JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 37, No. 6, November–December 2014

Nonlinear Multivariate Spline-Based Control Allocation for High-Performance Aircraft

H. J. Tol,* C. C. de Visser,† E. van Kampen,‡ and Q. P. Chu§
Delft University of Technology, 2629 HS Delft, The Netherlands

DOI: 10.2514/1.G000065

High-performance flight control systems based on the nonlinear dynamic inversion 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 nonlinear dynamic inversion with multivariate simplex spline-based control allocation. Three control allocation strategies that use novel expressions for the analytic Jacobian and Hessian of the multivariate spline models are presented. 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 nonlinear spline based controller is applied to control a high-performance aircraft (F-16) with a large flight envelope. The simulation results indicate that perfect feedback linearization can be achieved throughout the entire flight envelope, leading to a significant increase in tracking performance compared with ordinary polynomial-based nonlinear dynamic inversion.

Nomenclature

A_x, A_y, A_x	=	specific forces along the body $x/y/z$ axes, m/s^2
b	=	wing span, m
$b(\boldsymbol{x})$	=	barycentric transform of point $\boldsymbol{x}$
C	=	dimensionless coefficient
$\boldsymbol{c}$	=	B-coefficient vector
$\bar{c}$	=	mean aerodynamic chord, m
$\hat{d}$	=	total number of valid permutations
H	=	smoothness matrix
I	=	inertia matrix
J	=	total number of simplices
$\mathcal{J}$	=	cost function
l, m, n	=	aerodynamic moment around the body $x/y/z$ axes
p, q, r	=	roll, pitch, and yaw rate around the body $x/y/z$ axes, rad/s
p_s	=	static pressure, Pa
$\bar{q}$	=	dynamic pressure, Pa
S	=	wing area, m^2
S_d^r	=	spline space of degree d and continuity order r
$\mathcal{T}$	=	triangulation
t_j	=	simplex j
$\boldsymbol{u}$	=	input vector
u, v, w	=	velocity components along the body $x/y/z$ axes, m/s
V	=	airspeed, m/s
X	=	regression matrix
$\boldsymbol{x}$	=	state vector
$\boldsymbol{Y}$	=	observation vector
α, β	=	angle of attack and sideslip angle, rad
δ	=	control surface deflection, rad
λ	=	actuator lag time constant
$\boldsymbol{\epsilon}$	=	residual vector
κ	=	multi-index
$\boldsymbol{\nu}$	=	virtual input
ρ	=	air density, kg/m^3
$\boldsymbol{\tau}$	=	virtual input
ϕ, θ, ψ	=	roll, pitch, and yaw angles, rad

Subscripts

a	=	aileron
e	=	elevator
r	=	rudder
lef	=	leading-edge flap

Presented as Paper 2013-4925 at the AIAA Guidance, Navigation, and Control Conference, Boston, MA, 19–22 August 2013; received 31 May 2013; revision received 10 December 2013; accepted for publication 16 December 2013; published online 21 April 2014.

*Researcher, Faculty of Aerospace Engineering, Control and Simulation Division; hendrikus.tol@gmail.com.

†Assistant Professor, Faculty of Aerospace Engineering, Control and Simulation Division; c.c.devisser@tudelft.nl.

‡Assistant Professor, Faculty of Aerospace Engineering, Control and Simulation Division; E.vanKampen@tudelft.nl.

§Associate Professor, Faculty of Aerospace Engineering, Control and Simulation Division; q.p.chu@tudelft.nl.

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, whereas 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.

Because 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 μ 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 that 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 make 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 nonaffine in the control inputs, requiring a special coordinate transformation scheme to relate them to the aircraft states and control inputs [15].

The main contribution of this paper is a new nonlinear control scheme, indicated as SNDI, that combines nonlinear dynamic inversion with control allocation based an onboard simplex spline models. This contribution requires the development of new CA strategies that can be applied to simplex spline models that are nonaffine 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. Although the current SNDI focuses primarily on reducing static aerodynamic modeling errors, it should be seen as the first step toward 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. IV, 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 Sec. 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 $\bar{q}/p_s$ to optimize performance and has the following relationship [16]:

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

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

$$\dot{u} = \frac{1}{\lambda}(u_{\text{com}} - u) \tag{2}$$

in which the commanded input is bounded by $u_{\min} \le u_{\text{com}} \le u_{\max}$ and the deflection rate is bounded by $|\dot{u}| \le \dot{u}_{\text{lim}}$. 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 simulations are performed with a sample frequency of 100 Hz.

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 [11,18]. 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:

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

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

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

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 nonoverlapping simplices:

Table 1 Actuator model [16] and ([17] pp. 633–664)

	Deflection limit, deg	Rate limit, deg /s	Time constant, s lag
Elevator	±25.0	60	0.0495
Ailerons	±21.5	80	0.0495
Rudder	±30.0	120	0.0495
Leading-edge flap	0–25	25	0.136

$$\mathcal{T} := \bigcup_{j=1}^{J} t_j, \qquad t_i \cap t_j \in \{\varnothing, \tilde{t}\}, \qquad \forall\ t_i, t_j \in \mathcal{T} \tag{5}$$

with the edge simplex $\tilde{t}$ a k simplex with $0 \le k \le n-1$. 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):

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

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: $p^t(b(\boldsymbol{x}))$. The n-simplex t is defined as the convex hull of a set of $n+1$ unique, nondegenerate points in n-dimensional space:

$$t = \langle \boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \rangle \tag{7}$$

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

$$\boldsymbol{x} = \sum_{i=0}^{n} b_i \boldsymbol{v}_i, \qquad \sum_{i=0}^{n} b_i = 1 \tag{8}$$

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

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = [\,(\boldsymbol{v}_1 - \boldsymbol{v}_0) \quad (\boldsymbol{v}_2 - \boldsymbol{v}_0) \quad \cdots \quad (\boldsymbol{v}_n - \boldsymbol{v}_0)\,]^{-1}(\boldsymbol{x} - \boldsymbol{v}_0)$$
$$= \Lambda(\boldsymbol{x} - \boldsymbol{v}_0) \tag{9}$$

and

$$b_0 = 1 - \sum_{i=1}^{n} b_i \tag{10}$$

C. *B* Form

Each polynomial $p^{t_j}(b(\boldsymbol{x}))$ in Eq. (3) is expressed in the B form

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

with c_κ the B coefficient and κ the multi-index defined as

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

The multi-index has the following properties:

$$|\kappa| = \kappa_0 + \kappa_1 + \cdots + \kappa_n = d \tag{13}$$

$$\kappa! = \kappa_0! \kappa_1! \cdots \kappa_n! \tag{14}$$

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

$$\kappa \in \{(d, 0, 0 \cdot\cdot, 0), (d-1, 1, 0 \cdot\cdot, 0), (d-1, 0, 1, \cdot\cdot, 0), \ldots, (0, \cdot\cdot, 0, 1, d-1), (0, \cdot\cdot, 0, 0, d)\} \tag{15}$$

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

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

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

$$B_{t_j}^d(b(\boldsymbol{x})) := \left[\, B_{d,0,\cdot\cdot,0}^d(b(\boldsymbol{x})) \quad B_{d-1,1,\cdot\cdot,0}^d(b(\boldsymbol{x})) \quad \cdots \quad B_{0,\cdot\cdot,0,d}^d(b(\boldsymbol{x})) \,\right] = [B_\kappa^d(b(\boldsymbol{x}))]_{|\kappa=d|} \in \mathbb{R}^{1\times\hat{d}} \tag{17}$$

and the vector of B coefficients

$$\boldsymbol{c}^{t_j} := [c_\kappa^{t_j}]_{|\kappa|=d} \in \mathbb{R}^{\hat{d}\times 1} \tag{18}$$

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

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

D. 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 Eqs. (3) and (19), the B-form polynomials can be used as regressors for a new observation pair $(\boldsymbol{x}(i), y(i))$ as follows:

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

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

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

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

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

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:

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

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

$$B^d(i) := [\, B_{t_1}^d(i) \quad B_{t_2}^d(i) \quad \cdots \quad B_{t_j}^d(i)\,] = [B_{t_j}^d(i)]_{j=1}^{J} \in \mathbb{R}^{1\times J\cdot\hat{d}} \tag{24}$$

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

$$\boldsymbol{c} = [\boldsymbol{c}^{t_j}]_{j=1}^{J} \in \mathbb{R}^{J\cdot\hat{d}\times 1} \tag{25}$$

With the definitions in Eqs. (23–25), Eq. (20) can be written as

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

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

$$\boldsymbol{Y} = X\boldsymbol{c} + \epsilon \tag{27}$$

E. Spline Model Estimation

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

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

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

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

The estimated B coefficients can be calculated as follows:

$$\hat{\boldsymbol{c}} = \boldsymbol{C}_1 X^T \boldsymbol{Y} \tag{30}$$

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

$$c^{t_i}_{(\kappa_0,\ldots,\kappa_{n-1},m)} = \sum_{|\gamma|=m} c^{t_j}_{(\kappa_0,\ldots,\kappa_{n-1},0)+\gamma} B^m_\gamma(\boldsymbol{v}^*), \qquad 0 \le m \le r \tag{31}$$

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

$$H\boldsymbol{c} = 0 \tag{32}$$

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 Fig. 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 $\boldsymbol{\tau}$, 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 nonaffine 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 nonaffinity 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.

A. Nonlinear Dynamic Inversion

Consider the aircraft state equations in the input affine form:

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

$$\boldsymbol{\tau} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u}) \tag{34}$$

with $\boldsymbol{x} \in \mathbb{R}^n$ the state vector, $\boldsymbol{u} \in \mathbb{R}^m$ 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 $\boldsymbol{\nu}$:

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

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

$$\dot{\boldsymbol{x}} = \boldsymbol{\nu} \tag{36}$$

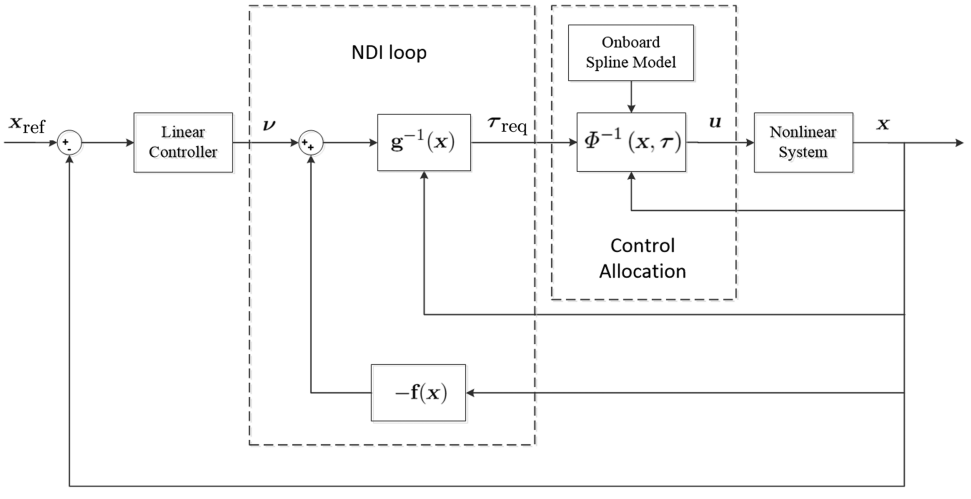


Fig. 1 Combined control structure: NDI inner loop and linear control outer loop combined with control allocation.

