AIRPLANE TRAJECTORY EXPANSION
FOR DYNAMICS INVERSION

by
Bruce C. Munro

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Aerospace Engineering

APPROVED:

[Signature]
W. C. Durham, Chairman

[Signature]
F. H. Lutze

[Signature]
E. M. Cliff

May 22, 1992
Blacksburg, Virginia
AIRPLANE TRAJECTORY EXPANSION
FOR DYNAMICS INVERSION

by

Bruce C. Munro

Committee Chairman: W. C. Durham
Aerospace Engineering

(ABSTRACT)

In aircraft research, there is keen interest in the procedure of determining the set of controls required to perform a maneuver from a definition of the trajectory. This is called the inverse problem. It has been proposed that if a complete set of states and state time derivatives can be derived from a trajectory then a model-following solution can allocate the controls necessary for the maneuver. This paper explores the problem of finding the complete state definition and provides a solution that requires numerical differentiation, fixed point iteration and a Newton's method solution to nonlinear equations. It considers trajectories that are smooth, piecewise smooth, and noise ridden. The resulting formulation was coded into a FORTRAN program. When tested against simple smooth maneuvers, the program output was very successful but demonstrated the limitations imposed by the assumptions and approximations in the development.
ACKNOWLEDGMENTS

I would like to gratefully acknowledge Dr. Wayne C. Durham for his support and encouragement as well as his tremendous contributions to this project. His insight as a test pilot lent great perspective and his ability as a communicator made the research as enjoyable as it was challenging. Additional thanks go to Dr. Durham and Dr. Frederick Lutze for the opportunity to participate in the research effort with NASA LaRC (NCC1-158).

I would also like to thank Brett Malone and Will Cordle who acted as liaisons during my internship. Many hours were saved due to their efforts.

Additionally, I would like to express appreciation to NASA LaRC for the use of its facilities. Keith Hoffler, Mark Strickland, Marilyn Ogburn, and John Foster deserve special recognition for their invaluable assistance and my appreciation for taking interest.

Dedicated to my mother and father, an endless source of inspiration.
# TABLE OF CONTENTS

List of Figures ........................................................................................................... vii
List of Tables .............................................................................................................. x
List of Symbols ......................................................................................................... xi

1. Introduction ........................................................................................................... 1

2. Problem Formulation .......................................................................................... 4

   2.1 Description of Trajectory (Input) ................................................................. 4
   2.2 State and State Derivatives Required (Output) .............................................. 5
   2.3 Dependence Property of Numerical Differentiation Scheme ..................... 6
   2.4 Assumptions ..................................................................................................... 7
   2.5 Formulation Objectives ................................................................................... 9

3. Algorithm for Trajectory Expansion .................................................................. 11

   3.1 Earth Axes Quantities .................................................................................. 11
   3.2 Wind Axes Considerations .......................................................................... 13
   3.3 Definition of Wind Axes Roll Angle ............................................................. 14
   3.4 Determination of $\alpha$ and $\beta$ ................................................................ 16
   3.5 Determination of $P$, $Q$, and $R$ ............................................................... 18
   3.6 Body Axes Velocity Components: $U$, $V$, $W$ ............................................ 20
   3.7 Euler Parameter Extraction ......................................................................... 20
   3.8 Complete State Definition .......................................................................... 22

4. Numerical Differentiation .................................................................................... 23

   4.1 Design Goals .................................................................................................. 23
   4.2 Data Pairs and Relation in Time ................................................................... 24
   4.3 Least Squares ................................................................................................. 26
## LIST OF FIGURES

1. Resetting the Origin ................................................................. 26

2. Least Squares A Matrix Optimization .......................................... 33

3. Lost Dynamics ........................................................................ 35

4. 3rd Order Least Squares Derivative ............................................ 38

5. Example of Data with Discontinuous Derivatives ......................... 40

6. Oscillations Across Corners ..................................................... 41

7. Transition Across Corners ....................................................... 43

8. Differentiating Noisy Input ...................................................... 46

9. Mapping Smoothness for Fixed Point ......................................... 53

10. Fixed Point Iteration Schematic ............................................... 54

11. Experimental Equipment - F/A-18 DMS Simulator ...................... 66

12. Trajectory 1: ......................................................................... 70

   12.1 Velocity at Center of Gravity ............................................. 70

   12.2 Heading Angle .................................................................. 71

   12.3 Flight Path Angle ............................................................ 72

   12.4 Wind Axes Roll Angle ...................................................... 73

   12.5 Angle of Attack ................................................................ 74

   12.6 Sideslip Angle .................................................................. 75

   12.7 Body Axes Roll Rate ......................................................... 76

   12.8 Body Axes Pitch Rate ......................................................... 77

   12.9 Body Axes Yaw Rate ......................................................... 78

   12.10 Body Axes Velocity Component ....................................... 79

   12.11 Body Axes Y Velocity Component ................................... 80
12.13 Euler Parameters ......................................................... 82

13. Trajectory 2: ................................................................. 85

13.1 Velocity at Center of Gravity ........................................ 85
13.2 Heading Angle ............................................................ 86
13.3 Flight Path Angle ...................................................... 87
13.4 Wind Axes Roll Angle .................................................. 88
13.5 Angle of Attack .......................................................... 89
13.6 Sideslip Angle ............................................................ 90
13.7 Body Axes Roll Rate .................................................... 91
13.8 Body Axes Pitch Rate ................................................... 92
13.9 Body Axes Yaw Rate .................................................... 93
13.10 Body Axes X Velocity Component ................................. 94
13.11 Body Axes Y Velocity Component ................................. 95
13.12 Body Axes Z Velocity Component ................................. 96
13.13 Euler Parameters ...................................................... 97
13.14 Changes from Modifying TEA CL vs \( \alpha \) Curve .......... 98

14. Trajectory 3: ................................................................. 101

14.1 Velocity at Center of Gravity ........................................ 101
14.2 Heading Angle .......................................................... 102
14.3 Flight Path Angle ...................................................... 103
14.4 Wind Axes Roll Angle .................................................. 104
14.5 Angle of Attack .......................................................... 105
14.6 Sideslip Angle ............................................................ 106
14.7 Body Axes Roll Rate .................................................... 107
14.8 Body Axes Pitch Rate ................................................... 108
14.9 Body Axes Yaw Rate ................................................................. 109
14.10 Body Axes X Velocity ............................................................... 110
14.11 Body Axes Y Velocity Component .............................................. 111
14.12 Body Axes Z Velocity Component .............................................. 112
14.13 Euler Parameters ................................................................. 113
14.14 TEA Approximation of Roll Angle ............................................. 114
14.15 Control Surfaces ................................................................. 115
LIST OF TABLES

1. Design Specifications for Least Squares Formulation..........................37
2. Differentiator Parameters........................................................................47
3. Airframe Program Variables.....................................................................62
4. Output Variables.......................................................................................63
5. F/A-18 Low Angle of Attack Data: M=0.6, 20,000 Ft.................................67
LIST OF SYMBOLS

- The derivative of an independent variable with respect to time

α - Angle of attack

˙α - Time rate of change of angle of attack

A - Least squares "A" matrix

AR - Aspect ratio

β - Angle of sideslip

˙β - Time rate of change of sideslip angle

b - Least squares "b" vector

b - Wing span

χ - Euler angle corresponding to a rotation about the z-axis in transforming from earth axes to wind axes

˙χ - Time rate of change of χ

c - Least Squares "c" vector, elements are coefficients to polynomial fit

C - Wind axes side-force

¯c - Mean aerodynamic chord

CD - Coefficient of drag, \( \frac{D}{q \cdot S} \)

CD0 - Zero lift drag coefficient

CL - Coefficient of lift, \( \frac{L}{q \cdot S} \)

CL0 - Zero angle of attack lift coefficient

CLα - Static stability derivative, change in lift due to change in angle of attack

CLα - Dynamic stability derivative, change in lift due to change in angle of attack rate"
C_{LQ} - Dynamic stability derivative, change in lift due to change in pitch rate

C_Y - Coefficient of body axes side-force, \( \frac{Y}{q \cdot S} \)

C_{Y\beta} - Static stability derivative, change in body axes side-force due to change in sideslip angle

C_{YP} - Dynamic stability derivative, change in body axes side-force due to change in roll rate

C_{YR} - Dynamic stability derivative, change in body axes side-force due to change in yaw rate

D - Drag

e - Airplane efficiency factor

\( \varepsilon \) - Margin of accuracy

F, G - Generic vector function operators

f, g - Generic scalar function operators

F_E - Force vector represented in earth axes

F_W - Force vector represented in wind axes

\phi - Euler angle corresponding to a rotation about the x-axis in transforming from earth axes to body axes

\phi_W - see \( \mu \)

F_{XE} - Earth x-axis component of force

F_{XW} - Wind x-axis component of force

F_{YE} - Earth y-axis component of force

F_{YW} - Wind y-axis component of force

F_{ZE} - Earth z-axis component of force

F_{ZW} - Wind z-axis component of force
g - Acceleration of gravity
γ - Euler angle corresponding to a rotation about the y-axis in transforming from earth axes to wind axes
\dot{γ} - Time rate of change of γ
η - History location; uniquely describes which slice of data is being curve fit
K - Drag due to lift coefficient, \( \frac{1}{\pi \cdot AR \cdot e} \)
k - Order of polynomial fit in least squares method
L - Lift
m - Aircraft mass
m - Maximum order of polynomial fit used in weighted average polynomial method
μ - Euler angle corresponding to a rotation about the x-axis in transforming from earth axes to wind axes
\dot{μ} - Time rate of change of μ
n - Number of samples in a standard time history; position of time history
P - Body axes roll rate
\dot{P} - Roll acceleration
\dot{P}_W - Wind axes roll rate
Q - Body axes pitch rate
\dot{Q} - Pitch acceleration
\dot{Q}_W - Wind axes pitch rate
θ - Euler angle corresponding to a rotation about the y-axis in transforming from earth axes to body axes
θ_w - see γ
\(q_0, q_1, q_2, q_3\) - Euler parameters
\(\bar{q}\) - Dynamic pressure, \(\frac{1}{2} \rho V^2\)
\(R\) - Body axes yaw rate
\(R^n\) - n-dimensional real space
\(\dot{R}\) - Yaw acceleration
\(R_W\) - Wind axes yaw rate
\(S\) - Wing area
\(T_{BE}\) - Transformation matrix from earth axes to body axes
\(T_{BW}\) - Transformation matrix from wind axes to body axes
\(T_{EB}\) - Transformation matrix from body axes to earth axes
\(T_{EW}\) - Transformation matrix from wind axes to earth axes
\(T_{WB}\) - Transformation matrix from body axes to wind axes
\(T_{WE}\) - Transformation matrix from earth axes to wind axes
\(T_{XW}\) - Wind x-axis component of thrust
\(T_{YW}\) - Wind y-axis component of thrust
\(T_{ZW}\) - Wind z-axis component of thrust
\(t\) - Time, time element of data pair
\(U\) - Body x-axis scalar component of velocity
\(\dot{U}\) - Body x-axis acceleration
\(V\) - Body y-axis scalar component of velocity
\(\dot{V}\) - Body y-axis acceleration
\(V\) - Velocity vector of aircraft center of gravity
\(V_c\) - Magnitude of \(V\)
\(W\) - Body z-axis scalar component of velocity
\(\dot{W}\) - Body z-axis acceleration
\( x \) - Generic independent variable, state, data element of data pair

\( \hat{x} \) - Approximation to generic state using least squares curve fit

\( \dot{x} \) - Time derivative of least squares approximation

\( X_m \) - State vector of model aircraft, \( X_m = \begin{bmatrix} q_s & q_i & q_z & q_r & P & Q & R & U & V & W \end{bmatrix}^T \)

\( \dot{x}_E \) - Earth x-axis component of velocity

\( \ddot{x}_E \) - Earth x-axis component of acceleration

\( Y \) - Body axes side-force

\( \dot{y}_E \) - Earth y-axis component of velocity

\( \ddot{y}_E \) - Earth y-axis component of acceleration

\( \psi \) - Euler angle corresponding to a rotation about the z-axis in transforming from earth axes to body axes

\( \psi_W \) - see \( \chi \)

\( \dot{z}_E \) - Earth z-axis component of velocity

\( \ddot{z}_E \) - Earth z-axis component of acceleration

* Non standard form
1. INTRODUCTION

In the analysis of the dynamics of aircraft, there exists a set of equations of motion such that if the forces and moments on the aircraft are known, then the trajectory of the aircraft may be fully predicted. In contrast, if only the trajectory of the aircraft is known, then the problem of determining the forces, moments and their instigating control inputs may be very difficult. This is known as the inverse problem.

The inverse problem has become an intense area of aerospace research due in large part to the increased agility of today's high performance aircraft. Trajectory optimization, target acquisition, advanced control power capabilities, computer augmented stability analysis and supermaneuverability issues are each focused on commanding an aircraft to follow a flight path that satisfies some performance criteria by determining and implementing a control law that takes full advantage of the craft's abilities. At the heart of this performance issue, lies the desire to follow a prescribed trajectory.

As an example, one simple challenge for an inverse solution is the tracking problem. A pilot may visualize a specific trajectory to acquire a target. Given this trajectory, an inverse solution can provide the command sequences required to produce that desired trajectory. Under some conditions, a pilot or conventional control law may have little difficulty commanding the aircraft because the controls have an intuitive response. Increasing the commanded input will increase a particular response.

However, other situations require an increased understanding of the effects of controls on the response of the aircraft. In some flight regimes, the nonlinear dynamics are dominant and the commanded inputs may not be instinctive. These nonlinearities can
manifest themselves to such extremes as even control reversals and can lead to pilot induced oscillations or other undesirable situation.

Some nonlinearities are the result of state of the aircraft. For example, some researchers have investigated the application of an inverse solution to maneuvers under extraordinary conditions such as high angle of attack or spin recovery. Other nonlinearities arise from the characteristics of the aircraft itself. Today's aircraft are introducing more exotic control devices or configurations. Canards, strakes, flap scheduling, differential stabilators and rudders, and thrust vectoring have become essential design considerations for these high performance craft. The combinations of these controls in some of the more elaborate configurations will produce effects much departed from linear theory. Finally, relaxed stability aircraft such as the X-29A may also require non-intuitive control inputs. These aircraft highly emphasize the need to consider handling qualities whose very definition is topic for debate. The use of an inverse solution can aid each of these or any design area which requires an augmented understanding of the effects of controls.

There are many proposed dynamics inversion solutions and they offer a wide variety of methodologies. Bugajski\textsuperscript{1} suggests an iterative procedure that inverts the dynamics' equations and uses a full nonlinear data-base to match output with desired output. This method requires selection of a desired set of angular accelerations to complete a maneuver and implements a guidance loop for correction. Gao and Hess\textsuperscript{2} developed an integration scheme so-called because it uses the equations of motion in forward fashion. A Newton's method solution updates the control law at every time step by reducing an error function. Alternatively, Lane\textsuperscript{3} searched for a control law derived by introducing a differentiation concept. This concept begins by assuming an equation model of the form $\dot{x} = A(x) + B(x) \cdot u$, and an output equation of the form $y = C(x)$. 
This idea suggests that the nonlinear output elements can be differentiated sufficiently to uncover the controls. This derivative can be assumed to have the form $y^{(d)} = A^*(x) + B^*(x) \cdot u$ and can be inverted as long as $B^*$ is non-singular. Kato asserts that angle of attack, sideslip angle, and body axes roll angle are variables directly controlled by elevator, rudder, and aileron respectively. Then each of the state equations can be written as functions of these variables. A nonlinear iterative procedure can be used which may either exclude or include control contributions to force.

Durham employs a model-following technique to solve the inverse problem. A complete description of this technique can be found in the literature. This type of control "attempts to make an actual airplane behave similar to a prescribed mathematical model..." Generally, the mathematical model is different from the controlled airplane. The model-following control provides the control law that will reduce the error between the model and the actual airplane. If the model is a mathematical representation of the actual airplane then the control law introduced is the solution to the inverse problem.

One part of this simplified mathematical model is the complete specification of aircraft states and state rates for a trajectory. The scope of the following investigation is to expand a specified trajectory into a complete set of these states and state rates. This trajectory expansion will become the input to the model-following control law.
2. PROBLEM FORMULATION

The goal of this work is to identify a procedure which will translate an aircraft trajectory into a full maneuver description characterized by the aircraft states and state rates. This will be referred to as the Trajectory Expansion Algorithm. This section introduces the definitions of the input and the output of the algorithm as well as essential assumptions, concepts, and objectives.

2.1 Description of Trajectory (Input)

If the atmosphere is still, then a complete description of a point mass trajectory can be given by the time history of its velocity vector \( \mathbf{V} \). The velocity vector can be described in many different reference frames. Etkin\(^7\) explains how any reference frame can be related to any other by three angles. For instance, the relationship between a vehicle-carried vertical frame and the wind axes frame is given by the set of angles \((\chi, \gamma, \mu)\). These angles represent successive rotations of the axes system about one axis. The first angle, \( \chi \), is the heading angle and denotes a rotation about the z-axis of the initial axes system. The second angle, \( \gamma \), is the flight path angle or angle of climb, and represents a rotation about the y-axis of an intermediate system. The final angle is the velocity bank angle, \( \mu \), which represents a rotation about the x-axis of the final system. It will be shown that this angle is not necessary for defining the velocity vector in the inertial reference frame, but may be included for completeness.
The vehicle-carried vertical frame can be aligned parallel to an earth fixed reference frame if the curvature of the earth is neglected. If the motion of the earth is also neglected then this becomes an inertial reference frame. Etkin defines the wind axes in the following manner. The center of gravity is the origin. The x-axis points along the velocity vector $V$, and the z-axis remains in the plane of symmetry of the vehicle if it has one. Otherwise, it is an arbitrary assignment. Therefore, in the wind axes system, the velocity vector can be given by:

$$\mathbf{v} = \begin{bmatrix} V_c \\ 0 \\ 0 \end{bmatrix}$$

The trajectory can therefore be described by a time history of the velocity vector in the wind axes system and the time histories of the angles that relate the wind axes system to an inertial reference frame.

