Algorithms*, Vol. 5, No. 4, 1993, pp. 193–204. doi:10.1007/BF02210502, URL https://doi.org/10.1007/BF02210502http://link.springer.com/10.1007/BF02210502.

[8] De Visser, C. C., "Global Nonlinear Model Identification with Multivariate Splines," Ph.D. thesis, Delft University of Technology, 2011.

[9] De Visser, C. C., Chu, Q. P., and Mulder, J. A., "A new approach to linear regression with multivariate splines," *Automatica*, Vol. 45, No. 12, 2009, pp. 2903–2909. doi:10.1016/j.automatica.2009.09.017.

[10] Lai, M.-J., and Schumaker, L. L., *Spline Functions on Triangulations*, 1st ed., Cambridge University Press, Cambridge, 2007. URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780511889479.

[11] Govindarajan, N., De Visser, C. C., and Krishnakumar, K., "A sparse collocation method for solving time-dependent HJB equations using multivariate B-splines," *Automatica*, Vol. 50, No. 9, 2014, pp. 2234–2244. doi:10.1016/j.automatica.2014.07.012, URL http://dx.doi.org/10.1016/j.automatica.2014.07.012.

[12] De Visser, C. C., Brunner, E., and Verhaegen, M., "On distributed wavefront reconstruction for large-scale adaptive optics systems," *Journal of the Optical Society of America A*, Vol. 33, No. 5, 2016, pp. 817–831. doi:10.1364/JOSAA.33.000817.

[13] Visser, T., De Visser, C. C., and Van Kampen, E.-J., "Quadrotor System Identification Using the Multivariate Multiplex B-Spline," *AIAA Atmospheric Flight Mechanics Conference*, American Institute of Aeronautics and Astronautics, Kissimmee, Florida, 2015, pp. 1–13. doi:doi:10.2514/6.2015-0747, URL https://doi.org/10.2514/6.2015-0747.

[14] Visser, T., De Visser, C. C., and van Kampen, E.-J., "Towards the multivariate simplotope spline : continuity conditions in a class of mixed simplotopic grids," , 2016.

[15] Sun, L. G., De Visser, C. C., and Chu, Q. P., "A New Substitution Based Recursive B-Splines Method for Aerodynamic Model Identification," *Advances in Aerospace Guidance, Navigation and Control: Selected Papers of the Second CEAS Specialist Conference on Guidance, Navigation and Control*, edited by Q. Chu, B. Mulder, D. Choukroun, E.-J. van Kampen, C. de Visser, and G. Looye, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 233–245.

[16] Awanou, G., Lai, M.-J., and Wenston, P., "The Multivariate Spline Method for Scattered Data Fitting and Numerical Solutions of Partial Differential Equations," *Wavelets and Splines*, Nashboro Press, Brentwood, Tennessee, 2005, pp. 24–75.

[17] Silva, J., Keviczky, T., and Verhaegen, M., "A Distributed Approach for Solving Wavefront Reconstruction Problems," *American Control Conference (ACC)*, Boston, 2016, pp. 6513–6518. URL ieeexplore.ieee.org/document/7526695/.

[18] van der Peijl, I. V., "Physical Splines for Aerodynamic Modelling of Innovative Control Effectors," Master's thesis, Delft University of Technology, 2017.

[19] Niestroy, M. A., Dorsett, K. M., and Markstein, K., "A Tailless Fighter Aircraft Model for Control-Related Research and Development," *AIAA Modeling and Simulation Technologies Conference*, 2017, pp. 1–18. doi:10.2514/6.2017-1757, URL http://arc.aiaa.org/doi/10.2514/6.2017-1757.

[20] Dorsett, K. M., and Mehl, D. R., "Innovative Control Effectors (ICE) Phase I," Tech. rep., Lockheed Martin Tactical Aircraft Systems, 1996. doi:WL-TR-96-3043, URL http://oai.dtic.mil/oai/oai?verb=getRecord{&}metadataPrefix=html{&}identifier=ADB041508.

[21] Dorsett, K. M., Fears, S. P., and Houlden, H. P., "Innovative Control Effectors (ICE) Phase II," Tech. rep., Lockheed Martin Tactical Aircraft Systems and Bihrle Applied Research, Inc., 1997. doi:WL-TR-96-3043.

[22] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J., "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, Vol. 3, No. 1, 2010, pp. 1–122. doi:10.1561/2200000016, URL http://www.nowpublishers.com/article/Details/MAL-016.

[23] Ghadimi, E., Teixeira, A., Shames, I., and Johansson, M., "Optimal Parameter Selection for the Alternating Direction Method of Multipliers (ADMM): Quadratic Problems," *IEEE Transactions on Automatic Control*, Vol. 60, No. 3, 2015, pp. 644–658. doi:10.1109/TAC.2014.2354892, URL http://arxiv.org/abs/1306.2454{%}0Ahttp://dx.doi.org/10.1109/TAC.2014.2354892.

[24] Wohlberg, B., "ADMM Penalty Parameter Selection by Residual Balancing," , 2017. URL http://arxiv.org/abs/1704.06209.

[25] He, B., Hou, L., and Yuan, X., "On full Jacobian decomposition of the augmented Lagrangian method for separable convex programming," *SIAM Journal on Optimization*, Vol. 25, No. 4, 2015, pp. 2274–2312. doi:10.1137/130922793, URL http://epubs.siam.org/doi/abs/10.1137/130922793.

[26] Chen, C., He, B., Ye, Y., and Yuan, X., "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Mathematical Programming*, Vol. 155, No. 1-2, 2016, pp. 57–79. doi:10.1007/s10107-014-0826-5, URL http://dx.doi.org/10.1007/s10107-014-0826-5.

[27] Wang, J. J., and Song, W., "An algorithm twisted from generalized ADMM for multi-block separable convex minimization models," *Journal of Computational and Applied Mathematics*, Vol. 309, 2017, pp. 342–358. doi:10.1016/j.cam.2016.02.001, URL http://dx.doi.org/10.1016/j.cam.2016.02.001.

[28] Deng, W., Lai, M. J., Peng, Z., and Yin, W., "Parallel Multi-Block ADMM with o(1 / k) Convergence," *Journal of Scientific Computing*, Vol. 71, No. 2, 2017, pp. 712–736. doi:10.1007/s10915-016-0318-2.

[29] Bertsekas, D. P., and Tsitsiklis, J. N. J., *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Belmont, Massachusetts, 1989. URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:parallel+and+distributed+computation:+numerical+methods{#}2.

[30] Goldstein, T., O'Donoghue, B., Setzer, S., and Baraniuk, R., "Fast Alternating Direction Optimization Methods," *SIAM Journal on Imaging Sciences*, Vol. 7, No. 3, 2014, pp. 1588–1623. doi:10.1137/120896219, URL http://epubs.siam.org/doi/10.1137/120896219.

[31] He, B. S., Yang, H., and Wang, S. L., "Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities," *Journal of Optimization Theory and Applications*, Vol. 106, No. 2, 2000, pp. 337–356. doi:10.1023/A:1004603514434, URL http://link.springer.com/10.1023/A:1004603514434.

[32] Wang, S. L., and Liao, L. Z., "Decomposition method with a variable parameter for a class of monotone variational inequality problems," *Journal of Optimization Theory and Applications*, Vol. 109, No. 2, 2001, pp. 415–429. doi:10.1023/A:1017522623963.

[33] Bey, J., "Simplicial grid refinement: on Freudenthal's algorithm and the optimal number of congruence classes," *Numerische Mathematik*, Vol. 85, No. 1, 2000, pp. 1–29. doi:0.1007/s002110050475.

3

Conclusions and Recommendations

This chapter presents the main conclusions of this thesis project in Section 3.1, reflecting on the project definition as set in the Preliminary Thesis in Part II. Section 3.2 provides recommendations for improvements of the algorithm and possible future research.

3.1. Conclusions

The objective was to develop a high fidelity aerodynamic model for the Innovative Control Effectors Aircraft with at least first order continuity throughout the domain, by taking a distributed system identification approach to construct a multivariate Simplotope B-Spline model.

Shortly it can be stated that this objective was met. A Simplotope B-Spline model with first order continuity was constructed using a distributed approach, but the method was different than thought in the Preliminary Thesis. Dual Ascent was initially chosen to be used as distributed solver, but in the end this did not enforce the continuity as desired. At that point, first a sequential version of the Alternating Direction Method of Multipliers (ADMM) was explored. Although having good convergence properties and constraint satisfaction, the sequential nature made it unable to be implemented in parallel, reducing the computation time and memory requirements of the estimation, but not the infinitely scalable method that was aimed at.

Therefore the Variable Splitting ADMM (VSADMM) method has been used. By decoupling the individual partitions and introducing a coupling coefficient to ensure global continuity, a fully parallel estimation routine was developed and successfully applied to Aerodynamic Model Identification (AMI). Where Dual Ascent required partitions of multiple simplotopes, the VSADMM routine ensures continuity even when every simplotope is a separate partition. All derivations and test results for the considered distributed approaches are given in Appendix B.

The first research question was *"How can a multivariate Simplotope B-Spline model of the ICE aircraft be constructed that can be used in modern model-in-the-loop controllers, and what are the advantages and disadvantages compared to the current simplex spline modelling approach?"*.

The Simplotope B-Spline model has been constructed using a distributed approach. This was possible for all dimensions of the submodel that appear in the ICE aerodynamic model. The memory requirements remained within limits because of the distributed approach, even on a computer with only 8GB RAM. The first order continuity has been enforced, which was more easily in terms of data requirements because of the use of simplotopes instead of simplices. However, the same dataset has been used as in [16], extend by using single imputation with cubic spline interpolated data points. To have a detailed model, correctly predicting the Simulink model as provided, using imputed datasets is inevitable.

The flexibility of the Simplotope B-Splines offered better adaptation to the datasets used for constructing the models, e.g. the submodels C_{*_6} used a $(2,1,1,1)$-simplotope model, with degrees $(3,1,1,3)$. The 1^{st} degree layers were possible because there was only a single simplex in those layers, and thus first order continuity was maintained.

In order achieve real time evaluation of all 108 Simplotope models, the existing evaluation function for the simplex models was extended. This evaluation function was written in C++, and the extension uses a new

class `cSimplotopeSpline`, consisting of several `cLayer` objects, that inherit the existing class `CBsplinen`. A detailed description is given in Appendix C.3.

"How does using Simplotope B-Splines affect the model estimation?" was the second research question.

The Simplotope B-Spline structure has several advantages. First it generally requires less coefficients to cover a similar domain with simplices. It gets rid of many edges and hence lowers the number of continuity constraints and possible discontinuities. Although globally the number of B-Coefficients are reduced, the amount of polynomial terms per simplotope is higher than in a simplex. This increases e.g. the size of the regression matrix, as a single data point is expressed in more polynomial terms.

The accuracy with the validation data is demonstrated in Appendix B.2, where also the distributed approach is assessed. The common trend for models of all dimensions is that the simplotope structure is better than the simplices at low noise intensities, in terms of validation residuals.

However, at high noise intensities the maximum residual can grow above 100% for simplotopes, where simplex splines are less affected by incresing noise levels. Also considering the better fit of simplotopes to the identification data, expressed in adjusted coefficient of determination, indicates at an overfitting regression. A possible explanation is that simplotope polynomials consist of high degree, cross-coupling terms and an incomplete polynomial. These terms will improve the fit to the identification data, but are unlikely to have a physical meaning and thus not have an improved effect on the validation result. This negative effect on the model quality with increasing noise levels should be considered when using Simplotope B-Splines in the future.

Because the ICE models were considered noiseless w.r.t. the original Simulink model, high dimensional simplotope layers were used for construction of the Simplotope B-Spline model. These showed the best validation results in terms of relative RMS and relative maximum residual, as comparisons in Appendix A present. It was also shown that the implemented Simplotope B-Splines are well able to follow the Simulink model, with relative RMS deviations between the two responses of at most 6.54%, which also includes inputs from outside the simplotope model domain that are extrapolated by clipping the boundary values.

The first order continuity that was enforced throughout the domain, resulted in visible smoother model responses. Where the zeroth order continuous simplex model showed ripples because of discontinuities, the newly developed model showed a smooth line following the Simulink model.

The last research question was *"How does the distributed system identification approach affect the model estimation?"*

The computational time can be highly reduced using the distributed approach. Because of the parallel properties of the VSADMM estimation routine, the computational complexity is only depending on the size of a single partition, and thus independent of refinement to the model's chosen tessellation. Refining the tessellation will result in more partitions, which will not increase the computational time if an equal amount of parallel pools is added. The communication between the partitions is limited to a vector of scalars, as big as there are smoothness conditions across the partitions' edges.

The basis polynomial of a simplotope spline has more terms than that of a simplex spline. Therefore, a simplotope has more B-Coefficients than a simplex spline in a single unit. Because the computational complexity is solely depending on unit size, the simplotopes require more computational effort than the simplices, provided that a fully parallel implementation is used. Using partitions of multiple simplotopes increases the computational effort, but showed better results in meeting the global continuity constraints.

The tests in Appendix B.2 showed that in almost all cases, the results of the distributed algorithm were similar to those of the globally estimated solution. Analyzing the residuals of the validation data, the distributed approach converges steadily to the global solution. In most cases the continuity constraints were well met, with some mismatch outliers of 10^{-4} that occurred at high noise intensities. The simplex splines showed overall better convergence towards the global solution.

Adaptive penalty factors ρ have been used for the VSADMM algorithm, to provide robustness against badly chosen values. All tests in Appendix B.2 use this adaptively updated ρ, but two different initial values are used: 1.0 and 0.01. The results of e.g. the 2- and 3-dimensional cases showed that the simplotopes benefit from a lower initial penalty factor, indicating that this still has an influence on the model estimation. Possible cause is, that compared to the simplex splines, the simplotopes have less continuity constraints and hence more free coefficients. Therefore they benefit more from loosened constraints and more freedom to optimize

the parameters. This suggests that the choice of penalty factor is still a choice that should be considered before using the estimation algorithm.

3.2. Recommendations

Based on the conclusions stated above, and the experience gained throughout the thesis project, some recommendations for future research or improvements of the estimation can be made.

Parallel Implementation Currently, VSADMM has only been implemented in a sequential fashion, as Matlab's `parfor` showed poor performance when it comes to parallel computation. However, the VSADMM routine is capable of a fully parallel implementation, which reduce the computation time significantly. An implementation on a GPU or on multiple cores are two of the possibilities that can be considered. Cloud computing services such as Microsoft Azure and Amazon EC3 can be used to run the program on multiple cores, without having the need to be in possession of these cores physically. Matlab offers possibilities with its built-in Parallel Computing Toolbox and the Distributed Computing Server.

Adaptive Implementation A logical next step would be to make the estimation algorithm adapt to new measurement data, in order to enhance the model quality, or make it fault tolerant in case of e.g. structural failures. It can be investigated how a recursive algorithm is able to adapt the simplotopes locally and makes these changes propagate towards its neighbors through the continuity conditions. A suggestion would be simply recomputing the regression matrix of a single partition, and start iterating again. However, a combination with Recursive Least Squares (RLS) would be more suitable, but the smoothness constraints add more complexity to this.

Internal Partition Constraints Taking a different approach towards internal constraints can be beneficial for the estimation of bigger sized partitions. This can either be done by using null-space projection for the internal constraints, which forces the algorithm to meet those at all times, or use a warm start by initializing the coefficients locally using a global method such as the iterative solver in [1].

Data Augmentation Currently, interpolated data values from the Simulink model were imputed in order to make a higher degree model possible. However, more advanced data augmentation methods exist and could be used, such as Multiple Imputation and Imputation-Posterior algorithm [9, 11, 15]. Moreover, prior knowledge of the model can be used in e.g. setting constraints on the derivatives, or transforming the B-Coefficients into Physical Coefficients, with an actual physical meaning, and using those to set restrictions on the model. These Physical Splines can also be used to make the model better behaving in noisy environments using prior knowledge to prevent overfitting of the model.

Full Rank Filter The global continuity matrix of the splines were not full rank. Generally, constrained optimization methods benefit from constraint matrices that are full rank, and so does ADMM [2]. Filtering the redundant constraints by adding them one by one required a considerable amount of time for big splines and was not regarded. If a better filtering method can be found, based on e.g. the simplex stars of the edges between the simplotopes, this might improve the convergence properties of ADMM and the quality of the solution.

Optimal Penalty Factor Currently, an adaptive penalty factor was used for robustness against badly chosen ones, but the initial choice still influenced the ADMM algorithm. The convergence rate can be improved considerably by choosing the exact right penalty factor [7]. Another possibility is to optimize the adaptive scheme such that it works better [8, 18].

II

Preliminary Thesis Report

Summary

The Innovative Control Effectors (ICE) aircraft is a high performance, tailless fighter which contains multiple novel control effectors. The control suite contains 13 control effectors, including a Spoiler Slot Deflector (SSD) and a All Moving Tip (AMT) on each wing, among others. This was required to ensure stable and controllable behavior in all areas of the flight envelope. Manually controlling this over-actuated, highly non-linear aircraft requires an advanced controller, that needs to be combined with a well designed control allocation algorithm. A key element for these algorithms is an accurate aerodynamic model to compute the aerodynamic coefficients and its derivatives.

An aerodynamic dataset of the ICE model is available, consisting of 108 submodels that are combined into one single model. Although an accurate Simplex B-Spline model has been developed, it was less accurate in approximating the control derivatives, and it required an extensive amount of time to be estimated. The primary objective of this thesis project is to develop a high fidelity aerodynamic model for the Innovative Control Effectors Aircraft with at least first order continuity through the domain, by taking a distributed system identification approach to construct a multivariate Simplotope B-Spline model.

Simplotope B-Splines are a more general version of the already used Simplex B-Spline. It has the property that it can decouple the dimensions, resulting in underlying structures that are better shaped to the data set. Furthermore it requires less coefficients and less continuity constraints to define model of better quality than Simplotope B-Splines. This is beneficial for the computation time and the fact that the dataset is relatively small.

The local nature of the Simplotopes and the sparse continuity matrix makes the model suitable for distributed optimization. The model's tessellation will be subdivided into smaller partitions, that are optimized locally. A dual ascent post-smoothing filter will be applied in order ensure the continuity between the partitions.

The functionality of the distributed Simplotope B-Spline model has been demonstrated and compared to Simplex B-Splines and global approaches. It showed that global Simplotope B-Splines have a higher approximation power than than Simplex B-Splines, while using 94% less time. Although using a distributed approach reduced the time by another 80%, the quality of the continuity constraints reduced. The cost function was optimized better locally, but on the partitions' edges some discontinuities in the model and its derivative were present. Solutions to solve this are to apply an overlap for a better local estimation of the parameters, or using a different distributed optimization method.

The demonstrator worked well for 2D and 3D cases, but needs to be generalized in order to work on all submodels of the ICE model. Furthermore, an efficient function to evaluate the model in real time needs to be written, as the current one is only optimized for Simplex B-Splines. With these tools the model can be validated and its quality assessed and compared to other methods. Analyzing the quality, it might be required to augment the dataset with new artificial data. With this additional data the degree can be increased to better approximate the ICE model.

Contents

List of Abbreviations

ADMM Alternating Direction Method of Multipliers.
AMT All Moving Tip.
ANN Artificial Neural Networks.
AOA Angle of Attack.

DLEF Differential Leading Edge Flap.

GLS Generalized Least Squares.

ICE Innovative Control Effectors.

LED Leading Edge Down.
LEU Leading Edge Up.

MCAR Missing Completely At Random.

OLS Ordinary Least Squares.

PF Pitch Flap.

RCS Radar Cross-Section.
RMS Root Mean Square Error.

SSD Spoiler Slot Deflector.

TED Trailing Edge Down.
TEU Trailing Edge Up.
TV Multi-Axis Thrust Vectoring.

WFR Wavefront Reconstruction.
WLS Weighted Least Squares.

List of Symbols

Roman letters

$\boldsymbol{B}$ Regression Matrix.
$\boldsymbol{b}$ Barycentric Coordinates.
C_{**} Submodel of the ICE model.
c_* B-Coefficients of a model.
d Degree of a model.
$\hat{d}$ Amount of coefficients in a Simplex.
D_u Directional Derivative in direction u.
H Continuity Matrix.
J Amount of Simplotopes in a Tessellation.
J Cost Function.
ℓ Number of Layers.
M Mach number.
N Number of Measurements.
n Dimension.
$\mathcal{N}$ Kernel of the Constraint Matrix.
R Number of Continuity Equations per Edge.
r, m Continuity Order.
Y Measurements.
y Lagrangian duals.
z Coupling Variable.

Greek letters

α Angle of Attack.
β Angle of Sideslip.
γ Multi-index for the Continuity Equations.
δ Control Deflector.
κ Multi-index for the Simplex B-Coefficients.
λ Lagrange Multipliers.
λ Multi-index for the Simplotope B-Coefficients.
σ^2 Variance.

1 Introduction

Driven by the desire of the United States' Government in the early 1990s to develop high performance fighters with low Radar Cross-Section, the Innovative Control Effectors project was initiated by Lockheed Martin. This project investigated the use of novel control effectors on tailless aircraft flying on the edge of its envelope. This resulted in a model of the Innovative Control Effectors (ICE) aircraft, which has not vertical tail, but contains a total of 13 control effectors to ensure stability and controllability in all parts of the flight envelope.

Being a highly non-linear, overeffected aircraft, controlling the ICE aircraft without an advanced controller is nearly impossible. This controller needs to be combined with a well designed control allocation algorithm, in order to optimally and efficiently divide the pilot's control stick input over the available control surfaces. Crucial of advanced control allocation algorithms is the information from an accurate aerodynamic model. This model is being used to estimate the control effectiveness of all the effectors, necessary for the allocation of the control input.

Currently an accurate multivariate Simplex B-spline model is available of the aerodynamics of the ICE aircraft. Although being accurate in estimating the values, the model showed discontinuities and inaccuracies in its first derivative, resulting in inaccurate control effectiveness matrices. In this thesis project a model will be developed with at least first order continuity between the subdomains of the aircraft. Challenges that will be faced in constructing such a model is the computation time, which will likely be extraordinary, and the fact that the current dataset might not be sufficient for estimating an accurate model, while having no possibility of increasing the measurement size.

This document is the Preliminary Thesis report, consisting of the findings of an extensive literature survey, combined with a chosen strategy, a plan to implement this strategy, and a demonstration to show the feasibility of the method. In Chapter 2 the ICE Aircraft is introduced more extensively, including a short history and overviews of the control effectors and the available data. In Chapter 3 the theory of aerodynamic model identification with B-splines is introduced, together with methods to solve the parameters efficiently, and possible approaches to enlarge the dataset without performing more measurements. The full project is defined in Chapter 4, providing research questions, the chosen methods and the suggested implementation, and demonstration of this method. The report is concluded in Chapter 5.

2

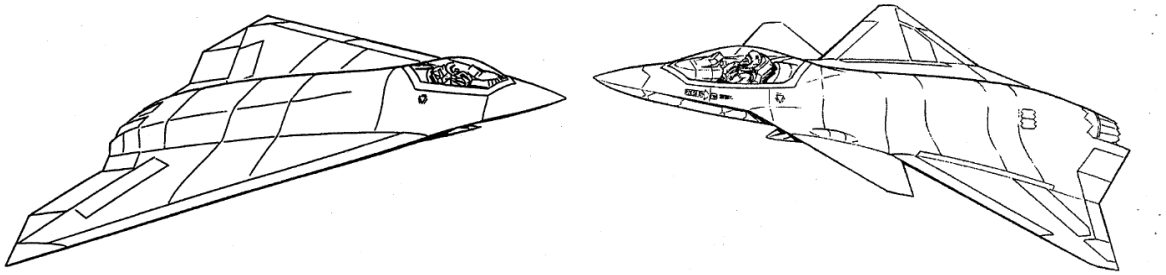
Innovative Control Effectors Aircraft

The Innovative Control Effectors (ICE) concept was introduced in the mid 1990s, as part of the focus of the US army on high performance fighter aircraft, with reduced visibility for enemy's radar systems. To lower this so-called Radar Cross-Section (RCS) many restrictions were placed on the design of an aircraft, e.g. the outer shape, placement of control surfaces and taking away any vertical control surfaces.

These restrictions pushed fighter aircraft into tailless designs. Because of the lack of vertical control surfaces, such as a rudder, directional control and stability introduce a major challenge. In order to meet state-of-the-art requirements for fighter aircraft new and innovative control suites had to be designed for tailless fighters. These novel control effectors introduced stability issues for stability and controllability, as especially the yaw control proved to be difficult.

For the purpose of designing a control suite for a high performance, tailless fighter, Phase 1 of the ICE program was initiated in 1993, followed by Phase 2 in 1996. This program was sponsored by Wright Laboratory (WL/FIGC) and the Naval Air Warfare Center (NAWCAD).

Two different baseline fighters were investigated: one land based aircraft for the US Air Force shown in Figure 2.1a, and a carrier based version for the US Navy in Figure 2.1b. The land based version is a delta wing with a 65 degree sweep angle. The carrier based version had 42 degree sweep delta wing, with a canard wing. Both versions only had one engine at the back and carry hidden weapons for low RCS purposes.



(a) The land based version.

(b) The carrier based version.

Figure 2.1: The baseline configurations studied during the ICE program [14]

During Phase 1 of the program, innovative methods to control and stabilize low RCS aircraft were investigated, while having no decrease in performance in any part of the flight envelope. Additionally, the potential of these control effectors for high Angle of Attack (AOA) flight capabilities were studied. Based on the baseline versions several innovative control suites were investigated by means of windtunnel tests, previous reearch, and control power simulation software. The best combinations that were found were worked out in more detail and studied more extensively during Phase 2 of the program.

The main goal of this phase was to extend the experimental information about the chosen control deflectors, e.g. the performance during high slip angle or angle of attack, and the the influence of the control

effectors on other closely located effectors. Drawn conclusions included that the All Moving Tips (AMTs) had the best, full envelope performance overall, and that the best position of the Spoiler Slot Deflectors (SSDs) were as outboard and forward as possible.

The version that is studied in this thesis is the land based version. The complete dataset that is provided by Lockheed Martin is focused on this version.

2.1. Control Effectors

Although during the two phases of the ICE project many types of control surfaces were studied, the version of the land based ICE aircraft that will be used in this project has a total of 13 control effectors of 6 different types. The ones that were chosen and studied are listed below.

All Moving Tips: The AMTs consist, as the name suggests, of the tips of the wing. These are not flaps that are able to move down, but the complete tip is able to rotate downwards. It was concluded after Phase 2 that the AMTs performed best overall as it comes to directional control in all parts of the flight envelope. By deflecting the AMTs Trailing Edge Down (TED) symmetrically, it could provide a nose down pitching moment at high AOA.

Differential Leading Edge Flaps: The Differential Leading Edge Flaps (DLEFs) are located at the leading edge of the wing. There are a total of four flaps, two inboard- and outboard flaps at each side of the wing. Although it did not outperform the other effectors in full envelope operation, they are useful in moderate tot high AOA. By using the flaps in a differential fashion, it can improve the controllability in in high AOA regions.

Elevons: The elevons are more conventional control effectors. They are located on the trailing edge of the wing, and on either side of the wing. When used symmetrically it can be used as an elevator, but when deflected asymmetrically it behaves as an aileron.

Spoiler Slot Deflectors: An SSD consists of multiple parts. First there is a conventional spoiler on the top of the wing. It is combined with a deflector at the same location, but at the lower side of the wing. This deflector is extended symmetrically with the spoiler. Between both is a hole in the wing that allows the air to flow through. Adding the deflector to a conventional spoiler resulted in higher effectiveness in high AOA and transonic regions, and beneficial hinge moments. However, the hole in the wing provides challenging structural integration, and when a SSD is deployed, it negatively influences the effectivenes of the control effectors on the trailing edge.

Pitch Flaps: There are two pitch flaps located on the trailing edge of the wing, inboard of the elevons. Although there are two, they can not move independently and are only deflected symmetrically in order to induce a pitch moment. Therefore the flaps are seen as one independent control deflector. The Pitch Flaps (PFs) can provide adequate pitch control throughout all AOAs.

Multi-Axis Thrust Vectorings: The single engine that is located at the center of the trailing edge, is able to provide vectored thrust in both pitch and yaw direction. Both directions are seen as independent directions.

In Table 2.1 an overview of all the different control effectors is given. It also provides the limits up to which they can be extended. Of all the effecters that are being used, the convention of what is considered to be a positive deflection is given. This can either be Trailing Edge Up (TEU), Trailing Edge Down (TED), or Leading Edge Down (LED)

The abbreviations as given in Table 2.1 are used to identify the right deflectors in the equations in Section 2.2. In the original model the dynamics of the actuation servos was not taken into account [21], but only a suggestion was given to model this as a first order delay. Based on these suggestions, the dynamics were implemented in order to have a more realistic model on which control systems can be tested. These dynamics are not relevant for this project, as these dynamics do not need to be approximated by using the windtunnel measurments.

Table 2.1: An overview of all the control effectors of the ICE aircraft, and their conventions, limits, and abbreviations [21].

Control Surface	**Convention Positive**	**Limits**	**Abbreviation**
Spoiler Slot Deflector	TEU	[0,60]	$LSSD$, $RSSD$
Differential Leading Edge Flaps	LED	[0,40]	$LIBLEF$, $RIBLEF$ (Inboard)
		[-40,40]	$LOBLEF$, $ROBLEF$ (Outboard)
All Moving Tips	TED	[0,60]	$LAMT$, $RAMT$
Elevons	TED	[-30,30]	LEL, REL
Pitch Flap	TED	[-30,30]	PF
Multi-Axis Thrust Vectoring	$\dot{p} > 0, \dot{q} > 0$	[-15,15]	TV

2.2. Aerodynamic Data and Model

Currently there is already an aerodynamic model created in `Simulink`, presented by Niestroy et al. [21]. The model is based on wind tunnel test data that has been generated during Phase 1 and Phase 2 of the ICE program.

The model has been divided into six main parts, corresponding to the six main aerodynamic coefficients of an aircraft. The forces are divided into an axial component C_A, a side component C_Y, and a normal component C_N. The moment coefficients are given as C_{LL}, C_M, and C_{LN}: the rolling, pitching, and yawing components respectively. Each of these six main coefficients are further subdivided into sixteen to eighteen subcoefficients, depending on the main coefficients.

Each of these subcoefficients represent either a contribution of a control deflector, a state of the aircraft, or a combination of both. The thrust vectoring is not included in the aerodynamic model as this is simulated in the propulsion model. Compared to the aerodynamic model described Niestroy et al. [21], the flexibility factors are not simulation in the model which will be used in the controller of the ICE aircraft, and thus is not a part of this project. How the force coefficients are built up is shown in Equations (2.1) to (2.3), with the δ_* being the control deflectors, with the abbreviations used in Table 2.1.

$$\begin{aligned} C_A = {} & C_{A_1}(\alpha, M) + C_{A_2}(\alpha, \beta, M) + C_{A_3}(\alpha, \beta, \delta_{LIBLEF}) + C_{A_4}(\alpha, \beta, \delta_{LIBLEF}, \delta_{OBLEF}, M) \\ & + C_{A_5}(\alpha, \delta_{LSSD}, \delta_{LEL}, M) + C_{A_6}(\alpha, \delta_{LSSD}, \delta_{RSSD}, \delta_{PF}, M) + C_{A_7}(\alpha, \beta, \delta_{LAMT}) + \\ & + C_{A_8}(\alpha, \delta_{LEL}, \delta_{LAMT}) + C_{A_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT}) + C_{A_{10}}(\alpha, \delta_{REL}, \delta_{RAMT}) \\ & + C_{A_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT}) + C_{A_{12}}(\alpha, \beta, \delta_{LSSD}) + C_{A_{13}}(\alpha, \beta, \delta_{RIBLEF}) \\ & + C_{A_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M) + C_{A_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M) + C_{A_{16}}(\alpha, \beta, \delta_{RAMT}) \\ & + C_{A_{17}}(\alpha, \beta, \delta_{RSSD}) \end{aligned} \tag{2.1}$$

$$\begin{aligned} C_Y = {} & C_{Y_1}(\alpha, M) + C_{Y_2}(\alpha, \beta, M) - C_{Y_3}(\alpha, \beta, \delta_{LIBLEF}) - C_{Y_4}(\alpha, \beta, \delta_{LIBLEF}, \delta_{OBLEF}, M) \\ & + C_{Y_5}(\alpha, \delta_{LSSD}, \delta_{LEL}, M) + C_{Y_6}(\alpha, \delta_{LSSD}, \delta_{RSSD}, \delta_{PF}, M) + C_{Y_7}(\alpha, \beta, \delta_{LAMT}) \\ & + C_{Y_8}(\alpha, \delta_{LEL}, \delta_{LAMT}) + C_{Y_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT}) - C_{Y_{10}}(\alpha, \delta_{REL}, \delta_{RAMT}) \\ & - C_{Y_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT}) + C_{Y_{12}}(\alpha, \beta, \delta_{LSSD}) + C_{Y_{13}}(\alpha, \beta, \delta_{RIBLEF}) \\ & + C_{Y_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M) - C_{Y_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M) - C_{Y_{16}}(\alpha, \beta, \delta_{RAMT}) \\ & - C_{Y_{17}}(\alpha, \beta, \delta_{RSSD}) \end{aligned} \tag{2.2}$$

$$\begin{aligned} C_N = {} & C_{N_1}(\alpha, M) + C_{N_2}(\alpha, \beta, M) + C_{N_3}(\alpha, \beta, \delta_{LIBLEF}) + C_{N_4}(\alpha, \beta, \delta_{LIBLEF}, \delta_{OBLEF}, M) \\ & + C_{N_5}(\alpha, \delta_{LSSD}, \delta_{LEL}, M) + C_{N_6}(\alpha, \delta_{LSSD}, \delta_{RSSD}, \delta_{PF}, M) + C_{N_7}(\alpha, \beta, \delta_{LAMT}) \\ & + C_{N_8}(\alpha, \delta_{LEL}, \delta_{LAMT}) + C_{N_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT}) + C_{N_{10}}(\alpha, \delta_{REL}, \delta_{RAMT}) \\ & + C_{N_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT}) + C_{N_{12}}(\alpha, \beta, \delta_{LSSD}) + C_{N_{13}}(\alpha, \beta, \delta_{RIBLEF}) \\ & + C_{N_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M) + C_{N_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M) + C_{N_{16}}(\alpha, \beta, \delta_{RAMT}) \\ & + C_{N_{17}}(\alpha, \beta, \delta_{RSSD}) + \frac{cq}{2V} C_{N_{18}}(\alpha, M) \end{aligned} \tag{2.3}$$

It can be seen that most of the coefficients are built up from coefficients that depend on the same aircraft state of control deflector, and that in fact all models can be estimated in a similar fashion. A small difference

occurs as the normal force components has an extra term which is being multiplied with $\frac{cq}{2V}$, based on the pitch rate.

