MODELING AND MODEL INVERSION ISSUES IN A NONLINEAR AUTOMATIC FLIGHT CONTROL SYSTEM FOR THE YAV-8B HARRIER V/STOL AIRCRAFT

by

Joseph F. Sifer and George Meyer

Memorandum No. UCB/ERL M89/4

24 January 1989

MODELING AND MODEL INVERSION ISSUES IN A NONLINEAR AUTOMATIC FLIGHT CONTROL SYSTEM FOR THE YAV-8B HARRIER V/STOL AIRCRAFT

by

Joseph F. Sifer and George Meyer

Memorandum No. UCB/ERL M89/4

24 January 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

MODELING AND MODEL INVERSION ISSUES IN A NONLINEAR AUTOMATIC FLIGHT CONTROL SYSTEM FOR THE YAV-8B HARRIER V/STOL AIRCRAFT

by

Joseph F. Sifer and George Meyer

Memorandum No. UCB/ERL M89/4

24 January 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Modeling and Model Inversion Issues in a Nonlinear Automatic Flight Control System for the YAV-8B Harrier V/STOL Aircraft[1]

Joseph F. Sifer[2]

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720

George Meyer

Flight Dynamics and Control Branch
NASA-Ames Research Center
Moffett Field, CA 94035

ABSTRACT

This report is a self-contained discussion of the application of differential geometric nonlinear control theory to the design of an automatic flight control system for the YAV-8B Harrier V/STOL aircraft. The report and the work it represents are the results of a continuing joint effort on the part of the Flight Dynamics and Control Branch at NASA/Ames Research Center, Moffett Field, and the control systems research group at the University of California - Berkeley. We consider a formal presentation of the model follower control system used at NASA/Ames as it is applied specifically to the Harrier aircraft. We discuss the application of sophisticated nonlinear curve fitting techniques, namely, tensor product spline interpolation, to the multidimensional force and moment generation process of the Harrier. We present the application of a hybrid algorithm for solving a system of nonlinear equations to the crucial aircraft trim operation of the model follower control system. Finally, we offer some preliminary results regarding the performance of the control system design and discuss the state of continuing work.

January 24, 1989

---

[1] We wish to thank Prof. C. A. Desoer for his thorough and constructively critical review of this report. The usefulness and quality of this report were improved many times over as a result of numerous rewrites and clarifications suggested by Prof. Desoer.

[2] Work completed as a research associate at NASA-Ames in the Flight Dynamics and Control Branch, supported under the SJSU/NASA-Ames Research and Development Program, (NASA)NCA-295. The publication of this technical memo was funded by grant (NASA)NAG 2-243.

To the loving memory of my grandmother,
Frances R. Sifer

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research adviser, Prof. Charles A. Desoer. His enthusiasm for learning, both in the role as teacher and as student, has been an inspiration. His meticulous and constructively critical review of this report has set a standard of excellence that I shall use throughout my career. His careful and congenial guidance of my program at UC-Berkeley has made an almost unbearable experience worthwhile. His quick wit and openness have bridged the adviser-student gap and have made us friends as well as colleagues. I consider myself a fortunate man for having had the opportunity to learn from such a great man.

I also am very grateful to Dr. George Meyer of the Flight Dynamics and Control Group at the NASA/Ames Research Center, Moffett Field. His assistance with and supervision of my work at Ames have been invaluable. His graceful demeanor and quiet disposition have assured me that there is a place for integrity and professionalism in the working world. His advice often proved to be the perfect complement to that of Prof. Desoer and, like Desoer, Dr. Meyer's quick wit and openness have made us friends as well as colleagues.

I also thank Prof. Shankar Sastry for his willingness to involve me in the Harrier project and his not-so-gentle prodding of me to take a position at Ames to work closely with Dr. Meyer. I also thank John Hauser for his useful suggestions and discussions regarding my work on the Harrier and Andy Teel for his willingness to act as a sounding board for the various incarnations of some of the concepts presented in this report.

I would also like to thank my family and friends for their support and encouragement during this stage of my life. I am especially grateful to Coach Lou Holtz for his tireless efforts at fielding a Fightin' Irish football team that brought a previously unexperienced level of excitement to this Domer's Saturday afternoons. Go Irish!

I wish to acknowledge the financial support received under the SJSU/NASA-Ames Research and Development Program, (NASA)NCA-295.

Table of Contents

Chapter 1 — Introduction

This report is a self-contained discussion of the application of differential geometric nonlinear control theory to the design of an automatic flight control system for the YAV-8B Harrier V/STOL aircraft. The report contains detailed presentations of the techniques and methodologies used as well as some preliminary results regarding the control system performance. The report and the work it represents are the results of a continuing joint effort on the part of the Flight Dynamics and Control Branch at NASA/Ames Research Center, Moffett Field, and the control systems research group at the University of California - Berkeley. The goal of this joint effort is the formalization of applications of differential geometric control theory to specific classes of nonlinear systems particularly those defined by high performance aircraft such as the Harrier. The main contributions of this report to our goal are presented in Chapters 2, 3, and 4.

In Chapter 2, we consider a formal presentation of the model follower control system used at Ames by Meyer, et. al., [MSH82], [MSH83G] as it is applied specifically to the Harrier aircraft. This involves the representation of the Harrier natural model state equation by a sixteen-dimensional first order nonlinear vector differential equation. Further, it includes the specification of the Harrier Brunovsky canonical model as well as the calculation of the nonlinear transformations maps T, which takes the natural model to the linear Brunovsky canonical model, and W, which takes the canonical back to the natural.

In Chapter 3, we discuss the application of sophisticated nonlinear curve fitting techniques, namely, tensor product spline interpolation, to the multidimensional force and moment generation process of the Harrier. Data generated from the complete simulation model of the Harrier aircraft is used as the basis for determining the tensor product spline model. Such a powerful modeling technique is used to accurately represent -- in the force and moment model -- the cross-coupling between the aircraft controls over the full range of control values. It is through such a technique that a highly accurate representation of the Harrier natural model is achieved and it is with such an accurate model that the calculation of the maps T and W are made more precise.

In Chapter 4, we present the application of a hybrid algorithm for solving a system of nonlinear equations to the crucial aircraft trim operation of the model follower control system. The method is a cleverly crafted blend of the standard Newton method for solving a system of nonlinear equations and the

method of steepest descent from engineering optimization theory. Use of the hybrid algorithm preserves the convergence properties of the Newton method and results in a more accurate and robust trim operation and, consequently, a more accurate W map.

In Chapter 5, we offer some preliminary results and conclusions regarding the performance of the control system design. We discuss the state of continuing work and offer suggestions regarding possible future work.

CHAPTER 2 — Model Follower Control Scheme

§ 2.1 *Introduction*

In this chapter, the model follower control system used in the application of the nonlinear control transformation theory at the NASA-Ames Research Center by Meyer, et. al., to multi-input aerodynamic systems is presented [MSH82], [HSMmi], [MSH83G], [MSH83C]. The general block diagram of this automatic flight control system is given in Figure 2.1.1. The goal of the model follower control system is the generation of the true aircraft command inputs **u** which cause the aircraft response **r** to follow commanded $\mathbf{r}^*$.

The control system of Fig. 2.1.1 consists of four main sections: the natural model representation of the aircraft; the Brunovsky canonical model representation of the aircraft; the transformation maps, T and W; and the linear regulator. The transformation map T takes the natural model state **x** and control **u** to the Brunovsky canonical model state **y** and control **w**. The commanded state y_c is then subtracted from the transformed state **y** producing the plant-to-model error e_y (i.e., $e_y = y - y_c$). The linear regulator is driven by the error e_y and produces a control correction δw which drives e_y to zero. The adjusted control **w** is transformed by map W from the Brunovsky canonical form to the natural model form, yielding the control input **u** which will cause **r** to follow $\mathbf{r}^*$.

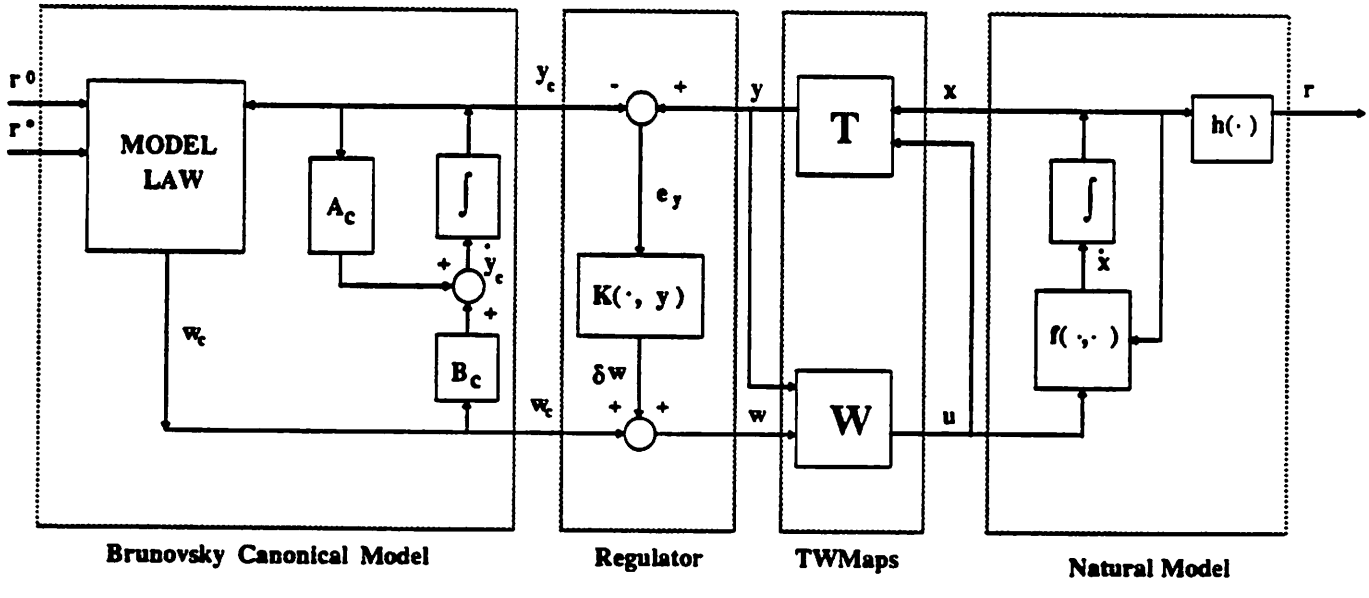

Figure 2.1.1 : Model Follower Control Scheme

Generation of Harrier input commands **u** which cause Harrier response **r** to follow commanded $\mathbf{r}^*$. Note that the regulator never sees the nonlinear natural model. All regulation is done on the Brunovsky canonical form.

In the subsequent sections of this chapter, the four components of the model follower control system as applied to the automatic control of the YAV-8B Harrier V/STOL aircraft are discussed in detail. Where appropriate, the general theoretical results of the Brunovsky canonical form and of the transformation of nonlinear systems to Brunovsky canonical form are presented.

§ 2.2 *Plant: YAV-8B Harrier Natural Model*

The YAV-8B Harrier V/STOL aircraft is represented as a rigid body moving in three dimensional space in response to gravity, aerodynamics, and propulsion. The state $\mathbf{x}$ and control $\mathbf{u}$ are defined as

$$\mathbf{x} := \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ \mathbf{C} \\ \omega \\ \mathbf{p}^1 \\ \mathbf{p}^2 \end{bmatrix} \tag{2.2.1}$$

$$\mathbf{u} := \begin{bmatrix} \mathbf{u}^M \\ \mathbf{u}^P \end{bmatrix} \tag{2.2.2}$$

with:

$\mathbf{r}$:= runway coordinates of body center of mass position;

$\mathbf{v}$:= runway coordinates of body center of mass velocity;

$\mathbf{C}$:= direction cosine matrix of body-fixed axes relative to runway-fixed axes;

ω := body coordinates of angular velocity relative to runway-fixed axes;

$\mathbf{p}^1$:= throttle position (p_1^1) and nozzle angle (p_2^1) ("power" positions);

$\mathbf{p}^2$:= throttle rate (p_1^2) and nozzle rate (p_2^2) ("power" rates);

$\mathbf{u}^M$:= three axes moment control as issued by the pilot for deflection of the ailerons, elevator, and rudder;

$\mathbf{u}^P$:= power control as issued by the pilot for adjustment of the nozzle angle and the throttle position;

Note that $\mathbf{r} \in \mathbb{R}^3$, $\mathbf{v} \in \mathbb{R}^3$, $\omega \in \mathbb{R}^3$, $\mathbf{C} \in SO(3)$, $\mathbf{p}^1 \in \mathbb{R}^2$, $\mathbf{p}^2 \in \mathbb{R}^2$, $\mathbf{u}^M \in \mathbb{R}^3$, and $\mathbf{u}^P \in \mathbb{R}^2$. In (2.2.1), there

is an abuse of notation in viewing matrix C as a sub-vector of the complete state vector x. Note C, which moves on the sphere SO(3), defines aircraft attitude and one possible representation for C is in terms of the standard Euler angles of **roll** (ϕ), **pitch** (θ), and **yaw** (ψ). The reader is directed to Appendix 1 for a formal definition of the Euler angles. Note that C is C_{BR} of Appendix 1 (see equation (A.1.8)). Note also that in Appendix 1, the coordinate frames utilized in the control design for the Harrier are given along with explicit expressions for the transformation matrices which relate the coordinate frames to one another. As a familiarity with this material is essential to the balance of this report, it is assumed that the reader will study Appendix 1 at this point.

All coordinate frames used in the control design problem at hand are right-hand-rule systems with the mutually orthogonal axes labeled as the 1-axis, 2-axis, and 3-axis. The inertial coordinate frame is taken as the runway-fixed axes which are the conventional 1-axis as north, 2-axis as east, and 3-axis as down (i.e., the N-E-D coordinate frame). The body coordinate frame is defined at the center of mass of the Harrier with the 1-axis directed along the line segment from cm to out the nose of the aircraft, the 2-axis directed from cm to out the right wing, and the 3-axis directed as the cross product of the 1-axis unit vector with the 2-axis unit vector dictates. By rotating the N-E-D frame through the standard Euler angles ϕ, θ, and ψ, the transformation to the body frame is achieved.

The state equation for the Harrier natural model representation is essentially a sixteen-dimensional first order nonlinear vector differential equation. The equation consists of the translational and rotational kinematic and dynamic equations for the Harrier with mass and moments of inertia normalized to 1:

$$\dot{\mathbf{r}} = \mathbf{v} \tag{2.2.3}$$

$$\dot{\mathbf{v}} = \mathbf{C}^T \mathbf{f}_B^F(\mathbf{x},\mathbf{u}) + g\,\mathbf{e}_3 \tag{2.2.4}$$

$$\dot{\mathbf{C}} = \mathbf{S}(\omega)\mathbf{C} \tag{2.2.5}$$

$$\dot{\omega} = \mathbf{f}^M(\mathbf{x},\mathbf{u}) \tag{2.2.6}$$

$$\dot{\mathbf{p}}^1 = \mathbf{p}^2 \tag{2.2.7}$$

$$\dot{\mathbf{p}}^2 = \mathbf{g}(\mathbf{u}^P) \tag{2.2.8}$$

where:

$f^F(x,u) :=$ runway coordinates of the components of force acting on the aircraft as a function of state x and control u;

$f^M(x,u) :=$ body coordinates of the moments acting on the aircraft as a function of state x and control u;

$g(u^P) :=$ a smoothed version of $u^P(t)$;

$S(\omega) :=$ a skew symmetric matrix of the angular velocity vector:

$$S(\omega) = \begin{bmatrix} 0 & \omega(3) & -\omega(2) \\ -\omega(3) & 0 & \omega(1) \\ \omega(2) & -\omega(1) & 0 \end{bmatrix} \tag{2.2.9}$$

With the specification of the state equation, the description of the natural model is complete. A thorough analysis of the form of equations (2.2.4) and (2.2.6) is given in a McDonnell Douglas Corporation simulation and modeling report [McD82]. It is this report which forms the basis for the simulation and modeling code developed at Ames for the Harrier. The simulation and modeling report shows that (2.2.4) has the form

$$\dot{v} = C^T f_B^F(v_{AB}, \omega, p^1, u^M) + g\, e_3 \tag{2.2.10}$$

and that (2.2.6) has the form

$$\dot{\omega} = f_B^M(v_{AB}, \omega, p^1, u^M) \tag{2.2.11}$$

where g is gravitational acceleration, e_3 is $[0, 0, 1]^T$, and v_{AB} is the wind-adjusted body coordinates of aircraft velocity. That is, $v_{AB} = Cv_{AR}$ with $v_{AR} := (v - v_w)$ where v_w is the runway-fixed coordinates of wind velocity. Figure 2.2.1 is a detailed block diagram of the natural model of the Harrier.

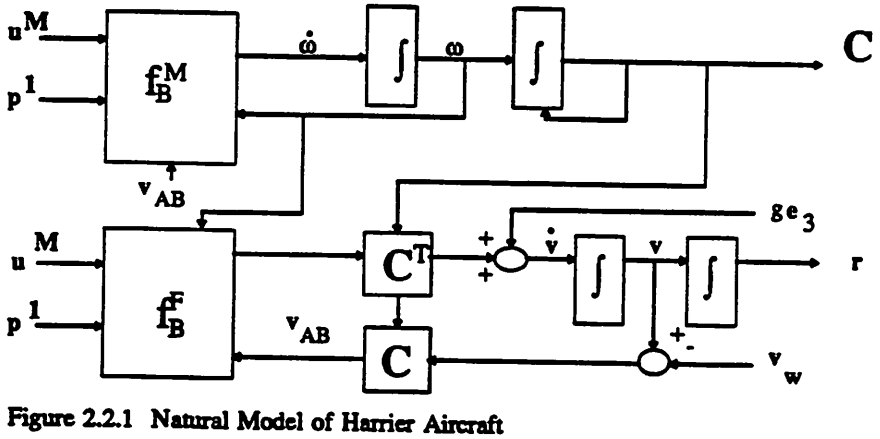

Figure 2.2.1 Natural Model of Harrier Aircraft

§ 2.3 *Brunovsky Canonical Model*

In this section, the Brunovsky canonical representation of the Harrier model is presented. First, the basic concept of the Brunovsky form is introduced and the typical characterization as strings of integrators is given [Ford83], [DeLuc87]. Then, these concepts are specialized to the Harrier model to provide a complete description of the canonical model of Figure 2.1.1.

