

# Nonlinear Multivariate Spline Based Control Allocation for High Performance Aircraft

H.J. Tol<sup>\*</sup>, C.C. de Visser<sup>†</sup>, E. van Kampen<sup>‡</sup>, Q.P. Chu<sup>§</sup>

*Delft University of Technology, Delft, The Netherlands.*

High performance flight control systems based on the nonlinear dynamic inversion (NDI) principle require highly accurate models of aircraft aerodynamics. In general, the accuracy of the internal model determines to what degree the system nonlinearities can be canceled; the more accurate the model, the better the cancellation, and with that, the higher the performance of the controller. In this paper a new control system is presented that combines NDI with multivariate simplex spline based control allocation. We present three control allocation strategies which use novel expressions for the analytical Jacobian and Hessian of the multivariate spline models. Multivariate simplex splines have a higher approximation power than ordinary polynomial models, and are capable of accurately modeling nonlinear aerodynamics over the entire flight envelope of an aircraft. This new method, indicated as SNDI, is applied to control a high performance aircraft (F-16) with a large flight envelope. The simulation results indicate that the SNDI controller can achieve feedback linearization throughout the entire flight envelope, leading to a significant increase in tracking performance compared to ordinary polynomial based NDI.

## Nomenclature

$A_x, A_y, A_z$	specific forces along the body X/Y/Z axis [ $m/s^2$ ]
$C$	dimensionless coefficient
$H$	smoothness matrix
$I$	inertia matrix
$J$	total number of simplices
$\mathcal{J}$	cost function
$S$	wing area [ $m^2$ ]
$S_d^r$	spline space of degree $d$ and continuity order $r$
$\mathcal{T}$	triangulation
$V$	airspeed [ $m/s$ ]
$X$	regression matrix
$Y$	observation vector
$b$	wing span [ $m$ ]
$b(\mathbf{x})$	barycentric transform of point $\mathbf{x}$
$\bar{c}$	mean aerodynamic chord [ $m$ ]
$\bar{q}$	dynamic pressure [ $Pa$ ]
$\hat{d}$	total number of valid permutations
$\mathbf{c}$	B-coefficient vector

<sup>\*</sup>Researcher, Faculty of Aerospace Engineering, Control and Simulation Division, hendrikus.tol@gmail.com, 2629 HS Delft, The Netherlands.

<sup>†</sup>Assistant Professor, Faculty of Aerospace Engineering, Control and Simulation Division, c.c.devisser@tudelft.nl, 2629 HS Delft, The Netherlands.

<sup>‡</sup>Assistant Professor, Faculty of Aerospace Engineering, Control and Simulation Division, E.vanKampen@tudelft.nl, 2629 HS Delft, The Netherlands.

<sup>§</sup>Associate Professor, Faculty of Aerospace Engineering, Control and Simulation Division, q.p.chu@tudelft.nl, 2629 HS Delft, The Netherlands.

$\mathbf{u}$	input vector
$\mathbf{x}$	state vector
$l, m, n$	aerodynamic moment around the body X/Y/Z axis
$p, q, r$	roll, pitch and yaw rate around the body X/Y/Z axis [rad/s]
$p_s$	static pressure [Pa]
$t_j$	simplex j
$u, v, w$	velocity components along the body X/Y/Z axis [m/s]
<i>Subscripts</i>	
$a, e, r, lef$	aileron, elevator, rudder, leading edge flap
<i>Symbols</i>	
$\alpha, \beta$	angle of attack and sideslip angle [rad]
$\epsilon$	residual vector
$\nu$	virtual input
$\tau$	virtual input
$\delta$	control surface deflection [rad]
$\kappa$	multi-index
$\phi, \theta, \psi$	roll, pitch and yaw angle [rad]
$\rho$	air density [ $kg/m^3$ ]

## I. Introduction

Nonlinear dynamic inversion (NDI) is a physical control approach in which the control law is explicitly defined in terms of an internal model [1]. The internal model for the system and input dynamics is used to cancel the nonlinearities after which a single linear controller can be used to control the system. A major advantage of NDI is that gain scheduling is avoided through the entire flight envelope. Furthermore, the simple structure of NDI allows easy and flexible design for all flying modes and is therefore a popular method for aircraft flight control [2, 3]. The NDI controller can be augmented with a control allocation (CA) module in the case that an aircraft has redundant or cross-coupled control effectors [4]. In this case the command variables are the three desired aerodynamic moments, while the actual control effector displacements are determined from the desired moments together with the effector constraints in a constrained optimization problem [4]. It is this particular form of NDI that is the subject of study of this paper.

Since the NDI control law is explicitly defined in terms of the internal model, NDI is sensitive to modeling errors. The accuracy of the internal model determines to what degree the system nonlinearities are canceled; the more accurate the model, the better the cancellation, and with that, the higher the performance of the flight controller. Some of these model inaccuracies can be handled by applying robust control techniques such as structured value ( $\mu$ ) synthesis [1, 5] or incremental NDI [6]. However, significant modeling errors will still lead to unwanted control system behavior.

Currently, most NDI controllers use polynomial structures for the system and input dynamics. It is well known that polynomial models have limited approximation power, which is directly proportional to their degree. As a result, many attempts have been made in the past to increase the accuracy of the onboard models, using for example neural networks [7–9]. In this paper a new approach is presented for increasing the accuracy of onboard models using multivariate simplex spline approximators.

A simplex spline approximator is an analytical function that consists of polynomial basis functions which are each defined on a simplex [10, pp. 18–25]. Any number of basis polynomials can be combined with predefined continuity by combining simplices into a geometric structure called a *triangulation*. The approximation power of simplex spline functions therefore is not only proportional to the polynomial degree, but also to the size and density of the underlying triangulation. The most significant advantages of simplex splines over other function approximators like neural networks is their linear-in-the-parameters property, their numerical stability, their transparency, and the ease with which they can be integrated into standard parameter estimation routines [11].

Multivariate simplex splines have recently been used in a framework for aerodynamic model identification [12–14], where it was shown that they can more accurately approximate both local and global scale system nonlinearities than methods based on ordinary polynomials. The proven advantages of the simplex splines as

powerful, numerically stable, and transparent nonlinear function approximators makes them well suited to replace current onboard models, thereby improving the performance and robustness of nonlinear model-based flight control systems.

Until now, however, no attempt has been made to design a flight controller that uses onboard simplex spline models. As it turns out, integrating a multivariate simplex spline based aerodynamic model into an inversion based flight control system is not trivial. The basis polynomials of the simplex splines are defined locally on each simplex in terms of barycentric coordinates instead of globally in terms of Cartesian coordinates [10, pp.18-25]. A direct consequence of this is that the simplex spline basis polynomials are non-affine in the control inputs, requiring a special coordinate transformation scheme to relate them to the aircraft states and control inputs [15]. See also Appendix B for an example of the non-affinity of the simplex basis polynomials.

The main contribution of this paper is a new nonlinear control scheme, indicated as SNDI, that combines nonlinear dynamic inversion with control allocation based on onboard simplex spline models. This contribution requires the development of new CA strategies that can be applied to simplex spline models that are non-affine functions of the aircraft states and control inputs. In this paper, three new CA strategies for simplex spline models are presented; a linear, a successive linear, and a fully nonlinear strategy.

The SNDI control method is demonstrated using a high fidelity F-16 simulation with which a number of high amplitude maneuvers are performed in nonlinear regions of the flight envelope. It is shown that SNDI results in higher reference tracking performance than ordinary polynomial NDI, in particular when performing high amplitude maneuvers in nonlinear regions of the flight envelope such as the high angle of attack and high angle of sideslip regions. While the current SNDI focuses primarily on reducing static aerodynamic modeling errors, it should be seen as the first step towards a forthcoming spline based *adaptive* nonlinear flight control system of the sort presented in [8,9].

The paper has the following outline: In Sec. II, the aircraft model used in this study is described. In Sec. III a preliminary on multivariate simplex splines is given. In Sec. IV an overview is given of the SNDI control approach, which is the augmentation of NDI with control allocation based on the onboard spline model. The NDI flight control design is given in Sec. V. In Sec. VI the F-16 aerodynamic model is identified using both simplex splines and polynomial model structures. In Sec. VII the three new approaches to control allocation based on the onboard spline model are presented. Finally, in Sec. VIII the spline based controller is evaluated and compared with a polynomial based controller followed by conclusions in section IX

## II. Aircraft Model

In this section the simulation model that will be used in the remainder of this work is introduced. The aircraft to be controlled is a model of the F-16 fighter aircraft from NASA which is based on a set of data tables based on wind tunnel measurements [16]. This model is used to generate simulated aerodynamic force and moment measurements which are used to estimate the multivariate spline based aerodynamic models in Sec. VI. The model has the traditional aerodynamic control surfaces: elevator, ailerons and rudders for pitch, roll and yaw control. In addition, the leading edge flap is scheduled with angle of attack and  $\frac{\bar{q}}{p_s}$  to optimize performance and has the following relationship [16]:

$$\delta_{lef} = 1.38 \frac{2s + 7.25}{s + 7.25} \alpha - 9.05 \frac{\bar{q}}{p_s} + 1.45 \quad (1)$$

Models for the actuators are included in the form of first order lags:

$$\dot{u} = \frac{1}{\tau} (u_{com} - u) \quad (2)$$

In which the commanded input is bounded by  $u_{min} \leq u_{com} \leq u_{max}$  and the deflection rate is bounded by  $|\dot{u}| \leq \dot{u}_{lim}$ . The time constants and actuator limits are listed in table 1 [16] and [17, pp.633-664]. For simulating the response and for flight control design the flat earth, body axis six degree of freedom equations of motion are used [17, pp. 107-116]. No external disturbances like wind gusts are added to the models and the sensor information is considered as not contaminated. All simulation are performed with a sample frequency of 100Hz.

	deflection limit	rate limit	time constant
Elevator	$\pm 25.0^\circ$	$60^\circ/\text{s}$	0.0495 s lag
Ailerons	$\pm 21.5^\circ$	$80^\circ/\text{s}$	0.0495 s lag
Rudder	$\pm 30.0^\circ$	$120^\circ/\text{s}$	0.0495 s lag
leading edge flap	$0^\circ - 25^\circ$	$25^\circ/\text{s}$	0.136 s lag

**Table 1. Actuator model [16], [17, pp.633-664]**

### III. Preliminaries on Multivariate Simplex Splines

This section serves as introduction to the general theory of multivariate simplex splines and the techniques that can be used for aerodynamic model identification using simplex splines. These techniques are based on the work presented in [11] and [18]. For a more in-depth coverage of simplex spline theory we refer to the work by Lai and Schumaker [10]. Additionally, a practical example of the use of multivariate simplex splines for scattered data approximation is presented in Appendix A.

#### A. Simplex Spline Functions and Spline Spaces

A simplex spline function consists of a set of basis polynomials of degree  $d$ , each defined on an individual simplex with predefined continuity between the simplices  $t_j$ :

$$s(\mathbf{x}) = \delta_1(\mathbf{x})p^{t_1}(\mathbf{x}) + \delta_2(\mathbf{x})p^{t_2}(\mathbf{x}) + \cdots + \delta_J(\mathbf{x})p^{t_J}(\mathbf{x}) = \sum_{j=1}^J \delta_j(\mathbf{x})p^{t_j}(\mathbf{x}) \quad (3)$$

with  $J$  the total number of simplices and with  $\delta_j(\mathbf{x})$  a membership function that relates data point  $\mathbf{x}$  to the simplex in which it is defined:

$$\delta_j(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in t_j \\ 0 & \text{if } \mathbf{x} \notin t_j \end{cases} \quad (4)$$

The approximation power of spline functions is partly determined by the triangulation structure. A triangulation is a partitioning of a domain into a set of  $J$  non-overlapping simplices:

$$\mathcal{T} := \bigcup_{j=1}^J t_j, \quad t_i \cap t_j \in \{\emptyset, \tilde{t}\}, \quad \forall t_i, t_j \in \mathcal{T} \quad (5)$$

with the edge simplex  $\tilde{t}$  a  $k$ -simplex with  $0 \leq k \leq n-1$ . The use of spline spaces provide a convenient notation for stating the degree, continuity and triangulation structure of a spline solution without having to specify individual spline functions [10, pp.127-141]:

$$S_d^r(\mathcal{T}) := \{s \in C^r(\mathcal{T}) : s|_t \in \mathcal{P}_d, \quad \forall t \in \mathcal{T}\} \quad (6)$$

with  $s$  the  $n$ -variate simplex spline function of degree  $d$  and continuity order  $r$  on the triangulation  $\mathcal{T}$  and with  $\mathcal{P}_d$  the space of all polynomials of total degree  $d$ .

#### B. Barycentric Coordinates

The individual basis polynomials of the spline function in eq. (3) are defined on simplices and expressed in terms of barycentric coordinates:  $p^t(b(\mathbf{x}))$ . The  $n$ -simplex  $t$  is defined as the convex hull of a set of  $n+1$  unique, non-degenerate points in  $n$ -dimensional space:

$$t = \langle \mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n \rangle \quad (7)$$

The barycentric coordinate system is a local coordinate system with respect to the  $n$ -simplex  $t$ . Every point  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  can be described in terms of a unique weighted vector sum of the vertices of simplex  $t$ :

$$\mathbf{x} = \sum_{i=0}^n b_i \mathbf{v}_i, \quad \sum_{i=0}^n b_i = 1 \quad (8)$$

Using these properties the barycentric coordinate  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  of  $\mathbf{x}$  with respect to the  $n$ -simplex  $t$  can be explicitly calculated with:

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} (\mathbf{v}_1 - \mathbf{v}_0) & (\mathbf{v}_2 - \mathbf{v}_0) & \cdots & (\mathbf{v}_n - \mathbf{v}_0) \end{bmatrix}^{-1} (\mathbf{x} - \mathbf{v}_0) = \Lambda (\mathbf{x} - \mathbf{v}_0) \quad (9)$$

and

$$b_0 = 1 - \sum_{i=1}^n b_i \quad (10)$$

### C. The B-form

Each polynomial  $p^{t_j}(b(\mathbf{x}))$  in eq. (3) is expressed in the B-form:

$$p^{t_j}(b(\mathbf{x})) = \sum_{|\kappa|=d} c_{\kappa}^{t_j} \frac{d!}{\kappa!} \prod_{i=0}^n b_i^{\kappa_i} = \sum_{|\kappa|=d} c_{\kappa}^{t_j} B_{\kappa}^d(b(\mathbf{x})) \quad (11)$$

with  $c_{\kappa}$  the B-coefficient and  $\kappa$  the multi-index defined as:

$$\kappa := (\kappa_0, \kappa_1, \dots, \kappa_n) \in \mathbb{N}^{n+1} \quad (12)$$

The multi-index has the following properties:

$$|\kappa| = \kappa_0 + \kappa_1 + \dots + \kappa_n = d \quad (13)$$

$$\kappa! = \kappa_0! \kappa_1! \cdots \kappa_n! \quad (14)$$

The elements of the multi-index are sorted lexicographically [19]:

$$\kappa \in \{(d, 0, 0 \dots, 0), (d-1, 1, 0 \dots, 0), (d-1, 0, 1, \dots, 0), \dots, (0, \dots, 0, 1, d-1), (0, \dots, 0, 0, d)\} \quad (15)$$

The total number of valid permutations of  $\kappa$ , and therefore total number of individual basis polynomials, is  $\hat{d}$ :

$$\hat{d} = \binom{d+n}{n} = \frac{(d+n)!}{n!d!} \quad (16)$$

Eq. (11) can also be written in vector form [11]. Define the vector of basis polynomials as:

$$B_{t_j}^d(b(\mathbf{x})) := \begin{bmatrix} B_{d,0,\dots,0}^d(b(\mathbf{x})) & B_{d-1,1,\dots,0}^d(b(\mathbf{x})) & \cdots & B_{0,\dots,0,d}^d(b(\mathbf{x})) \end{bmatrix} = [B_{\kappa}^d(b(\mathbf{x}))]_{|\kappa|=d} \in \mathbb{R}^{1 \times \hat{d}} \quad (17)$$

and the vector of B-coefficients:

$$\mathbf{c}^{t_j} := [c_{\kappa}^{t_j}]_{|\kappa|=d} \in \mathbb{R}^{\hat{d} \times 1} \quad (18)$$

With these definitions the per-simplex B-form in vector formulation is:

$$p^{t_j}(b(\mathbf{x})) = B_{t_j}^d(b(\mathbf{x})) \mathbf{c}^{t_j} \quad (19)$$

### D. The Regression Model using B-form Polynomials

This section presents the linear regression model structure from [11] for spline functions using the B-form polynomials. Using eq. (3) and eq. (19), the B-form polynomials can be used as regressors for a new observation pair  $(\mathbf{x}(i), y(i))$  as follows:

$$y(i) = \sum_{j=1}^J \delta_j(\mathbf{x}(i)) B_{t_j}^d(b(\mathbf{x}(i))) \mathbf{c}^{t_j} + \epsilon(i) \quad (20)$$

To improve readability the following shorthand notation is used for eq. (20):

$$y(i) = \sum_{j=1}^J \delta_j(i) B_{t_j}^d(i) \mathbf{c}^{t_j} + \epsilon(i) \quad (21)$$

These shorthand notations are used through the rest of the paper. Eq. (21) can be restated in matrix form. First define a per simplex  $\hat{d} \times \hat{d}$  diagonal data membership matrix for observation  $i$ :

$$D_{t_j}(i) = [(\delta_j(i))_{q,q}]_{q=1}^{\hat{d}} \quad (22)$$

The block diagonal full triangulation data membership matrix  $D$  for a single observation is a matrix with  $D_{t_j}$  blocks on the main diagonal:

$$D(i) = [(D_{t_j}(i))_{j,j}]_{j=1}^J \quad (23)$$

Using the per simplex vector of B-form basis polynomials from eq. (17), the full triangulation basis function vector  $B^d(i)$  for observation  $i$  is defined as:

$$B^d(i) := \begin{bmatrix} B_{t_1}^d(i) & B_{t_2}^d(i) & \cdots & B_{t_J}^d(i) \end{bmatrix} = \begin{bmatrix} B_{t_j}^d(i) \end{bmatrix}_{j=1}^J \in \mathbb{R}^{1 \times J \cdot \hat{d}} \quad (24)$$

Using the per simplex vector of B-form basis polynomials from eq. (18), the full triangulation vector of B-coefficients is constructed as:

$$\mathbf{c} = [\mathbf{c}^{t_j}]_{j=1}^J \in \mathbb{R}^{J \cdot \hat{d} \times 1} \quad (25)$$

With the definitions in eqs (23), (24) and (25), eq. (20) can be written as:

$$y(i) = B^d(i) D(i) \mathbf{c} + \epsilon(i) = X(i) \mathbf{c} + \epsilon(i) \quad (26)$$

which for all observations results in a linear regression scheme for multivariate simplex splines:

$$\mathbf{Y} = X \mathbf{c} + \epsilon \quad (27)$$

## E. Spline Model Estimation

Eq. (27) can be solved using an equality constrained ordinary least squares (LS) estimator. The LS problem needs to be constrained in order to guarantee continuity between the simplices. The B-coefficients can be estimated by solving the constrained least squares problem:

$$\min_{\mathbf{c}} \frac{1}{2} (\mathbf{Y} - X \mathbf{c})^T (\mathbf{Y} - X \mathbf{c}) \quad \text{subject to} \quad H \mathbf{c} = 0 \quad (28)$$

with  $H$  the smoothness matrix to guarantee continuity between the simplices. Using Lagrange multipliers, this leads to the following LS estimator:

$$\begin{bmatrix} \hat{\mathbf{c}} \\ \hat{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} X^T X & H^T \\ H & 0 \end{bmatrix}^+ \begin{bmatrix} X^T \mathbf{Y} \\ 0 \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} \begin{bmatrix} X^T \mathbf{Y} \\ 0 \end{bmatrix} \quad (29)$$

The estimated B-coefficients can be calculated as follows:

$$\hat{\mathbf{c}} = \mathbf{C}_1 X^T \mathbf{Y} \quad (30)$$

The smoothness matrix is computed using de Boor's continuity equations. The formulation in [20] and [10, pp.133-135] is used for the continuity equations for degree  $r$  between the edges of two neighboring simplices  $t_i$  and  $t_j$ :

$$c_{(\kappa_0, \dots, \kappa_{n-1}, m)}^{t_i} = \sum_{|\gamma|=m} c_{(\kappa_0, \dots, \kappa_{n-1}, 0) + \gamma}^{t_j} B_{\gamma}^m(\mathbf{v}^*), \quad 0 \leq m \leq r \quad (31)$$

with  $\gamma = (\gamma_0, \gamma_1, \dots, \gamma_n)$  a multi-index independent of  $\kappa$  and  $\mathbf{v}^*$  the out of edge vertex of simplex  $t_j$ . Using the valid permutations for the multi-indices  $\kappa$  and  $\gamma$  and combining the continuity equations for all edges the continuity equations can be written in vector form:

$$H \mathbf{c} = 0 \quad (32)$$

with  $H \in \mathbb{R}^{R \cdot E \times J \cdot \hat{d}}$ ,  $R$  the number of continuity conditions per edge and  $E$  the number of edges in the triangulation.