The moment coefficients are approximated by the model structure in Equations (2.3), (2.4) and (2.6). These have a similar structure as the previous coefficients as well, with the difference that the non-symmetric coefficients have two based on the roll and yaw rate, while the pitch moment coefficient has a term based on the pitch rate.

$$\begin{aligned} C_{LL} &= C_{LL_1}(\alpha, M) + C_{LL_2}(\alpha, \beta, M) - C_{LL_3}(\alpha, \beta, \delta_{LIBLEF}) - C_{LL_4}(\alpha, \beta, \delta_{LIBLEF}, \delta_{LOBLEF}, M) \\ &+ C_{LL_5}(\alpha, \delta_{LSSD}, \delta_{LEL}, M) + C_{LL_7}(\alpha, \beta, \delta_{LAMT}) \\ &+ C_{LL_8}(\alpha, \delta_{LEL}, \delta_{LAMT}) + C_{LL_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT}) - C_{LL_{10}}(\alpha, \delta_{REL}, \delta_{RAMT}) \\ &- C_{LL_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT}) + C_{LL_{12}}(\alpha, \beta, \delta_{LSSD}) + C_{LL_{13}}(\alpha, \beta, \delta_{RIBLEF}) \\ &+ C_{LL_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M) - C_{LL_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M) - C_{LL_{16}}(\alpha, \beta, \delta_{RAMT}) \\ &- C_{LL_{17}}(\alpha, \beta, \delta_{RSSD}) + \frac{bp}{2V} C_{LL_{18}}(\alpha, M) + \frac{br}{2V} C_{LL_{19}}(\alpha, M) \end{aligned} \tag{2.4}$$

$$\begin{aligned} C_{M} &= C_{M_1}(\alpha, M) + C_{M_2}(\alpha, \beta, M) + C_{M_3}(\alpha, \beta, \delta_{LIBLEF}) + C_{M_4}(\alpha, \beta, \delta_{LIBLEF}, \delta_{LOBLEF}, M) \\ &+ C_{M_5}(\alpha, \delta_{LSSD}, \delta_{LEL}, M) + C_{M_6}(\alpha, \delta_{LSSD}, \delta_{RSSD}, \delta_{PF}, M) + C_{M_7}(\alpha, \beta, \delta_{LAMT}) \\ &+ C_{M_8}(\alpha, \delta_{LEL}, \delta_{LAMT}) + C_{M_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT}) + C_{M_{10}}(\alpha, \delta_{REL}, \delta_{RAMT}) \\ &+ C_{M_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT}) + C_{M_{12}}(\alpha, \beta, \delta_{LSSD}) + C_{M_{13}}(\alpha, \beta, \delta_{RIBLEF}) \\ &+ C_{M_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M) + C_{M_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M) + C_{M_{16}}(\alpha, \beta, \delta_{RAMT}) \\ &+ C_{M_{17}}(\alpha, \beta, \delta_{RSSD}) + \frac{cq}{2V} C_{M_{18}}(\alpha, M) \end{aligned} \tag{2.5}$$

$$\begin{aligned} C_{LN} &= C_{LN_1}(\alpha, M) + C_{LN_2}(\alpha, \beta, M) - C_{LN_3}(\alpha, \beta, \delta_{LIBLEF}) - C_{LN_4}(\alpha, \beta, \delta_{LIBLEF}, \delta_{LOBLEF}, M) \\ &+ C_{LN_5}(\alpha, \delta_{LSSD}, \delta_{LEL}, M) + C_{LN_7}(\alpha, \beta, \delta_{LAMT}) + C_{LN_8}(\alpha, \delta_{LEL}, \delta_{LAMT}) \\ &+ C_{LN_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT}) - C_{LN_{10}}(\alpha, \delta_{REL}, \delta_{RAMT}) - C_{LN_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT}) \\ &+ C_{LN_{12}}(\alpha, \beta, \delta_{LSSD}) + C_{LN_{13}}(\alpha, \beta, \delta_{RIBLEF}) + C_{LN_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M) \\ &- C_{LN_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M) - C_{LN_{16}}(\alpha, \beta, \delta_{RAMT}) - C_{LN_{17}}(\alpha, \beta, \delta_{RSSD}) \\ &+ \frac{bp}{2V} C_{LN_{18}}(\alpha, M) + \frac{br}{2V} C_{LN_{19}}(\alpha, M) \end{aligned} \tag{2.6}$$

Studying the above equations it can be noted that creating a model of these contributions will have to be done in many dimensions: the smallest model has two dimensions, while the biggest ones have five. The windtunnel test data consists of a grid into several dimensions. This measurement frequency in each of these dimensions differs, also between the models, which the model structure has to be carefully selected for each of the submodels.

In Table 2.2 is an overview given that shows all the submodels that are part of the ICE aircraft's aerodynamic model. It can be seen that there is a substantial difference between the several submodels. This means that for every submodel a different tessellation has to be designed, together with a different degree of model. However, it can be exploited that the main coefficients consists of the same submodels, only with different values. The structure of the dataset for these submodels can be chosen equal as well, which means a great deal of the structure definition, as explained in more detail in Section 4.2, has to be done only once for all similar submodels.

In the model that was constructed by Niestroy et al. [21], the values of the subcoefficients are estimated by interpolating the available measurements. For most models this has been done linearly, while some contain a cubic spline. As can be seen in the range of the different submodels, not every area of the flight envelope is reached by all submodels. In some submodels the range of e.g. the AOA reaches 90 degrees, while others do not go further than 45 or even 30 degrees. When the current state goes beyond the range of the submodel, it is evaluated by extrapolating the model in a linear fashion. Another option that has been applied is to clip the border value of the model. These options are suggested as well when constructing a multivariate B-Spline model as well.

Table 2.2: Overview of the submodels of the ICE aircraft and their parameters, data range, and number of datapoints

Coefficient	Parameters	Data Range	Number of Datapoints
C_{*_1}	α	[-5.0, 90.0]	48
	Mach	[0.3, 2.16]	10
C_{*_2}	α	[-5, 90]	30
	β	[-30, 30]	15
	Mach	[0.3, 2.16]	8
C_{*_3}	α	[-2.5, 45]	20
	β	[-10, 10]	3
	δ_{LIBLEF}	[0, 40]	2
C_{*_4}	α	[-2.5, 45]	20
	β	[-10, 10]	3
	δ_{LIBLEF}	[0, 40]	2
	δ_{LOBLEF}	[-40, 40]	5
	Mach	[0.3, 1.2]	4
C_{*_5}	α	[-2.5, 90]	28
	δ_{LSSD}	[0, 60]	4
	δ_{LEL}	[-30, 30]	5
	Mach	[0.3, 2.16]	7
C_{*_6}	α	[-2.5, 90]	29
	δ_{LSSD}	[0, 60]	2
	δ_{RSSD}	[0, 60]	2
	δ_{PF}	[-30, 30]	5
	Mach	[0.3, 2.16]	7
C_{*_7}	α	[-2.5, 90]	29
	β	[-30, 30]	7
	δ_{LAMT}	[0, 60]	5
C_{*_8}	α	[-2.5, 42.5]	19
	δ_{LEL}	[-30, 30]	3
	δ_{LAMT}	[0, 60]	3
C_{*_9}	α	[-2.5, 42.5]	19
	δ_{LOBLEF}	[0, 40]	3
	δ_{LAMT}	[0, 60]	3
$C_{*_{10}}$	α	[-2.5, 42.5]	19
	δ_{REL}	[-30, 30]	3
	δ_{RAMT}	[0, 60]	3
$C_{*_{11}}$	α	[-2.5, 42.5]	19
	δ_{ROBLEF}	[0, 40]	3
	δ_{RAMT}	[0, 60]	3
$C_{*_{12}}$	α	[-2.5, 90]	29
	β	[-30, 30]	7
	δ_{LSSD}	[0, 60]	4
$C_{*_{13}}$	α	[-2.5, 45]	20
	β	[-10, 10]	3
	δ_{RIBLEF}	[0, 40]	2
$C_{*_{14}}$	α	[-2.5, 45]	20
	β	[-10, 10]	3
	δ_{RIBLEF}	[0, 40]	2
	δ_{ROBLEF}	[-40, 40]	5
	Mach	[0.3, 1.2]	4
$C_{*_{15}}$	α	[-2.5, 90]	28
	δ_{RSSD}	[0, 60]	4
	δ_{REL}	[-30, 30]	5
	Mach	[0.3, 2.16]	7
$C_{*_{16}}$	α	[-2.5, 90]	29
	β	[-30, 30]	7
	δ_{RAMT}	[0, 60]	5
$C_{*_{17}}$	α	[-2.5, 90]	29
	β	[-30, 30]	7
	δ_{RSSD}	[0, 60]	4
$C_{*_{18,19}}$	α	[0, 30]	13
	Mach	[0.6, 2.2]	6

3

Aerodynamic Model Identification

In this chapter an introduction is given into multivariate splines. First a general introduction of common model identification methods are given together with their advantages and disadvantages in Section 3.1. After that, Simplex Splines are being introduced in Section 3.2, and being expanded into Simplotope Splines.

These Splines will be used in order to identify the ICE's aerodynamic model, by using least squares optimization. The set up is shown in Section 3.3, including the triangulation method, regression matrices, and parameter estimation methods.

Additionally to the current global methods to estimate the coefficients of the model, several distributed methods are proposed that can be used to solve the linear system more efficiently. Combining such a solver in combination with B-splines results in a distributed approach that has not been applied to aerodynamic model identification yet.

3.1. Model Identification Methods

Many methods are used to identify aerodynamic models. The most common one is the ordinary polynomial, which is linear in the parameters and can thus be solved using least squares [18]. Disadvantages of using this method is that it generally is to be estimated globally, which results in inaccurate and noise sensitive models. Since increasing the model degree involves adding the higher degree terms to the lower degree polynomial, the size of the regression matrix and the amount of parameters increases considerably, resulting in enormous computation times for relatively low quality models.

Variations on solving ordinary polynomials such as the weighted- or generalized least squares result in higher quality models but do not deal with the global matrices that have to be inverted.

Artificial Neural Networks (ANN) were proven to be able to provide accurate input-output mappings to identify systems in general [17]. This method simulates biological neurons, dividing them over several layers, namely one input layer, one or more hidden layers, and an output layer. These neurons are connected are given weights. All incoming weights are summed and a neuron's basis function is activated when its sum is above a bias value. The weights determine the output of the system.

Models until arbitrary accuracy can be constructed using ANN, which is why this method is not only used for system identification but for many machine learning practices. On the other hand the system to be solved is non linear and global, resulting in computationally complex optimization problems. Moreover, the models are usually a black box, and likely to overfit the identification data.

Tensor Product splines combines multiple univariate functions by taking the tensor product [15]. However, it is not able to fit scattered data, which is most common when measurements are used to identify an aerodynamic model [1]. Although the ICE model's dataset is gridded and tensor product splines can be used for it [5], a general method is preferred for when off-grid and generally scattered measurements are added for model identification.

Multivariate B-splines take away these problems to a great extent. The splines result in local regression matrices which will be easier to invert and thus result in lower computation times. The only global part is the

sparse constraint matrix. Because all the parameters are estimated locally, the model is more accurate for a relatively small amount of parameters [10].

3.2. Multivariate B-Splines

Multivariate B-Spline use Bernstein-Bezier curves to represent polynomials on triangular patches. First the theory of the Splines applied on Simplices is given in Sections 3.2.1 and 3.2.2, giving the used basis polynomials, and the continuity conditions in order to acquire smoothness up to a chosen order. The theory Simplex Splines is expanded in Section 3.2.3 to be used on a more general form, the Simplotope.

3.2.1. Basis Splines

Multivariate B-Splines use domains that are triangulated into smaller subdomains. The subdomains have a Simplex shape, which is the generalization of a triangle in any number dimensions. The n-dimensional Simplex has $n+1$ vertices, which results e.g. in a line when defined in 1D, or a tetrahedron in 3D.

The barycentric coordinates $\mathbf{b}$ of an data point are defined locally in the simplex. In this coordinate system all points x get a set of weights b which are multiplied with the vertices v of the particular simplex, as can be seen in Equation (3.1).

$$x = \sum_{i=0}^{n} \boldsymbol{b}_i v_i \tag{3.1}$$

Every $n-$dimensional data point has $n+1$ barycentric coordinates. When the chosen point is within the convex hull of the vertices, all barycentric coordinates will be non-negative. Any point outside this hull can be transformed into barycentric coordinates as well, but will have at least one negative coordinate.

It was proven by de Boor [7] that any polynomial $p(\boldsymbol{b})$ can be written as the weighted sum of local Bernstein polynomials, as can be seen in Equation (3.2). It uses the the Bernstein basis polynomials of dimenions d as defined in Equation (3.3). By setting the function of the basis function to zero when a measurements is not contained in the hull of the chosen simplex, the basis function is made local. The property that all barycentric coordinates add up to one, makes that the basis is also stable.

$$p(\boldsymbol{b}) = \sum_{|\kappa|=d} c_\kappa B_\kappa^d(\boldsymbol{b}) \tag{3.2}$$

$$B_\kappa^d(\boldsymbol{b}) = \frac{d!}{\kappa!} b^\kappa \tag{3.3}$$

The weights c_κ are the B-coefficients for this particular simplex and define the shape of the model. Each coefficients corresponds to a single permutation of the multi-index $\kappa = (\kappa_0, \kappa_1, \ldots, \kappa_n)$, provided that $\kappa_i \leq d$ and $|\kappa| = d$.

The spatial location of all the B-Coefficients can be visualized in a simplex by giving them barycentric coordinates of their own, as given in Equation (3.4). The visualization of this so-called B-net on a 2D Simplex is shown in Figure 3.1 for for a 2^{nd} (left) and 3^{rd} (right) degree model.

$$\boldsymbol{b}(c_\kappa) = \frac{\kappa}{d} \tag{3.4}$$

The vertices are ordered using the B-net orientation rule, which means that the vertices of a simplex are ordered on vertex index. The multi-indices c_κ are sorted in lexicographical order with the highest coefficient belonging to the vertex with the highest index. In Figure 3.1 this would mean that the bottom left vertex has the highest index, the top vertex the second highest, and the bottom right the lowest.

The total number of B-coefficients $\hat{d}$ for a polynomial of degree d on an n-dimensional Simplex is computed by Equation (3.5). This also means that the polynomial will consist of $\hat{d}$ basis functions.

$$\hat{d} = \frac{(d+n)!}{d!n!} \tag{3.5}$$

As this is the amount of coefficients for a single simplex, the total global B-coefficients vector has a length of $\hat{d} \cdot J$, with J being the amount of simplices in the complete triangulation. Note that many of the coefficient will

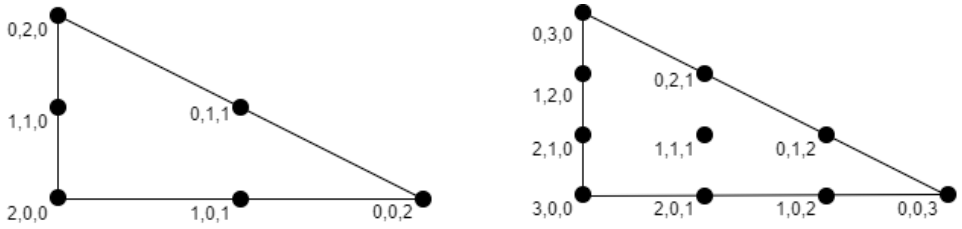


Figure 3.1: Visualization of the B-net on a 2D Simplex for a 2nd (left) and 3rd (right) degree model.

overlap, when they lay on the same edge but in different simplices. This means that when the coefficients are only computed locally, there will a discontinuous surface, as it is unlikely that the coefficients will be equal. To prevent discontinuities, smoothness constraints are introduced.

3.2.2. Continuity

As the Bernstein polynomials are local, continuity equations are introduced to ensure that the model has a single surface that is smooth up to a chosen order. For general orders of continuity $r < d$ between two simplices t_1 and t_2, the continuity equations can be formed by using Equation (3.6) [19]. The equations of all orders up to r are included, which means that 2nd order continuity also impliers there is 1st and 0th order continuity.

$$c^{t_1}_{\kappa_0,m,\kappa_1,\ldots,\kappa_n} = \sum_{|\gamma|=m} c^{t_2}_{(\kappa_0,0,\kappa_1,\ldots,\kappa_n)+\gamma} B^m_\gamma(v_*), \qquad 0 \le m \le r \tag{3.6}$$

For every permutation of the continuity multi-index $\gamma = [\gamma_0, \gamma_1, \ldots, \gamma_n]$, $0 \le \gamma_i \le m, |\gamma| = m$, a continuity equation is formed based on the out of edge vertex v_* of simplex t_2. The barycentric coordinates have to be computed based on the vertices of the simplex t_1. Additionally the condition $|\kappa| + |\gamma| = d$ has to hold for every condition.

The total amount of smoothness constraints per edge in an n-dimensional triangulation with C^r order continuity is given by Equation (3.7) [10].

$$R = \sum_{m=0}^{r} \frac{(d-m+n-1)!}{(d-m)!(n-1)!} \tag{3.7}$$

It can be seen in Equation (3.6) that the continuity are defined in terms of the B-coefficients of the simplices that share the edge. By moving all the coefficients to one side of the equation all the continuity conditions can be combined into one global smoothness matrix H, such that $H \cdot C = 0$ holds, with C all local coefficients combined in one vector. The matrix H has size $(R \cdot E) \times (\hat{d} \cdot J)$, with E the amount of edges, and J the amount of simplices.

3.2.3. Simplotope Splines

Simplotope B-Splines are a combination of Tensor Product Splines and Simplex B-Splines [30]. A Simplotope in n dimensions can be subdivided in ℓ layers, in which $1 \le \ell \le n$. Every layer will contain a subset of the dimensions, with the division indicated by the multi-index $\nu = (\nu_1, \ldots, \nu_\ell)$, in which $|\nu| = n$ and every layer i within a simplotope is a ν_i-dimensional simplex. The simplex in each layer will be triangulated individually and the tensor product of the individual models is taken, combining the layers in to one tessellation. Note that it is also possible to select a single layer, in which case a $\nu-$simplotope spline simply turns into a $n-$dimensional simplex spline.

The design choices for the model, such as the degree, triangulation, and the continuity order are defined per layer. This gives the opportunity to better adapt the underlying structure of the model to e.g. prior knowledge of the system that has to be identified, or the available dataset.

Since all the layers are smaller simplex splines, the theory explained in Section 3.2.1 applies. That means that any point within a simplotope is described by combining the barycentric coordinates $\boldsymbol{b}_i$ of each individual layer i. This results in the multi-index $\beta = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell)$.

In order to form basis polynomials in the simplotope the multi-indices κ of each layer is combined with the other layers, resulting in the multi-index $\lambda = (\lambda_1, \ldots, \lambda_\ell)$, in which λ_i has the same properties and restrictions as the multi-index κ of layer i. This results in the basis polynomials for simplotope splines as shown in Equations (3.8) and (3.9) [30].

$$\pi(\beta) = \sum_{|\lambda_i|=d_i, \forall i \in [1,\ell]} c_\lambda \boldsymbol{B}_\lambda(\beta) \tag{3.8}$$

$$\boldsymbol{B}_\lambda(\beta) = B^{d_1}_{\lambda_1}(\boldsymbol{b}_1) \cdot B^{d_2}_{\lambda_2}(\boldsymbol{b}_2) \cdot \ldots \cdot B^{d_\ell}_{\lambda_\ell}(\boldsymbol{b}_\ell) = \prod_{i=1}^{\ell} B^{d_i}_{\lambda_i}(\boldsymbol{b}_i) \tag{3.9}$$

In these equations, the degree vector $\boldsymbol{d} = (d_1, \ldots, d_\ell)$ indicates the chosen degree of an individual layer. The local B-coefficients c_λ that form the polynomial have similar properties as the the coefficients c_κ for simplices. Similarly, $B^{d_i}_{\lambda_i}$ is the basis polynomial of a simplex as described in Equation (3.3).

Applying the tensor product on the layers means that the B-net of the individual layers is copied over in the other layers. This results is that a total amount of B-coefficients that is the product of $\hat{d}$ of all layers.

Example 1. Take a $(2,1)$-simplotope spline with degrees $(2,3)$. The permutations of the multi-index for each layer are:

$$\boldsymbol{\lambda}_1 = \{(2,0,0), (1,1,0), (0,2,0), (0,1,1), (0,0,2)\}$$
$$\boldsymbol{\lambda}_2 = \{(3,0), (2,1), (1,2), (0,3)\}$$

The basis polynomials of the simplotope are found by taking the tensor product of these two simplex splines, resulting combining all the terms of both layers. Since $\hat{d}_1 = 6$, and $\hat{d}_2 = 4$, a total of 24 combinations of $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2)$, and thus coefficients in the simplotope, will result. Two possible combinations are $\boldsymbol{\lambda} = ((0,2,0),(2,1))$ and $\boldsymbol{\lambda} = ((0,1,1),(0,3))$. For the first combination, the basis polynomial of the simplotope can be found using Equation (3.9)

$$\begin{aligned}\boldsymbol{B}^{(2,3)}_{(0,2,0,2,1)}(\beta) &= B^2_{(0,2,0)}(\boldsymbol{b}_1) \cdot B^3_{(2,1)}(\boldsymbol{b}_2) \\ &= \frac{2!}{0!2!0!} b^0_{10} b^2_{11} b^0_{12} \cdot \frac{3!}{2!1!} b^2_{20} b^1_{21} = 3 b^2_{11} b^2_{20} b^1_{21}\end{aligned}$$

The expanded basis polynomial of the second combination is:

$$\begin{aligned}\boldsymbol{B}^{(2,3)}_{(0,1,1,0,3)}(\beta) &= B^2_{(0,1,1)}(\boldsymbol{b}_1) \cdot B^3_{(0,3)}(\boldsymbol{b}_2) \\ &= \frac{2!}{0!1!1!} b^0_{10} b^1_{11} b^1_{12} \cdot \frac{3!}{0!3!} b^0_{20} b^3_{21} = 2 b^1_{11} b^1_{12} b^3_{21}\end{aligned}$$

The value of the Simplotope Spline polynomial can be found by finding combination of individual layer's basis polynomials, multiplying them with their corresponding B-coefficient c_λ, and summing them as shown in Equation (3.8).

Continuity between simplotope splines are formed similarly as in simplex splines [31]. Adjacent simplotopes will always differ in only one layer, which will be called the out-of-edge layer. The continuity constraints can be found in two steps.

First the continuity equations in the out-of-edge-layer's triangulation is found by imposing the constraints as described in Equation (3.6). The second step is to copy these constraints in this layer for every copy of the B-net that is a result of the tensor product with the other layers. The total amount of constraints per edge is computed in Equation (3.10), with R_{ooe} and $\hat{d}_{ooe}$ the amount of constraints and coefficients in the out-of-edge layer respectively.

$$\boldsymbol{R} = R_{ooe} \frac{\prod_{i=1}^{\ell} \hat{d}_i}{\hat{d}_{ooe}} \tag{3.10}$$

Equivalently to continuity constraints in simplex splines, the ones for simplotope splines can be combined in one global smoothness matrix H, such that $H \cdot c = 0$, with c the global vector of all B-coefficients.

Example 2. Take a $(2,1,1)$-Simplotope of degrees $(3,3,4)$, with continuity order $(1,1,1)$. According to Equation (3.5), the amount of B-coefficients per layer $\hat{\boldsymbol{d}} = (10,4,5)$, resulting in a total of 200 coefficients to define this simplotope's polynomial. An edge in layer 1 will have 7 continuity conditions according to Equation (3.7), which is defined as R_{ooe}. Because of the tensor product between the individual layers, there are $4 \cdot 5 = \frac{200}{10} = 20$ copies of this layer's B-net, resulting in a total of $20 \cdot 7 = 140$ continuity conditions for an edge in this layer.

Accordingly, an edge in layer 2 will have $R_{ooe} = 2$, which is copied $\frac{200}{4} = 50$ times resulting in 100 conditions in this layer. At last, in the 3rd layer there are $\frac{200}{5} = 40$ copies of $R_{ooe} = 2$ conditions, resulting another 80 conditions.

Several opportunities are offered by simplotope splines compared to simplex splines. Since dimensions can be decoupled from each other by the use of layers, the structure can be better shaped to the dataset. When e.g. a dimension has a low amount of measurements, a lower degree polynomial or a more coarse triangulation can be chosen.

Because the shape of the simplotopes can be easier chosen that that of the simplices, a simplotope is easier to fill with measurements than a simplex [30]. This is beneficial when measurements are relatively scarce, as is the case with the ICE model. In general, the amount of coefficients and continuity conditions that are required to fully describe the model is lower in case of Simplotope Splines, as compared to Simplex Splines.

3.2.4. Other Properties

In this section various properties of the multivariate B-splines are given. First an explanation of how to compute directional derivatives of the model, and how they can be used in the ICE aerodynamic model. The other property is the sum of the different models, which might open an opportunity to combine the submodels of the ICE aircraft. Other advances in the Simplex Spline theory are described by De Visser [8], but the ones explained are considered the most relevant.

Directional Derivatives

It is possible to take the derivative of a polynomial in a chosen direction u, expressed in cartesian coordinates. As expected from polynomials, taking the m^{th} derivative of a polynomial with degree d, results in a polynomial of degree $d-m$. Although the this section describes taking the derivative of a Simplex spline, it works similarly in Simplotope splines. In that case the derivative is taken of the layer in which the direction is located. If the direction spans multiple layers, the vector is projected on the layers, after which the derivatives in individual layers are weighted and combined to the original direction.

The formula to compute the m^{th} in direction u is computed by Equation (3.11) [11]. In here a is the directional in local barycentric coordinates. It is computed by $a = b(w) - b(v)$, provided that $u = w - v$ in cartesian coordinates and both w and v are located in the convex hull of the simplex in which the derivative is taken.

$$D_u^m p(\boldsymbol{b}) = \frac{d!}{(d-m)!} B^{d-m}(\boldsymbol{b}) P^{d,d-m}(a) \cdot c \tag{3.11}$$

The de Casteljau iterations in matrix form $P^{d,d-m}(a)$ has sizes $\hat{d}^* \times \hat{d}$, with the dimensions being the amount of parameters for the lower degree polynomial and the original polynomial. The matrix can be formed by introducing two new multi-indices: θ, given $|\theta| = d$, and κ, given $|\kappa| = d-m$. Let $\boldsymbol{\Theta} = (\theta_1, \ldots, \theta_{\hat{d}}$ be all possible permutations for θ, and $\boldsymbol{K} = (\kappa_1, \ldots, \kappa_{\hat{d}^*})$, the de Casteljau matrix is defined as in Equation (3.12). For both $\boldsymbol{\Theta}$ and $\boldsymbol{K}$ the permutations have to be ordered in lexicographic order.

$$P^{d,d-m}(a) = \begin{bmatrix} P^m_{\theta_1-\kappa_1}(a) & P^m_{\theta_2-\kappa_1}(a) & \ldots & P^m_{\theta_{\hat{d}}-\kappa_1}(a) \\ P^m_{\theta_1-\kappa_2}(a) & \ddots & \ldots & P^m_{\theta_{\hat{d}}-\kappa_2}(a) \\ \vdots & \vdots & \ddots & \vdots \\ P^m_{\theta_1-\kappa_{\hat{d}^*}}(a) & P^m_{\theta_1-\kappa_{\hat{d}^*}}(a) & \ldots & P^m_{\theta_{\hat{d}}-\kappa_{\hat{d}^*}}(a) \end{bmatrix} \tag{3.12}$$

The basis functions $P^m_\gamma(b)$ are defined as in Equation (3.13). When γ contains any negative numbers, the basis polynomial is defined to be zero. That means that in the de Casteljau matrix zeros will appear whenever the a term in the difference $\theta - \kappa$ is negative.

$$P_\gamma^m(b) = \frac{m!}{\gamma!} b_\gamma^m \tag{3.13}$$

By using directional derivatives the quality of the model can be improved. When the directional derivative at a certain point in the domain is known, e.g. a control derivative, this information can be used and the model can be forced to have that value. There are two possibilities to achieve this. The first one is adding this derivative to the regression matrix, which is done with e.g. the Wavefront Reconstruction problem [9, 12]. Another possibility would be to add this as a constraint to the global constraint matrix H.

Directional derivatives are also being used in order to extrapolate the model beyond the measurement data. Splines tend to diverge beyond these boundaries and setting the second derivative to zero at the domain bounds, will results in better conditioned models [11]. Using these extra conditions at other locations except the bounds, could locally improve the model condition.

For control allocation methods it is desired to have accurate estimation of the control derivatives. These are required in order to construct the control effectiveness matrix. This one is necessary in modern control allocation methods [6, 26, 27].

Sums of Polynomials

Because of the fact that the Bernstein polynomials are linear in the parameter, the sum of two polynomials of equal degree will be a polynomial of that degree, with the B-coefficients being the sum of the corresponding B-coefficients of the original two polynomials. The sum of two polynomials $p(\boldsymbol{b})$ and $q(\boldsymbol{b})$ of equal degree d is summarized in Equation (3.14) [8].

$$p(\boldsymbol{b}) + q(\boldsymbol{b}) = \sum_{|\kappa|=d} (c_\kappa + d_\kappa) B_\kappa^d(\boldsymbol{b}) \tag{3.14}$$

The sum of two polynomials could be used in order to create one big aerodynamic model from all the small submodels that are given in the dataset. With simplex splines this is not possible, as all the dimensions are coupled and not all control effectors appear in all the models. However, as the dimensions in the simplotope splines are able to be decoupled, it might be a possibility to combine all the submodels into one model. This could possibly make the evaluation of the model easier and thus faster, which will make it more suitable for real time usage.

3.3. Parameter Estimation

Having the theory established, multivariate splines can be used in order to estimate an aerodynamic model given measured data points. The estimation problem for both the simplex and simplotope splines are similar. After the structure of the model has been determined in Section 3.3.1, the regression problem is set up in Section 3.3.2. Several global approaches to solve the parameters are explained in Section 3.3.3, while distributed methods are given in Section 3.3.4. Several other properties of the B-Splines which can be used to enhance the model are explained in Section 3.2.4. At last, to solve the problem of the limited dataset, methods to augment the amount of measurements are discussed in Section 3.4.

3.3.1. Triangulation

Before a model can be constructed the underlying structure has to be defined by triangulating the domain. In case of a simplex spline it would result in a single triangulation on the full domain in all available dimensions. When using simplotope splines this works similarly, but the domain is triangulated per layer. This means that there are a total of ℓ triangulations to be made of lower dimension than the full model. These triangulations are later combined into one tessellation as explained in Section 3.2.3.

There are multiple methods that can be used to triangulate the chosen domain. Normal Type I/II triangulations are most common, but it was not guaranteed to high quality triangulations in higher dimension.

Kuhn's triangulation is the better solution. The algorithm is easy to implement and guarantees conforming simplices. Bey [3] indicated that "it is often desirable that the number of congruence classes is not only finite but even as small as possible" for triangulations. Kuhn's algorithm is a good option to keep the amount of classes as low as possible: in 3D case it is even as low as 1.

Kuhn's triangulation in a selected hypercube can be constructed by finding S_n containing all the permutations of the set $(1,2,\ldots,n)$. Then, for every permutation $\pi \in S_n$ the triangulation of a hypercube is formed by initializing the first vertex with Equation (3.15), $x_\pi^{(0)}$ being a $n \times 1$ vector. Every next vertex until the last one using Equation (3.16), with $e^1,\ldots,e^n$ being the unit vectors in every dimension. The last vertex for each simplex is $(1,1,\ldots,1)$ and is common for all the simplices in the hypercube.

$$x_\pi^{(0)} = (0,0,\ldots,0)^T \tag{3.15}$$

$$x_\pi^{(j)} = x_\pi^{(j-1)} + e^{\pi(j)}, \quad 1 \le j \le n \tag{3.16}$$

Every n-dimensional hypercube will result in $n!$ Simplices when triangulated using Kuhn's triangulation. When Simplotope B-Splines are used in any number of dimensions, placing every dimension in its own layer will reduce the hypercube to a single simplotope. Kuhn's triangulation will only be used in 1D, which will give only a single simplex between two points.

3.3.2. Regression Problem

In order to estimate the B-coefficients of every simplotope, a cost function is set up to be minimized. First the regression matrices of the simplotopes have to be defined locally. This is $\boldsymbol{B_i}$ in Equation (3.17), which is a $N_i \times \hat{\boldsymbol{d}}$ matrix. It has a column for all the basis functions $\boldsymbol{B}_\lambda(\beta)$ of the chosen degrees, and a row for all datapoints N_i that are in the convex hull of the Simplotope. The local B-Coefficients c_λ^i are a $\hat{\boldsymbol{d}} \times 1$ columns vector.

$$Y = \boldsymbol{B_i} c_\lambda^i + \epsilon \tag{3.17}$$

All the regression matrices are combined as in Equation (3.18). As it can be seen, the total regression matrix is block diagonal with the local regression matrices on the diagonal.

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B_1} & 0 & \ldots & 0 \\ 0 & \boldsymbol{B_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & \boldsymbol{B_n} \end{bmatrix} \tag{3.18}$$

The most commonly used cost function in order to estimate the coefficients is a quadratic one as in Equation (3.19). This gives the possibility to solve the problem by an Ordinary Least Squares (OLS). If more information is available about the measurement data, it will be possible to extend the given OLS to a more advanced routine, e.g. Weighted Least Squares (WLS) or Generalized Least Squares (GLS).

$$J = \frac{1}{2}(Y - Bc)^T (Y - Bc), \quad s.t.\ H \cdot c = 0 \tag{3.19}$$

The minimization of the cost function is subject to the constraints as combined into the single, global matrix H. In general these will only consists of constraints to provide continuity between the different simplotopes in the domain. However, it can be extended with other sorts of constraints, of which examples are described in Section 3.2.4.

The matrix H will be a sparse matrix as equations will only exist between two simplotopes, resulting in the entries for all other simplotopes to be zero. Solving the minimization problem can be done using several methods, of which descriptions are given in Section 3.3.3. By exploiting the sparsity of the constraint matrix, distributed solvers might be used for this purpose as well. This can result in reduced computation times. Distributed solvers are explained in Section 3.3.4.

3.3.3. Global Approaches

The combination of the global cost function and its limitation to ensure smoothness is a equality constrained least squares problem. In literature several global methods are being used in order to solve this. Three methods that have been applied on Simplex B-Splines are explained in this section: Lagrange Multipliers, Kernel Based Method, and an iterative solver.

Lagrange Multipliers
The cost function can be extended into a Lagrangian $L(c,\lambda)$ in Equation (3.20). Here the Lagrange multipliers λ are used to ensure the constraints in H to be met.

$$L(c,\lambda) = \frac{1}{2}(Y - Bc)^T (Y - Bc) + \lambda^T Hc \tag{3.20}$$

By taking the derivative of the Lagrangian w.r.t. to both the coefficients and the Lagrangian multipliers, setting them to zero and rearranging the equations, the system in Equation (3.21) comes out [10].

$$\begin{bmatrix} \hat{c} \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} B^T B & H^T \\ H & 0 \end{bmatrix}^{-1} \begin{bmatrix} B^T Y \\ 0 \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} \begin{bmatrix} B^T Y \\ 0 \end{bmatrix} \tag{3.21}$$

The estimated parameters can now be found by $\hat{c} = C_1 \cdot B^T Y$. The necessary inverse only exists if the constraints matrix H is full rank. However, this will not always be true since duplicates of constraints may exist. This can either be solved by applying a filter on the $H-$matrix to make it full rank, or by using the Moore-Penrose pseudo inverse instead.

Although this is the simplest method in order to find the optimal solution, it is not preferred since it scales badly with the model size. This means that for models with a high amount of coefficients, the matrix inversion will take a considerable amount of time. This makes it unsuitable for e.g. recursive identification, but also for offline identification of large models.

Kernel Based Method
Another method of solving the constrained problem is by projecting the regression matrix on the null space of the constraint matrix [23, 24]. The $\mathcal{N} = \text{null}(H)$ space can be found by computing the singular value decomposition of the H-matrix. Generally the projection part is the most computationally complex step in the regression routine, although efficient algorithms exists [16]. This makes this method especially useful for recursive estimation, as $\mathcal{N}$ is invariant it only needs to be computed once at initiation of the estimation.

When the coefficients vector c is replaced $\mathcal{N} \cdot \tilde{c}$, the cost function from Equation (3.19) turns into Equation (3.22), which is an unconstrained problem. By defining $\tilde{B} = B\mathcal{N}$, the unconstrained coefficients $\tilde{c}$ can be solved using OLS. Afterwards, these unconstrained coefficients are multiplied with the null space to find the constrained coefficients.

$$J = \frac{1}{2}(Y - B \cdot \mathcal{N} \cdot c)^T (Y - B \cdot \mathcal{N} \cdot c) \tag{3.22}$$

Given R^* to be the amount of non redundant constraints, the computational complexity of solving the parameters by using the kernel based method is $O\left(\left(J \cdot \hat{d} - R^*\right)^2\right)$ [27]. However, this does not include finding the null space of the constraint matrix $\mathcal{N}$, which is generally the most expensive part. This makes the Kernel Based method preferred for recursive identification, as the latter part only has to be preformed at initialization of the algorithm. In general, the combination of both will still be faster than using Lagriange Multipliers, even for offline identification.

Iterative Method
To solve for the B-coefficients of the tessellation, Awanou et al. [2] proposed an iterative algorithm, and implemented this approach successfully in `Matlab` for trivariate B-splines of any degree d and any continuity $r < d$ between the splines. The iterative scheme was derived from the Lagrangian system as discussed earlier. However, where the matrix in the system required to be non-singular, the iterative scheme only required the regression matrix B to be positive definite on the kernel of the constraint matrix H. The full scheme consists of two steps, which are described in Equations (3.23a) and (3.23b), for $\epsilon > 0$.

$$\boldsymbol{c}^{(1)} = \left(B^T B + \frac{1}{\epsilon} H^T H\right)^{-1} \left(B^T Y - H^T \lambda^{(0)}\right) \tag{3.23a}$$

$$\boldsymbol{c}^{(k+1)} = \left(B^T B + \frac{1}{\epsilon} H^T H\right)^{-1} \left(B^T B \boldsymbol{c}^{(k)}\right) \tag{3.23b}$$

The iterative scheme slightly differs when there are additional constraints except for the smoothness constraints. An additional term is added at the left side which depends on ϵ and these constraints. It is proven that $\boldsymbol{c}^{(k)}$ converges to the estimate of the B-coefficients $\boldsymbol{c}$.