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

$$\dot{\boldsymbol{x}} = \boldsymbol{\nu} + \boldsymbol{\Delta} \tag{37}$$

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

$$\dot{\boldsymbol{\omega}} = I^{-1}\boldsymbol{M} - I^{-1}\boldsymbol{\omega} \times I\boldsymbol{\omega} \tag{38}$$

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

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

The required control moment has to be translated into control surface deflections based on the onboard model for $\boldsymbol{M}$. 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 $\boldsymbol{M}$ of the F-16 is identified using simplex splines (see Sec. VI). Simplex splines provide a significant increase in modeling accuracy compared with 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:

$$\boldsymbol{\tau} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u})\colon \mathbb{R}^m \to \mathbb{R}^l \tag{40}$$

The control allocation problem considers the inversion of this mapping:

$$\boldsymbol{u} = \boldsymbol{\Phi}^{-1}(\boldsymbol{x}, \boldsymbol{\tau})\colon \mathbb{R}^l \to \mathbb{R}^m \tag{41}$$

The control allocation problem can be stated as follows: Given a virtual command $\boldsymbol{\tau}$, determine $\boldsymbol{u}$ satisfying the actuator position and rate constraints, such that $\boldsymbol{\tau} = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u})$. The input will be determined based on the onboard spline model for $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u})$. 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:

$$\min_{\underline{\boldsymbol{u}} \le \boldsymbol{u} \le \bar{\boldsymbol{u}}} \mathcal{J} = \|\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{\tau}_{\text{req}}\| \tag{42}$$

2) Control minimization problem:

$$\min_{\underline{\boldsymbol{u}} \le \boldsymbol{u} \le \bar{\boldsymbol{u}}} \mathcal{J} = \|\boldsymbol{u}\| \quad \text{subject to } \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{\tau}_{\text{req}} \tag{43}$$

3) Mixed optimization problem:

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

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

Most existing methods derived for overactuated systems ($l < m$) for solving Eqs. (42–44) consider linear effector models of the form

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

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 pseudoinverse 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 pseudoinverse solution:

$$\boldsymbol{u} = G^{+}\boldsymbol{\tau} \tag{46}$$

with the pseudoinverse calculated as

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

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 pseudoinverse 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 Fig. 2. The controlled variables are the roll angle ϕ, angle of attack α, and sideslip angle β. 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:

$$H_{\text{pf}} = \frac{1}{\sigma s + 1}$$

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.

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:

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

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

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

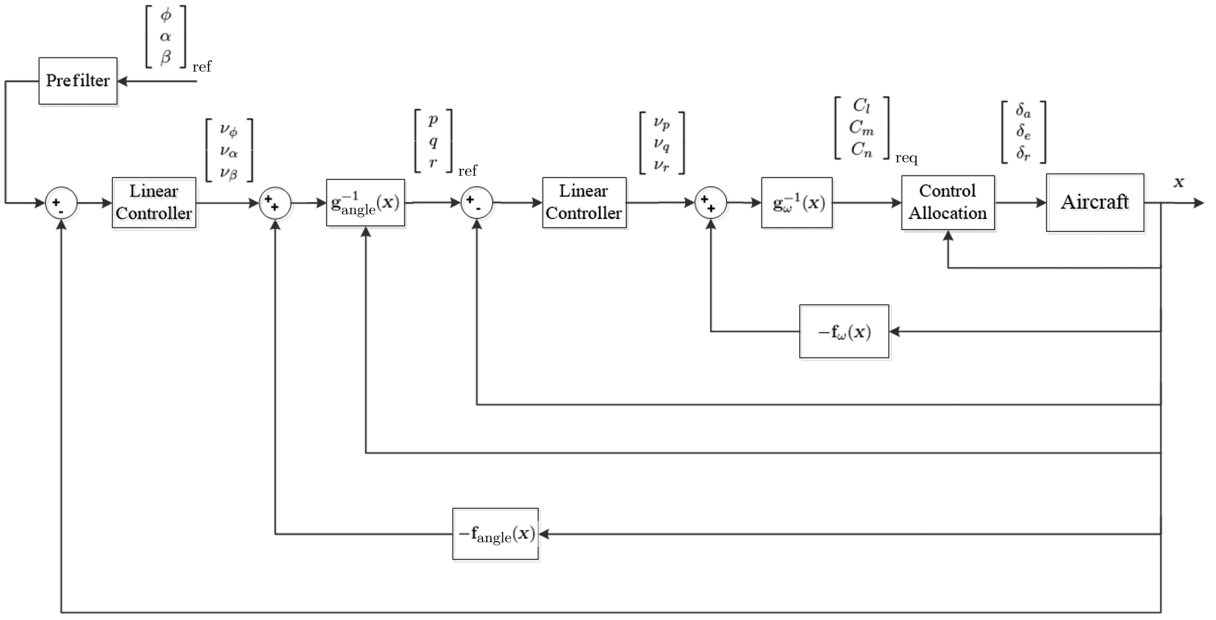


Fig. 2 Control setup. An inner rate NDI loop combined with an aerodynamic angle outer NDI loop.

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 ϕ, angle of attack α, and sideslip angle β. The dynamics are expressed in terms of required body angular rates. For the roll angle, this results in

$$\dot{\phi} = [1 \quad \sin\phi\tan\theta \quad \cos\phi\tan\theta]\begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{50}$$

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

For the angle of attack,

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

and, for the sideslip angle, this gives

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

Combining Eqs. (50), (52), and (53) and rewriting into the form of Eq. (35) gives

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

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 are 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 with a polynomial-based NDI controller.

The required variables to be estimated are the moment coefficients C_l, C_m, and C_n. In Sec. IV.B, the model is identified with simplex spline structures and polynomial structures using a training dataset consisting of 60,000 scattered points, which is generated with the wind-tunnel model. In Sec. VI.C, the polynomial and spline model are compared and validated using a validation dataset consisting of 10,000 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:

Table 2 Prefilter time constants and controller gains

Time constants	Controller gains	
$\sigma_{\phi} = 0.2$	$k_{\phi} = 2$	$k_p = 10$
$\sigma_{\alpha} = 0.4$	$k_{\alpha} = 2$	$k_q = 10$
$\sigma_{\beta} = 0.2$	$k_{\beta} = 2$	$k_r = 10$

$$
\begin{aligned}
-10\ \text{deg} \le \alpha \le 45\ \text{deg} \qquad & 50\ \text{m/s} \le V \le 300\ \text{m/s} \\
-30\ \text{deg} \le \beta \le 30\ \text{deg} \qquad & -21.5\ \text{deg} \le \delta_a \le 21.5\ \text{deg} \\
-90\ \text{deg/s} \le p \le 90\ \text{deg/s} \qquad & -25.0\ \text{deg} \le \delta_e \le 25.0\ \text{deg} \\
-90\ \text{deg/s} \le q \le 90\ \text{deg/s} \qquad & -30.0\ \text{deg} \le \delta_r \le 30.0\ \text{deg} \\
-90\ \text{deg/s} \le r \le 90\ \text{deg/s} \qquad & 0\ \text{deg} \le \delta_{\text{lef}} \le 25.0\ \text{deg}
\end{aligned} \tag{55}
$$

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:

$$
\begin{aligned}
C_m(\alpha,\beta,\tilde{q},\delta_e,\delta_{\text{lef}}) &= C_m(\alpha,\beta,\delta_e) + C_{m_{\delta_{\text{lef}}}}(\alpha,\beta)\delta_{\text{lef}} \\
&\quad + C_{m_q}(\alpha)\tilde{q} + C_{m_{q\delta_{\text{lef}}}}(\alpha)\delta_{\text{lef}}\tilde{q}
\end{aligned} \tag{56}
$$

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

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

where

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

Each modeling function in Eqs. (56–58) 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

Table 3 Aerodynamic model structures for estimating the F-16 wind-tunnel database

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

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 $\boldsymbol{Y}$, 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

$$\boldsymbol{C} = (X_p^T X_p)^{-1} X_p^T \boldsymbol{Y} \tag{59}$$

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

$$\begin{aligned} s(\alpha, \beta, \tilde{q}, \delta_e, \delta_{\text{lef}}) &= s_1(\alpha, \beta, \delta_e) + s_2(\alpha, \beta)\delta_{\text{lef}} \\ &\quad + s_3(\alpha)\tilde{q} + s_4(\alpha)\delta_{\text{lef}}\tilde{q} \end{aligned} \tag{60}$$

with

$$s_1 \in S_{d1}^{r_1}(\mathcal{T}), \qquad s_2 \in S_{d2}^{r_2}(\mathcal{T}), \qquad s_3 \in S_{d3}^{r_3}(\mathcal{T}), \qquad s_4 \in S_{d4}^{r_4}(\mathcal{T}) \tag{61}$$

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:

$$y(i) = [\, B_1^{d_1}(i)D_1(i) \quad B_2^{d_2}(i)D_2(i)\delta_{\text{lef}}(i) \quad B_3^{d_3}(i)D_3(i)\tilde{q}(i) \quad B_4^{d_4}(i)D_4(i)\delta_{\text{lef}}(i)\tilde{q}(i)\,][\, \boldsymbol{c}_1^T \quad \boldsymbol{c}_2^T \quad \boldsymbol{c}_3^T \quad \boldsymbol{c}_4^T\,]^T = X(i)\boldsymbol{c} \tag{62}$$

which, for all observations, results in the standard regression form [Eq. (27)]. The B-coefficient vectors $\boldsymbol{c}_1 - \boldsymbol{c}_4$ 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]

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

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

$$\begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \boldsymbol{c}_3 \\ \boldsymbol{c}_4 \end{bmatrix} = C_1 X^T \boldsymbol{Y} \tag{64}$$

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 with the original wind-tunnel model and validated against the validation dataset. The results from the model validation are listed in Table 4. Because the true model is known from the NASA wind-tunnel data tables, a direct comparison can be made, which is shown in Fig. 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 with 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%, whereas the polynomial model has a relative error rms of 11.15%.

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 nonaffine aerodynamic spline models requires the augmentation of the NDI structure with a separate control allocation module. This augmented structure was introduced in 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 $\boldsymbol{\tau}$:

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

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:

$$\tau_i = \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u}) \approx s(\boldsymbol{x}, \boldsymbol{u}) \tag{65}$$

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

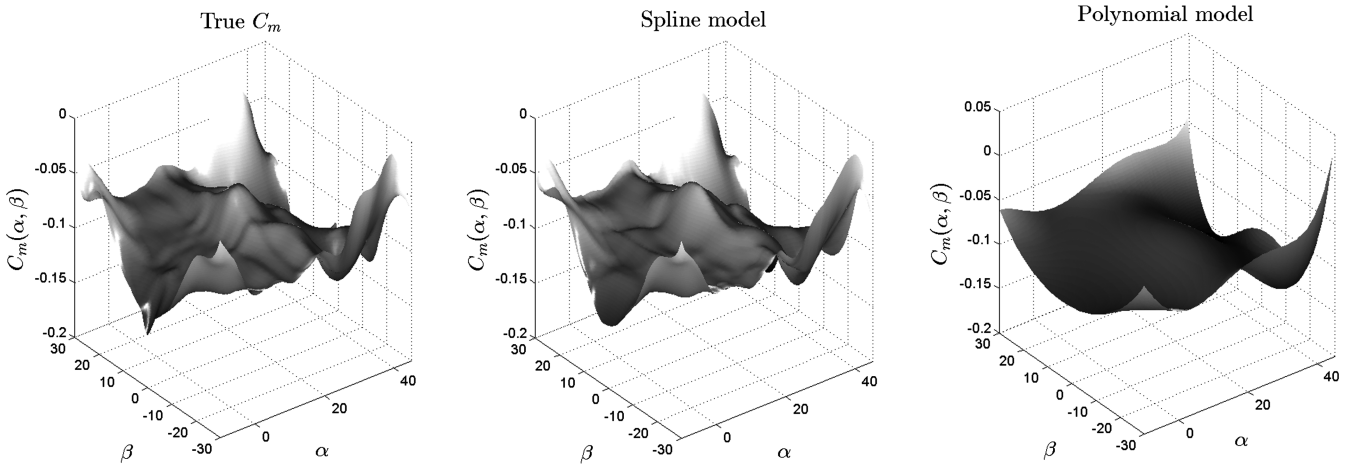
$$\begin{aligned} \tau_1 &= C_l \approx s(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{\text{lef}}), \\ \tau_2 &= C_m \approx s(\alpha, \beta, \tilde{q}, \delta_e, \delta_{\text{lef}}), \\ \tau_3 &= C_n \approx s(\alpha, \beta, \tilde{p}, \tilde{r}, \delta_a, \delta_r, \delta_{\text{lef}})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

The control allocation problem for spline-based aerodynamic models can be stated as follows: Given a virtual command $\boldsymbol{\tau}$, determine $\boldsymbol{u}$ satisfying the actuator limits, such that $\boldsymbol{\tau} = \boldsymbol{s}(\boldsymbol{x}, \boldsymbol{u})$. 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. VII.A. In Secs. VII.B–VII.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. The advantages of having an analytical expression over