## IV. Spline Based NDI Control: Component Overview

The approach used for spline based NDI control is the augmentation of NDI with control allocation based on the onboard aerodynamic spline model. The control diagram is shown in figure 1. All aircraft control laws based on NDI can be written in terms of required control moments when controlling the attitude using aerodynamic control surface deflections or in terms of control forces when controlling the airspeed using the throttle. The control forces and moments can be seen as a virtual control input  $\tau$  which have to be translated into actuator settings. This is also known as the process of control allocation [4]. When the NDI control law is defined in terms of the aircraft stability and control derivatives, the NDI control algorithm automatically involves some form of control allocation [4, 21]. By using polynomial models affine in the control input, the actuator settings can be calculated directly. However, when using non-affine simplex spline based aerodynamic models in terms of local barycentric coordinates, the actuator setting cannot be calculated directly and the NDI structure requires a separate control allocation module. See also the example in Appendix B for an illustration of the non-affinity of spline models. The separation of the control allocation task from the NDI control laws allows the development of general spline model based allocation strategies which are discussed in Sec. VII. This section focuses on the combined control structure, the formulation of the control allocation problem and existing solution methods.

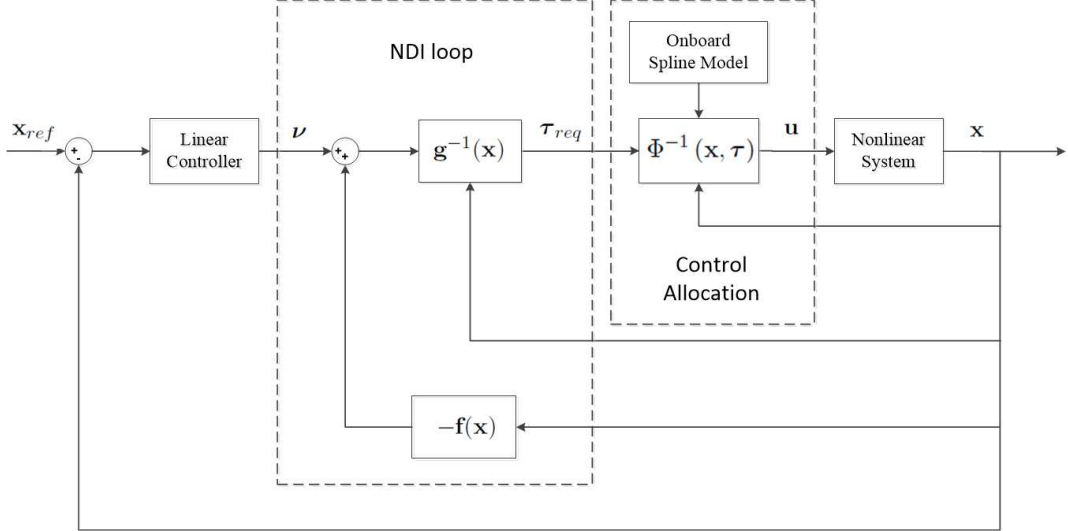


Figure 1. Combined control structure: NDI inner loop and linear control outer loop combined with control allocation.

### A. Nonlinear Dynamic Inversion

Consider the aircraft state equations in the input affine form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\boldsymbol{\tau} \quad (33)$$

$$\boldsymbol{\tau} = \boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}). \quad (34)$$

with  $\mathbf{x} \in \mathbb{R}^n$  the state vector,  $\mathbf{u} \in \mathbb{R}^m$  the control input vector and  $\boldsymbol{\tau} \in \mathbb{R}^l$  the virtual controls assumed to be a nonlinear function of the aircraft state and control input. The crux of NDI is to solve for the input  $\boldsymbol{\tau}$  by introducing an outer loop control input  $\boldsymbol{\nu}$ :

$$\boldsymbol{\tau}_{req} = \mathbf{g}^{-1}(\mathbf{x})(\boldsymbol{\nu} - \mathbf{f}(\mathbf{x})) \quad (35)$$

Which results in a closed-loop system with a decoupled linear input-output relation:

$$\dot{\mathbf{x}} = \boldsymbol{\nu} \quad (36)$$

NDI is based on the assumption that the internal model is exactly known such that the model is fully linearized. However this assumption is not realistic in practice and there will also be an inversion error

related to the feedback linearization. In this case the closed-loop system is given by:

$$\dot{\mathbf{x}} = \boldsymbol{\nu} + \boldsymbol{\Delta} \quad (37)$$

with  $\boldsymbol{\Delta}$  the inversion error. The required virtual control input  $\boldsymbol{\tau}_{req}$  is either the required moment for rotational control or the required force for translational control. For example, consider the aircraft rate control problem:

$$\dot{\boldsymbol{\omega}} = I^{-1}\mathbf{M} - I^{-1}\boldsymbol{\omega} \times I\boldsymbol{\omega} \quad (38)$$

Applying NDI results in the following control law for the required control moment:

$$\mathbf{M}_{req} = I (\boldsymbol{\nu} + I^{-1}\boldsymbol{\omega} \times I\boldsymbol{\omega}) \quad (39)$$

The required control moment has to be translated into control surface deflections based on the onboard model for  $\mathbf{M}$ . The accuracy of the onboard model determines to what extent the nonlinearities are canceled: the more accurate the model, the smaller the inversion error  $\boldsymbol{\Delta}$ , and with that, the higher the controller performance.

Instead of using the frequently used polynomial model structures, the model for  $\mathbf{M}$  of the F-16 is identified using simplex splines (see section VI). Simplex splines provide a significant increase in modeling accuracy compared to polynomials. In this paper the effect of the increased model accuracy on the NDI controller performance is investigated.

## B. Control Allocation

The mapping in eq. (34) maps the physical control inputs to the virtual controls:

$$\boldsymbol{\tau} = \boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}) : \mathbb{R}^m \rightarrow \mathbb{R}^l \quad (40)$$

The control allocation problem considers the inversion of this mapping:

$$\mathbf{u} = \boldsymbol{\Phi}^{-1}(\mathbf{x}, \boldsymbol{\tau}) : \mathbb{R}^l \rightarrow \mathbb{R}^m \quad (41)$$

The control allocation problem can be stated as: Given a virtual command  $\boldsymbol{\tau}$ , determine  $\mathbf{u}$  satisfying the actuator position and rate constraints, such that  $\boldsymbol{\tau} = \boldsymbol{\Phi}(\mathbf{x}, \mathbf{u})$ . The input will be determined based on the onboard spline model for  $\boldsymbol{\Phi}(\mathbf{x}, \mathbf{u})$ . Control allocation problems are often formulated as optimization problems with a least squares objective function subject to actuator constraints. With optimization based methods a cost function that relates control effort and the required demand is minimized. In [22, 23] three formulations are given:

1. Error minimization problem:

$$\min_{\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}} \mathcal{J} = \|\boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}) - \boldsymbol{\tau}_{req}\| \quad (42)$$

2. Control minimization problem:

$$\min_{\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}} \mathcal{J} = \|\mathbf{u}\| \quad \text{s.t.} \quad \boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}) = \boldsymbol{\tau}_{req} \quad (43)$$

3. Mixed optimization problem

$$\min_{\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}} \mathcal{J} = \|\boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}) - \boldsymbol{\tau}_{req}\| + \epsilon \|\mathbf{u}\| \quad (44)$$

The controls are constrained by their minimum ( $\underline{\mathbf{u}}$ ) and maximum ( $\bar{\mathbf{u}}$ ) values.

Most existing methods derived for over-actuated systems ( $l < m$ ) for solving eqs. (42)-(44) consider linear effector models of the form:

$$\boldsymbol{\tau} = \boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}) = G\mathbf{u} \quad (45)$$

with  $G$  an  $l \times m$  matrix. See [22, 23] for a survey on optimization based control allocation approaches for linear effector models. A popular and efficient solution for real-time control allocation is the pseudo inverse solution, see e.g. [21, 24, 25] for applications. When the actuator constraints are dropped, the solution to the



$l_2$  norm control effort minimization problem in eq. (43) using the linear effector model in eq. (45) has a pseudo inverse solution:

$$\mathbf{u} = G^+ \boldsymbol{\tau} \quad (46)$$

with the pseudo inverse calculated as:

$$G^+ = G^T (GG^T)^{-1} \quad (47)$$

Because  $GG^T$  can become singular it has to be replaced with the regularized matrix  $GG^T + \epsilon I$ . In [26,27] a redistribution scheme is used to account for the actuator limits in which all actuators that violate their bounds in the pseudo-inverse solution are saturated and removed from the optimization. Then, the problem is resolved with the remaining actuators as free variables. The procedure is repeated until all components have reached their limits, or until the solution of the reduced least-squares problem satisfies the constraints.

## V. NDI Flight Control Design

Two inversion loops have been implemented using a time-scale separated design [1]: an inner rate control loop and an outer aerodynamic angle control loop. The control setup is shown in figure 2. The controlled variables are the roll angle  $\phi$ , angle of attack  $\alpha$  and sideslip angle  $\beta$ . With this control setup maneuvers can be performed with zero sideslip. Furthermore it can be used to operate at a constant nonzero sideslip angle to compensate for the asymmetry in the case of crosswind or in the case of an asymmetric failure. To avoid unachievable commands due to the actuator constraints first order lag prefilters are used for the command variables:  $H_{pf} = \frac{1}{\sigma s + 1}$ . The prefilter time constants are chosen such that fast tracking is achieved while avoiding command saturation as much as possible. Only proportional control is used for feedback on the roll, pitch and yaw channels. The controller gains and prefilter time constants are listed in table 2. The inner and outer loop control laws are described in the next two sections.

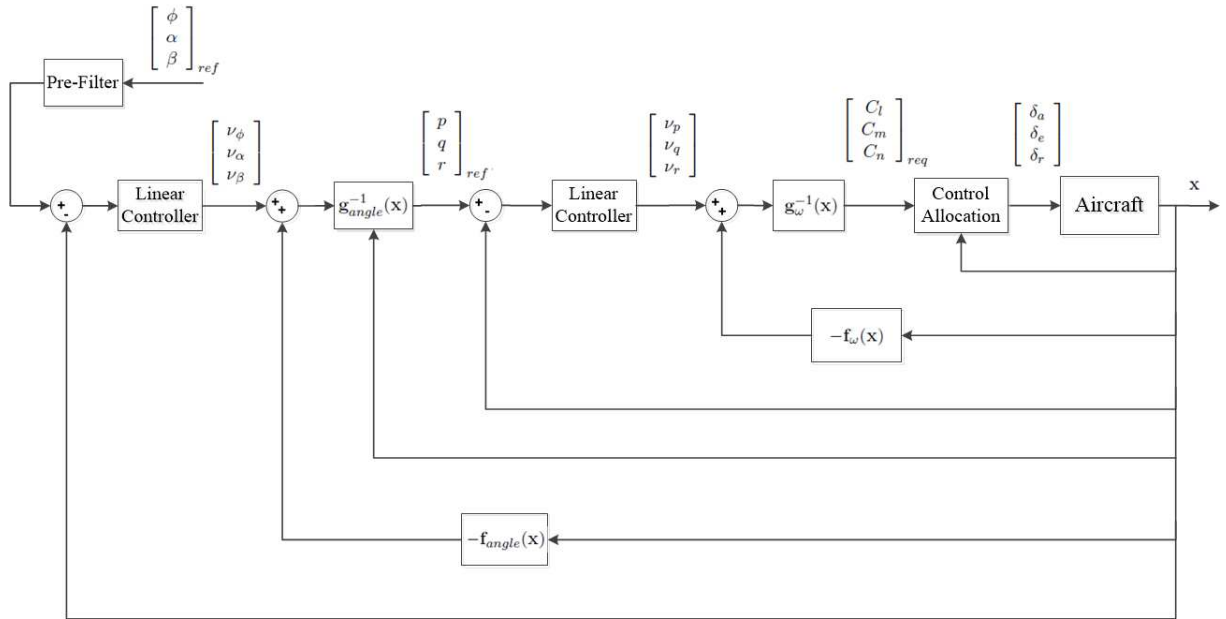


Figure 2. Control setup. An inner rate NDI loop combined with an aerodynamic angle outer NDI loop

Table 2. Prefilter time constants and controller gains

$\sigma_\phi = 0.2$	$k_\phi = 2$	$k_p = 10$
$\sigma_\alpha = 0.4$	$k_\alpha = 2$	$k_q = 10$
$\sigma_\beta = 0.2$	$k_\beta = 2$	$k_r = 10$

### A. Body Angular Rate Inner Loop

In the inner loop the system is influenced by commanding the moments of the aircraft. The inner loop quantities are the body angular rates:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \begin{bmatrix} l \\ m \\ n \end{bmatrix} - I^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (48)$$

Rewriting the aircraft dynamics into the form of eq. (35) gives:

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} = \frac{I}{\frac{1}{2}\rho V^2 S} \begin{bmatrix} b & 0 & 0 \\ 0 & \bar{c} & 0 \\ 0 & 0 & b \end{bmatrix}^{-1} \left\{ \begin{bmatrix} \nu_p \\ \nu_q \\ \nu_r \end{bmatrix} + I^{-1} \left( \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \right\} \quad (49)$$

### B. Aerodynamic Angle Outer Loop

The inner loop NDI, combined with the dynamics of the aircraft are now considered as one system that can be influenced by commanding the angular rates. The outer loop quantities are the roll angle  $\phi$ , angle of attack  $\alpha$  and sideslip angle  $\beta$ . The dynamics are expressed in terms of required body angular rates. For the roll angle this results in:

$$\dot{\phi} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (50)$$

$$= g_\phi(\mathbf{x})\boldsymbol{\omega} \quad (51)$$

For the angle of attack:

$$\begin{aligned} \dot{\alpha} &= \frac{d}{dt} \left( \tan^{-1} \frac{w}{u} \right) = \frac{u\dot{w} - w\dot{u}}{u^2 + w^2} \\ &= \frac{1}{u^2 + w^2} [u(A_z + g \cos \theta \cos \phi) - w(A_x - g \sin \theta)] + \begin{bmatrix} \frac{-uv}{u^2 + w^2} & 1 & \frac{-vw}{u^2 + w^2} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ &= f_\alpha(\mathbf{x}) + g_\alpha(\mathbf{x})\boldsymbol{\omega} \end{aligned} \quad (52)$$

and for the sideslip angle this gives:

$$\begin{aligned} \dot{\beta} &= \frac{d}{dt} \left( \sin \frac{v}{V} \right) = \frac{V\dot{v} - v\dot{V}}{V\sqrt{u^2 + w^2}} \\ &= \frac{1}{\sqrt{u^2 + w^2}} \left[ \frac{-uv}{V^2} (A_x - g \sin \theta) + \left( 1 - \frac{v}{V^2} \right) (A_y + g \sin \phi \cos \theta) - \frac{vw}{V^2} (A_z + g \cos \phi \cos \theta) \right] \\ &\quad + \begin{bmatrix} \frac{w}{\sqrt{u^2 + w^2}} & 0 & \frac{-u}{\sqrt{u^2 + w^2}} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ &= f_\beta(\mathbf{x}) + g_\beta(\mathbf{x})\boldsymbol{\omega} \end{aligned} \quad (53)$$

Combining eqs. (50), (52) and (53) and rewrite into the form of eq. (35) gives:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} g_\phi(\mathbf{x}) \\ g_\alpha(\mathbf{x}) \\ g_\beta(\mathbf{x}) \end{bmatrix}^{-1} \left( \begin{bmatrix} \nu_\phi \\ \nu_\alpha \\ \nu_\beta \end{bmatrix} - \begin{bmatrix} f_\phi(\mathbf{x}) \\ f_\alpha(\mathbf{x}) \\ f_\beta(\mathbf{x}) \end{bmatrix} \right) \quad (54)$$

## VI. Identification of the F-16 Aerodynamic Model

In this section methods from [11,13,14] are used to identify the F-16 aerodynamic model using multivariate splines and ordinary polynomials. The data used to identify the aerodynamic models is generated with a high-fidelity wind tunnel dataset of the F-16. This wind tunnel dataset should be seen as the 'real' F-16 aerodynamics, which are approximated with the multivariate spline and polynomial models. The two identified models are used for the performance assessment in Sec. VIII.B in which the spline based NDI controller is compared to a polynomial based NDI controller.

The required variables to be estimated are the moment coefficients  $C_l$ ,  $C_m$  and  $C_n$ . In Sec. B the model is identified with simplex spline structures and polynomial structures using a training dataset consisting of 60000 scattered points which is generated with the wind tunnel model. In Sec. C the polynomial and spline model are compared and validated using a validation dataset consisting of 10000 scattered points.

### A. Simulated Measurement Data

The NASA wind tunnel data tables are used to generate simulated measurement data. The training and validation datasets are obtained by randomly generating scattered datapoints within the following independent variable ranges:

$$\begin{array}{llll}
 -10^\circ & \leq \alpha \leq & 45^\circ & 50 \text{ m/s} \leq V \leq 300 \text{ m/s} \\
 -30^\circ & \leq \beta \leq & 30^\circ & -21.5^\circ \leq \delta_a \leq 21.5^\circ \\
 -90^\circ/s & \leq p \leq & 90^\circ/s & -25.0^\circ \leq \delta_e \leq 25.0^\circ \\
 -90^\circ/s & \leq q \leq & 90^\circ/s & -30.0^\circ \leq \delta_r \leq 30.0^\circ \\
 -90^\circ/s & \leq r \leq & 90^\circ/s & 0^\circ \leq \delta_{lef} \leq 25.0^\circ
 \end{array} \tag{55}$$

This results in a 10 dimensional dataset for the independent variables which is used to compute the dependent variables  $C_l$ ,  $C_m$  and  $C_n$  through the NASA wind tunnel model. The complete dataset may include physically infeasible data outside the operating region. However, this will not affect the identified model within the valid flight envelope.