This scheme converges quickly to a value of $\boldsymbol{c}$, which is especially useful for offline application. However, the kernel based method is preferred for recursive identification, as the methods required for this are already derived in literature [11, 23].

3.3.4. Distributed Approaches

Besides centralized optimization methods, properties of Simplotope B-Splines such as the sparsity of the global constraint matrix and the block diagonal regression matrix, raise the opportunity for distributed approaches. With models increasing in size in terms of dimensions, data, and constraints, global solvers will results in large complexities and thus large computation times.

A distributed approach is desirable in order to solve the coefficient estimation problem. This will divide the problem into several smaller problems, which can be solved faster. B-splines have an excellent structure to be used in combination with distributed methods. The cost functions and thus regression matrices are defined per simplotope, which means that they are local and not depending on its neighbors. The constraint matrix H makes the problem still a global one, but since it is highly sparse, because a simplotope only has continuity conditions with its direct neighbors, it is still suitable for distributed solving. However, as it is still global, the distributed methods will not be perfect and probably have a lower quality in terms of meeting the smoothness constraints. A considerable reduction of computation time might make this reduction in quality acceptable.

Distributed approaches have been proven to be useful for 2-dimensional system identification purposes as shown by e.g. De Visser et al. [12], Silva et al. [22]. However, a distributed approach for aerodynamic model identification in higher dimensional space is new, and has possibilities of infinite scaleability becasue of the parallel iterative schemes.

Several distributed methods are investigated, with Dual Ascent and Alternating Direction Method of Multipliers being the most promising methods [4].

Dual Ascent

By finding the Lagrangian as in Equation (3.20), the dual of this Lagrangian can be optimized by using a gradient method. When the problem is separable, the Dual Decomposition method can be used, which works similar but tries to optimize partitions of the domain separately. In the case of both simplex and simplotope splines this is the case, as each spline can be seen as an individual problem for which the cost function has to be solved. This would result in several smaller optimization problems, with cost functions for every partition i as in Equation (3.24).

Every partition would exists of several simplotopes. In order to have a perfect solution the constraint matrix has to be fully separable as well, which is not true. Initially, an option is to only include local constraints in the H_i-matrix, and applying a post-smoothing filter in order to link the partitions together.

$$J = \frac{1}{2}\left(Y - \boldsymbol{B_i}\, c_i\right)^T \left(Y - \boldsymbol{B_i}\, c_i\right), \quad s.t.\ H_i \cdot c_i = 0 \tag{3.24}$$

As there is no exchange in information between the partitions in this case, it could be solved in parallel and be distributed between several cores of a computer. This local estimation can be an initial step into the post-smoothing filter.

The dual ascent method has been successfully implemented in an optical Wavefront Reconstruction (WFR) problem by De Visser et al. [12], in which they developed the D-SABRE method. In here the post smoothing filter that was applied is summarized in Equations (3.25a) and (3.25b).

$$c_i(l+1) = \hat{c}_i^{loc} + R_{N_i} H_i^T y_i \tag{3.25a}$$

$$y_i(l+1) = y_i(l) + \alpha(l) H_{i,m} c_{i,m}(l+1) \tag{3.25b}$$

The locally estimated coefficients $\hat{c}_i^{loc}$ are computed separately and remain constant. Two different constraint matrices are used: H_i with all constraints within the partition, and $H_{i,m}$ containing all constraints between

partition i and its neighbors. The first step is repeated for all partitions i that are defined. R_{N_i} is defined as $\mathcal{N}_i(\mathcal{N}_i^T A_i^T A_i \mathcal{N}_i)^{-1}\mathcal{N}_i^T$. It uses $\mathcal{N}_i = \text{null}(H_i)$ in order to solve the local coefficients first, and to be usedi in the post-smoothing filter.

The second step is a centralized operation, as it computes the global dual vector. The vector $\alpha(l)$ is dependent on the iteration, but keeping it constant at 0.5 led to fast convergence by the WFR problem.

To speed up the post-smoothing, and to improve the model's quality, an overlap in the partitions could be used. By extending the partition during its initial estimation, the constraints with the partition's neighbors are already partly met, reducing the amount of iterations needed during the post-smoothing phase. On the downside, this will make the initial estimation computationally more demanding.

Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) works similarly to the Dual Ascent method, but by using the augmented Lagrangian instead it shows better convergence rates [4, 13].

The usability of ADMM in system identification has been used for WFR by Silva et al. [22]. The problem is solved similarly to the Dual Ascent method, but in order to couple the partitions as chosen in the domain with its neighbors, they introduce a coupling variable z_{ij} as in Equations (3.26a) and (3.26b).

$$H_i^j c_i = z_{ij} \tag{3.26a}$$

$$H_j^i c_j = -z_{ij} \tag{3.26b}$$

This coupling exists for each edge (i, j) in the tessellation. The submatrix H_i^j contains the part of the global matrix that concerns the B-coefficients of partitions i, and the continuity constraints between partition i and j and is multiplied with the coefficients of partition i. Note that this means that some rows will result in zero when multiplied with only the coefficients of i, as parts of it are coefficients of other matrices, which is the reason why a coupling variable is introduced. The coupling variables are used as the constraints of the optimization problem and requires two vectors of duals.

The coupling update is shown in Equation (3.27), which is one of the steps of the full algorithms presented by Silva et al. [22]. The coupling update is half the difference of the two sides of a single continuity equation, which should result in converging values.

$$z_{ij}(k+1) = \frac{1}{2}\left(H_i^j c_i - H_j^i c_j\right) \tag{3.27}$$

The algorithm for solving the algorithm is more extensive compared to the Dual Ascent, which is a disadvantage. An advantage of this method, opposed to the two dimensional problem for the WFR, is that "the framework presented is able to address more general multivariate B-splines" [22]. This has yet to be proven for the D-SABRE method.

3.4. Data Augmentation

While studying the data set that is provided for identifying the aerodynamic model, it could be noted that the amount of data in several dimensions, especially control effectors, is scarce [28]. This could possibly be a problem with estimating the coefficients of the splines, as a minimum amount of measurement points is required in order to make estimation algorithms possible.

Since performing more wind tunnel tests is not possible, using data augmentation method could be beneficial. The augmented data could be used to make the system's equations solvable. However, this data will always be guessed and based on the currently available measurements, not based on physical measurements, and thus fake in some sort. Methods to add the additional data as parameters of the optimization routine will not be beneficial, as adding another parameter means that the required amount of measurements is increased as well.

Possibilities to treat the desired datapoints as missing values of the windtunnel test, and to use statistical methods to estimate these missing values, while using them to solve the optimization problem. Statistical analysis with missing data is studied extensively by Little and Rubin [20], which is focusing on non engineering subjects, although these statistical methods are generally applicable. It has been determined that the values are Missing Completely At Random (MCAR), as they are missing by design of the test.

The most promising methods were Imputation, Multiple Imputation, and the Imputation-Posterior Algorithm.

3.4.1. Imputation

Imputation is the method by data points are added on the spots where no data is observed during tests and thus data is considered missing. Several options are available in order to estimate the values of these imputed data points [20]:

- *Mean Imputation:* A choice can be made to impute the mean of the other measurements. As the mean of all measurements will be far away from the chosen location, a mean of surrounding or close observations can be taken in order to be imputed in the data set.
- *Regression Imputation:* In this case the imputed value is based on a regression model or a interpolation that is obtained by using the observed data. This could be for example a linear model, a cubic spline, or another lower degree regression model.
- *Stochastic Regression Imputation:* A value is obtained similarly to Regression Imputation, but a residual is added in order to indicate the uncertainty in this data point.

Although imputing a single value is the easiest and most convenient method, an obvious disadvantage of imputation is that imputed, unknown values, are treated as known. This will result in biased estimates of the parameters if not explicitly accounted for.

A distinction can be made between the original measurements and the imputed values. One option is to use WLS by giving the actual dataset a higher weight than the imputed values, and thus considering them more important. Commonly these weights are based on the variance of that particular measurement, but as this is not known, these weights have to be estimated. As the weighting matrix works only in a relative fashion, the only the relative difference of the weights of the actual measurements and the imputed values are of importance.

An option van der Peijl [28] used was to add the values of the original dataset as constraints to the constraint matrix, forcing the estimation to find a solution that goes exactly through the measurements. Comparing this option to the weighting matrix, is this similar to a giving the original dataset an infinite weight.

3.4.2. Multiple Imputation

As pointed out above, single imputation can result in biases, which multiple imputation can account for. As the name suggests, with multiple imputation a total of $D \geq 2$ values are imputed for at each missing data point, and consequently the parameters are estimated D times as well. This allows to assess the variability due to the imputed values and gives a better indication of how this influenced the regression.

For every set of data $d = 1, \ldots, D$, including imputed values, the parameters are estimated, $\hat{\theta}_d$, and the corresponding covariance matrix W_d. A combined estimate of the parameters $\bar{\theta}_D$ is found in Equation (3.28), and the average within imputation variance W_D in Equation (3.29).

$$\bar{\theta}_D = \frac{1}{D} \sum_{d=1}^{D} \hat{\theta}_d \tag{3.28}$$

$$\bar{W}_D = \frac{1}{D} \sum_{d=1}^{D} W_d \tag{3.29}$$

Additionally to the average of the individual analyses, the variability of the parameters between the different imputations can be computed with Equation (3.30). This in-between variability is adjusted with $\frac{D+1}{D}$ and added to the average covariance of the imputations $\bar{W}_D$ to receive the total variability as shown in Equation (3.31). As the number of imputations D increase, the effect of B_D decreases.

$$B_D = \frac{1}{D-1} \sum_{d=1}^{D} \left(\hat{\theta}_d - \bar{\theta}_D\right)^T \left(\hat{\theta}_d - \bar{\theta}_D\right) \tag{3.30}$$

$$T_D = \bar{W}_D + \frac{D+1}{D} B_D \tag{3.31}$$

By adding these measures the effect of the imputed values is integrated out by taking the average. A extra component of uncertainty is added to the covariances in order to represent the statistical measures more realistically than with single imputed values.

A clear disadvantage of this method is that it requires more computations, as the whole process is repeated D times in order to get the parameters with the different imputed values. This could make multiple imputation not suitable as identifying the aerodynamic model of the ICE aircraft already required a significant amount of time.

3.4.3. Imputation-Posterior Algorithm

An iterative solution for an even better estimate is the Imputation-Posterior (IP) algorithm and is inspired by the Expectation-Maximization (EM) algorithm [25]. As the name suggests, it contains of two steps that are performed in an iterative manner: first a value is imputed based on an estimation of the model, after which the posterior distribution of the parameters is approximated.

The imputation step can be either a single imputation or multiple imputed values. The values for these imputations will be drawn from a distribution with mean $X\theta^{t-1}$ and variance Σ^{t-1}. For the first iteration values for the missing data have to be guessed.

The posterior step actually consists of two parts. First the parameters $\hat{\theta}^t$ in iteration t are estimated by using a generalized least squares formula in Equation (3.32). The covariance matrix is the first inverse as is normal in a least squares estimation. In case a multiple imputed approach is taken this would be the step for one of the data sets $d = 1,\ldots,D$ and the steps as described in Section 3.4.2 should be performed in order to get the averaged values.

$$\hat{\theta}^{t+1} = \left(X^T \left(\Sigma^t\right)^{-1} X\right)^{-1} \left(X^T \left(\Sigma^t\right)^{-1} Y^t\right) \tag{3.32}$$

The second posterior step is to compute the variance of the residuals in order to be used in weighting the samples. This is done with Equation (3.33).

$$\Sigma^{t+1} = \frac{1}{n}\left(Y^t - X\theta^{t+1}\right)\left(Y^t - X\theta^{t+1}\right)^T \tag{3.33}$$

This solution will converge to a solution that is only based on the information in the observed values, while the imputed data is only used in order to make the analysis easier [20, 25]. Disadvantage of this method is that it requires an iterative method in order to solve it. As identifying the ICE model is already computationally complex to analyze once, it will be worse when it has to be identified multiple times.

Solutions to this can be to do only a couple of iterations, which will not increase the computation time too much. Another option is to run the algorithm locally in the simplotopes or within partitions of the full domain. This will still result in a biased estimation of the parameters, but will be preferred over single imputation as the imputed values will at least make sense locally.

4

Project Definition

In this chapter a definition of the project will be given. This includes a detailed definition of the research objective and research questions in Section 4.1, in order to have a good understanding of what goals and objectives will be pursued during this project. In Section 4.2 a plan will be given that is able, based on literature, to full fill these goals and to answer the research questions.

4.1. Research Objective and Questions

At the start of this thesis there is already an aerodynamic model available for the ICE aircraft. This was the results of the thesis finished by Ivo van der Peijl [28, 29]. The goal here was to construct an aerodynamic model using multivariate simplex splines, with at least zeroth order continuity between the subdomains. This goal was achieved. Based on the conclusions drawn in this thesis there were some recommendations that could be made in order to make the model better and more suitable for use in control methods.

1. Only zeroth order continuity results in discontinuities in the first derivative. This has a negative effect on the quality of control allocation algorithms.
2. Wiggles occur at various points in the model.
3. The model took a long time to construct for this order model. If a higher order model was to be used an even higher computation time would be required. A stronger computer or different method could be used in order to reduce computation time.
4. The model required imputed data points in order to be solvable. A better method could be found in order to solve this problem.

Analyzing the recommendations stated above, it was determined that the primary objective of this thesis project is to develop a high fidelity aerodynamic model for the Innovative Control Effectors Aircraft with at least first order continuity throughout the domain, by taking a distributed system identification approach to construct a multivariate Simplotope B-Spline model.

Additionally to this objective, the goals is also to reduce the computation time. As the model needs to be identified, it cannot take unlimited time to be solved. The model needs to be evaluated in real time as well to be usable in a controller of the ICE aircraft.

The above objectives, and thorough review of the literature of B-splines and (distributed) solvers, the research questions summarizing the content of this project are:

1. *How can a multivariate Simplotope B-Spline model of the ICE aircraft be constructed that can be used in modern model-in-the-loop controllers, and what are the advantages and disadvantages compared to the current simplex spline modelling approach?*
    (a) Can the model be constructed for at least 5 dimensions?

(b) How can this model be constructed with first order continuity between the Simplotopes?

(c) How much extra measurements needs to be generated using data augmentation methods?

(d) How can this model be evaluated in real time?

2. *How does using Simplotope B-Splines affect the model estimation?*

   (a) How does it perform compared to Simplex B-splines?
   - Statistically
   - Computationally
   - Accuracy
   - Data Requirements

   (b) How does it perform compared to the look-up tables designed by Lockheed Martin?
   - Statistically
   - Computationally
   - Accuracy

3. *How does the distributed system identification approach affect the model estimation?*

   (a) How much can computational time be reduced compared to global approaches?

   (b) To what extent is the quality of the model affected?
   - Statistically
   - Computationally
   - Accuracy

4.2. Distributed System Identification Approach

Based on the literature review in Chapter 3 an approach was chosen to identify the aerodynamic model of the ICE aircraft. When looking at the several challenges in Section 4.1, a distributed approach is chosen that combines Simplotope B-Splines together with a distributed solver.

Simplotopes are easier to fill with data which is beneficial as the data set is relatively small in order to estimate a high degree model. Another advantage is that the dimensions of the model can be decoupled and better adapted to the shape of the data set: lower degrees can be chosen in dimensions that contain a lower amount of measurements. Additionally a simplotope spline requires in general less coefficients [30] to represent the model, and thus less datapoints within the simplotopes. This is better because when the continuity order is increased to one, a higher degree model is preferred in order to have the same approximation power.

The distributed solver is chosen because the higher quality model will require more computational power to be solved, as the degree will have to be increased. Therefore more coefficients have to be estimated and a bigger constraint matrix is used. Using a distributed solver has the potential to significantly reduce the computation time of the parameters, while only marginally reducing the quality of the model.

This new distributed approach is completely new for aerodynamic model identification in general, and higher dimensional system identification. Distributed solving adds the possibility of infinite scalability, because of the property that it can be solved in parallel and is thus invariant to the full size of the problem, and only depending on the smallest unit. As more accurate models require higher degree polynomials, a well scalable solver is highly beneficial. This distributed approach removes computation time as a limiting factor, opening opportunities of estimating highly accurate aerodynamic models.

For now, dual ascent is chosen as distributed solver. This has proven to work well in system identification applications, and more specifically in multivariate B-splines. As the application to Simplotope splines is similar this adaption should be possible. When this turns out not to be working as desired, it converges slowly or does not meet the constraints properly, a different approach such as ADMM can be implemented.

Optionally, a data augmentation method will be used in order to increase the amount of data required to make the least squared estimator solvable. First the options will be analyzed to solve the data scarcity by using the Simplotope splines, but if this results in models with lower than desired quality the amount of data should be increased by using one of the methods described in Section 3.4.

4.2.1. Estimation Algorithm

When the above indicated plan, the full estimation routine will consists of the steps that are explained below. These steps have to be carried out for each individual submodel of the complete ICE model, and thus be repeated multiple times. The general flow of the estimation algorithm is depicted in Figure 4.1. The parts of this flow diagram are explained in more detail in the rest of this section.

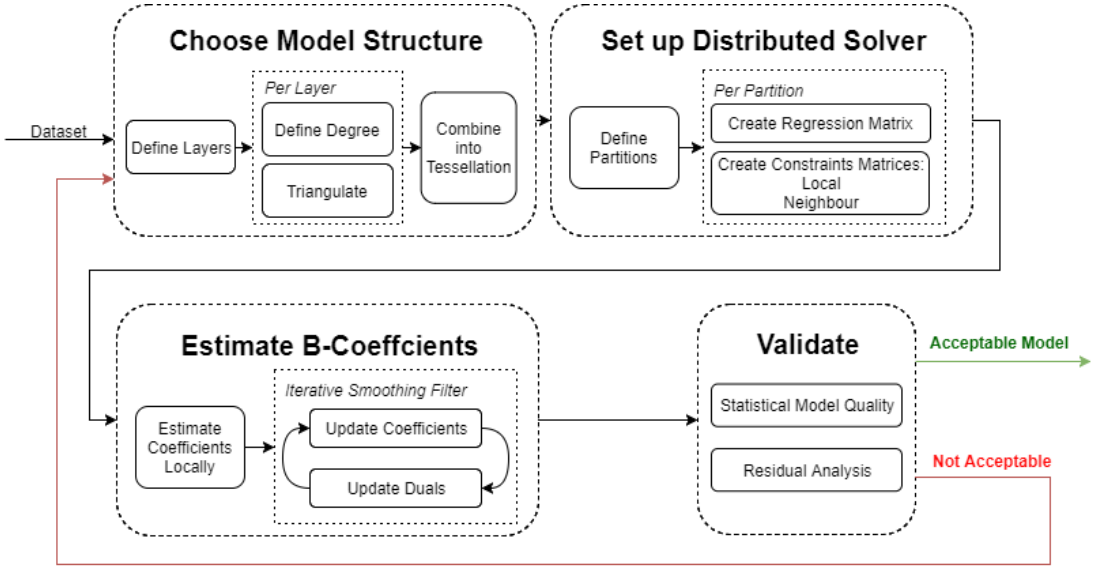


Figure 4.1: Flow diagram of the distributed B-coefficients estimation routine.

Define Layers and Degree Defining which layers is a design choice that has to be made beforehand. The layers consist of groups of dimensions that will be triangulated together. Visser et al. [30] stated: "If no analytic model is available, the shape of the data set can play a leading role. The goal would then be to make sure that the projections can be triangulated efficiently in the next step." Using a cubic approach, by assigning each dimension to its own layer, the amount of coefficients is minimized.

As the ICE model does not come with an analytical model, there is no prior information in this sense. This means that the focus should be on the shape of the dataset. That means that if a dimension e.g. contains many measurements, a finer triangulation or higher degree can be used compared to dimensions with only two or three datapoints.

Triangulate per Layer and Combine into Tessellation The first step of the triangulation is to divide the domain in hypercubes. A hypercube in each layer, possibly including both univariate and multivariate layers, can be triangulated using Kuhn's triangulation algorithm, explained in Section 3.3.1. All the individual layers' triangulations are combined into one tessellation.

Define Partitions The first part of the distributed solver is to divide all the simplotopes in partitions. A decision has to be made on the amount of partitions, how they will be divided over the dimensions, and which simplotope exactly fits in which partition.

Find Local Constraint- and Regression Matrices There are two local constraint matrices that have to be found. The first one has to define all the constraints between Simplotopes *within* one partition. The second one is the constraint matrix between Simplotopes within the partition and the Simplotopes in neighbouring partitions. Additionally, the regression matrix of the partition has to be found, which can be used to estimate the local coefficients. These regression matrices are block diagonal, with each simplotope's regression matrix on the diagonal.

Run the Distributed Optimization Routine The actual estimation is performed by the D-SABRE method in high dimensional spaces [12]. Two main steps have to be performed. First the local coefficients have to

be estimated by using any of the methods described in Section 3.3.3. After that, the iterative post smoothing filter from Section 3.3.4 is run in order to enforce continuity between the partitions.

4.2.2. Quality Analysis

After the model is estimated the quality of the model itself as well as that of the estimation routine has to be estimated. In order to At first the model can be validated by looking at several statistical metrics.
Computing the Root Mean Square Error (RMS) will give an indication of the approximation power of the model. The RMS is computed with the average of the squared errors in the validation points, as can be seen in Equation (4.1) [8]. The RMS term is normalized by dividing through the range of the data set.

$$RMS = \frac{1}{\max \boldsymbol{Y} - \min \boldsymbol{Y}} \sum_{n=1}^{N} \left(y_n - \hat{y}_n\right)^2 \tag{4.1}$$

Another statistical measure is the *parameter variance*. In least squares routines this can be computed by using the inverse of the regression matrix times its transpose, which is used to estimate the parameters. The resulting matrix will have the variances on the diagonal, and the covariances with the other parameters in the other spots. When using ordinary least squares to estimate the parameters, this matrix will have to be multiplied with the estimated variance of the residuals. This can be estimated with Equation (4.2) [20], which is similar to the RMS computation. The sum is divided by the amount of free data, which is defined as the number of measurements minus the amount of parameters used in order to estimate the model.

$$\hat{\sigma}^2 = \frac{1}{N-p} \sum_{n=1}^{N} \left(y_n - \hat{y}_n\right)^2 \tag{4.2}$$

As a new routine is being used for the estimation of the parameters it will be useful to assess the quality of this new method. There are several metrics that can be compared to see whether the new method is better or worse than previously used methods:

- *Computation time:* The goal is to lower the computation time. It can be estimated by running the routine multiple times and estimate a mean and a standard deviation. Additional option is to estimate the computational complexity analytically and compare these. For many used methods the computational complexity is already know in literature.
- *Meeting the constraints:* The global constraint matrix H has to be zero when multiplied with the parameters. The values of the global and distributed methods can be compared in order to have an indication whether using distributed solving has a negative effect on how well the constraints are met.
- *Goodness of Fit:* At last there is a cost function defined that has to be minimized. Comparing the values of the cost function of the different methods, a lower cost will indicate that the model approximates the identification data better. Another useful metric is the coefficient of determination, indicating how well the estimated model's variance explains the variance in the identification data.

It is expected that using distributed solvers will decrease the computation time, but that the constraints are not strictly met as in global methods and that the cost function might not be as low. The question in this will be whether this decrement in quality is acceptable compared to the decrease in computation time.

4.3. Demonstration of a Simplotope B-Spline

In this section a Simplotope spline is demonstrated, in order to show its ability to accurately estimate a submodel of the ICE aircraft. This has been done on two submodels indicating the basis aerodynamic coefficients. The first one, $C_{N_1}(\alpha, M)$ had to be constructed in 2D, while $C_{LN_2}(\alpha, \beta, M)$ is a 3D model. Both models have been solved using Simplotope Spline with each dimension its own layer, and with all dimensions in one layer, making it a Simplex spline. The difference in quality between these models is assessed.

As another demonstrator the Simplotope Spline models are solved using the distributed approach, in order to show the capability to work in higher dimensions, and assess its quality compared to a global solver.

The following models should only be considered as a proof of concept. Little time has been spent on finding a good model structure, which means that the final quality is likely to improve. The definition of the models that are chosen in order to compare the chosen methods are shown in Table 4.1. For the demonstration of

the distributed approach the same Simplotope model has been used, but with the partitions as defined in Section 4.3.2.

Table 4.1: Definition of the models use for the demonstration and comparison.

Metric		**2D Model:** $C_{N_1}(\alpha, M)$	**3D Model:** $C_{LN_2}(\alpha, \beta, M)$
Layers	*Simplex*	(2)	(3)
	Simplotope	(1,1)	(1,1,1)
Degree	*Simplex*	(3)	(3)
	Simplotope	(3,3)	(3,3,3)
Continuity	*Simplex*	(1)	(1)
	Simplotope	(1,1)	(1,1,1)
Gridpoints	α	[-5,15,25,35,50,90]	[-5,7.5,22.5,60,90]
	M	[.3,.95,1.6,2.16]	[0.3,1.2,2.16]
	β	-	[-30,0,30]

For the global approaches the Kernel Based method has been used. The same method was used to estimate the local coefficients of the partitions in the distributed approach.

4.3.1. Simplotope B-Spline

A comparison has been made between the Simplex and Simplotope structures for defining the aerodynamic model. The gridpoints are kept the same, with the only difference being that for the Simplotope structure each dimension is triangulated individually, and the Simplex structure has a single 2D/3D triangulation. All the results of the comparisons are shown in Table 4.2. Note that the time is measured at a HP Probook, with all the code written in Matlab. Optimization should be able to speed the process up, but in general the relative difference will not differ that much and these times will give an indication of the possible differences.

Table 4.2: Comparison of the Simplex structure with the Simplotope Structure.

Metric	**2D Model**		**3D Model**	
	Simplex	*Simplotope*	*Simplex*	*Simplotope*
Computation Time [s]	0.5	0.3	5.7	1.1
Number of Parameters [-]	300	240	1920	1024
Cost Function [-]	0.0037	0.0017	0.0089	0.0034
Coefficient of Determination [-]	0.97	0.99	0.75	0.89

The models for $C_{N_1}(\alpha, M)$ can be seen in Figures 4.2 and 4.3 for the Simplex and Simplotope structure respectively. On first sight there is not a big difference between the two models, and when comparing the results in Table 4.2 confirms this observation. However, some small differences can be noted and these are mostly beneficial for the Simplotope Spline. It can be seen that the the number of parameters is 20% less for the Simplotope spline, and that in all performance metric is still slightly better. The one that can be discussed about is the Coefficient of Determination, as a value close to 1.0 might suggest that the model is over-fitting the identification data.

As the aerodynamic model of $C_{LN_2}(\alpha, \beta, M)$ has too much dimensions to visualize in a single graph, two slices of the model, at Mach numbers of 0.3 and 2.16, are shown for both structures in Figures 4.4 to 4.7. Some differences can be seen already in the visualizations, especially in Figure 4.7. The model becomes slightly inaccurate at the edges, where wiggles start occurring in Figure 4.6, while the simplotope model remains stable as can be seen in Figure 4.7. In the same graph at negative angles of sideslip it can be seen that there is still some wiggles in the simplotope model. Because of the continuity order the shapes of the simplices are more linked and thus the inaccuracies on the edge of the domain will affect other simplices.

Numerically the differences can also be seen in Table 4.2. Especially the computational time of the Simplotopes is lower than that of the Simplex spline, a reduction of 80%. Analyzing the time of all the parts of the algorithm, there were two parts at which the Simplotope structure performed better than the Simplex

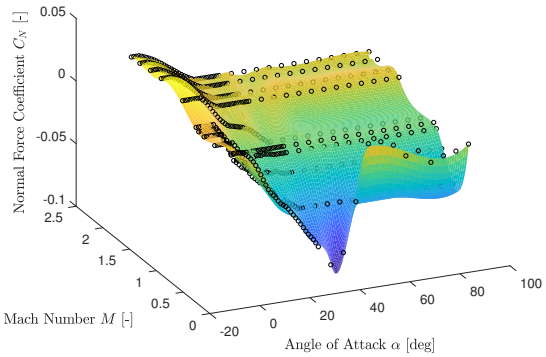


Figure 4.2: Simplex B-spline model of the subcoefficient $C_{N_1}(\alpha, M)$

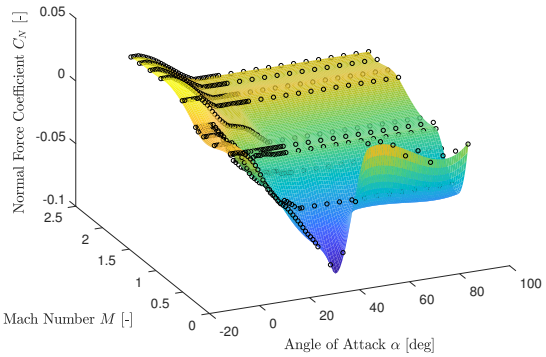


Figure 4.3: Simplotope B-spline model of the subcoefficient $C_{N_1}(\alpha, M)$

structure. First the determination of the continuity conditions. Where the Simplices require 2432 equations to ensure 1st order continuity, the Simplotopes only need 896. This is a direct result of the fact that every cube contains six Simplices that all require to be coupled, while the Simplotope does not have these. Next to that, most of the equations for the Simplotopes are copies of each other, and not new equations. The second part is actually solving the equations. Both finding the kernel requires more time, as well as finding all the coefficients. This is not surprising as the amount of coefficients is almost double for the Simplices.

While requiring less time and coefficients, the Simplotope model is better fitting the identification data: it has a better optimized cost function and a better coefficient of determination.

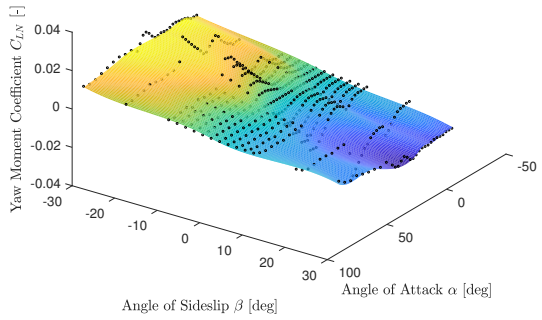


Figure 4.4: Simplex B-spline model of the subcoefficient $C_{LN_2}(\alpha, \beta, M)$ at $M = 0.30$

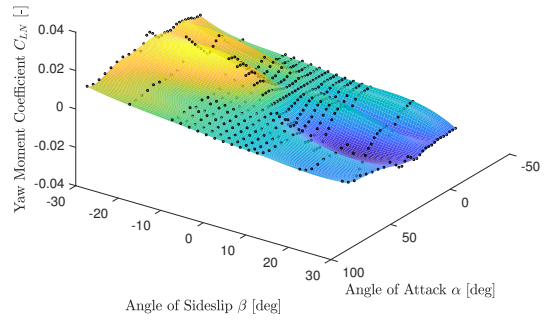


Figure 4.5: Simplotope B-spline model of the subcoefficient $C_{LN_2}(\alpha, \beta, M)$ at $M = 0.30$

Based on this analysis using Simplotopes is, especially in 3D, considerably better than the Simplices. The expectation is that this continues in higher dimensions, as the difference for the amount of coefficients and the continuity equations will only increase. This shows that choosing the Simplotope structure will not only be beneficial as for the low amount of required coefficients, but also for the computation time of the model. These two effect are both beneficial for the objectives of this project.

Note that no validation data has been used yet, but only comparisons have been made based on the identification data that is available. Additionally, all the condition numbers of the regression multiplications are high, suggesting an ill conditioned problem. As this is not desired, more data would be preferred and one of the data augmentation methods as stated in Section 3.4 could be used to create a better conditioned problem.

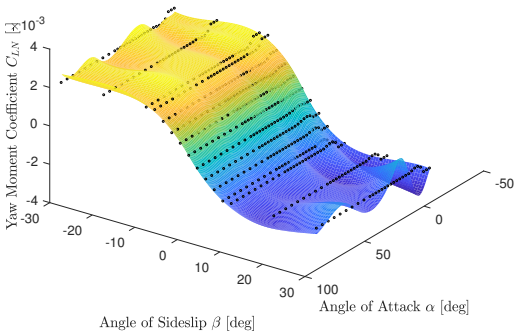


Figure 4.6: Simplex B-spline model of the subcoefficient $C_{LN_2}(\alpha, \beta, M)$ at $M = 2.16$

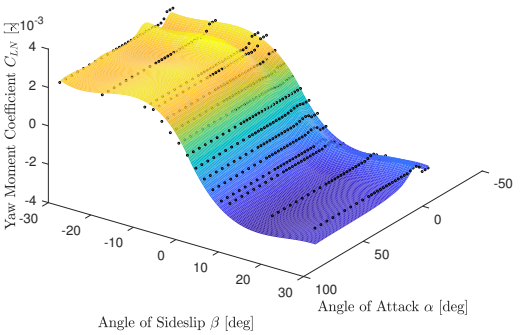


Figure 4.7: Simplotope B-spline model of the subcoefficient $C_{LN_2}(\alpha, \beta, M)$ at $M = 2.16$

4.3.2. Distributed Approach

As it has been determined that choosing Simplotopes is the better choice, a comparison for this structure has been made. A simple, manual division of the partitions has been made in order to show the performance of the different strategies for estimating the parameters. The triangulation of the 2D model has been divided into two partitions: the first 7 simplotopes, and the second 8. There are two partitions for the 3D model as well: one with all Simplotopes that have the first simplex of the Mach layer, and the other one with all the Simplotopes that have simplex two of the Mach layer. The parameters have been estimated by using an iteration constant of $\alpha(l) = 0.005$, and the maximum amount of iterations being 15. The results are shown in Table 4.3.

A comparison has been made how well the models satisfy the constraints as set by the continuity equations. The metric that has been used in order to compare these is the maximum norm of the continuity matrix multiplied with the estimated coefficients: $||H\boldsymbol{c}||_\infty$. The maximum norm is used as the maximum discontinuity better indicates the possible degredation of control methods using this aerodynamic model.

Table 4.3: Comparison of the Simplex structure with the Simplotope Structure.

Metric	**2D Model**		**3D Model**	
	Global	*Distributed*	*Global*	*Distributed*
Computation Time [s]	0.3	0.3	1.1	0.8
Cost Function [-]	0.0017	.0006	0.0034	0.0023
Coefficient of Determination [-]	0.99	0.99	0.89	0.92
Constraint Satisfaction [-]	0.000	0.061	0.000	0.029

Looking at the computation time of the 2D model it is surprising that the distributed model takes more time to solve then the global solver. However, the computation time of the 3D model is reduced by the distributed solver from 4.9 seconds to 1.0. Apparently the additional time that it takes to set up the different partitions, is more than the time won by solving the 2D model in a distributed manner. However, when the model size increases the time can be reduced considerably. As the model size of 4D and 5D models will only enlarge, it is expected that estimating the parameters in partitions will have an even bigger benefit in terms of computational time.

Another benefit that has been seen while exploring bigger models in terms of data, parameters, and continuity conditions, is that physical memory starts becoming a problem. The matrices that have to be inverted turn out to be full and not fit in the memory. In this case the only solution is to split up the domain in partitions and solve it in a distributed fashion.

The difference in the other metrics is also as expected. The distributed model will approximate the identification data better than the global model, since the parameters are estimated locally with less concern about the continuity equations. On the other hand, where the global model has all zeroes when the continuity con-

ditions are computed, the distributed model does not perform as well. The maximum value for the 2D model is 0.061, and for the 3D model 0.045. This indicates that the post smoothing filter does not perform optimally and still leaves some constraints unsatisfied.

At this moment a solution to improve the constraints has not been investigated yet. An option is to use an overlap, which will include the simplotopes just over the edge of a partition in order to create a better first estimate of the local coefficients. A different option is to change the distributed solving method to investigate whether that one is more appropriate to be used in this case. ADMM is a possible solution.

5

Conclusions and Future Work

Accurate aerodynamic models are necessary in order to ensure the quality of model-in-the-loop controllers. These models can be used for determining the virtual input, but also for optimal control allocation over the available control effectors. For this last use and accurate model of the ICE aircraft is required. The tailless aircraft has it has a total of 13 control effectors, resulting from the demand of having a high performance fighter, while maintaining a low RCS. These requirements put constraints on the shape of the aircraft, the freedom of control effectors, and had the tail removed. These limitations imposed challenges in the controllability and stability of the fighter, requiring state-of-the-art controllers and control allocation algorithms to make it steerable.

The dataset that was provided for identifying the aerodynamic model consists of 108 submodels, divided over 3 force and 3 moment coefficients, that each described an increment in a coefficient based on the aircraft's state or a control deflection. Using this dataset a model based on multivariate B-splines was created. Although this model approximated the dataset accurately, it showed room for improvement as it had discontinuities and sign changes in the first derivative, and it required a tremendous amount of time to estimate its parameters. Additionally, the provided dataset was not large enough in order to achieve high accuracy, which resulted in datapoints that had to be imputed.