The **Brunovsky canonical form** is a realization of a controllable linear system given by

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{w}, \quad \mathbf{y} \in \mathbb{R}^n, \mathbf{w} \in \mathbb{R}^m \tag{2.3.1}$$

where:

$$\mathbf{A} = \textit{block diagonal } [\mathbf{A}_1, \ldots, \mathbf{A}_m]$$

$$\mathbf{B} = \textit{block diagonal } [\mathbf{B}_1, \ldots, \mathbf{B}_m]$$

$$\mathbf{A}_i = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \ldots & 1 \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}_{(\kappa_i \times \kappa_i)} ; \quad \mathbf{B}_i = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix}_{(\kappa_i \times 1)} \qquad i = 1, \ldots, m.$$

The integers κ_i for i = 1, . . . , m are the orders of blocks $\mathbf{A}_i$ and are called the **Kronecker indices** of the system (in the literature, the term "controllability indices" is often used as well). Note that $\sum_{i=1}^{m} \kappa_i = n$ and that we order the κ_i by the rule $\kappa_1 \geq \kappa_2 \geq \ldots \geq \kappa_m$. Note also that

$$\textit{rank } [\mathbf{B}, \mathbf{AB}, \ldots, \mathbf{A}^{n-1}\mathbf{B}] = n$$

as required by controllability.

For the characterization of the Brunovsky canonical form as m decoupled strings or channels of integrators, consider the following partition of the state vector y:

$$\mathbf{y} = [y_1 \ldots y_{s_1} \mid y_{s_1+1} \ldots y_{s_2} \mid \ldots \mid y_{s_{m-1}+1} \ldots y_{s_m}]^T \tag{2.3.2}$$

where $s_i = \sum_{j=1}^{i} \kappa_j$.

Note that for i = 1, . . . , m and for j = 1, . . . , s_m,

$$\dot{y}_j = \begin{cases} y_{j+1} & j \neq s_i \\ w_i & j = s_i \end{cases} \tag{2.3.3}$$

Thus, it is apparent from (2.3.2) and (2.3.3) that the Brunovsky canonical form yields m strings or channels of integrators decoupled from each other where string i is κ_i integrators long with w_i as input to the first integrator, y_{s_i} as output of the first integrator/input to the second integrator, . . . , $y_{s_{i-1}+1}$ as output of the m-th integrator of the string (see Figure 2.3.1).

Figure 2.3.1 i^{th} string of κ_i integrators where ——●—— represents an integrator.

The following example illustrates the Brunovsky canonical form (see Figure 2.3.2).

EXAMPLE (2.3.1):

$$y \in \mathbb{R}^6 ; w \in \mathbb{R}^3$$

$$\kappa_1 = 3, \kappa_2 = 2, \text{ and } \kappa_3 = 1$$

note $\kappa_1 \geq \kappa_2 \geq \kappa_3$ and $\kappa_1 + \kappa_2 + \kappa_3 = 6 = n$

$$s_1 = 3, s_2 = 5, \text{ and } s_3 = 6$$

$$\dot{y} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} y + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} w$$

□

Harrier Brunovsky Model

The Brunovsky canonical model for the Harrier is illustrated in Figure 2.3.3. Note that

$$n = 16, m = 5,$$

$$\kappa_1 = \kappa_2 = \kappa_3 = 4, \kappa_4 = \kappa_5 = 2$$

$$s_1 = 4, s_2 = 8, s_3 = 12, s_4 = 14,$$

$$\text{and } s_5 = 16.$$

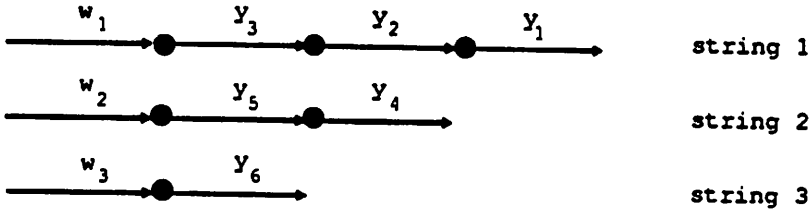

Figure 2.3.2 Brunovsky Canonical form for Example (2.3.1)

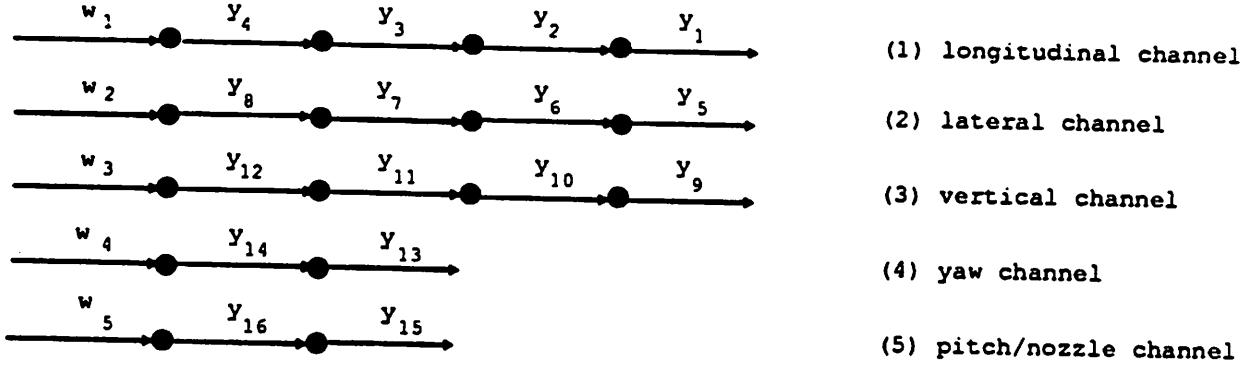

Figure 2.3.3 Harrier Brunovsky Canonical Model represented as five decoupled strings of integrators.

The following are the characterizations of the 5 channels of integrators of the model:

Channel (1):

Corresponds to the runway-fixed 1-axis component of position (y_1), velocity (y_2), acceleration (y_3), acceleration rate (y_4), and rate of acceleration rate (w_1);

Channel (2):

Corresponds to the runway-fixed 2-axis component of position (y_5), velocity (y_6), acceleration (y_7), acceleration rate (y_8), and rate of acceleration rate (w_2);

Channel (3):

Corresponds to the runway-fixed 3-axis component of position (y_9), velocity (y_{10}), acceleration (y_{11}), acceleration rate (y_{12}), and rate of acceleration rate (w_3);

Channel (4):

Corresponds to the yaw angle (y_{13}), the yaw angle rate (y_{14}), and the yaw angle acceleration (w_4);

Channel (5):

Corresponds to the pitch angle or the nozzle angle (y_{15}), the pitch angle rate or nozzle angle rate (y_{16}), and the pitch angle acceleration or nozzle angle acceleration (w_5). So, this channel is either a pitch channel or a nozzle channel. This is a consequence of the convention that for the Harrier that either pitch is commanded or nozzle is commanded. This will be clarified in the discussion of transformations T and W.

Model Law Generation

In Fig. 2.1.1, the model law generation system is shown as a subsystem of the Brunovsky canonical form block. The model law block requires an initial condition for the Brunovsky state and control ($\mathbf{r}^0$ in Fig. 2.1.1) as well as an endpoint condition on the Brunovsky state and control ($\mathbf{r}^*$ in Fig. 2.1.1) along with initial and final time values and generates polynomial expansions in time t of the time history trajectories for each of the channels of the Brunovsky form. The model law generation algorithm is summarized as follows. The given data is $t_o, t_f, \mathbf{y}(t_o), \mathbf{w}(t_o), \mathbf{y}(t_f)$, and $\mathbf{w}(t_f)$. A function $s(t) = y_i(t)$ is defined for $i = 1, 5, 9, 13, 15$. For $i = 1, 5, 9$, $s(t) = \sum_{j=0}^{9} a_i t^i$. Differentiate $s(t)$ four times and apply the initial and final conditions to yield a system of ten linearly independent equations in ten unknowns, $a_0, a_1, \ldots, a_9$. The time history of channels 1, 2, and 3 is completely specified by solving for $a_0, a_1, \ldots, a_9$ for each channel. For $i = 13, 15$, $s(t) = \sum_{j=0}^{5} a_i t^i$. Differentiate $s(t)$ twice and apply the initial and final conditions to yield a system of six linearly independent equations in six unknowns, $a_0, a_1, \ldots, a_5$. The time time history of channels 4 and 5 is completely specified by solving for $a_0, a_1, \ldots, a_5$ for each channel.

Thus, out of the model law generation pops the canonical control w and the canonical state y for all values of time in the given time interval (t_o, t_f).

§ 2.4 *Transformation Maps*, T and W

In this section, the transformation of multi-input nonlinear systems of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) + \sum_{i=1}^{m} \mathbf{g}_i(\mathbf{x}) u_i \tag{2.4.1}$$

to the linear Brunovsky canonical form is considered. The transformation of linear systems to Brunovsky canonical form is illustrated by way of an example. The necessary and sufficient conditions for the existence of a transformation for the general nonlinear case are then considered and a demonstrative example is given. Lastly, the transformation of the Harrier natural model to its Brunovsky canonical model is discussed (i.e., the T and W maps of Fig. 2.1.1 are formalized).

Any constant linear controllable system can be transformed to Brunovsky canonical form. That is, for the linear system

$$\dot{\mathbf{x}} = \hat{\mathbf{A}}\mathbf{x} + \hat{\mathbf{B}}\mathbf{u} \tag{2.4.2}$$

with $(\hat{\mathbf{A}}, \hat{\mathbf{B}})$ controllable, there exists a nonsingular transformation matrix T such that

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{w} \end{bmatrix} = \mathbf{T} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \tag{2.4.3}$$

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{w} \tag{2.4.4}$$

where (A, B) are in Brunovsky form. Consider the following example of a system of the form (2.4.2):

EXAMPLE (2.4.1):

$$\dot{\mathbf{x}} = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ -0.5 & -1.0 & -1.5 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 1.0 & -3.0 \\ 0 & 2.0 \end{bmatrix} \mathbf{u} \tag{2.4.5}$$

$$rank\ [\hat{\mathbf{B}} \mid \hat{\mathbf{A}}\hat{\mathbf{B}} \mid \hat{\mathbf{A}}^2\hat{\mathbf{B}}] = rank \begin{bmatrix} 0 & 0 & 2.0 & 0 & 0 & 0 \\ 1.0 & -3.0 & -1.0 & 0 & 0 & 0 \\ 0 & 2.0 & 0 & 0 & 0 & 0 \end{bmatrix} = 3 = n$$ therefore, $(\hat{\mathbf{A}}, \hat{\mathbf{B}})$ in (2.4.5) controllable;

$$\text{for } \mathbf{T} := \textit{block diagonal } [\mathbf{T}_1, \mathbf{T}_2] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix};$$

$$\mathbf{T}^{-1} := \textit{block diagonal } [\mathbf{T}_1^{-1}, \mathbf{T}_2^{-1}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -0.5 & 0.5 & -1.5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2.0 & 0 \\ 0 & 0 & 0 & 0 & 2.0 \end{bmatrix}$$

$$\mathbf{A} = \mathbf{T}_1 \hat{\mathbf{A}} \mathbf{T}_1^{-1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \quad \mathbf{B} = \mathbf{T}_1 \hat{\mathbf{B}} \mathbf{T}_2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and, therefore, $y_1 = x_1$, $y_2 = x_1 + 2x_2 + 3x_3$, $y_3 = x_3$, $w_1 = 0.5u_1$, $w_2 = 0.5u_2$, and $\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{w}$.

□

Necessary and Sufficient Conditions for Existence of Linearizing Transformations

The transformation of a nonlinear system described by

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}) \tag{2.4.6}$$

into a Brunovsky form is not, in general, possible. From a heuristic point of view, when the system (2.4.6) can be linearized (i.e., transformed to the Brunovsky canonical form under a mapping $\mathbf{T}(\mathbf{x}, \mathbf{u})$), the non-linearities of (2.4.6) are said to be non-intrinsic; that is, they are the unfortunate result of a modeling scheme which failed to recognize the inherent linear nature of the system.

There are four conditions to check for (2.4.6) to be linearizable. First, it is necessary to construct a one-to-one, invertible function $\mathbf{h}$ so as to define a new control variable $\hat{\mathbf{u}}$ such that

$$\hat{\mathbf{u}} = \mathbf{h}^{-1}(\mathbf{x}, \mathbf{u}) \tag{2.4.7}$$

$$\mathbf{u} = \mathbf{h}(\mathbf{x}, \hat{\mathbf{u}}) \tag{2.4.8}$$

and

$$\mathbf{F}(\mathbf{x}, \mathbf{h}(\mathbf{x}, \mathbf{u})) = \mathbf{f}(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{g}_i(\mathbf{x}) \hat{u}^i \tag{2.4.9}$$

The interpretation of (2.4.9) is that the new control variable $\hat{\mathbf{u}}$ enters linearly into the field $\mathbf{F}(\mathbf{x}, \mathbf{h}(\mathbf{x}, \hat{\mathbf{u}}))$. The remaining three conditions, the so-called **Hunt–Su conditions**, follow from results in differential geometry and are best given in terms of Lie brackets defined as follows.

If $\mathbf{f}$ and $\mathbf{g}$ are C^∞ vector fields on $\mathbb{R}^n$, the Lie bracket of $\mathbf{f}$ and $\mathbf{g}$ is

$$[\mathbf{f}, \mathbf{g}] = \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{f} - \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{g} \tag{2.4.10}$$

and the following relationships are defined recursively:

$$(ad^0 \mathbf{f}, \mathbf{g}) = \mathbf{g}$$

$$(ad^1 \mathbf{f}, \mathbf{g}) = [\mathbf{f}, \mathbf{g}]$$

$$(ad^2 \mathbf{f}, \mathbf{g}) = [\mathbf{f}, [\mathbf{f}, \mathbf{g}]]$$

$$(ad^k \mathbf{f}, \mathbf{g}) = [\mathbf{f}, (ad^{k-1} \mathbf{f}, \mathbf{g})].$$

For an application of the Lie bracket, note that the involutivity of a set of vector fields is verified using the Lie bracket. A set of C^∞ vector fields $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_r$ on $\mathbb{R}^n$ is said to be **involutive** if there exist C^∞ functions γ_{ijk} with

$$[\mathbf{h}_i, \mathbf{h}_j] = \sum_{k=1}^{r} \gamma_{ijk} \mathbf{h}_k, \quad 1 \le i, j \le r, i \ne j \tag{2.4.11}$$

To consider the transformation of (2.4.9) to the Brunovsky canonical form with Kronecker indices $\kappa_1 \ge \kappa_2 \ge \ldots \ge \kappa_m$, define the following sets:

$$C = \left\{ g_1, [f, g_1], \ldots, (ad^{\kappa_1 - 1} f, g_1), g_2, [f, g_2], \ldots, (ad^{\kappa_2 - 1} f, g_2), \ldots, g_m, [f, g_m], \ldots, (ad^{\kappa_m - 1} f, g_m) \right\}$$

$$C_j = \left\{ g_1, [f, g_1], \ldots, (ad^{\kappa_j - 2} f, g_1), g_2, [f, g_2], \ldots, (ad^{\kappa_j - 2} f, g_2), \ldots, g_m, [f, g_m], \ldots, (ad^{\kappa_j - 2} f, g_m) \right\},$$

for

$$j = 1, 2, \ldots, m.$$

Define the set $X \subseteq \mathbb{R}^n$ to consist of all n-tuples for which f and g are defined. An n-tuple $x \in X$ is said to be admissable. Then it can be shown [HSMmi] that the transformation is possible if and only if at each admissable x,

1. The set C spans an n-dimensional space;
2. Each C_j is involutive for $j = 1, 2, \ldots, m$;
3. The span of C_j equals the span of $C_j \cap C$ for $j = 1, 2, \ldots, m$.

For a linear system of the form (2.4.2), the spanning condition (1) on C is exactly the controllability condition and conditions (2) and (3) are satisified immediately.

Consider the following illustrative example [HSMmi]:

EXAMPLE (2.4.2):

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{bmatrix} = \begin{bmatrix} \sin(x_2) \\ \sin(x_3) \\ x_4^3 \\ x_5 + x_4^3 - x_1^{10} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_2$$

thus $\dot{x} = f + g_1 u_1 + g_2 u_2$. Making the necessary computations, it is clear that

$$[f, g_1] = \begin{bmatrix} 0 \\ -\cos(x_3) \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (ad^2 f, g_1) = \begin{bmatrix} \cos(x_2)\cos(x_3) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad [f, g_2] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix};$$

hence, $C = \left\{ g_1, [f,g_1], (ad^2 f, g_1), g_2, [f,g_2] \right\}$ spans a 5-dimensional space on the set

$V = \left\{ (x_1, x_2, x_3, x_4, x_5): -\frac{\pi}{2} < x_2, x_3 < \frac{\pi}{2} \right\}$. The appropriate Kronecker indices in this case are $\kappa_1 = 3$ and $\kappa_2 = 2$. Note that $C_1 \cap C = \left\{ g_1, [f,g_1], g_2, [f,g_2] \right\}$ is involutive since

$$[g_1, [f,g_1]] = \begin{bmatrix} 0 \\ \sin(x_3) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

and all other Lie brackets vanish, and $C_2 \cap C = \left\{ g_1, g_2 \right\}$ is trivially involutive. Therefore, there exists a transformation which maps the original nonlinear system to the appropriate Brunovsky canonical form.