2.2 State and State Derivatives Required (Output)

The required output of the Trajectory Expansion Algorithm is a state vector and a state derivative vector in order to describe the current condition of the aircraft model. The state vector will be defined as $x_m = \left[ q_0, q_1, q_2, q_3, P, Q, R, U, V, W \right]^T$ and the derivative vector will be defined by the time derivative of the state vector. $q_0$, $q_1$, $q_2$ and $q_3$ are the Euler parameters associated with the body axes to inertial axes transformation. These are used instead of the usual Euler angles, $\phi$, $\theta$, and $\psi$ to simplify error dynamics in the model-following formulation.$^6$ $P$, $Q$, and $R$ are the body axes angular rates around
the x, y, and z body axes respectively and \( U, V, \) and \( W \) are the components of the velocity in the body axes x, y, and z directions.

The state derivative vector is simply the time derivative of each element of the state vector and is denoted by the conventional dot, i.e., \( \dot{x}_m \). The input to the model-following procedure includes the state vector and a subset of the state derivative vector. This subset includes \( \dot{U}, \dot{P}, \dot{Q}, \) and \( \dot{R} \). The others are not necessary due to the nature of model-following which is explained in detail in the literature.

2.3 Dependence Property of Numerical Differentiation Scheme

Given that the time histories consist of data pairs of "time of sample" and "value of sample" (i.e. \((t, \chi)\)), a basic premise to the Trajectory Expansion Algorithm stipulates that the time histories of any of the states or state derivatives are accurate representations of time-dependent functions. Additionally, these functions can be differentiated at any point in the data stream and specifically at the end point.

In numerical approximation concepts such as curve fitting and interpolation, these data pairs are used directly to determine this functional dependence. For example, two points determine the function known as a straight line and the parameters of this line are the y-intercept and the slope. Elementary algebra taught that the slope is given by the "rise divided by the run" determined by the two abscissas and ordinates. The y-intercept is found similarly by applying the function model to the data pairs. Although this is an exact solution, approximation can be conceptualized in a similar manner. That is, the resulting approximation of a function is found by applying some numerical procedure to the abscissa and ordinate elements of the data pairs.

Because the approximate time dependent function is a result of this application, it can be said that the approximation is a function of the data pairs. If this function can be
analytically differentiated with respect to time, then the time derivative is also a function of the data pairs. Stated alternatively, a function $F$ maps the data to a function of time:

$$\dot{x}(t_n) = \frac{d}{dt} \left\{ F[x(t_1), x(t_2), \ldots, x(t_n)] \right\}_{\text{evaluated at } t = t_n} \quad (\text{EQ. 1})$$

At any instance in time, the data history can be divided into two subsets. The first subset is that data which has already past, and is fixed indisputable knowledge. These values may be considered constant during this fixed moment in time. The second subset is that data which is presently being inputted. This data may be considered transient and not yet fixed knowledge. Iterative schemes may update the present value of the data, constantly modifying it until this last value has converged. This concept is referred to as the dependence property.

Therefore given a past set of data pairs, the time derivative of the latest value is a function of that value only. Here the function $f$ maps the last data point to the time derivative of that data point.

$$\dot{x}_n = f(x_n) \quad \text{at a fixed moment in time with a given history} \quad (\text{EQ. 2})$$

2.4 Assumptions

The product of this work is intended to provide a full nonlinear inverse solution. In this paper, trajectory expansion is developed using many assumptions. Some of these assumptions are necessary to make the problem manageable. Without them, a significant increase in rigorous development is required as well as an increased number of inputs.
These will be termed *hard assumptions*. The other assumptions are termed *soft assumptions*. They are included to ease development. Although there is no significant conceptual demand in relieving soft assumptions, practically, it reduces the workload required to verify the method. These can be removed in the future.

The following assumptions are hard assumptions. First, the vehicle-carried reference frame is parallel to the earth-fixed reference frame and inertial reference frame. This requires that the curvature and motion (rotation and revolution) of the earth are assumed to be negligible. For flight conditions near the earth surface, these are valid approximations. Second, the vehicle is considered to be a rigid body which is one of the primary assumptions for the mathematical equations of motion used herein (Etkin\(^7\)). Next, it is assumed that the atmosphere is still. This may be an obvious assumption, but it is essential for the wind axes relationship to the trajectory. Fourth, the plane's mass is assumed to be constant. This is a reasonable assumption for relatively short maneuvers that can exclude the consumption of fuel. This assumption and the first assumption are required for a simplified application of Newton's second law. The next assumption is that the drag on the aircraft is approximated by a drag polar formula. The exact form of this equation is introduced in 3.1 *Wind Axes Considerations*. This assumption has two results. First, it places great emphasis on the drag due to lift while lumping other effects into a single constant and ignoring still others. Second, it can only approximate the drag due to lift, its main emphasis.

Each of the preceding hard assumptions could be avoided with a significant increase in arithmetic difficulty and complete knowledge of the properties that are assumed to be zero. For example, if knowledge included the exact nature of fuel consumption, the exact motion of the earth, a complete description of the winds aloft,
and more globally valid mathematical expressions (as in Newton's second law), then these assumptions would not be required.

The final hard assumption may be the largest approximation as well as the most difficult to remove. It is assumed that the contributions of controls are strictly as moment generators. This means that the control surface contribution to lift, side-force, and drag, are assumed to be zero. Although, for some aircraft this assumption may be significant, it is key to the solution development.

The following assumptions are soft assumptions. In the case of a trajectory specified without $\mu(t)$, the craft is assumed to be in coordinated flight. This is tantamount to letting $\beta(t)$ be zero and by definition is a soft assumption since it can be removed by specifying $\mu(t)$. Next, $\alpha$ and $\beta$ are assumed small. This allows for two more assumptions. First, thrust will be considered in x direction of the wind axes only. This assumption is made to reduce the complexity of the nonlinear solving routines. Second, the forces on the airplane are simple linear functions of $\alpha$, $\beta$, $\dot{\alpha}$, $P$, $Q$, and $R$ based on the stability derivative representation. This assumption seems very restrictive in the light that a nonlinear inverse solution is desired. It can be relaxed in the future using a sophisticated data-base and table look-up procedure. Incorporating such functions will greatly complicate computational considerations. For this development, the maneuvers were hand picked such that the dependencies indicated would be valid. Verifying methodology is the primary goal of the work contained herein.

2.5 Formulation Objectives

With the preceding assumptions, this work intends to sequentially expand a trajectory into the complete state definition. The method to follow is presented in an algorithm format that has been coded into a FORTRAN program. Areas of concern,
including numerical method matters and assumption modifications, are discussed separately to leave the discussion of the algorithm unhampred with the details.
3. ALGORITHM FOR TRAJECTORY EXPANSION

Using the preceding assumptions, the development of the trajectory expansion into a full complement of states and state derivatives is a straightforward process and is based on methods outlined by Durham in reference 6. As was stated before, the input is the time history of the velocity vector of the aircraft's center of mass, \( V(t) \), which for this formulation is specified by \( V_c(t) \), \( \chi(t) \), and \( \gamma(t) \).

3.1 Earth Axes Quantities

Using the Euler angles, the velocity vector can be transformed from the wind axes' reference frame into the earth axes. This is accomplished by a change of basis matrix defined by \( \chi \), \( \gamma \), and \( \mu \). It is denoted by \( T_{EW} \) (Etkin prefers to use the letter \( L \), for these matrices\(^7\)):

\[
T_{EW} = \begin{bmatrix}
\cos \gamma \cdot \cos \chi & \sin \mu \cdot \sin \gamma \cdot \cos \chi & \cos \mu \cdot \sin \gamma \cdot \cos \chi \\
\sin \mu \cdot \sin \gamma \cdot \sin \chi & -\cos \mu \cdot \sin \chi & +\sin \mu \cdot \sin \chi \\
\cos \gamma \cdot \sin \chi & \sin \mu \cdot \sin \gamma \cdot \sin \chi & \cos \mu \cdot \sin \gamma \cdot \sin \chi \\
-\sin \gamma & \sin \mu \cdot \cos \gamma & \cos \mu \cdot \cos \gamma
\end{bmatrix}
\]  

(EQ. 3)
Since \( V \) described in the wind axes is by definition its magnitude along the \( x \)-axis, the vector can be described in the earth axes as:

\[
V_E = T_{EW} \cdot V_W = T_{EW} \cdot \begin{bmatrix} V_c \\ 0 \\ 0 \end{bmatrix}
\]  
(EQ. 4)

Notice that the components in the earth reference frame do not depend on \( \mu \) during the transformation. The earth coordinates of the vector are:

\[
\begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{bmatrix} = \begin{bmatrix} V_c \cdot \cos \gamma \cdot \cos \chi \\ V_c \cdot \cos \gamma \cdot \sin \chi \\ -V_c \cdot \sin \gamma \end{bmatrix}
\]  
(EQ. 5)

The resulting time history of the inertial velocities is differentiated to give a time history of the model's inertial accelerations. Newton's second law, assuming a constant mass, states that the components of the force vector in this inertial frame are proportional to the components of acceleration by the factor of mass. This leads to:

\[
F_E = \begin{bmatrix} F_{xE} \\ F_{yE} \\ F_{zE} \end{bmatrix} = m \cdot \begin{bmatrix} \ddot{x}_E \\ \ddot{y}_E \\ \ddot{z}_E \end{bmatrix}
\]  
(EQ. 6)
3.2 Wind Axes Considerations

The time histories of the forces which must act on the aircraft to produce the trajectory are therefore available from equation 6. These earth referenced forces are transformed back into their wind axes representations. The forces as represented in the wind axes are the forces that will give insight into angle of attack and sideslip angle. In the wind axes, the forces can be broken into the components of lift, drag, side-force, thrust and weight.

\[
F_W = \begin{bmatrix}
T_{xw} - D - m \cdot g \cdot \sin \gamma \\
T_{yw} - C + m \cdot g \cdot \cos \gamma \cdot \sin \mu \\
T_{zw} - L + m \cdot g \cdot \cos \gamma \cdot \cos \mu
\end{bmatrix} = T_{WE} \cdot F_E \quad \text{(EQ. 7)}
\]

The assumption on the direction of thrust eliminates the \( T_{yw} \) and \( T_{zw} \) terms.

\[
\begin{bmatrix}
T_{xw} - D - m \cdot g \cdot \sin \gamma \\
-C + m \cdot g \cdot \cos \gamma \cdot \sin \mu \\
-L + m \cdot g \cdot \cos \gamma \cdot \cos \mu
\end{bmatrix} = T_{WE} \cdot F_E \quad \text{(EQ. 8)}
\]

In this system, there are three equations and four unknowns, \( T_{xw}, C, L, \) and \( \mu \). The drag coefficient can be given as a dependent function of lift using the parabolic drag polar equation, so drag is not an unknown.

\[
C_D = C_{D0} + K \cdot C_L^2 \quad \text{(EQ. 9)}
\]
Consider that to this point \( \mu \) has not been necessary. However, now \( \mu \) must either be known or calculated.

3.3 Definition of Wind Axes Roll Angle

To uniquely solve the system of three equations, an extra equation is necessary. If \( \mu(t) \) is specified in the trajectory input, then the fourth equation becomes \( \mu = \mu(t) \). The system is now solvable. If \( \mu(t) \) is not given at this point, then another equation must be determined and comes from the coordinated flight assumption. The side-force is assumed zero and the fourth equation becomes \( C = 0 \). \( \mu(t) \) can now be identified from the second component of the wind axes force vector.

\[
\tan \mu = \frac{F_{X_E} \cdot \sin \chi - F_{Y_E} \cdot \cos \chi}{F_{X_E} \cdot \sin \gamma \cdot \cos \chi + F_{Y_E} \cdot \sin \gamma \cdot \sin \chi + (F_{Z_E} - m \cdot g) \cdot \cos \gamma}
\]  

(EQ. 10)

This relationship has two limitations. These are continuity, and the notion of "up". First, even using a quadrant specific arc tangent function \( \mu \) can only be specified from 0 to \( 2\pi \) or -\( \pi \) to +\( \pi \). If a maneuver such as a roll is attempted, then these limits will be exceeded and a large discontinuity will appear in the data. This can be corrected by a decision making algorithm to be discussed later. Second, this equation provides an ambiguous result in terms of the orientation of the airplane. The equation will shift by \( \pi \) radians using a quadrant specific arc tangent function if both the numerator and the denominator are multiplied by -1. The best solution to the problem is to orient the equation to give a correct assessment of \( \mu \) assuming that \( F_{Z_W} < 0 \) is equivalent to "up". In this formulation, \( F_{Z_W} \) was defined negative as pointing away from the earth. Therefore, positive lift would point in the "up" direction. In practical terms, this corresponds to specifying that the aircraft flies right side up in a straight and level
condition. However, an aircraft in a negative g pushover type maneuver will return a value of \( \mu \) that is shifted by \( \pi \) radians so that the aircraft can "pull g's" instead of "pushing over." This is an example where \( \mu \) is maneuver specific and therefore must be specified. Whether this angle is specified or calculated using the assumption, \( T_{W} \), C, D, and L are now specified time histories.

At this point, the flight condition and the aircraft characteristics become important in calculations. The altitude, \( V_c(t) \) and the planform area are used to calculate the force due to dynamic pressure.

\[
\bar{q} = \frac{1}{2} \cdot \rho \cdot V^2 \tag{EQ. 11}
\]

From this quantity, each force coefficient may be calculated similarly to the lift coefficient in the following manner.

\[
C_L = \frac{L}{\bar{q} \cdot S} \tag{EQ. 12}
\]

The time histories of any of these coefficients can be made available as needed. These force coefficients are the cornerstones to determining \( \alpha \) and \( \beta \). They can be matched with a priori knowledge gathered from wind tunnel and flight test data. Ideally, their dependencies transcend variations in velocity and altitude (approximately and disregarding compressibility). Because this knowledge does exist or may be approximated, it is possible to set up a system to recover \( \alpha \) and \( \beta \) from the time histories of the coefficients.
3.4 Determination of $\alpha$ and $\beta$

The investigation continues with some of the soft assumptions as described earlier. The aircraft's identity is hidden within the aerodynamic force and moment coefficients. Knowing how these vary with the conditions on the aircraft can unlock the details of the trajectory. The precise approach is bound to affect the completion of the task. Presented here is the foundation based solely on one approach. This approach uses wind tunnel derived stability derivatives to approximate the state dependencies around a reference point. The reference point of this study is straight and level flight at a trim angle of attack.

Each of the force coefficients are functions of the states and the state time derivatives of the aircraft. For this discussion, only the most prominent contributors are considered. These are:

\[ C_C = f(\alpha, \beta, P, R) \]
\[ C_L = f(\alpha, \dot{\alpha}, \mathcal{Q}) \]
\[ C_D = f(C_L) \quad \text{this is a hard assumption} \]

Note, the assumption was made that the control surfaces are primarily moment generators and therefore are not included in the force equations. Other dependencies are assumed negligible in this derivation, however, possible additions are discussed on page 54. In a more advanced application, the contribution from more subtle sources can be included.

The dependence property of numerical differentiation stated that the time derivative of $\alpha$ is a function of the current value of $\alpha$ only. Therefore, the lift coefficient can be written as a generalized function in $\alpha$ and $\mathcal{Q}$. In the following equation, $g$ is a non
specific operator that represents the dependence property which relates the time derivative of \( \alpha \) to a function of \( \alpha \).

\[ C_L = f(\alpha, g(\alpha), Q) = f(\alpha, Q) \quad \text{(EQ. 13)} \]

This function can be expressed as a modified Taylor series expansion. The first modification is to disregard all higher order terms. Second, note that the reference condition of each state is zero, so that the absolute values are equivalent to the variation from the reference condition. Finally, static and dynamic stability derivatives can be substituted for the partial derivative quantities.

\[ C_L = C_{L0} + C_{La} \cdot \dot{\alpha} + C_{L\ddot{a}} \cdot \ddot{\alpha} + C_{LQ} \cdot Q \quad \text{(EQ. 14)} \]

Most sources non-dimensionalize the dynamic derivatives corresponding to angle of attack rate and pitch rate by a longitudinal factor equal to the mean aerodynamic chord divided by twice the velocity. This factor simply needs to be included accordingly.

Additionally, \( C_{LO} \) can be modified to take into account the varying measures of angle of attack. This variety may stem from the labeling of a fuselage reference line rather than using the mean chord line. For example some define \( \alpha = 0 \) at \( C_L = 0 \). \( C_{LO} \) of course would not be necessary in this case. However, other conventions require including \( C_{LO} \) to balance the equation and can be modified according to the location of the desired fuselage reference line.

The side-force equation is more particular. Since stability derivatives are more often given in terms of body axes side-force, another translation must be utilized. This translation changes bases from wind axes to body axes via the angles \( \alpha \) and \(-\beta\).
\[
T_{bw} = \begin{bmatrix}
\cos \alpha \cdot \cos \beta & -\cos \alpha \cdot \sin \beta & -\sin \alpha \\
\sin \beta & \cos \beta & 0 \\
\sin \alpha \cdot \cos \beta & -\sin \alpha \cdot \sin \beta & \cos \alpha 
\end{bmatrix}
\] (EQ. 15)

The body y-axis component of aerodynamic force, \(Y\), is only dependent on the known aerodynamic forces in the wind axes forces and the value of sideslip. Regardless of whether sideslip has been set to zero or not, a second equation has become available. Equating a similar series representation of the side-force coefficient with the transformation of the aerodynamic force vector in the wind axes, more familiar stability derivatives can be used to define a nonlinear equation in \(\beta\), \(P\), and \(Q\). This results in the following equation.

\[
C_Y = \frac{-D \cdot \sin \beta - C \cdot \cos \beta}{q \cdot S} = C_{Y\beta} \cdot \beta + C_{YP} \cdot P + C_{YR} \cdot R 
\] (EQ. 16)

Similar to the lift coefficient, the dynamic derivatives may be non-dimensionalized by a factor equal the span divided by twice the Velocity. To correct the above expression simply multiply those quantities by this factor.

Clearly, there exists a nonlinear system of equations with two equations and five unknowns. These unknowns are \(\alpha\), \(\beta\), \(P\), \(Q\), and \(R\). Equations 14 and 16 make up the state's relationship to the forces.