This need for improvements and the limitations of the model led to the primary objective of this thesis project to develop a high fidelity aerodynamic model for the Innovative Control Effectors Aircraft with at least first order continuity through the domain, by taking a distributed system identification approach to construct a multivariate Simplotope B-Spline model.

Simplotopes are a more general form of simplices. It allows the domain to be divided into layers, each with its own triangulation and degree, which makes it possible to adapt the model structure more to the shape of the dataset. Additionally, it requires less coefficients while achieving better approximation power, which is beneficial in datasets with a low amount of measurements.

The distributed solver that has been chosen is based on the D-SABRE method, which has been combined with multivariate Simplex splines for WFR. Although estimating the B-Coefficients of a Simplotope B-spline model is a global problem, the local nature of the Simplotopes and the sparsity of the global continuity matrix supports a distributed approach in order to save time. The D-SABRE divides the domain into partitions containing multiple simplices, estimates the B-Coefficients locally, and applies a post-smoothing filter based on dual ascent optimization.

Two proofs of concept has been built on lower degree models, one showing the usability of the Simplotope spline structure, the other one using Simplotope splines in combination with a distributed solver. It can be seen that when the Simplotope spline is compared with the same hypercubes as a Simplex spline, the Simplotope structure requires less coefficients, while being more accurate, and 80% less computation time for the 3D model.

However, the distributed solver was not as accurate as required. Although the computational time was decreased for the 3D Simplotope B-Spline by 80% more, it showed discontinuities between the chosen par-

titions, both in the first and zero order equations. The local estimation of the partitions were too far off to be fully solved by the post-smoothing filter. Possible solutions that are considered are using an overlap: use the simplotopes on the edge for the initial estimation of the partitions' coefficients, or using a different distributed estimation algorithm such as ADMM.

Besides improving the distributing solving routine to enhance its quality, there are several other tasks that need to be performed in the future months. First the code has to be adapted to make it work for all the submodels of the ICE model. This includes making it general for all dimensions, divide these dimensions over a chosen amount of layers, and triangulate the layers individually.

Additionally, the program needs to be evaluated in real time. The evaluation function is written in `C++` and highly optimized for Simplex splines. In order to make practical use of the Simplotope model, this evaluation function needs to be adapted and optimized for use with Simplotope Splines. The same holds for the determination of the derivatives, as the de Casteljau matrix needs to be defined for Simplotope splines.

During development of previous models, the amount of data in the provided set was not sufficient to make the optimization problem solvable. As Simplotope splines solves this partially, the dataset might need augmentation. If this applies, a data augmentation method, as suggested in Section 3.4, needs to be implemented in order to acquire a high accuracy model.

Bibliography

[1] I. J. Anderson, M. G. Cox, and J. C. Mason. Tensor-product spline interpolation to data on or near a family of lines. *Numerical Algorithms*, 5(4):193–204, apr 1993. ISSN 1017-1398. doi: 10.1007/BF02210502. URL https://doi.org/10.1007/BF02210502http://link.springer.com/10.1007/BF02210502.

[2] Gerard Awanou, Ming-Jun Lai, and Paul Wenston. The Multivariate Spline Method for Scattered Data Fitting and Numerical Solutions of Partial Differential Equations. In *Wavelets and Splines*, pages 24–75, Athens, 2005. ISBN 0-9728482-6-6.

[3] Jürgen Bey. Simplicial grid refinement: on Freudenthal's algorithm and the optimal number of congruence classes. *Numerische Mathematik*, 85(1):1–29, 2000. ISSN 0029599X. doi: 0.1007/s002110050475.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2010. ISSN 1935-8237. doi: 10.1561/2200000016. URL http://www.nowpublishers.com/article/Details/MAL-016.

[5] P. D. Bruce and M. G. Kellett. Modelling and identification of nonlinear aerodynamic functions using b-splines. In *Control '98. UKACC International Conference on (Conf. Publ. No. 455)*, pages 907–912 vol.2, Sep 1998. doi: 10.1049/cp:19980349.

[6] James M. Buffington. Modular Control Law Design for the Innovative Control Aircraft Effectors (ICE) Tailless Fighter Configuration. Technical report, Air Force Research Laboratory AFRL/VAAD, Wright Patterson AFB, 1999.

[7] Carl de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972. ISSN 10960430. doi: 10.1016/0021-9045(72)90080-9.

[8] C. C. De Visser. *Global Nonlinear Model Identification with Multivariate Splines*. PhD thesis, Delft University of Technology, 2011.

[9] C. C. De Visser and Michel Verhaegen. Wavefront reconstruction in adaptive optics systems using nonlinear multivariate splines. *Journal of the Optical Society of America A*, 30(1):82, 2013. ISSN 1084-7529. doi: 10.1364/JOSAA.30.000082. URL https://www.osapublishing.org/abstract.cfm?URI=josaa-30-1-82.

[10] C. C. De Visser, Q. P. Chu, and J. A. Mulder. A new approach to linear regression with multivariate splines. *Automatica*, 45(12):2903–2909, dec 2009. ISSN 00051098. doi: 10.1016/j.automatica.2009.09.017. URL http://linkinghub.elsevier.com/retrieve/pii/S0005109809004270.

[11] C. C. De Visser, Q. P. Chu, and J. A. Mulder. Differential constraints for bounded recursive identification with multivariate splines. *Automatica*, 47(9):2059–2066, 2011. ISSN 0005-1098. doi: 10.1016/j.automatica.2011.06.011. URL http://dx.doi.org/10.1016/j.automatica.2011.06.011.

[12] C. C. De Visser, Elisabeth Brunner, and Michel Verhaegen. On distributed wavefront reconstruction for large-scale adaptive optics systems. *Journal of the Optical Society of America A*, 33(5):817–831, 2016. ISSN 1520-8532. doi: 10.1364/JOSAA.33.000817.

[13] Wei Deng, Ming-Jun Lai, Zhimin Peng, and Wotao Yin. Parallel Multi-Block ADMM with o(1 / k) Convergence. *Journal of Scientific Computing*, 71(2):712–736, 2017. ISSN 08857474. doi: 10.1007/s10915-016-0318-2.

[14] Kenneth M. Dorsett and David R. Mehl. Innovative Control Effectors (ICE) Phase I. Technical report, Lockheed Martin Tactical Aircraft Systems, 1996. URL http://oai.dtic.mil/oai/oai?verb=getRecord{&}metadataPrefix=html{&}identifier=ADB041508.

[15] Gerald Farin. A History of Curves and Surfaces in CAGD. In *Handbook of computer aided geometric design*, chapter 1, pages 1–23. 1984.

[16] M Holmes, a Gray, and C Isbell. Fast SVD for large-scale matrices. *Workshop on Efficient Machine ...*, 1 (1):2–3, 2007. URL http://sysrun.haifa.il.ibm.com/hrl/bigml/files/Holmes.pdf.

[17] X. Hong, R.J. Mitchell, S. Chen, C.J. Harris, K. Li, and G.W. Irwin. Model selection approaches for non-linear system identification: a review. *International Journal of Systems Science*, 39(10):925–946, oct 2008. ISSN 0020-7721. doi: 10.1080/00207720802083018. URL http://www.tandfonline.com/doi/abs/10.1080/00207720802083018.

[18] V. Klein and E. A. Morelli. *Aircraft System Identification: Theory and Practice*. American Institute of Aeronautics and Astronautics, Inc., 2006. ISBN 1563478323.

[19] Ming-Jun Lai and Larry L. Schumaker. *Spline Functions on Triangulations*. Cambridge University Press, Cambridge, 1 edition, 2007. ISBN 9780511889479. URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780511889479.

[20] Roderick J A Little and Donald B Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition, 2002. ISBN 0471183865. doi: 10.2307/1533221.

[21] Michael A. Niestroy, Kenneth M. Dorsett, and Katherine Markstein. A Tailless Fighter Aircraft Model for Control-Related Research and Development. In *AIAA Modeling and Simulation Technologies Conference - AIAA SciTech Forum*, number January, pages 1–18. American Institute of Aeronautics and Astronautics, 2017. ISBN 978-1-62410-451-0. doi: 10.2514/6.2017-1757. URL http://arc.aiaa.org/doi/10.2514/6.2017-1757.

[22] João Silva, Tamás Keviczky, and Michel Verhaegen. A Distributed Approach for Solving Wavefront Reconstruction Problems. In *American Control Conference (ACC)*, pages 6513–6518, Boston, 2016. ISBN 9781467386814. URL ieeexplore.ieee.org/document/7526695/.

[23] L. G. Sun, C. C. De Visser, Q. P. Chu, and J. A. Mulder. Online Aerodynamic Model Identification Using a Recursive Sequential Method for Multivariate Splines. *Journal of Guidance, Control, and Dynamics*, 36(5):1278–1288, 2013. ISSN 0731-5090. doi: 10.2514/1.60375. URL http://arc.aiaa.org/doi/10.2514/1.60375.

[24] L. G. Sun, C. C. De Visser, and Qiping P. Chu. A New Substitution Based Recursive B-Splines Method for Aerodynamic Model Identification. In *Advances in Aerospace Guidance, Navigation and Control: Selected Papers of the Second CEAS Specialist Conference on Guidance, Navigation and Control*, pages 233–245. Springer, Berlin, Heidelberg, Berlin, Heidelber, 2013. doi: 10.1007/978-3-642-38253-6_15. URL https://link.springer.com/chapter/10.1007/978-3-642-38253-6{_}15.

[25] Martin A. Tanner and Wing Hung Wong. The Calculation of Posterior Distributions By Data Augmentation. *Journal of the American Statistical Association*, 82(398):528–540, 1987. ISSN 01621459. doi: 10.2307/2289457. URL http://wrap.warwick.ac.uk/24939/.

[26] H. J. Tol, C. C. De Visser, E. van Kampen, and Q. P. Chu. Nonlinear Multivariate Spline-Based Control Allocation for High-Performance Aircraft. *Journal of Guidance, Control, and Dynamics*, 37(6):1840–1862, 2014. ISSN 0731-5090. doi: 10.2514/1.G000065. URL http://arc.aiaa.org/doi/10.2514/1.G000065.

[27] H. J. Tol, C. C. De Visser, L. G. Sun, E. van Kampen, and Q. P. Chu. Multivariate Spline-Based Adaptive Control of High-Performance Aircraft with Aerodynamic Uncertainties. *Journal of Guidance, Control, and Dynamics*, 39(4):781–800, 2016. ISSN 0731-5090. doi: 10.2514/1.G001079. URL http://arc.aiaa.org/doi/10.2514/1.G001079.

[28] I. V. van der Peijl. Physical Splines for Aerodynamic Modelling of Innovative Control Effectors. 2017.

[29] I. V. van der Peijl. *Physical Splines for Aerodynamic Modelling of Innovative Control Effectors*. PhD thesis, Delft University of Technology, 2017.

[30] Tim Visser, C. C. De Visser, and Erik-Jan Van Kampen. Quadrotor System Identification Using the Multivariate Multiplex B-Spline. *AIAA Atmospheric Flight Mechanics Conference*, (January):1–13, 2015. doi: doi:10.2514/6.2015-0747.

[31] Tim Visser, C. C. De Visser, and Erik-Jan Van Kampen. Towards the multivariate simplotope spline: continuity conditions in a class of mixed simplotopic grids. 2016.

III

Appendices

A

Aerodynamic Model Identification Results

This appendix presents all of the identification results. In Appendix A.1, an overview of the chosen models is given, including the layer division and the amount of simplotopes, among others. Appendix A.2 provides a comparison between several possible structures for every submodel to justify the model choice. Appendix A.3 shows results of four different control inputs on the control effectors, and compares the Simplotope B-Spline model with the original Simulink model.

A.1. Details of the ICE Simplotope Spline Model

Table A.1 gives an overview of all the selected simplotope splines for the ICE submodels, a decision that has been made based on the validation results. All models have first order continuity between the subdomains, to provide a smooth, and differentiable surface that can be used in e.g. INCA [10].

The hypercube partitioning of the dataset, as well as the extended dataset, is taken from [16]. This involves single imputed data using cubic interpolation. In order to maintain simplicity, all the submodels' structures for every main coefficient have been kept the same. This is beneficial for evaluation purposes, as the polynomial bases of the submodels can be shared between the subcoefficients, saving computation time.

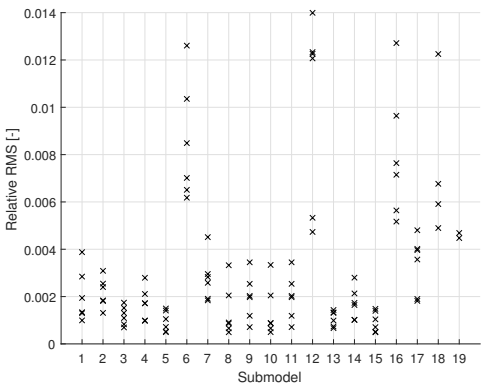
All the submodels are combined in Simulink, collecting the individual outputs into six main coefficients. In order to be able to evaluate all models in real time, an optimized `MEX` function has been written in C++. The components and activity flow of this evaluation function will be described in Appendix C.3.

Figure A.1 presents a quality assessment of the models that have been used. It provides a an overview of all RMS error in Figure A.1a and the maximum residual in Figure A.1b, both scaled with the range of the dataset. These values are based on the validation data as taken from the full dataset, dividing 80% identification and 20% validation. The maximum RMS of an submodel is 1.40%, while the maximum relative error that is found is 28.1%. Of all 108 submodels, the median of the RMS is at 0.192%, and that of the maximum residual at 2.33%.

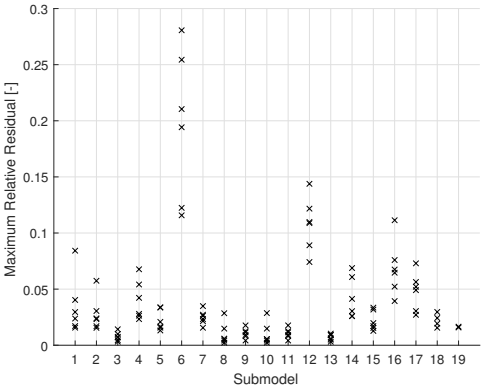
The plots in Figure A.2 show statistical quality of the chosen models. An assumption for Ordinary Least Squares (OLS) is that the residuals are white, which means that the identification residuals should be uncorrelated. Values in Figure A.2a that lie below 95%, are an indication that the residuals are not white and thus an OLS assumption is violated. As other models showed similar results, more research should be put into selecting the best model structure, or using a different estimation methods, such as Generalized Least Squares (GLS), or the Maximum Likelihood Estimation (MLE), using less strict assumptions and thus making a statistically better structured model. This is especially important when operating in noisy environments. The polynomials have a stable basis, as the B-Coefficients limit the values coming out of the model. The percentage of coefficients that lie in the data value range, are a measure of a well behaving spline, and should ideally be 100% [4]. It can be seen in Figure A.2b that most models do not have a 100% score, but mostly around 90-95%, with a single outlier at 60%.

Table A.1: Overview of the submodels of the ICE aircraft and their parameters, data range, and number of datapoints

Coefficient	Parameters	Data Range	Datapoints	Cubes	Layers	Degrees	Simplotopes	Coefficients
C_{*_1}	α	[-5.0, 90.0]	48	12	(1,1)	(7,6)	120	6,720
	Mach	[0.3, 2.16]	10	10				
C_{*_2}	α	[-5, 90]	30	6	(1,1,1)	(7,6,6)	144	56,448
	β	[-30, 30]	15	6				
	Mach	[0.3, 2.16]	8	4				
C_{*_3}	α	[-2.5, 45]	20	10	(1,1,1)	(3,3,3)	10	640
	β	[-10, 10]	3	1				
	δ_{LIBLEF}	[0, 40]	2	1				
C_{*_4}	α	[-2.5, 45]	20	5	(1,1,1,1)	(5,5,5,5)	10	45,360
	β	[-10, 10]	3	1				
	δ_{LIBLEF}	[0, 40]	2	1				
	δ_{LOBLEF}	[-40, 40]	5	1				
	Mach	[0.3, 1.2]	4	1				
C_{*_5}	α	[-2.5, 90]	28	10	(1,1,1,1)	(5,4,4,5)	160	144,000
	δ_{LSSD}	[0, 60]	4	2				
	δ_{LEL}	[-30, 30]	5	4				
	Mach	[0.3, 2.16]	7	2				
C_{*_6}	α	[-2.5, 90]	29	10	(2,1,1,1)	(3,1,1,3)	80	12,800
	δ_{LSSD}	[0, 60]	2	1				
	δ_{RSSD}	[0, 60]	2	1				
	δ_{PF}	[-30, 30]	5	2				
	Mach	[0.3, 2.16]	7	2				
C_{*_7}	α	[-2.5, 90]	29	10	(1,1,1)	(5,5,4)	40	7,200
	β	[-30, 30]	7	2				
	δ_{LAMT}	[0, 60]	5	2				
C_{*_8}	α	[-2.5, 42.5]	19	9	(1,1,1)	(4,3,3)	9	720
	δ_{LEL}	[-30, 30]	3	1				
	δ_{LAMT}	[0, 60]	3	1				
C_{*_9}	α	[-2.5, 42.5]	19	9	(1,1,1)	(4,3,3)	9	720
	δ_{LOBLEF}	[0, 40]	3	1				
	δ_{LAMT}	[0, 60]	3	1				
$C_{*_{10}}$	α	[-2.5, 42.5]	19	9	(1,1,1)	(4,3,3)	9	720
	δ_{REL}	[-30, 30]	3	1				
	δ_{RAMT}	[0, 60]	3	1				
$C_{*_{11}}$	α	[-2.5, 42.5]	19	9	(1,1,1)	(4,3,3)	9	720
	δ_{ROBLEF}	[0, 40]	3	1				
	δ_{RAMT}	[0, 60]	3	1				
$C_{*_{12}}$	α	[-2.5, 90]	29	15	(1,1,1)	(4,3,3)	90	7,200
	β	[-30, 30]	7	3				
	δ_{LSSD}	[0, 60]	4	2				
$C_{*_{13}}$	α	[-2.5, 45]	20	10	(1,1,1)	(3,3,3)	10	640
	β	[-10, 10]	3	1				
	δ_{RIBLEF}	[0, 40]	2	1				
$C_{*_{14}}$	α	[-2.5, 45]	20	5	(1,1,1,1)	(5,5,5,5)	10	45,360
	β	[-10, 10]	3	1				
	δ_{RIBLEF}	[0, 40]	2	1				
	δ_{ROBLEF}	[-40, 40]	5	1				
	Mach	[0.3, 1.2]	4	1				
$C_{*_{15}}$	α	[-2.5, 90]	28	10	(1,1,1,1)	(5,4,4,5)	160	144,000
	δ_{RSSD}	[0, 60]	4	4				
	δ_{REL}	[-30, 30]	5	2				
	Mach	[0.3, 2.16]	7	2				
$C_{*_{16}}$	α	[-2.5, 90]	29	10	(1,1,1)	(4,4,3)	40	4,000
	β	[-30, 30]	7	2				
	δ_{RAMT}	[0, 60]	5	2				
$C_{*_{17}}$	α	[-2.5, 90]	29	15	(1,1,1)	(4,4,3)	90	9,000
	β	[-30, 30]	7	3				
	δ_{RSSD}	[0, 60]	4	2				
$C_{*_{18,19}}$	α	[0, 30]	13	3	(1,1)	(5,5)	12	432
	Mach	[0.6, 2.2]	6	4				

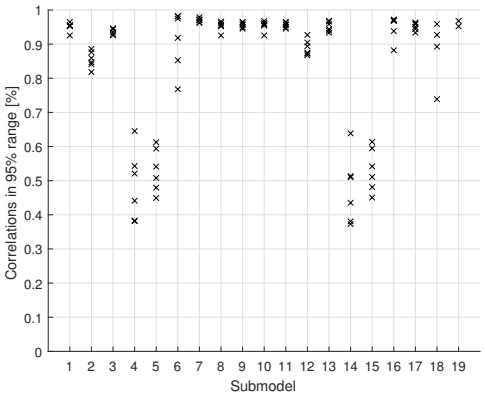


(a) Relative RMS error

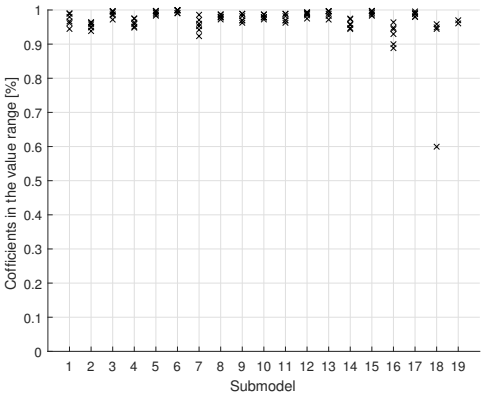


(b) Relative maximum residual

Figure A.1: Validation results of all implemented Simplotope Splines.



(a) Correlation between the Residuals



(b) Percentage Coefficients withing model range

Figure A.2: Statistical quality of all implemented Simplotope Splines.

A.2. Simplotope Model Comparisons

This section contains details of the models that have been studied to be part of the ICE Simplotope model. for which several model structures have been compared in order to find the best performing one. The models are selected based on their validation performance, which means that the relative RMS and maximum relative residual determined the choice. As a baseline, the model degrees, or transformed to the layer degrees, have been chosen from [16]. Generally, this degree has been increased by one, to compensate for the increment from zeroth to first order continuity.

An observation is that the simplotope splines have better performance than the simplex splines. This is partly because the data been assumed noiseless, and thus shows predictable behavior. As Appendices B.2 and B.4 will show, in high noise environments, the simplotope structure might induce overfitting, and thus show bad validation behavior. This is also indicated by the fact that the coefficient of determination come close to 1.0. In case noise-contaminated measurements are added to the set, the model structure should be reexamined.

The models of subcoefficients 3 and 13 in Figures A.5 and A.15 only show a single model, and no comparison whatsoever. Increasing the model degree in any layer, or for any other combination, did result in a singular regression matrix and thus unreliable results.

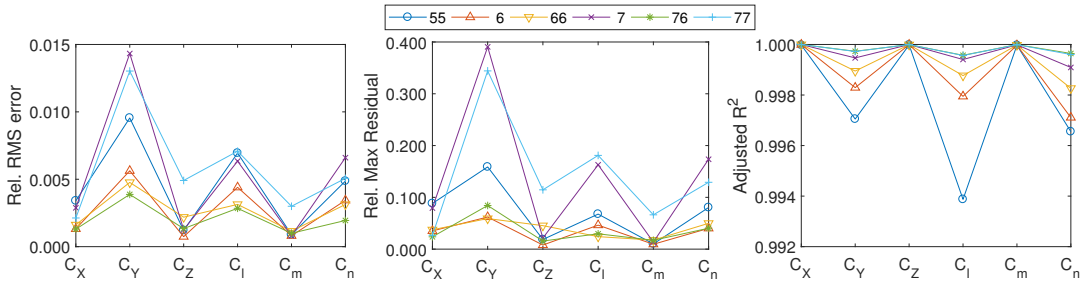


Figure A.3: RMS error, maximum residual, and coefficient of determination of $C_{*_1}(\alpha, M)$

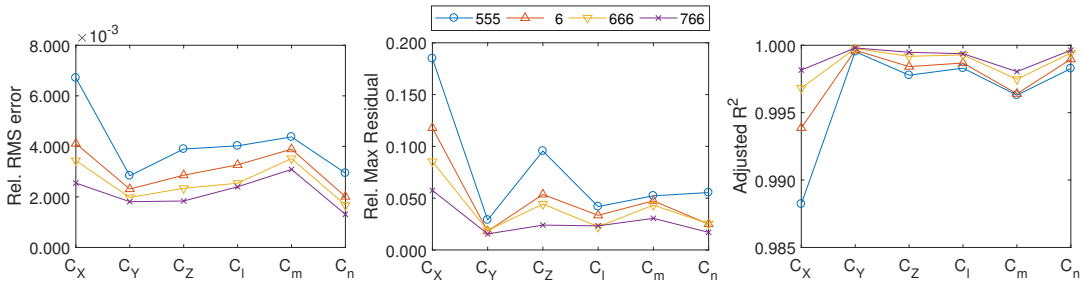


Figure A.4: RMS error, maximum residual, and coefficient of determination of $C_{*_2}(\alpha, \beta, M)$

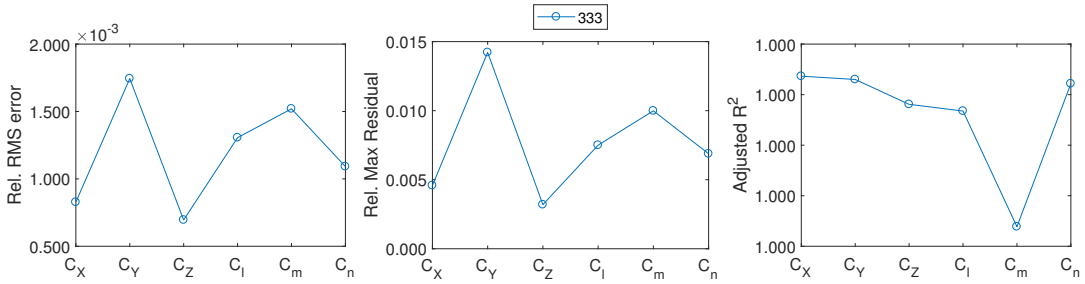


Figure A.5: RMS error, maximum residual, and coefficient of determination of $C_{*_3}(\alpha, \beta, \delta_{LIBLEF})$

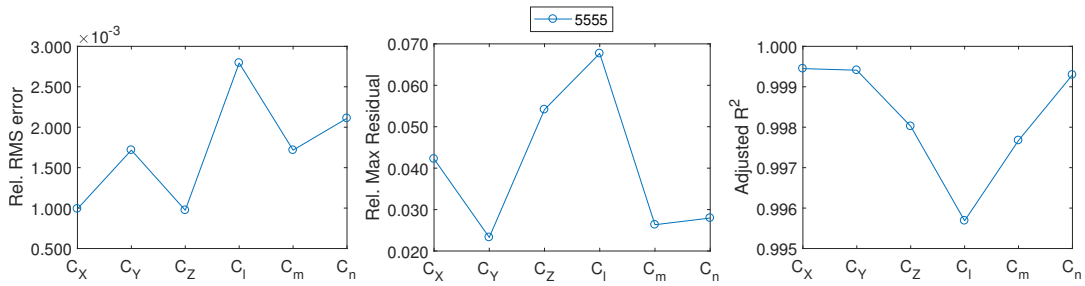


Figure A.6: RMS error, maximum residual, and coefficient of determination of $C_{*_4}(\alpha,\beta,\delta_{LIBLEF},\delta_{LOBLEF},M)$

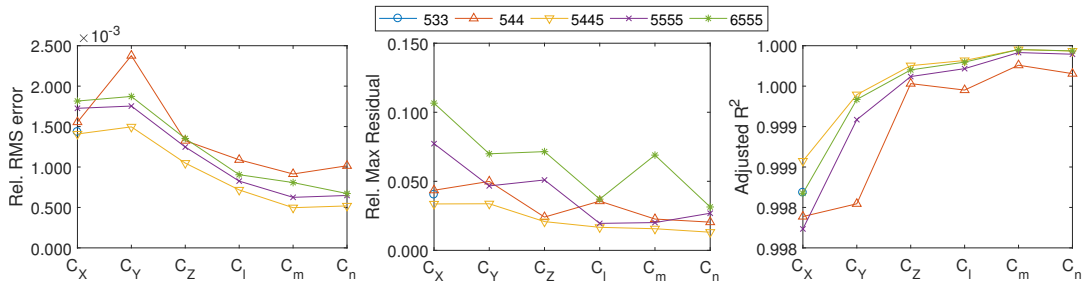


Figure A.7: RMS error, maximum residual, and coefficient of determination of $C_{*_5}(\alpha,\delta_{LSSD},\delta_{LEL},M)$

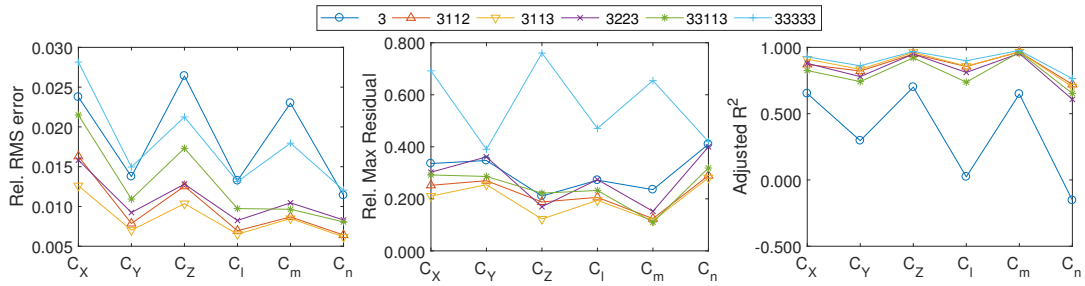


Figure A.8: RMS error, maximum residual, and coefficient of determination of $C_{*_6}(\alpha,\delta_{LSSD},\delta_{RSSD},\delta_{PF},M)$

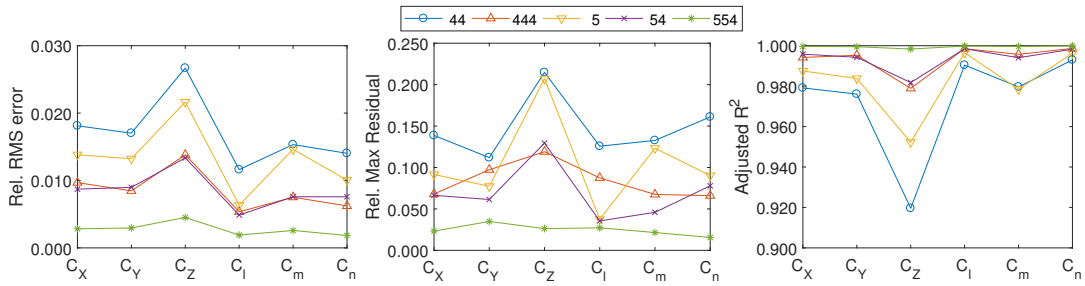


Figure A.9: RMS error, maximum residual, and coefficient of determination of $C_{*_7}(\alpha,\beta,\delta_{LAMT})$

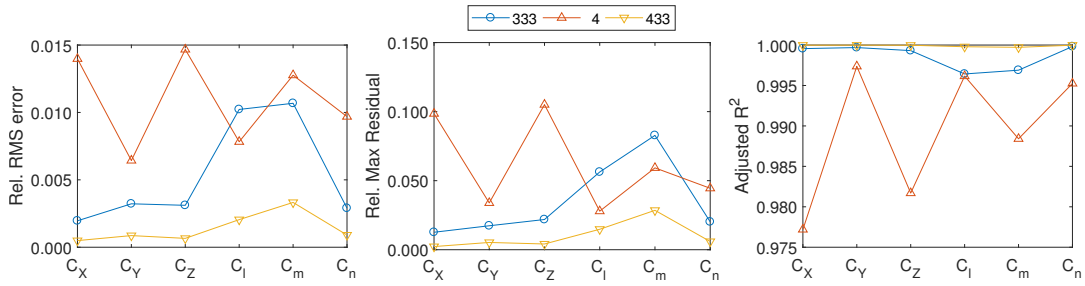


Figure A.10: RMS error, maximum residual, and coefficient of determination of $C_{*_8}(\alpha,\delta_{LEL},\delta_{LAMT})$

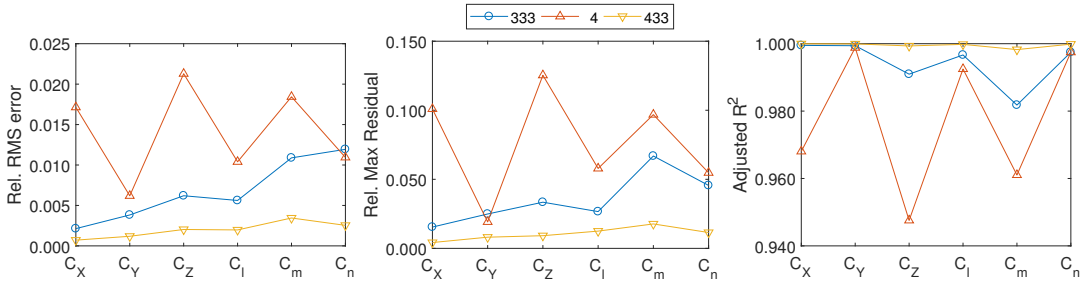


Figure A.11: RMS error, maximum residual, and coefficient of determination of $C_{*_9}(\alpha, \delta_{LOBLEF}, \delta_{LAMT})$

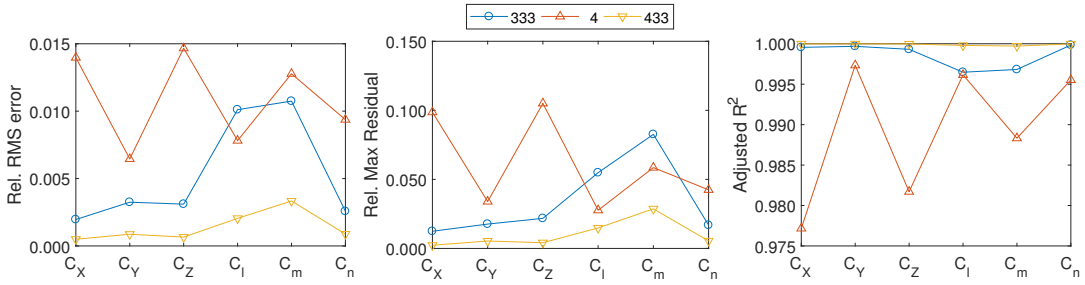


Figure A.12: RMS error, maximum residual, and coefficient of determination of $C_{*_{10}}(\alpha, \delta_{REL}, \delta_{RAMT})$

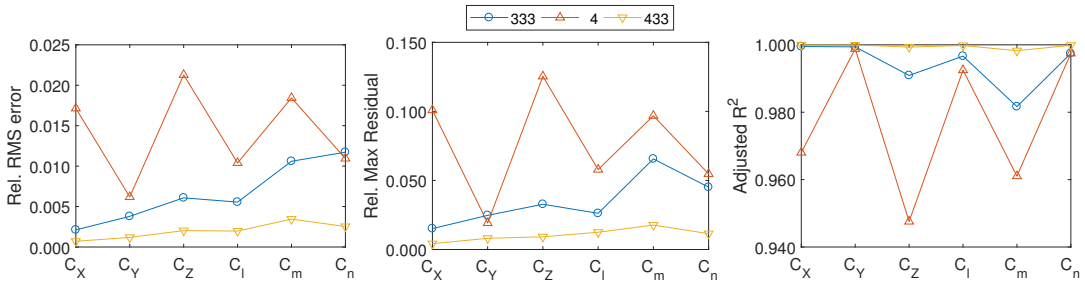


Figure A.13: RMS error, maximum residual, and coefficient of determination of $C_{*_{11}}(\alpha, \delta_{ROBLEF}, \delta_{RAMT})$

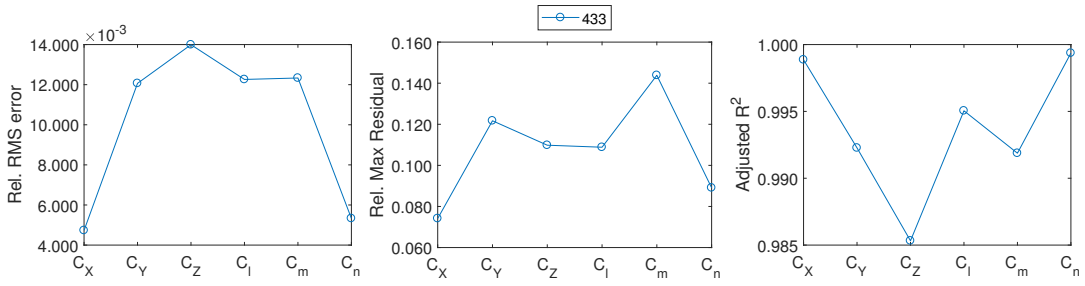


Figure A.14: RMS error, maximum residual, and coefficient of determination of $C_{*_{12}}(\alpha, \beta, \delta_{LSSD})$