□

General Comments Regarding Linearization

It can be shown that the construction of the transformation map T is equivalent to the solution of an over-determined system of partial differential equations [HSMmi]. This system of partial differential equations falls out of the proof for the existence of linearizing transformations for systems satisfying the linearization conditions. However, the control system designer who hopes to construct the transformation T by a precise application of the theory faces one serious limitation: for aircraft with complicated nonlinear dynamics, the functions f and g of (2.4.9) typically do not have smooth, analytic forms. As is the case with the Harrier, all that the designer has to work with is a highly detailed simulation model based on relationships represented by look-up tables generated from three sources of increasing accuracy: (i) computer-aided aircraft design results; (ii) wind tunnel test results both on small-scale and full-scale models; and (iii) flight test results. It is not possible, then, to check the Hunt-Su linearization conditions and, consequently, determine the precise transformation T. Thus, the control system designer needs to be led by sound engineering judgement in applying and developing an approximation to the true transformation theory. In [MSH82], [MSH83G], and [MSH83C], such was the approach taken for the design of a helicopter automatic pilot. This approach is specialized to the Harrier in the remainder of this section.

Note that for systems which can be accurately modeled by smooth, analytic functions, computer programs written in MACSYMA do exist which perform, symbolically, the necessary computations needed to verify the Hunt-Su conditions [Blank]. Also, at UC-Berkeley, a similar program is under development [TeelMS] with the hope of devising a means for approximately verifying Hunt-Su-like conditions via calculations done on a sophisticated and highly accurate smooth nonlinear model of the plant dynamics (e.g., tensor product spline functions as discussed in Chapter 3 of this report).

The approximation method used in the application of the transformation theory is based on the following premise: given a commanded trajectory (i.e., given $\mathbf{r}^*$ and $\mathbf{r}^\theta$ of Fig. 2.1.1 and the resulting trajectories generated by the Model Law block of Fig. 2.1.1), and the corresponding normalized forces and moments acting on the aircraft ($\dot{\mathbf{v}}$ and $\dot{\omega}$, respectively, of Fig. 2.2.1), we are able to -- under a reasonable set of assumptions -- invert the force and moment equations thus solving for the time history of the aircraft natural controls which will generate the commanded state trajectories. In this regard, the selected forces and moments calculated in the inversion process are interpreted as new independent controls for the system. Recasting the state equations in terms of these new controls yields a set of equations which are interpreted as the standard trim equations for aircraft state and control. Recalling that we have normalized mass and the moments of the inertia of the aircraft to unity, we define the term **trim** to mean a balance between desired accelerations and the actual forces and moments acting on the aircraft and we define those equations used to strike this balance the **trim equations**. Further, we define the commanded forces and moments to be the **trim conditions** and the corresponding controls (solved for in the trim process) the **trim variables**. Approximate **T** and **W** maps can then constructed.

Harrier Trim Equations and TW Maps

Consider equations (2.2.10) and (2.2.11). Six trim conditions (three components of force and three moments) can be commanded. Eight quantities are free to be specified in such a way as to achieve the trim conditions: the five control conditions (taken as the moment controls $\mathbf{u}^M$ and the power positions $\mathbf{p}^1$) and the aircraft attitude (as specified by the three Euler angles represented by the direction cosine matrix C) are taken as the trim variables where the values of these variables are to be such that they achieve the

commanded forces and moments. To achieve a square system (i.e, same number of trim conditions as trim variables), two additional trim conditions need to be specified. The convention for assigning the additional trim conditions has resulted in two distinct modes of operation for the Harrier: the nozzle free mode called **Mode 1** and the nozzle commanded mode called **Mode 2**. Mode 1 leads to an eight degree-of-freedom trim while Mode 2 results in a seven degree-of-freedom trim. Refering to the Brunovsky canonical model for the Harrier (see Fig. 2.3.3), Mode 1 corresponds to the fifth channel as a pitch channel while Mode 2 corresponds to the nozzle channel interpretation of channel (5).

Mode 1 operation, the nozzle free mode, is characterized by the specification of commanded pitch angle, $\theta_c(t)$, and commanded yaw angle, $\psi_c(t)$, as well as commanded moments, $\dot{\omega}_c(t)$, and commanded forces, $\dot{v}_c(t)$. The nozzle angle p_2^1 is free to be used in the trim process along with $\mathbf{u}^{\mathbf{M}}$, C, and p_1^1. Thus, there are eight trim variables and eight trim conditions for an eight degree-of-freedom trim.

Mode 2 operation, the nozzle commanded mode, is characterized by the specification of commanded nozzle angle, $\eta_c(t)$, and commanded yaw angle, $\psi_c(t)$ as well as commanded moments, $\dot{\omega}_c(t)$, and forces, $\dot{v}_c(t)$. The nozzle angle p_2^1 is commanded and, therefore, is not available as a trim variable. That is, over the time interval of interest, $p_2^1(t)$ is known. Thus, the trim operation becomes a seven degree-of-freedom trim of the moments control, the throttle position, and the attitude. Attention is restricted to Mode 2 operation of the Harrier for the balance of the report.

As motivation for determining a new independent control variable as in (2.4.7), so as to fascilitate the required trim operation, consider two special cases of commanded nozzle angle: (i) $\eta_c(t) = 90°$ for all t and (ii) $\eta_c(t) = 0°$ for all t. Define

$$\hat{\mathbf{u}} = \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \\ \hat{u}_4 \end{bmatrix} := \begin{bmatrix} \dot{\omega} \\ \dot{v}_* \end{bmatrix} \tag{2.4.12}$$

where $\dot{v}_*$ is $\dot{v}_{B,3}$ for case (i) and $\dot{v}_*$ is $\dot{v}_{B,1}$ for case (ii). In either case, (2.2.11) is invertible with respect to the pair $[(\hat{u}_1, \hat{u}_2, \hat{u}_3), \mathbf{u}^{\mathbf{M}}]$; that is, for specified normalized moments $\dot{\omega}$, (2.2.11) can be inverted to solve for the moment controls $\mathbf{u}^{\mathbf{M}}$. Recalling that p_1^1 is the throttle position, we note that in either case, (2.2.10)

is invertible with respect to the pair $[\hat{u}^4, p_1^1]$. That is, for case (i), (2.2.10) is invertible with respect to the pair $[\dot{v}_{B,3}, p_1^1]$ (where $\dot{v}_{B,3}$ is the body 3-axis component of translational acceleration) while for case (ii) it is invertible with repsect to the pair $[\dot{v}_{B,1}, p_1^1]$ (where $\dot{v}_{B,1}$ is the body 1-axis component of translational acceleration).

Observe that in both (i) and (ii), $\dot{v}_*$ is in the direction of the nozzle 1-axis. This suggests that in the general case $\dot{v}_*$ is $\dot{v}_{N,1}$, the nozzle 1-axis of acceleration. Therefore, in the general case, (2.2.10) expressed in nozzle coordinates is invertible with respect to the pair $[\dot{v}_{N,1}, p_1^1]$. Thus the specification of $\hat{\mathbf{u}}$ is complete with $\hat{u}_4$ taken to be $\dot{v}_{N,1}$ and a function $\mathbf{h}^1$: $\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^4 \rightarrow \mathbb{R}^4$ can be constructed such that

$$\begin{bmatrix} \mathbf{u}^M \\ p_1^1 \end{bmatrix} = \mathbf{h}^1(\mathbf{v}_{AB}, \omega, \hat{\mathbf{u}}) \tag{2.4.13}$$

With $\hat{\mathbf{u}}$ as the new vector of independent controls, the state equation (2.2.4), expressed in the nozzle coordinate frame, becomes

$$\dot{v}_{N,1} = \hat{u}_4 \tag{2.4.14}$$

$$\begin{bmatrix} \dot{v}_{N,2} \\ \dot{v}_{N,3} \end{bmatrix} = \mathbf{f}_N^0(\mathbf{v}_{AB}) + \varepsilon \mathbf{f}_N^1(\mathbf{v}_{AB}, \omega, \hat{\mathbf{u}}) \tag{2.4.15}$$

where $\varepsilon = 1$ and $\mathbf{f}_N^1$, representing parasitic effects, is such that $\mathbf{f}_N^1(\mathbf{v}_{AB}, \mathbf{0}, \mathbf{0}) = \mathbf{0}$. In (2.4.15) we know $\dot{v}_{N,2}$ and $\dot{v}_{N,3}$ and we also know ψ and $\mathbf{v}_{AR}$. Therefore, a function $\mathbf{h}^2$: $\mathbb{R}^3 \times \mathbb{R}^2 \times SO(2) \rightarrow SO(3)$ can be constructed so that the aircraft attitude given by

$$\mathbf{C}_c = \mathbf{h}^2(\mathbf{v}_{AR}, \mathbf{E}_2^N(\eta_c)\dot{\mathbf{v}}_{(N)c}, \mathbf{E}_3^R(\psi_c)) \tag{2.4.16}$$

results in the commanded acceleration, $\dot{\mathbf{v}}_{(N)c}$. Note that $\mathbf{C}_c$ is the aircraft attitude which is used to calculate $\mathbf{v}_{AB}$ from $\mathbf{v}_{AR}$; $\mathbf{v}_{AB}$ is then used in $\mathbf{f}_N^0$ of (2.4.15) to yield the commanded nozzle axes accelerations.

Equations (2.4.13) and (2.4.16) are the trim equations for the aircraft without the parasitic effects (i.e, $\varepsilon = 0$ in (2.3.15)). That is, for a given motion $[\mathbf{r}(t), \psi(t), \eta(t)]$, $t \geq 0$ and corresponding commanded moments and forces, the **trim** and **controls** are computed as follows:

$$\mathbf{r}_c = \mathbf{r}(t) \tag{2.4.17}$$

$$\mathbf{v}_c = \dot{\mathbf{r}}(t) \tag{2.4.18}$$

$$\mathbf{C}_c = \mathbf{h}^2(\mathbf{v}_{(AR)c}, \mathbf{E}_2^N(\eta_c)\dot{\mathbf{v}}_{(N)c}, \mathbf{E}_3^R(\psi_c(t))) \tag{2.4.19}$$

$$\omega_c = q(\dot{\mathbf{C}}_c(t)\mathbf{C}_c^T(t)) \tag{2.4.20}$$

$$\dot{\omega}_c = [\hat{u}_1, \hat{u}_2, \hat{u}_3]^T \tag{2.4.21}$$

$$\begin{bmatrix}\mathbf{u}^M \\ p_1^1\end{bmatrix} = \mathbf{h}^1(\mathbf{C}_c\mathbf{v}_{(AR)c}, \omega_c, \hat{\mathbf{u}}) \tag{2.4.22}$$

where the function $q(\cdot)$ extracts ω from $\mathbf{S}(\omega) = \dot{\mathbf{C}}\mathbf{C}^T$. The time derivatives required in the above calculations are computable because of the selection of the Brunovsky canonical model for the Harrier (see Fig. 2.3.3) and the model law generation process as described at the end of section 2.3.

The transformation which changes the natural representation of the Harrier to the Brunovsky canonical representation is a follows. The coordinate change $\mathbf{T}(\mathbf{x})$ is given by the following equations:

$$[y_1, y_5, y_9]^T = \mathbf{r} \tag{2.4.23}$$

$$[y_2, y_6, y_{10}]^T = \mathbf{v} \tag{2.4.24}$$

$$[y_3, y_7, y_{11}, y_{13}]^T = [(\dot{v}_{N,1}, f_{N,1}^0, f_{N,2}^0)\mathbf{C}_{\mathbf{BN}}{}^T\mathbf{C}_{\mathbf{RB}}{}^T, \mathbf{E}_3^R(\psi)]^T \tag{2.4.25}$$

$$[y_4, y_8, y_{12}, y_{14}]^T = [(\frac{\partial}{\partial \mathbf{C}}\mathbf{C}_{\mathbf{RB}}\mathbf{C}_{\mathbf{BN}}(\dot{v}_{N,1}, f_{N,1}^0, f_{N,2}^0)^T\omega)^T, \dot{\omega}_3]^T \tag{2.4.26}$$

$$[y_{15}, y_{16}]^T = [\mathbf{E}_2^R(\theta), \dot{\omega}_2]^T \tag{2.4.27}$$

The control variable change, $\mathbf{u} = \mathbf{W}(\mathbf{y}, \mathbf{w})$, is given in two steps:

Step 1: determine $\dot{\omega}$ and $\dot{v}_{N,1}$ by

$$\dot{\omega} = (\frac{\partial}{\partial \mathbf{C}}\mathbf{C}_{\mathbf{RB}}\mathbf{C}_{\mathbf{BN}}(\dot{v}_{N,1}, f_{N,1}^0, f_{N,2}^0)^T)^{-1}[w_1, w_2, w_3]^T \tag{2.4.28}$$

$$\dot{v}_{N,1} = (\mathbf{C}_{NB}\mathbf{C}_{BR}[y_3, y_7, y_{11}]^T)_1 \tag{2.4.29}$$

Step 2: calculate $\mathbf{u}^M$ and p_1^1 by

$$\begin{bmatrix}\mathbf{u}^M \\ p_1^1\end{bmatrix} = \mathbf{h}^1(\mathbf{v}_{AB}, \omega, \hat{\mathbf{u}}). \tag{2.4.30}$$

Note that the effects of the various approximations made in the contruction of these transformations are relegated to the regulator.

§ 2.5 *Linear Regulator*, $K_0(e_y, w_0)$

As the details of the regulator block of Fig. 2.1.1 for the Harrier control system design are quite similar to those for the helicopter control system design considered in [MSH82], [HSMmi], [MSH83G], and [MSH83C] and as these details add little useful background supporting the balance of the report, they are omitted.

The general contruction of the regulator follows that of the decoupled strings of the Brunovsky canonical form. That is, the error in each channel of the Brunovsky form of the Harrier is regulated independently. For each channel, the commanded canonical states are compared with the current estimates as computed from the T-map given by (2.4.23) through (2.4.27). The sum of these state errors, appropriately weighted, is taken as the correction term for the Brunovsky control variable for that channel. In this way, the total regulator output δw is constructed and added to the open-loop command w_c, resulting in the total canonical control w (see Fig. 2.1.1). The control w is then transformed by means of the W-map given by (2.4.28) through (2.4.30) into the natural control u which, in turn, drives the actual plant.

CHAPTER 3 — Nonlinear Modeling of Force and Moment Equations

§ 3.1 *Introduction*

In this chapter, the techniques used to determine a smooth nonlinear model of the Harrier simulation model are presented. The force and moment equations (i.e., (2.2.10) and (2.2.11)) are modeled as four dimensional tensor product spline functions of the roll, pitch, yaw commands, and throttle position, parameterized by wind-adjusted body coordinates of translational velocity, body coordinates of angular velocity, and nozzle angle. That is, for nominal values of v_{AB}, ω, and p_2^1, (2.2.10), expressed in the nozzle coordinate frame, is interpolated with a three dimensional vector of tensor product spline functions denoted by $\mathbf{m}_0^{\mathbf{F}}(\mathbf{u}^{\mathbf{M}}, p_1^1)$ and (2.2.11) is interpolated with a three dimensional vector of tensor spline product functions denoted by $\mathbf{m}_0^{\mathbf{M}}(\mathbf{u}^{\mathbf{M}}, p_1^1)$. Each of the six tensor product spline functions is a four dimensional interpolant of spline order three (i.e., polynomial order two) in each dimension.

In the subsequent sections of this chapter, the theory of spline representations of piecewise polynomial functions as well as the application of this theory to the approximation of functions of several variables via tensor products is discussed. The software which implements the tensor product function modeling is then discussed and the modeling of (2.2.10) and (2.2.11) is presented in detail along with a comparative view of this tensor product spline model to the full simulation model.

§ 3.2 *Tensor Product Spline Functions*

Before presenting the formal definition of tensor product spline functions, several fundamental notions need to be introduced [deBr78],[deBr79].

Spline functions are representations of piecewise polynomial functions by linear combinations of basis spline functions. For the formal definition of a piecewise polynomial (pp) function, consider the strictly increasing finite sequence of points $\zeta := (\zeta_i)_1^{l+1} \subset \mathbb{R}$ and let $k \in \mathbf{N}$. If $p_1, p_2, \ldots, p_l$ is any sequence of l polynomials, each of order k (i.e., degree $\leq k-1$), then a **piecewise polynomial (pp)** function f of **order k** is any possibly discontinuous function or efined by

$$f(x) := p_i(x) \text{ if } \zeta_i < x < \zeta_{i+1}, \quad i = 1, 2, \ldots, l. \tag{3.2.1}$$

The ζ_i are called **breakpoints** and often the function f is extended such that

$$f(x) = \begin{cases} p_1(x) \text{ if } x \le \zeta_1 \\ p_l(x) \text{ if } \zeta_{l+1} \le x \end{cases} \tag{3.2.2}$$

EXAMPLE (3.2.1) - discontinuous pp function

$l = 3, k = 4, \zeta = (0, 1.0, 5.0, 10.0), p_1 = x^3+x+6, p_2 = x^2-2$, and $p_3 = 3x+1$, so

$$f(x) = \begin{cases} x^3+x+6 & 0 < x < 1 \\ x^2-2 & 1 < x < 5 \\ 3x+1 & 5 < x < 10 \end{cases}$$

Note that $p_1(1) \neq p_2(1)$ and $p_2(5) \neq p_3(5)$, so $f(x)$ is not continuous at $x = 1$ and $x = 5$.

□

EXAMPLE (3.2.2) - continuous pp function

$l = 4, k = 3, \zeta = (0, 1.0, 2.0, 3.0, 4.0), p_1 = x^2+x-6, p_2 = x^2-5x, p_3 = -3x$, and $p_4 = x^2-18$, so

$$f(x) = \begin{cases} x^2+x-6 & 0 < x < 1 \\ x^2-5x & 1 < x < 2 \\ -3x & 2 < x < 3 \\ x^2-18 & 3 < x < 4 \end{cases}$$

Note that $p_1(1) = p_2(1)$, $p_2(2) = p_3(2)$, and $p_3(3) = p_4(3)$, so $f(x)$ is continuous.