### B. Model Identification

The following model structures were assumed for the moment coefficients:

$$C_m(\alpha, \beta, \tilde{q}, \delta_e, \delta_{lef}) = C_m(\alpha, \beta, \delta_e) + C_{m_{\delta_{lef}}}(\alpha, \beta)\delta_{lef} + C_{m_q}(\alpha)\tilde{q} + C_{m_q\delta_{lef}}(\alpha)\delta_{lef}\tilde{q} \tag{56}$$

$$\begin{aligned}
 C_l(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef}) &= C_l(\alpha, \beta) + C_{l_{\delta_{lef}}}(\alpha, \beta)\delta_{lef} + C_{l_{\delta_a}}(\alpha, \beta)\delta_a + C_{l_{\delta_r}}(\alpha, \beta)\delta_r + \\
 &C_{l_r}(\alpha)\tilde{r} + C_{l_r\delta_{lef}}(\alpha)\delta_{lef}\tilde{r} + C_{l_p}(\alpha)\tilde{p} + C_{l_p\delta_{lef}}(\alpha)\delta_{lef}\tilde{p}
 \end{aligned} \tag{57}$$

$$\begin{aligned}
 C_n(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef}) &= C_n(\alpha, \beta) + C_{n_{\delta_{lef}}}(\alpha, \beta)\delta_{lef} + C_{n_{\delta_a}}(\alpha, \beta)\delta_a + C_{n_{\delta_r}}(\alpha, \beta)\delta_r + \\
 &C_{n_r}(\alpha)\tilde{r} + C_{n_r\delta_{lef}}(\alpha)\delta_{lef}\tilde{r} + C_{n_p}(\alpha)\tilde{p} + C_{n_p\delta_{lef}}(\alpha)\delta_{lef}\tilde{p}
 \end{aligned} \tag{58}$$

where

$$\tilde{p} = pb/2V, \quad \tilde{q} = q\bar{c}/2V, \quad \tilde{r} = rb/2V$$

Each modeling function in eqs. (56), (57), (58) is estimated using simplex splines and polynomials over the independent variable ranges in eq. (55). The objective is to make the best possible polynomial model such that a valid comparison between the polynomial based NDI controller and the SNDI controller can be made. In [28] a polynomial model is created from a slightly simplified version [17, pp.633-664] of the original wind tunnel database using a modeling technique based on orthogonal modeling functions [29]. The regression structures of this polynomial model are used as initial model structures to estimate the database after which more regressors are added to further improve the model. The resulting polynomial structures for each modeling function are listed in the third column of table 3.

For estimating the polynomial model, all observations are combined in the observation matrix  $\mathbf{Y}$ , and the regressors are combined in the regression matrix  $X$  resulting in the standard regression form (eq. (27)).

Using an ordinary least squares estimator for the model parameters gives:

$$\mathbf{C} = (X_p^T X_p)^{-1} X_p^T \mathbf{Y} \quad (59)$$

with  $X_p$  the regression matrix for the polynomial model. For the multivariate spline model each sub-function is estimated with a spline function. For the pitch moment coefficient  $C_m$  this results in:

$$s(\alpha, \beta, \tilde{q}, \delta_e, \delta_{lef}) = s_1(\alpha, \beta, \delta_e) + s_2(\alpha, \beta) \delta_{lef} + s_3(\alpha) \tilde{q} + s_4(\alpha) \delta_{lef} \tilde{q} \quad (60)$$

with:

$$s_1 \in S_{d1}^{r1}(\mathcal{T}), \quad s_2 \in S_{d2}^{r2}(\mathcal{T}), \quad s_3 \in S_{d3}^{r3}(\mathcal{T}), \quad s_4 \in S_{d4}^{r4}(\mathcal{T}) \quad (61)$$

The selected spline spaces for each modeling function are listed in the second column of table 3. Using eq. (26) the model structure for  $C_m$  in eq. (60) can be written in linear regression form as follows:

$$\begin{aligned} y(i) &= \begin{bmatrix} B_1^{d1}(i) D_1(i) & B_2^{d2}(i) D_2(i) \delta_{lef}(i) & B_3^{d3}(i) D_3(i) \tilde{q}(i) & B_4^{d4}(i) D_4(i) \delta_{lef}(i) \tilde{q}(i) \end{bmatrix} \begin{bmatrix} \mathbf{c}_1^T & \mathbf{c}_2^T & \mathbf{c}_3^T & \mathbf{c}_4^T \end{bmatrix}^T \\ &= X(i) \mathbf{c} \end{aligned} \quad (62)$$

which for all observations results in the standard regression form (eq. (27)). The B-coefficient vectors  $\mathbf{c}_1$  to  $\mathbf{c}_4$  for this regression problem can be solved using the constrained least squares estimator from eq. (30). To guarantee continuity between the simplices, a global smoothness matrix needs to be defined to combine the continuity conditions for all four spline regressors. The global smoothness matrix in this case is [12]:

$$H_g = \begin{bmatrix} H_1 & 0 & 0 & 0 \\ 0 & H_2 & 0 & 0 \\ 0 & 0 & H_3 & 0 \\ 0 & 0 & 0 & H_4 \end{bmatrix} \in \mathbb{R}^{\sum_{i=1}^4 R_i \cdot E_i \times \sum_{i=1}^4 J_i \cdot \hat{d}_i} \quad (63)$$

with  $H_1$  to  $H_4$  the smoothness matrices for the spline functions  $s_1$  to  $s_4$  respectively. Substitution of eq. (62) for  $X\mathbf{c}$  and eq. (63) for  $H$  in eq. (29), gives the following estimator for the combined B-coefficients:

$$\begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \\ \mathbf{c}_4 \end{bmatrix} = C_1 X^T \mathbf{Y} \quad (64)$$

with  $C_1$  as in eq. (30). The spline model for the roll and yaw moment coefficients  $C_l$  and  $C_n$  are estimated using the same approach.

### C. Model Validation and Comparison

The polynomial and spline based aerodynamic models are compared to the original wind tunnel model and validated against the validation dataset. The results from the model validation are listed in table 4. Since the true model is known from the NASA wind tunnel data tables, a direct comparison can be made which is shown in figure 3. From this figure the nonlinearities of the moment coefficients can be observed. The spline model has a higher approximation power and is better able to model these nonlinearities at a global scale compared to the polynomial model. This can also be seen from RMS values of the model error. For example, the spline model for  $C_m$  has a relative error RMS of 2.72% while the polynomial model has a relative error RMS of 11.15%.

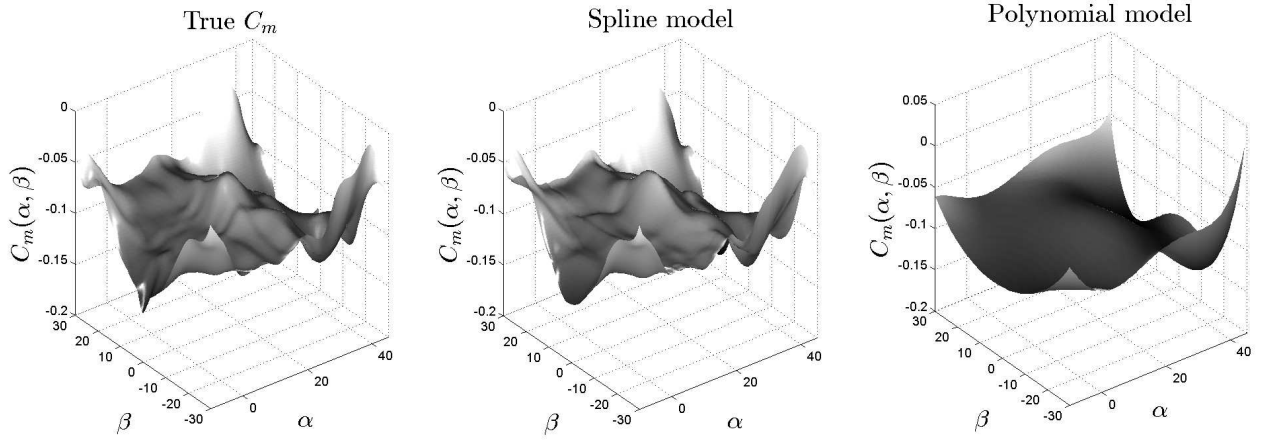
**Table 3. Aerodynamic Model Structures for estimating the F-16 wind tunnel database**

Function	Spline Model Structure	Polynomial Model Structure
$C_m(\alpha, \beta, \delta_e)$	$S_6^1(\mathcal{T}_{48})$	$a_0 + a_1\alpha + a_2\alpha\beta^2 + a_3\alpha^2\beta + a_4\alpha^2\beta^4 + a_5\alpha^3 +$ $a_6\alpha^5 + a_7\beta^2 + a_8\delta_e + a_9\alpha\delta_e + a_{10}\alpha\beta^2\delta_e +$ $a_{11}\alpha^2\beta^2\delta_e + a_{12}\alpha^3\delta_e + a_{13}\alpha^3\beta^2\delta_e + a_{14}\beta^2\delta_e +$ $a_{15}\delta_e^2 + a_{16}\alpha\delta_e^2 + a_{17}\alpha^2\delta_e^2 + a_{18}\alpha^3\beta^2\delta_e^2 + a_{19}\beta^2\delta_e^2 + a_{20}\delta_e^3$
$C_{m_{\delta_{lef}}}(\alpha, \beta)$	$S_5^1(\mathcal{T}_8)$	$b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^2\beta + b_4\alpha^3\beta + b_5\alpha^4 + b_6\alpha^4\beta$
$C_{m_q}(\alpha)$	$S_5^0(\mathcal{T}_4)$	$c_0 + c_1\alpha + c_2\alpha^2 + c_3\alpha^3 + c_4\alpha^4 + c_5\alpha^5$
$C_{m_{q\delta_{lef}}}(\alpha)$	$S_3^0(\mathcal{T}_4)$	$d_0 + d_1\alpha + d_2\alpha^2 + d_3\alpha^3$
$C_l(\alpha, \beta)$	$S_5^1(\mathcal{T}_{32})$	$e_0\beta + e_1\alpha\beta + e_2\alpha^2\beta + e_3\alpha^3\beta + e_4\alpha^4\beta + e_5\beta^3 +$ $e_6\alpha\beta^3 + e_7\alpha^2\beta^3 + e_8\alpha^3\beta^3 + e_9\alpha^4\beta^3$
$C_{l_{\delta_{lef}}}(\alpha, \beta)$	$S_5^1(\mathcal{T}_8)$	$f_0\alpha^2 + f_1\alpha^4 + f_2\alpha^6 + f_4\beta$
$C_{l_{\delta_a}}(\alpha, \beta)$	$S_4^1(\mathcal{T}_8)$	$g_0 + g_1\alpha + g_2\beta + g_3\alpha^2 + g_4\alpha\beta + g_5\alpha^2\beta + g_6\alpha^3$
$C_{l_{\delta_r}}(\alpha, \beta)$	$S_4^1(\mathcal{T}_8)$	$h_0 + h_1\alpha + h_2\beta + h_3\alpha\beta + h_4\alpha^2\beta + h_5\alpha^3\beta + h_6\beta^2$
$C_{l_r}(\alpha)$	$S_5^0(\mathcal{T}_4)$	$i_0 + i_1\alpha + i_2\alpha^2 + i_3\alpha^3$
$C_{l_{r\delta_{lef}}}(\alpha)$	$S_3^0(\mathcal{T}_4)$	$j_0 + j_1\alpha + j_2\alpha^2$
$C_{l_p}(\alpha)$	$S_3^0(\mathcal{T}_4)$	$k_0 + k_1\alpha + k_2\alpha^2 + k_3\alpha^3 + k_4\alpha^4 + k_5\alpha^5$
$C_{l_{p\delta_{lef}}}(\alpha)$	$S_3^0(\mathcal{T}_4)$	$l_0 + l_1\alpha + l_2\alpha^2$
$C_n(\alpha, \beta)$	$S_5^1(\mathcal{T}_{32})$	$m_0\beta + m_1\alpha\beta + m_2\alpha^2\beta + m_3\alpha^3\beta + m_4\beta^3 +$ $m_5\alpha\beta^3 + m_6\alpha^2\beta^3 + m_7\alpha^2 + m_8\alpha^3$
$C_{n_{\delta_{lef}}}(\alpha, \beta)$	$S_4^1(\mathcal{T}_8)$	$n_0\alpha^2\beta + n_1\alpha^4\beta + n_2\alpha^6\beta$
$C_{n_{\delta_a}}(\alpha, \beta)$	$S_3^1(\mathcal{T}_8)$	$o_0 + o_1\alpha + o_2\beta + o_3\alpha\beta + o_4\alpha^2\beta + o_5\alpha^3\beta$ $+ o_6\alpha^2 + o_7\alpha^3 + o_8\beta^3 + o_9\alpha\beta^3$
$C_{n_{\delta_r}}(\alpha, \beta)$	$S_5^1(\mathcal{T}_8)$	$p_0 + p_1\alpha + p_2\beta + p_3\alpha\beta + p_4\alpha^2\beta + p_5\alpha^2 + p_6\beta^2$
$C_{n_r}(\alpha)$	$S_4^0(\mathcal{T}_5)$	$q_0 + q_1\alpha + q_2\alpha^2 + q_3\alpha^3 + q_4\alpha^4 + q_5\alpha^5$
$C_{n_{r\delta_{lef}}}(\alpha)$	$S_3^0(\mathcal{T}_4)$	$r_0 + r_1\alpha + r_2\alpha^2 + r_3\alpha^3$
$C_{n_p}(\alpha)$	$S_3^0(\mathcal{T}_4)$	$s_0 + s_1\alpha + s_2\alpha^2 + s_3\alpha^3 + s_4\alpha^4 + s_5\alpha^5$
$C_{n_{p\delta_{lef}}}(\alpha)$	$S_1^0(\mathcal{T}_2)$	$t_0\alpha$

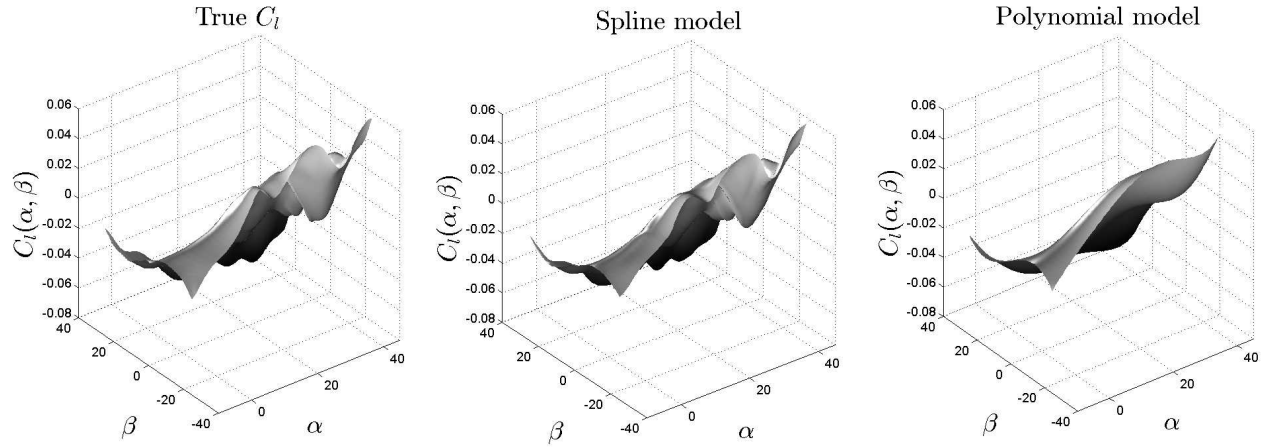
**Table 4. Model validation performance parameters**

Performance Parameter	Spline Model			Polynomial Model		
	error RMS	relative error RMS*	max error RMS	error RMS	relative error RMS	max error RMS
$C_l$ validation	0.0029	6.86%	0.0232	0.0085	19.95%	0.0744
$C_m$ validation	0.0042	2.72 %	0.0350	0.0172	11.15%	0.1186
$C_n$ validation	0.0043	7.83%	0.0236	0.0097	17.50%	0.0471

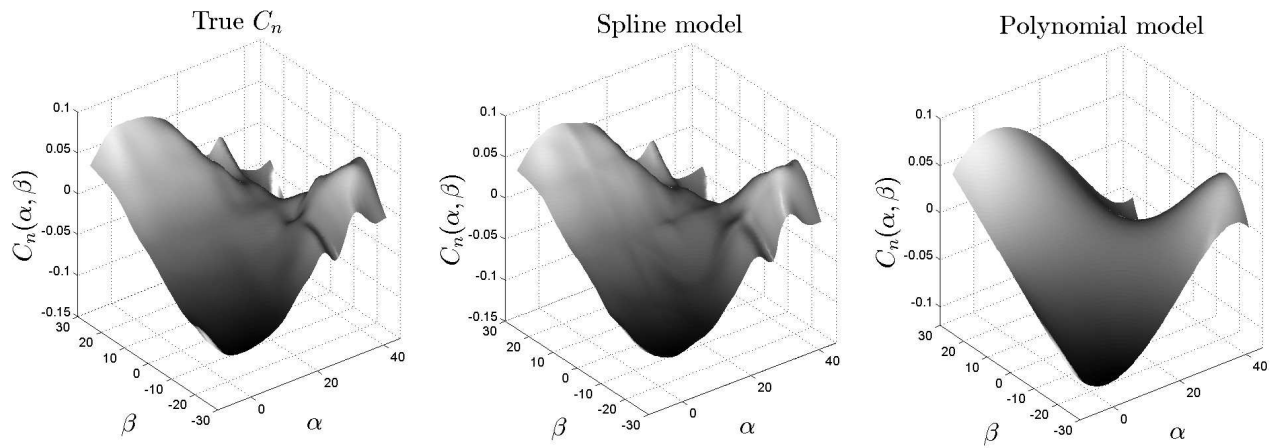
\* The relative error RMS is defined as:  $RMS_{rel}(\epsilon) = \frac{RMS(\epsilon)}{RMS(\mathbf{Y}_v)}$



(a)  $C_m$  surface plots ( $\tilde{q} = 0.04$ ,  $\delta_e = 5^\circ$ ,  $\delta_{lef} = 15^\circ$ )



(b)  $C_l$  surface plots ( $\tilde{p} = 0.0106$ ,  $\tilde{r} = 0.0106$ ,  $\delta_a = 5^\circ$ ,  $\delta_r = 5^\circ$ ,  $\delta_{lef} = 15^\circ$ )



(c)  $C_n$  surface plots ( $\tilde{p} = 0.0106$ ,  $\tilde{r} = 0.0106$ ,  $\delta_a = 5^\circ$ ,  $\delta_r = 5^\circ$ ,  $\delta_{lef} = 15^\circ$ )

**Figure 3. Model comparison**

## VII. Spline Model based Control Allocation

This section contains the main contribution of the paper. It discusses the process of control allocation for system models based on multivariate splines that are not affine in the inputs. The use of non-affine aerodynamic spline models requires the augmentation of the NDI structure with a separate control allocation module. This augmented structure was introduced in the Sec. IV. All NDI flight control laws can be written in terms of required forces or moments which can be seen as a virtual input  $\boldsymbol{\tau}$ :

$$\boldsymbol{\tau}_{req} = g^{-1}(\mathbf{x})(\boldsymbol{\nu} - \mathbf{f}(\mathbf{x}))$$

This required demand has to be translated into control surface deflections based on the onboard spline model. The model for  $\boldsymbol{\tau}$  is assumed to be a nonlinear function of the aircraft state and control input and is approximated with a spline function:

$$\tau_i = \Phi(\mathbf{x}, \mathbf{u}) \approx s(\mathbf{x}, \mathbf{u}) \quad (65)$$

For example, the virtual controls for the control setup in figure 2 are the moment coefficients for which the model has been estimated by spline functions in the previous section:

$$\tau_1 = C_l \approx s(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef}), \quad \tau_2 = C_m \approx s(\alpha, \beta, \tilde{q}, \delta_e, \delta_{lef}), \quad \tau_3 = C_n \approx s(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef})$$

The control allocation problem for spline based aerodynamic models can be stated as: Given a virtual command  $\boldsymbol{\tau}$ , determine  $\mathbf{u}$  satisfying the actuator limits, such that  $\boldsymbol{\tau} = \mathbf{s}(\mathbf{x}, \mathbf{u})$ . This section presents a control allocator that is formulated in terms of analytical expressions for the Jacobian and Hessian of the spline model. This allocator requires the analytical derivation of the gradient and Hessian of a B-form simplex polynomial and is presented in Sec. A. In Sections B - D the analytical expressions are used to formulate three control allocation strategies that can be specifically applied to spline models; two linear strategies and one nonlinear strategy. See also appendix A for a control allocation example. The advantages of having an analytical expression over a numerical approximation is that it is exact and computationally more efficient to calculate. For example, the central difference approximation of the second derivative  $\frac{\partial^2 f}{\partial x_i \partial x_j}$  requires four evaluations of function  $f$  compared to one evaluation of the second derivative when having an analytical expression [30, pp.884].