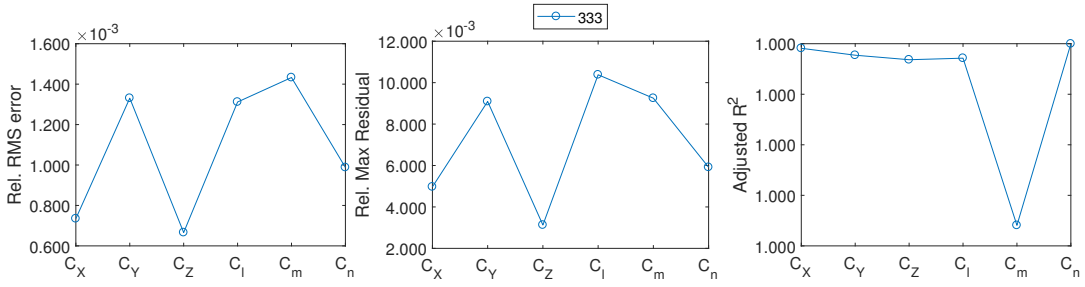


Figure A.15: RMS error, maximum residual, and coefficient of determination of $C_{*_{13}}(\alpha, \beta, \delta_{RIBLEF})$

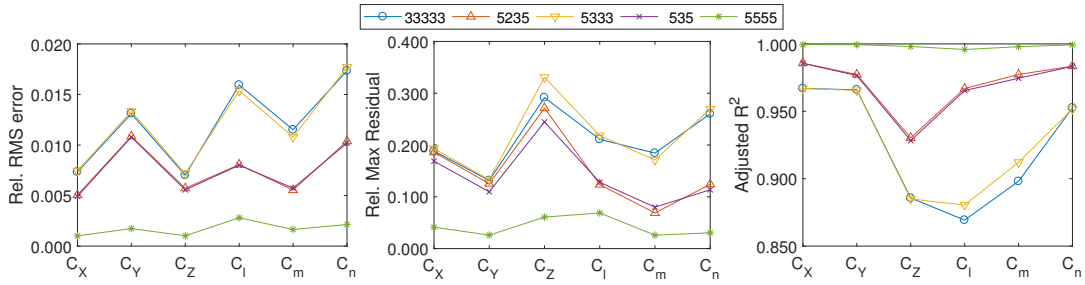


Figure A.16: RMS error, maximum residual, and coefficient of determination of $C_{*_{14}}(\alpha, \beta, \delta_{RIBLEF}, \delta_{ROBLEF}, M)$

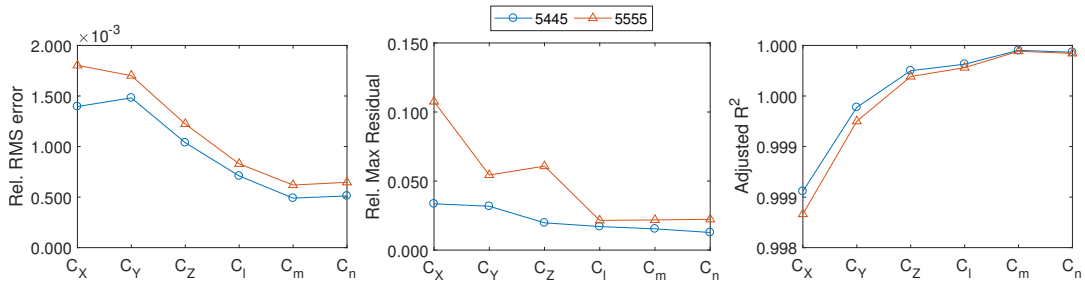


Figure A.17: RMS error, maximum residual, and coefficient of determination of $C_{*_{15}}(\alpha, \delta_{RSSD}, \delta_{REL}, M)$

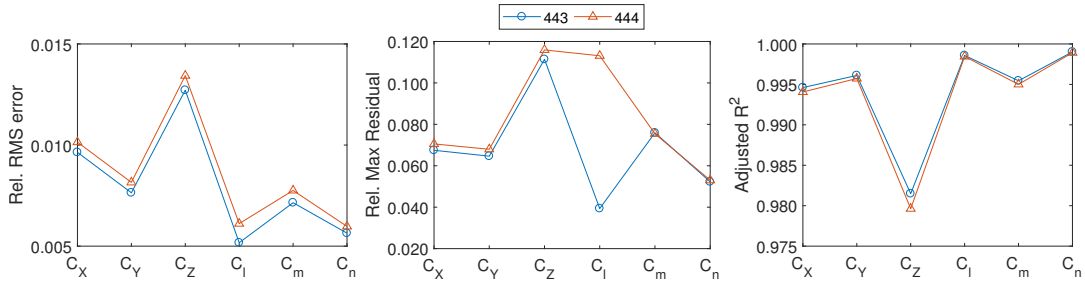


Figure A.18: RMS error, maximum residual, and coefficient of determination of $C_{*_{16}}(\alpha, \beta, \delta_{RAMT})$

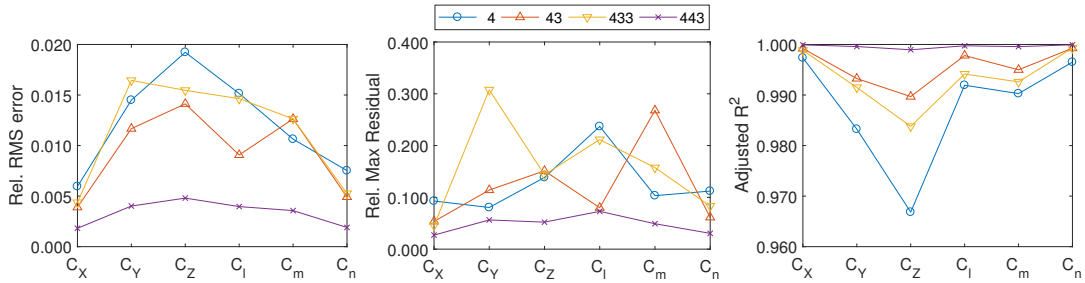


Figure A.19: RMS error, maximum residual, and coefficient of determination of $C_{*_{17}}(\alpha, \beta, \delta_{RSSD})$

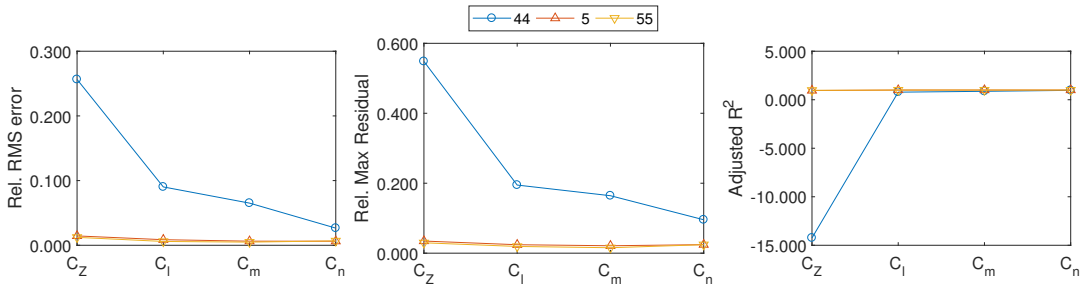


Figure A.20: RMS error, maximum residual, and coefficient of determination of $C_{*_{18}}(\alpha, M)$

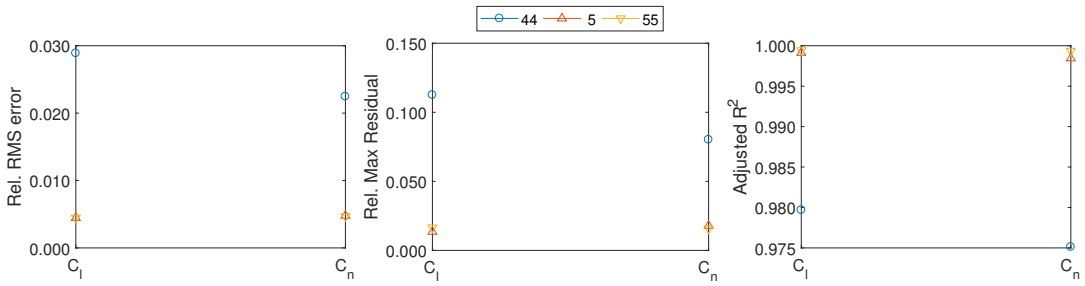


Figure A.21: RMS error, maximum residual, and coefficient of determination of $C_{*_{19}}(\alpha, M)$

A.3. Model Responses

This section contains the responses of the selected simplotope models, to a set of four chosen input combinations. The first input is none, where the ICE aircraft simply flies from the approximated trim condition. The second input is a combination of a symmetric doublet of the Elevons, followed by block inputs on both AMTs. Third, block inputs on the Differential Leading Edge Flaps and on the Spoiler Slot Deflectors. The last combination puts a block input on all possible effectors.

For all deflections, linear actuator models are used to provide a more realistic response to the inputs [13]. The low bandwidth actuator in Equation (A.1) is used for the Differential Leading Edge Flaps, all others have high bandwidth actuators as in Equation (A.2).

$$H_l = \frac{(18)(100)}{(s+18)(s+100)} \qquad \text{(A.1)} \qquad\qquad H_h = \frac{(40)(100)}{(s+40)(s+100)} \qquad \text{(A.2)}$$

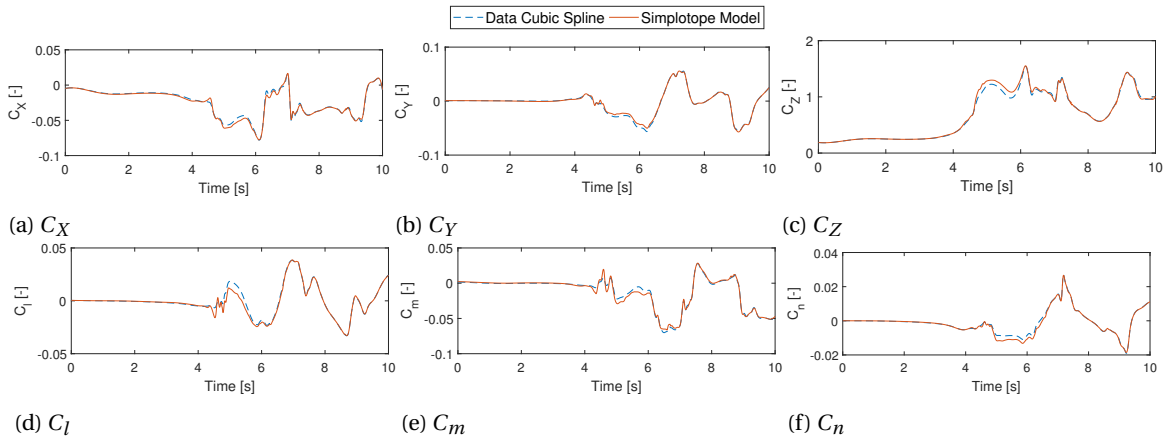
All inputs are given from a level trim state, flying at 183 m/s (600 ft/s), at an altitude of 500 ft. A summary of the results is given in Table A.2, which provides the RMS between the Simulink Model response, and the Simplotope model response, scaled by the range of the respones. The results show that the maximal deviation is 6.54%.

Table A.2: Relative RMS between the Simulink data and the Simplotope model during 10 seconds.

Input	C_X	C_Y	C_Z	C_l	C_m	C_n
None	2.14%	1.90%	2.82%	3.36%	2.75%	2.18%
Elevons / AMTs	1.75%	4.39%	2.50%	1.99%	1.23%	2.78%
DLEFs / SSDs	1.54%	2.10%	1.71%	6.54%	2.76%	1.68%
All	1.20%	1.56%	1.55%	6.41%	2.49%	3.03%

The coming subsections give figures containing comparisons between the two responses for every main coefficients, the aerodynamic states, and the inputs. It can be seen in all plots, that the simplotope models manage to follow the Simulink model's responses well. Moreover, the outputs show no major ripples or discontinuities, meaning that enforcing the first order continuity has its effect, and also that the distributed approach worked well.

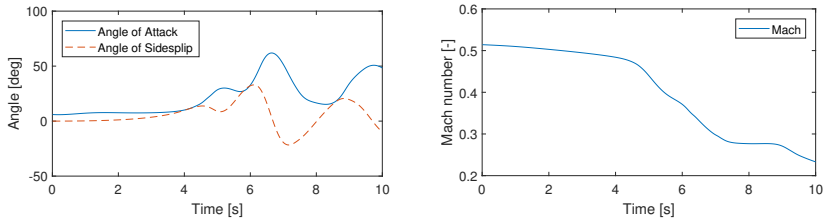
Zero Input



(a) C_X (b) C_Y (c) C_Z

(d) C_l (e) C_m (f) C_n

Figure A.22: Simplotope model responses compared with ICE's Simulink model after no inputs

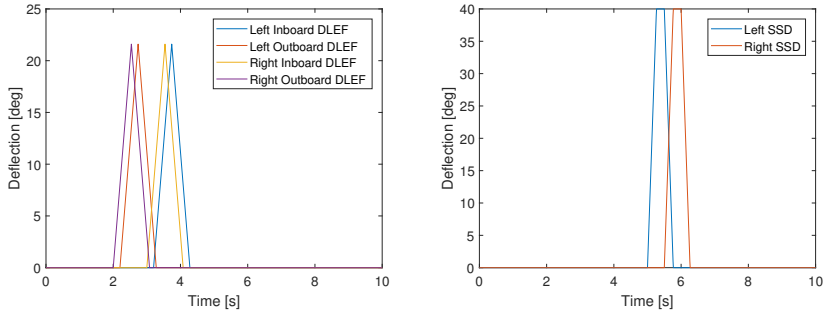


(a) Angle of Attack and Angle of Sideslip

(b) Mach number

Figure A.23: Aerodynamic states' responses after no inputs.

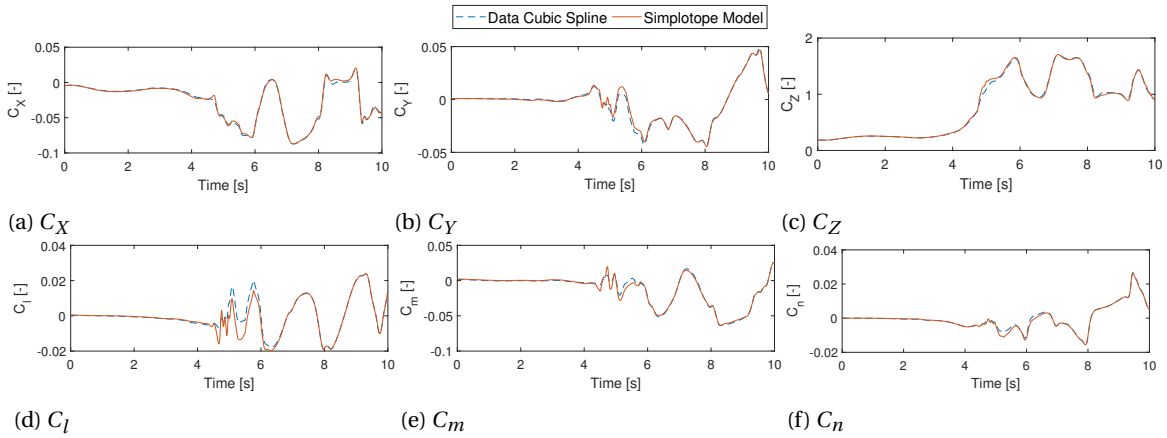
Leading Edge Flaps - Spoiler Slot Deflectors



(a) Leading Edge Flaps Inputs

(b) Spoiler Slot Deflectors inputs

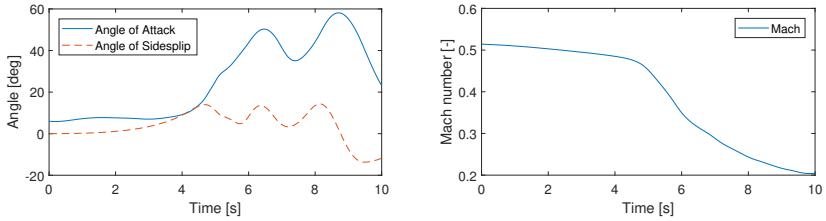
Figure A.24: DLEF and SSD inputs



(a) C_X (b) C_Y (c) C_Z

(d) C_l (e) C_m (f) C_n

Figure A.25: Simplotope model responses compared with ICE's Simulink model after DLEF and SSD inputs

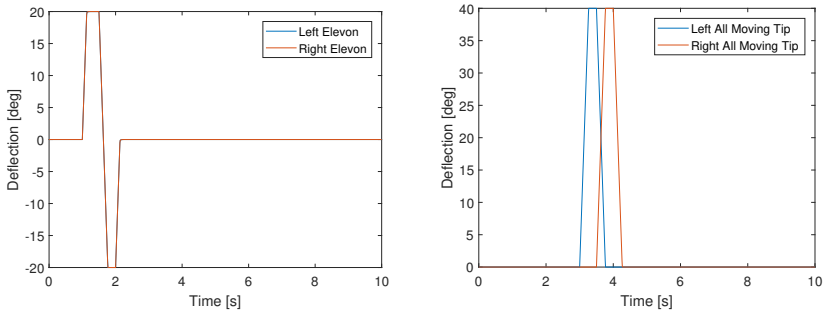


(a) Angle of Attack and Angle of Sideslip

(b) Mach number

Figure A.26: Aerodynamic states' responses to the DLEF and SSD inputs

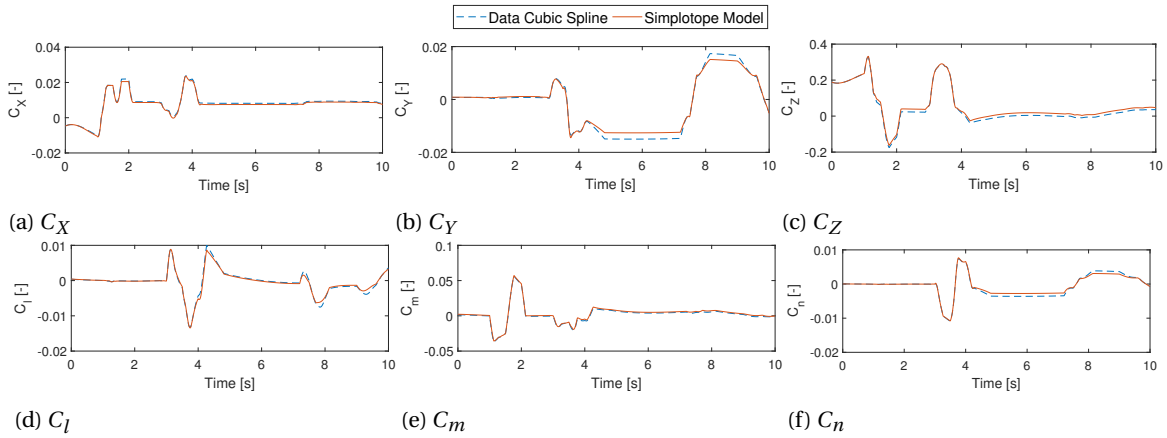
Elevons - All Moving Tips



(a) Elevons Inputs

(b) All Moving Tips inputs

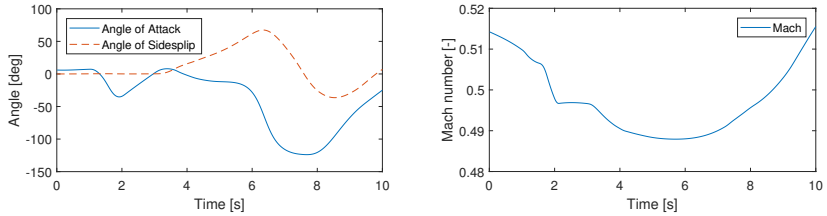
Figure A.27: Elevons and AMT inputs



(a) C_X (b) C_Y (c) C_Z

(d) C_l (e) C_m (f) C_n

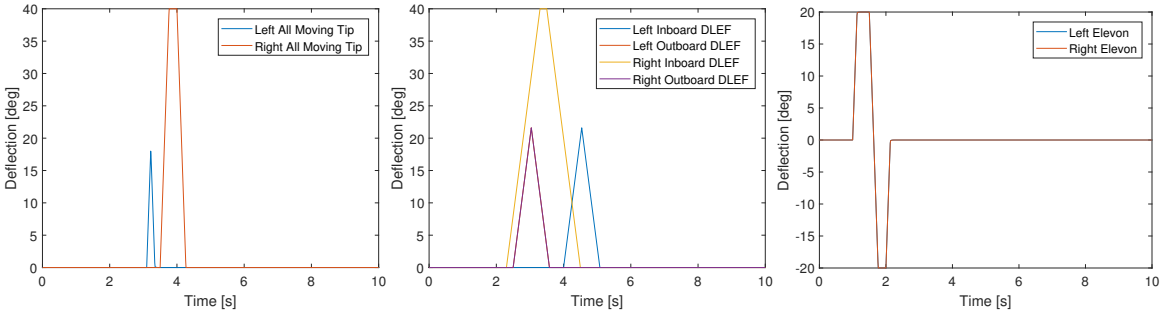
Figure A.28: Simplotope model responses compared with ICE's Simulink model after elevons and AMT inputs



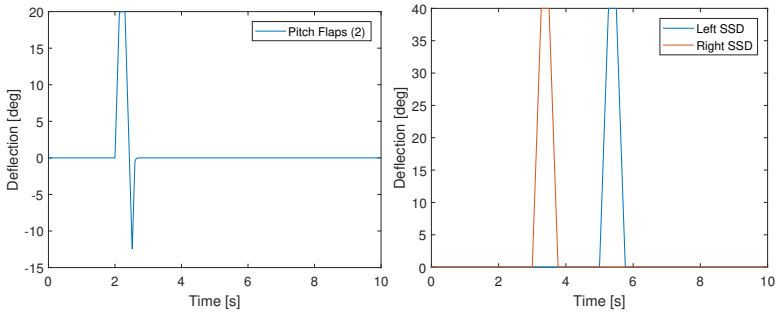
(a) Angle of Attack and Angle of Sideslip (b) Mach number

Figure A.29: Aerodynamic states' responses to the elevons and AMT inputs

All Deflected

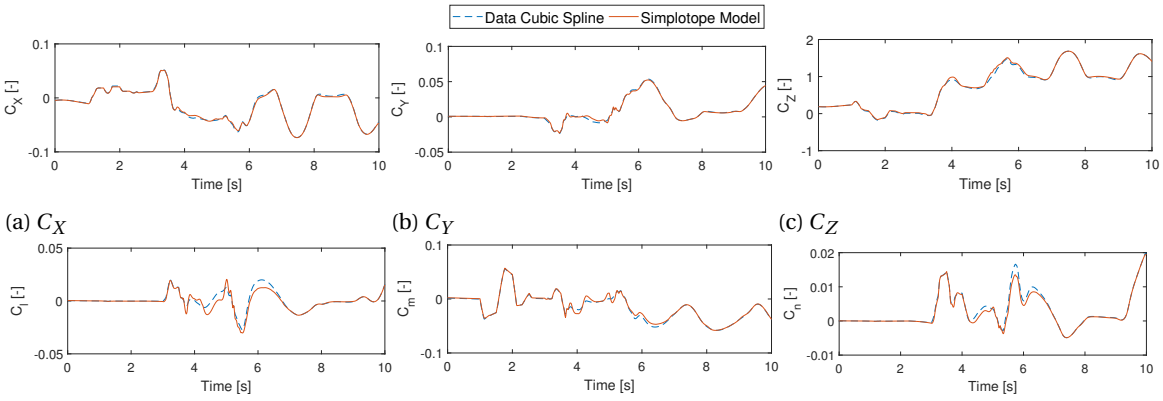


(a) All Moving Tips Inputs (b) Leading Edge Flaps inputs (c) Elevons inputs



(d) Pitch Flaps Inputs (e) Spoiler Slot Deflectors inputs

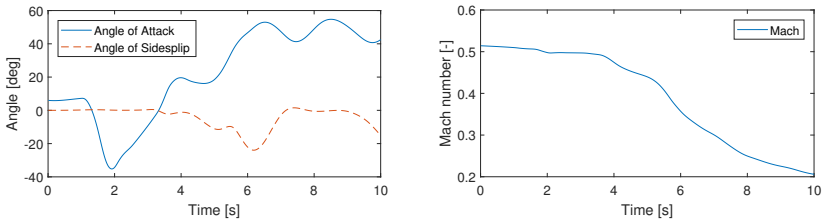
Figure A.30: All control deflectors' inputs



(a) C_X (b) C_Y (c) C_Z

(d) C_l (e) C_m (f) C_n

Figure A.31: Simplotope model responses compared with ICE's Simulink model after all effectors inputs



(a) Angle of Attack and Angle of Sideslip (b) Mach number

Figure A.32: Aerodynamic states' responses to all effectors inputs

B

Distributed Approaches

During this thesis project, several distributed approaches are investigated and tested. This tests initially included the capability to meet the global continuity constraints. The approach that finally has been chosen, is the Variable Splitting Alternating Direction Methods of Multipliers (VSADMM). The derivation for VSADMM with Simplotope B-Splines is given in Appendix B.1. Results of tests at various noise intensities, and comparisons with the global solution are shown in Appendix B.2. VSADMM was chosen because of ability to work completely in parallel.

Although Sequential ADMM showed good capabilities of converging to the optimal solution, it is not able to work in parallel. This derivation and test results are provided in Appendices B.3 and B.4 respectively. During early stages of this thesis project, as well as in the Preliminary Thesis Report in Part II, Dual Ascent was suggested as distributed solving method. However, early investigation showed that it did not function sufficiently in terms of meeting the global continuity constraints, as will be demonstrated in Appendix B.5.

The setup of a distributed approach involves the subdivision of all simplotopes into $P \leq J$ partitions. These partitions can consist of a single simplotope, but can also contain multiple simplotopes.

B.1. Derivation Variable Splitting ADMM

VSADMM was the parallel version of ADMM that was chosen to identify the ICE-aircraft's aerodynamic model. With VSADMM, coupling coefficients $\boldsymbol{z}_p$ are introduced for every partition p, such that $\mathbf{H}_p\boldsymbol{c}_p = \boldsymbol{z}_p$. Here, $\mathbf{H}_p$ are the columns of $\mathbf{H}$ that are being multiplied by $\boldsymbol{c}_p$. Since the global continuity conditions should still hold, the sum of all coupling coefficients has to equal zero. The problem that has to be optimized is given in Equation (B.1).

$$\begin{aligned} &\text{argmin}_{\boldsymbol{c}}\ \mathscr{J}(\boldsymbol{c}) = \textstyle\sum_{p=1}^{P} \frac{1}{2}||\mathbf{X}_p\boldsymbol{c}_p - \boldsymbol{Y}_p||_2^2 \\ &\text{subject to} \quad \mathbf{H}_p\boldsymbol{c}_p = \boldsymbol{z}_p \\ &\qquad\qquad \textstyle\sum_{p=1}^{P} \boldsymbol{z}_p = 0 \end{aligned} \tag{B.1}$$

The Lagrangian for this problem is formulated in Equation (B.2). The quadratic augmentation at the end, with $\rho > 0$, is for convergence purposes, as it lays an extra penalty on non-feasible solutions. When this problem is feasible, this term will equal zero, and thus has no effect on the final, optimal solution.

$$\mathscr{L} = \sum_{p=1}^{P} \frac{1}{2}||\mathbf{X}_p\boldsymbol{c}_p - \boldsymbol{Y}_p||_2^2 + \mathscr{I}_{\mathscr{Z}} + \lambda^T\left(\mathbf{H}_p\boldsymbol{c}_p - \boldsymbol{z}_p\right) + \frac{\rho}{2}||\mathbf{H}_p\boldsymbol{c}_p - \boldsymbol{z}_p||_2^2 \tag{B.2}$$

The constraints on the coupling coefficients $\boldsymbol{z}$ are changed into the indicator function $\mathscr{I}_{\mathscr{Z}}$. This indicator will be zero if the solution is part of the set $\mathscr{Z}$, and infinity when not. This convex set is defined as [6]:

$$\mathscr{Z} = \left\{ [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_P] : \sum_{p=1}^{P} \boldsymbol{z}_p = 0 \right\}$$

First taking the derivative w.r.t. the coefficients in Equation (B.3), and setting it equal to zero results in optimal values of the coefficients $\boldsymbol{c}_p$ as described in Equation (B.4). This can be done, since the problem is

fully decoupled per partition w.r.t. the coefficients. This means that the derivative will not include coefficients of any other partition.

$$\nabla_{\boldsymbol{c}_p}\mathscr{L} = \left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)\boldsymbol{c}_p - \mathbf{X}_p^T\boldsymbol{Y}_p + \lambda_p^T\mathbf{H}_p + \rho\mathbf{H}_p^T\left(\mathbf{H}_p\boldsymbol{c}_p - \boldsymbol{z}_p\right) = 0 \tag{B.3}$$

$$\boldsymbol{c}_p = \left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\left(\mathbf{X}_p^T\boldsymbol{Y}_p + \rho\mathbf{H}_p^T\boldsymbol{z}_p - \mathbf{H}_p^T\lambda\right) \tag{B.4}$$

Updating the coupling coefficients is more complex, as there are constraints between the partitions put on these. The optimal iteration is found in two steps. At first, its constraints are ignored, and simply the derivative is taken as in Equations (B.5) and (B.6).

$$\nabla_{\boldsymbol{z}_p}\mathscr{L} = -\lambda_p - \rho\left(\mathbf{H}_p\boldsymbol{c}_p - \boldsymbol{z}_p\right) = 0 \tag{B.5}$$

$$\boldsymbol{z}_p = \mathbf{H}_p\boldsymbol{c}_p + \frac{\lambda_p}{\rho} \tag{B.6}$$

Now this estimation has to be corrected for the constraints that are put on the coupling constraints. This means that the sum over all partitions has to equal zero, in order to have global smoothness. A division was made between two types of constraints: internal constraints over edges within a partition, of which the constraint indices are collected in the set $\mathscr{H}_p$; and the external constraints over edges with the partitions' neighbors, whose indices are collected in $\mathscr{H}_{p,n}$. These are both subsets of the set of indices for all constraints $\mathscr{H} = (1,\ldots,R)$.

For every constraint j, the correction is shown in Equation (B.7) [2], where m_j is the number of partitions that appears in that constraint. If j is internal, m_j is 1, and if j is external, m_j is 2. The latter is true, because knowledge of the smoothness constraints gives that a continuity constraint is always between exactly two simplotopes.

$$\boldsymbol{z}_{p,j} = \left(\mathbf{H}_{p,j}\boldsymbol{c}_p + \frac{\lambda_{p,j}}{\rho}\right) - \frac{1}{m_j}\left(\sum_{i=1}^{P}\mathbf{H}_{i,j}\boldsymbol{c}_i + \frac{\lambda_{i,j}}{\rho}\right), \quad \forall\, j \in \left(\mathscr{H}_p \cup \mathscr{H}_{p,n}\right) \tag{B.7}$$

Summing all coupling coefficients over the partitions confirms that this update inherently enforces the constraints put on the set of coupling coefficients.

What results is a fully distributed, parallel, and iterative solver. The steps for every partitions p are given in Equations (B.8) to (B.10). The coupling coefficient update is defined for both possibilities of m_j. For internal edges, the update reduces to 0. Coupling coefficients for constraints in which partition p does not appear, the coupling coefficient is not updated, since this is not relevant.

$$\boldsymbol{z}_{p,j}^{k+1} = \begin{cases} \left(\mathbf{H}_{p,j}\boldsymbol{c}_p^k + \frac{\lambda_{p,j}^k}{\rho}\right) - \frac{1}{2}\left(\sum_{i=1}^{P}\mathbf{H}_{i,j}c_i^k + \frac{\lambda_{i,j}^k}{\rho}\right), & \forall\, j \in \mathscr{H}_{p,n} \\ 0, & \forall\, j \in \mathscr{H}_p \end{cases} \tag{B.8}$$

$$\boldsymbol{c}_p^{k+1} = \left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\left(\mathbf{X}_p^T\boldsymbol{Y}_p + \rho\mathbf{H}_p^T\boldsymbol{z}_p^{k+1} - \mathbf{H}_p^T\lambda_p^k\right) \tag{B.9}$$

$$\lambda_p^{k+1} = \lambda_p^k + \rho\left(\mathbf{H}_p\boldsymbol{c}_p^{k+1} - \boldsymbol{z}_p^{k+1}\right) \tag{B.10}$$

Note that in Equation (B.9), the terms $\mathbf{X}_p$, $\boldsymbol{Y}_p$, and $\mathbf{H}_p$ do not change in the process of iterating. This means that two matrices can be pre-computed. First the term $\left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\left(\mathbf{X}_p^T\boldsymbol{Y}\right)$, which can be seen as an initial solution, results in a $\hat{d} \times 1$ vector per partition. The other one is $\left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\mathbf{H}_p^T$, which results in a $\hat{d} \times R$ matrix that needs to be multiplied with $\rho\boldsymbol{z}_p - \lambda_p$ to get the coefficients estimate. The resulting equation is given in Equation (B.11).

$$\boldsymbol{c}_p^{k+1} = \tilde{\boldsymbol{c}}_{p,init} + \left[\left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\mathbf{H}_p^T\right]\left(\rho\boldsymbol{z}_p^{k+1} - \lambda_p^k\right) \tag{B.11}$$

There is a possibility of changing this procedure slightly in order to speed up convergence to a final solution. Although this solution is of high quality, the validation results differed from the global solution. In

this case, the update of the coupling coefficients is changed by removing the dual terms, and the next estimate of the coupling coefficient is simply the average both sides of the edge. The calculation is given in Equation (B.12).

$$\boldsymbol{z}_p^{k+1} = \left(\mathbf{H}_p \boldsymbol{c}_p^k\right) - \frac{1}{2}\left(\sum_{p=1}^{P} \mathbf{H}_i \boldsymbol{c}_i^k\right) \tag{B.12}$$

B.2. Variable Splitting ADMM Tests

In this section, several tests of the Variable Splitting ADMM (VSADMM) algorithm are shown. In order to assess the performance of the algorithm in different cases, multiple combinations of layers are made. Noise has been added artificially in order to compare performance in different environments. For a noise intensity of k%, the noise is generated by using a uniform distribution $v = U(-0.5, 0.5)$, and Equation (B.13) [4].

$$v(k) = 0.01 \cdot k \cdot (\max \boldsymbol{Y} - \min \boldsymbol{Y}) \cdot v \tag{B.13}$$

Four different noise levels are used for comparison: 0%, which means the original dataset, 1%, 10%, and 100%. Noise has only been added to the identification data, whereas the validation data is kept clean. Two different starting values for the penalty factor have been used in the VSADMM algorithm: $\rho = 1.0$ and $\rho = 0.01$. Although the adaptive penalty factor is in place, an assessment has been made to what extent the initial choice may influence the results.

Four different models are being used to demonstrate the working of VSADMM, one for every dimension of the ICE Spline model: 2, 3, 4, and 5. Submodel $C_{X_1}(\alpha, M)$ is demonstrated in Appendix B.2.1, $C_{X_2}(\alpha, \beta, M)$ in Appendix B.2.2, $C_{X_5}(\alpha, M, \delta_{LSSD}, \delta_{LEL})$ in Appendix B.2.3, and $C_{X_{14}}(\alpha, \beta, M, \delta_{RIBLEF}, \delta_{ROBLEF})$ in Appendix B.2.4.

B.2.1. 2-Dimensional

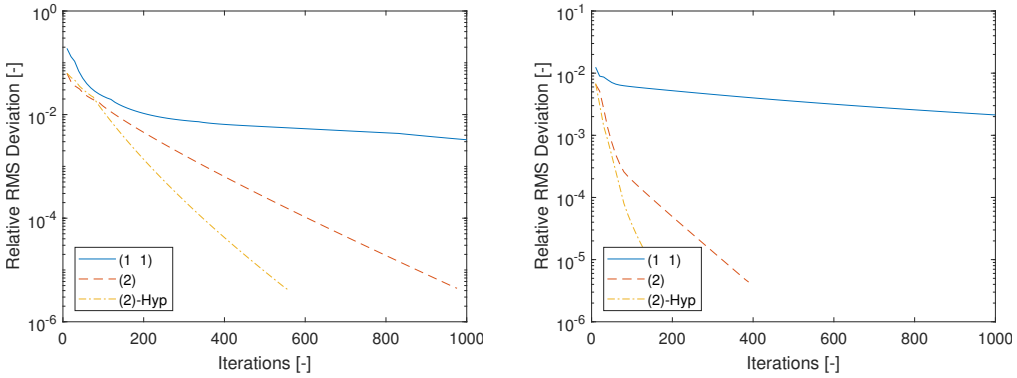
The used 2-dimensional model is $C_{X_1}(\alpha, M)$, depending on Angle of Attack and Angle of Sideslip. The details of the different combinations of layers are shown in Table B.1. The third combination has the same triangulation as the second one, but differs in partitions size. This is done on the one hand to show that it also works in bigger size partitions, and on the other hand to see if this is beneficial in terms of convergence properties.

Table B.1: Combinations of $C_{X_1}(\alpha, M)$ used for noise comparisons.