□

Note that the linear space of pp functions of order k with breakpoint sequence ζ is denoted by $\mathbb{P}_{k,\zeta}$. The linear subspace of $\mathbb{P}_{k,\zeta}$ consisting of those elements which satisfy continuity conditions specified by the finite nonnegative integer sequence ν is denoted by $\mathbb{P}_{k,\zeta,\nu}$. Note that the i^{th} element of the sequence ν is the required continuity condition at ζ_i. For $f \in \mathbb{P}_{k,\zeta,\nu}$, $\nu_i = j$ means that at breakpoint ζ_i, $f^{(n-1)}$ is continuous for $n = 1, \ldots, j$.

For the definition of the **kth divided difference** of a pp function f at the points $\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}$, consider the polynomial p of order $k+1$ which agrees with f at the points $\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}$. The word "agrees" in the above definition has the following interpretation: for the finite sequence of points $\tau = \tau_i, \tau_{i+1}, \ldots, \tau_{i+k}$, the polynomial p is said to agree with function f on these points if there is *m_i–fold*

agreement at each of the points τ_i , where m_i is the number of times the number τ_i appears in the finite sequence τ and where m_i–*fold* agreement at τ_i is given by the condition

$$p^{(j-1)}(\tau_i) = f^{(j-1)}(\tau_i) \text{ for } j = 1, \ldots, m_i. \tag{3.2.3}$$

The k^{th} divided difference of f at points $\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}$ is the leading coefficient of p (i.e., the coefficient of x^k) and is denoted by

$$[\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}]f. \tag{3.2.4}$$

EXAMPLE (3.2.3) - divided difference calculation

Consider the piecewise polynomial $f(x)$ from EXAMPLE (3.2.1) and let $i = 1$, $k = 2$, $\tau_1 = (2.0)^{1/2}$, and $\tau_2 = \tau_3 = 6.0$ so that $f(\tau_1) = 0$, $f(\tau_2) = 19$, and $f^{(1)}(\tau_2) = 3$. Denote $p(x)$ by $p(x) = a_2x^2 + a_1x + a_0$. The second divided difference of f at the points τ_1, τ_2, τ_3 is found by solving

$$\begin{bmatrix} 4 & 2 & 12 \\ 16 & 6 & 1 \\ 12 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 19 \\ 3 \end{bmatrix}$$

for a_2, a_1, a_0. Doing so yields $a_2 = -0.3684$ and, clearly, $[1, 2, 2]f = a_2$.

□

Consider next a nondecreasing sequence t of real numbers. For the following definition, t is said to be the **knot sequence**. The **i^{th} (normalized) basis spline** or **B–spline** of order k for knot sequence t is denoted by $B_{i,k,t}$ (often shortened to B_i) and is defined by the rule

$$B_{i,k,t}(x) := (t_{i+k} - t_i)[t_i, \ldots, t_{i+k}](t - x)_+^{k-1} \tag{3.2.5}$$

for all $x \in \mathbb{R}$. Note that $(t - x)_+$ is the notation for function truncation. That is,

$$(t - x)_+ := \max(0, t - x). \tag{3.2.6}$$

Thus, i^{th} order B-splines are viewed as appropriately scaled i^{th} divided differences of the truncated power function.

EXAMPLE (3.2.4) - B-spline calculation

Let $k = 3, i = 1, t_1 = 1, t_2 = 2, t_3 = 3, t_4 = 4$ so that the 1^{st} normalized B-spline for knot sequence $\mathbf{t} = (t_j)_1^4$

is

$$B_{1,3,t}(x) = (t_4 - t_1)[t_1, t_2, t_3, t_4](t - x)_+^2$$

For $x = 2$, we determine

$$B_{1,3,t}(2) = (3)[1, 2, 3, 4](t - x)_+^2$$

To do so, denote $(t - x)_+^2$ by $f(t)$ and let $p(t)$ denote $p(t) = a_3t^3 + a_2t^2 + a_1t + a_0$. Observe that $f(t_1) = f(t_2) = 0$, $f(t_3) = 1$, and $f(t_4) = 4$. Evaluating $p(t)$ at t_1, t_2, t_3, t_4 yields the following system of linear algebraic equations in a_3, a_2, a_1, a_0

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \\ 27 & 9 & 3 & 1 \\ 64 & 16 & 4 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 4 \end{bmatrix}$$

Solving the system of equations yields $a_3 = 0.0811$ and, therefore, $[1, 2, 3, 4](t - 2)_+^2 = 0.0811$. Thus, $B_{1,3,t}(2) = (3)(0.0811) = 0.2342$. Making this calculation for all $x \in \mathbb{R}$ specifies $B_{1,3,t}(x)$.

□

Every space $\mathbb{P}_{k,\zeta,\nu}$ has a basis consisting of such B-splines. The Curry-Schoenberg theorem gives conditions on the knot sequence t in terms of the breakpoint sequence ζ and the continuity condition sequence ν which guarantee the existence of n B-splines of order k for knot sequence t which form a basis for the *n–dimensional* space $\mathbb{P}_{k,\zeta,\nu}$.

Curry – Schoenberg Theorem:

For a given strictly increasing sequence $\zeta := (\zeta_i)_1^{l+1}$ and a given non-negative integer sequence $\nu := (\nu_i)_2^l$ with $\nu_i \le k$ for all i, set

$$n := k + \sum_{i=2}^{l}(k - \nu_i) = kl - \sum_{i=2}^{l}\nu_i = dim\, \mathbb{P}_{k,\zeta,\nu} \tag{3.2.7}$$

and let $\mathbf{t} := (t_i)_1^{n+k}$ be any nondecreasing sequence so that

(i) $t_1 \le t_2 \le \ldots \le t_k \le \zeta_1$ and $\zeta_{l+1} \le t_{n+1} \le \ldots \le t_{n+k}$;

(ii) for $i = 2, \ldots, l$, the number ζ_i occurs exactly $k - \nu_i$ times in t.

Then, the sequence $B_{1,k,t}, B_{2,k,t}, \ldots, B_{n,k,t}$ of B-splines of order k for the knot sequence t is a basis for $\mathbb{P}_{k,\zeta,\nu}$ on the interval $[t_k, t_{n+1}]$.

Comments

• Note that the dimensionality of $\mathbb{P}_{k,\zeta,\nu}$ is specified in terms of k and ν.

• Note also that the number of continuity conditions at breakpoint ζ_i plus the number of knots placed at ζ_i must equal k, the B-spline order. Clearly, then, fewer knots at a breakpoint corresponds to greater smoothness there (i.e., more continuity).

• The finite sequence ν is extended to include endpoint continuity conditions (ν_1 and ν_{l+1}) which are identically zero. This is consistent with the fact that the B-spline basis provides a valid representation for elements of $\mathbb{P}_{k,\zeta,\nu}$ only on the interval $[t_k, t_{n+1}]$.

A **spline function** of order k with knot sequence t is any linear combination of B-splines of order k for knot sequence t. The collection of all such functions is denoted by $\$_{k,t}$. In symbols,

$$\$_{k,t} := \left\{ \Sigma_i \, \alpha_i B_{i,k,t} \mid \alpha_i \in \mathbb{R} \text{ for } \textit{all } i \in \mathbb{N} \right\}. \tag{3.2.8}$$

The **B–representation** for function $f \in \mathbb{P}_{k,\zeta,\nu}$ is a spline function representation for f which consists of

(i) the integers k and n, respectively giving the order of f (as a piecewise polynomial function) and the number of linear parameters (i.e., $n = kl - \sum_{i=0}^{l+1} \nu_i = \textit{dim}\, \mathbb{P}_{k,\zeta,\nu}$ where $\nu_1 = \nu_{l+1} = 0$ meaning no continuity conditions at the end points);

(ii) the sequence $\mathrm{t} = (t_i)_1^{n+k}$ containing the knots (possibly partially coincident and constructed from ζ and ν as in the Curry-Schoenberg theorem) in increasing order;

(iii) the vector $\alpha = (\alpha_i)_1^n$ of coefficients of f with respect to the B-splines basis $(B_i)_1^n$ for $\mathbb{P}_{k,\zeta,\nu}$ on knot sequence t.

The B-representation for $f(\cdot)$ is denoted by $m(\cdot)$. Thus, the value of f at a point $x \in [t_k, t_{n+1}]$ as given by its B-representation is

$$m(x) = \sum_{i=1}^{n} \alpha_i B_i(x). \tag{3.2.9}$$

The α_i are determined from the n linearly independent equations which result from knowledge of the value of f at n distinct values of x. That is, given a set of n distinct data points for f, (3.2.9) generates n linearly independent equations in the α_i (note that the B-spline functions B_i are known and are determined in accordance with the Curry-Schoenberg theorem).

EXAMPLE (3.2.5) - B-representation for EXAMPLE (3.2.2)

From EXAMPLE (3.2.2), $l = 4$, $k = 3$, $\zeta = (0, 1, 2, 3, 4)$ and $\nu = (0, 1, 1, 1, 0)$. Therefore, $n = (4)(3) - (3) = 9$ and $m(x) = \sum_{i=1}^{9} \alpha_i B_i(x)$. The knot sequence is $n + k = 12$ long. The knots t_1, t_2, t_3 are placed at the endpoint ζ_1, knots t_3, t_4 are placed at ζ_3, knots t_5, t_6 are placed at ζ_3, knots t_7, t_8 are placed at ζ_4, and knots t_{10}, t_{11}, t_{12} are placed at the endpoint ζ_5. For $x = x_i$, for $i = 1, 2, \ldots, n$, make the following calculations

$$B_j(x_i) = (t_{i+k} - t_i)[t_1, t_2, t_3, t_4](t - x_i)_+^2 \text{ for } j = 1, 2, \ldots, 9$$

Evaluating $f(x)$ for $x = x_i$ yields the following system of nine linear algebraic equations in the nine unknowns, $\alpha_1, \alpha_2, \ldots, \alpha_9$

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \cdot \\ \cdot \\ \cdot \\ f(x_9) \end{bmatrix} = \mathbf{B} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \cdot \\ \alpha_9 \end{bmatrix}$$

where

$$\mathbf{B} = \begin{bmatrix} B_1(x_1) & B_2(x_1) & \ldots & B_9(x_1) \\ B_1(x_2) & B_2(x_2) & \ldots & B_9(x_2) \\ \cdot & \cdot & \ldots & \cdot \\ \cdot & \cdot & \ldots & \cdot \\ \cdot & \cdot & \ldots & \cdot \\ B_1(x_9) & B_2(x_9) & \ldots & B_9(x_9) \end{bmatrix}$$

Solving for the α_i completes the construction of the B-representation for $f(x)$.

□

Extension of B-representation to multivariate case

Consider U, a linear space of functions defined on some set X_1 into $\mathbb{R}$, and V, a linear space of functions defined on some set X_2 into $\mathbb{R}$. For each $u \in U$ and $v \in V$, define $w(\cdot,\cdot)$ to be the **tensor product of u with v** where

$$w(x_1, x_2) := u(x_1)v(x_2), \ \forall (x_1, x_2) \in X_1 \times X_2. \tag{3.2.10}$$

The tensor product of U with V is the set of all finite linear combinations of functions $w(\cdot,\cdot)$ on $X_1 \times X_2$ for some $u \in U$ and for some $v \in V$.

The extension of the B-representation to scalar-valued functions of several variables is made by considering finite linear combinations of the tensor product of B-spline functions of single variables. For example, the representation of a two-dimensional function $f(x_1, x_2)$ is by $m(x_1, x_2)$ where

$$m(x_1, x_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \alpha_{ij} U_i(x_1) V_j(x_2) \tag{3.2.11}$$

with $U_i(\cdot)$ and $V_j(\cdot)$ one-dimensional B-spline functions on the knot sequences $\mathbf{t}^1 := (t_i^1)_1^{n_1+k_1}$ and $\mathbf{t}^2 := (t_i^2)_1^{n_2+k_2}$, respectively.

Comments

• Note that (3.2.11) holds for $(x_1, x_2) \in \tau_1 \times \tau_2$ where $\tau_1 := [t_{k_1}, t_{n_1+1}]$ and where $\tau_2 := [t_{k_2}, t_{n_2+1}]$.

• The dimensionality and order of the tensor product spline (3.2.11) in x_1 need not agree with that in x_2. That is, in general, $n_1 \neq n_2$ and $k_1 \neq k_2$.

• In order to completely specify (3.2.11), the $n_1 \times n_2$ unknowns α_{ij} need to be found. The necessary $n_1 \times n_2$ linearly indepedent equations which yield the α_{ij} are specified by a set of $n_1 \times n_2$ distinct data points for f where for each of n_1 distinct x_1 values, $f(x_1, \cdot)$ is evaluated at n_2 distinct values of x_2. That is, a grid of $n_1 \times n_2$ data points of f is constructed for a set of ordered pairs (x_1, x_2) for n_1 distinct values of x_1 by a set of n_2 distinct values of x_2.

• It is instructive to observe that (3.2.11) for fixed x_1 reduces to (3.2.9) where $x = x_2$ and for fixed x_2, (3.2.11) reduces to (3.2.9) where $x = x_1$.

Extensions of the above to higher dimensions follow *mutatis mutandis*. Note that four dimensional tensor product splines are used to model Harrier force and moment equations. That is, functions of the form

$$m(x_1, x_2, x_3, x_4) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3}\sum_{l=1}^{n_4}\alpha_{ijkl}U_i^1(x_1)U_j^2(x_2)U_k^3(x_3)U_l^4(x_4) \qquad (3.2.12)$$

are used to build smooth nonlinear models for the Harrier force and moment equations.

Comments

• It should be clear to the reader that the functions $U_i^1(\cdot)$, $U_j^2(\cdot)$, $U_k^3(\cdot)$, and $U_l^4(\cdot)$ are one dimensional B-splines defined on the knot sequences $\mathbf{t}^1 := (t_m^1)_1^{n_1+k_1}$, $\mathbf{t}^2 := (t_m^2)_1^{n_2+k_2}$, $\mathbf{t}^3 := (t_m^3)_1^{n_3+k_3}$, and $\mathbf{t}^4 := (t_m^4)_1^{n_4+k_4}$, respectively.

• The dimensionality and order of the tensor product spline (3.2.12) in x_i need not agree with that in x_j for $i \neq j$ for $i, j = 1, 2, 3, 4$. That is, in general, $n_i \neq n_j$ and $k_i \neq k_j$ for $i, j = 1, 2, 3, 4$.

• The complete specification of (3.2.12) requires the determination of the $n_1 \times n_2 \times n_3 \times n_4$ B-spline coefficients, α_{ijkl}. A grid of data points for f is constructed for a set of n_1 distinct values of x_1 by n_2 distinct values of x_2 by n_3 distinct values of x_3 by n_4 distinct values of x_4. In this way, $n_1 \times n_2 \times n_3 \times n_4$ linearly independent equations in α_{ijkl} are specified and determination of the values of α_{ijkl} follows.

• Note that for fixed x_i, x_j, x_k, (3.2.12) reduces to (3.2.9) with $x = x_l$ where $i \neq j \neq k \neq l$ for $i, j, k, l = 1, 2, 3, 4$. That is, freezing three of the variables in (3.2.12) reduces it to a one-dimensional B-spline representation.

§ 3.3 *Tensor Product Spline Software*

FORTRAN subroutines for spline and tensor spline calculations exist and are contained in the BSPLINE and TENSORBS libraries. The code used to determine multidimensional tensor product spline interpolant functions and to evaluate these interpolants and/or existing partial derivatives of the interpolants was written by William Nye [Nye86]. Nye's program is a generalization of the two-dimensional and three-dimensional versions written by Ronald Boisvert [Bois82] which were, in turn, the result of

Boisvert's re-vision and extension of code written by Carl deBoor [deBr78]. The code is in the public domain and is, therefore, readily available.

In this section, the structure and purpose of the subroutines used in multidimensional tensor product interpolation and evaluation are given as a prelude to discussing the code used to generate the relevant Harrier models. The routines of interest are BNINK, BNFAC, BTPCF, BKNOT, BNSLV, and BNVAE. Note that in the paragraphs which follow, the symbol "*" is used to indicate scalar multiplication as is the FORTRAN convention and FORTRAN-like notation is used when refering to variables and arrays used in the relevant subroutines. Further, for the sake of illustration, FORTRAN pseudo-code is given; that is, expressions which are approximately FORTRAN statements are made so as to better explain the operation of the subroutines of interest.

The subroutine BNINK determines the n-dimensional k-order tensor spline interpolant function for the given grid of data. Note that in the name "BNINK" B stands for B-representation; N, the dimensionality of the function; and INK, for k-th order interpolant. The I/O (i.e, input/output) listing for BNINK is as follows

BNINK(N, X, NX, FCN, KX, TX, BCOEF, WORK, IFLAG)

where the input/output variables are

N: input integer scalar representing the number of dimensions;

X: input real one-dimensional array of size NX(1)+NX(2)+...+NX(N) containing adjacent vectors of x-abcissae;

NX: input integer array of size N with each entry greater than or equal to 3 and with NX(m) being the number of abcissae in the x_m direction;

FCN: input real one-dimensional array of size NX(1)*NX(2)*...*NX(N) containing adjacent cross-product row-major-order function values to interpolate; for example, consider the four-dimensional case where the function values at $(x_1(i), x_2(j), x_3(k), x_4(l))$ are stored in a four dimensional array VAL(i, j, k, l) then the following pseudo-code describes how the one-dimensional array FCN is loaded:

```
M = 0
DO L = 1,4
      DO K = 1,4
            DO J = 1,4
                  DO I = 1,4
                  M = M + 1
                  FCN(M) = VAL(I,J,K,L)
                  END DO
            END DO
      END DO
END DO
```

KX: input integer array of size N where KX(m) is the order of the spline pieces in the x_m direction and where KX(m) must be at least 2 and is less than NX(m);

TX: input/output real one-dimensional array of size NX(1)+KX(1)+...+NX(m)+KX(m) containing the knot sequences for each direction, where the NX(m)+KX(m) long knot sequence for direction x_m is loaded in TX starting at TX(1+(NX(1)+KX(1)+...+NX(m-1)+KX(m-1)); note that if IFLAG is set to 0 on input, the knot sequences are chosen by BNINK as given by BKNOT while an input IFLAG of 1 means that the user has supplied as input the knot sequences in TX;

BCOEF: output real one-dimesional array of size NX(1)*NX(2)*...*NX(N) of adjacent cross-product row-major-order coefficients of the B-spline interpolant; this may be the same array as FCN.