### A. Gradient and Hessian of the B-form Simplex Polynomial

In this section two theorems are provided for the gradient and the Hessian of a B-form simplex polynomial  $p_j^t(b(\mathbf{x}))$  with respect to the spline state  $\mathbf{x}$ . In the following sections an expression for the barycentric coordinate  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  as an affine function of the spline state  $\mathbf{x}$  is required.

The barycentric coordinates  $(b_1, \dots, b_n)$  given by eq. (9) can be expressed as an affine function of  $\mathbf{x}$  is derived as follows::

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \Lambda(\mathbf{x} - \mathbf{v}_0) = \Lambda\mathbf{x} - \Lambda\mathbf{v}_0 = \Lambda\mathbf{x} + \mathbf{k}_n \quad (66)$$

with  $\mathbf{k}_n = -\Lambda\mathbf{v}_0$ . And the component  $b_0$  as follows:

$$\begin{aligned} b_0 = 1 - \sum_{i=1}^n b_i &= 1 - \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \\ &= - \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} \Lambda\mathbf{x} + \left(1 - \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} \mathbf{k}_n\right) \\ &= \Lambda_0\mathbf{x} + k_0 \end{aligned} \quad (67)$$

Combining (66) and (67) results in:

$$\begin{bmatrix} b_0 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \Lambda_0 \\ \Lambda \end{bmatrix} \mathbf{x} + \begin{bmatrix} k_0 \\ \mathbf{k}_n \end{bmatrix} = A\mathbf{x} + \mathbf{k} \quad (68)$$

Eq. (68) is used to derive the first and second partial derivatives of a B-form basis polynomial  $B_\kappa^d(b(\mathbf{x}))$ . The first partial derivative is given by the following lemma.

**Lemma 1.** *Let the barycentric coordinate  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  of  $\mathbf{x}$  with respect to the  $n$ -simplex  $t$  be an affine function of  $\mathbf{x}$  given by:*

$$b(\mathbf{x}) = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix}_{t_j} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \mathbf{k} = A_{t_j} \mathbf{x} + \mathbf{k} \quad (69)$$

In that case the partial derivative of a B-form basis polynomial  $B_\kappa^d(b(\mathbf{x}))$  with respect to  $x_i$  is given by:

$$\frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial x_i} = \mathbf{a}_i^T \nabla_{\mathbf{b}} B_\kappa^d(b(\mathbf{x})) \quad (70)$$

with  $\nabla_{\mathbf{b}} B_\kappa^d(b(\mathbf{x}))$  the gradient of the B-form polynomial with respect to the barycentric coordinate  $\mathbf{b}$ :

$$\nabla_{\mathbf{b}} B_\kappa^d(b(\mathbf{x})) = \begin{bmatrix} \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_0} & \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_1} & \cdots & \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_n} \end{bmatrix}^T \quad (71)$$

*Proof.* This proof uses the multi-variable chain rule:

$$\frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial x_i} = \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_0} \frac{\partial b_0(\mathbf{x})}{\partial x_i} + \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_1} \frac{\partial b_1(\mathbf{x})}{\partial x_i} + \cdots + \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_n} \frac{\partial b_n(\mathbf{x})}{\partial x_i} \quad (72)$$

Eq. (72) can be written in vector form:

$$\frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial x_i} = \begin{bmatrix} \frac{\partial b_0(\mathbf{x})}{\partial x_i} & \frac{\partial b_1(\mathbf{x})}{\partial x_i} & \cdots & \frac{\partial b_n(\mathbf{x})}{\partial x_i} \end{bmatrix} \begin{bmatrix} \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_0} & \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_1} & \cdots & \frac{\partial B_\kappa^d(b(\mathbf{x}))}{\partial b_n} \end{bmatrix}^T \quad (73)$$

$$= \mathbf{a}_i^T \nabla_{\mathbf{b}} B_\kappa^d(b(\mathbf{x})) \quad (74)$$

with  $\mathbf{a}_i$  the  $i$ -th column of  $A_{t_j}$ . □

The second partial derivative of a B-form basis polynomials is given by the second lemma.

**Lemma 2.** *Let the barycentric coordinate  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  of  $\mathbf{x}$  with respect to the  $n$ -simplex  $t$  be an affine function of  $\mathbf{x}$  given by:*

$$b(\mathbf{x}) = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix}_{t_j} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \mathbf{k} = A_{t_j} \mathbf{x} + \mathbf{k} \quad (75)$$

In that case the second partial derivative of a B-form basis polynomial  $B_\kappa^d(b(\mathbf{x}))$  with respect to  $x_i, x_j$  is given by:

$$\frac{\partial^2 B_\kappa^d(b(\mathbf{x}))}{\partial x_i \partial x_j} = \mathbf{a}_i^T \nabla_{\mathbf{b}}^2 B_\kappa^d(b(\mathbf{x})) \mathbf{a}_j \quad (76)$$



with  $\nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x}))$  the Hessian of the B-form polynomial with respect to the barycentric coordinate  $\mathbf{b}$ :

$$\nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x})) = \begin{bmatrix} \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_0^2} & \dots & \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_0 \partial b_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_n \partial b_0} & \dots & \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_n^2} \end{bmatrix} \quad (77)$$

*Proof.* It is shown that  $\frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial x_i \partial x_j} = \mathbf{a}_i^T \nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x})) \mathbf{a}_j$ :

$$\begin{aligned} \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial x_i \partial x_j} &= \frac{\partial}{\partial x_j} \frac{\partial B_{\kappa}^d(b(\mathbf{x}))}{\partial x_i} = \frac{\partial}{\partial x_j} \mathbf{a}_i^T \nabla_{\mathbf{b}} B_{\kappa}^d(b(\mathbf{x})) \quad (\text{by Lemma 1}) \\ &= \mathbf{a}_i^T \begin{bmatrix} \frac{\partial}{\partial x_j} \frac{\partial B_{\kappa}^d(b(\mathbf{x}))}{\partial b_0} \\ \vdots \\ \frac{\partial}{\partial x_j} \frac{\partial B_{\kappa}^d(b(\mathbf{x}))}{\partial b_n} \end{bmatrix} = \mathbf{a}_i^T \begin{bmatrix} \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_0^2} \frac{\partial b_0}{\partial x_j} + \dots + \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_0 \partial b_n} \frac{\partial b_n}{\partial x_j} \\ \vdots \\ \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_n \partial b_0} \frac{\partial b_0}{\partial x_j} + \dots + \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_n^2} \frac{\partial b_n}{\partial x_j} \end{bmatrix} \quad (\text{Chain rule}) \\ &= \mathbf{a}_i^T \begin{bmatrix} \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_0^2} & \dots & \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_0 \partial b_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_n \partial b_0} & \dots & \frac{\partial^2 B_{\kappa}^d(b(\mathbf{x}))}{\partial b_n^2} \end{bmatrix} \begin{bmatrix} \frac{\partial b_0(\mathbf{x})}{\partial x_j} \\ \vdots \\ \frac{\partial b_n(\mathbf{x})}{\partial x_j} \end{bmatrix} = \mathbf{a}_i^T \nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x})) \mathbf{a}_j \end{aligned}$$

with  $\mathbf{a}_i$  and  $\mathbf{a}_j$  the i-th and j-th column of  $A_{t_j}$ .  $\square$

The lemmas for the partial derivatives are now used to derive the gradient and the Hessian of a B-form simplex polynomial  $p^{t_j}(b(\mathbf{x}))$ . The first theorem provides the gradient:

**Theorem 1** (Gradient of a simplex polynomial in terms of Cartesian coordinates). *Let the barycentric coordinate  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  of  $\mathbf{x}$  with respect to the  $n$ -simplex  $t$  be an affine function of  $\mathbf{x}$  given by:*

$$b(\mathbf{x}) = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix}_{t_j} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \mathbf{k} = A_{t_j} \mathbf{x} + \mathbf{k} \quad (78)$$

In that case the gradient of the B-form simplex polynomial  $p^{t_j}(b(\mathbf{x}))$  of degree  $d$  with respect to the spline state  $\mathbf{x}$  is given by:

$$\nabla_{\mathbf{x}} p^{t_j}(b(\mathbf{x})) = A_{t_j}^T \nabla_{\mathbf{b}} B_{t_j}^d(b(\mathbf{x})) \mathbf{c}^{t_j} \quad (79)$$

with  $\nabla_{\mathbf{b}} B_{t_j}^d(b(\mathbf{x}))$  the vector of B-form polynomial gradients given by:

$$\nabla_{\mathbf{b}} B_{t_j}^d(b(\mathbf{x})) =: \begin{bmatrix} \nabla_{\mathbf{b}} B_{d,0,\dots,0}^d(b(\mathbf{x})) & \nabla_{\mathbf{b}} B_{d-1,1,\dots,0}^d(b(\mathbf{x})) & \dots & \nabla_{\mathbf{b}} B_{0,\dots,0,d}^d(b(\mathbf{x})) \end{bmatrix} = [\nabla_{\mathbf{b}} B_{\kappa}^d(b(\mathbf{x}))]_{|\kappa|=d} \in \mathbb{R}^{n+1 \times \hat{d}} \quad (80)$$

and with  $\mathbf{c}^{t_j}$  the vector of B-coefficients given by eq. (18):

$$\mathbf{c}^{t_j} := [c_{\kappa}^{t_j}]_{|\kappa|=d} \in \mathbb{R}^{\hat{d} \times 1}$$

*Proof.* This proof starts by showing that  $\frac{\partial p^{t_j}(b(\mathbf{x}))}{\partial x_i} = \mathbf{a}_i^T \nabla_{\mathbf{b}} B_{t_j}^d(b(\mathbf{x})) \mathbf{c}^{t_j}$

$$\begin{aligned} \frac{\partial p^{t_j}(b(\mathbf{x}))}{\partial x_i} &= \frac{\partial}{\partial x_i} \sum_{|\kappa|=d} c_{\kappa}^{t_j} B_{\kappa}^d(b(\mathbf{x})) = \sum_{|\kappa|=d} c_{\kappa}^{t_j} \frac{\partial}{\partial x_i} B_{\kappa}^d(b(\mathbf{x})) = \sum_{|\kappa|=d} c_{\kappa}^{t_j} \mathbf{a}_i^T \nabla_{\mathbf{b}} B_{\kappa}^d(b(\mathbf{x})) \quad (\text{by Lemma 1}) \\ &= \mathbf{a}_i^T \sum_{|\kappa|=d} c_{\kappa}^{t_j} \nabla_{\mathbf{b}} B_{\kappa}^d(b(\mathbf{x})) \quad (81) \end{aligned}$$

Using the definitions in eq. (80) and eq. (18), eq (81) can be written in vector form:

$$\frac{\partial p^{t_j}(b(\mathbf{x}))}{\partial x_i} = \mathbf{a}_i^T \nabla_{\mathbf{b}} B_{t_j}^d(b(\mathbf{x})) \mathbf{c}^{t_j} \quad (82)$$

Combining the partial derivatives for all  $x_i$  gives:

$$\nabla_{\mathbf{x}} p^{t_j}(b(\mathbf{x})) = \begin{bmatrix} \frac{\partial p^{t_j}(b(\mathbf{x}))}{\partial x_1} \\ \frac{\partial p^{t_j}(b(\mathbf{x}))}{\partial x_2} \\ \vdots \\ \frac{\partial p^{t_j}(b(\mathbf{x}))}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix} \nabla_{\mathbf{b}} B_{t_j}^d(b(\mathbf{x})) \mathbf{c}^{t_j} = A_{t_j}^T \nabla_{\mathbf{b}} B_{t_j}^d(b(\mathbf{x})) \mathbf{c}^{t_j} \quad (83)$$

which proves the theorem.  $\square$

The following theorem provides the Hessian of a B-form simplex polynomial  $p^{t_j}(b(\mathbf{x}))$

**Theorem 2** (Hessian of a B-form simplex polynomial in terms of Cartesian coordinates). *Let the barycentric coordinate  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  of  $\mathbf{x}$  with respect to the  $n$ -simplex  $t$  be an affine function of  $\mathbf{x}$  given by:*

$$b(\mathbf{x}) = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix}_{t_j} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \mathbf{k} = A_{t_j} \mathbf{x} + \mathbf{k} \quad (84)$$

In that case the Hessian of the B-form simplex polynomial  $p^{t_j}(b(\mathbf{x}))$  of degree  $d$  with respect to the spline state  $\mathbf{x}$  is given by:

$$\nabla_{\mathbf{x}}^2 p^{t_j}(b(\mathbf{x})) = A_{t_j}^T \left( \sum_{|\kappa|=d} c_{\kappa}^{t_j} \nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x})) \right) A_{t_j} = A_{t_j}^T \Gamma_{t_j} A_{t_j} \quad (85)$$

*Proof.* This proof start by showing that:  $\frac{\partial^2 p^{t_j}(b(\mathbf{x}))}{\partial x_i \partial x_j} = \mathbf{a}_i^T \left( \sum_{|\kappa|=d} c_{\kappa}^{t_j} \nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x})) \right) \mathbf{a}_j$ :

$$\begin{aligned} \frac{\partial^2 p^{t_j}(b(\mathbf{x}))}{\partial x_i \partial x_j} &= \frac{\partial^2}{\partial x_i \partial x_j} \sum_{|\kappa|=d} c_{\kappa}^{t_j} B_{\kappa}^d(b(\mathbf{x})) = \sum_{|\kappa|=d} c_{\kappa}^{t_j} \frac{\partial^2}{\partial x_i \partial x_j} B_{\kappa}^d(b(\mathbf{x})) \\ &= \sum_{|\kappa|=d} c_{\kappa}^{t_j} \mathbf{a}_i^T \nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x})) \mathbf{a}_j \quad (\text{by Lemma 2}) \\ &= \mathbf{a}_i^T \left( \sum_{|\kappa|=d} c_{\kappa}^{t_j} \nabla_{\mathbf{b}}^2 B_{\kappa}^d(b(\mathbf{x})) \right) \mathbf{a}_j = \mathbf{a}_i^T \Gamma_{t_j} \mathbf{a}_j \end{aligned}$$

Combining the second partial derivatives for all  $x_i, x_j$  gives:

$$\nabla_{\mathbf{x}}^2 p^{t_j}(b(\mathbf{x})) = \begin{bmatrix} \frac{\partial^2 p^{t_j}(b(\mathbf{x}))}{\partial x_1^2} & \cdots & \frac{\partial^2 p^{t_j}(b(\mathbf{x}))}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 p^{t_j}(b(\mathbf{x}))}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 p^{t_j}(b(\mathbf{x}))}{\partial x_n^2} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T \Gamma_{t_j} \mathbf{a}_1 & \cdots & \mathbf{a}_1^T \Gamma_{t_j} \mathbf{a}_n \\ \vdots & \ddots & \vdots \\ \mathbf{a}_n^T \Gamma_{t_j} \mathbf{a}_1 & \cdots & \mathbf{a}_n^T \Gamma_{t_j} \mathbf{a}_n \end{bmatrix} \quad (86)$$

$$= \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix} \Gamma_{t_j} \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{bmatrix} = A_{t_j}^T \Gamma_{t_j} A_{t_j} \quad (87)$$

Which proves the theorem.  $\square$

## B. Strategy 1: Linear Control Allocation

With this strategy the control allocation problem is solved for a linear approximation of the onboard spline model. Consider the spline structure given by eq. (3):

$$s(\mathbf{x}) = \delta_1 p^{t_1}(b(\mathbf{x})) + \delta_2 p^{t_2}(b(\mathbf{x})) + \cdots + \delta_j p^{t_j}(b(\mathbf{x})), \quad 1 \leq j \leq J$$

With  $\delta_j = 1$  if  $\mathbf{x} \in t_j$  and  $\delta_j = 0$  if  $\mathbf{x} \notin t_j$ . Suppose that the current state  $\mathbf{x}_0$  is located within simplex  $t_j$ . Then by linearizing the local polynomial  $p^{t_j}(\mathbf{x})$  around the current state, the linearized model becomes the global representation for the original spline model. At the current state  $\mathbf{x}_0$ , each spline function can be represented by a single simplex polynomial:

$$\tau_i = p^{t_j}(\mathbf{x}), \quad \mathbf{x}_0 \in t_j \quad (88)$$

Consider the affine formulation of the barycentric coordinates  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  given by eq. (68):

$$b(\mathbf{x}) = A_{t_j} \mathbf{x} + \mathbf{k} \quad (89)$$

Let the spline state  $\mathbf{x}$  consist of  $n$  aircraft states and  $m$  control inputs:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n & u_1 & u_2 & \dots & u_m \end{bmatrix}^T = \begin{bmatrix} \mathbf{x}_a^T & \mathbf{u}^T \end{bmatrix}^T \quad (90)$$

Since the Cartesian to barycentric coordinate system transformation is a linear, one-to-one transformation, the barycentric coordinates  $b(\mathbf{x}) = (b_0, b_1, \dots, b_n)$  can be parameterized as an affine function of the control input  $\mathbf{u}$  for a fixed aircraft state  $\mathbf{x}_a$ :

$$\begin{aligned} b(\mathbf{u}) &= \begin{bmatrix} A_{\mathbf{x}_a, t_j} & A_{\mathbf{u}, t_j} \end{bmatrix} \begin{bmatrix} \mathbf{x}_a \\ \mathbf{u} \end{bmatrix} + \mathbf{k} \\ &= A_{\mathbf{u}, t_j} \mathbf{u} + A_{\mathbf{x}_a, t_j} \mathbf{x}_a + \mathbf{k} \\ &= A_{\mathbf{u}, t_j} \mathbf{u} + \tilde{\mathbf{k}} \end{aligned} \quad (91)$$

With  $A_{\mathbf{x}_a, t_j} \in \mathbb{R}^{n \times l}$  and  $A_{\mathbf{u}, t_j} \in \mathbb{R}^{n \times m}$  the partitions of  $A_{t_j}$ . Using this parameterization the simplex polynomial can be expressed as a function only dependent on the control input  $\mathbf{u}$ :  $p^{t_j}(\mathbf{u})$ . By theorem 1 the gradient of the simplex polynomial with respect to the control input is given by:

$$\nabla_{\mathbf{u}} p^{t_j}(\mathbf{u}) = A_{\mathbf{u}, t_j}^T \nabla_{\mathbf{b}} B_{t_j}^d(\mathbf{u}) \mathbf{c}^{t_j} \quad (92)$$