3.5 Determination of \(P\), \(Q\), and \(R\)

Three more equations are necessary to solve the system. These will come from kinematic relationships that consist of axes system identities. In the literature, there is a clear derivation of body axes angle rates to body axes Euler angle rates. The derivation is equally applicable to the wind axes rates and angles. With the time history of the
Euler angles available, they are differentiated numerically and used in the corresponding wind axes equation to give the wind axes angle rates.

\[
\begin{bmatrix}
P_w \\
Q_w \\
R_w
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & -\sin \gamma \\
0 & \cos \mu & \sin \mu \cdot \cos \gamma \\
0 & -\sin \mu & \cos \mu \cdot \sin \gamma
\end{bmatrix}
\begin{bmatrix}
\dot{\mu} \\
\dot{\gamma} \\
\dot{\chi}
\end{bmatrix}
\] (EQ. 17)

Subsequently, using equation 15, the wind axes rates are related to the body axes angle rates by the following relationship.

\[
\begin{bmatrix}
P \\
Q \\
R
\end{bmatrix} =
\begin{bmatrix}
P_w + \dot{\alpha} \cdot \sin \beta \\
Q_w + \dot{\alpha} \cdot \cos \beta \\
R_w - \dot{\beta}
\end{bmatrix}
\] (EQ. 18)

With the dependence property of the numerical differentiation scheme, \( \dot{\alpha} \) and \( \dot{\beta} \) are considered to just be functions of \( \alpha \) and \( \beta \) respectively. The system defined by equations 14, 16, and 18 can now give five nonlinear equations for the five unknowns, \( \alpha \), \( \beta \), \( P \), \( Q \), and \( R \). This set of nonlinear equations can be solved in a fixed-point iteration scheme. A detailed description of this method is given in the discussion of numerical methods on page 48.
3.6 Body Axes Velocity Components: $U, V, W$

Once $\alpha$ and $\beta$ are determined $U$, $V$, and $W$ are straightforward from a transformation from wind to body axes. Using equation 15 and the velocity vector representation in the wind axes, the body axes components become:

\[
\begin{bmatrix}
U \\
V \\
W
\end{bmatrix} =
\begin{bmatrix}
V_c \cdot \cos \alpha \cdot \cos \beta \\
V_c \cdot \sin \beta \\
V_c \cdot \sin \alpha \cdot \cos \beta
\end{bmatrix} \quad \text{(EQ. 19)}
\]

3.7 Euler Parameter Extraction

$P$, $Q$, $R$, $U$, $V$, and $W$ can now be differentiated numerically. A complete state vector and derivative vector have now been defined or are available considering the complete set of Euler angles. As was previously indicated, the state vector desired uses Euler parameters instead of Euler angles to define the body axes direction. In order to acquire these quantities, another nonlinear system of equations is defined from the change of basis matrices. Recall that $T_{BE}$ is the transformation from earth to body axes. Another way to get from earth to body axes system is to use an intermediate transformation. The two step transformation will progress from earth to wind and then from wind to body.\(^7\) Mathematically, the transformation is represented by a matrix multiplication.

\[
T_{BE} = T_{BW} \cdot T_{WE}
\]

$T_{BE}$ can be represented as a function of the Euler parameters $q_0$ through $q_3$ just as it can be defined in terms of Euler angles. The matrix as represented by these parameters is:
$$T_{BE} = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2 \cdot (q_1 \cdot q_2 - q_0 \cdot q_3) & 2 \cdot (q_1 \cdot q_3 - q_0 \cdot q_2) \\
2 \cdot (q_1 \cdot q_2 - q_0 \cdot q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2 \cdot (q_2 \cdot q_3 + q_0 \cdot q_1) \\
2 \cdot (q_1 \cdot q_3 + q_0 \cdot q_2) & 2 \cdot (q_2 \cdot q_3 - q_0 \cdot q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{bmatrix}$$

(EQ. 20)

Euler parameters can be defined to described different vector relationships. For this derivation, these parameters are used to substitute for the description provided by the body axes Euler angles. The following equations are obtained from the general definitions and their derivation can be found in the literature.

$$q_0 = \cos \frac{\psi}{2} \cdot \cos \frac{\theta}{2} \cdot \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cdot \sin \frac{\theta}{2} \cdot \sin \frac{\phi}{2}$$  \hspace{1cm} (EQ. 21)

$$q_1 = \cos \frac{\psi}{2} \cdot \cos \frac{\theta}{2} \cdot \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \cdot \sin \frac{\theta}{2} \cdot \cos \frac{\phi}{2}$$ \hspace{1cm} (EQ. 22)

$$q_2 = \cos \frac{\psi}{2} \cdot \sin \frac{\theta}{2} \cdot \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cdot \cos \frac{\theta}{2} \cdot \sin \frac{\phi}{2}$$  \hspace{1cm} (EQ. 23)

$$q_3 = \sin \frac{\psi}{2} \cdot \cos \frac{\theta}{2} \cdot \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \cdot \sin \frac{\theta}{2} \cdot \sin \frac{\phi}{2}$$ \hspace{1cm} (EQ. 24)

$T_{BW}$ is a function of $\alpha$ and $\beta$ as given in equation 15 and $T_{WE}$ is the transpose of the matrix given in equation 3 as a function of $\chi$, $\gamma$, and $\mu$. Setting the matrices equal and carrying out the multiplication gives nine elements on each side of the equation that can be set equal.

However, this system represents nine equations with only four unknowns. Selecting four terms from the 3x3 matrix on the left side and the corresponding entries of
the result on the right-hand side offers a set of nonlinear equations which will determine unique values of the Euler parameters. The existence of a solution is dependent upon the terms selected. It is necessary to select terms which give unique information. One adequate combination is the upper right hand 2x2. This consists of the (1,2), (1,3), (2,2), and (2,3) entries.

A Newton's method solver with analytical derivatives is an adequate method to find a solution to this system. However, convergence will fail if these initial values occur at particular values that cause specific pitfalls in the solving routine. A detailed description of this method is given on page 54.

3.8 Complete State Definition

The entire state vector has been defined. The numerical differentiator can be used to calculate the complete state derivative vector. As was stated previously, only the time derivatives of the angle rates and the body x-axis component of velocity need be differentiated. The trajectory expansion algorithm has completed its task by computing the necessary input for the inverse solution. This data is available for every sampling instance and makes use only of prior knowledge.

The key assumptions that this algorithm relies on are the effectiveness of the numerical differentiator and the soft assumptions as stated on page 7. The detailed analysis of the differentiator is discussed in the next section.
4. NUMERICAL DIFFERENTIATION

4.1 Design Goals

It is necessary in the above formulation to be able to calculate a derivative of a state based on its history. Numerical differentiation is inherently difficult because small errors in data lead to larger errors in the derivative. Gerald and Wheatley\(^9\) have suggested that one approach to finding numerical derivatives to noisy data is to smooth the data and then take the derivative of the resulting function. This derivative can be taken at the last data point which is necessary for the algorithm to operate in real time. Additionally it was asserted without explanation on page 6 that the numerical derivative of the data history was a function of the last data point only. It can be shown that a least squares curve fit differentiated at the endpoint exhibits these properties as well as other desirable characteristics.

Sometimes the data streams are not well behaved. Complications such as occurrences of piecewise smooth data or noisy measurements will hamper any derivative scheme including the least squares method discussed here. The goal is to create a method such that it is accurate when the data is continuous; it can transition smoothly between two piecewise continuous histories; and should the data contain a high density of discontinuous pieces (this may be considered noise), the derivative will resist tracking the discontinuities (noise).
4.2 Data Pairs and Relation in Time

Consider a generic state, $x$, which may be any one of the states. If the value of $x$ is changing in time, then a series of $n$ data pairs of the data, $x$ at a particular time, $t$, will make up a time history of $x$. Or:

$$\text{time history}_\eta = \{ (t_1, x_1), (t_2, x_2), \ldots, (t_n, x_n) \}$$

where $(t_i, x_i)$ is considered one data pair. The subscript $\eta$ on the time history identifies where along the data stream a curve is being fit. The complete system can be expressed in the following statements. The term labeled "$t_i$" will be referred to as the time element and the term labeled "$x_i$" will be referred to as the data element. Together they make up one data pair. A string of $n$ data pairs make up a subset of the time history which is labeled $\text{time history}_\eta$.

When a measurement is taken, a new data pair, $(t_{n+1}, x_{n+1})$, is available and is referred to as the measurement instance. As the next instance of $x$ occurs, it can be added on to the end and the oldest instance can be removed, thereby constantly maintaining $n$ values in the sampling.

$$\text{time history}_{\eta+1} = \{ (t_2, x_2), (t_3, x_3), \ldots, (t_\eta, x_\eta), (t_{\eta+1}, x_{\eta+1}) \}$$

Fixing the sampling rate means that the spacing between each successive time element will be identical. If each individual spacing is constant, then the relationship between any two data pairs of successive time history will also remain constant.
For example, the time between $t_1$ and $t_n$ in time history $\eta$ is equal to the time between $t_2$ and $t_{n+1}$ in time history $\eta+1$. Since time is relative, the history can be set to an arbitrary origin and all data pairs can be set relative to that new origin. The previous history might just as well been written as:

$$\text{time history}_{\eta+1} = \{(t_1,x_2), (t_2,x_3), \ldots, (t_{n-1},x_n), (t_n,x_{n+1})\}$$

as long as $t_1$ in this history is tracked relative to an absolute time. The nature of the curve has not changed, only its relation to other events has. In reality, the curve has translated along the time axis by one time step, but it is identical in all other respects, most importantly the time derivatives. As long as the origins of the time histories are used to specify events, there is no need to use the actual time elements in analysis of the nature of the curve. This will help reduce the computational demands on the algorithm as well as aid the dependence property. Figure 1 illustrates the concept that resetting the time origin does not change the nature of the derivative.

To specify all time elements, it is necessary to know only the sample size, the sampling rate and a fixed value for one time constant, the origin. All others will be measured from that fixed value.
4.3 Least Squares

It is assumed that the data that is being fitted is not particularly unique in character. Ideally, if this data is smooth, a polynomial of undetermined order could exactly model the curve represented by a time history. If there is a degree of noise in the data, a least squares curve can give its closest representation (given the format of the function). For a set sample size, $n$, and desired polynomial fit of order, $k$, a least squares curve is found by solving a $k+1$ order linear system. The requirement is that $n$ must be greater than or equal to $k+1$.

The following equations are representative of the least squares polynomial curve fitting as described in references 10 and 17. Some of the notation has been modified to better represent the work included herein. To begin, the system can be defined by the
least squares A matrix and b vector. The solution is a vector, c, whose elements are the coefficients of the polynomial. The system to solve is:

\[ A \cdot c = b \]  

(EQ. 25)

The A matrix is a function of the abscissa value of the data set only. In this derivation, the abscissa is the time element, \( t_i \) for \( i = 1 \) to \( n \), and the A matrix is:

\[
A = \begin{bmatrix}
\sum_{i=1}^{n} t_i^0 & \sum_{i=1}^{n} t_i^1 & \cdots & \sum_{i=1}^{n} t_i^k \\
\sum_{i=1}^{n} t_i^1 & \sum_{i=1}^{n} t_i^2 & \cdots & \sum_{i=1}^{n} t_i^{k+1} \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{i=1}^{n} t_i^k & \sum_{i=1}^{n} t_i^{k+1} & \cdots & \sum_{i=1}^{n} t_i^{2k}
\end{bmatrix}
\]  

(EQ. 26)

For the time histories considered, the time origin is being reset to a constant reference point for every data occurrence. The time element for each data pair is therefore constant from one instance to the next, so the resulting A matrix is also constant. This avoids the problem of having to recalculate this matrix at every time step. Also, A can be tailored according to its condition number, by varying the number and quality of the time elements. The quality refers to the spacing defined by the sampling rate and the positioning of the first time element as suggested in figure 1.
The right hand side is the \( \mathbf{b} \) vector and must be calculated at every time step because it is dependent on the data elements. It is given by:

\[
\mathbf{b} = \left\{ \sum_{i=1}^{n} x_i, \sum_{i=1}^{n} t_i \cdot x_i, \sum_{i=1}^{n} t_i^2 \cdot x_i, \ldots, \sum_{i=1}^{n} t_i^k \cdot x_i \right\}
\] (EQ. 27)

The solution vector defines the polynomial approximation of the data, \( \hat{x}(t) \). It is given by

\[
\mathbf{c} = \left\{ c_0, c_1, c_2, \ldots, c_k \right\}
\] (EQ. 28)

These coefficients define the least squares polynomial which has the form:

\[
\hat{x}(t) = c_0 + c_1 \cdot t + c_2 \cdot t^2 + c_3 \cdot t^3 + \ldots + c_k \cdot t^k
\] (EQ. 29)
This function is denoted by a "\^\#\". This signifies that the function is a fitted curve and only approximates the data. The next step is to differentiate the new function analytically. The resulting time derivative function is

\[
\hat{x}(t) = c_1 + 2 \cdot c_2 \cdot t + 3 \cdot c_3 \cdot t^2 + \ldots + k \cdot c_k \cdot t^{k-1}
\]  

(EQ. 30)

This function can be evaluated at any time, t, but is specifically valid within the limits of the time history. For the expansion algorithm, this function is evaluated at the endpoint, t_n.

4.4 Dependence Property and Least Squares

In light of this discussion, the dependence property of the numerical differentiation scheme will be reiterated and explored in further detail. It was asserted on page 6 that the numerical differentiation process can be thought of as a function of the last data point entered. This can be demonstrated by using the method of a least squares curve fit as defined above and manipulating the terms. The key prerequisites to the manipulation require that there is a fixed sampling rate so that the time scale origin may be reset as described in 4.5 Data Pairs and Relation in Time.

To begin, consider the system that is to be solved for the coefficients of the least squares polynomial fit. This was shown in equation 25 where A and b were given by equations 25 and 27 respectively. The coefficient matrix, c, can be written as

\[
c = A^{-1} \cdot b
\]  

(EQ. 31)

Now, it can be shown that the b vector can be written as the sum of two quantities. One which contains the last data point and the other that contains the rest of the previous history.
Recall the expression for \( \mathbf{b} \), and see that it can be dissected into separate components in the following manner:

\[
\mathbf{b} = \begin{bmatrix}
\sum_{i=1}^{n} x_i \\
\sum_{i=1}^{n} t_i \cdot x_i \\
\vdots \\
\sum_{i=1}^{n} t_i^k \cdot x_i
\end{bmatrix} = \begin{bmatrix}
1 & 1 & \cdots & 1 & 0 \\
t_1 & t_2 & \cdots & t_{n-1} & 0 \\
t_1^2 & t_2^2 & \cdots & t_{n-1}^2 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
t_1^k & t_2^k & \cdots & t_{n-1}^k & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots \\
x_{n-1} \\
x_n
\end{bmatrix} + \begin{bmatrix}
1 \\
t_n \\
t_n^2 \\
\vdots \\
t_n^k
\end{bmatrix} \cdot x_n
\]

(EQ. 32)

It has already been shown that when the time scale is made constant by continuously resetting the origin, the least squares \( \mathbf{A} \) matrix is a constant. Although, the matrix is ill-conditioned, it is non-singular. Therefore, it has an inverse and, in addition, is constant by the previous logic. Equally constant, are both the \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) quantities in the above equation. Since the first \( n-1 \) data elements in the time history are established knowledge, they are then unable to change and are also held constant during a single measurement instance. These will of course change at the next data occurrence. So, the only variable in either the \( \mathbf{b} \) or \( \mathbf{c} \) vectors is the last data point entered.
Equation 30 defined the derivative of any time contained within the limits of the history in terms of these coefficients. It can be rewritten as

\[
\dot{x}(t_n) = \begin{bmatrix} 0 & 1 & 2 \cdot t_n & 3 \cdot t_n^2 & \cdots & k \cdot t_n^{k-1} \end{bmatrix} \cdot [\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}] = [0 \ 1 \ 2 \cdot t_n \ 3 \cdot t_n^2 \ \cdots \ k \cdot t_n^{k-1}] \cdot c
\]

**(EQ. 33)**

Since the last time element has been defined by the resetting procedure, it too is a constant. The time derivative of the last data entry depends only upon the order of the polynomial and the data element. This is the justification for the dependence property and the basis for the numerical methods that employ it.

As a side note, it can also now be seen that the functional relationship is linear at every data occurrence. Consider that the derivative can be written as

\[
\dot{x} = \begin{bmatrix} 0 & 1 & \cdots & k \cdot t_n^{k-1} \end{bmatrix} \cdot A^{-1} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ 0 \end{bmatrix} + b_1 \cdot x_1 + b_2 \cdot x_n
\]

**(EQ. 34)**

This equation is in slope-intercept form where \( b_1 \) uniquely specifies the product that is the intercept and \( b_2 \) specifies the product that is the slope.
4.6 Least Squares Parameter Considerations

The questions to consider are what sample size and what order polynomial are required to best represent the data streams that are likely to be encountered. It is desired to keep the algorithm as simple as possible. This will allow for easier programming and less computer time. Computer CPU time is at a premium due to its initial manufacturing expense and the calculative limitations of the system. It is desired to use this solution in real time. A complex solution would either require expensive hardware or simply fail to run fast enough.

The first consideration relates to task performance which refers to how quickly and efficiently the algorithm can be performed. Examination of the least squares $\mathbf{A}$ and $\mathbf{b}$ matrices shows that a low sampling and low order polynomial greatly reduce the calculation requirements on these two quantities. The dimensions are smaller and each element requires fewer calculations. In addition, solving a lower order system, (by inverting the $\mathbf{A}$ matrix or a Gaussian solution) is orders of magnitude cheaper time-wise for each order reduced.

In contrast, there are also computational motivations to having a large time history. The least squares $\mathbf{A}$ matrix is ill-conditioned by nature.\textsuperscript{10} This means that there can be significant round-off error when the system is solved. When optimizing the condition number with respect to the size of the sample, it is found that there is an exponential decrease in condition number with increasing sample size. In general, this would not be the case, but in this derivation the time components are measured in milliseconds. The magnitudes are less than 1. When raised to a power, these values become very small. But as data entries are added, the individual matrix elements increase in a manner that decreases the condition number.
Although there is no bearing on the sample size, some mention should also be given to the time elements of the samples. It was previously stated that fixing the time elements of the history aids in calculations without adversely affecting the time derivative. The location of the first time element will fix the relationship of the rest of the time elements to the natural origin where real time is equal to zero. This initial time element will be called the *time scale origin*. It would be reasonable to suppose that there may be some effect of the position of this origin.

Figure 2 illustrates optimizing the A matrix with respect to the sampling size and the location of the time origin. The vertical axis is the base ten logarithm of the condition number that results from sampling sizes ranging from 4 to 27 and the location
of the time scale origin which ranges from zero to starting at a point such that equal data samples lie both above and below zero. Obviously, the condition number is influenced more by the sample size. In fact, this illustration represents a 24,000% decrease in condition number as a result of increasing the number of data entries from 4 to 27.