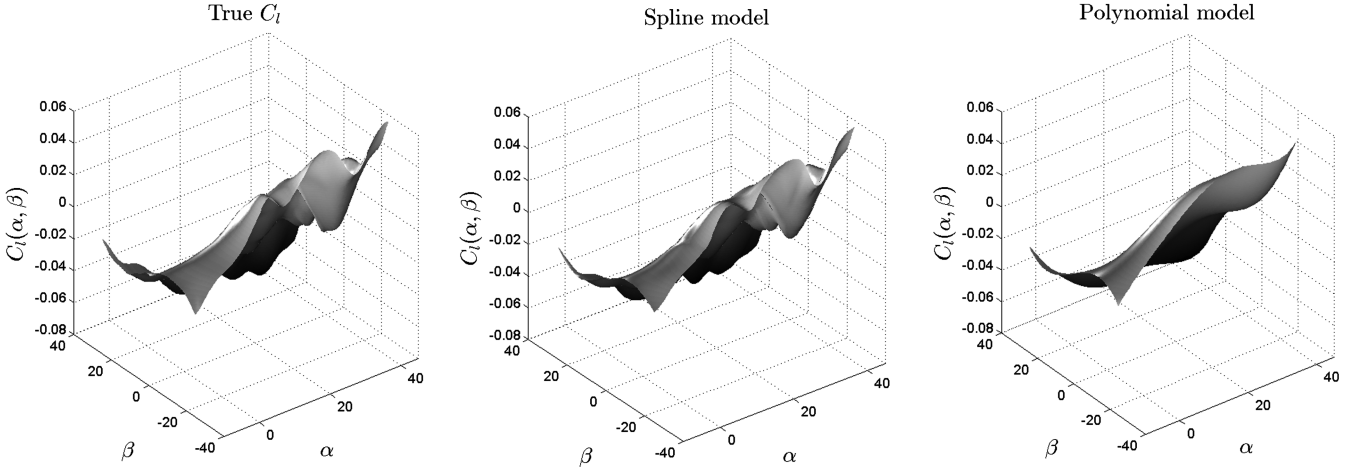
Table 4 Model validation performance parameters

	Spline model			Polynomial model		
Performance parameter	Error rms	Relative error rms,[a] %	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

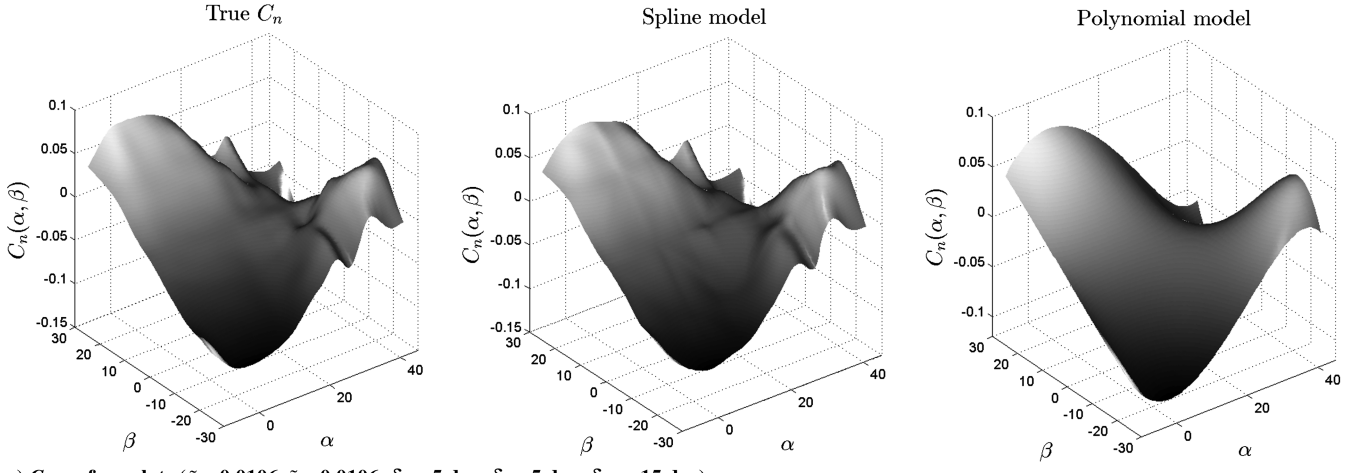
[a]The relative error rms is defined as $\text{RMS}_{\text{rel}}(\boldsymbol{\epsilon}) = [\text{RMS}(\boldsymbol{\epsilon})]/[\text{RMS}(\boldsymbol{Y}_v)]$.



a) C_m surface plots ($\tilde{q}$ = 0.04, δ_e = 5 deg, δ_{lef} = 15 deg)



b) C_l surface plots ($\tilde{q}$ = 0.0106, $\tilde{r}$ = 0.0106, δ_a = 5 deg, δ_r = 5 deg, δ_{lef} = 15 deg)



c) C_n surface plots ($\tilde{q}$ = 0.0106, $\tilde{r}$ = 0.0106, δ_a = 5 deg, δ_r = 5 deg, δ_{lef} = 15 deg)

Fig. 3 Model comparison.

a numerical approximation is that it is exact and computationally more efficient to calculate. For example, the central difference approximation of the second derivative $\partial^2 f/\partial x_i \partial x_j$ requires four evaluations of function f compared with one evaluation of the second derivative when having an analytical expression ([30] p. 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 $p_j^t(b(\boldsymbol{x}))$ with respect to the spline state $\boldsymbol{x}$. In the following sections, an expression for the barycentric coordinate $b(\boldsymbol{x}) = (b_0, b_1, \ldots, b_n)$ as an affine function of the spline state $\boldsymbol{x}$ is required.

The barycentric coordinates $(b_1, \ldots, b_n)$ given by Eq. (9) can be expressed as an affine function of $\boldsymbol{x}$ and are derived as follows:

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

with $\boldsymbol{k}_n = -\Lambda \boldsymbol{v}_0$, and the component b_0 as follows:

$$b_0 = 1 - \sum_{i=1}^{n} b_i = 1 - [\,1 \quad 1 \quad \ldots \quad 1\,] \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = -[\,1 \quad 1 \quad \ldots \quad 1\,]\Lambda \boldsymbol{x} + (1 - [\,1 \quad 1 \quad \ldots \quad 1\,]\boldsymbol{k}_n) = \Lambda_0 \boldsymbol{x} + k_0 \tag{67}$$

Combining Eqs. (66) and (67) results in

$$\begin{bmatrix} b_0 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \Lambda_0 \\ \Lambda \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} k_0 \\ \boldsymbol{k}_n \end{bmatrix} = A\boldsymbol{x} + \boldsymbol{k} \tag{68}$$

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

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

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

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

$$\frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial x_i} = \boldsymbol{a}_i^T \nabla_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x})) \tag{70}$$

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

$$\nabla_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x})) = \begin{bmatrix} \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0} & \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_1} & \cdots & \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n} \end{bmatrix}^T \tag{71}$$

Proof: This proof uses the multivariable chain rule

$$\frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial x_i} = \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0}\frac{\partial b_0(\boldsymbol{x})}{\partial x_i} + \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_1}\frac{\partial b_1(\boldsymbol{x})}{\partial x_i} + \cdots + \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0}\frac{\partial b_n(\boldsymbol{x})}{\partial x_n} \tag{72}$$

Equation (72) can be written in vector form:

$$\frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial x_i} = \begin{bmatrix} \frac{\partial b_0(\boldsymbol{x})}{\partial x_i} & \frac{\partial b_1(\boldsymbol{x})}{\partial x_i} & \cdots & \frac{\partial b_n(\boldsymbol{x})}{\partial x_i} \end{bmatrix} \times \begin{bmatrix} \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0} & \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_1} & \cdots & \frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n} \end{bmatrix}^T \tag{73}$$

$$= \boldsymbol{a}_i^T \nabla_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x})) \tag{74}$$

with $\boldsymbol{a}_i$ the ith column of A_{t_j}. □

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

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

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

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

$$\frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial x_i \partial x_j} = \boldsymbol{a}_i^T \nabla^2_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x}))\boldsymbol{a}_j \tag{76}$$

with $\nabla^2_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x}))$ the Hessian of the B-form polynomial with respect to the barycentric coordinate $\boldsymbol{b}$:

$$\nabla^2_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x})) = \begin{bmatrix} \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0^2} & \cdots & \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0 \partial b_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n \partial b_0} & \cdots & \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n^2} \end{bmatrix} \tag{77}$$

Proof: It is shown that $\frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial x_i \partial x_j} = \boldsymbol{a}_i^T \nabla^2_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x}))\boldsymbol{a}_j$:

$$\frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_j}\frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial x_i} = \frac{\partial}{\partial x_j}\boldsymbol{a}_i^T \nabla_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x}))$$

(by Lemma 1)

$$= \boldsymbol{a}_i^T \begin{bmatrix} \frac{\partial}{\partial x_j}\frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0} \\ \vdots \\ \frac{\partial}{\partial x_j}\frac{\partial B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n} \end{bmatrix} = \boldsymbol{a}_i^T \begin{bmatrix} \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0^2}\frac{\partial b_0}{\partial x_j} + \cdots + \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0 \partial b_n}\frac{\partial b_n}{\partial x_j} \\ \vdots \\ \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n \partial b_0}\frac{\partial b_0}{\partial x_j} + \cdots + \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n^2}\frac{\partial b_n}{\partial x_j} \end{bmatrix}$$

(Chain rule)

$$= \boldsymbol{a}_i^T \begin{bmatrix} \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0^2} & \cdots & \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_0 \partial b_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n \partial b_0} & \cdots & \frac{\partial^2 B^d_\kappa(b(\boldsymbol{x}))}{\partial b_n^2} \end{bmatrix} \begin{bmatrix} \frac{\partial b_0(\boldsymbol{x})}{\partial x_j} \\ \vdots \\ \frac{\partial b_n(\boldsymbol{x})}{\partial x_j} \end{bmatrix}$$

$$= \boldsymbol{a}_i^T \nabla^2_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x}))\boldsymbol{a}_j$$

with $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ the ith and jth column of A_{t_j}. □

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

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

$$b(\boldsymbol{x}) = [\,\boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \cdots \quad \boldsymbol{a}_n\,]_{t_j} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \boldsymbol{k} = A_{t_j}\boldsymbol{x} + \boldsymbol{k} \tag{78}$$

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

$$\nabla_x p^{t_j}(b(\boldsymbol{x})) = A_{t_j}^T \nabla_{\boldsymbol{b}} B^d_{t_j}(b(\boldsymbol{x}))\boldsymbol{c}^{t_j} \tag{79}$$

with $\nabla_{\boldsymbol{b}} B^d_{t_j}(b(\boldsymbol{x}))$ the vector of B-form polynomial gradients given by

$$\nabla_{\boldsymbol{b}} B^d_{t_j}(b(x)) =: [\nabla_{\boldsymbol{b}} B^d_{d,0,\cdots,0}(b(\boldsymbol{x})) \;\; \nabla_{\boldsymbol{b}} B^d_{d-1,1,\cdots,0}(b(\boldsymbol{x})) \;\; \cdots \;\; \nabla_{\boldsymbol{b}} B^d_{0,\cdots,0,d}(b(\boldsymbol{x}))] = [\nabla_{\boldsymbol{b}} B^d_\kappa(b(\boldsymbol{x}))]_{|\kappa=d|} \in \mathbb{R}^{n+1 \times \hat{d}} \tag{80}$$

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

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

Proof: This proof starts by showing that $\frac{\partial p^{t_j}(b(\boldsymbol{x}))}{\partial x_i} = a_i^T \nabla_{\boldsymbol{b}} B^d \boldsymbol{c}^{t_j}$:

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

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

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

Combining the partial derivatives for all x_i gives

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

which proves the theorem. □

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

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

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

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

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

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

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

Combining the second partial derivatives for all x_i, x_j gives

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

$$= \begin{bmatrix} \boldsymbol{a}_1^T \\ \vdots \\ \boldsymbol{a}_n^T \end{bmatrix} \Gamma_{t_j} [\,\boldsymbol{a}_1 \quad \cdots \quad \boldsymbol{a}_n\,] = A_{t_j}^T \Gamma_{t_j} A_{t_j} \tag{87}$$

which proves the theorem. □

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):