The linearized model around the current control input  $\mathbf{u}_0$  for the complete effector model becomes:

$$\begin{bmatrix} \tau_1 \\ \vdots \\ \tau_i \\ \vdots \\ \tau_m \end{bmatrix} = \begin{bmatrix} p_1(\mathbf{u}_0) \\ \vdots \\ p_i(\mathbf{u}_0) \end{bmatrix} + \begin{bmatrix} \nabla_{\mathbf{u}}^T p_1(\mathbf{u}_0) \\ \vdots \\ \nabla_{\mathbf{u}}^T p_i(\mathbf{u}_0) \end{bmatrix} \begin{bmatrix} \Delta u_1 \\ \vdots \\ \Delta u_m \end{bmatrix} \quad (93)$$

which can be written in vector form:

$$\boldsymbol{\tau} = \mathbf{p}(\mathbf{x}_0, \mathbf{u}_0) + G \Delta \mathbf{u} \quad (94)$$

The linearized model is directly related to the aircraft control input and any linear control allocation method can now be applied such as the redistributed pseudo-inverse solution [26, 27] or constrained linear programming techniques [31]. The approach elaborated here is based on the pseudo-inverse solution. Using the linearized model in eq. (94) the error between the model and required output can be written as:

$$\begin{aligned} \mathbf{e} &= \mathbf{p}(\mathbf{x}_0, \mathbf{u}_0) + G \Delta \mathbf{u} - \boldsymbol{\tau}_{req} \\ &= G \Delta \mathbf{u} - \tilde{\boldsymbol{\tau}}_{req} \end{aligned} \quad (95)$$

with

$$\tilde{\boldsymbol{\tau}}_{req} = \boldsymbol{\tau}_{req} - \mathbf{p}(\mathbf{x}_0, \mathbf{u}_0) \quad (96)$$

The control minimization problem (eq. (43)) can now be formulated as:

$$\min_{\Delta \mathbf{u} \leq \Delta \mathbf{u} \leq \Delta \bar{\mathbf{u}}} \mathcal{J} = \|\Delta \mathbf{u}\| \quad \text{s.t.} \quad G\Delta \mathbf{u} = \tilde{\boldsymbol{\tau}}_{req} \quad (97)$$

By dropping the actuator constraints, the incremental control input can be calculated using the pseudo-inverse solution in eq. (46):

$$\Delta \mathbf{u} = G^+ \tilde{\boldsymbol{\tau}}_{req} \quad (98)$$

Actuator constraints can then be taken into account by applying the redistribution scheme from [26,27]. By linearizing the spline model and computing the optimal solution at each time step the new control input vector becomes:

$$\mathbf{u}(t+1) = \mathbf{u}(t) + \Delta \mathbf{u} \quad (99)$$

### C. Strategy 2: Successive Linear Control Allocation

The approach discussed in the previous section may produce inaccurate solutions in the case of highly nonlinear effector models. In this case the linearized model might be inaccurate resulting in large allocation errors. In this section a successive linear approach is presented to account for the nonlinearities in which the control allocation problem is solved for a sequence of linear approximations of the onboard spline model. In the previous section the spline model was linearized around current input  $\mathbf{u}(t_0)$  for which the pseudo-inverse solution is applied. Solutions with better accuracy can be obtained by successively calculating the pseudo-inverse solution for several initial conditions in the feasible set for  $\mathbf{u}$  and selecting the one that yields the lowest value for the control allocation error. This approach consists of four steps: First define a feasible subset  $\Omega$  for  $\mathbf{u}$ :

$$\Omega = \{\mathbf{u} \in \mathbb{R}^m | \underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}\} \subset \mathbb{R}^m \quad (100)$$

Where  $\underline{\mathbf{u}}$  and  $\bar{\mathbf{u}}$  are lower and upper bounds. Second, define a tuple consisting of  $k$  initial conditions in the feasible subset:

$$\mathcal{V}_1 := (\mathbf{u}_{0_1}, \mathbf{u}_{0_2}, \dots, \mathbf{u}_{0_k}) \in \Omega \quad (101)$$

Third, linearize the spline model around each initial condition  $(\mathbf{x}_0, \mathbf{u}_{0_k})$  to obtain the formulation in eq. (94) and calculate the incremental control input  $\Delta \mathbf{u}$  for all trials  $k$  through eqs. (98) and (99) to obtain a set of optimal solutions:

$$\mathcal{V}_2 = (\mathbf{u}_{0_1} + \Delta \mathbf{u}_1, \mathbf{u}_{0_2} + \Delta \mathbf{u}_2, \dots, \mathbf{u}_{0_k} + \Delta \mathbf{u}_k) = (\mathbf{u}_1^*, \mathbf{u}_2^*, \dots, \mathbf{u}_k^*) \quad (102)$$

Fourth, calculate the control allocation error based on the onboard spline model:

$$\mathcal{V}_3 = (\|\Phi(\mathbf{x}_0, \mathbf{u}_1^*) - \boldsymbol{\tau}_{req}\|, \|\Phi(\mathbf{x}_0, \mathbf{u}_2^*) - \boldsymbol{\tau}_{req}\|, \dots, \|\Phi(\mathbf{x}_0, \mathbf{u}_k^*) - \boldsymbol{\tau}_{req}\|) \quad (103)$$

and select the input that yields the lowest value for the control allocation error. The solution is iterated by repeating these steps at each sample instant. This approach requires a definition of the feasible set  $\Omega$  and the number of trials  $k$ . The upper and lower bounds in eq. (100) should not only be determined by the actuator position constraints, but should rather be given by small deviations around the current control signal:

$$\underline{\mathbf{u}}(t+1) = \max \{\mathbf{u}(t) - \boldsymbol{\varepsilon}, \mathbf{u}_{min}\} \quad (104)$$

$$\bar{\mathbf{u}}(t+1) = \min \{\mathbf{u}(t) + \boldsymbol{\varepsilon}, \mathbf{u}_{max}\} \quad (105)$$

The deviation  $\boldsymbol{\varepsilon}$  should be determined based on the knowledge of the control actuators. For example, for model given by eq. (2) a good choice would be  $\boldsymbol{\varepsilon} = \boldsymbol{\tau} \dot{\mathbf{u}}_{lim}$  such that maximum deflection can be achieved within the subset  $\Omega$ .

### D. Strategy 3: Nonlinear Control allocation

In this section a nonlinear solver for the control allocation problem is presented that minimizes the sum of square errors between the onboard aerodynamic spline model and the required demand:

$$\min_{\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}} \|\mathbf{s}(\mathbf{u}) - \boldsymbol{\tau}_{req}\|_2^2 = \sum_{i=1}^N (s_i(\mathbf{u}) - \tau_{req_i})^2 = \sum_{i=1}^N e_i(\mathbf{u})^2 \quad (106)$$

This is a constrained nonlinear optimization problem which usually requires a large number of iterations and function evaluations to solve. A common approach to avoid complex programming routines is to drop the actuator constraints and to linearize the model at each sample instant as was shown in the previous sections. The main disadvantage of this approach is that it results in large allocation errors in case of significant nonlinear models. In that case nonlinear solvers can provide more flexibility in handling nonlinearities. The solver suggested here emphasizes a combination of both: an efficient nonlinear solver that can be implemented analytically by matrix computations and which requires a small number of iterations to converge to a solution. With the analytical expressions for the gradient and Hessian derived in the previous section any second order optimization method, such as a sequential quadratic programming approach [32, 33], can be applied to solve the control allocation problem. The solver presented here is based on the Levenberg-Marquardt algorithm [34]. Consider the spline model representation given by eq. (3):

$$s(\mathbf{x}) = \delta_1 p^{t_1}(b(\mathbf{x})) + \delta_2 p^{t_2}(b(\mathbf{x})) + \dots + \delta_J p^{t_J}(b(\mathbf{x})) = \sum_{j=1}^J \delta_j p^{t_j}(\mathbf{x}) \quad (107)$$

With  $b(\mathbf{x})$  the barycentric coordinate of  $\mathbf{x}$  with respect to the  $n$ -simplex  $t_j$ . Let  $\mathbf{x}$  consist of  $l$  aircraft states and  $m$  control inputs:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n & u_1 & u_2 & \dots & u_m \end{bmatrix}^T = \begin{bmatrix} \mathbf{x}_a^T & \mathbf{u}^T \end{bmatrix}^T$$

Consider the parameterized barycentric coordinates as a function of  $\mathbf{u}$  given by eq. (91):

$$b(\mathbf{u}) = A_{\mathbf{u}, t_j} \mathbf{u} + \tilde{\mathbf{k}}$$

With  $A_{\mathbf{u}, t_j} \in \mathbb{R}^{n \times m}$ . Using this parameterization the spline function can be expressed as a function only dependent on the control input  $\mathbf{u}$ :

$$s_i(\mathbf{u}) = \sum_{j=1}^J \delta_j p^{t_j}(\mathbf{u}) \quad (108)$$

By theorem 1 the gradient of the spline function with respect to the control input is given by:

$$\nabla_{\mathbf{u}} s_i(\mathbf{u}) = \sum_{j=1}^J \delta_j A_{\mathbf{u}, t_j}^T \nabla_{\mathbf{b}} B_{t_j}^d(\mathbf{u}) \mathbf{c}^{t_j} \quad (109)$$

and by theorem 2 the Hessian is given by:

$$\nabla_{\mathbf{u}}^2 s_i(\mathbf{u}) = \sum_{j=1}^J \delta_j A_{\mathbf{u}, t_j}^T \Gamma_{t_j} A_{\mathbf{u}, t_j} \quad (110)$$

with  $\Gamma_{t_j}$  as given in (85). Let the Jacobian of the complete spline model  $\mathbf{s}(\mathbf{u})$  be defined as:

$$\nabla \mathbf{s}(\mathbf{u}) = \begin{bmatrix} \frac{\partial s_1(\mathbf{u})}{\partial u_1} & \dots & \frac{\partial s_1(\mathbf{u})}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_N(\mathbf{u})}{\partial u_1} & \dots & \frac{\partial s_N(\mathbf{u})}{\partial u_m} \end{bmatrix} = \begin{bmatrix} \nabla_{\mathbf{u}}^T s_1(\mathbf{u}) \\ \vdots \\ \nabla_{\mathbf{u}}^T s_N(\mathbf{u}) \end{bmatrix} \quad (111)$$

Then the gradient and Hessian of the objective function  $\mathcal{J}$  are given by:

$$\nabla \mathcal{J}(\mathbf{u}) = 2 \nabla_{\mathbf{u}}^T \mathbf{s}(\mathbf{u}) (\mathbf{s}(\mathbf{u}) - \boldsymbol{\tau}_{req}) \quad (112)$$

$$\nabla^2 \mathcal{J}(\mathbf{u}) = 2 \nabla^T \mathbf{s}(\mathbf{u}) \nabla \mathbf{s}(\mathbf{u}) + 2 \sum_{i=1}^N \nabla_{\mathbf{u}}^2 s_i(\mathbf{u}) (s_i(\mathbf{u}) - \tau_{req_i}) \quad (113)$$

Consider the second order approximation of the least squares objective in eq. (106) at  $\mathbf{u}(t_0) = \mathbf{u}_0$ :

$$\mathcal{J}(\mathbf{u}) \approx \mathcal{J}(\mathbf{u}_0) + \nabla_{\mathbf{u}}^T \mathcal{J}(\mathbf{u}_0) (\mathbf{u} - \mathbf{u}_0) + \frac{1}{2} (\mathbf{u} - \mathbf{u}_0)^T \nabla_{\mathbf{u}}^2 \mathcal{J}(\mathbf{u}) (\mathbf{u} - \mathbf{u}_0) = \bar{\mathcal{J}}(\mathbf{u}) \quad (114)$$

A good estimate for the solution of the unconstrained optimization problem is obtained by setting  $\frac{\partial \bar{\mathcal{J}}}{\partial \mathbf{u}}$  and solving for  $\mathbf{u}$ . This results in:

$$\mathbf{u}^* = \mathbf{u}_0 - (\nabla^2 \mathcal{J}(\mathbf{u}_0))^{-1} \nabla \mathcal{J}(\mathbf{u}) \quad (115)$$

To improve the result, the procedure can be repeated to obtain Newton's algorithm:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - (\nabla^2 \mathcal{J}(\mathbf{u}_k))^{-1} \nabla \mathcal{J}(\mathbf{u}_k) = \mathbf{u}_k + \mathbf{d}_k^{(n)} \quad (116)$$

With  $\mathbf{d}_k^{(n)}$  the Newton search direction. A property of the least squares objective function in eq. (106) is that if the error is small, i.e.  $\mathbf{u}_k$  is close to  $\mathbf{u}^*$ , the Hessian of the objective function can be approximated by:

$$\nabla^2 \mathcal{J}(\mathbf{u}) \approx 2 \nabla_{\mathbf{u}}^T \mathbf{s}(\mathbf{u}) \nabla \mathbf{s}(\mathbf{u}) = \bar{\nabla}_{\mathbf{u}}^2 \mathcal{J}(\mathbf{u}) \quad (117)$$

Substituting eq. (117) in eq.(116) results in the Gauss-Newton algorithm:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - (\bar{\nabla}^2 \mathcal{J}(\mathbf{u}_k))^{-1} \nabla \mathcal{J}(\mathbf{u}_k) = \mathbf{u}_k + \mathbf{d}_k^{(gn)} \quad (118)$$

Although an analytical expression for the Hessian of the spline function  $\nabla_{\mathbf{u}}^2 s_i(\mathbf{u})$  is available, using the approximation in eq. (117) avoids its evaluation making the solver more efficient. Furthermore, the Gauss-Newton Hessian matrix  $\bar{\nabla}_{\mathbf{u}}^2 \mathcal{J}(\mathbf{u})$  is always positive definite and therefore guarantees that the search direction  $\mathbf{d}_k^{(gn)}$  is a decent direction. The advantage of Gauss-Newton's algorithm is that it shows good local convergence, i.e. when the initial solution  $\mathbf{u}_0$  is chosen close to the optimal solution  $\mathbf{u}^*$ . This is often the case for the control allocation problem. For example, suppose that the global optimal solution  $\mathbf{u}^*(t_0)$  at time  $t = t_0$  is found. Then when using a small step size  $\Delta t$  it is likely that the optimal solution  $\mathbf{u}^*(t_0 + \Delta t)$  at time  $t_0 + \Delta t$ , is located in the neighborhood<sup>a</sup>  $\mathcal{N}$  of  $\mathbf{u}(t_0)$ . When this assumption is valid, the optimal solution at  $t_0 + \Delta t$  can be found quickly using the Gauss-Newton algorithm with  $\mathbf{u}^*(t_0)$  as initial feasible solution. However, the Gauss-Newton algorithm shows poor convergence when the initial solution:  $\mathbf{u}_0$  is far from  $\mathbf{u}^*$  and might diverge. Furthermore, the algorithm may not be defined when the Hessian is singular. The Levenberg-Marquardt algorithm [34] overcomes this problem and increases the robustness by adaptively varying between the Gauss-Newton search direction and the steepest descent search direction:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - (\bar{\nabla}^2 \mathcal{J}(\mathbf{u}_k) + \eta_k I)^{-1} \nabla \mathcal{J}(\mathbf{u}_k) = \mathbf{u}_k + \mathbf{d}_k^{(lm)} \quad (119)$$

Where  $\eta_k$  controls both the magnitude and direction of  $\mathbf{d}_k$ . When  $\eta_k$  is zero, the search direction  $\mathbf{d}_k^{(lm)}$  is identical to the Gauss-Newton search direction  $\mathbf{d}_k^{(gn)}$ . If on the other hand  $\eta_k$  goes to infinity, the search direction tends towards the steepest descent direction, with magnitude tending to zero:  $\eta_k \rightarrow \infty$ ,  $\mathbf{d}_k^{(lm)} \rightarrow -\nabla \mathcal{J}(\mathbf{u}_k)/\eta_k$ . The steepest decent direction shows fast initial progress when the initial solution is far from the optimum. So in case of divergence,  $\eta_k$  must be increased by a factor  $v$  such that:

$$\mathcal{J}(\mathbf{u}_{k+1}) < \mathcal{J}(\mathbf{u}_k) \quad (120)$$

In [34] it is proved that a sufficient large  $\eta_k$  exist such that eq. (120) holds. The Levenberg-Marquardt algorithm does not take the actuator limits into account. A frequently used approach to handle actuator limits is to add a barrier function to the objective function [35]. Barrier functions keep the iterates away from the boundaries. However, in case of actuator saturation, the optimal solution to the control allocation problem is often on the boundaries and thus, the use of barrier functions can result in less accurate solutions. For this reason, the limits are incorporated by clipping the components of the control vector that exceed their limits at their allowable values.

The nonlinear control allocation algorithm can be summarized as: Let  $\mathbf{u}_k = \mathbf{u}(t_0)$ ,  $\eta_k = \eta_0$ ,  $v > 1$ :

1. Try an update:  $\mathbf{u}_{try} = \mathbf{u}_k - (\bar{\nabla}^2 \mathcal{J}(\mathbf{u}_k) + \eta_k I)^{-1} \nabla \mathcal{J}(\mathbf{u}_k)$
2. Saturate controls:  $\mathbf{u}_{try} = \min \{ \max \{ \mathbf{u}_{try}, \underline{\mathbf{u}} \}, \bar{\mathbf{u}} \}$
3. Evaluate the objective:  $\mathcal{J}(\mathbf{u}_{try}) = \|\mathbf{s}(\mathbf{u}_{try}) - \boldsymbol{\tau}_c\|_2^2$
4. Update solution:

---

<sup>a</sup>The neighborhood of point  $\mathbf{u}(t_0)$  could be defined as an open ball with center  $\mathbf{u}(t_0)$  and a small radius  $\epsilon$

- if  $\mathcal{J}(\mathbf{u}_{try}) \leq \mathcal{J}(\mathbf{u}_k)$  accept solution:  $\mathbf{u}_{k+1} = \mathbf{u}_{try}$ ,  $\eta_{k+1} = \eta_k/\nu$
- if  $\mathcal{J}(\mathbf{u}_{try}) > \mathcal{J}(\mathbf{u}_k)$  retract the update:  $\mathbf{u}_{k+1} = \mathbf{u}_k$ ,  $\eta_{k+1} = \eta_k\nu$

Choosing a small initial value for  $\eta_0$ , e.g. 0.005, leads to fast convergence when the initial solution  $\mathbf{u}_0$  is close to the optimal solution  $\mathbf{u}^*$ . This is a reasonable assumption as described above. The choice of  $\nu$  is arbitrary, but a value of 10 has been found to be a good a choice.

## VIII. Evaluation of the Spline Based NDI Controller

In this section the spline based NDI controller is evaluated. In Sec. A the control allocation strategies are evaluated and in Sec. B a performance assessment is made by comparing the SNDI controller with a polynomial based NDI controller.