Layer Division	Dimensions	Continuity	Simplotopes	Partitions
(1,1)	(5,5)	(1,1)	120	120
(2)	(5)	(1)	240	240
(2)	(5)	(1)	240	120

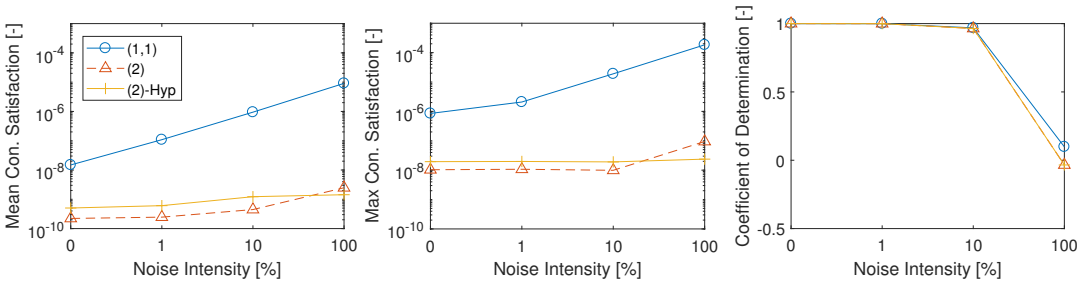
The results show that both pure simplex models have better performance if it comes to convergence in Figure B.1 and constraint satisfaction in Figure B.2. The hypercube partitioning is slightly worse when it comes to constraint satisfaction, but better when it comes to convergence, and the coefficients of determination are equal, as expected. It can also be seen in Figure B.3, that the validation performance of the (1,1)-model is slightly off compared to the global solution, where the simplex models in Figures B.4 and B.5 are equal to their global optimum.

A comparison between the different layer divisions shows that the simplex model have a worse fitness to the identification data, but this becomes beneficial at high noise intensities. Where the RMS residual is close 0.3 for the simplotope, it is below 0.2 for the simplex models. When it comes to the maximum residual, the simplotope model has a 4 times higher maximum than the simplex model.

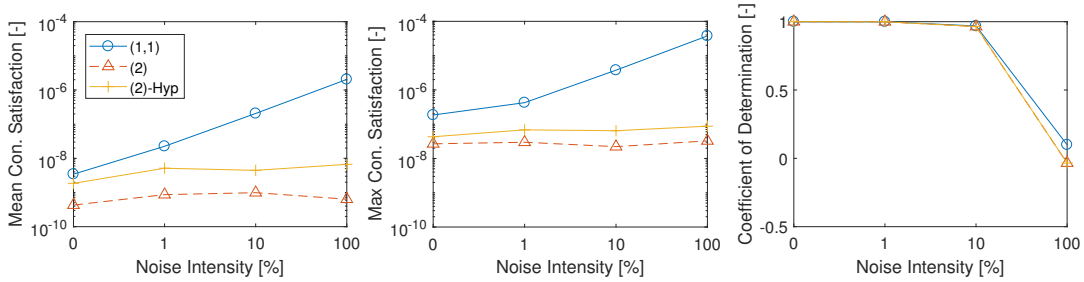


(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.1: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_1}(\alpha, M)$.

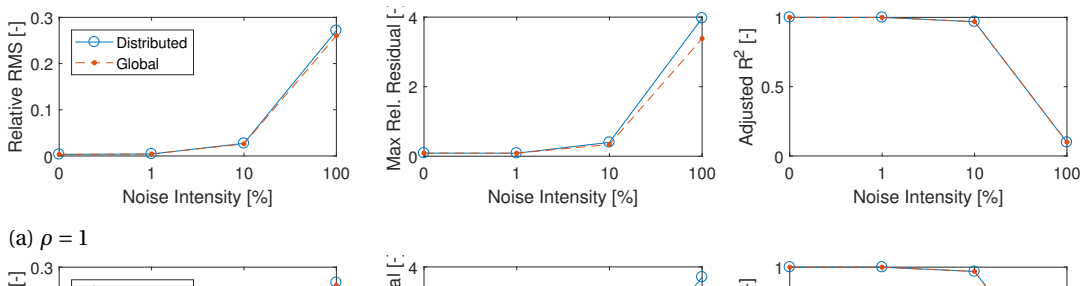


(a) $\rho = 1$

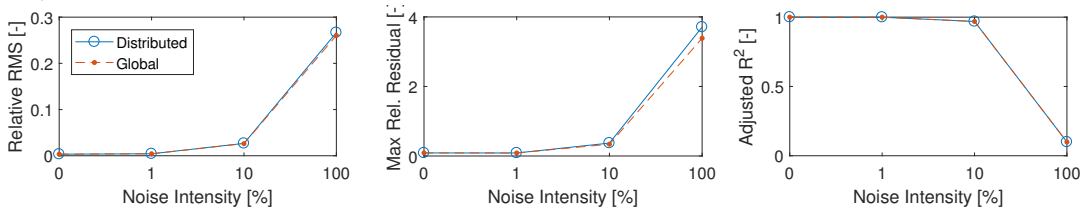


(b) $\rho = 0.01$

Figure B.2: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_1}(\alpha, M)$ at various noise intensities.

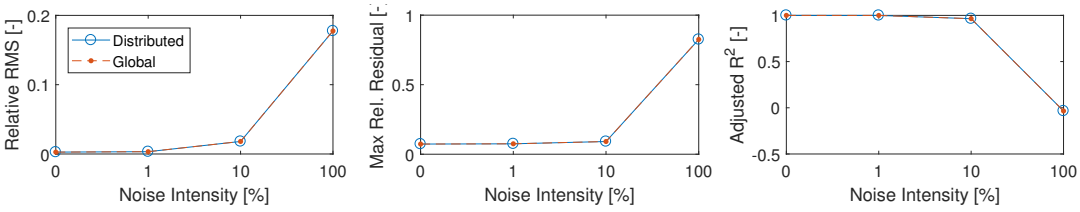


(a) $\rho = 1$

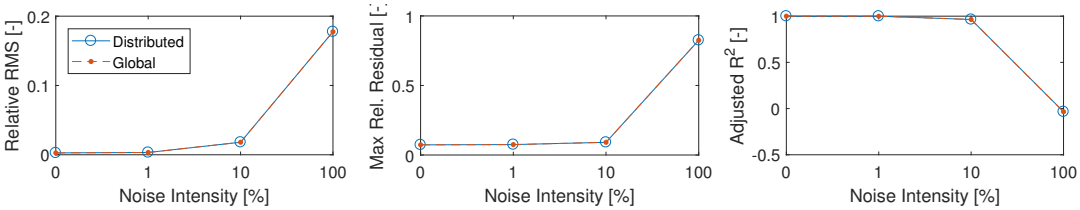


(b) $\rho = 0.01$

Figure B.3: Validation results and coefficient of determination of the 2D $(1,1)$-spline.

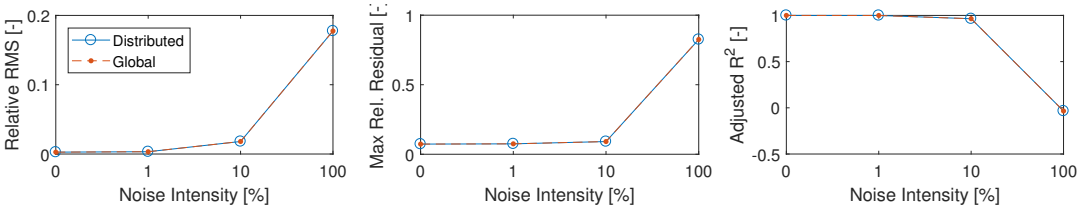


(a) $\rho = 1$

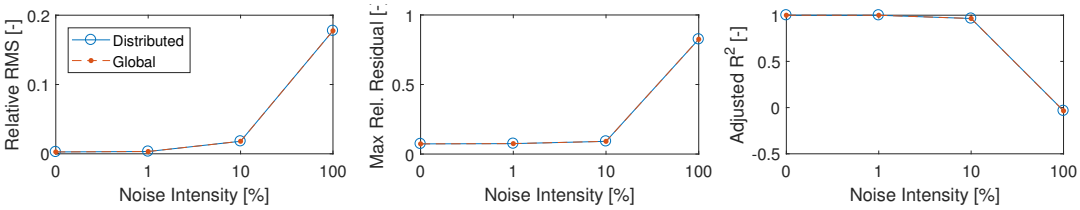


(b) $\rho = 0.01$

Figure B.4: Validation results and coefficient of determination of the 2D (2)-spline.



(a) $\rho = 1$



(b) $\rho = 0.01$

Figure B.5: Validation results and coefficient of determination of the 2D (2)-spline with Hypercube partitions.

B.2.2. 3-Dimensional

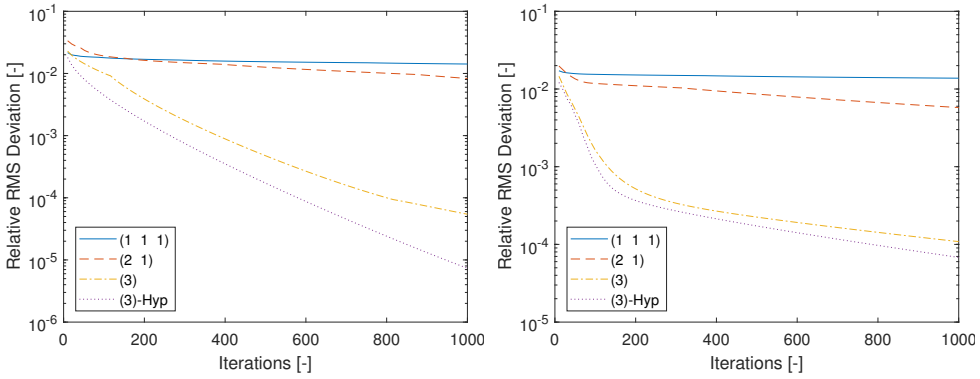
The 3-dimensional model that is analyzed is $C_{X_2}(\alpha, \beta, M)$, depending on Angle of Attack, Angle of Sideslip, and Mach number. The combinations are shown in Table B.2. The $(2,1)$-simplotope puts α and β together in one layer, while the (3)-simplotope contains all variables. The last model differs in the partitioning of the simplotopes: 6 simplotopes, which are all in a single hypercube, are put together in one partition, in order to assess the differences its performance.

Table B.2: Combinations of $C_{X_2}(\alpha, \beta, M)$ used for noise comparisons.

Layer Division	Dimensions	Continuity	Simplotopes	Partitions
$(1,1,1)$	$(5,5,5)$	$(1,1,1)$	144	144
$(2,1)$	$(5,5)$	$(1,1)$	296	296
(3)	(5)	(1)	864	864
(3)	(5)	(1)	864	144

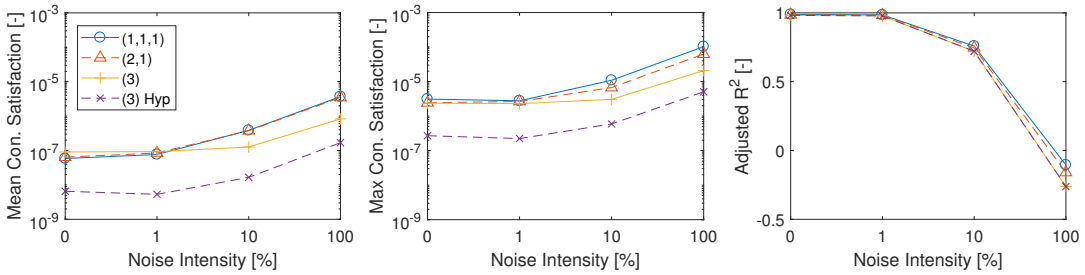
The convergence towards the global solution is shown in Figure B.6, and it shows that the simplotope models have worse convergence towards the global optimum. The hypercube partitionining has a better convergence rate than the individual simplex partitions. Figure B.7 shows that all constraints are well satisfied, and that the simplotope models perform better at low penalty factor, where the simplex models benefit from higher penalty factors. The fitness to the identification data becomes better, the more layers are used.

At 100% noise intensity the validation performance of the simplotope models in Figures B.8 and B.9 are sligthly off the global results. The simplex models in Figures B.9 and B.9 are nearly equal. The simplotope models perform better than at low noise intensities, but as the noise in the measurements increases, the simplex models remain more stable in terms of validation performance. This suggests that the simplotopes may overfit the identification models.

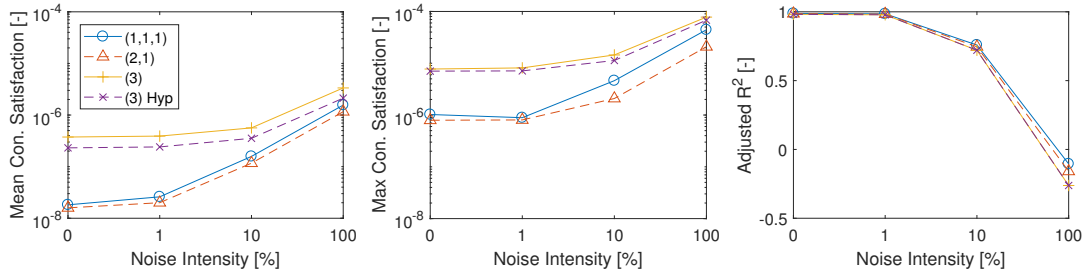


(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.6: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_2}(\alpha, \beta, M)$

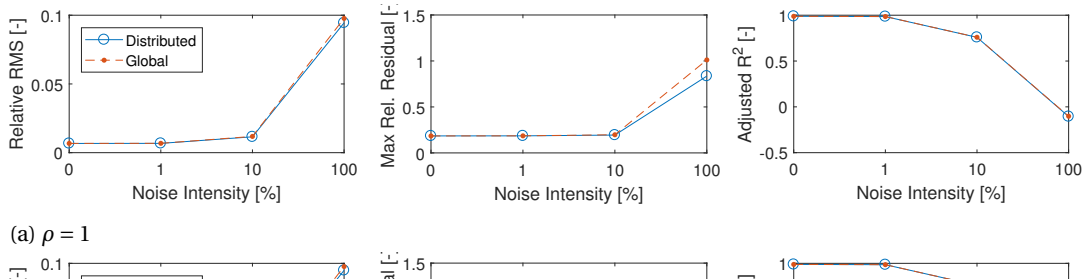


(a) $\rho = 1$

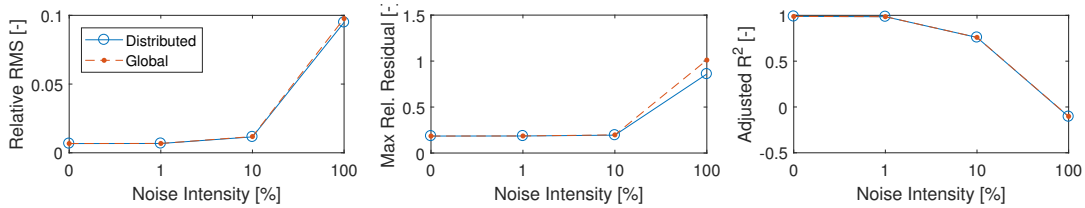


(b) $\rho = 0.01$

Figure B.7: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_2}(\alpha, \beta, M)$ at various noise intensities.

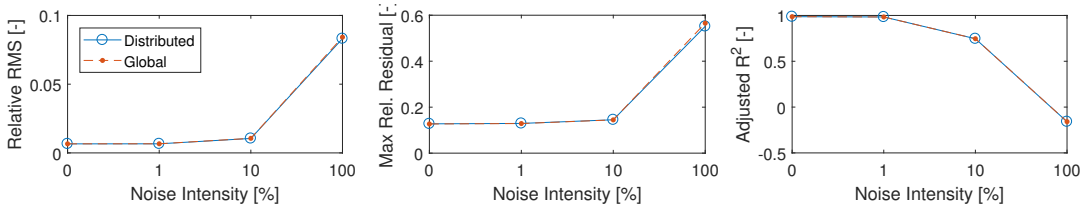


(a) $\rho = 1$

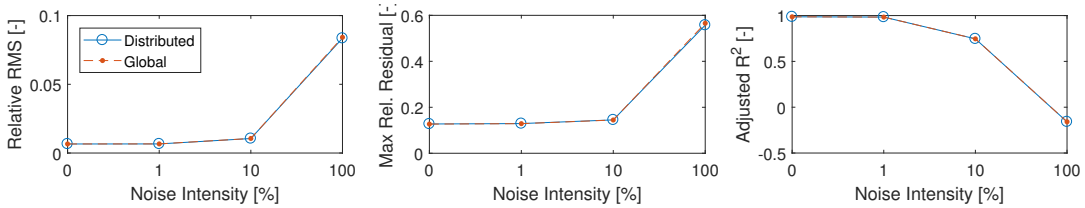


(b) $\rho = 0.01$

Figure B.8: Validation results and coefficient of determination of the 3D $(1,1,1)$-spline.

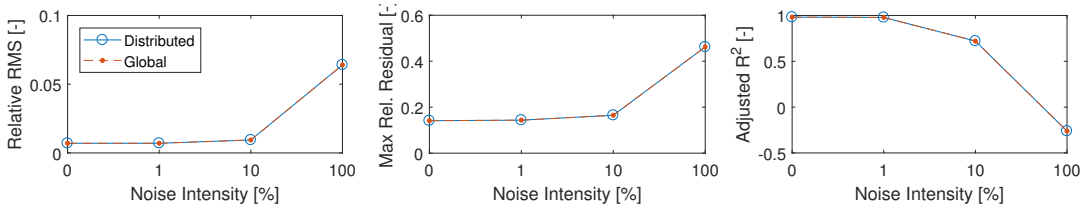


(a) $\rho = 1$

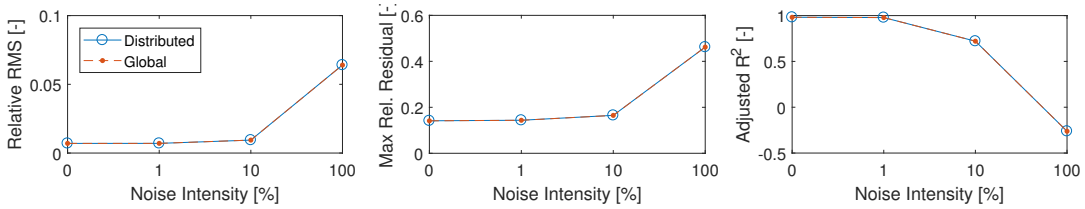


(b) $\rho = 0.01$

Figure B.9: Validation results and coefficient of determination of the 3D $(2,1)$-spline.

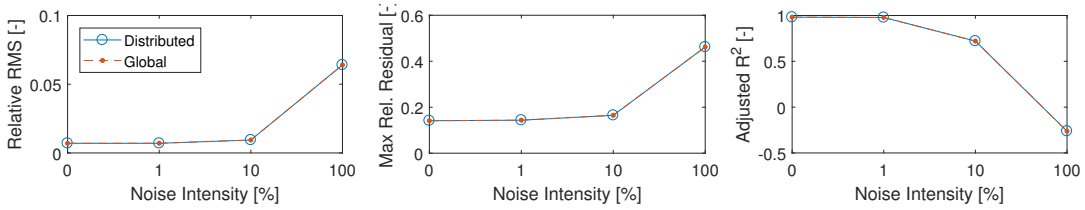


(a) $\rho = 1$

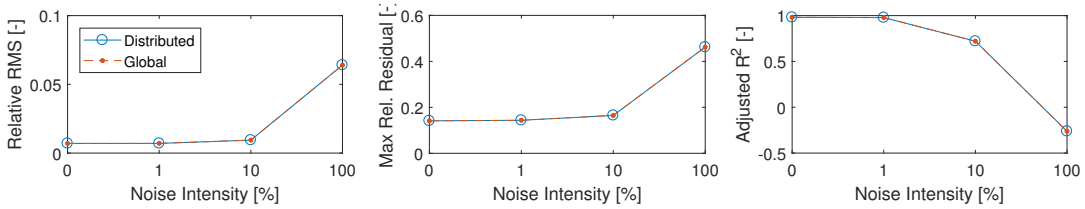


(b) $\rho = 0.01$

Figure B.10: Validation results and coefficient of determination of the 3D (3)-spline.



(a) $\rho = 1$



(b) $\rho = 0.01$

Figure B.11: Validation results and coefficient of determination of the 3D (3)-spline with Hypercube partitions.

B.2.3. 4-Dimensional

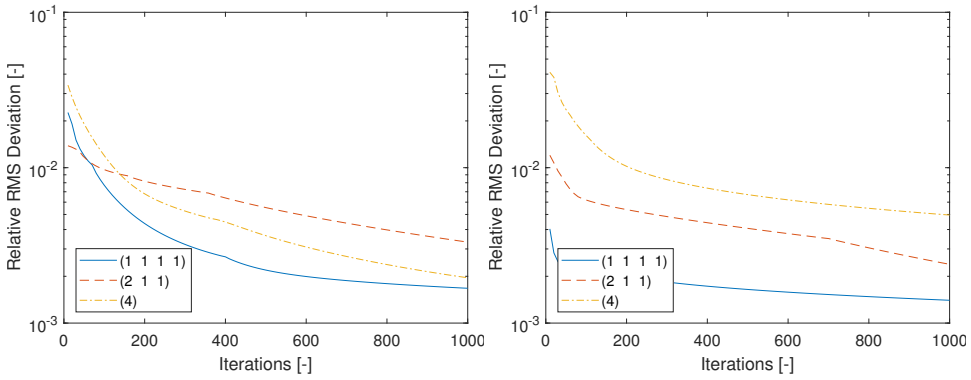
The details of the 4-dimensional model tests are given in Table B.3. Coefficient $C_{X_5}(\alpha, M, \delta_{LSSD}, \delta_{LEL})$ is dependent on the Angle of Attack, Mach number, the deflection of the left SSD, and its interaction with the left elevon. The $(2,1,1)$ model groups the aerodynamic states α and M.

Table B.3: Combinations of $C_{X_5}(\alpha, M, \delta_{LSSD}, \delta_{LEL})$ used for noise comparisons.

Layer Division	Dimensions	Continuity	Simplotopes	Partitions
$(1,1,1,1)$	$(4,4,3,3)$	$(1,1,1,1)$	160	160
$(2,1,1)$	$(4,3,3)$	$(1,1,1)$	320	320
(4)	(4)	(1)	1440	1440

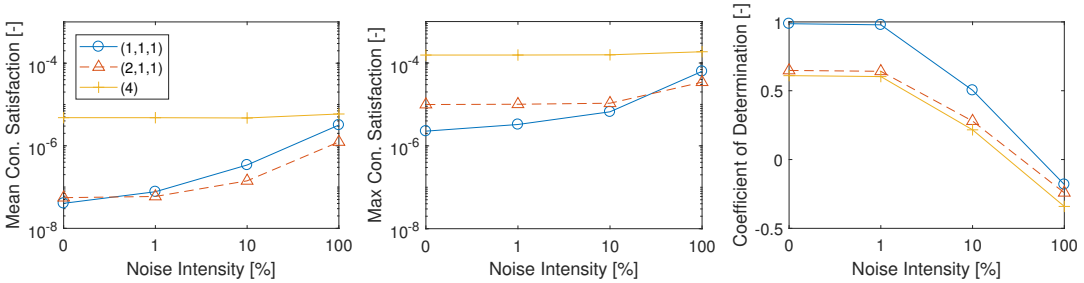
All models slowly converge to the optimal solution, as shown in Figure B.12, but the difference are still relatively high as compared to the other dimensional models. The constraint satisfaction of the simplex model in Figure B.13 can be considered bad. Especially for the low penalty factor in Figure B.13b it can be seen that inequalities of close to 10^{-3} occur. However, an improvement is shown in Figure B.12b, which indicates that a higher penalty factor is beneficial in this case. More iterations can improve the estimation as well.

Where the validation results in Figures B.14 and B.15 are close to the global validation results, the results of the pure simplex model in Figure B.16 is slightly off. However, at high noise intensities, it performs substantially better than the simplotope models.

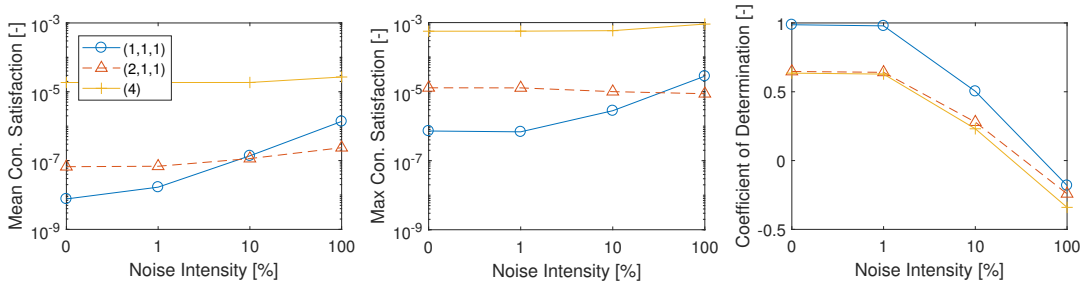


(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.12: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_5}(\alpha, M, \delta_{LSSD}, \delta_{LEL})$.

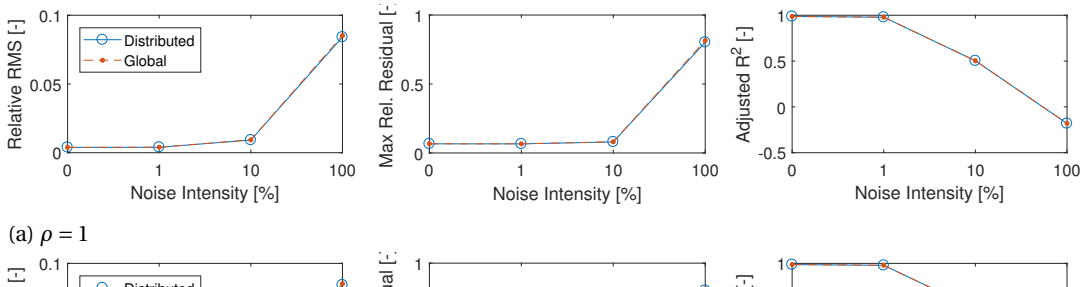


(a) $\rho = 1$

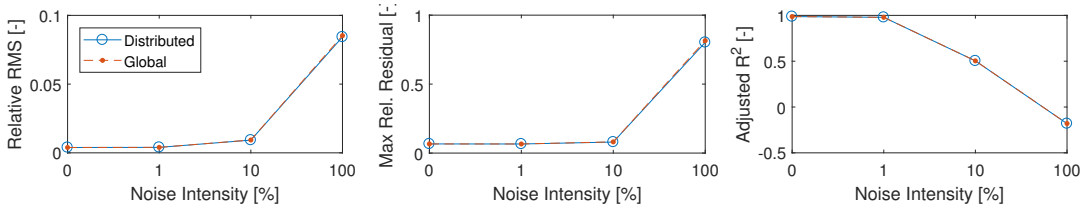


(b) $\rho = 0.01$

Figure B.13: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_5}(\alpha, M, \delta_{LSSD}, \delta_{LEL})$ at various noise intensities.

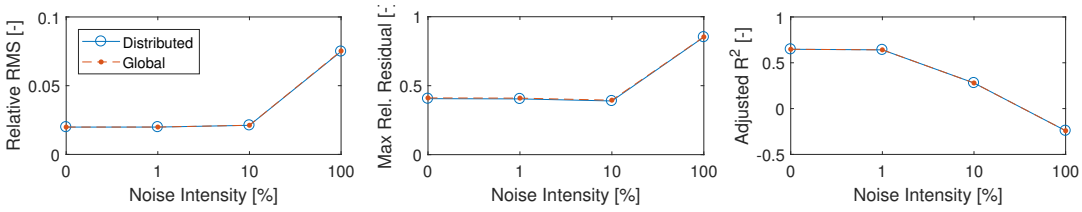


(a) $\rho = 1$

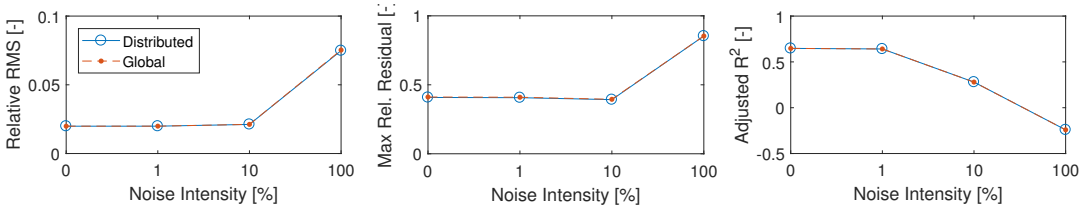


(b) $\rho = 0.01$

Figure B.14: Validation results and coefficient of determination of the 4D $(1,1,1,1)$-spline.

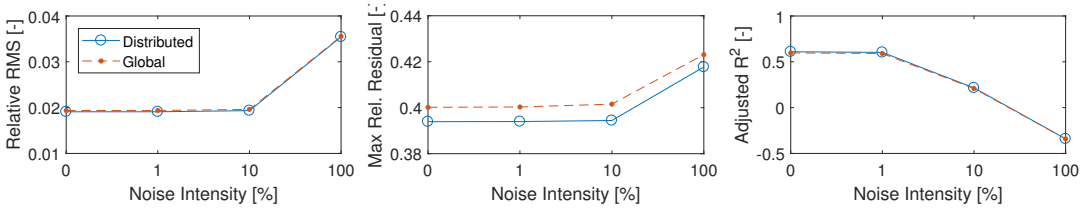


(a) $\rho = 1$

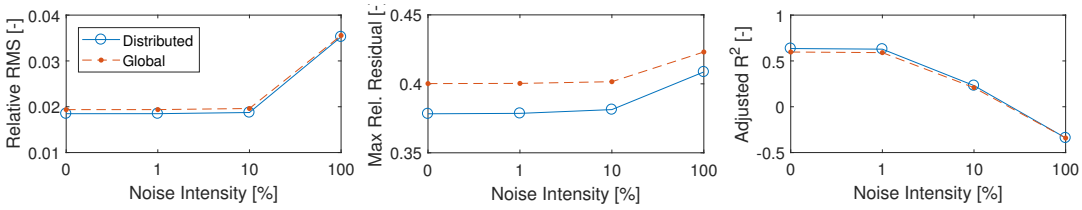


(b) $\rho = 0.01$

Figure B.15: Validation results and coefficient of determination of the 4D $(2,1,1)$-spline.



(a) $\rho = 1$



(b) $\rho = 0.01$

Figure B.16: Validation results and coefficient of determination of the 4D (4)-spline.

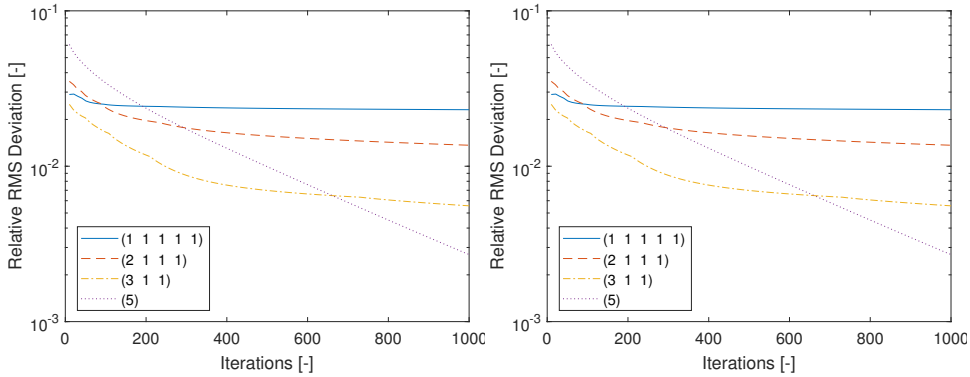
B.2.4. 5-Dimensional

Model $C_{X_{14}}(\alpha,\beta,M,\delta_{RIBLEF},\delta_{ROBLEF})$ contains the interaction of the three aerodynamic states, and the right inboard leading edge flap, and the right outboard leading edge flap. The combinations are shown in Table B.4. The $(2,1,1,1)$ groups α and β in the first layer, and the $(3,1,1)$ adds the Mach number.

Table B.4: Combinations of $C_{X_{14}}(\alpha,\beta,M,\delta_{RIBLEF},\delta_{ROBLEF})$ used for noise comparisons.

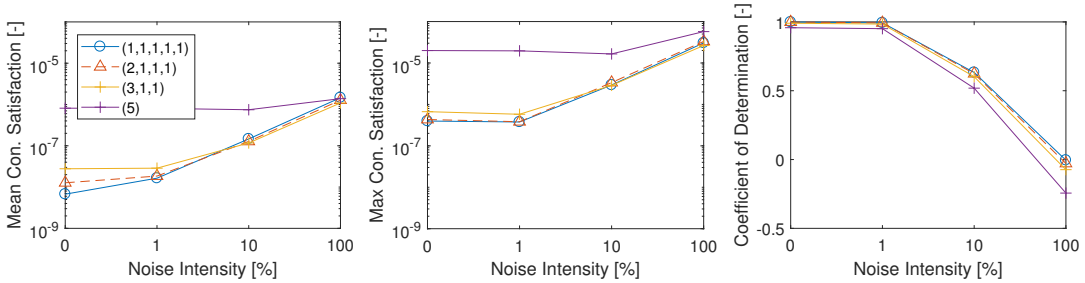
Layer Division	Dimensions	Continuity	Simplotopes	Partitions
$(1,1,1,1,1)$	$(4,4,4,4,4)$	$(1,1,1,1,1)$	5	5
$(2,1,1,1)$	$(4,4,4,4)$	$(1,1,1,1)$	10	10
$(3,1,1)$	$(4,4,4)$	$(1,1,1)$	30	30
(5)	(4)	(1)	600	600

All models remain at a few percents of the global solution, with only the $(3,1,1)$- and (5)-models getting below 1%. However, it is the (5)-model that has the worst constraint satisfaction in Figure B.18, with both penalty factors, while all other constraints are well satisfied. The constraint satisfaction of the (5)-model improved with a factor of 10 when ρ was improved from 0.01 to 1, pointing out that an increasing penalty factor works beneficially. For all but the (5) model, in Figures B.19 to B.21, the validation results are relatively close to the global optimum. However, the pure simplex model again shows best performance when the noise intensity gets higher. In this case, the difference between the 0% and 100% noise intensity is almost none.

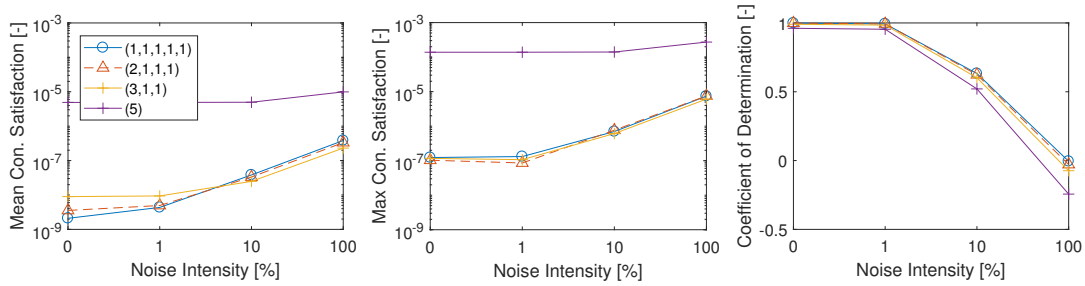


(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.17: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_{14}}(\alpha,\beta,M,\delta_{RIBLEF},\delta_{ROBLEF})$.

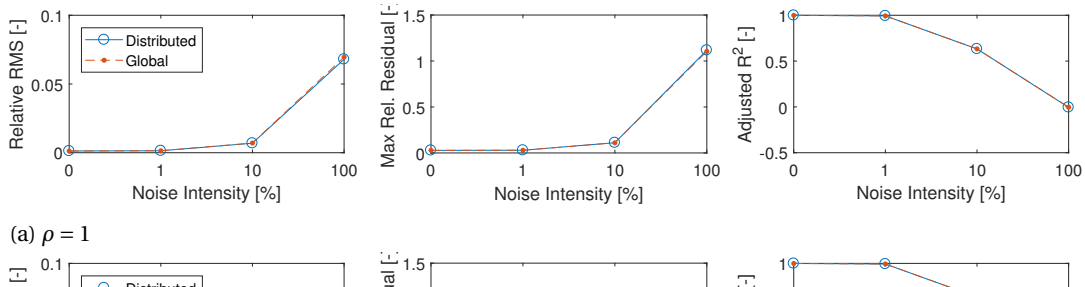


(a) $\rho = 1$

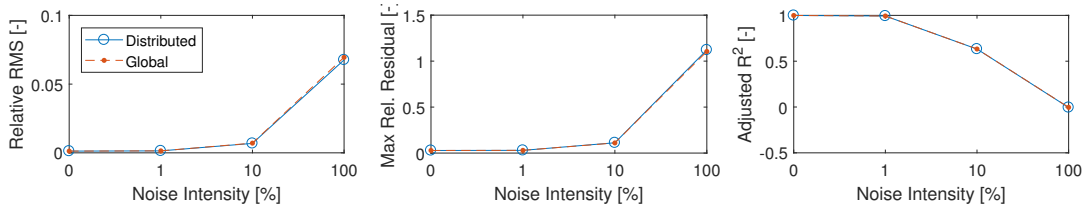


(b) $\rho = 0.01$

Figure B.18: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_{14}}(\alpha, \beta, M, \delta_{RIBLEF}, \delta_{ROBLEF})$ at various noise intensities.