Note that WORK in the I/O description for BNINK represents an assortment of WORK arrays needed by BNINK for storage and note that the integer IFLAG, on output, provides a status indication for the just completed execution of BNINK.

BNINK makes calls to the subroutines BTPCF and BKNOT. The I/O statement for these two routines are

BTPCF(X, N, FCN, LDF, NF, T, K, BCOEF, WORK)

BKNOT(X, N, K, T).

The subroutine BKNOT is used by BNINK to choice the knot sequences to be used in the interpolant determination process. BKNOT is called N times by BNINK, once for each direction x_m. Note that this subroutine is not called if the user supplies the knot sequence to BNINK in array TX. In each direction x_m, BKNOT places K (spline order in the x_m direction) knots at each endpoint as is consistent with the requirement that there be no continuity conditions at the endpoints. The remaining knots are placed at data points

if N (number of of data points in the x_m direction) is even and between data points if N is odd. The knots are stored in the array T. Note that because the routine used to evaluate the spline interpolant does so by approaching the point from the right, the right-most knot is shifted slightly to the right to insure proper interpolation at X(N).

The subroutine **BTPCF** computes B-spline interpolation coefficients for NF sets of data stored in the array FCN and stores the coefficients in the array BCOEF. Each interpolation is based on the N abcissae stored in the array X and the N+K knots stored in the array K. That is, the user of BTPCF supplies NF sets of data points, each set N elements long, to BTPCF. BTPCF then calculates the B-spline coefficients for each set of data points; that is, BTPCF fits NF B-spline interpolants as functions of X. This amounts to solving NF equations of the form

$$\mathbf{A}\mathbf{x}_i = \mathbf{b}_i \tag{3.3.1}$$

where A is a square matrix of dimension N with each element of A the appropriately evaluated B-spline tensor product funtion, $\mathbf{x}_i$ is an N-long vector of unknown B-spline interpolant coefficients to be solved for, and $\mathbf{b}_i$ is the right-hand-side N-long vector as given in FCN for $i = 1, 2, \ldots,$ NF. Note that BTPCF solves these NF systems of N equations in a computationally efficient manor by making use of the LU factorization of the matrix A supplied by the subroutine BNFAC. In the LU factorization, L is a lower triangular matrix with main diagonal entries all 1 and with entries l_{ij} below the diagonal equal to the multiplier of row j which is subtracted from row i during the Gaussian elimination process. The matrix U is the coeffienct matrix which appears after elimination and before back-substitution. For each $\mathbf{b}_i$, one need only solve the system given by $\mathbf{L}\mathbf{c}_i = \mathbf{b}_i$ for $\mathbf{c}_i$ by forward-substitution then solve the system $\mathbf{U}\mathbf{x}_i = \mathbf{c}_i$ for $\mathbf{x}_i$ by backward substitution. This operation requires N^2 calculations while starting from the non-factorized A would require $\frac{N^3}{3}$ calculations.

Efficient calculation of B-spline coefficients

BNINK calls BTPCF N times, each time interchanging the role of the FCN and BCOEF arrays. For example, the pseudo-code for the determination of the B-spline interpolant coefficients for a four-dimensional function is

```
CALL BTPCF(X, NX, FNC, NX(1), NX(2)*NX(3)*NX(4), TX, KX, BCOEF, WORK)
CALL BTPCF(X, NX, BCOEF, NX(2), NX(1)*NX(3)*NX(4), TX, KX, FNC, WORK)
CALL BTPCF(X, NX, FNC, NX(3), NX(2)*NX(1)*NX(4), TX, KX, BCOEF, WORK)
CALL BTPCF(X, NX, BCOEF, NX(4), NX(2)*NX(3)*NX(1), TX, KX, FNC, WORK)
DO J = 1, NX(1)*NX(2)*NX(3)*NX(4)
     BCOEF(J) = FNC(J)
END DO
```

This is in accordance with de Boor's method for computational efficiency in the manipulation of tensor products. This is the main result in [deBr79] and the interested reader is directed to this reference for an inductive proof of the generalization of the above pseudo-code.

The subroutine **BNVAE** evaluates the tensor product B-spline interpolant constructed by the routine BNINK or one of the partial derivatives of the interpolant at a specified point. Note that BNVAE performs linear extrapolation for points outside the region where the B-spline interpolant is defined. In the name "BNVAE," the B stands for B-representation, N for dimensionality of the tensor product, VA for interpolant value, and E for the linear extrapolation feature. BNVAE has the I/O statement

BNVAE(N, XVAL, IDX, TX, NX, KX, BCOEF, WORK)

where the integer N and the arrays TX, NX, KX, and BCOEF must be unchanged from the last call to BNINK. XVAL is a real one-dimensional array of size N where XVAL(m) is the m-th coordinate of the evaluation point. IDX is an integer array of size N where IDX(m) is the derivative number of the interpolant to evaluate along the direction x_m. For example, in the four-dimensional case, IDX(1)=IDX(2)=IDX(3)=IDX(4)=1 means evaluate the first derivative of the interpolant in each direction. BNVAE constructs the tensor spline function from the information in NX, KX, and BCOEF. In the four dimensional case, for example, it constructs equation (3.2.12).

The pseudo-code for modeling the Harrier force and moments equations as given by (2.2.10) and (2.2.11) is as follows. We make the following comments before presenting the pseudo-code.

Comments

• The roll, pitch, and yaw commands (the pseudo-code variables X1, X2, and X3, respectively) are normalized, taking on values in the interval [−1, 1]. The throttle position (X4) is normalized to the interval [0, 1].

• Let X# denote either X1, X2, X3, or X4. There are four abcissae in each array X# and they are spaced evenly oven the appropriate interval. This corresponds to a square four-dimensional grid of 256 4-tuples (x_1, x_2, x_3, x_4).

• The order of spline in each direction is 3 (i.e., KX(m)=3 for m=1, 2, 3, 4). Therefore, there are 7 knots in each direction, 3 at each endpoint, meaning one of the interior data points has one knot (continuity condition of 2) and the other has no knots (continuity condition of 3). We are interested in at least a continuity condition of 2 at all interior breakpoints while using as few data points as possible to minimize the number of function calls. For the k=3 case, this requires that n=4.

• The subroutine SIMOD of the pseudo-code below represents YBNTRU of the true code. It returns in TEMP the forces (in g's) and the moments (in $\frac{rad}{s^2}$) for given v_{AB} (VAB), ω (OMEGA), p_2^1 (NOZ), X1, X2, X3, and X4.

• The integer NMOD is set to 1 intially for the generation of the initial tensor spline model and reset to 1 every twenty computer cycles (i.e, every second) so as to obtain a new tensor spline model. For the computer cycles between model update, NMOD is set equal to 2 and a correction term Y2 - Y1 is added to the value given by the current tensor spline model, Y. This scheme is justifiable because the velocities, both angular and translational, and the nozzle position do not change all that much in a 1 second interval.

```
C create data arrays
C
      DO I=1,4
         X1(I) = -1.0 + REAL(I-1)*(2./3.)
         X2(I) = X1(I)
         X3(I) = X2(I)
         X4(I) = 0.0 + REAL(I-1)*(1./3.)
      END DO
C determine spline interpolants
C
      M = 0
        IF (NMOD.NE.1) GOTO 99
           DO I=1,3
              SAVE(I)=VAB(I)
              SAVE(I+3)=OMEGA(I)
           END DO
              SAVE(7)=NOZ
           DO L=1,4
              DO K=1,4
                 DO J=1,4
                    DO I=1,4
                       M=M+1
           CALL SIMOD(TEMP,VAB,OMEGA,NOZ,X1(I),X2(J),X3(K),X4(L))
                    FCN1(M)=TEMP(1)
                    FCN2(M)=TEMP(2)
                    FCN3(M)=TEMP(3)
                    FCN4(M)=TEMP(4)
                    FCN5(M)=TEMP(5)
                    FCN6(M)=TEMP(6)
                    END DO
                 END DO
              END DO
              X(L)=X1(L)
              X(L+4)=X2(L)
              X(L+8)=X3(L)
              X(L+12)=X4(L)
              NX(L)=4
              KX(L)=3
           END DO
           N=4
           IFLAG=0
           CALL BNINK(N,X,NX,FCN1,KX,TX,BCOEF,WORK,IFLAG)
           CALL BNINK(N,X,NX,FCN2,KX,TX,BCOEF,WORK,IFLAG)
           CALL BNINK(N,X,NX,FCN3,KX,TX,BCOEF,WORK,IFLAG)
           CALL BNINK(N,X,NX,FCN4,KX,TX,BCOEF,WORK,IFLAG)
           CALL BNINK(N,X,NX,FCN5,KX,TX,BCOEF,WORK,IFLAG)
           CALL BNINK(N,X,NX,FCN6,KX,TX,BCOEF,WORK,IFLAG)
```

```
C calculate interpolant value at XVAL (NMOD.EQ.1)
            Y(1)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(2)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(3)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(4)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(5)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(6)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            GOTO 999

C calculate interpolant value at XVAL (NMOD.NE.1)
99  CONTINUE
            Y(1)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(2)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(3)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(4)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(5)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)
            Y(6)=BNVAE(N,XVAL,IDX,TX,NX,KX,BCOEF,WORK)

            DO I=1,3
               TEMPS1(I)=SAVE(I)
               TEMPS2(I)=SAVE(I+3)
            END DO
            TEMPS3=SAVE(7)
            CALL SIMOD(Y2,VAB,OMEGA,NOZ,XVAL(1),XVAL(2),XVAL(3),XVAL(4))
            CALL SIMOD(Y1,TEMPS1,TEMPS2,TEMPS3,XVAL(1),XVAL(2),XVAL(3),XVAL(4))
            DO I=1,6
               Y(I)=Y(I)+(Y2(I)-Y1(I))
            END DO
999 CONTINUE
```

Extensive analysis of the tensor spline model has been conducted. The tensor spline interpolants for many different values of v_{AB}, ω, and p_2^1 were tested for many different values of $\mathbf{u}^M$ and p_1^1. To be specific,

- body 1-axis component of v_{AB} was varied from 30 ft/s to 370 ft/s with 10 ft/s increments; the body 2-axis component, from 30 ft/s to 70 ft/s with 5 ft/s increments; the body 3-axis component, from 0 ft/s to 70 ft/s with 5 ft/s increments;
- the components of ω were varied from 0 rad/s to 1 rad/s with 0.025 increments;
- p_2^1 was varied over its full range of values [−5°, 120°] with 10° steps.

For each set of values v_{AB}, ω, and p_2^1, the controls $\mathbf{u}^M$ and p_1^1 were stepped through their respective intervals with increment size 0.1. Data generated included the simulation model values, the tensor spline model values, the absolute error between the two model values, and the relative errors. Typically, the relative error between the two model values remained between 1% - 5% with results as good as 0.25% and as bad

as 10%. For values too small for a relative error interpretation, the absolute error revealed accuracy up to the fourth decimal place.

CHAPTER 4 — Model Inversion Issues

§ 4.1 *Introduction*

For given time histories of the nozzle angle and yaw angle, the trim equations (2.4.17) through (2.4.22) determine the the controls $\mathbf{u}^{\mathbf{M}}$ and p_1^1 and aircraft attitude C for which the commanded forces and moments are achieved. The function $\mathbf{h}^1$ represents the inversion of the four functions $\mathbf{f}_{\mathbf{B}}^{\mathbf{M}}$ and $f_{N,1}^{F}$ with respect to the controls $\mathbf{u}^{\mathbf{M}}$ and p_1^1. The function $\mathbf{h}^2$ represents the inversion of $\mathbf{f}_{\mathbf{N}}^{\mathbf{9}}$ and $\mathbf{E}_2^N(\psi)$ with respect to the attitude C. Performing simultaneously the indicated inversions is equivalent to solving a system of seven nonlinear algebraic equations of the form $\mathbf{Y}(\mathbf{x}) = \mathbf{0}$ in seven unknowns $\mathbf{x}$. In this chapter, attention is focused on the method used to solve this system of equations as specified by their tensor spline interpolant.

This method is a modification of the Powell hybrid method. The Powell hybrid method represents a compromise between the Newton iteration and the method of steepest descent. Thus, in the subsequent sections, the Newton method and the method of steepest descent are discussed separately. Then, the Powell hybrid method is presented along with a discussion of the modifications made on the method as it is applied to the problem at hand. Also, an assesment of the method's performance on the Harrier tensor product spline model is given.

§ 4.2 *Newton Method* and *the Method of Steepest Descent*

Consider the system of n at least $\mathbf{C}^1$ nonlinear real-valued functions $f_1(\cdot), f_2(\cdot), \ldots, f_n(\cdot)$ in n variables $x_1, x_2, \ldots, x_n$. Formally, define the vector-valued function

$$\mathbf{f}(\mathbf{x}) := \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \cdot \\ \cdot \\ \cdot \\ f_n(\mathbf{x}) \end{bmatrix} \tag{4.2.1}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and $\mathbf{f}$: $\mathbb{R}^n \rightarrow \mathbb{R}^n$. We want to solve for the value(s) of $\mathbf{x}$ for which

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{4.2.2}$$

holds (i.e., the determination of the zeroes of the function f). When the function f is one-to-one and invertible, this is equivalent to determining $h(\cdot) = f^{-1}(\cdot)$ and evaluating it at 0.

Newton method

The Newton method is as follows. Given a current estimate of the solution, x_k, a Taylor expansion of f about x_k truncated to the first order yields

$$f_i(x) \sim f_i(x_k) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x_j - x_{k,j}) \quad \text{for } i = 1, 2, \ldots, n. \tag{4.2.3}$$

Adopting the standard notation J for the Jacobian matrix, define

$$J := \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{4.2.4}$$

Using the notation J_k to mean the value of the Jacobian at x_k, we can rewrite (4.2.3) as

$$f(x) \sim f(x_k) + J_k(x - x_k) =: \Theta_k(x). \tag{4.2.5}$$

In $\mathbb{R}^n \times \mathbb{R}^n$, the n-dimensional "surface" $\left\{(x, \Theta_k(x)) \mid x \in \mathbb{R}^n\right\}$ is the tangent "plane" at $(x_k, f(x_k))$ to the n-dimensional "surface" $\left\{(x, f(x)) \mid x \in \mathbb{R}^n\right\}$.

Consider x_{k+1}, the next estimate to x^* where x^* denotes the solution of $f(x) = 0$. Apply this condition to x_{k+1} (i.e, require $f(x_{k+1}) = 0$) and make the substitution of x_{k+1} for x in the right-hand-side of (4.2.5). Solving for $(x_{k+1} - x_k)$ assuming J_k nonsingular gives

$$\delta_k := (x_{k+1} - x_k) = -H_k f(x_k) \tag{4.2.6}$$

where $H_k = J_k^{-1}$. Equation (4.2.6) is said to be the **Newton step** and, with it, the next estimate of the

solution is given by the **Newton iteration**

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k \,. \tag{4.2.7}$$

The sequence $(\mathbf{x}_i)_0^\infty$ constructed according to the Newton step will converge **quadratically** to $\mathbf{x}^*$ provided the initial estimate $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}^*$. Note that quadratic convergence is also known as **root rate 2 convergence.** (The sequence $(\mathbf{x}_i)_0^\infty$ is said to **converge with root rate r** for $r \geq 1$ if there exist $M \in (0,\infty)$, $\varepsilon \in (0,1)$, and $i_o \in \mathbb{N}$ such that for all $i \geq i_o$, $\|\mathbf{x}_i - \mathbf{x}^*\| \leq M\varepsilon^{r^i}$.) Note that the Newton iteration fails to converge in many cases. What can be shown is that if $\mathbf{x}_0$ is sufficiently close to the root $\mathbf{x}^*$, the Newton method will generate a sequence of iterates which converge to $\mathbf{x}^*$ [Luen73]. So called "quasi-Newton" algorithms have been invented to improve convergence to $\mathbf{x}^*$ when $\mathbf{x}_0$ is not sufficiently close to $\mathbf{x}^*$. The Powell hybrid method is one such algorithm.

Method of steepest descent

Consider the problem

$$\min f(\mathbf{x}) \quad \text{for } \mathbf{x} \in \mathbb{R}^n \tag{4.2.8}$$

with $f : \mathbb{R}^n \to \mathbb{R}$ continuosly differentiable. Denote the gradient of $f(x)$ by $\nabla f(x)$. Consider the following standard result from optimization theory [Polak88]: If $\mathbf{x} \in \mathbb{R}^n$ is such that $\nabla f(\mathbf{x}) \neq \mathbf{0}$, then any vector $\mathbf{h} \in \mathbb{R}^n$ such that $\nabla f(\mathbf{x})^T \mathbf{h} < 0$ is a **descent direction** for $f(\cdot)$ at $\mathbf{x}$ (i.e., there exists a positive $\lambda \in \mathbb{R}$ such that $f(\mathbf{x} + \lambda\mathbf{h}) - f(\mathbf{x}) < 0$). Algorithms invented to solve (4.2.8) typically generate sequences $(\mathbf{x}_i)_0^\infty$ with the descent direction property; that is, $f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) < 0$. Note that the construction of sequences with the descent direction property is not sufficient to guarantee that the algorithm will converge to a minimum. Convergence proofs for optimization algorithms are often subtle, rooted in some fairly sophisticated real analysis. The interested reader is refered to [Luen73] and [Polak88] for a detailed treatment of this and other issues in engineering optimization.