### A. Evaluation of the Control Allocation Strategies

In this section the three control allocation strategies are applied to the F-16 simulation model. The allocation of the control input for a required demand  $C_{l_{req}}$ ,  $C_{m_{req}}$  and  $C_{n_{req}}$  is based on the spline models identified in Sec. VI. Consider the spline model structures for the moment coefficients given by eqs. (56) -(58):

$$C_l(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef}) = s_{11}(\alpha, \beta) + s_{12}(\alpha, \beta)\delta_{lef} + s_{13}(\alpha, \beta)\delta_a + s_{14}(\alpha, \beta)\delta_r + s_{15}(\alpha)\tilde{r} + s_{16}(\alpha)\delta_{lef}\tilde{r} + s_{17}(\alpha)\tilde{p} + s_{18}(\alpha)\delta_{lef}\tilde{p} \quad (121)$$

$$C_m(\alpha, \beta, \tilde{q}, \delta_e, \delta_{lef}) = s_{21}(\alpha, \beta, \delta_e) + s_{22}(\alpha, \beta)\delta_{lef} + s_{23}\tilde{q} + s_{24}(\alpha)\delta_{lef}\tilde{q} \quad (122)$$

$$C_n(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef}) = s_{31}(\alpha, \beta) + s_{32}(\alpha, \beta)\delta_{lef} + s_{33}(\alpha, \beta)\delta_a + s_{34}(\alpha, \beta)\delta_r + s_{35}(\alpha)\tilde{r} + s_{36}(\alpha)\delta_{lef}\tilde{r} + s_{37}(\alpha)\tilde{p} + s_{38}(\alpha)\delta_{lef}\tilde{p} \quad (123)$$

The leading edge flap  $\delta_{lef}$  is scheduled as a function of the angle of attack to optimize performance. At each sample instant the required moment coefficients  $C_{l_{req}}$ ,  $C_{m_{req}}$  and  $C_{n_{req}}$  have to be translated into an actuator settings  $\delta_a$ ,  $\delta_e$  and  $\delta_r$  such that:

$$\tilde{C}_{l_{req}} = s_{13}(\alpha, \beta)\delta_a + s_{14}(\alpha, \beta)\delta_r \quad (124)$$

$$\tilde{C}_{m_{req}} = s_{21}(\alpha, \beta, \delta_e) \quad (125)$$

$$\tilde{C}_{n_{req}} = s_{33}(\alpha, \beta)\delta_a + s_{34}(\alpha, \beta)\delta_r \quad (126)$$

with

$$\tilde{C}_{l_{req}} = C_{l_{req}} - [s_{11}(\alpha, \beta) + s_{12}(\alpha, \beta)\delta_{lef} + s_{15}(\alpha)\tilde{r} + s_{16}(\alpha)\delta_{lef}\tilde{r} + s_{17}(\alpha)\tilde{p} + s_{18}(\alpha)\delta_{lef}\tilde{p}] \quad (127)$$

$$\tilde{C}_{m_{req}} = C_{m_{req}} - [s_{22}(\alpha, \beta)\delta_{lef} + s_{23}(\alpha)\tilde{q} + s_{24}(\alpha)\delta_{lef}\tilde{q}] \quad (128)$$

$$\tilde{C}_{n_{req}} = C_{n_{req}} - [s_{31}(\alpha, \beta) + s_{32}(\alpha, \beta)\delta_{lef} + s_{35}(\alpha)\tilde{r} + s_{36}(\alpha)\delta_{lef}\tilde{r} + s_{37}(\alpha)\tilde{p} + s_{38}(\alpha)\delta_{lef}\tilde{p}] \quad (129)$$

The F-16 lateral dynamics are affine in the control input, and so are the spline approximations for  $C_l$  and  $C_n$ . Therefore, the control allocation strategies are evaluated by performing a number of high amplitude angle of attack maneuvers using the control structure in figure 2. Feedback on the roll and yaw channel is only used for stabilization. The angle of attack response for the three allocation strategies is shown in figures 4, 5 and 6. The plots contain three simulations starting at different trim conditions for the angle of attack:  $\alpha = 5^\circ$ ,  $\alpha = 15^\circ$  and  $\alpha = 25^\circ$ . In addition, figure 6 contains a subplot with the number of iterations performed by the algorithm at each time step. To reduce the computational load the maximum number of iterations is set to 10. For the successive linear method five initial conditions uniformly distributed in the feasible set for  $\delta_e$  are used. The approach described in Sec.VII.B is used for defining the the feasible set. The plot shows the angle of attack response for the three strategies starting at  $\alpha = 15^\circ$ . The performance is evaluated by comparing the allocation error, which is the error between the required moment delivered by the NDI controller and the actual moment delivered the control allocator:

$$\Delta C_m(t) = C_{m_{req}}(t) - C_m(t) \quad (130)$$

The RMS values and maximum error for the three strategies are listed in table 5 and the MATLAB executions times are listed in table 6

At moderate angles of attack, the performance of the linear strategy is comparable to the successive linear and nonlinear strategies. At higher angles of attack, the nonlinearities cause large allocation errors which in turn results in a poor tracking performance and possibly unstable system, see the lower left plot of figure 4. Maneuverability at higher angles of attack can be improved by using the successive linear or the nonlinear control allocation method.

The nonlinear allocation strategy is the benchmark algorithm for coping with nonlinear aerodynamics as it results in significant lower allocation errors in the high angle of attack regions. However, the nonlinear optimization techniques may be too costly computationally for online applications. Although the average computational load for the nonlinear strategy is lower than for the successive strategy, during maneuvering the number of required iterations for the algorithm to converge increases as can be seen from figure 6.b. In turn, this results in high maximum computation loads as can be seen from table 6, while the computational load of the successive strategy is fixed by design, i.e. the selection of the trials  $k$ . The successive linear strategy is a performance optimization with respect to complexity and computational efficiency; full envelope tracking can be achieved while nonlinear optimization is avoided. However, it requires careful selection of the initial conditions and number of trials.

**Table 5. Performance assessment of the control allocation techniques for SNDI**

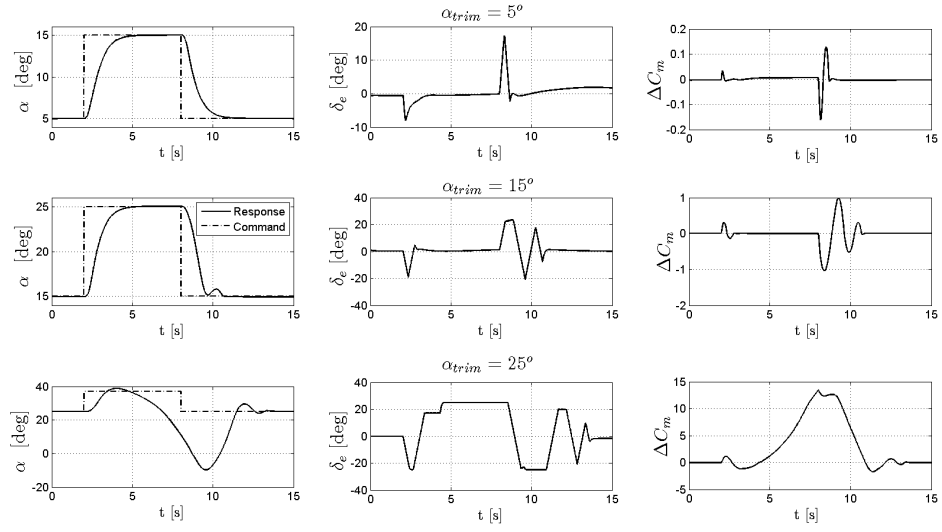
Condition	$\alpha_0 = 5^\circ$		$\alpha_0 = 15^\circ$		$\alpha_0 = 25^\circ$	
Parameter	RMS $\Delta C_m^*$	Max $ \Delta C_m $	RMS $\Delta C_m$	Max $ \Delta C_m $	RMS $\Delta C_m$	Max $ \Delta C_m $
Linear	0.0222	0.1667	0.2453	1.0451	5.4956	13.4828
Suc. linear	0.0225	0.1669	0.2666	1.0452	1.1355	3.9967
Nonlinear	0.0223	0.1671	0.2125	1.0466	1.0957	3.9160

\* The inversion error is defined as:  $\Delta C_m(t) = C_{m_{req}}(t) - C_m(t)$

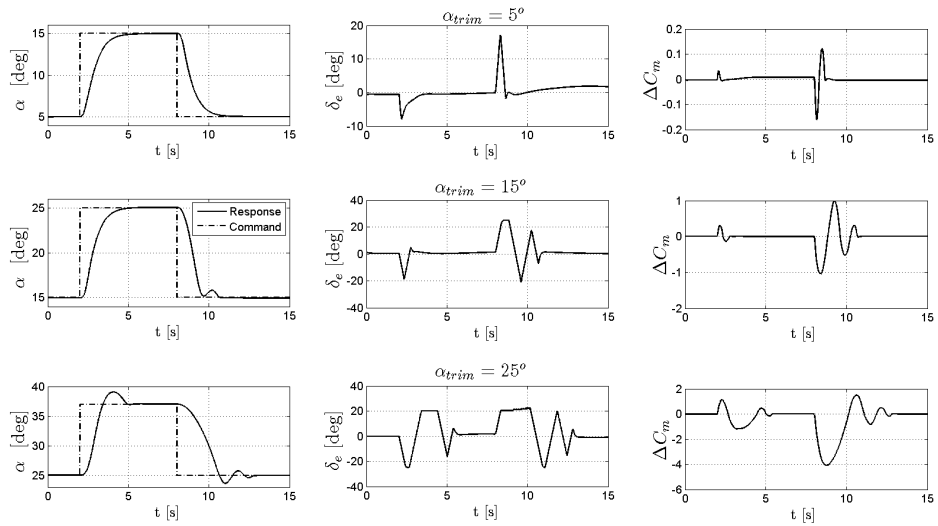
**Table 6. MATLAB execution times**

Condition	$\alpha_0 = 5^\circ$		$\alpha_0 = 15^\circ$		$\alpha_0 = 25^\circ$	
	average time [ms]	maximum time [ms]	average time [ms]	maximum time [ms]	average time [ms]	maximum time [ms]
Linear	14.5	17.4	14.8	18.3	14.7	17.1
Suc. linear	32.0	34.3	32.9	35.1	32.6	40.1
Nonlinear	17.6	80.1	19.4	80.0	22.3	94.9

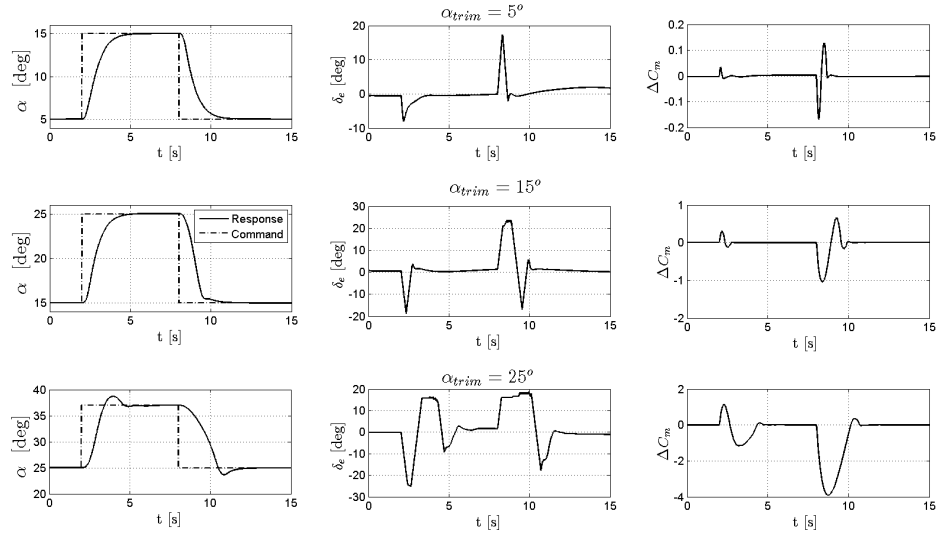




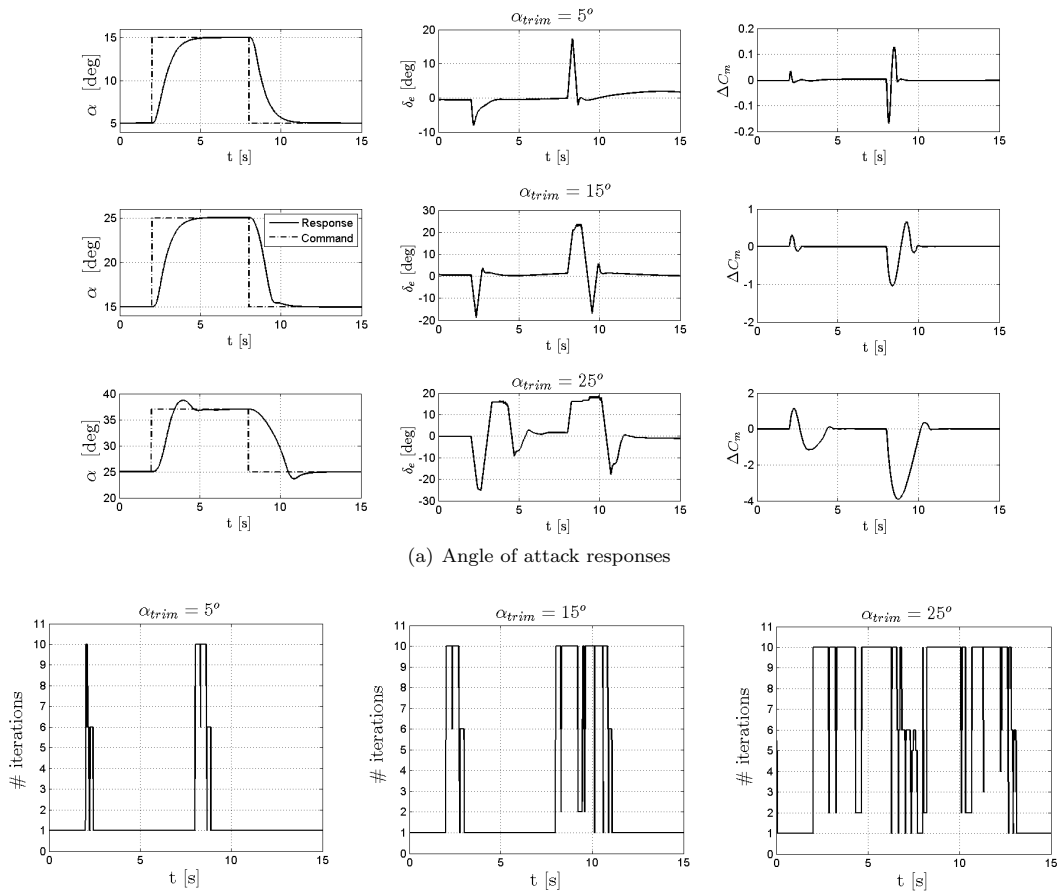
**Figure 4. SNDI with linear control allocation**



**Figure 5. SNDI with successive linear control allocation**



**Figure 6. SNDI with nonlinear control allocation**



**(b) Number of iterations performed by the nonlinear control allocation algorithm**

**Figure 7. SNDI with nonlinear control allocation**

## B. Performance Assessment

In this section the spline based NDI controller is evaluated. In order to evaluate the controller properly, its performance is compared with a polynomial based NDI controller using the models identified in section VI. To make a fair comparison, nonlinear control allocation is applied for both controllers. The control effort and required demand for the roll, pitch and yaw channels are combined in one least squares objective:

$$\min_{\delta_e, \delta_a, \delta_r} \mathcal{J} = \left\| \begin{bmatrix} C_l(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef}) - C_{l_{req}} \\ C_m(\alpha, \beta, \tilde{q}, \delta_e, \delta_{lef}) - C_{m_{req}} \\ C_n(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{lef}) - C_{n_{req}} \end{bmatrix} \right\| \quad (131)$$

for which the nonlinear control allocation algorithm is applied.

The performance assessment is conducted with two maneuvers which cover a large part of the flight envelope:

**Maneuver 1:** Roll command ( $\phi_{com} = 40^\circ$ ) and angle of attack command ( $\alpha_{com} = 15^\circ$ ) performed simultaneously with constant sideslip command  $\beta_{com} = 5^\circ$ : See figure 8 and table 7.

**Maneuver 2:** Roll command ( $\phi_{com} = 15^\circ$ ) and high angle of attack command ( $\alpha_{com} = 40^\circ$ ) with zero sideslip. The outer loop controller gains are decreased to:  $k_\phi = 1$ ,  $k_\alpha = 1.5$ ,  $k_\beta = 1$  figure 9 and table 8.

Each figure shows a comparison between the tracking response, the control inputs, the control allocation error given by eq. (130) and the model error between the true and estimated moment coefficients:

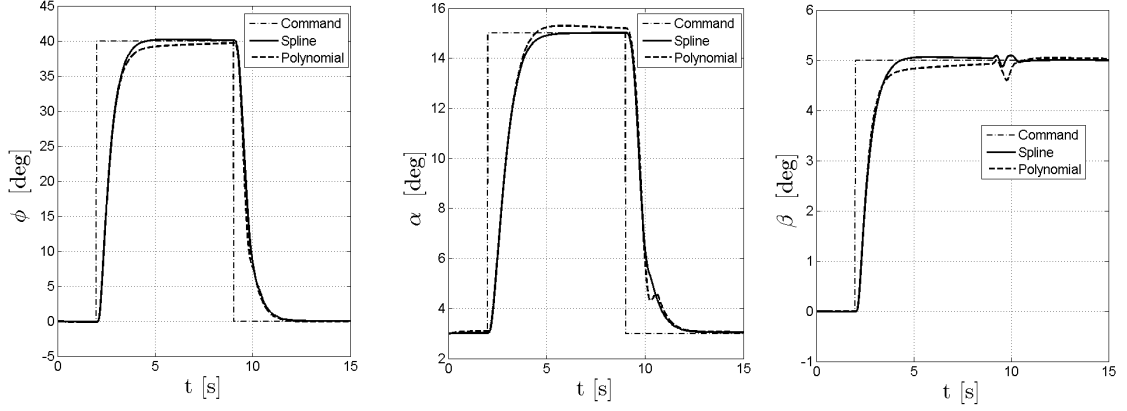
$$\xi C(t) = C(t) - \hat{C}(t) \quad (132)$$

An assessment of the performance is made based on the RMS values of the model errors and control allocation errors which are listed in the corresponding tables. The flight trajectories of the four maneuvers are shown in figure 10.

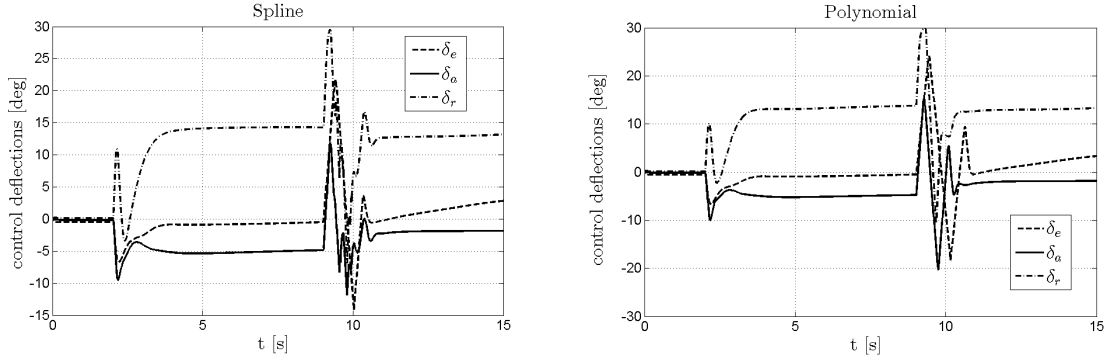
Maneuver 1 is conducted in the low angle of attack region which contains moderate nonlinear aerodynamics. The spline model is better able to accurately model these nonlinearities as compared to the polynomial model, resulting in lower model errors and in turn lower control allocation errors. Oscillations can be observed in the sideslip angle response which are caused by the actuator limits and coupling effect between the control axis due to the allocation errors.