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

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

$$\tau_i = p^{t_j}(\boldsymbol{x}), \qquad \boldsymbol{x}_0 \in t_j \tag{88}$$

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

$$b(\boldsymbol{x}) = A_{t_j}\boldsymbol{x} + \boldsymbol{k} \tag{89}$$

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

$$\boldsymbol{x} = [\,x_1 \quad x_2 \quad \ldots \quad x_l \quad u_1 \quad u_2 \quad \ldots \quad u_m\,]^T = [\,\boldsymbol{x}_a^T \quad \boldsymbol{u}^T\,]^T \tag{90}$$

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

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

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

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

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

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

which can be written in vector form:

$$\boldsymbol{\tau} = p(\boldsymbol{u}_0) + G\Delta\boldsymbol{u} \tag{94}$$

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 pseudoinverse solution [26,27] or constrained linear programming techniques [31]. The approach elaborated here is based on the pseudoinverse solution. Using the linearized model in Eq. (94), the error between the model and required output can be written as

$$\boldsymbol{e} = p(\boldsymbol{u}_0) + G\Delta\boldsymbol{u} - \boldsymbol{\tau}_{\text{req}} \quad = G\Delta\boldsymbol{u} - \tilde{\boldsymbol{\tau}}_{\text{req}} \tag{95}$$

with

$$\tilde{\boldsymbol{\tau}}_{\text{req}} = \boldsymbol{\tau}_{\text{req}} - p(\boldsymbol{u}_0) \tag{96}$$

The control minimization problem [Eq. (43)] can now be formulated as

$$\min_{\Delta\underline{\boldsymbol{u}} \le \Delta\boldsymbol{u} \le \Delta\bar{\boldsymbol{u}}} \mathcal{J} = \|\Delta\boldsymbol{u}\| \quad \text{subject to } G\Delta\boldsymbol{u} = \tilde{\boldsymbol{\tau}}_{\text{req}} \tag{97}$$

By dropping the actuator constraints, the incremental control input can be calculated using the pseudoinverse solution in Eq. (46):

$$\Delta\boldsymbol{u} = G^{+}\tilde{\boldsymbol{\tau}}_{\text{req}} \tag{98}$$

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

$$\boldsymbol{u}(t+1) = \boldsymbol{u}(t) + \Delta\boldsymbol{u} \tag{99}$$

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 new 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 preceding section, the spline model was linearized around current input $\boldsymbol{u}(t_0)$, for which the pseudoinverse solution is applied. Solutions with better accuracy can be obtained by successively calculating the pseudoinverse solution for several initial conditions in the feasible set for $\boldsymbol{u}$ 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 Ω for $\boldsymbol{u}$:

$$\Omega = \{\boldsymbol{u} \in \mathbb{R}^m | \underline{\boldsymbol{u}} \le \boldsymbol{u} \le \bar{\boldsymbol{u}}\} \subset \mathbb{R}^m \tag{100}$$

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

$$\mathcal{V}_1 := (\boldsymbol{u}_{0_1}, \boldsymbol{u}_{0_2}, \ldots, \boldsymbol{u}_{0_k}) \in \Omega \tag{101}$$

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

$$\mathcal{V}_2 = (\boldsymbol{u}_{0_1} + \Delta\boldsymbol{u}_1, \boldsymbol{u}_{0_2} + \Delta\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{0_k} + \Delta\boldsymbol{u}_k) = (\boldsymbol{u}_1^*, \boldsymbol{u}_2^*, \ldots, \boldsymbol{u}_k^*) \tag{102}$$

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

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

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 Ω 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:

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

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

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

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:

$$\min_{\underline{\boldsymbol{u}} \le \boldsymbol{u} \le \bar{\boldsymbol{u}}} = \|\boldsymbol{s}(\boldsymbol{u}) - \boldsymbol{\tau}_{\text{req}}\|_2^2 = \sum_{i=1}^{N} (s_i(\boldsymbol{u}) - \tau_{\text{req}_i})^2 = \sum_{i=1}^{N} e_i(\boldsymbol{u})^2 \tag{106}$$

This is a constrained nonlinear optimization problem that 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 preceding 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 preceding 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):

$$s(\boldsymbol{x}) = \delta_1 p^{t_1}(b(\boldsymbol{x})) + \delta_2 p^{t_2}(b(\boldsymbol{x})) + \cdots + \delta_j p^{t_j}(b(\boldsymbol{x})) = \sum_{j=1}^{J} \delta_j p^{t_j}(\boldsymbol{x}) \tag{107}$$

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

$$\boldsymbol{x} = [\,x_1 \quad x_2 \quad \ldots \quad x_l \quad u_1 \quad u_2 \quad \ldots \quad u_m\,]^T = [\,\boldsymbol{x}_a^T \quad \boldsymbol{u}^T\,]^T$$

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

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

with $A_{\boldsymbol{u},t_j} \in \mathbb{R}^{(n+1)\times m}$. Using this parameterization, the spline function can be expressed as a function only dependent on the control input $\boldsymbol{u}$:

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

By Theorem 1, the gradient of the spline function with respect to the control input is given by

$$\nabla_{\boldsymbol{u}} s_i(\boldsymbol{u}) = \sum_{j=1}^{J} \delta_j A^T_{\boldsymbol{u},t_j} \nabla_{\boldsymbol{b}} B^d_{t_j}(\boldsymbol{u}) \boldsymbol{c}^{t_j} \tag{109}$$

and by Theorem 2, the Hessian is given by

$$\nabla^2_{\boldsymbol{u}} s_i(\boldsymbol{u}) = \sum_{j=1}^{J} \delta_j A^T_{\boldsymbol{u},t_j} \Gamma_{t_j} A_{\boldsymbol{u},t_j} \tag{110}$$

with Γ_{t_j} as given in Eq. (85). Let the Jocabian of the complete spline model $\boldsymbol{s}(\boldsymbol{u})$ be defined as

$$\nabla \boldsymbol{s}(\boldsymbol{u}) = \begin{bmatrix} \frac{\partial s_1(\boldsymbol{u})}{\partial u_1} & \cdots & \frac{\partial s_1(\boldsymbol{u})}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_N(\boldsymbol{u})}{\partial u_1} & \cdots & \frac{\partial s_N(\boldsymbol{u})}{\partial u_m} \end{bmatrix} = \begin{bmatrix} \nabla^T_{\boldsymbol{u}} s_1(\boldsymbol{u}) \\ \vdots \\ \nabla^T_{\boldsymbol{u}} s_N(\boldsymbol{u}) \end{bmatrix} \tag{111}$$

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

$$\nabla \mathcal{J}(\boldsymbol{u}) = 2\nabla^T_{\boldsymbol{u}} \boldsymbol{s}(\boldsymbol{u})(\boldsymbol{s}(\boldsymbol{u}) - \boldsymbol{\tau}_{\text{req}}) \tag{112}$$

$$\nabla^2 \mathcal{J}(\boldsymbol{u}) = 2\nabla^T \boldsymbol{s}(\boldsymbol{u}) \nabla \boldsymbol{s}(\boldsymbol{u}) + 2 \sum_{i=1}^{N} \nabla^2_{\boldsymbol{u}} s_i(\boldsymbol{u})(s_i(\boldsymbol{u}) - \tau_{\text{req}_i}) \tag{113}$$

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

$$\begin{aligned} \mathcal{J}(\boldsymbol{u}) \asymp & \mathcal{J}(\boldsymbol{u}_0) + \nabla^T_{\boldsymbol{u}} \mathcal{J}(\boldsymbol{u}_0)(\boldsymbol{u} - \boldsymbol{u}_0) \\ & + \frac{1}{2}(\boldsymbol{u} - \boldsymbol{u}_0)^T \nabla^2_{\boldsymbol{u}} \mathcal{J}(\boldsymbol{u})(\boldsymbol{u} - \boldsymbol{u}_0) = \bar{\mathcal{J}}(\boldsymbol{u}) \end{aligned} \tag{114}$$

A good estimate for the solution of the unconstrained optimization problem is obtained by setting $\partial \bar{\mathcal{J}} / \partial \boldsymbol{u} = 0$ and solving for $\boldsymbol{u}$. This results in

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

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

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

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

$$\nabla^2 \mathcal{J}(\boldsymbol{u}) \approx 2\nabla^T_{\boldsymbol{u}} \boldsymbol{s}(\boldsymbol{u}) \nabla \boldsymbol{s}(\boldsymbol{u}) = \bar{\nabla}^2_{\boldsymbol{u}} \mathcal{J}(\boldsymbol{u}) \tag{117}$$

Substituting Eq. (117) in Eq. (116) results in the Gauss–Newton algorithm:

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

Although an analytical expression for the Hessian of the spline function $\nabla^2_{\boldsymbol{u}} s_i(\boldsymbol{u})$ is available, using the approximation in Eq. (117) avoids its evaluation, making the solver more efficient. Furthermore, the Gauss–Newton Hessian matrix $\bar{\nabla}^2_{\boldsymbol{u}} \mathcal{J}(\boldsymbol{u})$ is always positive definite and therefore guarantees that the search direction $\boldsymbol{d}^{(\text{gn})}_k$ is a decent direction. The advantage of Gauss–Newton's algorithm is that it shows good local convergence (i.e., when the initial solution $\boldsymbol{u}_0$ is chosen close to the optimal solution $\boldsymbol{u}^*$). This is often the case for the control allocation problem. For example, suppose that the global optimal solution $\boldsymbol{u}^*(t_0)$ at time $t = t_0$ is found. Then, when using a small step size Δt, it is likely that the optimal solution $\boldsymbol{u}^*(t_0 + \Delta t)$ at time $t_0 + \Delta t$ is located in the neighborhood[¶] N of $\boldsymbol{u}(t_0)$. When this assumption is valid, the optimal solution at $t_0 + \Delta t$ can be found quickly using the Gauss–Newton algorithm with $\boldsymbol{u}^*(t_0)$ as initial feasible solution. However, the Gauss–Newton algorithm shows poor convergence when the initial solution $\boldsymbol{u}_0$ is far from $\boldsymbol{u}^*$ 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:

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

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

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

In [34], it is proven that a sufficient large η_k exists, 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 the 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 follows: Let $\boldsymbol{u}_k = \boldsymbol{u}(t_0)$, $\eta_k = \eta_0$, $\upsilon > 1$:

1) Try an update: $\boldsymbol{u}_{\text{try}} = \boldsymbol{u}_k - (\bar{\nabla}^2 \mathcal{J}(\boldsymbol{u}_k) + \eta_k I)^{-1} \nabla \mathcal{J}(\boldsymbol{u}_k)$.
2) Saturate controls: $\boldsymbol{u}_{\text{try}} = \min\{\max\{\boldsymbol{u}_{\text{try}}, \underline{\boldsymbol{u}}\}, \bar{\boldsymbol{u}}\}$.
3) Evaluate the objective: $\mathcal{J}(\boldsymbol{u}_{\text{try}}) = \|\boldsymbol{s}(\boldsymbol{u}_{\text{try}}) - \boldsymbol{\tau}_{\text{req}}\|^2_2$.
4) Update solution:
   a) If $\mathcal{J}(\boldsymbol{u}_{\text{try}}) <= \mathcal{J}(\boldsymbol{u}_k)$, accept solution: $\boldsymbol{u}_{k+1} = \boldsymbol{u}_{\text{try}}$, $\eta_{k+1} = \eta_k / \upsilon$.
   b) If $\mathcal{J}(\boldsymbol{u}_{\text{try}}) > \mathcal{J}(\boldsymbol{u}_k)$, retract the update: $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k$, $\eta_{k+1} = \eta_k \upsilon$.

Choosing a small initial value for η_0 (e.g., 0.005) leads to fast convergence when the initial solution $\boldsymbol{u}_0$ is close to the optimal solution $\boldsymbol{u}^*$. This is a reasonable assumption, as described earlier. The choice of υ is arbitrary, but a value of 10 has been found to be a good choice.