The optimization shows that the condition number is the least for a given sampling size if the sampling's time scale is centered about zero. Specifically, there must be equal entries whose numerical value are greater than zero and those whose values are less than zero.

It can be seen from equation 25, that an odd number of equally spaced time elements in this configuration would generate a zero in every other element of the A matrix. In general, the time scale origin should be set equal to

\[
t_1 = -\left(\frac{n - 1}{2 \cdot \text{sampling rate}}\right), \text{ for odd } n = \# \text{ of samples} \quad \text{(EQ. 35)}
\]

Again this will improve the condition number although its effect is not as dramatic. Additionally, the matrix is sparse which can aid in numerical computations.

The second consideration centers around the algorithms ability to match the data time history. It can be seen that a large time history fitted with a low order polynomial can wash out the real time history. For example, suppose it was desired to model a short period oscillation with a period of 2 seconds. If the sampling size also covered a time period of 2 seconds and the polynomial order was 1, the oscillation would be completely lost. Least squares would approximate the oscillation with its mean value. The derivative would be zero, which, in general, is not true. Figure 3 illustrates this point.

This is similar to an aliasing affect caused by violating the sampling theorem. The sampling theorem states that a bandlimited signal of finite energy with no frequency
Figure 3 Lost Dynamics

Higher than $f_m$ hertz may be completely recovered from its samples taken at the rate of $2f_m$ per second (Barkat). In this sense, there must be a minimum number of samples to recover the signal. In the case of the polynomial fit, the order must be of certain degree no matter how many samples there are. Choosing the polynomial degree is in a sense like choosing the "sampling rate" in the spirit of the sampling theorem. Choosing a third order polynomial is like reducing the "sampling rate" to four samples (see page 26).

On the other hand, if the order of the polynomial is too large, a phenomenon sometimes referred to as the "zigzag" or "wiggle" problem may develop. It is also known as instability. This occurs when the data does not exactly fit any polynomial function. The least squares method will add numerous inflection points to reduce the
error instead of using a lower order polynomial to smooth out the data. This resembles a wiggle. Therefore a dilemma exists over the order of the polynomial.

4.7 Least Squares Parameters

A solid design requirement would be that the least squares approximations accurately model at least the natural modes of the aircraft. Forced oscillations due to structural modes may occur at a somewhat higher frequency, but at some point, these and similar dynamics lose their significance especially in this rigid body concept.

Regardless of the sample size, the curve fit must be able to include a point of inflection. Even if the data is non-oscillatory, any non-monotonic behavior will require one. A third order polynomial is required because it is the lowest order polynomial that includes a point of inflection.

Perhaps the best candidate to examine is the short period. MIL-F-8785C\(^1\) details frequency and damping requirements to meet level 1 flying qualities. Hypothesizing that aircraft manufactures seek to produce aircraft with these flying qualities, 8785C provides a reasonable area of search for desired oscillation modeling. For example, level 1 for 10 g's/degree n/\(\alpha\) is to have a maximum short period natural frequency of 6 radians per second. If this is indicative of the type of dynamics that are required for fitting, then the least squares fit can be designed to it. That short period natural frequency corresponds to a period of about 1 second. Since there are two inflection points for every period, half the period or 0.5 second will contain only one. This should roughly guide the largest sampling size allowed if a third order polynomial is to be used. The oscillation cannot be accurately modeled with a much larger sample size.

On the other hand, for a third order polynomial, the least squares method demands at least four points. The minimum time spanned should therefore be three
Table 1  Design Specifications for Least Squares Formulation

<table>
<thead>
<tr>
<th>Time History Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 0.5 seconds of data spanned (Roughly).</td>
</tr>
<tr>
<td>• Time origin reset to have equal number of points above and below zero</td>
</tr>
<tr>
<td>• Odd number of data pairs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Polynomial Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• At least a 3rd order polynomial, ( k \geq 3 )</td>
</tr>
<tr>
<td>• Larger history to reduce condition number</td>
</tr>
</tbody>
</table>

divided by the sampling rate. It was already shown that for the type of numbers used in calculating the \( A \) matrix, the larger sampling has a more desirable condition number.

A band has now been defined where the size of the sample must fall. Table 1 summarizes these specifications for designing a least squares differentiator.

Figure 4 demonstrates using a least squares approximation on a smooth curve. A sine wave with a period of 1 second is differentiated both analytically and using a least squares method of the above parameters. This figure shows the comparison of the resulting derivatives which are cosine waves. The discrepancy that appears at the peaks gets larger as the number of entries increases and goes to zero as the entries decrease. It is apparent that the derivative needs to be tailored to individual needs.
4.8 Smoothness of Data Streams

Swokowski\textsuperscript{15} defines a smooth curve as one whose parametric representation has continuous derivatives which are never simultaneously zero. It is reasonable to expect that the physical laws that govern the dynamics of aircraft will be smooth in a still atmosphere. Yet, considering step control inputs, course sampling rates or noise in the measuring sensors, smoothness is certainly not the rule. Hopefully, the data will exhibit piecewise smoothness where it is smooth over some interval, $t_0 < t < t_1$.

The purpose of finding a function that will fit the data is so that the derivative can be calculated at the right hand endpoint, $t = t_n$. If our data is only piecewise smooth
then our derivative may or may not be valid at its endpoint, \( t_n \). Figure 5 illustrates this by showing an example where there is a jump in derivative at 0.1 second.

In this case, an ordinary polynomial cannot describe the exact nature over the entire interval. In fact, at the point of discontinuity, the data may not be subject to the same laws as the previous data. Consequently, the point itself is not a product of the history, and trying to apply the concepts implicit in the assumptions would be inaccurate. Instead, it would be desirable to be able to transition between the two pieces where these concepts do apply in a continuous fashion.
Figure 5 Example of Data with Discontinuous Derivative

4.9 Data Containing a Single Point of Discontinuous Derivative

Considered in this section, is the case where smooth data intervals are joined together such that the derivative of the data is discontinuous at the joint. This point will be referred to as a corner. For example, suppose the data is made up of two linear pieces as in figure 5. Analytically, the derivative within the limits of either piece would be a constant corresponding to the slope. But at the corner, the derivative may be either of those slopes depending on unspecified boundaries of the pieces. When a polynomial is fit across these boundaries, a decaying oscillation develops that lasts for the duration of the time that the corner is contained in the history. This oscillation is evidence of the "zigzag" problem previously discussed. The nature, especially the period, of the
Figure 6 Oscillations Across Corners

oscillation depends upon the order of the polynomial. Figure 6 demonstrates this oscillation on the example above using the derivative of a third order polynomial.

Because each order polynomial has a unique oscillatory nature, it was found that different orders bounded a curve that displayed the desired transition properties aforementioned. A weighted average of these polynomials was designed to capture this asymptotic transition while still retaining adequate differentiation results.

The weighting takes advantage of two facts. First, the first order polynomial, namely a straight line, has a stabilizing quality. By its nature, it lags the data and does not react aggressively to tight curvatures. Secondly, in most cases, the highest order fit desired is generally a third order. The least squares result of trying to fit a large order
polynomial to data produced by a lower order polynomial is that the coefficients of the higher powers approach zero when the data is smooth. When the data is not smooth, the oscillations resulting from instability of the higher order polynomials is averaged out.

Define any particular endpoint time derivative of a least squares polynomial of order \( k \) as \( \dot{x}_k \). Specifically, the least squares polynomial fit of the data is of order \( k \). Therefore, the derivative is actually of order \( k-1 \). The weighted average derivative that results in continuous transition across corners is given by the following equation.

\[
\dot{x}_{\text{avg}} = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \frac{\dot{x}_j}{j} \quad m = \text{largest order polynomial} \quad (\text{EQ. 36})
\]

Notice that by the summation, the \( \dot{x}_{\text{avg}} \) is of order \( m-1 \). Not only does the computer code have to solve an \( m+1 \)th order system, but it must solve for every order less than that as well. This dramatically increases the calculating demands on the computer. The benefits of the increased ability to transition must be weighed against the additional requirements. Using the example of a corner in figure 5, figure 7 shows the success of the weighted average derivative in transitioning between linear regions.
Figure 7 Transition Across Corners

Special precautions apply to the weighted average derivative. Including a linear fit (first order polynomial) in the weighting has both a positive and a negative effect. The positive effect is that it provides stability across the corners as demonstrated. The negative effect is that it introduces a time lag when the data is not made of straight lines. The larger the curvature (or smaller the radius of curvature\textsuperscript{16}) is, the larger the error due to the lag. The averaging can be done while removing the dependency on the linear fit. This significantly increases the accuracy of the derivative, but it will either decrease the stability of the transition or it will increase the order of polynomials required, thereby
increasing the computational intensity required. Another tradeoff situation is present. For these purposes, the linear dependency was kept.

There are two thresholds that must be observed. In the original least squares formulation, it was noted that there must be $k+1$ points to calculate a polynomial of order $k$. So, to calculate the weighted average derivative where the largest order of the least squares polynomial fit was $m$, there must be at least $m+1$ points. However, there is a secondary consideration. Recall that the period of oscillations across a corner depended on the order of the polynomial. If the frequency of the higher order oscillations is too great, the averaging loses its effectiveness in canceling out the oscillations. This may be because the peaks of the high oscillation polynomials may lie between set time elements and therefore are never seen in the averaging. Therefore, a significant increase in the number of points is necessary. The exact quantity of points depends on the order but may be approximated by $2 \cdot m$.

The second threshold is that the highest order required depends on what degree of canceling is desired. The higher this order is, the more canceling will occur. This results in a less oscillatory approach, but an increased demand on computation ability.

Another concern that affects this form is the frequency of corner occurrence. If the corners lie too close to each other then large pulses or spikes in the derivative may appear. The size of the pulses depends on the magnitude of change in direction at the corner. As an example, suppose the data has two corners very close to each other such that the first corner represents a large increase in slope and the second represents a change back to the original slope. As the time between the two corners goes to zero, the data assumes a step input character. The weighted derivative captures the corresponding derivatives impulse character because of the initial reactions to the corner (see figure 7). This may or may not be desirable depending on sensitivity desired.
When these step inputs correspond to noise, then obviously it is not desirable. The differentiation scheme is tracking the noise too closely and erroneous answers result. The data can be considered noisy when the density of corners reaches a degree such that the weighted average does not have time to adequately respond and transition across the corners. This noisy character is not specifically targeted as measurement noise. The data may be from true measurements of a turbulent atmosphere, but this does not conform to the predetermined concept of the data's dynamics.

4.10 Noise Ridden Data

If the data is noisy, the least squares differentiation technique, including the weighted average concept, will produce noisy derivatives. This is another symptom of the "zigzag" problem and because the differentiator was designed to react quickly to corners. It has no method for determining the status of the last point whether it is noise or true data. The most immediate solution is that if the data stream is known to be noisy, using a high sampling rate and a first order polynomial fit will give the best results.

Figure 8 illustrates the limited ability to smooth using a weighted average least squares derivative. The input was a random number between ±0.5 added to a cosine wave. The figure compares a backward difference method, a averaged least squares polynomial using 79 entries (used to magnify improvement), and the derivative of the curve without any noise added in. Although there is a level of smoothing to the weighted average it is still noisy enough to create problems in the second derivative. This smoothing ability naturally improves with the number of entries used but this also causes problems previously discussed including time lag.
Figure 8 Differentiating Noisy Input

The other alternative is to pre-process the data with a filtering technique. Filtering techniques are not considered in this work because of the increase in complexity. In general, either information about the type of noise or complete histories must be defined in order to apply these techniques. Since the scope of this investigation was limited to methods which could be applied simply in a real-time fashion, these techniques are left for future investigation.
Table 2 Differentiator Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Time Derivative of Least Squares Polynomial Fit</td>
</tr>
<tr>
<td>Order</td>
<td>Weighted Average - highest order $m = 13$</td>
</tr>
<tr>
<td>Sampling Size</td>
<td>$n = 27$ entries (based on 50 Hz sample)</td>
</tr>
<tr>
<td>Fixed Time Scale Origin</td>
<td>$t_1 = -0.26$, $\Delta t = 0.02$, $t_{14} = 0$, $t_{27} = 0.26$</td>
</tr>
</tbody>
</table>

4.11 Differentiator Final Parameters

The previous discussions have addressed the most prominent issues of numerical differentiation. The final version of the least squares numerical differentiation method was decided upon taking these issues into consideration. The specific differentiator parameters are listed in the table 2.
5. NUMERICAL METHODS

5.1 Topics for Discussion

Although the majority of effort was spent in defining the methodology for the differentiation scheme, the trajectory expansion algorithm depends very heavily on other numerical methods. These methods are focused on solving systems of equations for unknowns. The trajectory expansion algorithm asserted three different instances of systems to be solved. These systems are nonlinear and, in cases, difficult to express analytically.

The first is a very simple system and requires little consideration. The system is discussed on page 13 and concerns the determination of the wind axes forces. The system is easily reducible to two individual linear equations. The only nonlinearity occurs in one of the assumptions which offers no additional complexity.

The next instance of a nonlinear system occurs on page 18, in the determination of $\alpha$, $\beta$, $P$, $Q$, and $R$. The system is complicated because of the trigonometric dependence on $\alpha$ and $\beta$ and the functional dependence as described by the dependence property on page 6. However, because the stability coefficient representations allow these two quantities to be set apart easily, a fixed point iteration scheme is well suited to the task.

The last nonlinear system to be solved was discussed in extracting the Euler parameters on page 20. This instance is a classic for solving nonlinear equations. The elements of the Euler parameter transition matrix are explicitly defined in terms of the unknowns. This makes this system well suited for a Newton Method solution using a Jacobian matrix with exact derivatives.
Additionally, it was noted that the angle \( \mu \) must be kept track of if it is not an explicit input. This will keep large discontinuities from appearing in the data. The method for tagging this angle as used in the trajectory expansion algorithm is discussed in depth.

5.2 Wind Axes Forces System

The first system requires little exploration. There are two paths the input can follow. Either \( \mu \) is specified at every occurrence or the wind axes side-force is assumed to be zero. If the latter is the case, then \( \mu \) can be found from equation 10.

Once this is done the forces in the inertial reference frame are transformed back into the wind axes. To avoid unnecessary rigor, the following order of calculations is suggested. First, if \( C \) (side-force) has not been declared to be zero, calculate it by examining the second element of the vectors on each side of the equation. Isolate \( C \) by adding the weight component of the force to the right hand side. \( C \) is now an available time history. Next, do the same for the lift component, \( L \), which is also available as a time history. Use \( L \) to calculate \( C_L \) and then \( D \) using the drag polar formula. Finally, a time history of \( T_{X_W} \) is gotten from the other known quantities in the first element. The wind axes force decomposition is complete and these can now be considered available time histories.

5.3 Aircraft Incidence and Angle Rates Using Fixed Point

As was indicated before, a solution to a nonlinear system of equations is required to find \( \alpha, \beta, P, Q, \) and \( R \). Recall, to find these, there was a system of five nonlinear equations that need to be solved. These consisted of equations 14, 16 and the vector equation 18 as outlined on page 18. The following discussion is dedicated to
investigating the intricacies of the solution mechanism. Included are solution type, computational demands, initial approximations and miscellaneous considerations.

Two nonlinear system algorithms were considered. These were the Newton's method and the fixed point algorithm. In general, both a Newton's method or a fixed point type scheme are well suited to this task. Computationally, there is no significant savings in either the number of calculations or the CPU time required. The Newton method scheme requires fewer iterations to converge, but each iteration requires a numerical approximation to the Jacobian matrix, which increases the number of calculations per iteration. Also considering that the Jacobian matrix is a 5x5 matrix, the demands on the Gaussian elimination subroutine also increases the number of calculations per iteration.

Secondly, a significant increase in the robustness of solution does not exist. Both methods had equal difficulty handling more complex situations. A more complex situation may be understood to be when $\beta$ is very large ($25^\circ$ may be considered large enough to cause a problem) or the level of noise in the data reaches a significant level. The exact threshold of these quantities are complicated by multiple dependencies.

For this development, the fixed point method was chosen, because it did not require yet another numerical derivative as would be necessary for the Jacobian matrix.