Observe that $\mathbf{h} = -\nabla f(\mathbf{x})$ is a descent direction for $f(\cdot)$. This observation leads to the following formalization of the steepest descent method for solving (4.2.8). Given a current estimate $\mathbf{x}_k$ of the solution $\hat{\mathbf{x}}$ of (4.2.8), the **search direction** $\mathbf{h}_k := -\nabla f(\mathbf{x}_k)$ is computed. If $\nabla f(\mathbf{x}) = \mathbf{0}$, the algorithm quits, setting

$\hat{\mathbf{x}} = \mathbf{x}_k$. Otherwise, the **steplength** $\lambda_k = \lambda(\mathbf{x}_k) := \textit{argmin}\ f(\mathbf{x}_k + \lambda \mathbf{h}_k)$ is calculated and the next estimate of the solution to (4.2.8) is computed as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{h}_k \tag{4.2.9}$$

Note that the main objection to the method of steepest descent is that the computation of the steplength λ_k is almost as difficult a problem to solve as (4.2.8). In practice, as with the Powell hybrid method, an inexact line search is done to calculate λ_k^* via some implementable steplength rule. The inexact λ_k^* is used in place of λ_k in (4.2.9).

§ 4.3 *Powell Hybrid Method*

In the paragraphs which follow, first we present an outline of the method, omitting details not needed for a general understanding of how the algorithm works. Second we present the details of the various components of the algorithm.

Powell hybrid algorithm

Data

(1) the current estimate $\mathbf{x}_k$ of the solution to (4.2.2) as well as the corresponding function value $\mathbf{f}(\mathbf{x}_k)$;

(2) an approximation to the Jacobian matrix at $\mathbf{x}_k$, $\hat{\mathbf{J}}(\mathbf{x}_k)$;

(3) an approximation to the inverse of the Jacobian matrix at $\mathbf{x}_k$, $\hat{\mathbf{H}}(\mathbf{x}_k) := \hat{\mathbf{J}}^{-1}(\mathbf{x}_k)$;

(4) a $n \times n$ matrix Π of n directions in the space of variables, and an associated vector of integers, π; Π and π are used to store the history of previous iterations that is needed to meet out requirements of linear independence;

(5) a steplength parameter Δ_k;

(6) a solution tolerance parameter ε.

Step 0

Set $k = 0$.

Step 1

Compute the steplength parameter Δ_k. See subsection "Δ_k calculation" for details.

Step 2

Compute the correction vector δ_k subject to the condition that if $\|\delta_k\| \leq \Delta_k$ then compute δ_k by the Newton correction; else, compute Δ_k by the correction rule defined from the steepest descent algorithm as applied to test function $F(\cdot)$ where

$$F(\mathbf{x}) := \sum_{i=1}^{n} f_i(\mathbf{x})^2 \tag{4.3.1}$$

See subsection "δ_k calculation" for details.

Step 3

Compute the **Powell hybrid iteration** given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k \tag{4.3.2}$$

where δ_k is calculated as in Step 2 and δ_k also satisfies a "sufficiency of independence" condition defined below.

Step 4

If $f_l(\mathbf{x}_k) \leq \varepsilon$ for $l = 1, \ldots, n$, the algorithm stops; otherwise, replace k by $k + 1$ and go to Step 1.

□

General observations

• Note that δ_k represents a compromise between the Newton iteration and the method of steepest descent applied to the function $F(\cdot)$ where $F : \mathbb{R}^n \rightarrow \mathbb{R}$. The balance between the two methods is governed by the steplength parameter Δ_k. For Δ_k sufficiently large, δ_k is the pure Newton step. For Δ_k small, δ_k is exactly a multiple of the predicted gradient of $F(\mathbf{x})$ where δ_k is such that $F(\mathbf{x}_k + \delta_k) < F(\mathbf{x}_k)$.

• The correction vectors δ_k interpolate between the Newton and steepest descent corrections in a way that gives fast ultimate convergence.

• The Powell hybrid method proves to be a very good algorithm for calculations that involve searching in many variable space. For each iteration, typically one can manage with only one new calculation of the objective functions, $f_l(\mathbf{x}_k)$. As discussed subsequently, this is due to the clever use of the steplength parameter Δ_k, where, on each iteration, the correction vector δ_k is predicted subject to the condition that

$\|\delta_k\| \le \Delta_k$ and the steplength parameter Δ_k is adjusted automatically, according to the success of δ_k.

• Step 3 of the algorithm is to update $\mathbf{x}_k$ by setting $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$ provided that δ_k is sufficiently independent of the last $\min(n-1,k-1)$ δ_i steps, a condition which is checked by making use of the information in Π and π. Note that for $\mathbf{a}_i \in \mathbb{R}^n$ for $i = 1, 2, \ldots j$ with $j \le n$ and $\mathbf{b} \in \mathbb{R}^n$, **b** is said to be **sufficiently independent** or, simply, **s–i** of the set of $\mathbf{a}_i$ if the least angle between **b** and any vector in the subspace spanned by the vectors $\mathbf{a}_i$ is not less than thirty degrees. If the s-i condition is not satisfied, δ_k is adjusted until it is satisfied.

• Because of the s-i condition on the correction vectors δ_k the iterations of the algorithm do not include any search directions along lines in the space of the variables, so most iterations require only one function evaluation.

• With the calculation of δ_k, $\mathbf{f}(\mathbf{x}_k + \delta_k)$ is determined and Δ_k is updated to obtain Δ_{k+1}. The update of Δ_k is such that Δ_{k+1} is made as large as possible so as to correct by the Newton iteration as often as possible, hence resulting in fast ultimate convergence of the algorithm.

δ_k calculation

To ease the discussion, we define

(1) **Newton correction, γ_k** (see (4.2.6))

$$\gamma_k := -\mathbf{J}_k^{-1}\mathbf{f}(\mathbf{x}_k) \tag{4.3.3}$$

(2) **steepest descent direction, $\mathbf{g}_k$**

$$\mathbf{g}_k := -\mathbf{J}_k\mathbf{f}(\mathbf{x}_k) = -0.5\nabla F(\mathbf{x}_k) \tag{4.3.4}$$

The steplength parameter Δ_k is defined as a means of limiting the size of the correction vector δ_k in the sense that $\|\delta_k\| \le \Delta_k$ is required. The determination of δ_k is by the following three-step algorithm:

(1) if $\|\gamma_k\| \le \Delta_k$, then set $\delta_k = \gamma_k$ and the algortihm stops; if not, continue to (2);

(2) if $\mu\|\mathbf{g}_k\| \ge \Delta_k$ where $\mu = \dfrac{\|\mathbf{g}_k\|^2}{\|\mathbf{J}_k\mathbf{g}_k\|^2}$, then set $\delta_k = \dfrac{\Delta_k\mathbf{g}_k}{\|\mathbf{g}_k\|}$ and the algorithm stops; if not, con-

tinue to (3);

(3) set δ_k equal to a point on the straight line joining the points μg_k and γ_k; that is, determine the parameter $\chi \geq 0$ such that $\| (1-\chi)\mu g_k + \chi\gamma_k \| = \Delta_k$; solving for χ reveals

$$\chi = \frac{\Delta_k^2 - \| \mu g_k \|^2}{(\gamma_k - \mu g_k, \mu g_k) + [((\gamma_k, \mu g_k) - \Delta_k^2)^2 + (\| \gamma_k \|^2 - \Delta^2)(\Delta^2 - \| \mu g_k \|^2)]^{1/2}} \tag{4.3.5}$$

and then set $\delta_k = (1-\chi)\mu g_k + \chi\gamma_k$.

Note that in determining the correction vector δ_k, the function $\mathbf{f}(\cdot)$, its Jacobian, and the inverse of the Jacobian each need be calculated only once. So, in any one iteration of the Powell hybrid method, the cost in terms of function, Jacobian, and inverse Jacobian calculations is minimal. Note that if δ_k fails to satisfy the s-i condition, it is adjusted by perturbing its components until the s-i condition is satisfied while still satisfying the condition $\| \delta_k \| \leq \Delta_k$.

Δ_k calculation

The adjustment of the steplength parameter Δ_k is done so that Δ_{k+1} is made as large as possible subject to the condition that each approximation to the Jacobian matrix $\hat{\mathbf{J}}(\cdot)$ provides a good prediction of the difference of the function difference $\mathbf{f}(\mathbf{x}_k + \delta_k) - \mathbf{f}(\mathbf{x}_k)$. To determine Δ_{k+1}, it is assumed that both $\mathbf{f}(\mathbf{x}_k + \delta_k)$ and $F(\mathbf{x}_k + \delta_k)$ are known and that the approximations $(\phi_1, \ldots, \phi_n)$ and Φ of $f_1(\mathbf{x}_k + \delta_k), \ldots, f_n(\mathbf{x}_k + \delta_k)$ and $F(\mathbf{x}_k + \delta_k)$, respectively, have been calculated where

$$\phi_l := f_l(\mathbf{x}_k) + \sum_{j=1}^{n} \mathbf{J}_{lj}\delta_{k,j} - f_l(\mathbf{x}_k + \delta_k), \; l = 1, 2, \ldots, n \tag{4.3.6}$$

$$\Phi := \sum_{k=1}^{n} \phi_k^2 - F(\mathbf{x}_k + \delta_k). \tag{4.3.7}$$

The revision of Δ_k is based on just how well (4.3.6) and (4.3.7) approximate the indicated functions. The measure used to assess the quality of these approximations is given by the condition that if

$$F(\mathbf{x}_k + \delta_k) > F(\mathbf{x}) - 0.1(F(\mathbf{x}) - \Phi) \tag{4.3.8}$$

then $\Delta_{k+1} = \max(\frac{1}{2}\Delta_k, DSTEP)$ where DSTEP is a default steplength which depends on the accuracy of the computer. Otherwise, Δ_{k+1} is set equal to Δ_k (no change) or Δ_{k+1} is set to a value greater than Δ_k with the

motivation being that the larger the Δ_{k+1} the fewer iterations are typically needed.

The manner in which the update of Δ_k results in an increase of the steplength parameter is as follows. The value of ρ^2 which just causes (4.3.8) to fail is determined; that is, setting

$$\sum_{l=1}^{n}[|f_l(x_k+\delta_k)|+(\rho^2-1)|f_l(x_k+\delta_k)-\phi_l|]^2=F(x_k)-0.1(F(x_k)-\Phi) \tag{4.3.9}$$

and solving for ρ^2 reveals

$$\rho^2=1+\frac{DMULT}{SP+(SP^2+DMULT-SS)^{1/2}} \tag{4.3.10}$$

where

$$DMULT=F(x_k)-0.1(F(x)-\Phi)-F(x_k+\delta_k);$$

$$SP=\sum_{l=1}^{n}|f_l(x_k+\delta_k)(f_l(x_k+\delta_k)-\phi_l)|;$$

$$SS=\sum_{l=1}^{n}(f_l(x_k+\delta_k)-\phi_l)^2.$$

Δ_i is increased only after two calculations of ρ have been made, where ρ is the positive square root of ρ^2. This is done to limit how often Δ_i is increased and this step is to be viewed as a conservative fix for increasing Δ_i. Both calculations of ρ are made since the last reduction in the value of Δ_i. The lesser of the two ρ is selected and it is called $\hat{\rho}$. Then, Δ_{k+1} is determined as follows

$$\Delta_{k+1}=\min(\Lambda\Delta_k, DMAX) \tag{4.3.11}$$

where

$\Lambda=\min(2,\hat{\rho},\sigma)$ where $\sigma=1$ initially or after Δ_i has been decreased and, otherwise, $\sigma=\frac{\rho_-}{\Lambda_-}$ where ρ_- and Λ_- are the previous values of the variables $\hat{\rho}$ and Λ, respectively, and

DMAX is a generous estimate of the distance as measure by the Euclidean norm of the solution from the intial guess.

Calculation of Jacobian and inverse Jacobian matrices

The modification to the Powell hybrid method as detailed in [MJDP70] is in the calcultion of Jacobian matrix. Where applicable, we make use of BNVAE's ability to calculate partial derivatives of the interpolant function (with respect to the defining variables) to determine entries in the Jacobian matrix. That is, if the function corresponding to the entry j_{il} of $\mathbf{J}_k$ of the Jacobian matrix to be approximated is modeled by a tensor spline interpolant, then j_{il} is given by

$$j_{il} \sim \text{BNVAE(N,XVAL,IDX,TX,NX,BCOEF,WORK)}$$

where TX, KX, NX, BCOEF are unchanged from the last call to BNINK and where KX(m)=0 for all m except for $m = l$, where it is set to 1. The reader should refer to the discourse in Chapter 3 of this report for details on the FORTRAN function BVNAE. Where BNVAE is not applicable, the Jacobian entries are calculated by a **forward–difference** approximation. That is, if we are interested in the value of the $(il)^{th}$ entry j_{il} of the Jacobian at $\mathbf{x}_k$, $\mathbf{J}_k$, approximate j_{ij} by

$$j_{il} \sim \frac{f_i(x_l + \Delta) - f_i(x_l)}{\Delta} \tag{4.3.12}$$

where Δ is a positive real on the order of 0.0001. The inverse to $\hat{\mathbf{J}}_k$ is determined initially from Gaussian elimination; subsequently, it is is updated by rank-one corrections. The details of the rank-one correction are not given; the interested reader is referred to pages 193-194 of [Luen73].

§ 4.4 Powell Hybrid Method Software

The routine HYBRD is a FORTRAN program found in the MINPACK library written at Argonne National Laboratory by B.S. Garbow, K.E. Hillstrom, and J.J. More [GHM80], [Gar80], [More80]. HYBRD finds a zero of a system of N nonlinear equations in N variables by the Powell hybrid method. The user supplies a subroutine which calculates the objective functions. The Jacobian is determined as described in section 4.3. HYBRD's I/O statement is

HYBRD(FCN,N,X,FVEC,TOL,INFO,WA,LWA)

where

FCN: the name of the user-supplied subroutine which calculates the functions;

N: a positive integer input variable set to the number of functions and variables;

X: a real array of length N; on input, X must contain an initial estimate of the solution vector; on output, X contains the final estimate of the solution vector;

FVEC: an output array of length N which contains the functions evaluated at the output X;

TOL: a nonnegative real input variable; termination occurs when the algorithm estimates that the relative error between the error and the solution is at most TOL in all components;

INFO: an output integer flag used to indicate the status of the recent run of HYBRD; it is set to 0 for improper input parameters; set to 1 for successful execution; set to 2 for FCN calls exceeding 200*(N+1); set to 3 for TOL too small; and set to 4 for iteration not making good progress;

WA: a work array of length LWA;

LWA: a positive integer input variable not less than (N*(N*3+13))/2.

§ 4.5 *Inversion of Harrier Tensor Product Spline Model*

We recast the Harrier trim equations in the form $Y(x) = 0$. That is, define

$$Y(u^M, p_1^1, C) := \left\{ \begin{bmatrix} m_0^M(u^M, p_1^1) \\ m_0^F(u^M, p_1^1) \\ E_R^2(\psi) \end{bmatrix} - \begin{bmatrix} \dot{\omega}_c \\ \dot{v}_{N,c} \\ E_R^2(\psi_c) \end{bmatrix} \right\} \tag{4.5.1}$$

HYBRD is used to solve for u^M, p_1^1, and C for given $\dot{\omega}$, $\dot{v}_N$, and ψ. Note that the HYBRD routine performed well under extensive testing.

Chapter 5 — Preliminary Results and Concluding Remarks

§ 5.1 *Introduction*

In this chapter, we offer some preliminary results regarding the performance of the control system design of Fig. 2.1.1. Also, we discuss briefly the state of continuing work.

§ 5.2 *Preliminary results*

Using the nonlinear automatic flight control system design presented in this report, we simulate the commanded short-take-off of the Harrier. The Harrier is initially at rest. (Note that the nozzle rest position is 90°.) The Harrier is commanded to achieve -- in 20 seconds time -- a runway-fixed 1-axis velocity of 75 ft/s with an acceleration of 5 ft/s, a runway-fixed 3-axis velocity of -25 ft/s with an acceleration of -3 ft/s and a nozzle position of 75°. Figures 5.2.1 through 5.2.5 show the actual Harrier response to the commanded flight condition. In all cases, perfect tracking is attained as the Harrier responds with smooth time history trajectories.

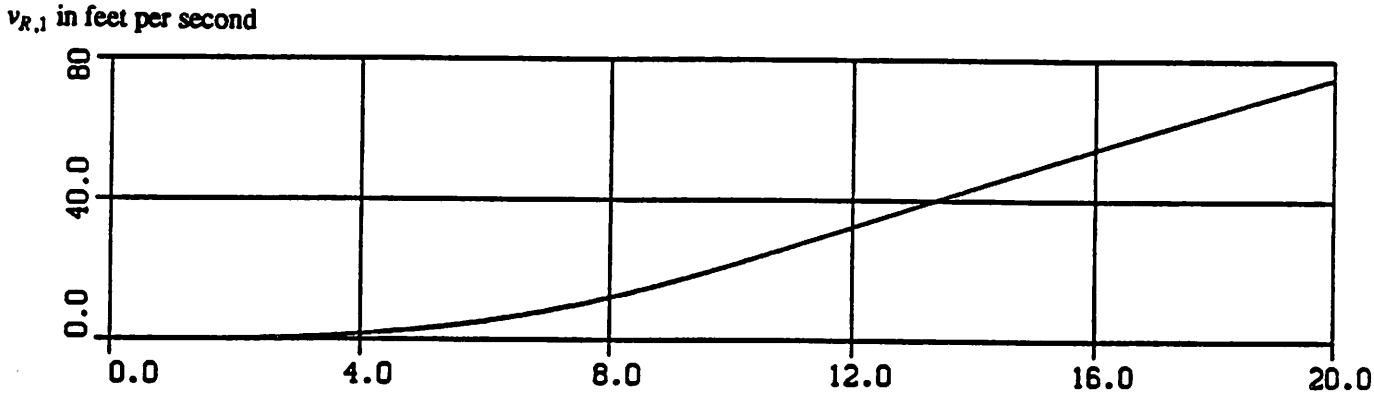

Figure 5.2.1 Harrier response: runway-fixed 1-axis component of velocity as a function of time.