VIII. Evaluation of the Spline-Based NDI Controller

In this section, the spline-based NDI controller is evaluated. In Sec. VIII.A, the control allocation strategies are evaluated, and in Sec. VIII.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 $C_{l_{\text{req}}}$, $C_{m_{\text{req}}}$, and $C_{l_{\text{req}}}$ is based on the spline models

[¶]The neighborhood of point $\boldsymbol{u}(t_0)$ could be defined as an open ball with center $\boldsymbol{u}(t_0)$ and a small radius $\boldsymbol{\epsilon}$.

identified in Sec. VI. Consider the spline model structures for the moment coefficients given by Eqs. (56–58):

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

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

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

The leading-edge flap δ_{lef} is scheduled as a function of the angle of attack to optimize performance. At each sample instant, the required moment coefficients $C_{l_{\text{req}}}$, $C_{m_{\text{req}}}$, and $C_{n_{\text{req}}}$ have to be translated into an actuator setting δ_a, δ_e, and δ_r, such that

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

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

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

with

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

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

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

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 Fig. 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 Figs. 4–6. The plots contain three simulations starting at different trim conditions for the angles of attack $\alpha = 5, 15,$ and 25 deg. In addition, Fig. 7 contains a subplot with the number of iterations performed by the nonlinear control allocation 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 δ_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 $\alpha = 15$ deg. 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:

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

The rms values and maximum error for the three strategies are listed in Table 5, and the MATLAB execution 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 Fig. 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 because 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 Fig. 7b. In

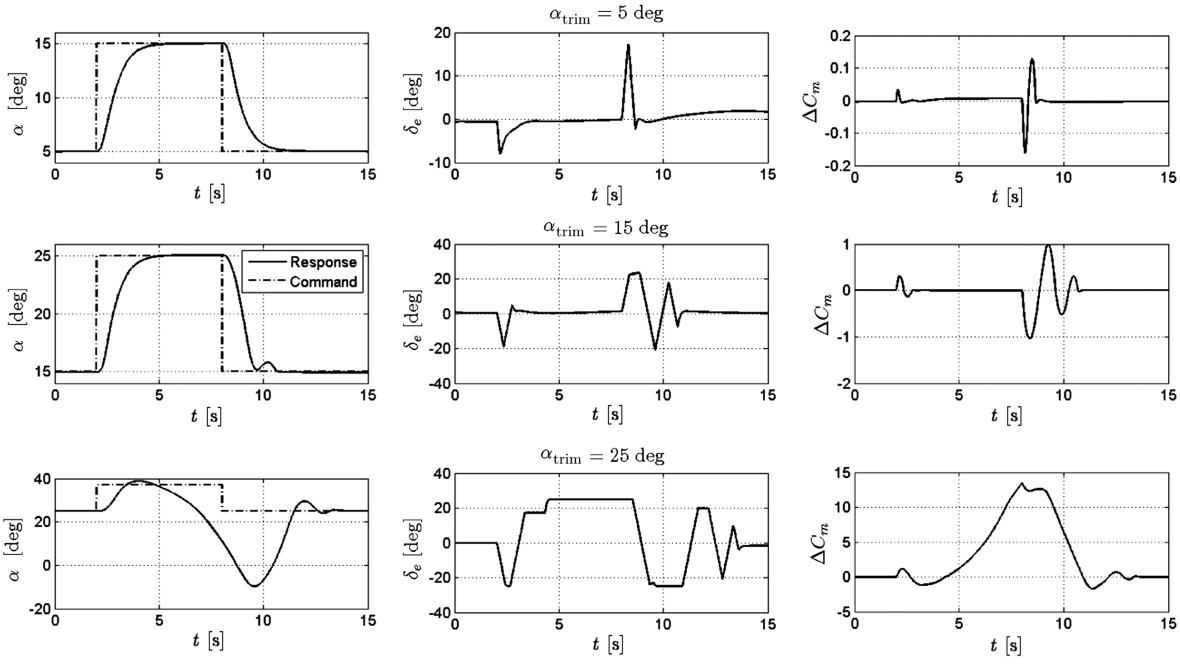


Fig. 4 SNDI with linear control allocation.

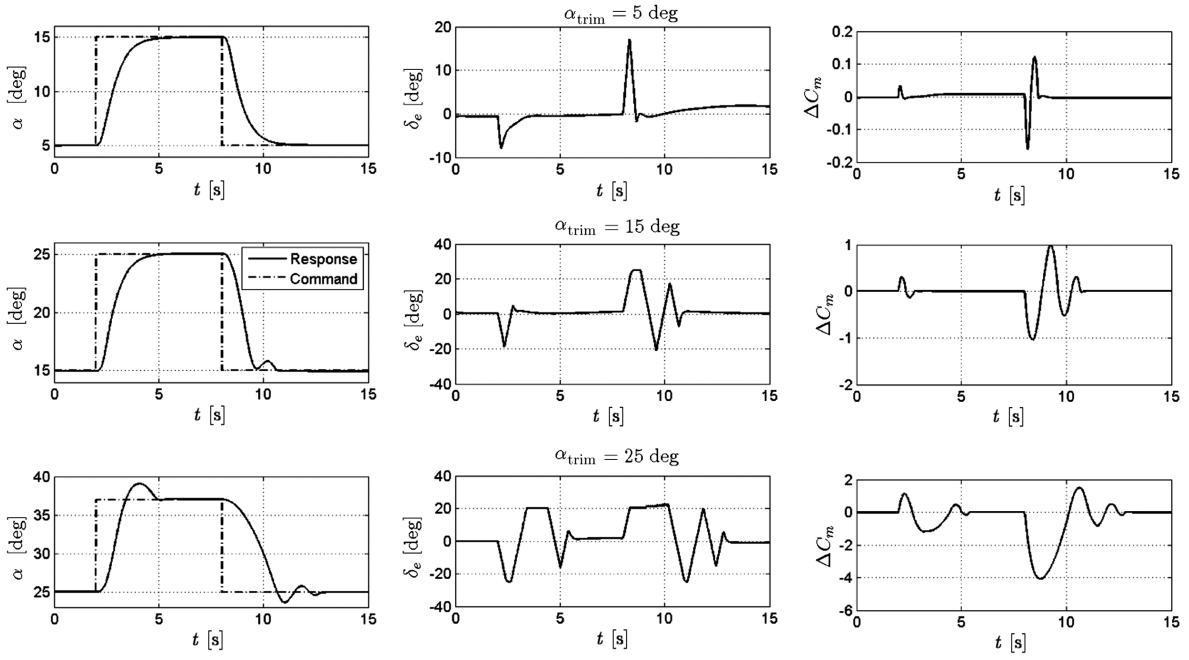


Fig. 5 SNDI with successive linear control allocation.

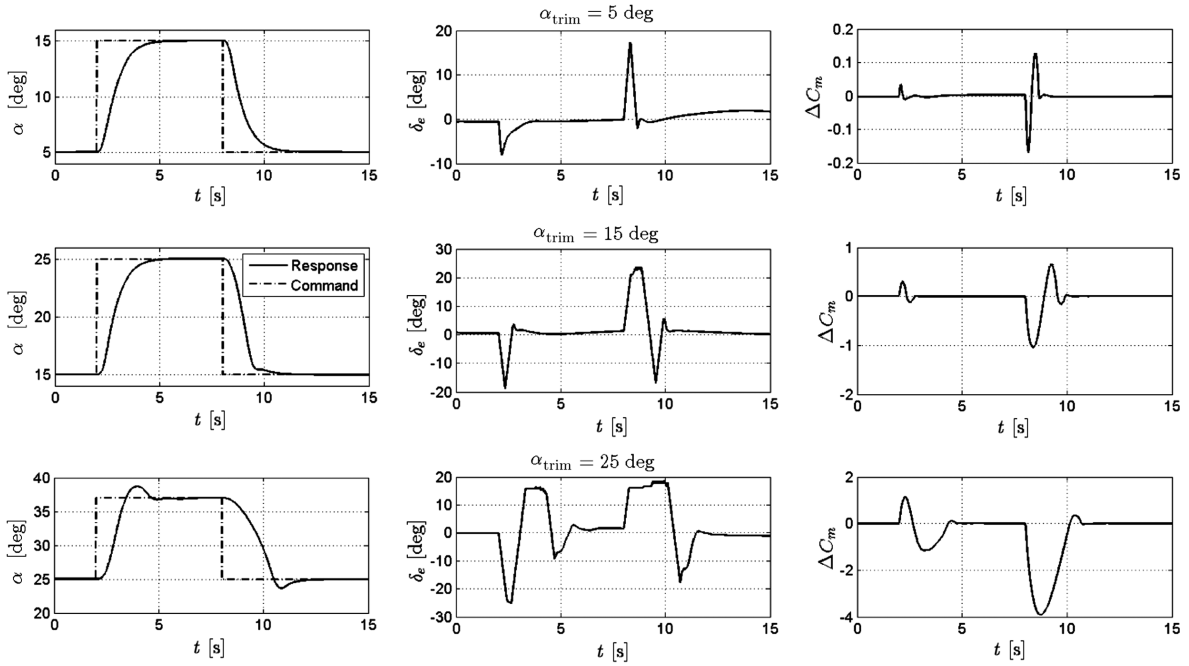


Fig. 6 SNDI with nonlinear control allocation.

turn, this results in high maximum computation loads, as can be seen from Table 6, whereas 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	$\alpha_0 = 5$ deg		$\alpha_0 = 15$ deg		$\alpha_0 = 25$ deg	
Parameter	RMSΔC_m[a]	Max$\lvert\Delta C_m\rvert$	RMSΔC_m	Max$\lvert\Delta C_m\rvert$	RMSΔC_m	Max$\lvert\Delta C_m\rvert$
Linear	0.0222	0.1667	0.2453	1.0451	5.4956	13.4828
Successive 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

[a]The inversion error is defined as $\Delta C_m(t) = C_{m_{\text{req}}}(t) - C_m(t)$.

Table 6 MATLAB execution times

Condition	$\alpha_0 = 5$ deg		$\alpha_0 = 15$ deg		$\alpha_0 = 25$ deg	
	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
Successive 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

B. Performance Assessment

In this section, the spline-based NDI controller is evaluated. To evaluate the controller properly, its performance is compared with a polynomial-based NDI controller using the models identified in Sec. 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:

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

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:

1) Maneuver 1: Roll command ($\phi_{\text{com}} = 40$ deg) and angle-of-attack command ($\alpha_{\text{com}} = 15$ deg) performed simultaneously with constant sideslip command $\beta_{\text{com}} = 5$ deg (see Fig. 8 and Table 7).

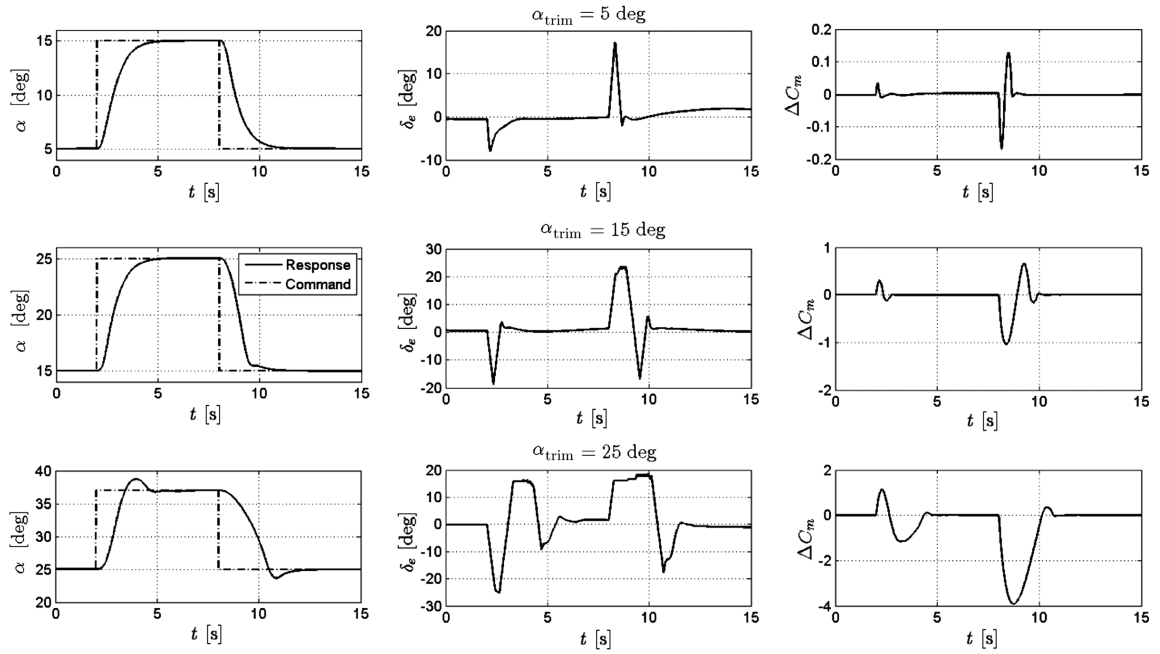
2) Maneuver 2: Roll command ($\phi_{\text{com}} = 15$ deg) and high-angle-of-attack command ($\alpha_{\text{com}} = 40$ deg) with zero sideslip. The outer-loop controller gains are decreased to $k_\phi = 1$, $k_\alpha = 1.5$, and $k_\beta = 1$ (Fig. 9 and Table 8).