First, it is important to examine some of the other symbols that appear in the system of equations. Equation 17 defines the wind axes angle rates that are arguments of the body angle rate equations. This vector is dependent on $\chi$, $\gamma$, and $\mu$ and their time derivatives. The angles themselves are inputs except perhaps $\mu$ which will already have been calculated if it is not an input. The numerical differentiator computes the time derivatives as they are input and therefore these are available. Therefore equation 17 is fully defined and may be considered a constant across iterations.
Although the system was described as having five unknowns, it can be reduced. In fact, the system can truly be expressed as a nonlinear system in two equations using the following logic. \( P, Q, \text{ and } R \) are functions of the wind axes angle rates, \( \alpha, \beta, \dot{\alpha}, \text{ and } \dot{\beta} \). \( \dot{\alpha} \) and \( \dot{\beta} \) are functions of \( \alpha \) and \( \beta \) using the dependence property and the wind axes angle rates are known quantities. Therefore, the body axes angle rates can be written simply as functions of \( \alpha \) and \( \beta \). However, \( \alpha \) and \( \beta \) were expressed as functions of themselves (including the dependence property) and the body axes angle rates. This results in a circular argument and the following statement.

\[
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} = G \left( \begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} \right) \tag{EQ. 37}
\]

The fixed point method is based on an iterative procedure whereby

\[
x^{(k)} = G \left( x^{(k-1)} \right) \tag{EQ. 38}
\]

In this case, \( x \) is a vector of \( \alpha \) and \( \beta \). The initial values, \( \alpha(0) \) and \( \beta(0) \), are determined assuming the angles are the only contributors to lift and sideforce respectively.
The Contraction Mapping Theorem states that a fixed point exists if \( G \) is a continuous function and that \( G(x) \in D \) whenever \( x \in D \) where \( D \subset \mathbb{R}^n \).\(^{17}\) Moreover this series will converge to that fixed point if \( G \) has continuous partial derivatives and a constant \( K < 1 \) exists with

\[
\left| \frac{\partial g_i(x)}{\partial x_j} \right| \leq \frac{K}{n} \quad \text{whenever } x \in D, \ D \subset \mathbb{R}^n
\]  

(EQ. 39)

Equation 34 shows that the least squares derivative is linear in its dependence on the last data point. The other factors are produced from trigonometric functions which are smooth. The arithmetic result of the individual terms may not be smooth. Examining the equations reveals that the evaluation of the matrix multiplication contains trigonometric functions multiplied multiple times. This mapping can be represented geometrically. Figure 9 illustrates the nature of this mapping using typical state values and aerodynamic data. Different state values and aerodynamic data will produce different surfaces but their nature will be similar. Each surface represents one element of \( G \) and are smooth over the range of \( \alpha \) and \( \beta \) considered (\( \pm 60^\circ \) for mathematical interest only). Since each vector component is smooth, the mapping is continuous and has continuous derivatives (Burden\(^{17}\)). Therefore to guarantee convergence, it is necessary to specify the subspace \( D \) and the initial values for \( \alpha \) and \( \beta \) such that equation 39 is true.
Figure 9  Mapping Smoothness for Fixed Point

To mold the system into the form prescribed in equation 37, equations 14 and 16 can be modified into the following.

\[
\alpha_{n+1} = \frac{C_L - C_{L,0} - C_{L,\dot{\alpha}} \cdot \dot{\alpha}_n - C_{L,Q} \cdot Q_n}{C_{L,\alpha}} \quad \text{(EQ. 40)}
\]

\[
\beta_{n+1} = \frac{-C_D \cdot \sin \beta_n - C_C \cdot \cos \beta_n - C_{Y,P} \cdot P_n - C_{Y,R} \cdot R_n}{C_{Y,\beta}} \quad \text{(EQ. 41)}
\]

To get the procedure started, initial values must be guessed at. Assume \( \dot{\alpha} \), P, Q, and R contributions to the force equations are zero. In this formulation, these force equations were given by linearized representations using the static and dynamic stability derivatives to describe the dependence of the force to the states around a reference condition. Therefore, this assumption is equivalent to approximating the stability derivatives such as \( C_L \dot{\alpha} \), \( C_{L,Q} \), \( C_{Y,P} \), and \( C_{Y,R} \) to be zero. In general, these coefficients
are much smaller than the force dependencies on $\alpha$ or $\beta$ and therefore this justifies the initial guesses for $\alpha$ and $\beta$. $\alpha_0$ can be described by

$$\alpha_0 = \frac{C_L - C_{L,0}}{C_{L,\alpha}} \quad \text{(EQ. 42)}$$

To simplify the initial calculation for $\beta$, it is also assumed that the sideslip dependence on body axes side-force can be used as a stand-in for wind axes side-force (although in the opposite direction). Therefore, $\beta_0$ is given by

$$\beta_0 = \frac{C_C}{-C_{Y\beta}} \quad \text{(EQ. 43)}$$

The initial values for $P$, $Q$, and $R$ are calculated as given in equation 17 after $\alpha$ and $\beta$ are numerically differentiated. $P$, $Q$, and $R$ are once again calculated as cited by equation 18. All successive iterations are calculated as has already been described until the norm of the change in $\alpha$ and $\beta$ has sufficiently approached zero. Figure 10 illustrates the iterative procedure that takes place until convergence.
It can be noted that many secondary effects can be added without further complication. For instance, the additional dependencies on the angular accelerations and the time derivative of $\beta$ can be included in the force equations provided those characteristic derivatives are available. This would require an extra step to run the numerical differentiator after each iteration. The Mach dependence can be added with no extra step because Velocity is an input at each data sample. The speed of sound is a known quantity reflected in the altitude and therefore $\Delta$Mach is given. Although control surface contributions remain a significant limitation, most other dependencies may be added without loss of generality.

5.4 Euler Parameters Using Newton's Method

The system of equations for finding the Euler parameter representation of the aircraft's orientation is given in the discussion on page 20. The system was formulated by setting two transformation matrices equal to each other. In this case, a Newton method solver is well suited to this problem. The Newton method is an optimization technique that uses a Jacobian matrix in determining a search direction for minimizing a vector equation $F(x) = 0.17$. Because the functions are easily expressed in terms of the variables, the Jacobian matrix can be computed rather than approximated.
Since only four of the nine possible equations are needed, it is necessary to pick four that provide unique information. The upper right hand 2x2 sub matrix provides a system that satisfies this requirement. Putting these elements into the format discussed in the literature, these equations are

\[
\mathbf{F}(\mathbf{x}) = \begin{cases} 
2 \cdot (q_1 \cdot q_2 + q_0 \cdot q_3) - T_{BE_{(1,2)}} \\
2 \cdot (q_1 \cdot q_3 - q_0 \cdot q_2) - T_{BE_{(1,3)}} \\
q_0^2 - q_1^2 + q_2^2 - q_3^2 - T_{BE_{(2,2)}} \\
2 \cdot (q_2 \cdot q_3 + q_0 \cdot q_1) - T_{BE_{(2,3)}} 
\end{cases} = 0
\]  

(EQ. 44)

where \( \mathbf{x} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \)

The Jacobian is calculated by the partial derivative of each of these elements with respect to each variable or

\[
\mathbf{J}_{i,j} = \frac{\partial F_i}{\partial x_j} \quad \text{for } i \& j = 1 \text{ to } 4
\]  

(EQ. 45)

The system is solved in an iterative fashion by updating the value of \( \mathbf{x} \) according to the following formula.

\[
\Delta \mathbf{x} = \mathbf{J}^{-1} \cdot -\mathbf{F}(\mathbf{x})
\]  

(EQ. 46)
The solution is said to have converged when the norm of $\Delta x$ falls below some preset accuracy requirement, $\varepsilon$.

The initial values for this iterative procedure can be approximated by guessing at the Euler angles corresponding to the body to earth transformation, $\phi$, $\theta$, and $\psi$. The initial values of the Euler parameters are then calculated as functions of these angles. This initial guesses of the angles are as follows:

\[
\begin{align*}
\phi_0 &= \mu \\
\theta_0 &= \gamma + \alpha \\
\psi_0 &= \chi - \beta
\end{align*}
\]  

(EQ. 47)

The Euler angles can be put into in equations 21 through 24 to get initial guess for the Euler parameters.

There are dangers in these initial guesses that warrant investigation. The problem arises when an initial guess for an angle is exactly $\pi/2$. The routine begins in a hole and the initial search distance is so small that the routine may declare convergence before it has really converged. Perturbing these initial Euler angles ($\phi$ and $\theta$ specifically) guesses away from values of $\pi/2$ solves this problem.

For validation, a rigorous test was imposed on the solver. Test cases that included negative angles, angles of numerous rotations, and every perturbation of single or multiple $\pi/2$ angles challenged the solver. The test cases were translated into a transition matrix and modest initial guesses were applied. In every case, the solver demonstrated to be an extremely effective and accurate method for determining the Euler parameters.
5.5 Qualification of the Wind Axes Roll Angle, $\mu$

It has been noted that if $\mu$ is not one of the specified inputs to the trajectory expansion algorithm, it may be calculated by taking advantage of a coordinated flight assumption that the side-force is zero. Equation 10 shows the relationship that can be developed by making this assumption. Two caveats were introduced. The first was that of keeping the angle continuous across boundaries and the other was the problem in specifying "up". These issue are briefly addressed here.

The first consideration is a practical one. In order to expand the function result of an inverse tangent function to the complete range of one revolution, a quadrant specific tangent function must be used. In addition, this removes any singularity that may occur if the denominator is zero. In FORTRAN, either the ATAN2 or DATAN2 function is adequate. This requires entering two arguments. These arguments are the x and y coordinates of a point on a unit circle. When individually specified, these coordinates can determine quadrant by the positive or negative sign of each term. The result of the function is a radian measure between -$\pi$ and $\pi$.

With experience, it is found that the equation has a characteristic that judges the concept of "up." Initially it was found that a positive value in the $F_{ZE}$ term (all other forces neutral) results in a $\mu$ equal to 180°. This is due to the numerator of the equation. Since an aircraft flies right side up in straight and level flight, both the numerator (y argument) and denominator (x argument) were multiplied by -1 hypothesized to have canceled out in the derivation. This resolved most normal flight considerations, but the equation still exhibits a "preference" for one maneuver type over the other. As suggested before, and due to the reasons above, this equation returns a $\mu$, that is consistent with an aircraft pulling g's rather than pushing. The readily available solution to this uncertainty, is to simply specify the roll angle as an input. A more general
solution is to just include an indicator of positive or negative g's. The equation could then be modified accordingly. If neither is desirable, then the available solution is still valid, although it may differ from experience by the 180° shift during negative g maneuvers.

To make angle determination more complicated, a problem arises in a roll maneuver where more than 360° is traversed or the limits are surpassed. Obviously, if the bounds of the arctangent function are violated, a jump occurs in the data stream. A jump of this magnitude and direction leads to erroneous results. For example, if the craft goes from 179° to 181° over the time interval, the unaided arc tangent function would issue the results of 179° and -179°. The angle itself is not as great of a concern as the time derivative, which may yield an erroneous -358° per time step. This certainly cannot be tolerated. Therefore, it is desirable to keep track of the position of \( \mu \) and correct for the discontinuity.

An algorithm was designed to keep track of the last value of \( \mu \) and the number of physical rolls that have been performed. This value is referred to as the cycle. It is equal to the number of completed rolls plus 1. The next value of \( \mu \) that is calculated from the arctangent function is called a guess value. The guess value is adjusted to correspond with the counted number of rolls. If there is a large discrepancy between the last value of \( \mu \) and the adjusted guess value (i.e. 6 radians) then it is determined that the aircraft has either added or subtracted 1 from the cycle number. In other words it has completed another roll maneuver. The cycle is updated and \( \mu \) is corrected with the new value of cycle.
6. DISCUSSION OF COMPUTER CODE - TEA.FOR

The trajectory expansion algorithm has been implemented into a FORTRAN code titled TEA.FOR (Trajectory Expansion Algorithm). All the features previously discussed were integrated into the code which is listed in the Appendix. The required knowledge for the current version as defined by both hard and soft assumptions is presented for interested operators.

6.1 Input

The inputs are the time histories of $V_C$, $\chi$, $\gamma$, and if desired $\mu$. These variables should be set up in an ASCII style file with the VM1 designation INPUT FILE A1. These variables are to be in column format separated by at least 2 spaces and such that the columns do not exceed 80 characters. The format of the numbers may be either fixed decimal, scientific notation, or a mixture. There are to be no column headers in the file. The columns are to be in the following order: Time of Sample, $V_C$, $\chi$, $\gamma$, $\mu$ (if required). The Time of Sample is for book-keeping purposes only and should correspond to the fixed sampling rate.

Initial conditions are assumed to be steady flight. Therefore, the initial conditions determine the time history for the $n=27$ samples before initiation of the maneuver. The first row of the data file is inputted and held constant over this range. $V_C$ and $\chi$ may be entered as desired but $\gamma$ and $\mu$ are required to be zero. Within the code, $V$, $P$, $Q$, and $R$ are set to zero and $U$ and $W$ are found by finding conditions such that lift equals weight. The program will stop automatically at the end of the file so no special codes are required there.
6.2 Required Code Specifications

Table 3 is a list of the necessary quantities which are to be specified before the code is compiled and run. All can be found in the first part of the program's main body and modified according to airframe. Earlier it was noted that the dynamic derivatives, may be non-dimensionalized. The ones included in the program are of this form. For example:

\[ C_{L\alpha} = \frac{\partial C_L}{\partial \left( \frac{\alpha \bar{c}}{2V} \right)} \]

\[ C_{YP} = \frac{\partial C_Y}{\partial \left( \frac{Pb}{2V} \right)} \]

\[ C_{LQ} = \frac{\partial C_L}{\partial \left( \frac{Q\bar{c}}{2V} \right)} \]

\[ C_{YR} = \frac{\partial C_Y}{\partial \left( \frac{Rb}{2V} \right)} \]

(EQ. 48)
Table 3 Airframe Program Variables in Order of Appearance

<table>
<thead>
<tr>
<th>FORTRAN VARIABLE</th>
<th>QUANTITY</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP</td>
<td>1 / sampling rate</td>
<td>seconds</td>
</tr>
<tr>
<td>MASS</td>
<td>aircraft mass</td>
<td>slugs</td>
</tr>
<tr>
<td>S</td>
<td>Planform Area</td>
<td>square feet</td>
</tr>
<tr>
<td>E</td>
<td>Span Efficiency</td>
<td>dimensionless</td>
</tr>
<tr>
<td>B</td>
<td>wing span</td>
<td>feet</td>
</tr>
<tr>
<td>CBAR</td>
<td>mean chord</td>
<td>feet</td>
</tr>
<tr>
<td>ALTO</td>
<td>initial altitude</td>
<td>feet</td>
</tr>
<tr>
<td>CL0</td>
<td>$C_{L0}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CLA</td>
<td>$C_{L\alpha}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CLQ</td>
<td>$C_{LQ}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CLAD</td>
<td>$C_{L \alpha}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CLM</td>
<td>$C_{LM}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CYB</td>
<td>$C_{YB}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CYR</td>
<td>$C_{YR}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CYP</td>
<td>$C_{YP}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CDO</td>
<td>$C_{DO}$</td>
<td>dimensionless</td>
</tr>
<tr>
<td>CDM</td>
<td>$C_{DM}$</td>
<td>dimensionless</td>
</tr>
</tbody>
</table>

6.3 Output

Output is delivered into 80 column formatted output files with FORTRAN unit labels 11 through 19. These can be directed with either the OPEN statement or the VM1 CMS FILEDEF command. The files contain all input and output streams carrying four significant figures. Each file column has a header that identifies the column. Table 4 is a partial list of those quantities.

There are two separate output styles. The first is a long form that has all major quantities including input. The second is the short form and only contains data that is necessary for the model following algorithm. The program will prompt the user for the type of output desired.
<table>
<thead>
<tr>
<th>FORTRAN Variable</th>
<th>Column Header</th>
<th>Quantity</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>time</td>
<td>seconds</td>
</tr>
<tr>
<td>VC</td>
<td>VC</td>
<td>V_c</td>
<td>feet/second</td>
</tr>
<tr>
<td>HWANG(1)</td>
<td>CHI</td>
<td>χ</td>
<td>radians</td>
</tr>
<tr>
<td>HWANG(2)</td>
<td>GAMMA</td>
<td>γ</td>
<td>radians</td>
</tr>
<tr>
<td>HWANG(3)</td>
<td>MU</td>
<td>μ</td>
<td>radians</td>
</tr>
<tr>
<td>CHID</td>
<td>CHIDOT</td>
<td>Ẋ</td>
<td>radian/second</td>
</tr>
<tr>
<td>Gammad</td>
<td>Gammadot</td>
<td>Ẏ</td>
<td>radian/second</td>
</tr>
<tr>
<td>MUD</td>
<td>MUDOT</td>
<td>Ẋ E</td>
<td>feet/second</td>
</tr>
<tr>
<td>EXVEL</td>
<td>XEDOT</td>
<td>Ẋ E</td>
<td>feet/second</td>
</tr>
<tr>
<td>EYVEL</td>
<td>YEDOT</td>
<td>Ẏ E</td>
<td>feet/second</td>
</tr>
<tr>
<td>EZVEL</td>
<td>ZEDOT</td>
<td>Ẋ E</td>
<td>feet/second</td>
</tr>
<tr>
<td>EXACC</td>
<td>XE2DOT</td>
<td>Ẋ E</td>
<td>feet/second²</td>
</tr>
<tr>
<td>EYACC</td>
<td>YE2DOT</td>
<td>Ẏ E</td>
<td>feet/second²</td>
</tr>
<tr>
<td>EZACC</td>
<td>ZE2DOT</td>
<td>Ẋ E</td>
<td>feet/second²</td>
</tr>
<tr>
<td>WBANG(1)</td>
<td>ALPHA</td>
<td>α</td>
<td>radian</td>
</tr>
<tr>
<td>WBANG(2)</td>
<td>BETA</td>
<td>β</td>
<td>radian</td>
</tr>
<tr>
<td>ALPHAD</td>
<td>ALPHADOT</td>
<td>α</td>
<td>radian/second</td>
</tr>
<tr>
<td>BETAD</td>
<td>BETADOT</td>
<td>β</td>
<td>radian/second</td>
</tr>
<tr>
<td>PW</td>
<td>PW</td>
<td>P_W</td>
<td>radian/second</td>
</tr>
<tr>
<td>QW</td>
<td>QW</td>
<td>Q_W</td>
<td>radian/second</td>
</tr>
<tr>
<td>RW</td>
<td>RW</td>
<td>R_W</td>
<td>radian/second</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>radian/second</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
<td>Q</td>
<td>radian/second</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>radian/second</td>
</tr>
<tr>
<td>PD</td>
<td>PDOT</td>
<td>P</td>
<td>radian/second²</td>
</tr>
<tr>
<td>QD</td>
<td>QDOT</td>
<td>Q</td>
<td>radian/second²</td>
</tr>
<tr>
<td>RD</td>
<td>RDOT</td>
<td>Ẋ R</td>
<td>radian/second²</td>
</tr>
<tr>
<td>BXVEL</td>
<td>U</td>
<td>U</td>
<td>feet/second</td>
</tr>
<tr>
<td>BYVEL</td>
<td>V</td>
<td>V</td>
<td>feet/second</td>
</tr>
<tr>
<td>BZVEL</td>
<td>W</td>
<td>W</td>
<td>feet/second</td>
</tr>
<tr>
<td>BXACC</td>
<td>UDOT</td>
<td>Ẋ U</td>
<td>feet/second²</td>
</tr>
<tr>
<td>BYACC</td>
<td>VDOT</td>
<td>Ẋ V</td>
<td>feet/second²</td>
</tr>
<tr>
<td>BZACC</td>
<td>WDOT</td>
<td>Ẋ W</td>
<td>feet/second²</td>
</tr>
<tr>
<td>Q(0)</td>
<td>Q0</td>
<td>q_0</td>
<td>dimensionless</td>
</tr>
<tr>
<td>Q(1)</td>
<td>Q1</td>
<td>q_1</td>
<td>dimensionless</td>
</tr>
<tr>
<td>Q(2)</td>
<td>Q2</td>
<td>q_2</td>
<td>dimensionless</td>
</tr>
<tr>
<td>Q(3)</td>
<td>Q3</td>
<td>q_3</td>
<td>dimensionless</td>
</tr>
</tbody>
</table>
7. SAMPLE TRAJECTORIES

7.1 Demonstration Setup

In order to demonstrate the ability of the trajectory expansion algorithm, it is necessary to take trajectory data that includes complete time histories of all state and state derivative elements and use it to compare with TEA output. The airframe must be well documented and the maneuvers must test individual aspects of aircraft dynamics. However, it is just as important to show the weaknesses of the algorithm as showing the strengths. A minimally dynamic aircraft performing uninteresting maneuvers would hide these weaknesses and be counter productive.