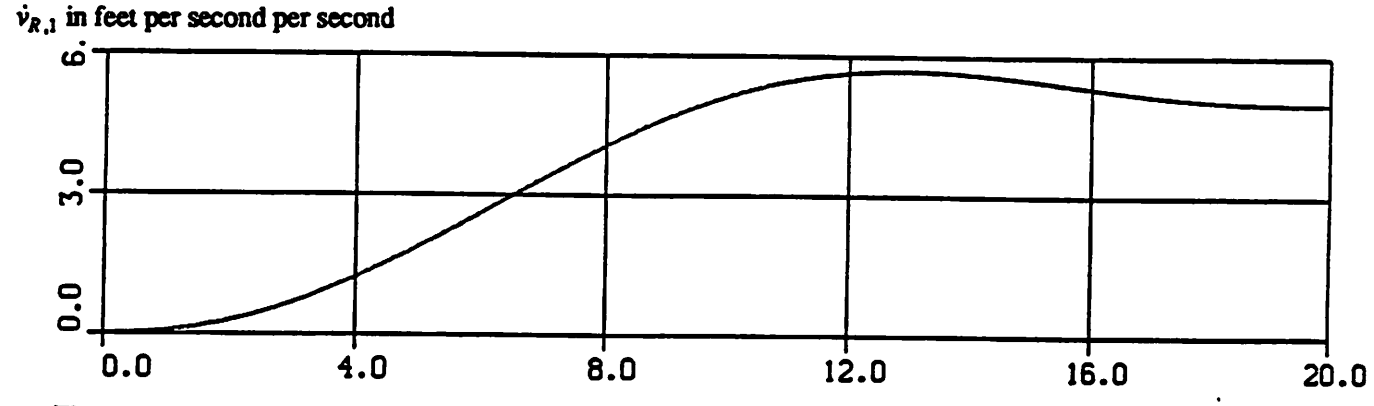

Figure 5.2.2 Harrier response: runway-fixed 1-axis component of acceleration as a funtion of time.

$v_{R,3}$ in feet per second

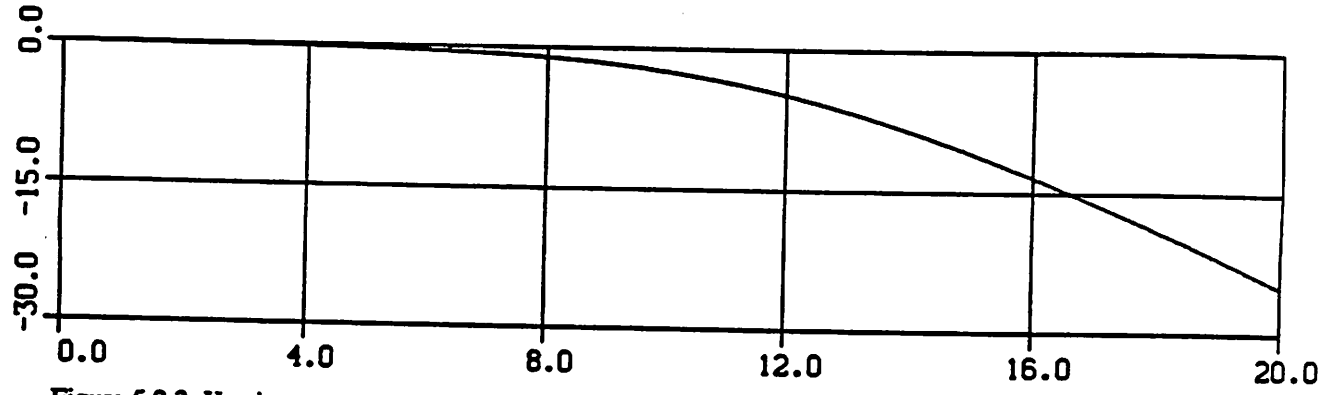

Figure 5.2.3 Harrier response: runway-fixed 3-axis component of velocity as a function of time.

t in seconds

$\dot{v}_{R,3}$ in feet per second per second

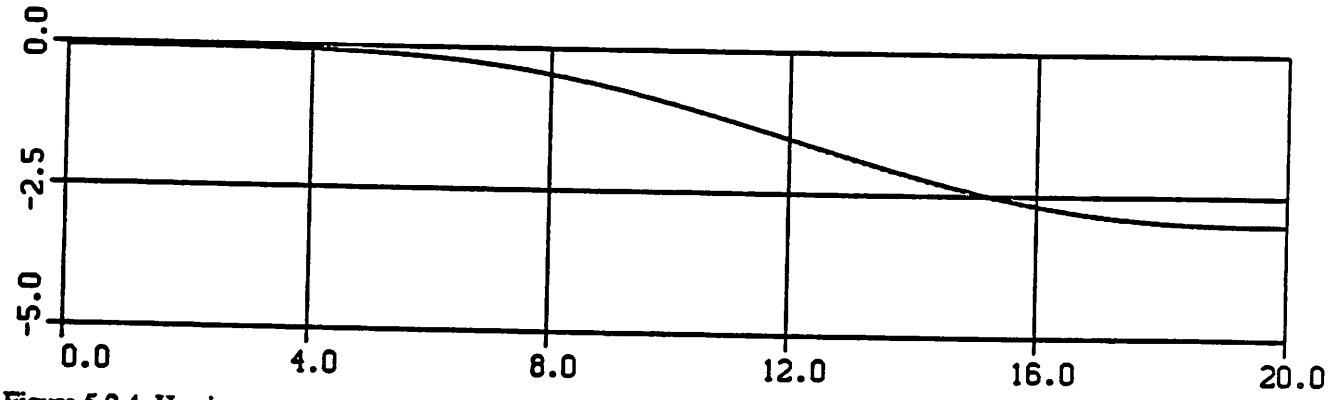

Figure 5.2.4 Harrier response: runway-fixed 3-axis component of acceleration as a funtion of time.

t in seconds

p_2^1 in degrees

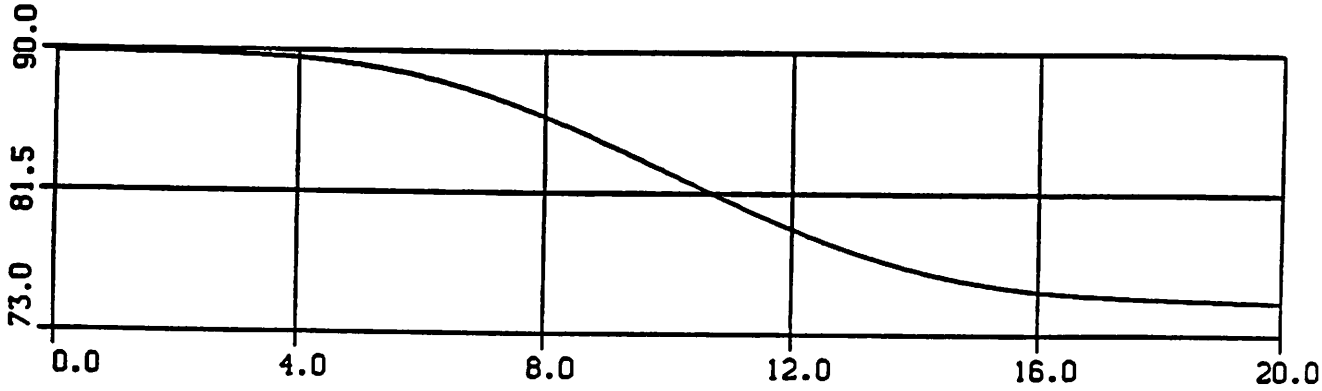

Figure 5.2.5 Harrier response: nozzle angle as a function of time.

t in seconds

§ 5.3 *State of continuing work*

This report has emphasized the methodologies used in designing the control system of Fig. 2.1.1 for the Harrier aircraft. We have presented the results of just one simulation as a means of illustrating the performance capabilities of the control design. However, an exhaustive testing of the control design is to be conducted in the near future to thoroughly verify the design performance. We are also working on the extension of the control design to Mode 1 operation of the Harrier. This requires 5-dimensional tensor product spline interpolants for modeling the force and moment processes because the nozzle position p_2^1 is now treated as a trim variable. That is, the spline models must be functions of p_2^1 and can no longer be parameterized by p_2^1.

§ 5.4 *Concluding remarks*

In this report, we have considered the design of a nonlinear automatic flight control system for the YAV/8B Harrier V/STOL aircraft. The design theory (Chapter 2) along with the techniques used to handle the necessary nonlinear modeling (Chapter 3) and nonlinear function inversion (Chapter 4) have been presented in detail. Preliminary results have been given as well as a brief status report on continuing work.

APPENDIX 1 — Harrier Coordinate Frames

Every good engineer knows the value of properly selected coordinate frames in problem solving and the need to be able to readily transform from one set of coordinates to a more revealing set of coordinates. To this end, in this appendix is presented a complete listing of the different right-hand-rule coordinate frames used in the Harrier control system design presented in this report. Also presented are the transformation matrices which relate the coordinate frames to one another [SMN86], [Shef83], [McRu73].

There are five coordinate frames in which the Harrier kinematics and dynamics are described. Each coordinate frame is defined by an orthonormal dextral set of unit vectors. The coordinate frames are:

(1) North-East-Down or N-E-D coordinate frame (also called the inertial or **runway–fixed axes**);

(2) Harrier body coordinate frame or, simply, **body axes**;

(3) Harrier heading angle coordinate frame or, simply, **heading axes**;

(4) Harrier nozzle angle coordinate frame or, simply, **nozzle axes**;

(5) **stability axes** coordinate frame.

Note that (2) through (5) are attached to the center of mass of the Harrier while (1) has its origin at runway origination position.

Figure A.1.1 illustrates the relationships between these different coordinate frames. Each directed arrow represents the transformation from the initial coordinates to the final coordinates by way of the direction cosine transformation matrix C_{KL} where K represents the final coordinate system and L, the initial. The notation adopted is R for runway-fixed axes, B for body axes, H for heading axes, N for nozzle axes, and S for stability axes. Thus, for example, C_{RH} is the transformation from heading axes to runway-fixed axes: if $\mathbf{h} = [h_1, h_2, h_3]^T$ is a triple defining a point in the heading axes, then $\mathbf{r} = [r_1, r_2, r_3]^T$ where $\mathbf{r} = C_{RN}\mathbf{h}$ is the triple defining this point in the runway axes.

The five coordinate systems are illustrated in Figures A.1.2(a) through A.1.2(e). Descriptions of each coordinate frame are provided in the following paragraphs.

The runway-fixed or N-E-D coordinate frame (Fig. A.1.2(a)) has its 1-axis directed north, its 2-axis directed east, and its 3-axis directed down as given by the cross product of the 1-axis unit vector with the 2-axis unit vector.

The body axes system (Fig. A.1.2(b)) has its 1-axis directed out the nose of the aircraft along the line segment originating at the aircraft center of mass, its 2-axis out the right wing of the aircraft along the line segment originating at the cm and perpendicular to the line segment defining the 1-axis, and its 3-axis directed as given by the cross product of the 1-axis unit vector with the 2-axis unit vector (note that the 3-axis is directed parallel to the landing gear of the aircraft).

The three Euler angles, **roll** (ϕ), **pitch** (θ), and **yaw** (ψ), are defined as those angles through which the runway-fixed coordinate frame is rotated to define the attitude of the aircraft (i.e., to transform the runway-fixed frame to the body frame). The first rotation is by the angle ϕ about the runway-fixed 1-axis, defining an intermediate coordinate frame called I_1. The second rotation is about the 2-axis of I_1 by the angle θ, defining a second intermediate frame called I_2. The third and last rotation is by the angle ψ about the 3-axis of I_2, making the transformation to the body frame complete.

For the definition of the heading axes system, consider the angle P by

$$P = \tan^{-1}\left[\frac{v_E}{v_N}\right] \tag{A.1.1}$$

where v_E is the velocity component directed east (i.e., the N-E-D 2-axis component of velocity) and v_N is the velocity component directed north (i.e., the N-E-D 1-axis component of velocity). That is, to determine the angle P, project the aircraft flight path onto the horizontal (i.e., the N-E) plane forming path p as shown in Fig. A.1.2(c). The angle P is that angle made between North and the projected aircraft velocity vector at a point on path p. The heading axes system is formed by rotating the N-E-D frame about its 3-axis by the angle P. The angle P is said to be the **path angle**. Therefore, the heading 2-axis is out the right wing of the aircraft oriented along a unit vector perpendicular to the 1-axis unit vector, and the 3-axis directed as given by the cross product of the 1-axis unit vector with the 2-axis unit vector.

The nozzle axes system (Fig. A.1.2(d)) has its 1-axis directed along the vector which is rotated up from the body 1-axis of the aircraft by the **nozzle angle** η. Note that in defining the nozzle axes system it

is assumed that the nozzle is located at the cm of the aircraft. The the nozzle axes system is formed by rotating the body axes system about the body 2-axis by the angle η: that is, the nozzle axes system is obtained from the body axes system by rotating the latter by the angle η about the body 2-axis. The nozzle 2-axis is directed out the right wing on the aircraft and the nozzle 3-axis is directed as given by the cross product of the 1-axis unit vector with the 2-axis unit vector.

For the definition of the stability axes system, consider the following relations where $v_{B,1}$ is the body 1-axis component of velocity, $v_{B,2}$ is the body 2-axis component of velocity, and $v_{B,3}$ is the body 3-axis component of velocity:

$$v = (v_{B,1}^2 + v_{B,2}^2 + v_{B,3}^2)^{1/2} \tag{A.1.2}$$

$$\alpha = \tan^{-1}\left[\frac{v_{B,3}}{v_{B,1}}\right] \tag{A.1.3}$$

$$\beta = \sin^{-1}\left[\frac{v_{B,2}}{v}\right] \tag{A.1.4}$$

The angle α is called the **angle of attack**. Rotating the body axes system about its 2-axis by α defines the stability axes system (Fig. A.1.2(e)). The angle β is called the **sideslip angle** is the angle between the velocity vector of the aircraft and the 1-axis of the stability axes system.

Before giving the expressions for the various transformation matrices, the concept of an elementary direction cosine matrix needs to be introduced. A rotation by an angle ξ about the k-axis (where k is either 1, 2, or 3) of a coordinate frame K induces the following transformation matrices:

$$\mathbf{E}_1^K := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\xi) & \sin(\xi) \\ 0 & -\sin(\xi) & \cos(\xi) \end{bmatrix} \tag{A.1.5}$$

$$\mathbf{E}_2^K := \begin{bmatrix} \cos(\xi) & 0 & -\sin(\xi) \\ 0 & 1 & 0 \\ \sin(\xi) & 0 & \cos(\xi) \end{bmatrix} \tag{A.1.6}$$

$$\mathbf{E}_3^K := \begin{bmatrix} \cos(\xi) & \sin(\xi) & 0 \\ -\sin(\xi) & \cos(\xi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.1.7}$$

Utilizing the elementary direction cosine matrix notation, we specify the following relationships for the transformations pictured in Fig. A.1.1:

$$C_{BR} = E_3^{I'}(\psi)E_2^{I'}(\theta)E_1^{R}(\phi) \tag{A.1.8}$$

$$C_{RB} = C_{BR}^{T} \tag{A.1.9}$$

$$C_{HR} = E_2^{R}(P) \tag{A.1.10}$$

$$C_{RH} = C_{HR}^{T} \tag{A.1.11}$$

$$C_{NB} = E_2^{B}(\eta) \tag{A.1.12}$$

$$C_{BN} = C_{NB}^{T} \tag{A.1.13}$$

$$C_{SB} = E_2^{B}(\alpha) \tag{A.1.14}$$

$$C_{BS} = C_{SB}^{T} \tag{A.1.15}$$

The above listing of transformations is complete in the sense that starting in any coordinate frame a transformation to any other frame is available either directly from the listing above or via combinations of transformations given above. For example, see Fig. A.1.1 to transform the stability axes into the heading axes, first transform to body then to N-E-D then to heading.