Maneuver 2 is performed in the high angle of attack region which is very nonlinear compared to the low angle of attack region. In this region the controls saturate quickly and therefore the controllers gains are decreased. Furthermore, by decreasing the gains the effect of the model error on the controller performance can be better observed. Actuator saturation might actually mask this effect. In this maneuver a large part of the angle of attack region is traversed. In this region the nonlinearities have increased to the point that the polynomial based controller is unable to track the commanded angle of attack, while the spline based controller still shows adequate tracking performance. From figure 9 it can be observed that the polynomial based NDI controller is better able to stabilize the roll and sideslip angle compared to the spline based NDI controller. The actuators for the rudder and aileron of the SNDI controller saturate more quickly, resulting in larger control allocation errors, and in turn large oscillations for the roll and sideslip angle. It must be noted, however, that the polynomial based NDI controller fails to track the angle of attack reference of  $40^\circ$ , and as a result is operating at much lower angle of attack than the SNDI controller (i.e.  $15^\circ$  vs.  $40^\circ$  for the SNDI controller). In fact, the significant difference between the trajectories for Maneuver 4 in figure 10 clearly illustrates the capability of the SNDI controlled F-16 to out-maneuver the polynomial NDI controlled F-16.

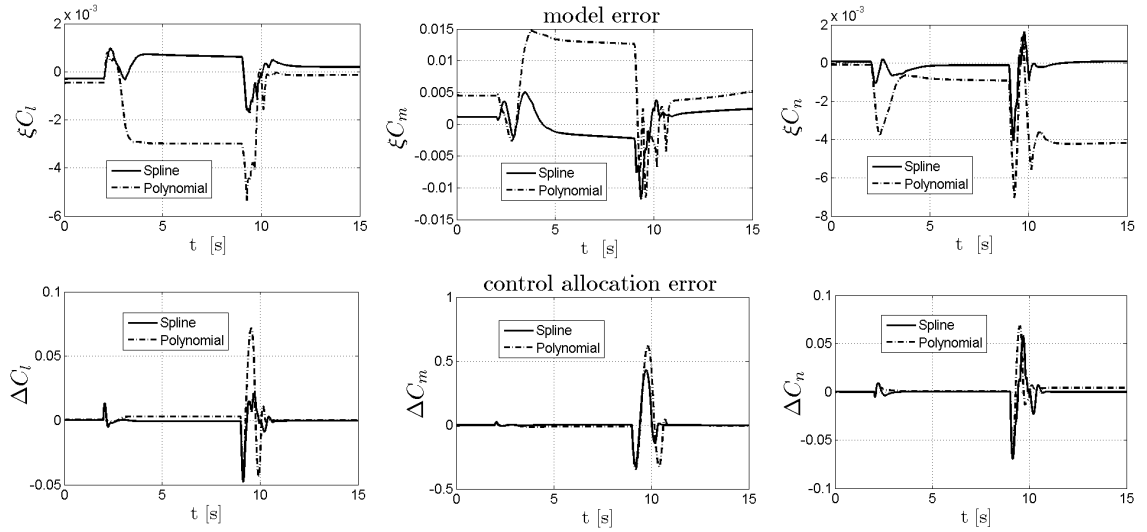
From these results it can be concluded that when operating in the linear part of the flight envelope, the use of SNDI does not provide a significant increase in tracking performance compared to polynomial NDI. However, in the operating region with significant nonlinear aerodynamics, SNDI provides a significant increase in controller performance resulting in improved maneuvering capabilities.



(a) Command variables



(b) Control deflections

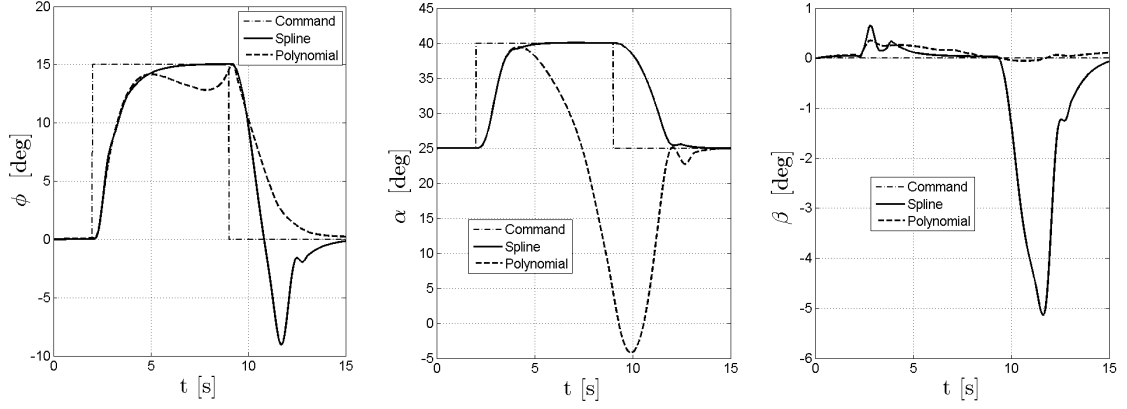


(c) Model errors and control allocation errors

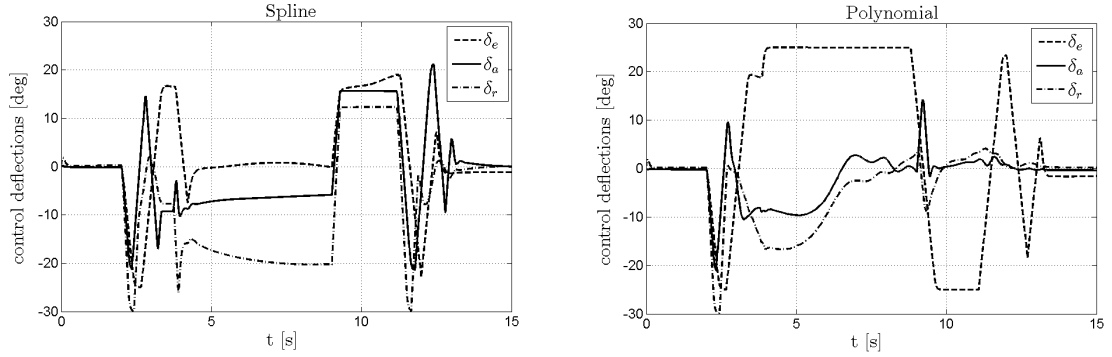
**Figure 8. Results maneuver 1: Roll command ( $\phi_{com} = 40^\circ$ ) and angle of attack ( $\alpha_{com} = 15^\circ$ ) command performed simultaneously with constant sideslip command  $\beta_{com} = 5^\circ$**

**Table 7. Performance parameters maneuver 2**

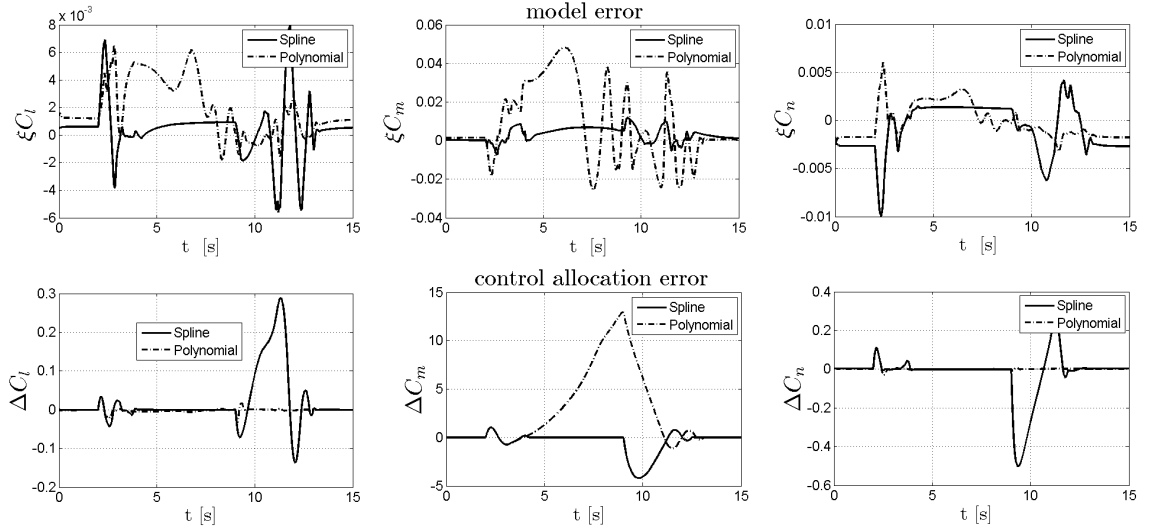
performance parameter	RMS $\xi C_l$	RMS $\xi C_m$	RMS $\xi C_n$	RMS $\Delta C_l$	RMS $\Delta C_m$	RMS $\Delta C_n$
spline	0.0005	0.0025	0.005	0.0055	0.0748	0.0102
polynomial	0.0021	0.0088	0.0028	0.0115	0.1135	0.0106



(a) Command variables



(b) Control deflections

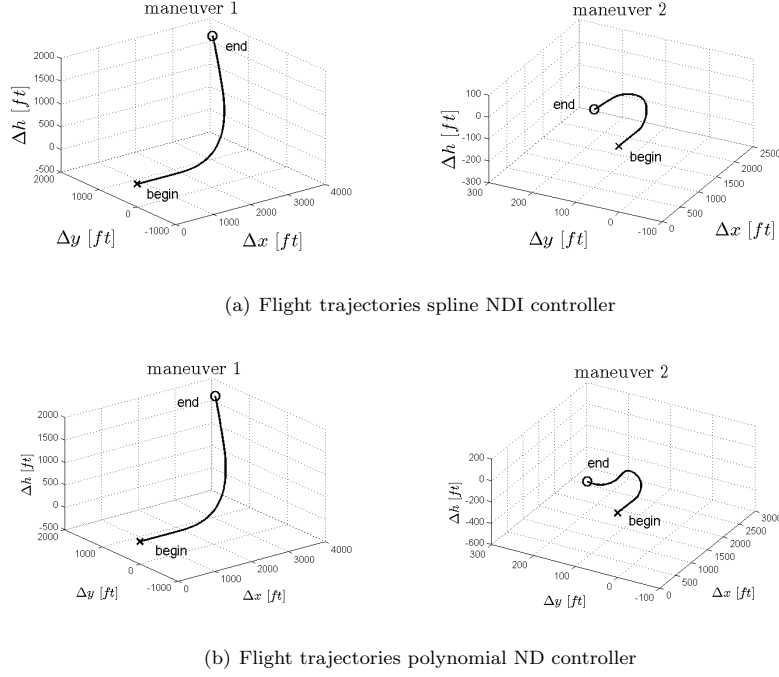


(c) Model errors and control allocation errors

**Figure 9. Results maneuver 2: Roll command ( $\phi_{com} = 15^\circ$ ) and high angle of attack command ( $\alpha_{com} = 40^\circ$ ) with with zero sideslip. The outer loop controller gains are decreased to:  $k_\phi = 1$ ,  $k_\alpha = 1.5$ ,  $k_\beta = 1$**

**Table 8. Performance parameters maneuver 4**

performance parameter	RMS $\xi C_l$	RMS $\xi C_m$	RMS $\xi C_n$	RMS $\Delta C_l$	RMS $\Delta C_m$	RMS $\Delta C_n$
spline	0.0020	0.0047	0.0027	0.0718	1.1883	0.1181
polynomial	0.0028	0.0212	0.0018	0.0061	4.7248	0.0130



**Figure 10. Flight trajectories of the four maneuvers**

## IX. Conclusions

High performance flight control systems based on the nonlinear dynamic inversion (NDI) principle require highly accurate models of aircraft aerodynamics. In this paper a new nonlinear control method, indicated as SNDI, is presented that combines NDI with multivariate spline model based control allocation. The goal of SNDI is to improve the tracking performance of current NDI based flight controllers by improving the accuracy of their on-board aerodynamic models. This is achieved by replacing current aerodynamic model implementations with multivariate simplex splines, a powerful type of function approximator.

Three new CA strategies are presented for the simplex spline approximators; a linear, a nonlinear, and a successive linear strategy. The linear CA strategy is computationally efficient, but can result in significant allocation errors in nonlinear regions of the flight envelope. The nonlinear approach produces the smallest allocation errors at the cost of having to solve a computationally expensive nonlinear optimization problem. The successive linear approach aims to strike a balance between computational efficiency and allocation error; it calculates the control input at a number of local linearization points and then selects the input resulting in the smallest allocation error. The choice of CA strategy depends on the available computational resources. On platforms with limited resources, the successive linear approach should be used. When sufficient computational resources are available, the nonlinear strategy is the preferred strategy.

The SNDI method is demonstrated with a number of simulated maneuvers using a high fidelity F-16 simulation. The tracking performance of SNDI is compared directly with NDI based on ordinary polynomial models in four high amplitude maneuvers flown with the F-16 simulation. The results show that SNDI provides a significantly improved tracking performance, especially in nonlinear regions of the flight envelope such as the high angle of attack and high angle of sideslip regions.

## Appendix A Tutorial on data approximation with multivariate simplex splines

In this appendix a simple 7-step tutorial example is presented of using multivariate simplex splines to approximate a 2-dimensional scattered (i.e. non-gridded) dataset. This dataset consists of the measurement locations  $\mathbf{x}$ , which are chosen arbitrarily as follows (see also figure 11):

$$\begin{aligned}\mathbf{x} &= [\mathbf{x}_1, \mathbf{x}_2] \\ &= \begin{bmatrix} 0 & 0.3 & 0.5 & 0.6 & 1.0 & 1.0 & 0 & 0.2 & 0.6 & 0.8 \\ 1.0 & 0.5 & 0.9 & 0.8 & 0 & 1.0 & 0 & 0.1 & 0.2 & 0.7 \end{bmatrix}^T,\end{aligned}$$

this dataset can be seen as independent measurements made on some aircraft states. The dependent measurement values are generated with a sine function as follows:

$$\begin{aligned}y(\mathbf{x}) &= \sin(\mathbf{x}_1 + \mathbf{x}_2) \\ &= \begin{bmatrix} 0.841 & 0.717 & 0.985 & 0.985 & 0.841 & 0.909 & 0 & 0.296 & 0.717 & 0.997 \end{bmatrix}^T,\end{aligned}$$

which can be seen as the values calculated for a force or moment coefficient. The aim is to approximate  $y(\mathbf{x})$  with a bivariate simplex spline function of degree 2 and continuity order 1, that is, the spline function  $s(\mathbf{x}) \in \mathcal{S}_2^1(\mathcal{T})$  that minimizes  $\|s(\mathbf{x}) - y(\mathbf{x})\|$  on the triangulation  $\mathcal{T}$ .

**Step 1:** Define the triangulation. In this case, a Delaunay triangulation is created of the convex hull of  $\mathbf{x}$ , resulting in a triangulation consisting of 2 triangles  $t_1 = \langle \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_3 \rangle$  and  $t_2 = \langle \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \rangle$ , see figure 11.

**Step 2:** Explicitly define the spline model structure in the form of (3). In this case, the spline model structure is as follows:

$$\begin{aligned}s(\mathbf{x}) &= \delta_1(\mathbf{x})p^{t_1}(\mathbf{x}) + \delta_2(\mathbf{x})p^{t_2}(\mathbf{x}) \\ &= \delta_1(\mathbf{x})B_{t_1}^2(\mathbf{x})\mathbf{c}^{t_1} + \delta_2(\mathbf{x})B_{t_2}^2(\mathbf{x})\mathbf{c}^{t_2}.\end{aligned}\tag{A.1}$$

The structure of the individual B-form polynomials  $B_{t_j}^2(\mathbf{x})\mathbf{c}^{t_j}$  in (A.1) is found by first determining the set of multi-indices  $\kappa$  using (15) and the fact that  $d = |\kappa| = 2$ :

$$\kappa \in \{(2, 0, 0), (1, 1, 0), (1, 0, 1), (0, 2, 0), (0, 1, 1), (0, 0, 2)\}$$

Clearly, there are 6 (i.e.  $\hat{d} = 6$ ) valid multi-indices, resulting in B-form polynomials consisting of 6 individual basis functions  $B_\kappa^2(b(\mathbf{x}))$ . Using this result with (17) the vectors of per-triangle basis polynomials  $B^2(\mathbf{x})$  for both triangles can then be constructed as follows:

$$\begin{aligned}B_{t_j}^2(\mathbf{x}) &= \begin{bmatrix} B_{2,0,0}^2(b(\mathbf{x})) & B_{1,1,0}^2(b(\mathbf{x})) & B_{1,0,1}^2(b(\mathbf{x})) & B_{0,2,0}^2(b(\mathbf{x})) & B_{0,1,1}^2(b(\mathbf{x})) & B_{0,0,2}^2(b(\mathbf{x})) \end{bmatrix}, \quad j = 1, 2 \\ &= \begin{bmatrix} b_0^2 & 2b_0b_1 & 2b_0b_2 & b_1^2 & 2b_1b_2 & b_2^2 \end{bmatrix}, \quad j = 1, 2\end{aligned}$$

the corresponding global vector of basis functions is:

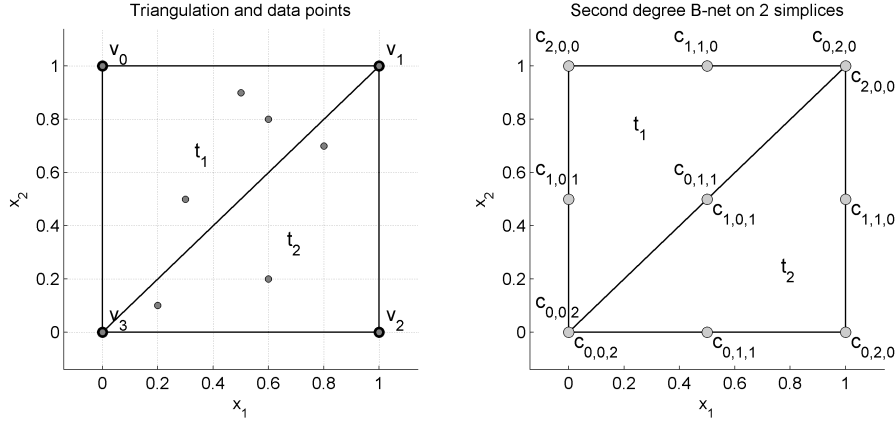
$$B^2(\mathbf{x}) = \begin{bmatrix} B_{t_1}^2(\mathbf{x}) & B_{t_2}^2(\mathbf{x}) \end{bmatrix}$$

The vectors of per-triangle B-coefficients  $\mathbf{c}^{t_j}$  then are:

$$\begin{aligned}\mathbf{c}^{t_1} &= [c_{200}^{t_1} \quad c_{110}^{t_1} \quad c_{101}^{t_1} \quad c_{020}^{t_1} \quad c_{011}^{t_1} \quad c_{002}^{t_1}]^T \\ \mathbf{c}^{t_2} &= [c_{200}^{t_2} \quad c_{110}^{t_2} \quad c_{101}^{t_2} \quad c_{020}^{t_2} \quad c_{011}^{t_2} \quad c_{002}^{t_2}]^T\end{aligned}$$

and the corresponding global B-coefficient vector is:

$$\mathbf{c} = [\mathbf{c}^{t_1} \quad \mathbf{c}^{t_2}]^T$$



**Figure 11.** A triangulation consisting of 2 triangles with a 2-dimensional scattered dataset (left), and (right) the

**Step 3:** Assign measurement data to simplices, and calculate respective barycentric coordinates. The following data assignment is made:

$$\mathbf{x}^{t_1} = \mathbf{x}(i) \in t_1 : i = 1, 2, 3, 4, \quad \mathbf{x}^{t_2} = \mathbf{x}(i) \in t_2 : i = 5, 6, 7, 8, 9, 10$$

that is,  $t_1$  and  $t_2$  contain respectively 4 and 6 data points. Note that while data points  $\mathbf{x}(6)$  and  $\mathbf{x}(7)$  located respectively at vertices  $\mathbf{v}_1$  and  $\mathbf{v}_3$  are assigned to  $t_2$ , they could also have been assigned to  $t_1$  or to both  $t_1$  and  $t_2$ . Using (9) and (10) to calculate the barycentric coordinates of  $\mathbf{x}^{t_1}$  and  $\mathbf{x}^{t_2}$  results in:

$$b(\mathbf{x}^{t_1}) = \begin{bmatrix} 1.0 & 0 & 0 \\ 0.2 & 0.3 & 0.5 \\ 0.4 & 0.5 & 0.1 \\ 0.2 & 0.6 & 0.2 \end{bmatrix}, \quad b(\mathbf{x}^{t_2}) = \begin{bmatrix} 0 & 1.0 & 0 \\ 1.0 & 0 & 0 \\ 0 & 0 & 1.0 \\ 0.1 & 0.1 & 0.8 \\ 0.2 & 0.4 & 0.4 \\ 0.7 & 0.1 & 0.2 \end{bmatrix}$$

Note that Matlab provides the built-in function `tsearchn` which calculates both data membership and barycentric coordinates of a given dataset and triangulation.