In March of 1992, NASA Langley Research Center provided the use of the Langley Differential Maneuvering Simulator (DMS). The DMS is a twin dome fixed-base simulator which incorporates computer generated imaging, CRT and heads-up display in the cockpit, and a full envelope math model for the F/A-18 Hornet into a comprehensive research tool. The F/A-18 Hornet is a high performance, highly maneuverable fighter/attack aircraft. Because the aircraft is integral to current research and available on the DMS, it was an ideal choice for the test. Figure 11 shows schematics of the DMS and a sketch of the F/A-18.

The pilot was assigned to fly three low angle of attack maneuvers to test the TEA.FOR FORTRAN code against. One advantage to the computer simulation is that measurement noise is not a factor and the resulting data is very clean.

The data was sampled every 0.032 seconds or approximately 31 Hz. Recall that the least squares A matrix was designed to 50 Hz. Using the same number of entries of 27 stretches the design history span of 0.5 to 0.864 second, but was not changed in order to highlight the algorithm's limitations.
All maneuvers were initiated at a straight and level flight condition of Mach 0.5 at 25,000 ft. The three maneuvers were 1) a straight and level acceleration from Mach 0.5 to Mach 0.7, 2) a pull-up to a flight path angle of 45 degrees, and 3) a 2 g steady level turn. The DMS stored the complete complement of the states, state rates, control deflections as well as most any other pertinent information. The trajectory output of these maneuvers was then input into the TEA code in order to test the validity of the algorithm. The output from the TEA code was then matched with the DMS results.

TEA.FOR had to be calibrated with the F/A-18 stability derivatives as described in the previous section. This data was acquired from a combination of simulator data and F/A-18 Stability and Control Data Report. The data taken from the report was picked off of charts corresponding to Mach = 0.6 at 20,000 feet for angle of attack = 6 degrees. Table 5 lists these quantities.
Differential Maneuvering Simulator (DMS)

3-View Sketch of F/A-18

Figure 11 Experimental Equipment - DMS & F/A-18 Model
### Table 5  F/A-18 Low Angle of Attack Data:

<table>
<thead>
<tr>
<th>FORTRAN VARIABLE</th>
<th>VALUE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP</td>
<td>0.032</td>
<td></td>
</tr>
<tr>
<td>MASS</td>
<td>1006</td>
<td>=32366 lbf/g</td>
</tr>
<tr>
<td>S</td>
<td>400.0</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>0.1206266</td>
<td>K defined instead of E</td>
</tr>
<tr>
<td>B</td>
<td>37.42</td>
<td></td>
</tr>
<tr>
<td>CBAR</td>
<td>11.52</td>
<td></td>
</tr>
<tr>
<td>ALTO</td>
<td>25000</td>
<td></td>
</tr>
<tr>
<td>CL0</td>
<td>0.017982</td>
<td></td>
</tr>
<tr>
<td>CLA</td>
<td>5.327194</td>
<td></td>
</tr>
<tr>
<td>CLQ</td>
<td>3.0</td>
<td>non-dimensionalized by cbar/2V</td>
</tr>
<tr>
<td>CLAD</td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>CLM</td>
<td>0.0</td>
<td>Not used</td>
</tr>
<tr>
<td>CYB</td>
<td>-1.044</td>
<td></td>
</tr>
<tr>
<td>CYR</td>
<td>0.232132</td>
<td>non-dimensionalized by b/2V</td>
</tr>
<tr>
<td>CYP</td>
<td>1.435144</td>
<td>&quot;</td>
</tr>
<tr>
<td>CDO</td>
<td>0.0135814</td>
<td></td>
</tr>
<tr>
<td>CDM</td>
<td>0.0</td>
<td>Not used</td>
</tr>
</tbody>
</table>

7.2 TEA Results

There are many aspects of the results that are worth discussion. First of all, it should be stated in advance that the results are less than a perfect match. However, they do correspond very favorably. Without making excuses, it should be noted that the F/A-18's control surfaces make up a considerable amount of the wing area. The fully moving horizontal tail has an area of 88.1 square feet. This is more than 20 percent of the wing planform. Therefore, the assumption that control surfaces are strictly moment generators may be tasking for this aircraft. Secondly, the "linear" region of the $C_L$ versus $\alpha$ curve extends to about 0 and 14 degrees angle of attack, but the slope is $\alpha$ dependent after that. Thirdly, the F/A-18 has two sets of aerodynamic data; one set is for flaps retracted and
very complex for this aircraft and why it is a true challenge for the TEA code. Finally, as mentioned, the data was sampled at 31 Hz which may hamper the differentiator's ability to model correctly oscillations at the higher end of the spectrum. The version used was designed for a 50 Hz sample rate and therefore showcases the lag caused by the linear dependence of the averaged polynomial.

7.3 Trajectory #1: Straight and Level Acceleration

The maneuver begins at 25,000 ft and Mach 0.5 and is terminated at Mach 0.7. Altitude was held constant. The results are presented in figures 12.1 - 12.13. Most pages contain two graphs. In general, these are the state and the corresponding state time derivative.

The results show a very good match between DMS and TEA. The plot of flight path angle time derivative shows the differentiator's success. Although difficult to see, a certain amount of lag was introduced due to the averaged polynomial as expected. The plot of angle of attack shows an excellent agreement in character although a certain displacement exists. The displacement may be caused by many things. The most obvious is the small time lag. Secondly, picking points off of wind tunnel charts is not very accurate. These quantities are valid to only two significant figures at the most. Also the plots used were nearest flight condition rather than identical conditions. Thirdly, atmospheric density data is interpolated in TEA by crude methods and may differ from the simulation approximation. This is evident in the dynamic pressure. The DMS showed dynamic pressure equal to 137.5 lbf/ft² compared to the TEA of about 137.0. These small differences certainly can explain the small shift in angle of attack.

Also at about 2.5 seconds, a small anomalous hump appears in the TEA results. The duration of the hump corresponds directly to the input of thrust. Therefore, this
strongly suggests a lift dependency on either x-axis acceleration or it is a thrust effect. Neither effect was included in the formulation. Either way, if the thrust contributes to the lift directly, then TEA produces increases angle of attack to explain it.

These same error proliferate in the angle of attack time derivative, the pitch rate and acceleration, and the body z-axis component of velocity and its time derivative. Each of these disputes can be attributed directly to the original differences in angle of attack. This suggests some fine tuning of the constants but, overall, the method is still valid.
Figure 12.1 Velocity at Center of Gravity
Figure 12.2 Heading Angle
Flight Path Angle

Figure 12.3 Flight Path Angle
Figure 12.4 Wind Axes Roll Angle
Figure 12.5 Angle of Attack
Figure 12.6 Sideslip Angle
Figure 12.7 Body Axes Roll Rate
Figure 12.8  Body Axes Pitch Rate
Figure 12.9  Body Axes Yaw Rate
Figure 12.10 Body Axes X Velocity Component
Figure 12.11  Body Axes Y Velocity Component
Figure 12.12 Body Axes Z Velocity Component
Figure 12.13 Euler Parameters
7.4 Trajectory #2: Pull Up to 45 Degrees Flight Path Angle

This maneuver begins at Mach 0.5 and 25,000 ft. The pilot then inputs aft stick to initiate a climb until $\gamma = 45$ degrees. The results are presented in figures 13.1-13.14. The layout is the same as in trajectory #1.

Each observation from trajectory #1 is evident in trajectory #2. There is some time lag, the aerodynamic model is limited in accuracy, and the atmospheric data is slightly different. Instead of a discrepancy caused by throttle input, there is a small difference in angle of attack at the time that the elevator is deflected. Again this is a symptom of the limitation of the control surface assumption. This effect is demonstrated throughout the longitudinal states and derivatives. However, there is also a new discrepancy. The angle of attack plot shows that the DMS result to be much higher than the TEA result for a large portion of the maneuver. This can be explained in the observation that above 14 degrees AOA, the F/A-18 $C_L$ versus $\alpha$ curve can no longer be approximated as linear for large regions. It can be seen that the two versions of the $\alpha$ trace depart at approximately 14 degrees. Examining the control data report reveals that the slope begins to decrease at approximately this same point. An experiment was done to test the possibility that this is the primary cause of the discrepancy. A two stage linear $C_L$ versus $\alpha$ curve was integrated into TEA. The first stage was valid for angles of attack below 14 degrees or $C_L = 1.2$ and the second stage was used for above 1.2. The second stage was roughly an average slope for a range of 10 additional degrees. Figure 13.14 shows the angle of attack trace that occurs when this approach is implemented. This figure suggests that indeed, it is the nonlinearity that is causing the discrepancy. There is good agreement below 14 degrees where the data is linear and at 18 degrees where the slope of the second level was designed for. If there had been a large number
of intermediate levels to smooth out the transition, the output would better correlate with the simulation results.

This suggests an improvement to the algorithm. An interpolation routine can be used in place of $C_L\alpha$. After the dynamic derivative effects are taken out of the overall lift coefficient the lift curve slope is interpolated to find the corresponding value of angle of attack. This function could be used up to $C_{L\text{max}}$ because beyond that point there is an ambiguity in answer.

Other plots show good agreement to further support the ability of the algorithm.
Figure 13.1 Velocity at Center of Gravity
Figure 13.2 Heading Angle
Figure 13.3 Flight Path Angle
Figure 13.4 Wind Axes Roll Angle
Figure 13.5 Angle of Attack
Figure 13.6 Sideslip Angle
Figure 13.7 Body Axes Roll Rate
Figure 13.8 Body Axes Pitch Rate
Figure 13.9 Body Axes Yaw Rate
Figure 13.10 Body Axes X Velocity Component
Figure 13.11 Body Axes Y Velocity Component
Figure 13.12 Body Axes Z Velocity Component
Figure 13.13 Euler Parameters
Figure 13.14 Changes Resulting from Modifying TEA CL vs. $\alpha$ Curve
7.5 Trajectory #3: 2 g steady level turn

This is the first maneuver that incorporates lateral and directional quantities. The agreement between TEA and DMS has degraded somewhat, although it still shows promise. In general the directional quantities of sideslip and yaw rate show large discrepancies, while the longitudinal quantities of angle of attack and pitch rate show minor discrepancies. Once again, these discrepancies can be traced to the control surface contributions to the forces on the aircraft. The results are presented in figures 14.1 - 14.15.

One uncompromised success is depicted in figure 14.14. The previous trajectories were strictly longitudinal. This meant that there was no reason for testing the calculation of roll angle, $\phi$, assuming it is not available. Although the other plots in the series were created with $\phi$ as an input, the program was rerun specifically to test this calculation. The plot shows excellent correlation between the two curves.

The angle of attack trace is very good in character, but there is a marked vertical shift that resembles the problem discussed in trajectory #2. The difference is that $\alpha$ is shifted in the opposite direction. The discrepancy can be explained by examining the DMS time history. The time history of the leading and trailing edge flaps correspond directly to the differences in the traces. Since flaps are used to directly produce lift, less angle of attack is needed by the DMS than TEA. The amount of lift reduction will appear as a direct displacement in $\alpha$. Figure 14.15 shows the control trace. This control difference and the less dramatic initial stabilator hump manifest in all other longitudinal plots.

The sideslip trace is a significantly more discouraging event. In this case, a discrepancy exists that is not bad in absolute terms, but relatively changes the character. This discrepancy can also be attributed to the control surfaces. Figure 14.15 show the
rudder activity during the maneuver. It seems that the surfaces are contributing greatly to the side-force component because the activity of the rudders coincides with the discrepant sideslip. If the rudders contribute directly to sideforce than TEA will generate sideslip to mimic the motions. The widely varying sideslip angle directly influences the yaw rate and yaw rate time derivative in the form of sideslip time derivative. This is more discouraging than the angle of attack discrepancy because of the assumptions that went in to the algorithm. Flaps are direct force generators that were not included in the assumptions of this method. In contrast, the rudder contribution is a key assumption which appears to be inaccurate at least for this aircraft. The overall error is small in absolute terms, however, and may prove to be acceptable.
Figure 14.1 Velocity at Center of Gravity
Figure 14.2 Heading Angle
Figure 14.3 Flight Path Angle
Figure 14.4 Wind Axes Roll Angle
Figure 14.5 Angle of Attack
Figure 14.6 Sideslip Angle
Figure 14.7 Body Axes Roll Rate
Figure 14.8  Body Axes Pitch Rate
Figure 14.9 Body Axes Yaw Rate
Figure 14.10 Body Axes X Velocity Component
Figure 14.11 Body Axes Y Velocity Component
Figure 14.12 Body Axes Z Velocity Component
Figure 14.13 Euler Parameters
Figure 14.14 TEA Approximation of Roll Angle
Figure 14.15 Control Surfaces
8. CONCLUSION

The trajectory expansion algorithm has shown promise as a valid tool in dynamics inversion. It has provided interesting insight into numerical differentiation, control surface contribution to linear forces, the importance of an accurate aerodynamic model, and the ability to calculate trajectory states for even high performance aircraft.

It was found that these calculations could be made in real time for smooth data. This process was aided by implementing a least squares differentiation technique. By using a weighted average concept, piecewise smooth data could be handled in a manner that was intuitive and provided minimal disruption to the state calculation. Minor noise in the data would not be excessively aggravated by this least squares method, although the results still contain an amount of noise. Severely noisy data could be moderately attenuated with least squares methods, but the return on the investment was not substantial enough to warrant classification as successful. Future work should include filtering techniques.

The major limitation was found to be the integral assumption that ignores direct force contributions from the control surfaces. An aircraft with numerous large control surfaces such as the F/A-18 is going to have some significant force contributions from these surfaces.

Additionally, an available aerodynamic database could be easily integrated and would greatly improve the accuracy of the results. This would allow trajectories that extend to $C_{L\text{max}}$. Only limitations on the direction of thrust need to be overcome. These two additions would allow the trajectory expansion algorithm to extend to high $\alpha$ research, arguably the most important research arena of the next generation of fighter.
REFERENCES


6 Durham, W., Lutze, F., Barlas, R., Munro, B., "Nonlinear Model-Following Control Application to Airplane Control," AIAA-91-2635-CP


APPENDIX TEA.FOR PROGRAM LISTING

C----............................................................................C
C----............................................................................C
C
C
C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C
C QW(27) = QW, WIND AXES PITCH RATE
C RW(27) = RW, WIND AXES YAW RATE
C F(27) = F, BODY AXES RCLL RATE
C Q(27) = Q, BODY AXES PITCH RATE
C R(27) = R, BODY AXES YAW RATE
C
C FW = TIME DERIVATIVE OF PW
C QW = TIME DERIVATIVE OF QW
C RW = TIME DERIVATIVE OF RW
C F = TIME DERIVATIVE OF F
C Q = TIME DERIVATIVE OF Q
C R = TIME DERIVATIVE OF R
C
C EVEL(27) = EARTH FIXED VELOCITY IN X DIRECTION
C EYVEL(27) = EARTH FIXED VELOCITY IN Y DIRECTION
C EZVEL(27) = EARTH FIXED VELOCITY IN Z DIRECTION
C
C EXACC = EARTH FIXED ACCELERATION IN X DIRECTION
C EYACC = EARTH FIXED ACCELERATION IN Y DIRECTION
C EZACC = EARTH FIXED ACCELERATION IN Z DIRECTION
C
C BWVEL(27) = BODY FIXED VELOCITY IN X DIRECTION
C BYVEL(27) = BODY FIXED VELOCITY IN Y DIRECTION
C BZVEL(27) = BODY FIXED VELOCITY IN Z DIRECTION
C
C BXACC = BODY FIXED ACCELERATION IN X DIRECTION
C BYACC = BODY FIXED ACCELERATION IN Y DIRECTION
C BZACC = BODY FIXED ACCELERATION IN Z DIRECTION
C
C FORCE(1) = EARTH FIXED FORCE IN X
C FORCE(2) = EARTH FIXED FORCE IN Y
C FORCE(3) = EARTH FIXED FORCE IN Z
C
C OTHER VARIABLES:
C CYCLEN = WHICH 2PI INCREMENT NU IS IN
C G = ACCELERATION OF GRAVITY, 32.174 PT/S^2
C PI = 3.1415927 ETC. DOUBLE PRECISION DACOS(-1)
C I,J,L = FOR NEXT COUNTERS
C X = DRAG DUE TO LIFT COEFFICIENT
C QI(4) = EULER PARAMETERS
C
C ASSUMPTIONS:
C 1) IF NU IS NOT GIVEN, SIDEFORCE IS ASSUMED 0 SO NU MAY BE
C CALCULATED.
C 2) LIFT AND SIDEFORCE ARE PRIMARILY FUNCTIONS OF ALPHA AND
C BETA RESPECTIVELY, THIS ALLOWS THESE TO PARAMETERS TO BE
C CALCULATED(ALPHA AND BETA).
C 3) CODE - DRAG DUE TO ELEVATOR DEFLECTION NEGLECTED
C
C $declare
PROGRAM MAIN
DOUBLE PRECISION PI
DOUBLE PRECISION E,AR
DOUBLE PRECISION VC,SWANG(3),WANG(3), QI(0:3)
DOUBLE PRECISION ALPHA(27),BETA(27),CHI(27),GAMMA(27),MU(27)
DOUBLE PRECISION ALPHAD,BETAD,CHID,GAMMAD,MUD
DOUBLE PRECISION XVEL(27),XVEL(27),XVEL(27)
DOUBLE PRECISION EXACC, EYACC, EZACC
DOUBLE PRECISION A(13,13), XDUM(27)
DOUBLE PRECISION BXVEL(27),BYVEL(27),BZVEL(27)
DOUBLE PRECISION BXACC, BYACC, BZACC
DOUBLE PRECISION FORCE(3),aerof(3),qs
DOUBLE PRECISION FW(27),QM(27),RW(27),P(27),Q(27),R(27)
DOUBLE PRECISION PWD, FWD, RWD, PD, QD, RD
DOUBLE PRECISION MACH, ALTO, ALT, DM
DOUBLE PRECISION J1HANK(5), J1DIF(5)
DOUBLE PRECISION XPOS, YPOS
INTEGER  I,ITNUM,CUTNUM, COUNT,OUTYPE

LOGICAL ITRATE

C-----------------------------------------------------------------------------------
C
* LIFT STABILITY & CONTROL DERIVATES
  DOUBLE PRECISION CLO, CLA, CLQ, CLAD, CLM
  COMMON /LIFT/CLO, CLA, CLQ, CLAD, CLM