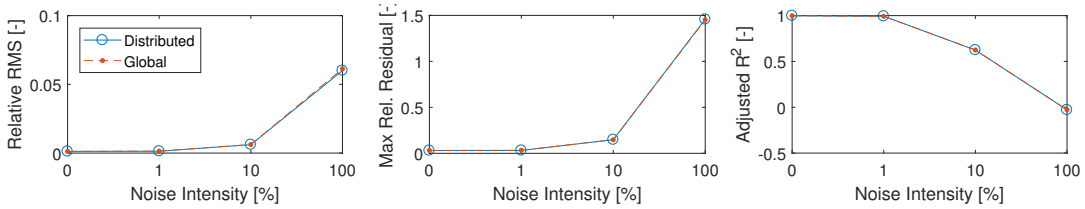


(a) $\rho = 1$

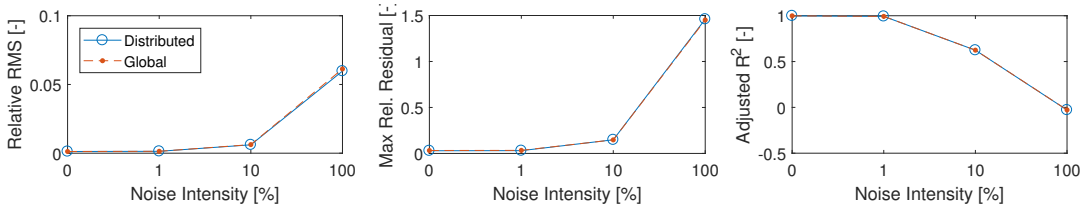


(b) $\rho = 0.01$

Figure B.19: Validation results and coefficient of determination of the 5D $(1,1,1,1,1)$-spline.

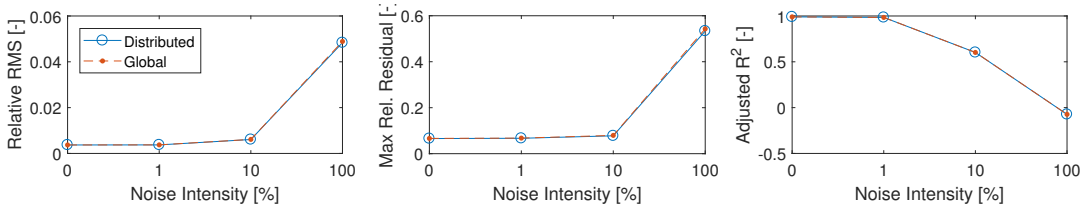


(a) $\rho = 1$

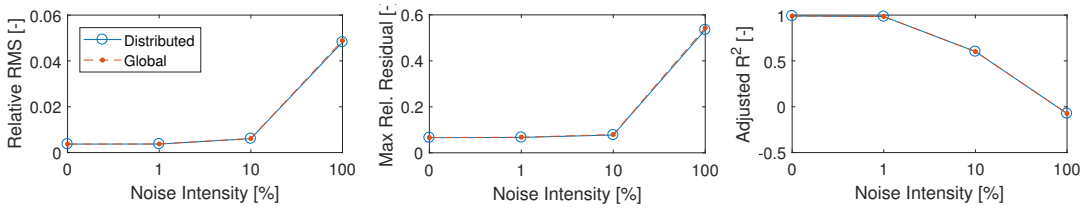


(b) $\rho = 0.01$

Figure B.20: Validation results and coefficient of determination of the 5D $(2,1,1,1)$-spline.

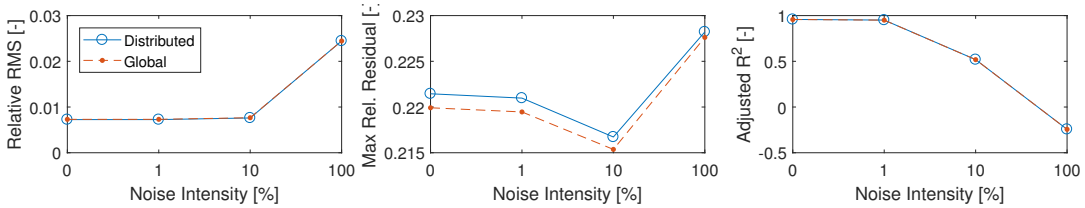


(a) $\rho = 1$

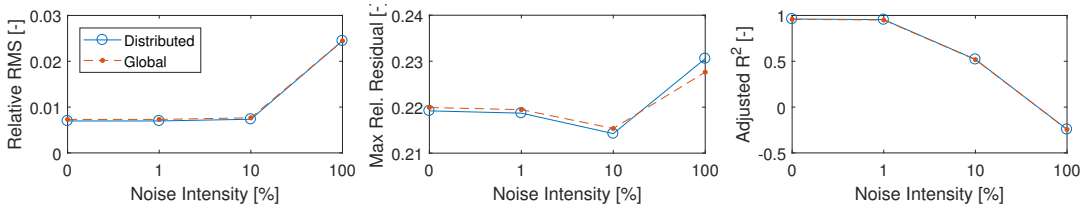


(b) $\rho = 0.01$

Figure B.21: Validation results and coefficient of determination of the 5D $(3,1,1)$-spline.



(a) $\rho = 1$



(b) $\rho = 0.01$

Figure B.22: Validation results and coefficient of determination of the 5D (5)-spline.

B.3. Derivation Sequential ADMM

The OLS problem that is to be optimized is given in Equation (B.14). The regression matrix $\mathbf{X}_p$ for every partition is multiplied with the B-coefficients of partition p and compared to the measurements $\boldsymbol{Y}_p$. The continuity constraints that are placed on the optimal solution are summarized in the matrix $\mathbf{H}$.

$$\begin{aligned} &\text{argmin}_{\boldsymbol{c}} \mathscr{J}(\boldsymbol{c}) = \sum_{p=1}^{P} \tfrac{1}{2} ||\mathbf{X}_p \boldsymbol{c}_p - \boldsymbol{Y}_p||_2^2 \\ &\text{subject to } \mathbf{H}\boldsymbol{c} = 0 \end{aligned} \tag{B.14}$$

To solve this problem, the augmented Lagrangian is formed and given by Equation (B.15). The augmentation is the quadratic term at the end, which includes the penalty factor $\rho > 0$. This places an extra penalty on non-feasible solutions. This does not change the optimal solution, because if the solution is feasible, the squared term should be equal to zero. It only changes the convergence properties of the following iterative scheme.

$$\mathscr{L} = \sum_{p=1}^{P} \frac{1}{2} ||\mathbf{X}_p \boldsymbol{c}_p - \boldsymbol{Y}_p||_2^2 + \lambda^T (\mathbf{H}\boldsymbol{c}) + \frac{\rho}{2} ||\mathbf{H}\boldsymbol{c}||_2^2 \tag{B.15}$$

The continuity matrix $\mathbf{H}$ can be subdivided for all partitions, in two sets: the columns that are multiplied with the coefficients of partition p, and the columns that are multiplied with the coefficients of all other partitions. The latter are indicated in this derivation with the subscript $\neg p$. This structure is given in Equations (B.16) and (B.17).

$$\mathbf{H} \cdot \boldsymbol{c} = \begin{bmatrix} \mathbf{H}_1 & \dots & \mathbf{H}_p & \dots & \mathbf{H}_p \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_p \\ \vdots \\ \boldsymbol{c}_P \end{bmatrix} \tag{B.16}$$

$$\mathbf{H} \cdot \boldsymbol{c} = \begin{bmatrix} \mathbf{H}_p & \mathbf{H}_{\neg p} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{c}_p \\ \boldsymbol{c}_{\neg p} \end{bmatrix} \tag{B.17}$$

Finding the optimal solution in this case requires taking the derivative w.r.t. the coefficients of partition p. The resulting equation is given in Equation (B.18), which can be rewritten into Equations (B.19) and (B.20).

$$\nabla_{\boldsymbol{c}_p} \mathscr{L} = \mathbf{X}_p^T \mathbf{X}_p \boldsymbol{c}_p - \mathbf{X}_p^T \boldsymbol{Y}_p + \lambda^T \mathbf{H}_p + \rho \mathbf{H}_p^T \mathbf{H}\boldsymbol{c} \tag{B.18}$$

$$\nabla_{\boldsymbol{c}_p} \mathscr{L} = \mathbf{X}_p^T \mathbf{X}_p \boldsymbol{c}_p - \mathbf{X}_p^T \boldsymbol{Y}_p + \lambda^T \mathbf{H}_p + \rho \left(\mathbf{H}_p^T \mathbf{H}_p \boldsymbol{c}_p + \mathbf{H}_p^T \mathbf{H}_{\neg p} \boldsymbol{c}_{\neg p} \right) \tag{B.19}$$

$$\nabla_{\boldsymbol{c}_p} \mathscr{L} = \left(\mathbf{X}_p^T \mathbf{X}_p + \rho \mathbf{H}_p^T \mathbf{H}_p \right) \boldsymbol{c}_p - \mathbf{X}_p^T \boldsymbol{Y}_p + \lambda^T \mathbf{H}_p + \rho \mathbf{H}_p^T \mathbf{H}_{\neg p} \boldsymbol{c}_{\neg p} = 0 \tag{B.20}$$

$$\boldsymbol{c}_p = \left(\mathbf{X}_p^T \mathbf{X}_p + \rho \mathbf{H}_p^T \mathbf{H}_p \right)^{-1} \left(\mathbf{X}_p^T \boldsymbol{Y}_p - \lambda^T \mathbf{H}_p - \rho \mathbf{H}_p^T \mathbf{H}_{\neg p} \boldsymbol{c}_{\neg p} \right) \tag{B.21}$$

By setting $\nabla_{\boldsymbol{c}_p} \mathscr{L}$ in Equation (B.20) equal to zero and solving it, the update formula in Equation (B.21) is found. By alternatingly updating the coefficients of all partitions, the iteration scheme in Equations (B.22) to (B.24) is found, followed by the dual update in Equation (B.25). This method is called Sequential ADMM in this thesis, because of the fact that the updates can only be updated sequentially, since every next partition in a single iteration, depends on the updated values of the previous ones. Trying to parallelize the updates, by using the values of the previous update, resulted in substantially worse convergence properties, and was even divergent in many cases.

$$\boldsymbol{c}_1^{k+1} = \left(\mathbf{X}_1^T \mathbf{X}_1 + \rho \mathbf{H}_1^T \mathbf{H}_1 \right)^{-1} \left(X_1^T \boldsymbol{Y}_1 - \mathbf{H}_1^T \lambda^k - \sum_{j=2}^{P} \rho \mathbf{H}_1^T \mathbf{H}_j \boldsymbol{c}_j^k \right) \tag{B.22}$$

$$\dots$$

$$\boldsymbol{c}_p^{k+1} = \left(\mathbf{X}_p^T \mathbf{X}_p + \rho \mathbf{H}_p^T \mathbf{H}_p \right)^{-1} \left(\mathbf{X}_p^T \boldsymbol{Y}_p - \mathbf{H}_p^T \lambda^k - \sum_{i=1}^{p-1} \rho \mathbf{H}_p^T \mathbf{H}_i \boldsymbol{c}_i^{k+1} - \sum_{j=p+1}^{P} \rho \mathbf{H}_p^T \mathbf{H}_j \boldsymbol{c}_j^k \right) \tag{B.23}$$

$$\dots$$

$$\boldsymbol{c}_P^{k+1} = \left(\mathbf{X}_p^T \mathbf{X}_p + \rho \mathbf{H}_p^T \mathbf{H}_p \right)^{-1} \left(\mathbf{X}_p^T \boldsymbol{Y}_P - \mathbf{H}_p^T \lambda^k - \sum_{i=1}^{P-1} \rho \mathbf{H}_p^T \mathbf{H}_i \boldsymbol{c}_i^{k+1} \right) \tag{B.24}$$

$$\lambda^{k+1} = \lambda^k + \rho \mathbf{H}\boldsymbol{c} \tag{B.25}$$

In the general case as described above, a lot of multiplications are done that have no effect on partitions p. Using the knowledge of the system that is being optimized, it can be said that if two partitions i and j do not share an edge, the multiplications $\mathbf{H}_i^T\mathbf{H}_j$ and $\mathbf{H}_j^T\mathbf{H}_i$, will equal zero. This means that only direct neighbors have to be included in the update step of partition p. Using the subscript $n \subseteq N_p$ for these direct neighbors, the update is now given by Equation (B.26), in which N_p is the set of direct neighbors of partition p.

$$\boldsymbol{c}_p = \left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\left(\mathbf{X}_p^T\boldsymbol{Y}_p - \lambda^T\mathbf{H}_p - \rho\mathbf{H}_p^T\mathbf{H}_n\boldsymbol{c}_n\right) \tag{B.26}$$

The new iteration scheme then becomes as in Equations (B.27) to (B.29)

$$\boldsymbol{c}_1^{k+1} = \left(\mathbf{X}_1^T\mathbf{X}_1 + \rho\mathbf{H}_1^T\mathbf{H}_1\right)^{-1}\left(X_1^T\boldsymbol{Y}_1 - \mathbf{H}_1^T\lambda^k - \sum_{j>1}^{|N_1|}\rho\mathbf{H}_1^T\mathbf{H}_{n_j}\boldsymbol{c}_{n_j}^k\right) \tag{B.27}$$

$$\cdots$$

$$\boldsymbol{c}_p^{k+1} = \left(\mathbf{X}_p^T\mathbf{X}_p + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\left(\mathbf{X}_p^T\boldsymbol{Y}_p - \mathbf{H}_p^T\lambda^k - \sum_{i<p}^{|N_p|}\rho\mathbf{H}_p^T\mathbf{H}_{n_i}\boldsymbol{c}_{n_i}^{k+1} - \sum_{j>p}^{|N_p|}\rho\mathbf{H}_1^T\mathbf{H}_{n_j}\boldsymbol{c}_{n_j}^k\right) \tag{B.28}$$

$$\cdots$$

$$\boldsymbol{c}_P^{k+1} = \left(\mathbf{X}_P^T\mathbf{X}_P + \rho\mathbf{H}_P^T\mathbf{H}_P\right)^{-1}\left(\mathbf{X}_P^T\boldsymbol{Y}_P - \mathbf{H}_P^T\lambda^k - \sum_{i<P}^{|N_P|}\rho\mathbf{H}_P^T\mathbf{H}_{n_i}\boldsymbol{c}_{n_i}^{k+1}\right) \tag{B.29}$$

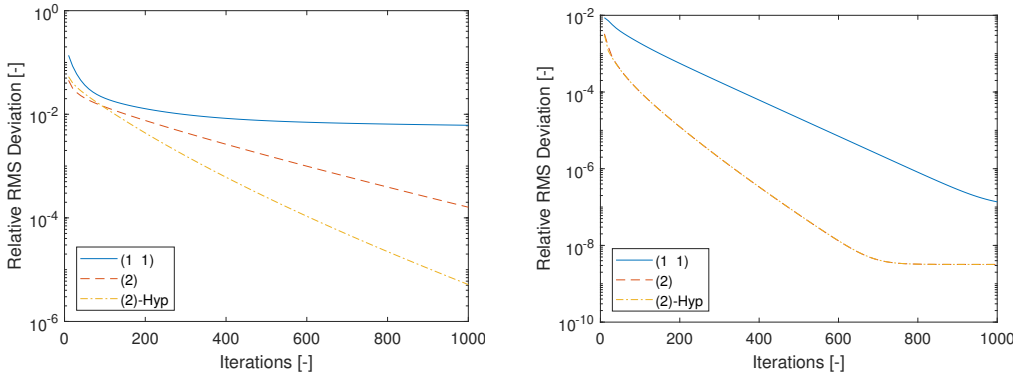
$$\lambda^{k+1} = \lambda^k + \rho\mathbf{H}\boldsymbol{c} \tag{B.30}$$

Using only the neighbors saves high amount of unnecessary computations, especially when P is high. Although it can not work in parallel, it has still good properties in converging to a good solution, as is shown in Appendix B.3. Although this method has not been used to construct the final models, Variable Splitting ADMM in Appendix B.1 is used, it is still included as this is a novel method as well. This may still save time when it comes to large systems, and also lowers the RAM requirements. It also showed a considerable improvement when it comes to meeting the constraints compared to Dual Ascent in Appendix B.5.

B.4. Sequential ADMM Tests

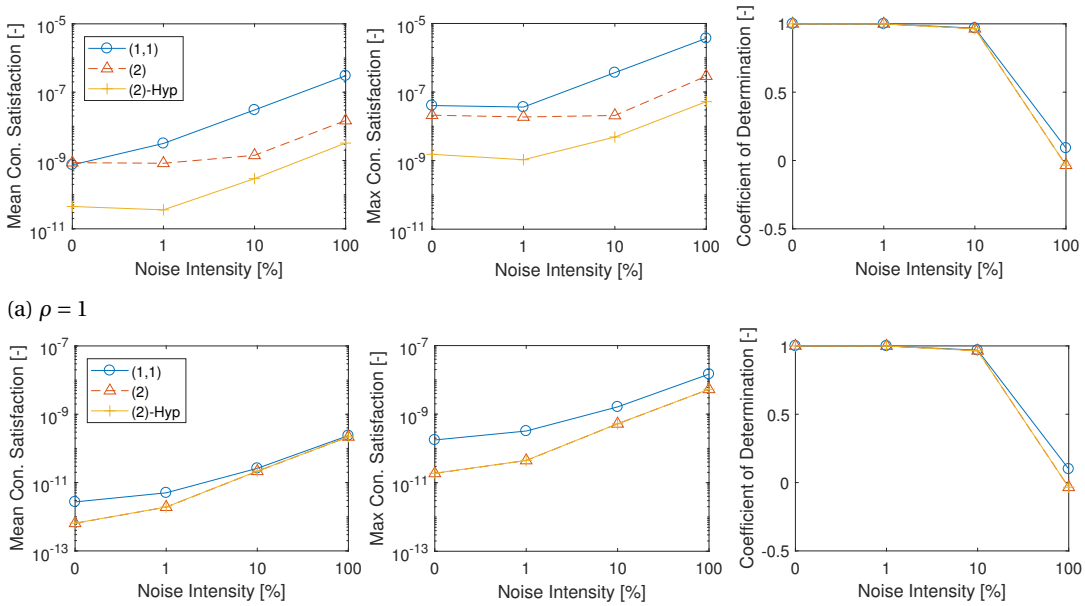
The Sequential ADMM test has also been implemented in Matlab and tested in order to show its performance. This performance has been tested for two different penalty factors, $\rho = 1.0$ and $\rho = 0.01$, to see whether this affects the quality of the estimation. Convergence plots have been shown in order to assess the algorithm's ability to converge to the global solution. Additionally, the performance in terms of constraints satisfaction and the coefficient of determination are shown. This has been done for the same 2-, 3-, 4-, and 5-Dimensional models as in Appendix B.2.

B.4.1. 2-Dimensional



(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.23: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_1}(\alpha, M)$ with sequential ADMM.

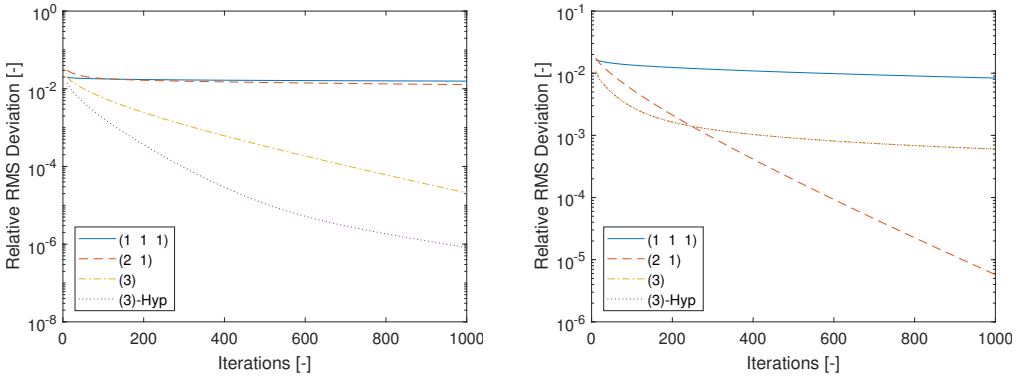


(a) $\rho = 1$

(b) $\rho = 0.01$

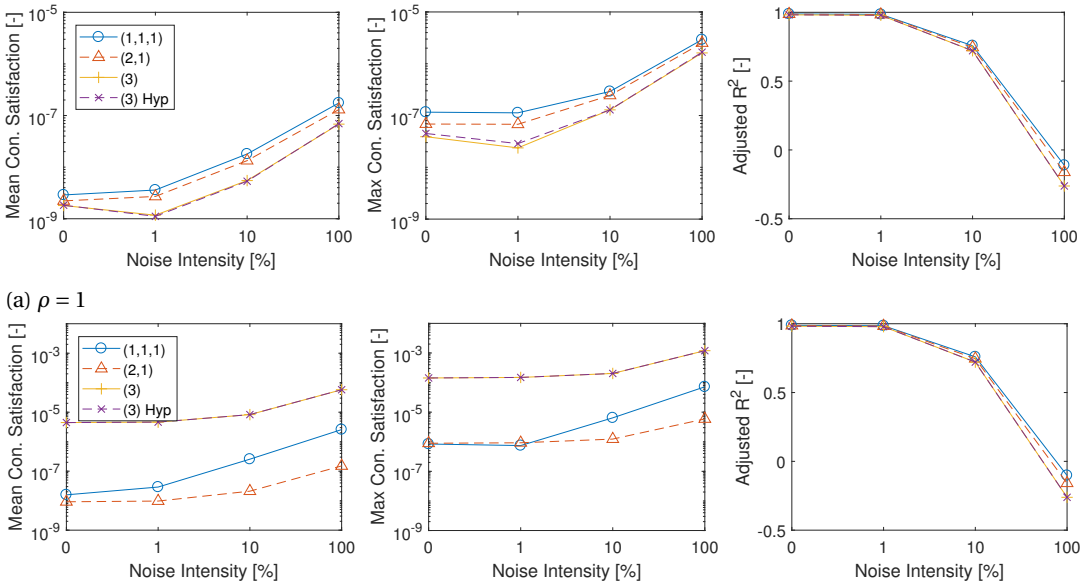
Figure B.24: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_1}(\alpha, M)$ at various noise intensities with sequential ADMM.

B.4.2. 3-Dimensional



(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.25: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_2}(\alpha, \beta, M)$ with sequential ADMM.

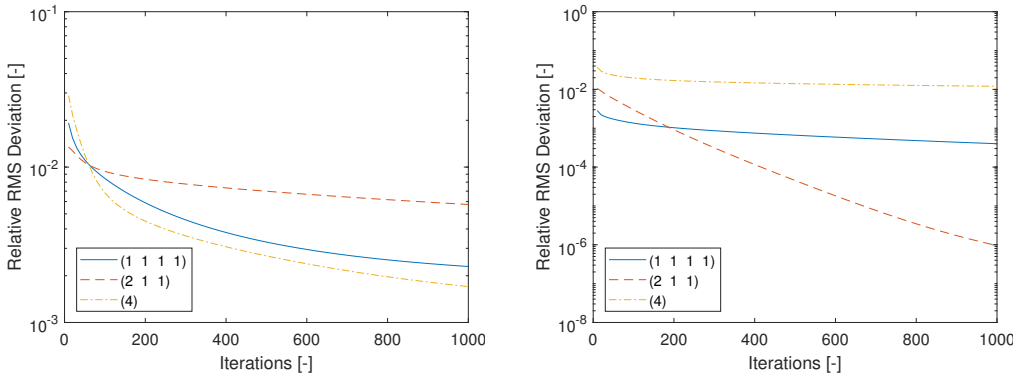


(a) $\rho = 1$

(b) $\rho = 0.01$

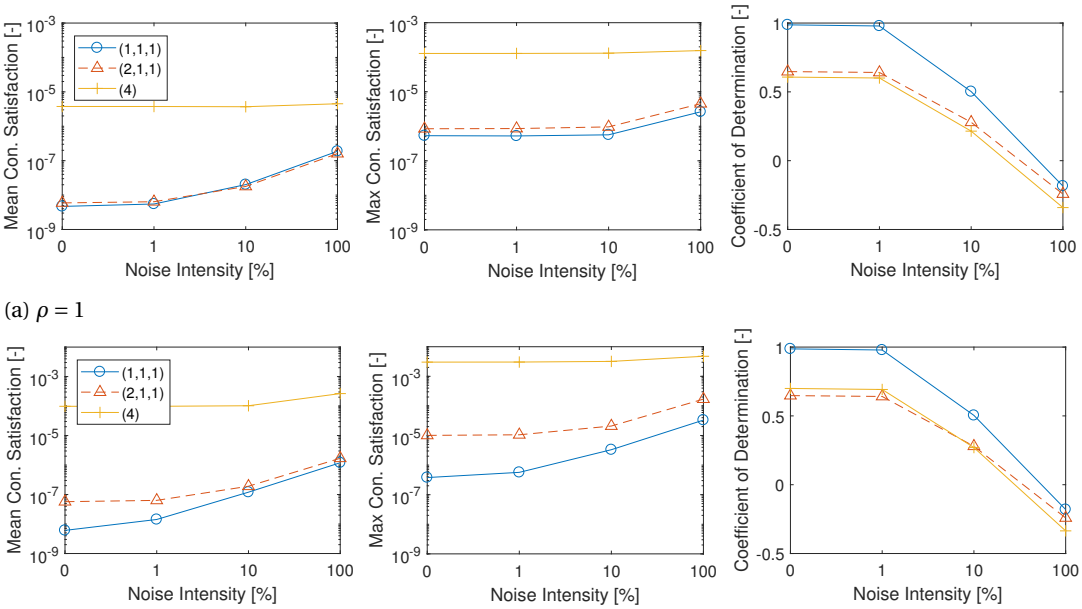
Figure B.26: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_2}(\alpha, \beta, M)$ at various noise intensities with sequential ADMM.

B.4.3. 4-Dimensional



(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.27: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_5}(\alpha, M, \delta_{LSSD}, \delta_{LEL})$ with sequential ADMM.

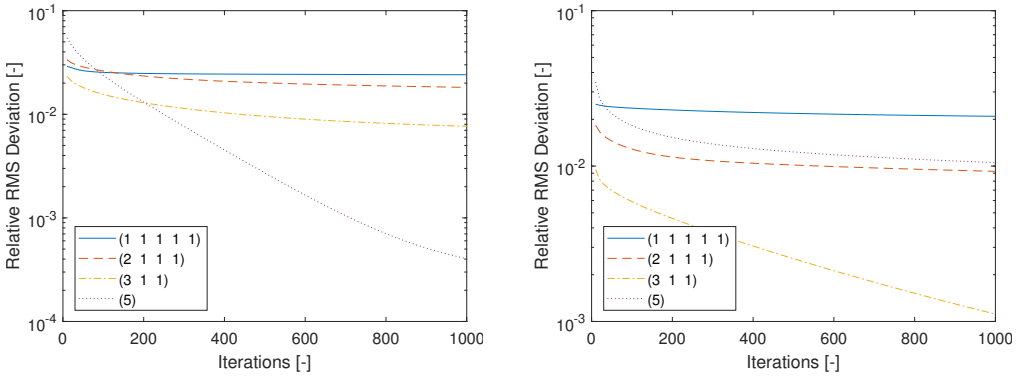


(a) $\rho = 1$

(b) $\rho = 0.01$

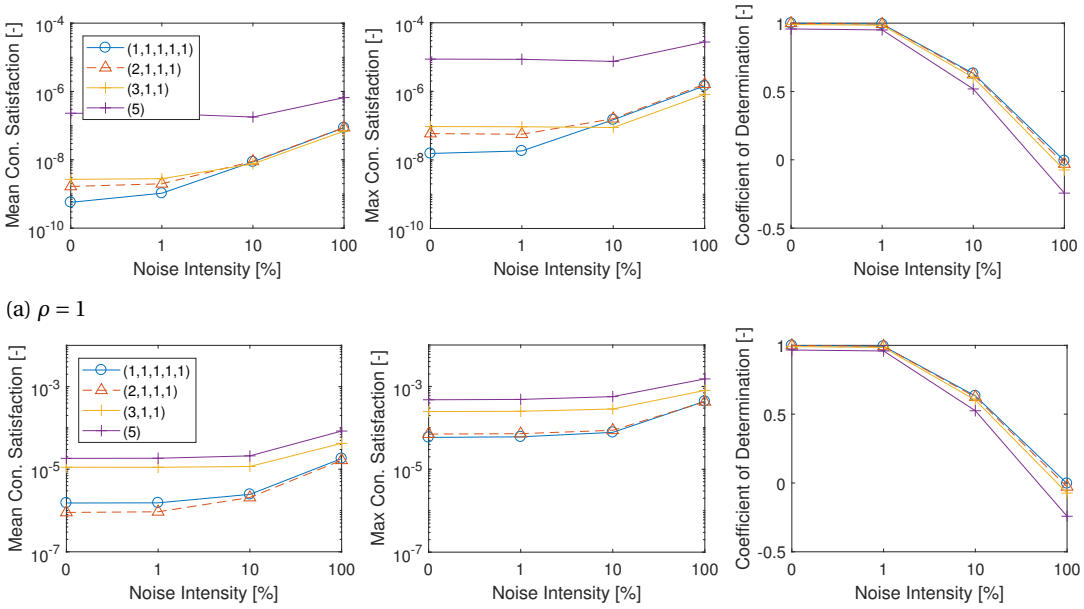
Figure B.28: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_5}(\alpha, M, \delta_{LSSD}, \delta_{LEL})$ at various noise intensities with sequential ADMM.

B.4.4. 5-Dimensional



(a) $\rho = 1$ (b) $\rho = 0.01$

Figure B.29: Convergence of the distributed algorithm towards the global optimum, at 0% noise intensity for $C_{X_{14}}(\alpha, \beta, M, \delta_{RIBLEF}, \delta_{ROBLEF})$ with sequential ADMM.



(a) $\rho = 1$

(b) $\rho = 0.01$

Figure B.30: Mean and maximum constraint satisfaction, and the coefficients of determination for $C_{X_{14}}(\alpha, \beta, M, \delta_{RIBLEF}, \delta_{ROBLEF})$ at various noise intensities with sequential ADMM.

B.5. Dual Ascent Tests

Dual Ascent is a distributed method that had been used in an early stage of this thesis project. Dual Ascent has been used in combination with multivariate Simplex B-Splines already in Wavefront Reconstruction (WFR) problems [5], is a relatively easy implementable algorithm, and able to work in parallel because of the local cost functions of Simplex B-Splines [3].

The algorithm that has been used, is given below in Equations (B.31) to (B.33).

$$\boldsymbol{c}_i^{l+1} = \hat{\boldsymbol{c}}_i^{loc} + R_{N_i}\mathbf{H}_i^T \boldsymbol{y}_i^l, \qquad \forall i \in P \tag{B.31}$$

$$\boldsymbol{v}^{l+1} = \mu \cdot \boldsymbol{v}_i^l + \alpha^l \mathbf{H}\boldsymbol{c}^{l+1} \tag{B.32}$$

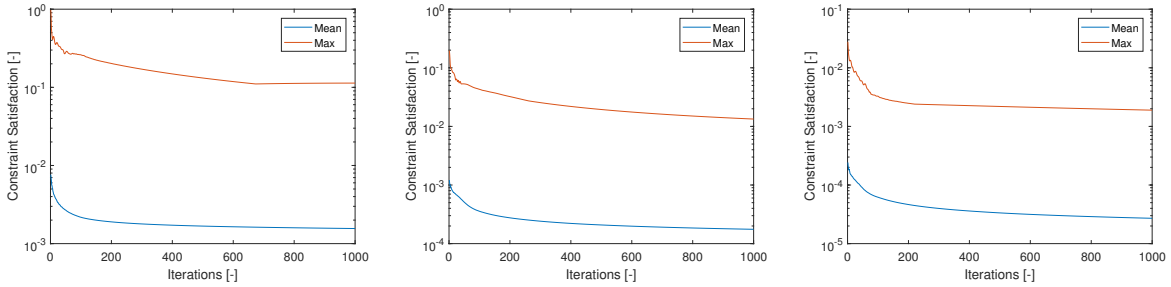
$$\boldsymbol{y}^{l+1} = \boldsymbol{y}_i^l + \boldsymbol{v}_i^{l+1} \tag{B.33}$$

The locally estimated coefficients $\hat{\boldsymbol{c}}_i^{loc}$ are computed at initialization and remain constant. The first step is repeated for all partitions i that are defined. R_{N_i} is defined as $\mathcal{N}_i(\mathcal{N}_i^T\mathbf{X}_i^T\mathbf{X}_i\mathcal{N}_i)^{-1}\mathcal{N}_i^T$. It uses $\mathcal{N}_i = \text{null}(\mathbf{H}_i)$, with $\mathbf{H}_i$ the internal constraints of the partition. The local estimation can be done either using the above mentioned null-space, or using an iterative solver [1, 14].

The second and the third steps are centralized operations, and computes the global dual vector. First, in the second step, Nesterov's momentum theory has been used [12], emphasizing the search direction in which the past iterations have ascended. It is defined by the momentum rate $0 \leq \mu < 1$, which reduces the momentum, and the learning rate α^l, which gives weight to the new search direction. The momentum rate is important to forget the previous search directions, and give new information a chance. In case all new updates are in the same direction, momentum will build up an accelerate the search. Momentum might cause 'overshoot' of the optimal solution, but still reaches this optimum faster.

In the demonstrations described below, the momentum rate was allowed to slowly increase from 0.7 to 0.95, by the formula $\mu^{l+1} = \mu^l \cdot 0.95 + 0.05$, in order to speed up convergence in the last, slowest region. The learning rate was chosen to be $5 \cdot 10^{-9}$, which was the highest learning rate possible, before the algorithm became unstable and diverged.

The model that has been used is a $(1,1,1)$-simplotope tessellation of $C_{\mathbf{Y}_2}(\alpha, \beta, M)$ of degrees $(5,5,5)$, with a hypercube grid of $[6,6,4]$, and a partitioning of $[3,2,2]$, resulting in 12 partitions of 12 simplotopes. Three different tests with overlaps of 0, 1, and 2 for the initial estimation of $\hat{\boldsymbol{c}}_i^{loc}$ have been performed. The main issue was whether the algorithm would be able to satisfy the global constraints, which is compared in Figure B.31.



(a) Overlap = 0 (b) Overlap = 1 (c) Overlap = 2

Figure B.31: Mean and Maximum constraint mismatch of the dual ascent algorithm.

The trend indicates that increasing the overlap improves the estimation, but also increases the complexity of the initial estimation. However, it shows that with an overlap of 2 simplotopes in all directions, the maximum constraint mismatch still remains above 10^{-3} after 1,000 iterations. Moreover, it does not seem to improve anymore.

At this point initial explorations with Sequential ADMM had been performed, which showed significant improvement in constraint satisfaction, in less iteration and less time. Therefore Dual Ascent was laid down and not considered anymore, and the focus changed to ADMM.

C

Code Documentation

In this appendix all software is documented, consisting of three main parts. Firstly, the estimation's preparation code that is written in Matlab. Then there are two implementations in C++, one for the VSADMM estimation of the coefficients, and one for the evaluation of all 108 simplotope models in real time.

C.1. Matlab Estimator Description

In this section the set up for the coefficient estimator is documented, not including the C++ implementation of the VSADMM algorithm to estimate the coefficients. This setup is written in Matlab and prepares the model structure for coefficient estimation, which includes establishing the triangulation, dividing the identification- and validation data over the Simplotopes, find the continuity conditions, and set up the partitions. This all is fed to the VSADMM routine, with the pre-calculations of all the inverses done in Matlab as well. First the main variables and their components are defined, then the flow of the software is given.