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:

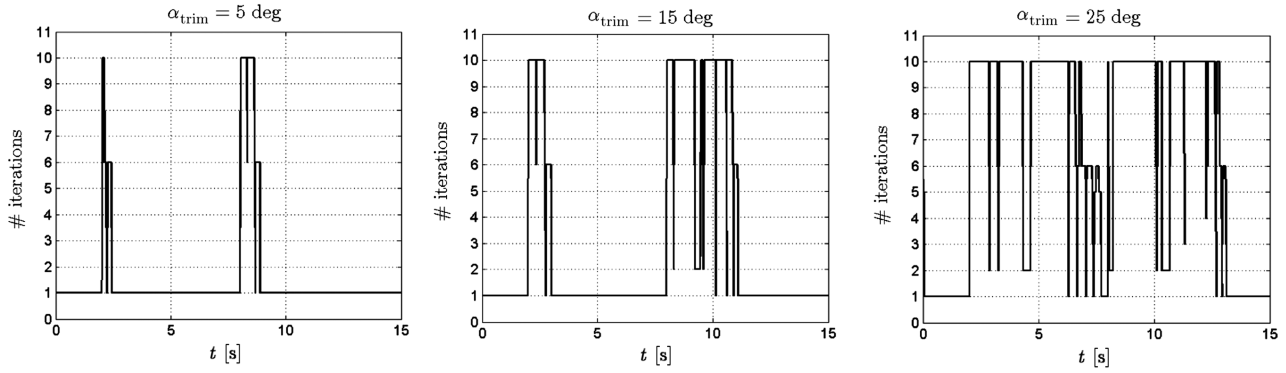
$$\xi C(t) = C(t) - \hat{C}(t) \tag{132}$$

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 Fig. 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 with the polynomial model, resulting in lower model errors and, in turn,

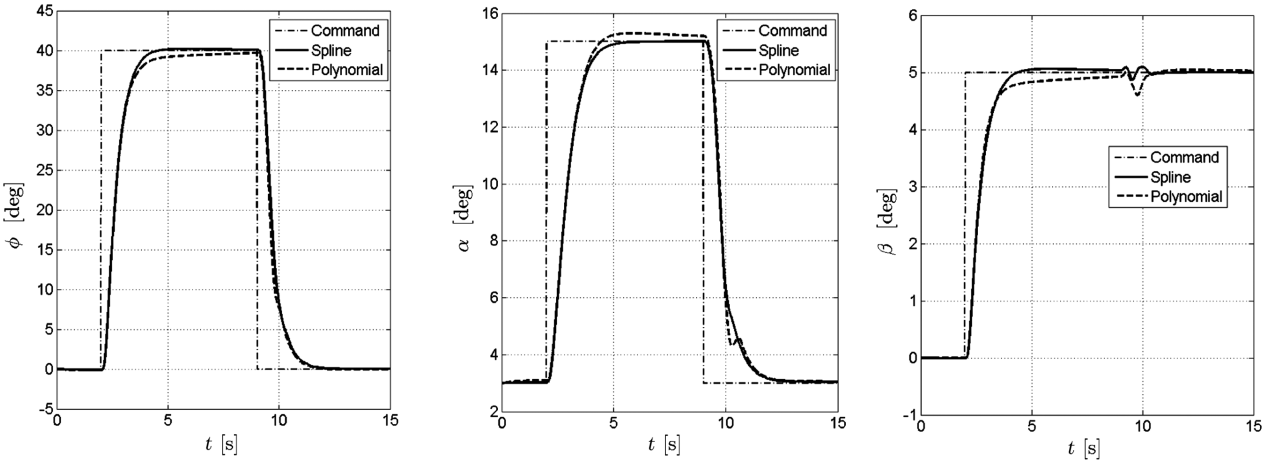


a) Angle-of-attack responses

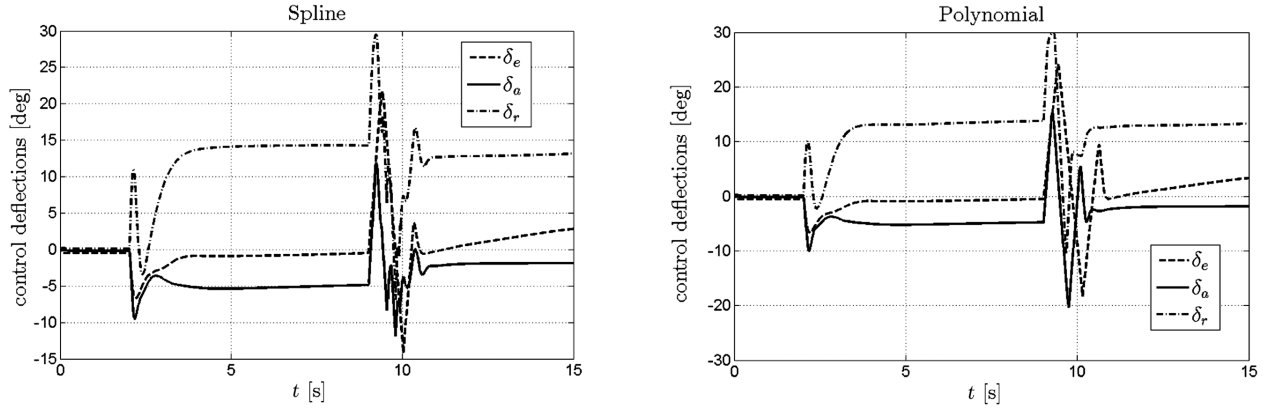


b) Number of iterations performed by the nonlinear control allocation algorithm

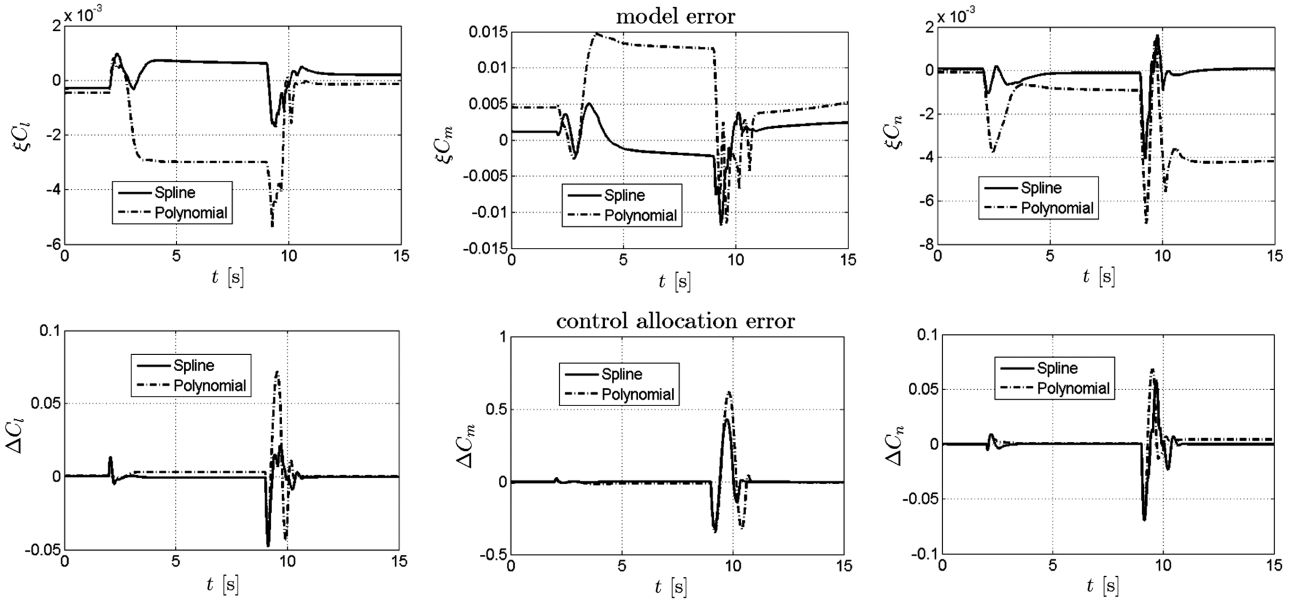
Fig. 7 SNDI with nonlinear control allocation.



a) Command variables



b) Control deflections

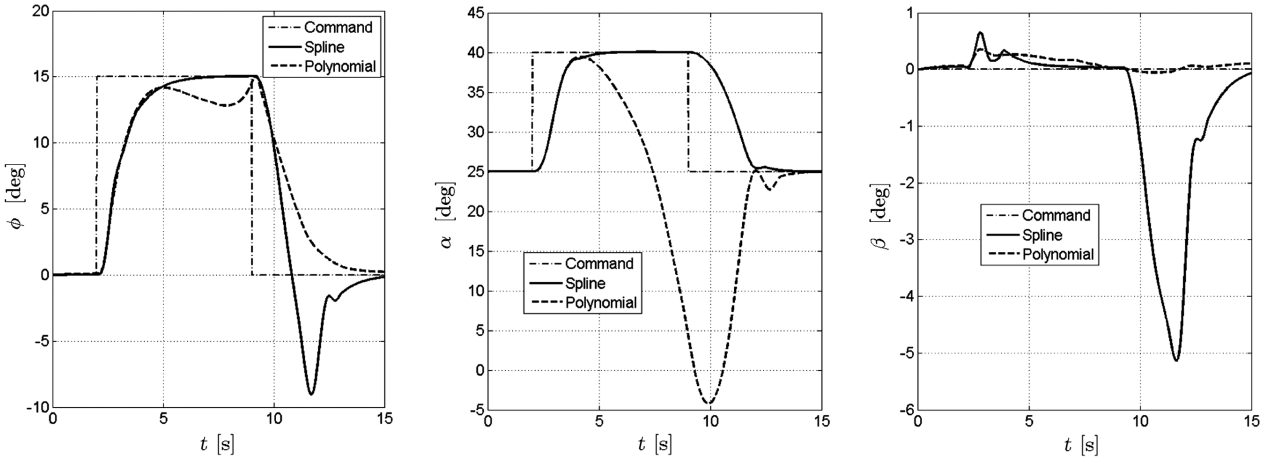


c) Model errors and control allocation errors

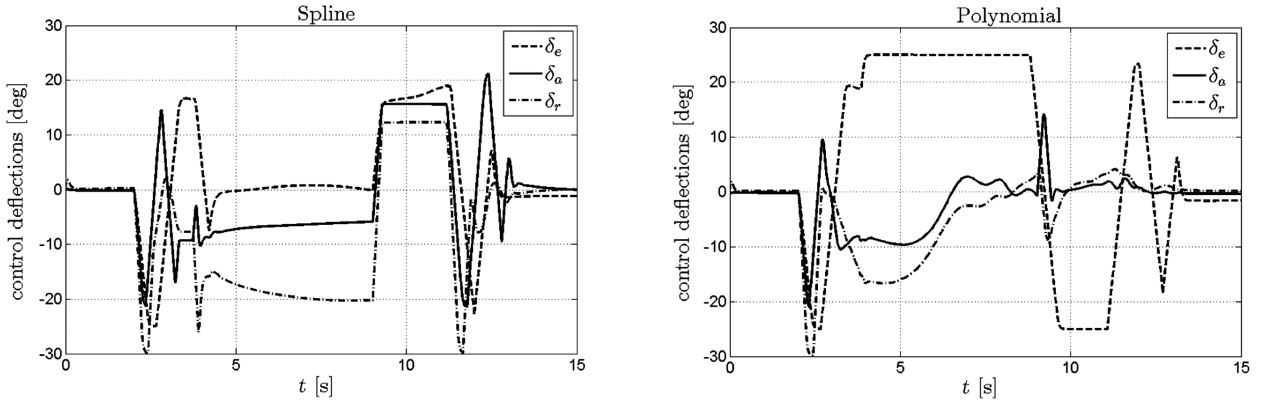
Fig. 8 Results maneuver 1: Roll command ($\phi_{\text{com}} = 40$ deg) and angle-of-attack ($\alpha_{\text{com}} = 15$ deg) command performed simultaneously with constant sideslip command $\beta_{\text{com}} = 5$ deg.