* Y FORCE STABILITY & CONTROL DERIVATIVES
  DOUBLE PRECISION CYB, CYP, CYR
  COMMON /Y/CYB, CYP, CYR

* DRAG TERMS
  DOUBLE PRECISION CDO, CDM, K
  COMMON /DRAG/CDO, CDM, K

C-----------------------------------------------------------------------------------
C
* PLANE CHARACTERISTICS
  DOUBLE PRECISION MASS, S, B, CBAR
  COMMON /PLANE/MASS, S, B, CBAR

* ATMOSPHERE CONDITION
  DOUBLE PRECISION G, RHO, ASOUND, MACHO
  COMMON /PSYS/G, RHO, ASOUND, MACHO

* MU CHARACTERISTICS
  LOGICAL MUINC
  INTEGER CYCLEN
  COMMON /LOG/MUINC, CYCLEN

* TIME QUANTITIES
  DOUBLE PRECISION STEP, TIME

121
COMMON /CLOCK/STEP,TIME
C-------------------------------------------------------------C
WRITE(*,*) ' FLY.FOR 1.A ITERATIVE METHOD'
WRITE(*,*) '-------------------------------'
13 WRITE(*,*) ' 1) LONG OR 2) SHORT OUTPUT:
READ(*,*) OUTYPE
IF (OUTYPE.NE.1.AND.OUTYPE.NE.2) GOTO 13
C ***** Output File ****************************
C ***** Output file  *****************************
OPEN(100,FILE=' ')
OPEN(101,FILE='ERROR.DAT')
OPEN( 1,FILE='F.001')
OPEN( 2,FILE='F.002')
OPEN( 3,FILE='F.003')
OPEN( 4,FILE='F.004')
IF (OUTYPE.EQ.1) THEN
OPEN( 5,FILE='F.005')
OPEN( 6,FILE='F.006')
OPEN( 7,FILE='F.007')
OPEN( 8,FILE='F.008')
OPEN( 9,FILE='F.009')
END IF
C ***** CONSTANTS ****************************
PI=3.141592653589793D0
G=32.174D0
STEP=0.03220
C****** PLANE CHARACTERISTICS ***************
MASS= 32366.0D0/G
S=400.0D0
E=1.0D0
B=37.42D0
CBAR=11.52
AR=B*B/S
X=1/(PI*AR*B)
X=0.1206265855d0
ALT=5000.0D0
ALTALT=ALT
XPOS=0.0D0
ZPOS=0.0D0
C***** LIFT DERIVATIVES **********************
CLO = -0.027166982d0
C 0.017982
CIA = 5.627194D0
< 5.327194
CLQ = 3.0D0
CLAD = 2.3D0
CLM = 0.0D0

C******* Y FORCE DERIVATIVES **************
CYB = -1.044D0
CYR = 0.232132D0
CPF = 3.435144685D0

C******* DRAG DERIVATIVES ***************
CDO = 0.0135914D0
CDM = 0.0D0

C*****************************************************************
WRITE(*, '(A)') ' MI INCLUDED (Y/F) : '
READ(*, *) MUIINC

WRITE(*, '(A)') ' ENTER SKIP COUNTER : '
READ(*, *) OUTNUM

WRITE(*, '(A)') ' ITERATE (Y/F) ? : '
READ(*, *) ITRATE

C************************************************************************
CALL ATMOS(ALT, RHO, ASOUND)

CALL INIT(OUTYPE, VC, ALPHA, BETA, CHI, GAMMA, KU, PW, QW, RW, P, Q, R,
+ EXVEL, EVEL, EVELX, EVELY, EVELZ,
+ A)
DO 14 I=1,5
14 ITBANK(I)=0.0D0

MACHO=VC/ASOUND

CHANG(1)=CHI(1)
CHANG(2)=GAMMA(1)
CHANG(3)=MU(1)
OCOUNT=OUTNUM-1

15 CONTINUE
OCOUNT=OCOUNT+1
WRITE(*, 1001) ' TIME = ', TIME, ' ALT = ', ALT, ' ITNUM = ', ITNUM
ALT=ALT-EVEL(27)*STEP
XPOS=XPOS-EXVEL(27)*STEP
YPOS=YPOS+EVEL(27)*STEP
CALL ATMOS(ALT, RHO, ASOUND)
MACH=VC/ASOUND
DM=MACH-MACHO

123
CALL FORCES(VC, HWANG, A, EXVEL, EVVEL, 
+ FORCE, EXACC, EYACC, EZACC)

CALL AERO(VC, FORCE, HWANG, HWANG, MU(27), aero, aq)

CALL WBDOT(WBANG, A, ALPHA, BETA, ALPHAD, BETAD)

CALL HBDOT(HWANG, A, CHI, GAMMA, MD, CHID, GAMMD, MUD)

CALL WRADE(HWANG, CHID, GAMMD, MUD, A, PW, QW, RW, PHD, QMD, RMD)

CALL BRADE(WBANG, ALPHAD, BETAD, A, 
+ PW(27), QW(27), RW(27), P, Q, R, PD, QD, RD)

IF (ITRATE) THEN

ITNUM = 0
ITRATE = .FALSE.
DO WHILE (.NOT. ITRATE)
ITNUM = ITNUM + 1
CALL ITERAT(A, aero, VC, qs, DM, PW(27), QW(27), RW(27), 
+ ALPHA, ALPHAD, BETA, BETAD, P(27), Q(27), R(27))
ITDIF(1) = ALPHA(27) - ITBANK(1)
ITDIF(2) = BETA(27) - ITBANK(2)
ITDIF(3) = P(27) - ITBANK(3)
ITDIF(4) = Q(27) - ITBANK(4)
ITDIF(5) = R(27) - ITBANK(5)
ITBANK(1) = ALPHA(27)
ITBANK(2) = BETA(27)
ITBANK(3) = P(27)
ITBANK(4) = Q(27)
ITBANK(5) = R(27)
CALL NC(5, ITDIF, ITRATE)
END DO

CALL SLOPE(27, STEP, A, P, XDUM, P(27), PB)
CALL SLOPE(27, STEP, A, Q, XDUM, Q(27), QD)
CALL SLOPE(27, STEP, A, R, XDUM, R(27), RD)
END IF

CALL BODY(VC, MEANG, A, BXVEL, BYVEL, BZVEL, BXACC, BYACC, BZACC)
CALL EULPAR(CHI(27), GAMMA(27), MU(27), ALPHA(27), BETA(27), QI)

IF (OCCOUNT.EQ.OUTNUM) THEN
OCCOUNT = 0
C------------------- SHORT OUTPUT ----------------------------------C
IF (OUTYP, EQ, 2) THEN
WRITE( 1, 1000) TIME, (QI(I), I=0, 3)
WRITE( 2, 1000) TIME, BXVEL(27), BXACC, BYVEL(27), BZVEL(27)

124
WRITE(3,1000)TIME,P(27),PD,Q(27),QD,R(27),RD
WRITE(4,1000)TIME,(HWANG(I),I=1,3),ALPHA(27),BETA(27)
END IF

C-----------------LONG OUTPUT---------------------------------------------------C
IF (OUTYPE.EQ.1) THEN
WRITE(1,1000)TIME,VC,(HWANG(I),I=1,3)
WRITE(2,1000)TIME,(HWANG(I),I=1,3),CHID,GAMMA,HUG
WRITE(3,1000)TIME,EXVEL(27),EYVEL(27),EZVEL(27),
+EXACC,SYACC,RZACC
WRITE(4,1000)TIME,VC,ALPHA(27),BETA(27),ALPHAD,BETAD
WRITE(5,1000)TIME,PW(27),QW(27),RW(27),PD,QD,RD
WRITE(6,1000)TIME,P(27),Q(27),R(27),PD,QD,RD
WRITE(7,1000)TIME,BXVEL(27),BYVEL(27),BZVEL(27),
+BXACC,SYACC,BZACC
WRITE(8,1000)TIME,(QI(I),I=0,3)
WRITE(9,1000)TIME,XPOS,YPOS,ALT,qe/400.000
END IF

END IF

CALL INPUT(TIME,VC,HWANG,MUINC)
GOTO 15

1000 FORMAT(1X,1F7.3,3E12.4)
1001 FORMAT(1AS,1F7.3,1AS,1F12.3,1AS,I4)
1004 FORMAT(1X,1F7.3,3E19.6)
STOP
END

C-----------------------------------------------C
C SUBROUTINE ADVANC PREPARES THE TIME HISTORY FOR FINDING THE SLOPE AT
C THE ENDPOINT BY CHANGING THE POSITION BY ONE TIME STEP. THIS MUST BE
C USED WHEN CALCULATING THE DERIVATIVE IN THE INITIAL CALCULATION BUT NOT
C IN THE NEWTON SOLVER.
C
C INPUT AND OUTPUT: HIST(27) = TIME HISTORY OF STATE
C-----------------------------------------------C
SUBROUTINE ADVANC(HIST)
DOUBLE PRECISION HIST(27)
INTEGER I
DO 10 I=1,26
10 HIST(I)=HIST(I+1)
RETURN
END

C-----------------------------------------------C
C SUBROUTINE ATMOSPHERE "APPROXIMATES" A STANDARDS ATMOSPHERE PROFILE OF
C DENSITY AND SPEED OF SOUND. IT IS THE PRODUCT OF 4 PIECEWISE LEAST
C SQUARES CURVETITS USING DATA FROM U.S. STANDARD ATMOSPHERE, 1962 FROM
C "AERODYNAMICS FOR ENGINEERS", BERTIN, JOHN J. AND SMITH, M.L. 1979
C P. 6.
C INPUT:
C ALT: ALTITUDE (FT)
C
C OUTPUT:
C RHO: ATMOSPHERIC DENSITY (SLUG/FT^3)
C AS: SPEED OF SOUND (FT/S)
C
C NOTES:
C DATA ENDORSED 0 - 100,000 FT
C MAX ERROR APPROX 11%
C
C******************************************************************************
C SUBROUTINE ATMOS(ALT,RHO,AS)
C DOUBLE PRECISION ALT,RHO,AS
C DOUBLE PRECISION R1,R2,R3,R4
C DOUBLE PRECISION A11,A12,A13
C DOUBLE PRECISION A31,A32,A33
C PARAMETER(R1= 2.376364730D-03,
C 1 R2= -6.795398570D-08,
C 2 R3= 6.71470831D-13,
C 3 R4= -2.259179870D-18,
C 4 A11= 1.116398600D+03,
C 5 A12= -3.816722530D-03,
C 6 A13= -7.867478710D-09,
C 7 A31= 9.224622260D+02,
C 8 A32= 7.096340900D-04,
C 9 A33= -2.528454449D-10)
C IF (ALT.LT.0.000) ALT=0.000
C RHO=R1+R2*ALT+R3*ALT^2+R4*ALT^3+ALT^4
C IF(ALT.LE.36165.62645D0) THEN
C AS=A11+A12*ALT+A13*ALT^2
C ELSE IF(ALT.GT.36165.62645D0.AND. ALT.LT.65818.82244D0) THEN
C AS=968.07415D0
C ELSE IF(ALT.GE.65818.82244D0.AND. ALT.LE.100000.000D0) THEN
C AS=A31+A32*ALT+A33*ALT^2
C ELSE
C WRITE(*,*) 'ERROR: ALTITUDE BEYOND RANGE:',ALT
C END IF
C RETURN
C END
C******************************************************************************
C SUBROUTINE AERO CALCULATES THE AERODYNAMIC ALPHA,BETA & CT
C FROM THE ANGLES CHI, GAMMA, MU(IF INCLUDED) AND THE EARTH FIXED
C FORCES THAT WERE CALCULATED IN FORCE ASSUMING CL IS A FUNCTION OF
C ALPHA ONLY.
C ------
C INPUT:
C FORCE(1,2,7)= FXE,FYE,FZE
C CHANG(1,2,3)=CHI,GAMMA,MU

126
C
C OUTPUT:
C  WBANG(1,2,3) = ALPHA, BETA, CT
C
C SUBROUTINES USED:
C  QUALMU
C  TRANS
C  MULT
C
SUBROUTINE AERO(VCM, FORCE, HWANG, WBANG, MU27, aerof, qe)
DOUBLE PRECISION VCM, FORCE(3), HWANG(3), WBANG(3), MU27
DOUBLE PRECISION TME(3,3), TWF(3,3), RHS(3)
DOUBLE PRECISION MUVUG, MUTRUE
DOUBLE PRECISION aerof(3), TXK, DRAG
DOUBLE PRECISION NUM, DEN, QS

* PLANE CHARACTERISTICS
  DOUBLE PRECISION MASS, S, B, CBAR
  COMMON /PLANE/MASS, S, B, CBAR

* ATMOSPHERE CONDITION
  DOUBLE PRECISION G, RHO, ASOUND, MACHO
  COMMON /PHYS/G, RHO, ASOUND, MACHO

* LIFT DERIVATIVES
  DOUBLE PRECISION CLO, CLA, CLQ, CLAD, CLM
  COMMON /LIFT/CLO, CLA, CLQ, CLAD, CLM

* Y FORCE STABILITY & CONTROL DERIVATIVES
  DOUBLE PRECISION CYB, CYP, CYR
  COMMON /Y/CYB, CYP, CYR

* DRAG TERMS
  DOUBLE PRECISION CDO, CDM, K
  COMMON /DRAG/CDO, CDM, K

* MU CHARACTERISTICS
  LOGICAL MUINC
  INTEGER CYLEN
  COMMON /LOG/MUINC, CYLEN

* TIME QUANTITIES
  DOUBLE PRECISION STEP, TIME
  COMMON /CLOCK/STEP, TIME

QS=RHO*VC*VC*S/2.0DC

C
C IF MU IS NOT INCLUDED MUST ASSUME THAT C = 0 TO FIND MU, ALSO ASSUME
C THAT THRUST IN Y DIRECTION IS THRUST(2) = 0
C
C IF (.NOT.MUINC) THEN
NUM=DSIN(HWANG(1))*FORCE(1)-DCOS(HWANG(1))*FORCE(2)

DEN=DSIN(HWANG(2))*DCOS(HWANG(1))*FORCE(1)
++DSIN(HWANG(2))*DSIN(HWANG(1))*FORCE(2)
++DCOS(HWANG(2))*FORCE(3)-MASS*G*DCOS(HWANG(2))

MUGUES=DATAN2(-NUM,-DEN)

CALL QALMUL(CYCLEN,MUGUES,KU27,MUTRUE)
HWANG(3)=MUTRUE

END IF

CALL TRANS(HWANG,TEW,TWE)
CALL MULT(TWE,FORCE,RHS)

aero(2)=(RHS(2)-MASS*G*DCOS(HWANG(2))*DSIN(HWANG(3))
aero(3)=(RHS(3)-MASS*G*DCOS(HWANG(2))*DCOS(HWANG(3))
aero(1)=-(cd+qs+((aero(3))**2.0d0)*K/qs)

Drag=cd*qs+((aero(3))**2.0d0)*K/qs
Twx=RHS(1)+mass*g*dsin(hwang(2))+drag

WBANG(1)=(-aero(3)/qs-CLO)/CLA
WBANG(2)=(-aero(2)/qs)/(-CTB)
WBANG(3)=Twx/qs

RETURN
END

C---------------------------------------------------------------
C SUBROUTINE BODY CALCULATES BODY VELOCITIES AND THEIR DERIVATIVES
C INPUT:
C VC
C WBANG = ALPHA BETA
C
C OUTPUT:
C BKVEL,BYVEL,BSVEL
C BKACC,BYACC,BSACC
C
C SUBROUTINES USED:
C ADVANC
C SLOPE
C
C---------------------------------------------------------------

SUBROUTINE BODY(VC,WBANG,A,BKVEL,BYVEL,BSVEL,BKACC,BYACC,BSACC)
DOUBLE PRECISION A(13,13), XDM(27)
DOUBLE PRECISION VC,WBANG(3)
DOUBLE PRECISION BKVEL(27),BYVEL(27),BSVEL(27)
DOUBLE PRECISION BKACC,BYACC,BSACC,XV,YV,EV

128
* TIME QUANTITIES

   DOUBLE PRECISION STEP,TIME
   COMMON /CLOCK/STEP,TIME

   XV = VC*DCOS(WBANG(1))*DCOS(WBANG(2))
   YV = VC*DSIN(WBANG(2))
   ZV = VC*DSIN(WBANG(1))*DCOS(WBANG(2))

   CALL ADVANC(BXVEL)
   CALL ADVANC(BYVEL)
   CALL ADVANC(BZVEL)

   CALL SLOPE(27,STEP,A,BXVEL,XDUM,XV,BXACC)
   CALL SLOPE(27,STEP,A,BYVEL,XDUM,YV,BYACC)
   CALL SLOPE(27,STEP,A,BZVEL,XDUM,ZV,BZACC)

   RETURN
END

C---------------------------------------------------------------C
C SUBROUTINE BRATE IS USED TO CALCULATE BODY ANGULAR RATES FROM
C THE BODY EULER ANGLE RATES AND THEN COMPUTE THE DERIVATIVES.
C
C VARIABLES:
C
C INPUT:
C
   WBANG: ALPHA, BETA
C
   ALPHAD, BETAD: DERIVATIVES OF ALPHA & BETA
C
   PW, QW, RW: WIND RATES
C
C OUTPUT:
C
   P, Q, R(27): RATE HISTORIES
C
   PD, QD, RD: ANGULAR ACCELERATIONS IN WIND & BODY AXES
C
C SUBROUTINES USED:
C
   TRANS
C
   MULT
C
   ADVANC
C
   SLOPE
C---------------------------------------------------------------C

SUBROUTINE BRATE(WBANG,ALPHAD,BETAD,A,PW,QW,RW,P,Q,R,PD,QD,RD)

   DOUBLE PRECISION A(13,13),XDUM(27)
   DOUBLE PRECISION WBANG(3)
   DOUBLE PRECISION ALPHAD,BETAD
   DOUBLE PRECISION PW,QW,RW,P(27),Q(27),R(27)
   DOUBLE PRECISION PD,QD,RD
   DOUBLE PRECISION RATE(3)
   DOUBLE PRECISION TMB(3,3),TMH(3,3),RHS(3),BACANG(3)