C.1.1. Main Variables

Four structs are the main variables used in the estimation algorithm. The `Spline` struct contains the main properties of the complete spline. The `Simplotopes` cell of structs contains the information of the simplotopes, such as the the data in that simplotope. Every simplotope has an own cell. The `Edges` has a cell entry for the edges in each layer, and an extra entry to define the edges between the simplotopes. At last, each partition has an own struct in the `Partitions` cell of structs. This is subdivided in Local, Neighbors, and Overlap information.

Spline

Input	All cartesian coordinates of the dataset
Data	All measurement values of the dataset
nu	Vector containing the dimension of each layer
d	Vector containing the degree of each layer
r	Vector containing the continuity order of each layer
p	Vector containing the partition division for each layer
overlap	Overlap setting for the distributed estimator
Grid	Cell vector containing the hypercube divisions for each layer
T	Cell vector containing the triangulations of each layer
V	Cell vector containing the Vertices of each layer
Kappa	The κ division of the B-Polynomials
MultiCo	The multinomial coefficient for the B-Polynomials
GlobalH	The Global Continuity Matrix
AtEdge	Vector containing to which edge each continuity constraint belongs
Analysis	Struct containing (validation) analysis of the found solution

Simplotopes

T	Vector containing the simplices in each layer this simplotope is composed of
Edges	Matrix with the edges of this simplotope, describing the simplotopes this edge is made of, the out of edge layer, and the edge in that layer
Data	Vector with measurement values that are contained in the convex hull of this simplotope
Bar	Matrix with Barycentric coordinates of the measurements that are in this Simplotope's convex hull
Cart	Matrix with Cartesian coordinates of the measurements that are in this Simplotope's convex hull
Validation	Struct with the Values, Barycentric-, and Cartesian coordinates of the Validation measurements that are contained in the convex hull of this simplotope
Bc	B-Coefficients describing the B-Polynomial in this Simplotope

Edges

Cell Entries 1:nLay	Matrix for each layer, containing edge information per Layer: [Simplex 1, Simplex 2, Out of Edge Vertex 1, Out of Edge Vertex 2]
Cell Entry end	Matrix containing information of the edges between Simplotopes: [Simplotope 1, Simplotope 2, Out of Edge Layer, Edge in that Layer]

Partitions

Local:

Simplotopes	Vector containing simplotopes in this partition
Edges	Vector containing indices of the internal edges in this partition
Constraints	Matrix with continuity conditions between simplotopes in this partition
ConstraintsIt	Matrix with continuity conditions between simplotopes within this partition and neighbors, but only the columns for this partition's B-Coefficients
iCon	Vector containing global indices of which continuity conditions are in Constraints
iBc	Vector containing global indices of which B-Coefficients this partition has
XtX	Matrix with the regression matrix times its transpose
Data	Vector with data values of the measurement within simplotopes of this partition
XtY	Vector with the transposed regression matrix times the data vector

Neighbors:

Simplotopes	Vector containing simplotopes neighboring this partition
Constraints	Matrix with continuity conditions between simplotopes within this partition and neighbors
Edges	Vector containing indices of the edges between this partition and neighboring simplotopes
ConstraintsNoLoc	Matrix with continuity conditions between simplotopes within this partition and neighbors, excluding the columns for this partition's B-Coefficients
iCon	Vector containing global indices of which continuity conditions are in Contraints
iBc	Vector containing global indices of which B-Coefficients this partition and the neighbors have
iBcNoLoc	*iBc* but excluding the B-Coefficients of this partition

Overlap:

Same entries as ***Local***, but including the overlap Simplotopes

C.1.2. Components of the Estimation Algorithm

The components of the Variable Splitting ADMM algorithm has been depicted in a flow diagram in Figure C.1. All components are put together in `DistributedSimplotopeEstimator.m`. In this section the components are explained in further detail.

ModelProperties.m and CreateSimplotopes.m

In `ModelProperties` the dataset is loaded and divided over the `Spline` struct, together with the simplotope layers, degrees, continuity order, hypercube partitioning for the triangulation, and the partition division. The grid is set per dimension, whereas the partitioning for the distributed estimator is set per simplotope layer.

In `CreateSimplotopes` the grid- and layer division is used in order to form a triangulation per layer with `bsplinen_CDVtriangulateExt2()`. The individual triangulations are combined using the tensor product between all of the layers. Note, the order as set in this part is important, as it is used later on, in the evaluation function.

The edges between the simplices are found per layer by looping through all the simplices, compare the vertices, and select the ones that have a single vertex different. The edges between simplotopes are found by using the same strategy, but then by comparing which simplices they have in each layer. This gives an edge between two simplotopes if they differ in one layer, and they share an edge in that layer.

GetPolynomial.m
The basis polynomial of a single simplotope is found per layer, including the κ indices, as well as the multinomial coefficients, which are combined into a tensor product. In order to speed up computation of the regression matrices, the indices of the tensor product are saved. Using these, the full polynomial of the simplotope spline can be quickly computed by combining the individual layers' basis function.

Cart2Bary.m
The membership search and the barycentric coordinates are computed using `MEX` function `eval_BarycentricCoordinates()`, that has a similar structure as the evaluation function explained in Appendix C.3, but returns the barycentric coordinates and the simplotope it belongs to. The input coordinates and data are divided over the simplotopes, as well a division made between the identification and validation measurements. This division is basen on a vector of uniformly distributed random numbers compared to the ratio of validation points $Spline.ValidationRatio$.

ContinuityConditions.m
Before the continuity conditions are generated, first the an upper bound on the total amount of continuity conditions are computed in order to pre-allocate the matrices that will be used when to collect all constraints. The conditions are formed per layer in the conventional manner. Because of the tensor product between the layers these conditions are copied for every copy of the B-net. Since the global constraint matrix has to be saved in a sparse matrix, and indexing is slow in Matlab, all the conditions are first saved as coordinate-value triplets and later transformed into a sparse constraint matrix.

CreatePartitions.m
The creation of the partitions is either based on a single number, in which case the number of partitions is divided by that, or on the most frequently used division per layer. In case an overlap is given, this one is found based on the edges of the simplotopes in each of the partitions. All edges are divided over the partitions, distinguishing internal edges, between simplotopes in the partitions, and external edges, with neighboring partitions' simplotopes. Based on these edges, subsets of the global constraint matrix are made, consisting of those internal of the partitions, and the ones with the neighbors. Indices of the particular B-Coefficients and constraints that are used are saved for later use.

CalcBasisPolynomial2.m and CreateRegressionMatrices.m
The regression matrices are found in this function, but basically links through to the `MEX` function `BasPol()`. There are multiple options: either the actual regression matrix is requested, but generally it is more convenient in terms of memory requirements to directly compute the the regression matrix times its transpose $\mathbf{X}^T\mathbf{X}$, as well as the regression matrix transposed times the measurement vector $\mathbf{X}^t\boldsymbol{Y}$. These are combined into a regression matrix per partition. Optionally a weight matrix can be passed or differential regression points can be used.

VariableSplittingADMM.m
The main contribution of this thesis. The variable splitting algorithm consists of two stages. First initialization during which the inverses are being pre-computed and saved in order to be used in the iterations. The iterative phase is implemented in a C++ `MEX` function, as explained in the next section.

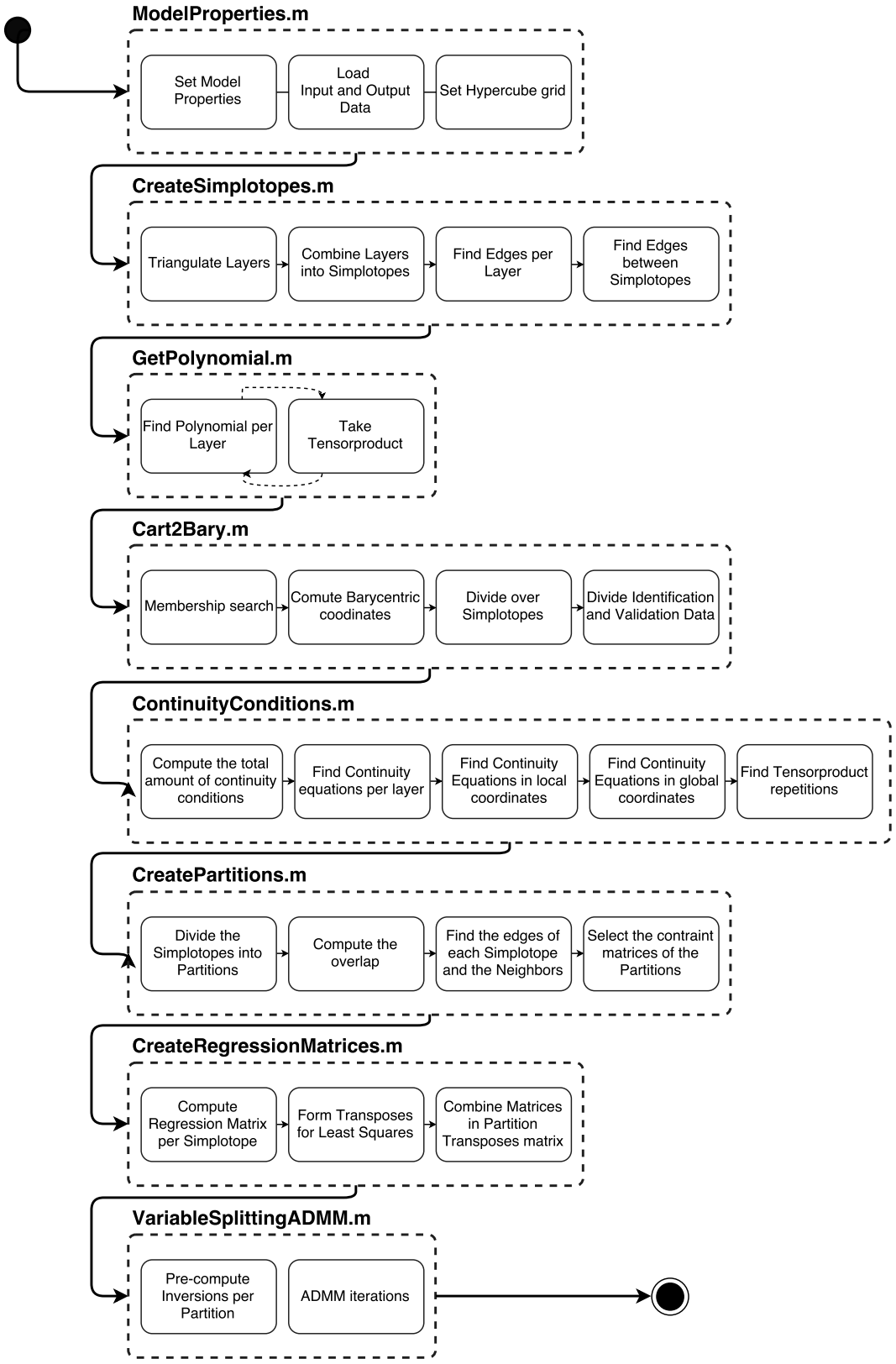


Figure C.1: Flow diagram of all the activities during the Variable Splitting ADMM estimation routine.

C.2. C++ VariableSplitting ADMM Description

The iterative phase of the VSADMM routine has been implemented in C++ to speed up this part. It does not include a parallel implementation, all partitions' coefficients are computed sequentially, but it does decrease the computation time, because C++ handles iterative processes better and faster than Matlab. Since it is possible that there are over 1,000 partitions, that have to be iterated 1,000 times, using the C++ algorithm for this part can result in a speedup of 5-10 times.

It consists of two main classes, as shown in Figure C.2: the `Partition` class contains the main components for every partition, which includes the inverses and the initial estimation, the local constraint matrix, and the indices of all the constraints. The `vsADMM` class is the iterative scheme, that links everything together, and thus makes sure that all partitions iterate, and also receive the correct coupling coefficients for the new iteration. The C++ classes require the `Eigen`[1], which is used for intuitive matrix operations. This toolbox needs to be included while compiling the code.

The iterator can be used through the `MEX` function `mexVariableSplittingADMM()`. The iterator is initiated by passing through a struct with the following options:

1. *rho*: a scalar, setting the penalty factor ρ of the estimation
2. *itmax*: a scalar, setting the maximum amount of iterations for the iterative procedure
3. *ABSTOL*: a scalar, setting the absolute tolerance of the stopping criterion
4. *RELTOL*: a scalar, setting the relative tolerance of the stopping criterion
5. *AdaptSensitivity*: a scalar, providing the μ of the adaptive penalty factor
6. *itmin*: a scalar, setting the minimum amount of iterations before the adaptive penalty factor can break the iterations
7. *getIntermediateResults*: a scalar, determining whether intermediate results should be saved (> 0), or not (< 0), per 10 iterations

These options are saved into *vsadmmOptions* and passed to the constructor. Partitions can be added to the environment by passing 4-7 arguments:

1. *ReginvCon*: a $\hat{d}_p \times R_p$ matrix, containing the cached $\left(\mathbf{X}^T\mathbf{X} + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\mathbf{H}_p^T$
2. *XtYADMM*: a $\hat{d}_p \times 1$ vector, containing the cached initial estimation $\left(\mathbf{X}^T\mathbf{X} + \rho\mathbf{H}_p^T\mathbf{H}_p\right)^{-1}\left(\mathbf{X}^T\boldsymbol{Y}\right)$
3. *Con*: a $R_p \times \hat{d}_p$ sparse matrix, containing the matrix constraints of partition p, $\mathbf{H}_p$
4. *iC*: a $R_p \times 1$ vector, containing the indices of the *Con* matrix, $\mathscr{H}_p \cup \mathscr{H}_{p,n}$
5. *LocalConstraints*: a $|\mathscr{H}_p| \times 1$ vector, containing the local indices of the *Con* vector that are internal
6. *Coeff*: a $\hat{d}_p \times 1$ vector, containing the initial values of the B-Coefficients
7. *Duals*: a $R_p \times 1$ vector, containing the initial values of the duals

The number of coefficients in partition p is indicated with $\hat{d}_p$. The last three entries, *LocalConstraints*, *Coeff*, and *Duals* are optional. When no local constraints are passed, this simply means that there are none and it is kept empty. Extra caution has to be exercised here, as the local indices has to start with index base 0. If no coefficients or duals are passed, these are initiated at all zeroes.

The iterative procedure can be started by passing no arguments, where it will iterate either for a maximum of *itmax* times, or until the stopping criterion is reached, or when the adaptive penalty factor algorithm determines that the residuals are too far apart. The output is a struct *OutputStruct*, with the following entries:

1. *nIterations*: a scalar, indicating how many iterations the algorithm performed
2. *AdaptRho*: a scalar, indicating whether the penalty factor should be adapted upwards (1), downwards (-1), or the stopping criterion is reached (10)
3. *Norms*: a $nIterations \times 4$ matrix, indicating, for every iteration, the norms of the dual residual, primal residual, the primal tolerance, and the dual tolerance.
4. *IntermediateResults*: a $J \cdot \hat{d} \times x$ matrix, with the intermediate B-Coefficients at x moments during the iterations
5. *IntermediateIterations*: a $1 \times x$ vector, with the number of iterations at which the above intermediate results are obtained

[1] http://eigen.tuxfamily.org/

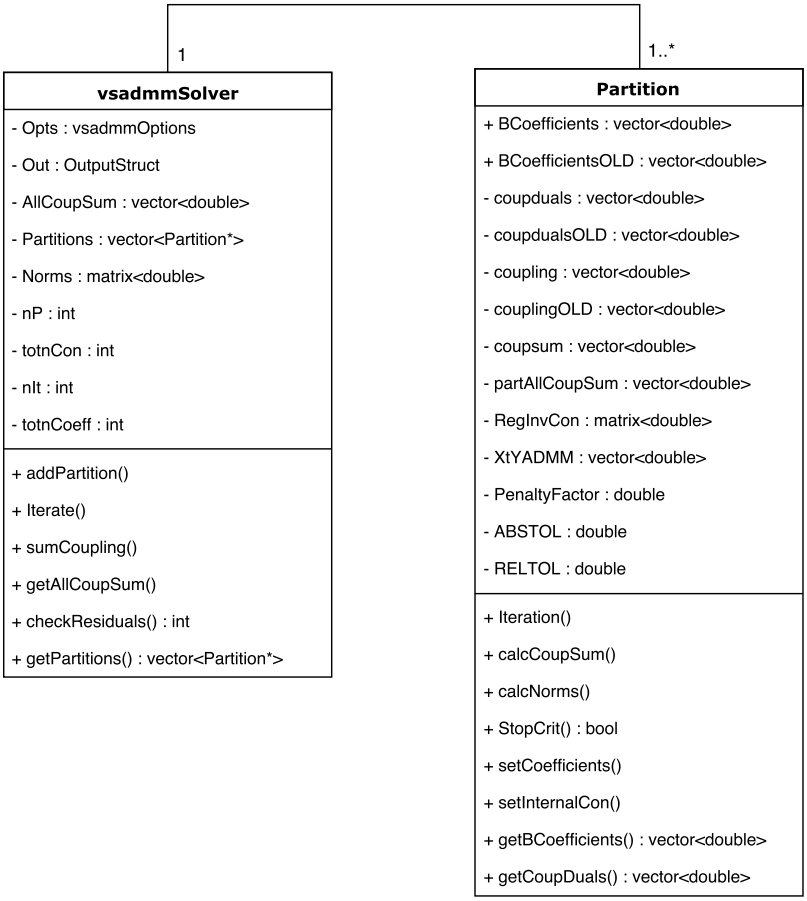


Figure C.2: Class Diagram of the C++ VSADMM estimator.

The last two outputs are only retrieved when the *getIntermediateResults* setting in the input options is greater than zero. Otherwise, only the results at the iteration when the itertive scheme breaks out because of the adaptive penalty factor is included.

C.3. C++ Evaluation Function Description

This section contains documentation of the C++ implemented code of the evaluation function. To compile the function, the Boost[2] portable library is required, as well as the Matrix Template Library 4 (MTL4)[3].

First a class diagram of the underlying C++ structure is shown in Figure C.3. This provides the main components of the used evaluation function. The classes `cBSplinen`, `cTriangulation`, and `cSimplex` are taken from the previously implemented Simplex evaluation function, developed by Coen de Visser. All newly added classes and functions are based on the already existing code, while some functions could even be copied for major parts.

The `cLayer` class represents a layer of a simplotope spline, inheriting all the properties of a simplex spline. It only does not use the coefficients of this spline, or the direct evaluation methods. Every layer is part of a `cSimplotopeSpline`, which has B-Coefficients, and combines the polynomial bases. All simplotope splines can be placed in a `cSimplotopeSplineSet`, in order to perform the evaluation of all 108 simplotope models of the ICE aircraft.

Two different interfaces with Matlab exist in the form of `MEX` functions: `eval_SimplotopeSplineSet()` for direct evaluation, and `evalDerivatives_SimplotopeSplineSet()`, for evaluation of the derivatives of all splines in the set.

Direct evaluation starts in the simplotope spline set, as Figure C.4 shows, looping through all simplotopes in the set. Each layer of a simplotope computes its own simplex polynomial basis, by performing a membership search and computing the barycentric coordinates. These polynomials are combined in the simplotope spline, by taking the tensor product between them all, and multiplying each term with the B-Coefficients. The output is returned to the simplotope spline set. In the spline set the output is combined in a vector for all simplotope splines.

The simplotopes also have a setting that links together submodels with the same underlying structure, i.e. the same submodels for different main coefficients. In this case, the polynomial bases do not have to be calculated again, and that part is skipped. All bases are shared and only the final output is computed in the simplotope spline.

Evaluation of the derivatives is done in a similar method, shown in Figure C.5. A derivative in any direction can only exist per layer. This means that only the derivative layer changes, while the other remain the same. This property has been used for evaluation, by firstly computing all polynomials of the parent-spline. Then per for every layer the derivative polynomials are computed, combined with the other directions' layers of the parent-spline, and the output computed using the derivative B-Coefficients. In this way all the required bases are only computed once, and thus the computational effort minimized.

Similar to the direct evaluation, the bases can be shared between different submodels to reduce unnecessary computations. The computation of the bases is not shown in Figure C.5, but is done similarly as in Figure C.4.

The simplotope spline set can be initialized by simply adding a spline, which can be done by passing 5-7 arguments to the `eval_SimplotopeSplineSet()` function. All possible arguments are:

1. *nu*: a $1 \times \ell$ array, indicating the dimensions of the layers
2. *Deg*: a $1 \times \ell$ array, indicating the degrees of the layers
3. *V*: a $1 \times \ell$ cell array, containing the vertices of the individual layers
4. *T*: a $1 \times \ell$ cell array, containing the triangulations of the individual layers
5. *Coeffs*: a $J \cdot \hat{d} \times 1$ array, containing the B-Coefficients of the simplotope spline
6. *dimidx*: a $1 \times n$ array, indicating the global dimension ID of all dimensions
7. *sametrias*: a scalar, indicating the simplotope spline ID in the set, sharing the same structure as this spline, which is all of the above except the B-Coefficients.

[2] http://www.boost.org/
[3] http://new.simunova.com/index.html#en-mtl4-index-html

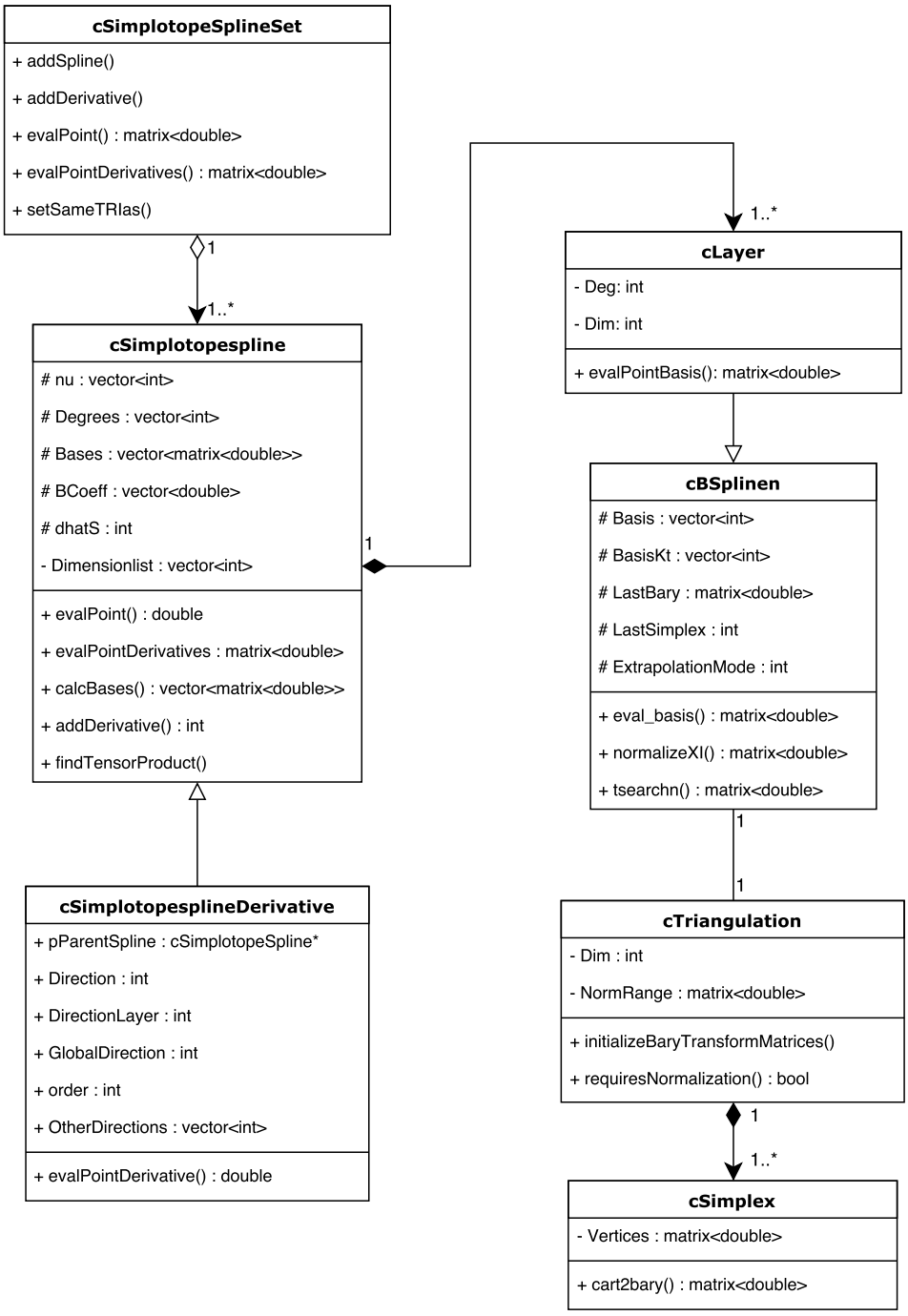


Figure C.3: Class Diagram of the C++ evaluation function.

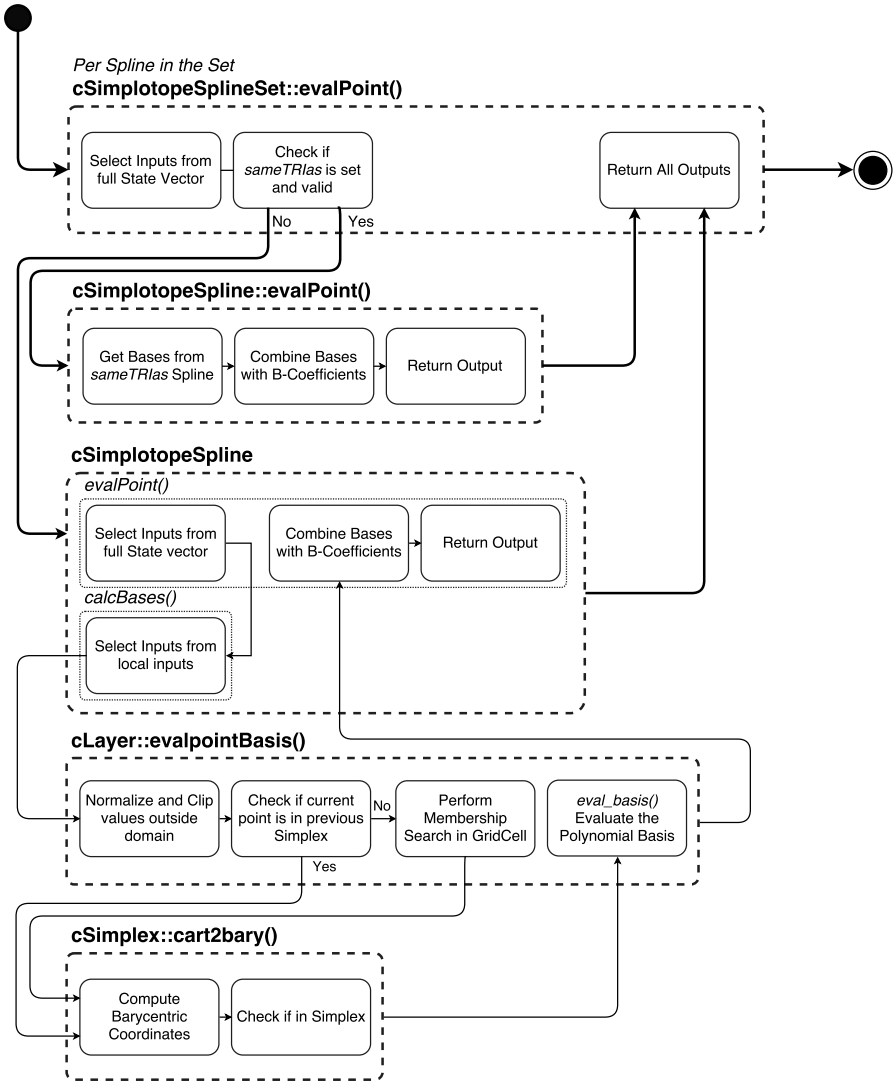


Figure C.4: Activity flow of the C++ evaluation function.

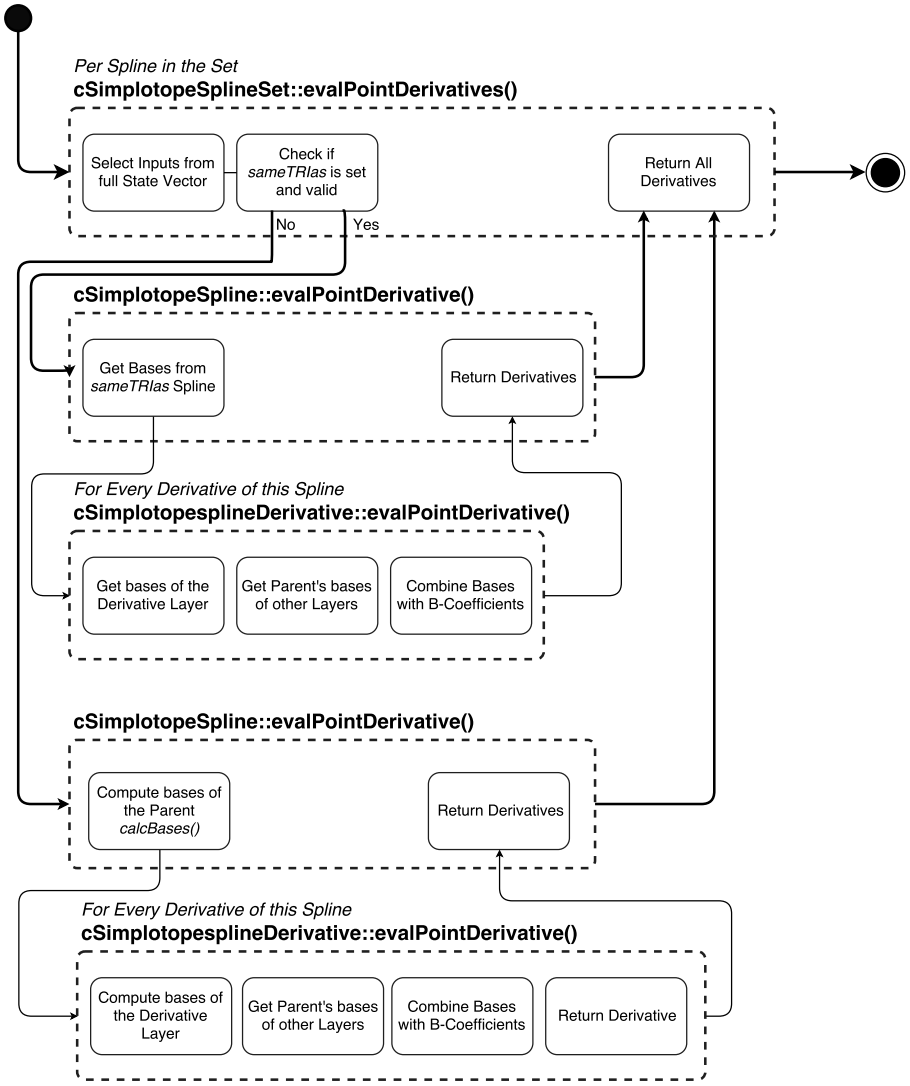


Figure C.5: Activity flow of the C++ evaluation function of the derivatives.

The last two entries are optional. When the dimensions IDs are not set, these are simply starting at 0. When there is no scalar given indicating the *sametrias*, this is set as -1, which means that it is not depending on anything.

When the intention is to evaluate the derivatives of the simplotope models, the same arguments as stated above can be passed to the `evalDerivatives_SimplotopeSplineSet()` function in order to add simplotopes to the spline set. A derivative can be added by passing 2 or 3 arguments:

1. *dir*: a scalar, indicating the direction of the derivative, in local terms
2. *Coeffs*: a $J \cdot \hat{d}_{-1} \times 1$ array, containing the B-Coefficients of the derivative spline
3. *ID*: a scalar, indicating which spline in the spline set this derivative belongs to

The amount of coefficients for the derivative spline is shown as $\hat{d}_{-1}$. The *ID* argument is optional, and when it is not set, the derivative will be added to the last simplotope that has been added to the set.

All classes can be destructed by passing no arguments to its `MEX` function. With this command, all splines are deleted, including the layers, triangulations, and simplices.

Bibliography

[1] Gerard Awanou, Ming-Jun Lai, and Paul Wenston. The Multivariate Spline Method for Scattered Data Fitting and Numerical Solutions of Partial Differential Equations. In *Wavelets and Splines*, pages 24–75. Nashboro Press, Brentwood, Tennessee, 2005. ISBN 0-9728482-6-6.

[2] Dimitri P. Bertsekas and John N. Jn Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, Massachusetts, 1989. ISBN 1-886529-01-9. URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:parallel+and+distributed+computation:+numerical+methods{#}2.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2010. ISSN 1935-8237. doi: 10.1561/2200000016. URL http://www.nowpublishers.com/article/Details/MAL-016.

[4] C. C. De Visser. *Global Nonlinear Model Identification with Multivariate Splines*. PhD thesis, Delft University of Technology, 2011.

[5] C. C. De Visser, Elisabeth Brunner, and Michel Verhaegen. On distributed wavefront reconstruction for large-scale adaptive optics systems. *Journal of the Optical Society of America A*, 33(5):817–831, 2016. ISSN 1520-8532. doi: 10.1364/JOSAA.33.000817.

[6] Wei Deng, Ming Jun Lai, Zhimin Peng, and Wotao Yin. Parallel Multi-Block ADMM with o(1 / k) Convergence. *Journal of Scientific Computing*, 71(2):712–736, 2017. ISSN 08857474. doi: 10.1007/s10915-016-0318-2.

[7] Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. Fast Alternating Direction Optimization Methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014. ISSN 1936-4954. doi: 10.1137/120896219. URL http://epubs.siam.org/doi/10.1137/120896219.

[8] B. S. He, H. Yang, and S. L. Wang. Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities. *Journal of Optimization Theory and Applications*, 106(2):337–356, 2000. ISSN 0022-3239. doi: 10.1023/A:1004603514434. URL http://link.springer.com/10.1023/A:1004603514434.

[9] Roderick J A Little and Donald B Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, Hoboken, New Jersey, 2014. ISBN 9781119013563.

[10] Ismael Matamoros and C. C. De Visser. Incremental Nonlinear Control Allocation for a Tailless Aircraft with Innovative Control Effectors. In *2018 AIAA Guidance, Navigation, and Control Conference*, number January in AIAA SciTech Forum, page 220. American Institute of Aeronautics and Astronautics, jan 2018. ISBN 978-1-62410-526-5. doi: doi:10.2514/6.2018-1116. URL https://doi.org/10.2514/6.2018-1116.

[11] Tim P. Morris, Ian R. White, and Patrick Royston. Tuning multiple imputation by predictive mean matching and local residual draws. *BMC Medical Research Methodology*, 14(1), 2014. ISSN 14712288. doi: 10.1186/1471-2288-14-75.

[12] Yurii Nesterov. A method of solving a convex programming problem with convergence rate O (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[13] Michael A. Niestroy, Kenneth M. Dorsett, and Katherine Markstein. A Tailless Fighter Aircraft Model for Control-Related Research and Development. In *AIAA Modeling and Simulation Technologies Conference*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, jan 2017. ISBN 978-1-62410-451-0. doi: doi:10.2514/6.2017-1757. URL http://arc.aiaa.org/doi/10.2514/6.2017-1757https://doi.org/10.2514/6.2017-1757.

[14] L. G. Sun, C. C. De Visser, and Qiping P. Chu. A New Substitution Based Recursive B-Splines Method for Aerodynamic Model Identification. In Qiping Chu, Bob Mulder, Daniel Choukroun, Erik-Jan van Kampen, Coen de Visser, and Gertjan Looye, editors, *Advances in Aerospace Guidance, Navigation and Control: Selected Papers of the Second CEAS Specialist Conference on Guidance, Navigation and Control*, pages 233–245. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-38253-6.

[15] Martin A. Tanner and Wing Hung Wong. The Calculation of Posterior Distributions By Data Augmentation. *Journal of the American Statistical Association*, 82(398):528–540, 1987. ISSN 01621459. doi: 10.2307/2289457. URL `http://wrap.warwick.ac.uk/24939/`.

[16] I. V. van der Peijl. Physical Splines for Aerodynamic Modelling of Innovative Control Effectors. Master's thesis, Delft University of Technology, 2017.

[17] Tim Visser, C. C. De Visser, and Erik-Jan Van Kampen. Quadrotor System Identification Using the Multivariate Multiplex B-Spline. In *AIAA Atmospheric Flight Mechanics Conference*, number January in AIAA SciTech Forum, pages 1–13, Kissimmee, Florida, jan 2015. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-340-7. doi: doi:10.2514/6.2015-0747. URL `https://doi.org/10.2514/6.2015-0747`.

[18] S. L. Wang and L. Z. Liao. Decomposition method with a variable parameter for a class of monotone variational inequality problems. *Journal of Optimization Theory and Applications*, 109(2):415–429, 2001. ISSN 00223239. doi: 10.1023/A:1017522623963.