Table 7 Performance parameters maneuver 1

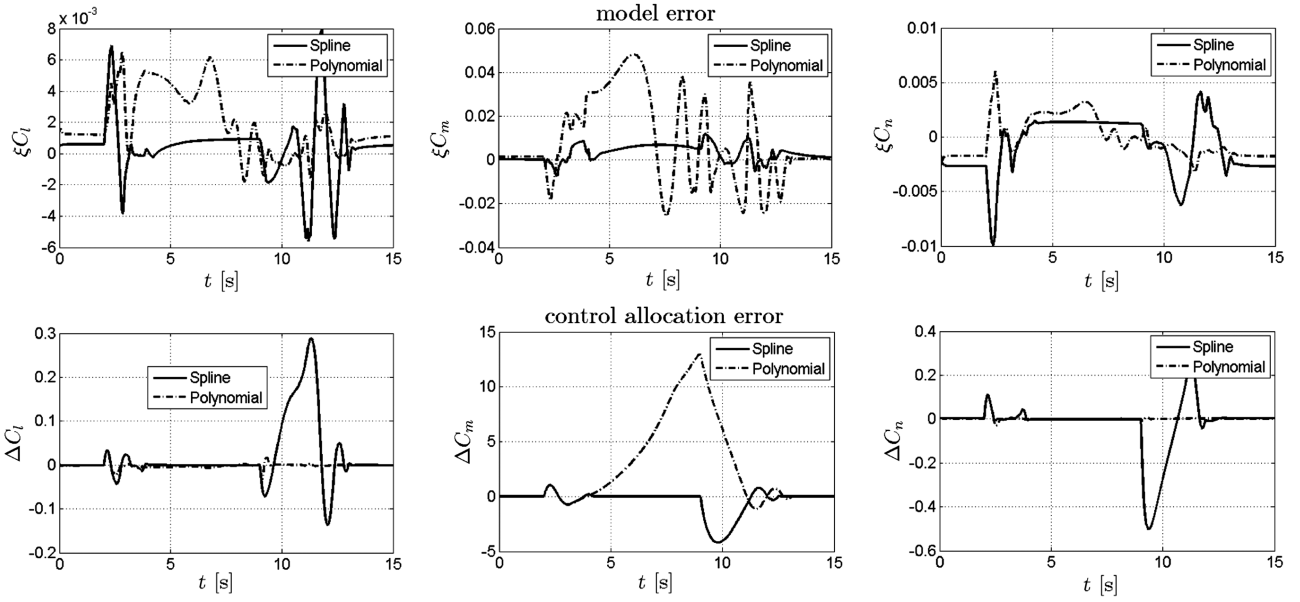
Performance parameter	RMSξC_l	RMSξC_m	RMSξC_n	RMSΔC_l	RMSΔC_m	RMSΔC_n
Spline	0.0005	0.0025	0.005	0.0055	0.0748	0.0102
Polynomial	0.0021	0.0088	0.0028	0.0115	0.1135	0.0106



a) Command variables



b) Control deflections

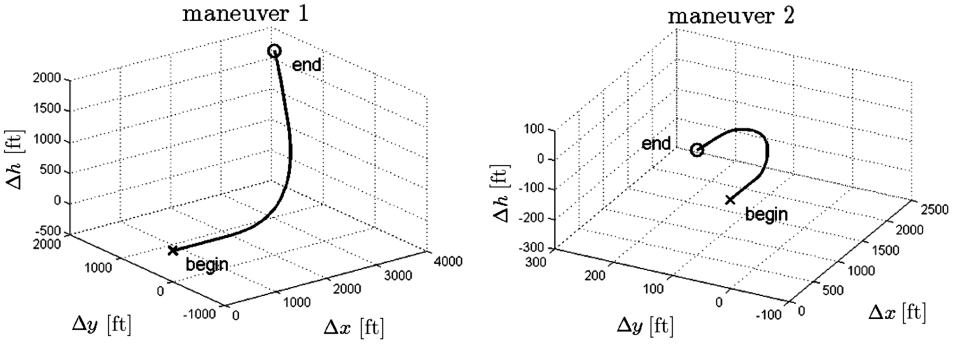


c) Model errors and control allocation errors

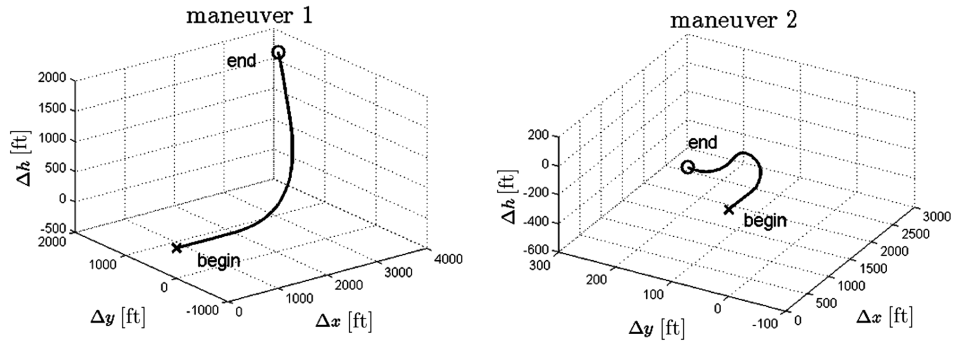
Fig. 9 Results maneuver 2: Roll command ($\phi_{com} = 15$ deg) and high-angle-of-attack command ($\alpha_{com} = 40$ deg) with zero sideslip. Outer-loop controller gains are decreased to $k_\phi = 1$, $k_\alpha = 1.5$, $k_\beta = 1$.

Table 8 Performance parameters maneuver 2

Performance parameter	RMSξC_l	RMSξC_m	RMSξC_n	RMSΔC_l	RMSΔC_m	RMSΔC_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



a) Flight trajectories spline NDI controller



b) Flight trajectories polynomial NDI controller

Fig. 10 Flight trajectories of the four maneuvers.

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 with the low-angle-of-attack region. In this region, the controls saturate quickly and, therefore, the controller 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, whereas the spline-based controller still shows adequate tracking performance. From Fig. 9, it can be observed that the polynomial-based NDI controller is better able to stabilize the roll and sideslip angle compared with 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 deg and, as a result, is operating at much lower angle of attack than the SNDI controller (i.e., 15 vs 40 deg for the SNDI controller). In fact, the significant difference between the trajectories for maneuver 2 in Fig. 10 clearly illustrates the capability of the SNDI-controlled F-16 to outmaneuver 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 with polynomial NDI. However, in the operating region with significant nonlinear aerodynamics, SNDI provides a significant increase in controller performance, resulting in improved maneuvering capabilities.

IX. Conclusions

High-performance flight control systems based on the 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 onboard 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, nonlinear, and 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 two 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 seven-step tutorial example is presented, using multivariate simplex splines to approximate a two-dimensional scattered (i.e., nongridded) dataset. This dataset consists of the measurement locations $\boldsymbol{x}$, which are chosen arbitrarily as follows (see also Fig. A1):

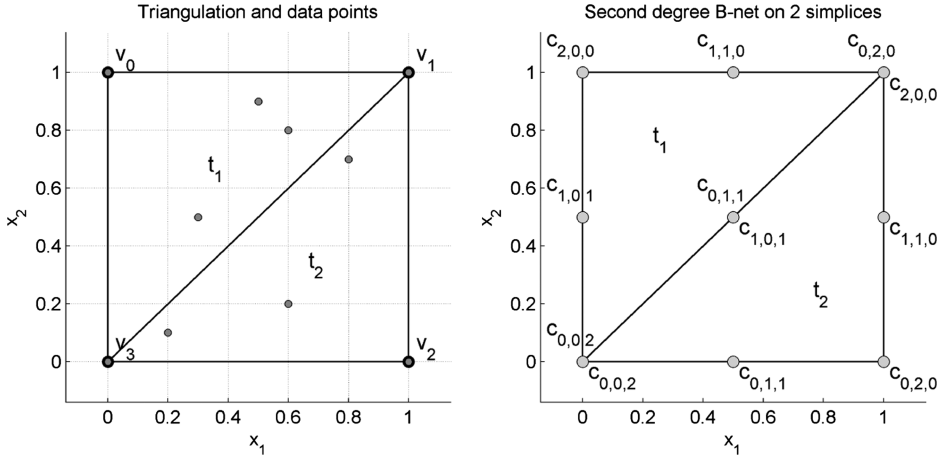


Fig. A1 Triangulation consisting of two triangles with a two-dimensional scattered dataset (left) and the *B*-coefficient net (right).

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

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:

$$y(\boldsymbol{x}) = \sin(\boldsymbol{x}_1 + \boldsymbol{x}_2) \quad = [\,0.841 \quad 0.717 \quad 0.985 \quad 0.985 \quad 0.841 \quad 0.909 \quad 0 \quad 0.296 \quad 0.717 \quad 0.997\,]^T$$

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

1) Step 1: Define the triangulation. In this case, a Delaunay triangulation is created of the convex hull of $\boldsymbol{x}$, resulting in a triangulation consisting of two triangles $t_1 = \langle \boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_3 \rangle$ and $t_2 = \langle \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3 \rangle$ (see Fig. A1).

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

$$\begin{aligned} s(\boldsymbol{x}) &= \delta_1(\boldsymbol{x}) p^{t_1}(\boldsymbol{x}) + \delta_2(\boldsymbol{x}) p^{t_2}(\boldsymbol{x}) \\ &= \delta_1(\boldsymbol{x}) B^2_{t_1}(\boldsymbol{x}) \boldsymbol{c}^{t_1} + \delta_2(\boldsymbol{x}) B^2_{t_2}(\boldsymbol{x}) \boldsymbol{c}^{t_2} \end{aligned} \tag{A1}$$

The structure of the individual B-form polynomials $B^2_{t_j}(\boldsymbol{x})\boldsymbol{c}^{t_j}$ in Eq. (A1) is found by first determining the set of multi-indices κ using Eq. (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 six (i.e., $\hat{d} = 6$) valid multi-indices, resulting in B-form polynomials consisting of six individual basis functions $B^2_\kappa(b(\boldsymbol{x}))$. Using this result with Eq. (17), the vectors of per-triangle basis polynomials $B^2(\boldsymbol{x})$ for both triangles can then be constructed as follows:

$$\begin{aligned} B^2_{t_j}(\boldsymbol{x}) &= [\,B^2_{2,0,0}(b(\boldsymbol{x})) \quad B^2_{1,1,0}(b(\boldsymbol{x})) \quad B^2_{1,0,1}(b(\boldsymbol{x})) \quad B^2_{0,2,0}(b(\boldsymbol{x})) \quad B^2_{0,1,1}(b(\boldsymbol{x})) \quad B^2_{0,0,2}(b(\boldsymbol{x}))\,], \qquad j = 1,2 \\ &= [\,b_0^2 \quad 2b_0b_1 \quad 2b_0b_2 \quad b_1^2 \quad 2b_1b_2 \quad b_2^2\,], \qquad j = 1,2 \end{aligned}$$

The corresponding global vector of basis functions is

$$B^2(\boldsymbol{x}) = [\,B^2_{t_1}(\boldsymbol{x}) \quad B^2_{t_2}(\boldsymbol{x})\,]$$

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

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

and the corresponding global B-coefficient vector is

$$\boldsymbol{c} = [\,\boldsymbol{c}^T \quad \boldsymbol{c}^T\,]^T$$

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

$$\begin{aligned} \boldsymbol{x}^{t_1} &= \boldsymbol{x}(i) \in t_1\colon\ i = 1,2,3,4, \\ \boldsymbol{x}^{t_2} &= \boldsymbol{x}(i) \in t_2\colon\ i = 5,6,7,8,9,10 \end{aligned}$$

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

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

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

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

$$y(i) = B^2(i)D(i)\boldsymbol{c} + \boldsymbol{\epsilon}(i) \quad = X(i)\boldsymbol{c}$$