**Step 4:** Formulate the B-form regression matrix. Using (26) the following regression model structure is found for a single observation  $i$ :

$$\begin{aligned} y(i) &= B^2(i)D(i)\mathbf{c} + \epsilon(i) \\ &= X(i)\mathbf{c} \end{aligned}$$

for example, for the third ( $i = 3$ ) observation  $\mathbf{x}(3) = (0.5, 0.9)$  (and  $b(\mathbf{x}(3)) = (0.4, 0.5, 0.1)$ ) the regression model is:

$$\begin{aligned} 0.985 &= \begin{bmatrix} B^2(3) & B^2(3) \end{bmatrix} D(3)\mathbf{c} + \epsilon(3) \\ &= \begin{bmatrix} B^2(3) & B^2(3) \end{bmatrix} \begin{bmatrix} I_{6 \times 6} & 0_{6 \times 6} \\ 0_{6 \times 6} & 0_{6 \times 6} \end{bmatrix} \mathbf{c} + \epsilon(3) \\ &= \begin{bmatrix} b_0^2 & 2b_0b_1 & 2b_0b_2 & b_1^2 & 2b_1b_2 & b_2^2 & 0_{1 \times 6} \end{bmatrix} \mathbf{c} \\ &= \begin{bmatrix} 0.16 & 0.4 & 0.08 & 0.25 & 0.1 & 0.01 & 0_{1 \times 6} \end{bmatrix} \mathbf{c} + \epsilon(3) \end{aligned}$$



The complete regression matrix  $X$  for the given set of 10 observations is:

$$X = \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.04 & 0.12 & 0.2 & 0.09 & 0.3 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.16 & 0.4 & 0.08 & 0.25 & 0.1 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.04 & 0.24 & 0.08 & 0.36 & 0.24 & 0.04 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 & 0.02 & 0.16 & 0.01 & 0.16 & 0.64 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.04 & 0.16 & 0.16 & 0.16 & 0.32 & 0.16 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.49 & 0.14 & 0.28 & 0.01 & 0.04 & 0.04 \end{bmatrix}$$

**Step 5:** Formulate the smoothness matrix using the theory from [11]. The continuity conditions for the given triangulation are formulated using (31) which must be adapted for the currently used triangulation in the sense that the location of the '0' and the 'm' in the multi-index is determined by the non-zero value in the multi-index of the B-coefficient located at the respective *out-of-edge* vertices, see [11] for more details. In this case the general continuity conditions for continuity between  $t_1$  and  $t_2$  are found by reformulating (31) into:

$$c_{(m,\kappa_0,\kappa_1)}^{t_1} = \sum_{|\gamma|=m} c_{(\kappa_0,0,\kappa_1)+\gamma}^{t_2} B_{\gamma}^m(\mathbf{v}_0) = c_{(\kappa_0,0,\kappa_2)}^{t_2}$$

The  $C^0$  continuity conditions (i.e.  $m=0$ ) of  $t_1$  with respect to  $t_2$  are:

$$c_{(0,\kappa_0,\kappa_1)}^{t_1} = \sum_{|\gamma|=0} c_{(\kappa_0,0,\kappa_1)}^{t_2} B_{\gamma}^0(\mathbf{v}_0) = c_{(\kappa_0,0,\kappa_2)}^{t_2}$$

The  $C^1$  continuity conditions (i.e.  $m=1$ ) of  $t_1$  with respect to  $t_2$  are:

$$c_{(1,\kappa_0,\kappa_1)}^{t_1} = \sum_{|\gamma|=1} c_{(\kappa_0,0,\kappa_1)+\gamma}^{t_2} B_{\gamma}^1(\mathbf{v}_0)$$

for example, the  $C^1$  continuity condition for the coefficient  $c_{1,1,0}^{t_1}$  is:

$$\begin{aligned} c_{(1,1,0)}^{t_1} &= \sum_{|\gamma|=1} c_{(1,0,0)+\gamma}^{t_2} B_{\gamma}^1(\mathbf{v}_0) \\ &= c_{2,0,0}^{t_2} B_{1,0,0}^1(\mathbf{v}_0) + c_{1,1,0}^{t_2} B_{0,1,0}^1(\mathbf{v}_0) + c_{1,0,1}^{t_2} B_{0,0,1}^1(\mathbf{v}_0) \\ &= c_{2,0,0}^{t_2} - c_{1,1,0}^{t_2} + c_{1,0,1}^{t_2} \end{aligned}$$

where it should be noted that  $b(\mathbf{v}_0) = (1, -1, 1)$ . The complete smoothness matrix is constructed by formulating the continuity conditions for all continuity orders and for all edges, moving all term to the right hand side. The smoothness matrix for  $C^1$  continuity between  $t_1$  and  $t_2$  then is:

$$H = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 \end{bmatrix}$$

where the first three rows correspond to the  $C^0$  conditions, and the last two rows to the  $C^1$  conditions.

**Step 6:** Formulate parameter estimation problem. The LS estimator for the B-coefficients can now be formulated these results with (29). In this case, the following values for the B-coefficients are estimated:

$$\hat{\mathbf{c}} = \begin{bmatrix} 0.842 & 1.1 & 0.626 & 0.926 & 1.23 & -0.0192 & 0.926 & 1.05 & 1.23 & 0.841 & 0.581 & -0.0192 \end{bmatrix}^T$$

**Step 7: Model validation.** The approximation accuracy of the simplex spline function can now be determined by evaluating the spline function at specific test points. For example, if the current spline function is evaluated at  $\mathbf{x}$  the following set of function outputs is found:

$$\begin{aligned} s(\mathbf{x}) &= X\hat{\mathbf{c}} \\ &= \begin{bmatrix} 0.842 & 0.737 & 0.979 & 0.975 & 0.841 & 0.926 & -0.0192 & 0.315 & 0.719 & 0.975 \end{bmatrix}^T \end{aligned}$$

## Appendix B Example: non-affine spline models in control allocation

This example illustrates the non-affine nature of spline models and the practical implementation of the control allocation strategies in Sec. VII. Consider the following spline model for  $\tau$ :

$$\tau(\mathbf{x}) = \delta_1 p^{t_1}(b(\mathbf{x})) + \delta_2 p^{t_2}(b(\mathbf{x})) + \cdots + \delta_J p^{t_J}(b(\mathbf{x})), \quad 1 \leq j \leq J$$

Each basis polynomial is defined on an individual simplex  $t_j$  (see also figure 11) in terms of local barycentric coordinates  $(b_0, b_1, \dots, b_n)$ . The Cartesian to barycentric coordinate transformation is a linear one-to-one transformation given by eqs. (9),(10). Let  $\tau(\mathbf{x})$  be a bivariate spline function ( $n = 2$ ) with  $b(\mathbf{x}) = (b_0, b_1, b_2)$  and with the spline state  $\mathbf{x}$  consisting of one aircraft state and one control input:  $\mathbf{x} = [x_a \ u]^T$ . The first step is to select the basis polynomial  $p^{t_j}$  in which the current state  $(x_a(t_0), u(t_0))$  is defined for the control allocation process:

$$p^{t_j}(b(\mathbf{x})) = \sum_{\kappa} c_{\kappa}^{t_j} \frac{d!}{\kappa!} b_0^{\kappa_0} b_1^{\kappa_1} b_2^{\kappa_2}, \quad (x_a, u) \in t_j \quad (\text{B.1})$$

In the right hand side of eq. (B.1) the explicit representation for the B-form polynomial  $p^{t_j}$  given by eq. (11) is used. The next step is to parameterize the B-form polynomial in terms of the control input  $u$  for a fixed aircraft state  $x_a$ . Let simplex  $t_j$  be given by:

$$t_j = \langle (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2) \rangle = \left\langle \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \right\rangle$$

Using eq. (9) and (10) it follows that the barycentric components of  $[x_a \ u]^T$  are given by:

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ u \end{bmatrix} \quad (\text{B.2})$$

$$b_0 = 1 - b_1 - b_2 = 1 - x_a \quad (\text{B.3})$$

Combining eqs. (B.2),(B.3) and writing as an affine function of the spline state gives:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ u \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{bmatrix}_{t_j} \begin{bmatrix} x_a \\ u \end{bmatrix} + \mathbf{k} \quad (\text{B.4})$$

At the current aircraft state  $x_a(t_0)$  the barycentric coordinates can be parametrized as an affine function of  $u$  as follows:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} u + \begin{bmatrix} 1 - x_a(t_0) \\ x_a(t_0) \\ 0 \end{bmatrix} = \mathbf{a}_1 u + \tilde{\mathbf{k}} \quad (\text{B.5})$$

By substituting the parameterizations in eq. (B.5) for  $b_0, b_1$  and  $b_2$  in eq. (B.1), the simplex polynomial can be expressed as a function only dependent of  $u$ :

$$\begin{aligned} p^{t_j}(u) &= \sum_{\kappa} c_{\kappa}^{t_j} \frac{d!}{\kappa!} (1 - x_a(t_0))^{\kappa_0} (-u + x_a(t_0))^{\kappa_1} u^{\kappa_2} \\ &= \sum_{\kappa} c_{\kappa}^{t_j} B_{\kappa}^d(u) \end{aligned} \quad (\text{B.6})$$

By theorem 1 the gradient of the basis polynomial  $p^{t_j}$  with respect to  $u$  is given by:

$$\begin{aligned}
\nabla_u p^{t_j}(u) &= \frac{p^{t_j}(u)}{\partial u} = \mathbf{a}_2^T \sum_{\kappa} c_{\kappa}^{t_j} \nabla_b B_{\kappa}^d(u) \\
&= \begin{bmatrix} 0 & -1 & 1 \end{bmatrix} \left( \sum_{\kappa} c_{\kappa}^{t_j} \frac{d!}{\kappa!} \begin{bmatrix} \kappa_0 (1 - x_a(t_0))^{\kappa_0-1} (-u + x_a(t_0))^{\kappa_1} u^{\kappa_2} \\ (1 - x_a(t_0))^{\kappa_0} \kappa_1 (-u + x_a(t_0))^{\kappa_1-1} u^{\kappa_2} \\ (1 - x_a(t_0))^{\kappa_0} (-u + x_a(t_0))^{\kappa_1} \kappa_2 u^{\kappa_2-1} \end{bmatrix} \right) \\
&= \mathbf{a}_2^T \nabla_b B_{t_j}^d(u) \mathbf{c}^{t_j}
\end{aligned} \tag{B.7}$$

With the parameterization of the simplex polynomial and the derivation of the gradient the three control allocation strategies in Sec. VII can be applied. For example, applying the linear strategy the incremental control input for a required demand  $\tau_{req}$  is given by:

$$\Delta u = (\nabla_u p^{t_j}(u(t_0)))^{-1} (\tau_{req} - p^{t_j}(u(t_0))) \tag{B.8}$$

With the nonlinear strategy this process is repeated and the solution is iterated through the Levenberg-Marquardt algorithm.

## References

- <sup>1</sup>Reiner, J., Balas, G. J., and Garrard, W. L., "Flight Control Design Using Robust Dynamic Inversion and Time-Scale Separation," *Automatica*, Vol. 32, No. 11, 1996, pp. 1493–1504.
- <sup>2</sup>Lane, S. H. and Stengel, R. F., "Flight Control Design Using Non-linear Inverse Dynamics," *Automatica*, Vol. 24, No. 4, 1988, pp. 471–483.
- <sup>3</sup>Walker, G. P. and Allen, D. A., "X-35b stovl Flight Control Law Design and Flying Qualities," *Biennial International Powered Lift Conference and Exhibit*, 2002.
- <sup>4</sup>Durham, W. C., "Constrained Control Allocation," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 16, 1993, pp. 717–725.
- <sup>5</sup>Reigelsperger, W. C. and Banda, S. S., "Nonlinear Simulation of a Modified F-16 with Full-envelope Control Laws," *Control Engineering Practice*, Vol. 6, No. 3, 1998, pp. 309–320.
- <sup>6</sup>Sieberling, S., "Robust Flight control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 33, 2010, pp. 1732–1742.
- <sup>7</sup>Kim, B. S. and Calise, A. J., "Nonlinear Flight Control Using Neural Networks," *AIAA Journal of Guidance, Control and Dynamics*, Vol. 20, No. 1, 1997, pp. 26–33.
- <sup>8</sup>Calise, A. J., Lee, S., and Sharpe, M., "Nonlinear Adaptive Flight Control Using Neural Networks," *IEEE Control Systems*, Vol. 18, 1998, pp. 14–25.
- <sup>9</sup>Calise, A. J., Lee, S., and Sharpe, M., "Development of a Reconfigurable Flight Control Law for Tailles Aircraft," *AIAA Journal of Guidance, Control and Dynamics*, Vol. 24, No. 5, 2001, pp. 896–902.
- <sup>10</sup>Lai, M. J. and Schumaker, L. L., *Spline Functions over Triangulations*, Cambridge University Press, 2007.
- <sup>11</sup>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.
- <sup>12</sup>de Visser, C. C., Chu, Q. P., and Mulder, J. A., "Global Nonlinear Aerodynamic Model Identification with Multivariate Splines," *AIAA Atmospheric Flight Mechanics Conference*, No. AIAA-2009-5726, 2009.
- <sup>13</sup>de Visser, C. C., Chu, Q. P., and Mulder, J. A., "A Multidimensional Spline Based Global Nonlinear Aerodynamic Model for the Cessna Citation II," *AIAA Atmospheric Flight Mechanics Conference*, No. AIAA-2010-7950, 2010.
- <sup>14</sup>de Visser, C. C., Chu, Q. P., and Mulder, J. A., "Validating the Multidimensional Spline Based Global Aerodynamic Model for the Cessna Citation II," *AIAA Atmospheric Flight Mechanics Conference*, No. AIAA-2011-6356, 2011.
- <sup>15</sup>de Visser, C. C., van Kampen, E., Chu, Q. P., and Mulder, J. A., "Intersplines: A New Approach to Globally Optimal Multivariate Splines Using Interval Analysis," *Reliable Computing*, Vol. 17, 2012.
- <sup>16</sup>Nguyen, L. T., Ogburn, M. E., Gilbert, W. P., Kibler, K. S., Brown, P. W., and Deal, P. L., "Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability," Tech. Rep. 1538, NASA, 1979.
- <sup>17</sup>Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*, John Wiley & Sons, Hoboken, New Jersey, 2nd ed., 2003, pp. 107–116, Appendix A pp.633–664.
- <sup>18</sup>de Visser, C. C., Chu, Q. P., and Mulder, J. A., "Differential Constraints for Bounded Recursive Identification with Multivariate Splines," *Automatica*, Vol. 47, No. 9, 2011, pp. 2059–2066.
- <sup>19</sup>Hu, X. L., Han, D. F., and Lai, M. J., "Bivariate Splines of Various Degrees for Numerical Solution of Partial Differential Equations," *SIAM Journal on Scientific Computing*, Vol. 29, 2007, pp. 1338 – 1354.
- <sup>20</sup>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, TN, Athens, 2005.

- <sup>21</sup>Lombaerts, T. J. J., Looye, G. H. N., Chu, Q. P., and Mulder, J. A., "Design and Simulation of Fault Tolerant Flight Control Based on a Physical Approach," *Aerospace Science and Technology*, Vol. 23, No. 1, 2012, pp. 151–171.
- <sup>22</sup>Bodson, M., "Evaluation of Optimization Methods for Control Allocation," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 703–711.
- <sup>23</sup>Härkegard, O., "Efficient Active Set Algorithms for Solving constrained Least Squares problems in Aircraft Control Allocation," *Proceedings IEEE conference on decision and control*, Vol. 2, 2002, pp. 1295–1300.
- <sup>24</sup>Davidson, J. B., Lallman, F. J., and Bundick, W. T., "Real-time Adaptive Control Allocation applied to a High Performance Aircraft," *5th SIAM Conference on Control & its Applications*, 2001.
- <sup>25</sup>Alwi, H. and Edwards, C., "Fault Tolerant Control Using Sliding Modes with On-line Control Allocation," *Automatica*, Vol. 44, 2007, pp. 1859–1866.
- <sup>26</sup>Eberhardt, R. and Ward, D. G., "Indirect Adaptive Flight Control System Interactions," *International Journal of Robust and Nonlinear Control*, Vol. 9, No. 14, 1999, pp. 1013–1031.
- <sup>27</sup>Virnig, J. C. and Bodden, D. S., "Multivariable Control Allocation and Control Law Conditioning when Control Effectors Limit," *AIAA Guidance, Navigation and Control Conference and Exhibit*, No. AIAA-94-3609-CP, 1994.
- <sup>28</sup>Morelli, E. A., "Global Nonlinear Parametric Modeling with Application to F-16 Aerodynamics," Tech. rep., NASA, 1997.
- <sup>29</sup>Morelli, E. A., "Global Nonlinear Aerodynamic Modeling using Multivariate Orthogonal Functions," *AIAA Journal of Aircraft*, Vol. 32, No. 2, 1995, pp. 270–277.
- <sup>30</sup>Abramowitz, M. and Stegun, I. A., *Handbook of mathematical functions*, National Bureau of Standards, Washington, D.C. 20402, 1964.
- <sup>31</sup>Doman, D. B., Ngo, A., Leggett, D. B., Saliers, M. A., and Pachter, M., "Development of a Hybrid Direct-Indirect Adaptive Control System for the X-33," *Proceedings of the AIAA Guidance, Control, and Dynamics Conference*, 2000.
- <sup>32</sup>Luo, Y., Yurkovich, S., Oppenheimer, M. W., and Doman, D. B., "Model-Predictive Dynamic Control Allocation Scheme for Reentry Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 30, 2007, pp. 100–113.
- <sup>33</sup>Johansen, T. A., Fossen, T. I., and Berge, S. P., "Constrained Nonlinear Control Allocation with Singularity Avoidance Using Sequential Quadratic Programming," *IEEE Transactions on Control Systems Technology*, Vol. 12, No. 1, 2004.
- <sup>34</sup>Marquardt, D. W., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM J. Appl. Math.*, Vol. 111, 1963, pp. 431–441.
- <sup>35</sup>Johansen, T. A., "Optimizing Nonlinear Control Allocation," *43rd IEEE Conference on Decision and Control*, Vol. 4, 2004, pp. 3435 – 3440.