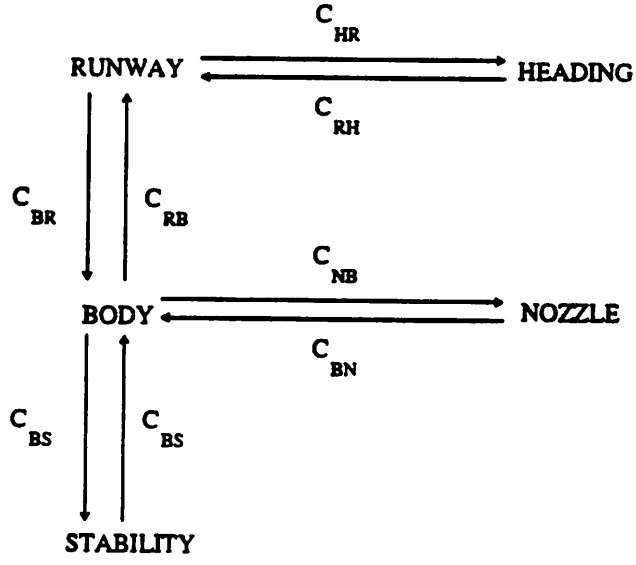

Figure A.1.1 Transformation relationships for the Harrier coordinate frames

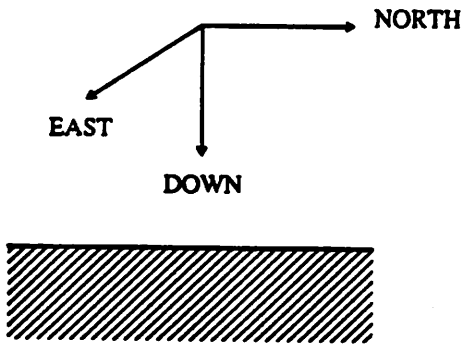

Figure A.1.2(a) Definition of the runway-fixed coordinate frame

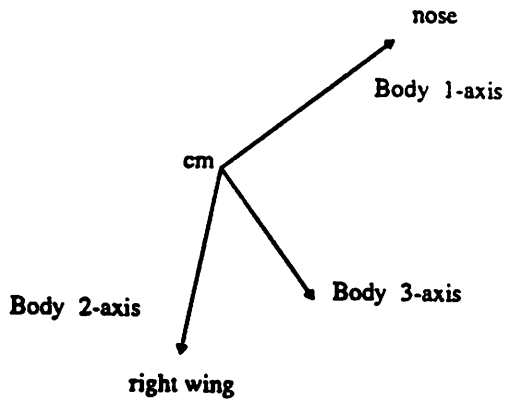

Figure A.1.2 (b) Definition of body coordinate frame

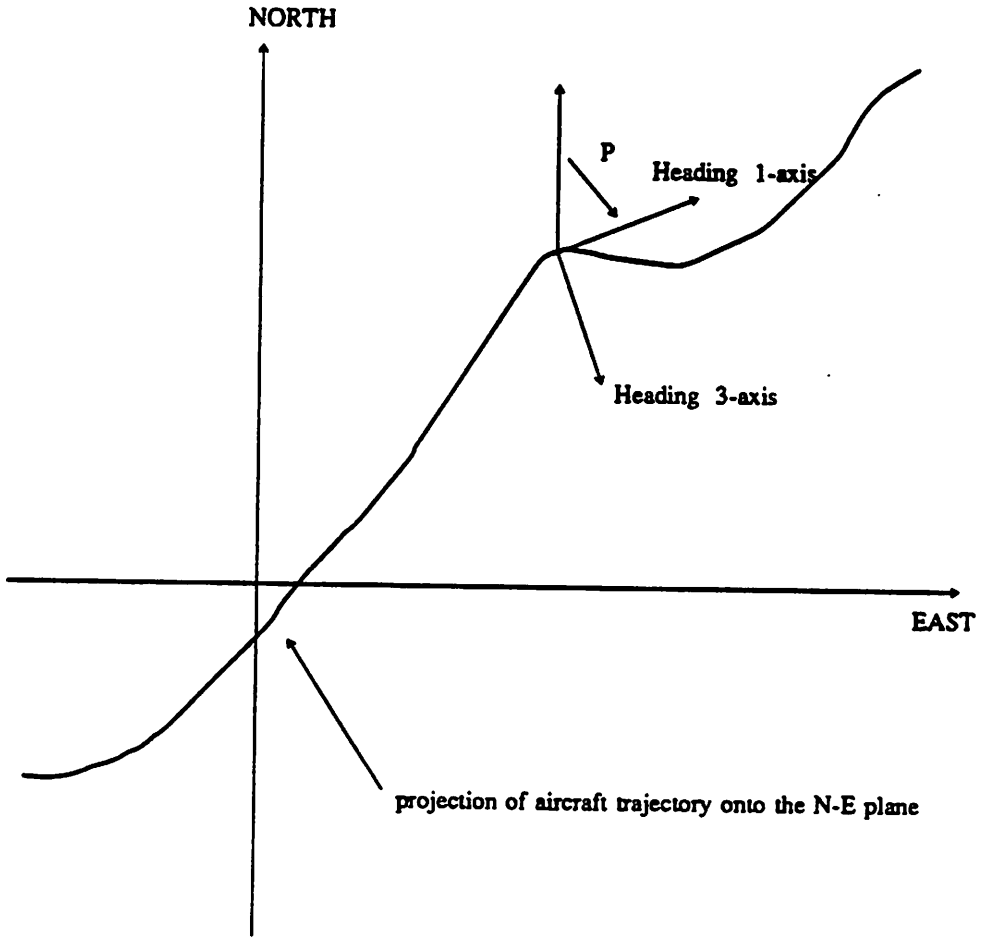

Figure A.1.2 (c) Definition of the Heading angle coordinate frame.

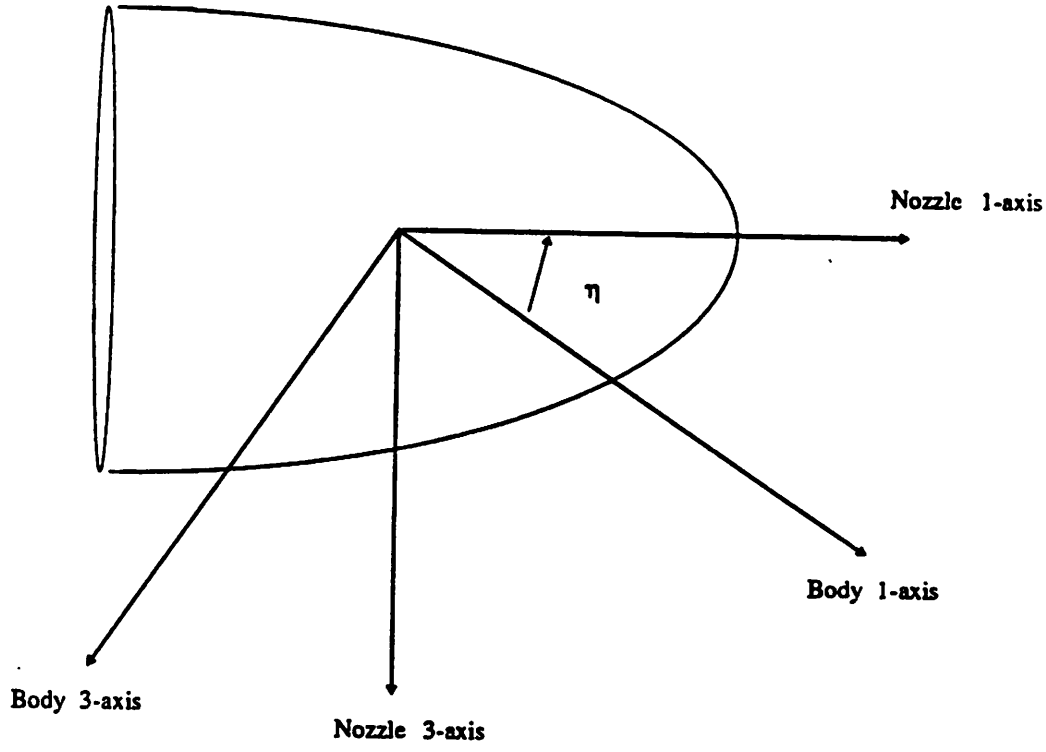

Figure A.1.2 (d) Definition of the nozzle angle coordinate frame

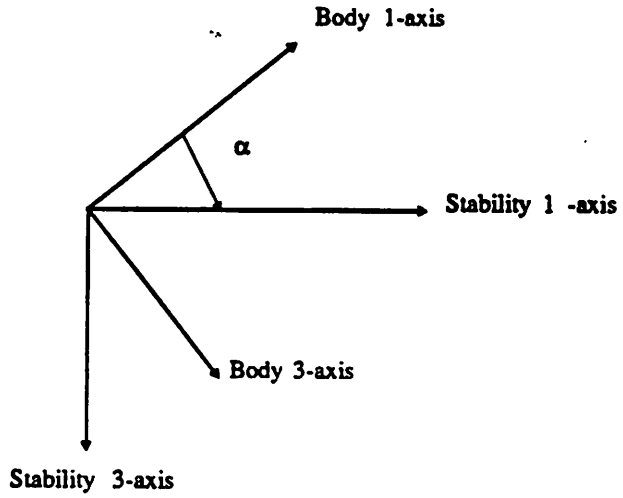

Figure A.1.2 (e) Definition of stability axes coorddinate frame

List of Symbols

1. Vectors and Matrices

$\mathbf{x}$ natural model state vector, p.4

$\mathbf{u}$ natural model control vector, p.4

$\mathbf{y}$ Brunovsky canonical model state vector, p.4

$\mathbf{w}$ Brunovsky canonical model control vector, p.4

$\mathbf{e_y}$ plant-to-model error vector, p.4

$\delta\mathbf{w}$ Brunovsky canonical model control correction, p.4

$\mathbf{r}$ runway coordinates of body center of mass position, p.5

$\mathbf{v}$ runway coordinates of body center of mass velocity, p.5

$\mathbf{C}$ direction cosine matrix of body-fixed axes relative to runway-fixed axes, p.5

ω body coordinates of angular velocity relative to runway-fixed axes, p.5

$\mathbf{p}^1$ throttle position and nozzle angle, p.5

$\mathbf{p}^2$ throttle rate and nozzle rate, p.5

$\mathbf{u}^M$ three axes moment controls, p.5

$\mathbf{u}^P$ throttle and nozzle controls, p.5

$S(\omega)$ skew symmetric matrix of the angular velocity vector, ω, p.9

$\mathbf{e}_i$ the i^{th} column of an $n \times n$ identity matrix, p.9

$\mathbf{J}_k$ Jacobian matrix of system $\mathbf{f}$ at $\mathbf{x}_k$, p.39

$\mathbf{H}_k$ $\mathbf{J}_k^{-1}$, the inverse of nonsingular $\mathbf{J}$, p.39

$\hat{\mathbf{J}}_k$ approximate $\mathbf{J}_k$, p.41

$\hat{\mathbf{H}}_k$ approximate $\mathbf{H}_k$, p.41

$\mathbf{x}_k$ current estimate of solution to $\mathbf{f}(\mathbf{x}) = 0$, p.39

δ_k update to $\mathbf{x}_k$ to give $\mathbf{x}_{k+1} := \mathbf{x}_k + \delta_k$, p.39

$\mathbf{E}_i^K(\xi)$ for i = 1, 2, 3, the elementary direction cosine matrix about axis i of the K coordinate frame by ξ degrees, p.51

C_{KI} direction cosine matrix of the K coordinate frame to the L coordinate frame where K, L may be R for runway, B for body, H for heading angle, N for nozzle angle and S for stability axes, p.53-54

2. Real and Integer Scalars

ϕ roll angle, p.5

θ pitch angle, p.5

ψ yaw angle, p.5

g gravitational acceleration, $9.8\frac{m}{s^2}$, $32.2\frac{ft}{s^2}$, $1\text{-}g$, p.39

η nozzle angle, p.52

P path angle, p.52

α angle of attack, p.53

β sideslip angle, p.53

Δ_k steplength parameter of Powell hybrid method, p.41

κ_i Kronecker index or controllability index, p.42

s_i $:= \sum_{j=1}^{i} \kappa_i$, p.8

n state vector size, p.8

m control vector size, p.8

3. Functions, Maps, General Analysis

$T(\cdot)$ transformation map from natural model state and control to Brunovsky canonical model state and control, p.4

$W(\cdot)$ transformation map of Brunovsky state and control to natural model state and control, p.4

$f^F(\cdot)$ runway coordinates of the components of force acting on the aircraft, p.8

$f^M(\cdot)$ body coordinates of the moments acting on the aircraft, p.8

$\mathbf{f}_N^0(\cdot)$ non-parasitic nozzle 2- and 3- axes components of force acting on the aircraft, p.19

$\mathbf{f}_N^1(\cdot)$ parasitic nozzle 2- and 3- axes components of force acting on the aircraft, p.19

$\mathbf{h}^1(\cdot)$ function which recovers natural model controls from natural model state, commanded moments and commanded nozzle 1-axis acceleration, p.18

$\mathbf{h}^2(\cdot)$ function which determines aircraft commanded attitdue from position, velocity, commanded acceleration, and path angle, p.19

$\mathbf{m}_0^F(\cdot)$ vector of tensor product spline functions modeling $\mathbf{f}^F$ for given state and nozzle control, p.22

$\mathbf{m}_0^M(\cdot)$ vector of tensor product spline functions modeling $\mathbf{f}^M$ for given state and nozzle control, p.22

$B_{i,k,t}(\cdot)$ the i^{th} (normalized) B-spline of order k for knot sequence t , p.24

$B_i(\cdot)$ short-hand for $B_{i,k,t}(\cdot)$, p.24

$(t-x)_+$ function truncation, p.24

$m(\cdot)$ B-representation of $f(\cdot)$, p.24

$\mathbf{m}(\cdot)$ B-representation of $\mathbf{f}(\cdot)$, p.24

$\mathbf{f}(\cdot)$ system of at least n C^1 functions in n variables, p.38

$\nabla f(\cdot)$ gradient of $f(\cdot)$, p.40

$F(\cdot)$ steepest descent cost function in Powell hybrid method, p.41

4. Sequences and Sets

$(\zeta_i)_1^{l+1}$ breakpoint sequence for piecewise polynomial function, p.5

$(t_i)_1^{n+k}$ knot sequence for B-representation of piecewise polynomial function, p.24

$(\nu_i)_1^{l+1}$ integer sequence of continuity conditions for B-representation, p.25

$\mathbb{N}$ set of nonnegative integers, namely, {0, 1, 2, . . .}

$\mathbb{R}$ field of real numbers

$\mathbb{R}^n$ n-dimensional space of real valued n-tuples defined over the field of real numbers

$\mathbb{P}_{k,\zeta}$ linear space of piecewise polynomial functions of order k with breakpoint sequence ζ, p.24

$\mathbb{P}_{k,\zeta,\nu}$ subspace of $\mathbb{P}_{k,\zeta}$ consisting of elements satisfying the continuity conditions specified by ν, p.23

References

[MSH82]
G. Meyer, R. Su, and L.R. Hunt, "Application of Nonlinear Transformations to Automatic Flight Control," 1982 IEEE Automatic Control Conference, Arlington, VA June 13-16, 1982.

[MHS83G]
G. Meyer, L.R. Hunt, and R. Su, "Design of a Helicopter Autopilot by Means of Linearizing Transformations," Guidance and Control Panel 35th Symposium, AGARD CP 321, pp. 4-1 - 4.11.

[MSH83C]
G. Meyer, R. Su, and L.R. Hunt, "Applications to aeronautics of the theory of transformations of nonlinear systems," 1983 CNRS Conference, pp153-162

[HSMmi]
L.R. Hunt, R. Su, and G. Meyer, "Design for Multi-input Nonlinear Systems," Differential Geometric Control Conference, Birkhauser, Boston, Cambridge, Mass., 1982

[SMN86]
G.A. Smith, G. Meyer, and M. Nordstrom, "Aircraft Automatic-Flight-Control System with Inversion of the Model in the Feed Forward Path Using a Newton Raphson Technique for Inversion," NASA TN July, 1986

[SM87]
G.A. Smith and G. Meyer, "Aircraft Automatic Flight Control System with Model Inversion," AIAA Journal of Guidance, Control, and Dynamics, Volume 10, Number 3, May-June 1987 pp. 269 - 275

[DeU87]
A. DeLuca and G. Ulivi, "Full Linearization of an Induction Motor Via Nonlinear State Feedback," Proceedings, Conference on Decision and Control, 1987, pp. 1765-1776.

[HSM88]
J.E. Hauser, S. Sastry, and G. Meyer, "Linearization Techniques for Flight Control Systems," IFAC Conference on Nonlinear Control, Milan Italy, June 1988 (submitted)

[Blank]
G.L. Blankenship and W. Akhrif, CONDENS: Control Design of Nonlinear Systems, A Software Application Package, University of Maryland, Dept. of Electrical Engineering and Computer Science, College Park, MD, 1987

[TeelMS]
A.R. Teel, Master's Project, University of California, Berkeley, Dept. of Electrical Engineering and Computer Sciences, work-in-progess

[Ford83]
C.H. Ford, "Numerical and Symbolic Methods for Transforming Control Systems to Canonical Form," Doctoral Dissertation, Texas Tech Univ., 1983

[Shev83]
R.S. Shevell, Fundamentals of Flight, Prentice Hall, Inc. Englewood Cliffs, NJ 07632, 1983.

[McAG73]
D. McRuer, I. Ashkenas, D. Graham, Aircraft Dynamics and Automatic Control, Princeton Univ. Press, Princeton, NJ 1973.

[McD82]
YAV-8B Simulation and Modeling. McDonnell Douglas Corporation, December 1982.

[deBr78]
C. deBoor, A Practical Guide to Splines, Springer-Verlag, Inc., New York, 1978

[deBr79]
C. DeBoor, "Efficient Computer Manipulation of Tensor Products," ACM Transactions on Mathematical Software, Vol. 5 (1979), pp. 173 - 182.

[Bois82]
R. Biosvert, BSPLINE and TENSORBS Fortran Routines, NBS Scientific Computing Division, National Bureau of Standards, Washington, D.C., 1982.

[Nye86]
W. Nye, Multidimensional Generalization of TENSORBS routines, Epsilon Aative, Inc, Berkeley, CA 1986.

[MJDP70]
M.J.D. Powell, "A Hybrid Method for Nonlinear Equations," in Numerical Methods for Nonlinear Algebraic Eqautions, P. Rabinowitz, ed., Gordon and Breach, 1970.

[MJDP70]
M.J.D. Powell, "A Fortran Subroutine for Solving Systems of Nonlinear Algebraic equations," in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, 1970.

[Rice83]
J.R. Rice, Numerical Methods, Software, and Analysis, McGraw Hill Book Company, New York, 1983.

[Polak88]
E. Polak, EECS 227a Engineering Optimization Class Notes, Univ. of Calif. - Berkeley, EECS Department, Fall 1988.

[Luen73]
D.G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley Publishing Company, Menlo Park CA, 1973.

[GHM80]
B.S. Garbow, K.E. Hillstrom, and J.J. More, MINPACK Project, Argonne, Ill., Argonne National Laboratory, 1980.

[Gar80]
B.S. Garbow, "Implementation Guide for MINPACK-1," Argonne, Ill., Argonne National

Laboratory, 1980.

[More80]
J.J. More, "User's Guide for MINPACK-1," Argonne, Ill., Argonne National Laboratory, 1980.