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

$$\begin{aligned} 0.985 &= [\,B^2(3) \quad B^2(3)\,]D(3)\boldsymbol{c} + \boldsymbol{\epsilon}(3) \\ &= [\,B^2(3) \quad B^2(3)\,]\begin{bmatrix} I_{6\times6} & 0_{6\times6} \\ 0_{6\times6} & 0_{6\times6} \end{bmatrix}\boldsymbol{c} + \boldsymbol{\epsilon}(3) \\ &= [\,b_0^2 \quad 2b_0b_1 \quad 2b_0b_2 \quad b_1^2 \quad 2b_1b_2 \quad b_2^2 \quad 0_{1\times6}\,]\boldsymbol{c} \\ &= [\,0.16 \quad 0.4 \quad 0.08 \quad 0.25 \quad 0.1 \quad 0.01 \quad 0_{1\times6}\,]\boldsymbol{c} + \boldsymbol{\epsilon}(3) \end{aligned}$$

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

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

5) Step 5: Formulate the smoothness matrix using the theory from [11]. The continuity conditions for the given triangulation are formulated using Eq. (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 nonzero 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 Eq. (31) into

$$c^{t_1}_{(m,\kappa_0,\kappa_1)} = \sum_{|\gamma|=m} c^{t_2}_{(\kappa_0,0,\kappa_1)+\gamma} B^m_\gamma(\boldsymbol{v}_0) = c^{t_2}_{(\kappa_0,0,\kappa_2)}$$

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

$$c^{t_1}_{(0,\kappa_0,\kappa_1)} = \sum_{|\gamma|=0} c^{t_2}_{(\kappa_0,0,\kappa_1)} B^0_\gamma(\boldsymbol{v}_0) = c^{t_2}_{(\kappa_0,0,\kappa_2)}$$

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

$$c^{t_1}_{(1,\kappa_0,\kappa_1)} = \sum_{|\gamma|=1} c^{t_2}_{(\kappa_0,0,\kappa_1)+\gamma} B^1_\gamma(\boldsymbol{v}_0)$$

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

$$\begin{aligned} c^{t_1}_{(1,1,0)} &= \sum_{|\gamma|=1} c^{t_2}_{(1,0,0)+\gamma} B^1_\gamma(\boldsymbol{v}_0) \\ &= c^{t_2}_{2,0,0} B^1_{1,0,0}(\boldsymbol{v}_0) + c^{t_2}_{1,1,0} B^1_{0,1,0}(\boldsymbol{v}_0) + c^{t_2}_{1,0,1} B^1_{0,0,1}(\boldsymbol{v}_0) \\ &= c^{t_2}_{2,0,0} - c^{t_2}_{1,1,0} + c^{t_2}_{1,0,1} \end{aligned}$$

where it should be noted that $b(\boldsymbol{v}_0) = (1, -1, 1)$. The complete smoothness matrix is constructed by formulating the continuity conditions for all continuity orders and for all edges, moving all terms 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 correspond to the C^1 conditions.

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

$$\hat{\boldsymbol{c}} = [\,0.842 \quad 1.1 \quad 0.626 \quad 0.926 \quad 1.23 \quad -0.0192 \quad 0.926 \quad 1.05 \quad 1.23 \quad 0.841 \quad 0.581 \quad -0.0192\,]^T$$

7) 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 $\boldsymbol{x}$, the following set of function outputs is found:

$$s(\boldsymbol{x}) = X\hat{\boldsymbol{c}} \quad = [\,0.842 \quad 0.737 \quad 0.979 \quad 0.975 \quad 0.841 \quad 0.926 \quad -0.0192 \quad 0.315 \quad 0.719 \quad 0.975\,]^T$$

Appendix B: Example of Nonaffine Spline Models in Control Allocation

This example illustrates the nonaffine nature of spline models and the practical implementation of the control allocation strategies in Sec. VII. Consider the following spline model for τ:

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

Each basis polynomial is defined on an individual simplex t_j (see also Fig. A1) in terms of local barycentric coordinates $(b_0, b_1, \ldots, b_n)$. The Cartesian to barycentric coordinate transformation is a linear one-to-one transformation given by Eqs. (9) and (10). Let $\tau(\boldsymbol{x})$ be a bivariate spline function ($n = 2$) with $b(\boldsymbol{x}) = (b_0, b_1, b_2)$ and with the spline state $\boldsymbol{x}$ consisting of one aircraft state and one control input: $\boldsymbol{x} = [\,x_a \quad u\,]^T$. 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:

$$p^{t_j}(b(\boldsymbol{x})) = \sum_{\kappa} c_{\kappa}^{t_j} \frac{d!}{\kappa!} b_0^{\kappa_0} b_1^{\kappa_1} b_2^{\kappa_2}, \qquad (x_a, u) \in t_j \tag{B1}$$

On the right-hand side of Eq. (B1), 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

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

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

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ u \end{bmatrix} \tag{B2}$$

$$b_0 = 1 - b_1 - b_2 = 1 - x_a \tag{B3}$$

Combining Eqs. (B2) and (B3) and writing as an affine function of the spline state gives

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ u \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = [\,\boldsymbol{a}_1 \quad \boldsymbol{a}_2\,]_{t_j} \begin{bmatrix} x_a \\ u \end{bmatrix} + \boldsymbol{k} \tag{B4}$$

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

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} u + \begin{bmatrix} 1 - x_a(t_0) \\ x_a(t_0) \\ 0 \end{bmatrix} = \boldsymbol{a}_2 u + \tilde{\boldsymbol{k}} \tag{B5}$$

By substituting the parameterizations in Eq. (B5) for b_0, b_1, and b_2 in Eq. (B1), the simplex polynomial can be expressed as a function only dependent of u:

$$\begin{aligned} p^{t_j}(u) &= \sum_{\kappa} c_{\kappa}^{t_j} \frac{d!}{\kappa!} (1 - x_a(t_0))^{\kappa_0} (-u + x_a(t_0))^{\kappa_1} u^{\kappa_2} \\ &= \sum_{\kappa} c_{\kappa}^{t_j} B_{\kappa}^{d}(u) \end{aligned} \tag{B6}$$

By Theorem 1, the gradient of the basis polynomial p^{t_j} with respect to u is given by

$$\begin{aligned} \nabla_u p^{t_j}(u) &= \frac{p^{t_j}(u)}{\partial u} = \boldsymbol{a}_2^T \sum_{\kappa} c_{\kappa}^{t_j} \nabla_b B_{\kappa}^{d}(u) \\ &= [\,0 \quad -1 \quad 1\,] \\ &\quad \times \left(\sum_{\kappa} c_{\kappa}^{t_j} \frac{d!}{\kappa!} \begin{bmatrix} \kappa_0 (1 - x_a(t_0))^{\kappa_0 - 1} (-u + x_a(t_0))^{\kappa_1} u^{\kappa_2} \\ (1 - x_a(t_0))^{\kappa_0} \kappa_1 (-u + x_a(t_0))^{\kappa_1 - 1} u^{\kappa_2} \\ (1 - x_a(t_0))^{\kappa_0} (-u + x_a(t_0))^{\kappa_1} \kappa_2 u^{\kappa_2 - 1} \end{bmatrix} \right) \\ &= \boldsymbol{a}_2^T \nabla_b B_{t_j}^{d}(u) \boldsymbol{c}^{t_j} \end{aligned} \tag{B7}$$

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 τ_{req} is given by

$$\Delta u = (\nabla_u p^{t_j}(u(t_0)))^{-1} (\tau_{\text{req}} - p^{t_j}(u(t_0)) \tag{B8}$$

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

References

[1] Reiner, J., Balas, G. J., and Garrard, W. L., "Flight Control Design Using Robust Dynamic Inversion and Time-Scale Separation," *Automatia*, Vol. 32, No. 11, 1996, pp. 1493–1504. doi:10.1016/S0005-1098(96)00101-X

[2] Lane, S. H., and Stengel, R. F., "Flight Control Design Using Non-Linear Inverse Dynamics," *Automatica*, Vol. 24, No. 4, 1988, pp. 471–483. doi:10.1016/0005-1098(88)90092-1

[3] Walker, G. P., and Allen, D. A., "X-35b Stovl Flight Control Law Design and Flying Qualities," *Biennial International Powered Lift Conference and Exhibit*, AIAA Paper 2002-6018, 2002.

[4] Durham, W. C., "Constrained Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 4, 1993, pp. 717–725. doi:10.2514/3.21072

[5] 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. doi:10.1016/S0967-0661(98)00024-0

[6] Sieberling, S., "Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 6, 2010, pp. 1732–1742. doi:10.2514/1.49978

[7] Kim, B. S., and Calise, A. J., "Nonlinear Flight Control Using Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 26–33. doi:10.2514/2.4029

[8] Calise, A. J., Lee, S., and Sharme, M., "Nonlinear Adaptive Flight Control Using Neural Networks," *IEEE Control Systems*, Vol. 18, No. 6, 1998, pp. 14–25. doi:10.1109/37.736008

[9] Calise, A. J., Lee, S., and Sharme, M., "Development of a Reconfigurable Flight Control Law for Tailles Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 5, 2001, pp. 896–902. doi:10.2514/2.4825

[10] Lai, M. J., and Schumaker, L. L., *Spline Functions over Triangulations*, Cambridge Univ. Press, New York, 2007.

[11] De Visser, C. C., Chu, Q. P., and Mulder, J. A., "New Approach to Linear Regression with Multivariate Splines," *Automatica*, Vol. 45, No. 12, 2009, pp. 2903–2909. doi:10.1016/j.automatica.2009.09.017

[12] De Visser, C. C., Chu, Q. P., and Mulder, J. A., "Global Nonlinear Aerodynamic Model Identification with Multivariate Splines," *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper 2009-5726, 2009.

[13] De Visser, C. C., Chu, Q. P., and Mulder, J. A., "Multidimensional Spline Based Global Nonlinear Aerodynamic Model for the Cessna Citation II," *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper 2010-7950, 2010.

[14] 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*, AIAA Paper 2011-6356, 2011.

[15] 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, pp. 153–191.

[16] 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," NASA TR-1538, 1979.

[17] Stevens, B. L., and Lewis, F. L., *Aircraft Control and Simulation*, 2nd ed., Wiley, Hoboken, NJ, 2003, pp. 107–116, 633–664.

[18] 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. doi:10.1016/j.automatica.2011.06.011

[19] 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, No. 3, 2007, pp. 1338–1354. doi:10.1137/060667207

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

[21] 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. doi:10.1016/j.ast.2011.07.004

[22] Bodson, M., "Evaluation of Optimization Methods for Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 703–711. doi:10.2514/2.4937

[23] 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, IEEE Publications, Piscataway, NJ, 2002, pp. 1295–1300.

[24] Davidson, J. B., Lalllman, F. J., and Bundick, W. T., "Real-Time Adaptive Control Allocation Applied to a High Performance Aircraft," *Society for Industrial and Applied Mathematics*, NASA, Tech. Rept. 20040086083, 2001.

[25] Alwi, H., and Edwards, C., "Fault Tolerant Control Using Sliding Modes with On-line Control Allocation," *Automatica*, Vol. 44, No. 7, 2007, pp. 1859–1866. doi:10.1016/j.automatica.2007.10.034

[26] 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. doi:10.1002/(ISSN)1099-1239

[27] 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*, AIAA Paper 1994-3609, 1994.

[28] Morelli, E. A., "Global Nonlinear Parametric Modeling with Application to F-16 Aerodynamics," NASA, Tech. Rept. 20040110310, 1997.

[29] Morelli, E. A., "Global Nonlinear Aerodynamic Modeling Using Multivariate Orthogonal Functions," *Journal of Aircraft*, Vol. 32, No. 2, 1995, pp. 270–277. doi:10.2514/3.46712

[30] Abramowitz, M., and Stegun, I. A., *Handbook of Mathematical Functions*, National Bureau of Standards, Washington, D.C., 1964.

[31] 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*, AIAA Paper 2000-4156, 2000.

[32] 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, No. 1, 2007, pp. 100–113. doi:10.2514/1.25473

[33] 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, pp. 211–216. doi:10.1109/TCST.2003.821952

[34] Marquardt, D. W., "Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM Journal on Applied Mathematics*, Vol. 11, No. 2, 1963, pp. 431–441. doi:10.1137/0111030

[35] Johansen, T. A., "Optimizing Nonlinear Control Allocation," *43rd IEEE Conference on Decision and Control*, Vol. 4, IEEE Publications, Piscataway, NJ, 2004, pp. 3435–3440.