* TIME QUANTITIES

   DOUBLE PRECISION STEP,TIME
COMMON /CLOCK/STEP,TIME

RHS(1)=PW+ALPHAD*OSIN(WBANG(2))
RHS(2)=Q*ALPHAD*DCOS(WBANG(2))
RHS(3)=RW-BETAD

BAOANG(1)=WANG(2)
BAOANG(2)=WANG(1)
BAOANG(3)=0.000

CALL TRANS(BAOANG,TWB,TBW)
CALL MULT(TBW,RHS,RATE)

CALL ADVANC(P)
CALL ADVANC(Q)
CALL ADVANC(R)

CALL SLOPE(27,STEP,A,P,XDUM,RATE(1),PD)
CALL SLOPE(27,STEP,A,Q,XDUM,RATE(2),QD)
CALL SLOPE(27,STEP,A,R,XDUM,RATE(3),RD)

RETURN
END

C------------------------------------------------------------------------C
C SUBROUTINE EULPAR SOLVES THE NONLINEAR EQUATIONS FOR EULER PARAMETERS
C Q0..3 AS GIVEN IN TBH.
C THIS ROUTINE USES NEUTON'S METHOD TO SOLVE THE NON-LINEAR SYSTEM.
C
C NEWTON'S METHOD FOR SYSTEMS ALGORITHM P.539
C "NUMERICAL ANALYSIS", BURDEN, R.L. AND J.D. FAIRES, 4TH ED. PWS-KENT
C BOSTON, 1989.
C
C "SUBROUTINE EULPAR", HUNING, BRUCE C. 1991
C
C INPUT: CHI,GAMMA,MU,ALPHA,BETA : EULER ANGLES
C OUTPUT: QI(0..3) : EULER PARAMETERS.
C
C SUBROUTINES USED:
C FN1
C JACOB1
C NC
C GAUSS
C------------------------------------------------------------------------C

SUBROUTINE EULPAR(CHI,GAMMA,MU,ALPHA,BETA,QI)
DOUBLE PRECISION CHI,GAMMA,MU,ALPHA,BETA,QI(0:3),PI2,TOL
DOUBLE PRECISION LHS(4),X(4),A(3),C(3),S(3)
DOUBLE PRECISION TBH(3,3),FX(4),MFX(4),Y(4),JX(4,4)
LOGICAL FOUND
INTEGER I,J,K

C------------------------
TOL=1.0D-16
PI2=DCOS(-1.0D0)/2.0D0
J=0

CALL TBHI(CHI,GAMMA,MU,ALPHA,BETA,TBH)

A(1)=(CHI-BETA)/2.0D0
A(2)=(GAMMA+ALPHA)/2.0D0
A(3)=MU/2.0D0

IF(ABS(ABS(A(1))-PI2).LT.TOL) A(1)=A(1)-0.5D0
IF(ABS(ABS(A(2))-PI2).LT.TOL) A(2)=A(2)-0.5D0
IF(ABS(ABS(A(3))-PI2).LT.TOL) A(3)=A(3)-0.5D0

LHS(1)=TBH(1,2)
LHS(2)=TBH(1,3)
LHS(3)=TBH(2,1)
LHS(4)=TBH(2,2)
15 J=J+1
C(1)=DCOS(A(1))
C(2)=DCOS(A(2))
C(3)=DCOS(A(3))

S(1)=DSIN(A(1))
S(2)=DSIN(A(2))
S(3)=DSIN(A(3))

X(1)=C(1)*C(2)*C(3)+S(1)*S(2)*S(3)
X(2)=C(1)*C(2)*S(3)-S(1)*S(2)*C(3)
X(3)=C(1)*S(2)*C(3)+S(1)*C(2)*S(3)
X(4)=S(1)*C(2)*C(3)-C(1)*S(2)*S(3)
K=0
FOUND=.FALSE.
DO WHILE(.NOT.FOUND)
CALL FN1(X,LHS,FX)
CALL JACOB1(X,JX)
DO 10 I=1,4
10 MFX(I) =FX(I)
CALL GAUSS(4,4,JX,MFX,Y)
DO 20 I=1,4
20 X(I)=X(I)+Y(I)
CALL NC(4,Y,FOUND)
K=K+1
IF((X.GT.16).AND.(J.LT.3)) THEN
   A(3)=A(3)-0.5D0
   GO TO 15
END IF
IF(K.GE.30) THEN
   FOUND=.TRUE.
   WRITE(*,*) 'ERROR : EULER PARAMETER NOT CONVERGED'
   STOP
END IF
END DO

DO 30 I=1,4
30 QI(I-1)=X(I)
RETURN
END

C=================================================================================================
C SUBROUTINE FN1 RETURNS THE F(X) OF THE 1, 2, 1, 3, 2, 3 & 2, 3, 3 EULER PARAMETERS
C=================================================================================================
SUBROUTINE FN1(X,LHS,F)
  DOUBLE PRECISION X(4),LHS(4),F(4)
  DOUBLE PRECISION Q0,Q1,Q2,Q3
  Q0=X(1)
  Q1=X(2)
  Q2=X(3)
  Q3=X(4)

  F(1)=-LHS(1)+2.0D0*(Q1*Q2+Q3*Q0)
  F(2)=-LHS(2)+2.0D0*(Q1*Q3-Q2*Q0)
  F(3)=-LHS(3)+2.0D0*(Q2*Q3+Q1*Q0)

  F(4)=-LHS(4)+Q0*Q0-Q1*Q1+Q2*Q2-Q3*Q3
RETURN
END

C=================================================================================================
C SUBROUTINE FORCES CALCULATES THE INERTIAL FORCES ON THE CRAFT
C BY FINDING THE ACCELERATIONS IN THE INERTIAL REFERENCE FRAME AND
C MULTIPLYING THEM BY MASS
C
C INPUT:
C TIME = TIME THAT SAMPLING WAS TAKEN
C VC = TOTAL VELOCITY OF CRAFT AT TIME.
C HWANG(1,2,3) = CHI,GAMMA,HU : EULER ANGLES
C EXVEL, EyVEL, EzVEL = EARTH FIXED VELOCITY HISTORIES
C
C OUTPUT:
C FORCE(1,2,3) = FXS, FYE, FZE FORCES IN INERTIAL REFERENCE FRAME
C EXACC, EyACC, EzACC = EARTH FIXED ACCELERATIONS
C
C SUBROUTINES USED:
C ADVANCE
C SLOPE
C
C=================================================================================================
SUBROUTINE FORCES(VC,HWANG,A,EXVEL,EYVEL,EZVEL,FORCE,
  + EXACC,EYACC,EZACC)
  DOUBLE PRECISION VC,HWANG(3)
  DOUBLE PRECISION A(13,13),XDUM(27),FORCE(3)
  DOUBLE PRECISION EXVEL(27),EYVEL(27),EZVEL(27)
DOUBLE PRECISION EXACC, EYACC, EZACC, XV, YV, ZV

* TIME QUANTITIES
  DOUBLE PRECISION STEP, TIME
  COMMON /CLOCK/STEP, TIME

* PLANE CHARACTERISTICS
  DOUBLE PRECISION MASS, S, B, CBAR
  COMMON /PLANE/MASS, S, B, CBAR

  XV = VC * DCOS(HWANG(2)) * DCOS(HWANG(1))
  YV = VC * DCOS(HWANG(2)) * DSIN(HWANG(1))
  ZV = -VC * DSIN(HWANG(2))

  CALL ADVANC(EXVEL)
  CALL ADVANC(EYVEL)
  CALL ADVANC(EZVEL)

  CALL SLOPE(27, STEP, A, EXVEL, XDUM, XV, EXACC)
  CALL SLOPE(27, STEP, A, EYVEL, XDUM, YV, EYACC)
  CALL SLOPE(27, STEP, A, EZVEL, XDUM, ZV, EZACC)

  FORCE(1) = EXACC * MASS
  FORCE(2) = EYACC * MASS
  FORCE(3) = EZACC * MASS
  RETURN
END

C-----------------------------------------------C
C SUBROUTINE HWDOT CALCULATES THE EULER ANGLE RATES (CHI, GAMMA, MU)
C OF THE CRAFT BASED ON THE TIME HISTORIES OF THE ANGLES OF THE CRAFT.
C THIS ROUTINE USES A LEAST SQUARES APPROXIMATION (3RD ORDER POLY.)
C TO FIND THE ACCELERATION AT THE LAST POINT.
C
C INPUT:
C    HWANG(1,2,3) = CHI, GAMMA, MU
C OUTPUT:
C    CHI, GAMMA, MU : UPDATED ANGLE BANKS
C    CHID, GAMMAD, MUD : ANGLE DERIVATIVES
C
C SUBROUTINES USED:
C    ADVANCE
C    SLOPE
C-----------------------------------------------C

SUBROUTINE HWDOT(HWANG, A, CHI, GAMMA, MU, CHID, GAMMAD, MUD)
DOUBLE PRECISION HWANG(3)
DOUBLE PRECISION A(13,13), XDUM(27)
DOUBLE PRECISION CHI(27), GAMMA(27), MU(27)
DOUBLE PRECISION CHID, GAMMAD, MUD

* TIME QUANTITIES
DOUBLE PRECISION STEP, TIME
COMMON /CLOCK/STEP, TIME
CALL ADVANC(CHI)
CALL ADVANC(GAMMA)
CALL ADVANC(MU)
CALL SLOPE(27,STEP,A,CHI,KDUM, HANG(1),CHD)
CALL SLOPE(27,STEP,A,GAMMA,KDUM,HANG(2),GAMMAD)
CALL SLOPE(27,STEP,A,MU,KDUM, HANG(3),MUD)
RETURN
END

C------------------------------------------------------------
C SUBROUTINE JACBI1 RETURNS THE JACOBIAN MATRIX FOR EULER PARAMETERS USING
C THE 1, 2, 1, 3 & 3, 3 ELEMENTS OF THE TSM TRANSFORMATION MATRIX
C------------------------------------------------------------

SUBROUTINE JACBI1(X,J)
DOUBLE PRECISION X(4),J(4,4)
DOUBLE PRECISION Q0,Q1,Q2,Q3
Q0=X(1)
Q1=X(2)
Q2=X(3)
Q3=X(4)

J(1,1) = 2.0D0*Q3
J(1,2) = 2.0D0*Q2
J(1,3) = 2.0D0*Q1
J(1,4) = 2.0D0*Q0
J(2,1) = -2.0D0*Q2
J(2,2) = 2.0D0*Q3
J(2,3) = -2.0D0*Q0
J(2,4) = 2.0D0*Q1
J(3,1) = 2.0D0*Q1
J(3,2) = 2.0D0*Q0
J(3,3) = 2.0D0*Q3
J(3,4) = 2.0D0*Q2
J(4,1) = 2.0D0*Q0
J(4,2) = -2.0D0*Q1
J(4,3) = 2.0D0*Q2
J(4,4) = -2.0D0*Q3
RETURN
END

C------------------------------------------------------------
C SUBROUTINE INIT Initializes all time histories to equilibrium
C CONDITIONS:
C
C OUTPUT:
C ALPH(A(27))
C BETA (27)
C CHI (27)
C GAMMA(27)
C MU (27)
C PW (27)
C QW (27)
C RW (27)
C P (27)
C Q (27)
C R (27)
C EXVEL(27)
C EYVEL(27)
C EZVEL(27)
C sxVEL(27)
C syVEL(27)
C SxVEL(27)
C A (4,4)
C-------------------------------------------------------------------
SUBROUTINE INIT(OUTYPE,VC,ALPHA,BETA,CHI,GAMMA,MU,PW,QW,RW,P,Q,R,
+ EXVEL, EYVEL, EZVEL, SXVEL, SYVEL, SxVEL, SzVEL,
+ A)
DOUBLE PRECISION ALPHA(27),BETA(27),CHI(27),GAMMA(27),MU(27)
DOUBLE PRECISION PW(27),QW(27),RW(27),P(27),Q(27),R(27)
DOUBLE PRECISION EXVEL(27),EYVEL(27),EZVEL(27)
DOUBLE PRECISION SXVEL(27),SYVEL(27),SxVEL(27)
DOUBLE PRECISION A(13,13),XDOM(27)
DOUBLE PRECISION VC,NxWANG(3)
INTEGER I,OUTYPE

* PLANE CHARACTERISTICS
DOUBLE PRECISION MASS,S,B,CSAR
COMMON /PLANE/MASS,S,B,CSAR

* LIFT STABILITY & CONTROL DERIVATIVES
DOUBLE PRECISION CLO,CLA,CLQ,CLAD,CLM
COMMON /LIFT/CLO,CLA,CLQ,CLAD,CLM

* Y FORCE STABILITY & CONTROL DERIVATIVES
DOUBLE PRECISION CYB,CYP,CYR
COMMON /Y/CYB,CYP,CYR

* DRAG TERMS
DOUBLE PRECISION CDO,CDM,K
COMMON /DRAG/CDO,CDM,K

* ATMOSPHERE CONDITION
DOUBLE PRECISION G,RHO,ASOUND,MACHO
COMMON /PHYS/G,RHO,ASOUND,MACHO

* MU CHARACTERISTICS

135
LOGICAL MUINC
INTEGER CYCLEN
COMMON /LOG/MUINC,CYCLEN

* TIME QUANTITIES
DOUBLE PRECISION STEP,TIME
COMMON /CLOCK/STEP,TIME

CALL LEASTA(13,STEP,27.A.XDUM)

CALL INPUT(TIME,VC,HWANG,MUINC)

C
C-----ALPHA, BETA, CHI, GAMMA, MU INITIAL CONDITIONS--------------
C
DO 20 I=1,27
ALPHA(I)=(2.0D0*MASS*G/(RHO*VC*VC+5)-CLO)/CLA
BETA(I)=0.0D0
CHI(I)=HWANG(1)
GAMMA(I)=HWANG(2)
IF (MUINC) THEN
   MU(I)=HWANG(3)
ELSE
   MU(I)=0.0D0
END IF
20 CONTINUE

C
C-----EXVEL, BXVEL STC. INITIAL CONDITIONS------------------------
C
DO 10 I=1,27
BXVEL(I)=VC*DCOS(ALPHA(27))
BYVEL(I)=0.0D0
BXVEL(I)=VC*DSIN(ALPHA(27))
EXVEL(I)=VC*DCOS(HWANG(2))*DCOS(HWANG(1))
ETVEL(I)=VC*DCOS(HWANG(2))*DSIN(HWANG(1))
10 EXVEL(I)=-VC*DSIN(HWANG(2))

CYCLEN=1

C
C---------------------------------------------
C
DO 30 I=1,27
PW(I)=0.0D0
QW(I)=0.0D0
RW(I)=0.0D0
P(I)=0.0D0
Q(I)=0.0D0
R(I)=0.0D0
30 CONTINUE

2000 FORMAT(A7,6A12)
C-----------------STANDARD HEADINGS-------------------------------------

IF (OUTYPE.EQ.1) THEN
WRITE(1,2000) 'TIME','VC','CHI','GAMMA','MU'
WRITE(2,2000) 'TIME','CHI','GAMMA','MU',
+ 'CHIDOT','GAMMADOT','MUDOT'
WRITE(3,2000) 'TIME','XEDOT','YEDOT','ZEDOT',
+ 'KEZDOT','YEZDOT','ZEZDOT'
WRITE(4,2000) 'TIME','VC',
+ 'ALPHA','BETA','ALPHADOT','BETADOT'
WRITE(5,2000) 'TIME','F','QW','FW',
+ 'PDOT','QDOT','RDOT'
WRITE(6,2000) 'TIME','P','Q','R',
+ 'PDOT','QDOT','RDOT'
WRITE(7,2000) 'TIME','U','V','W','UDOT',
+ 'VDOT','RDOT'
WRITE(8,2000) 'TIME','Q0','Q1','Q2','Q3'
WRITE(9,2000) 'TIME','X','Y','ALT'
END IF

C-------------------SPECIALIZED SHORT HEADINGS-------------------------------

IF (OUTYPE.EQ.2) THEN
WRITE(1,2000) 'TIME','Q0','Q1','Q2','Q3'
WRITE(2,2000) 'TIME','U','UDOT','V','W'
WRITE(3,2000) 'TIME','P','PDOT','Q',
+ 'QDOT','R','RDOT'
WRITE(4,2000) 'TIME','CHI',
+ 'GAMMA','MU','ALPHA','BETA'
END IF

RETURN
END

C--------------------------------------------

C SUBROUTINE INPUT IS RESPONSIBLE FOR PROVIDING TIME HISTORIES OF
C VC AND CHI,GAMMA, MU
C
C INPUT: LAST TIME
C
C OUTPUT: TIME OF VALUES, VC, CHI, GAMMA, POSSIBLY MU

C-----------------------------

SUBROUTINE INPUT(TIME,VC,HWANG,MUINC)
DOUBLE PRECISION TIME,VC,HWANG(3)
LOGICAL MUINC

IF (MUINC) THEN
READ(100,*,END=10) TIME,VC,HWANG(1),HWANG(2),HWANG(3)
ELSE
READ(100,*,END=10) TIME,VC,HWANG(1),HWANG(2)
END IF

137
RETURN
10 WRITE(*,*) 'PROGRAM TERMINATED BY END OF FILE'
STOP
END

C-----------------------------------------------------------------------------
C SUBROUTINE ITERAT CALCULATES THE ITNUM-TH VALUE ALPHA, BETA, P, Q, R
C IN THE ITERATIVE SECTION OF FLY.
C INPUT:
C       FIXED
C       FM,QM,RW : WIND RATES
C       WINDF  : WIND FORCE COEFFICIENTS (CT-CD, CC, CL)
C       DM     : CHANGE IN MACH
C VARIABLE
C       ALPHA, BETA : LAST ALPHA, BETA
C       P, Q, R     : LAST BODY ANGLE RATES
C
C OUTPUT:
C       ALPHA, BETA : NEXT ALPHA, BETA
C       P, Q, R     : NEXT BODY ANGLE RATES
C
C NOTES:
C       ITERATIVE SCHEME HALTS WHEN THE LAST VALUES OF ALPHA, BETA, P, Q & R
C       APPROACH THE NEXT VALUES OF THE SAME
C
C-----------------------------------------------------------------------------

SUBROUTINE ITERAT(A,aerof,Vc,qs,DM,FM,GM,RW,
+                ALPHA,ALPHAD,BETA,BETAD,P,Q,R)
DOUBLE PRECISION A(13,13),FM,GM,RW,ALPHA(27),BETA(27)
DOUBLE PRECISION ALPHAD,BETAD
DOUBLE PRECISION P,Q,R
DOUBLE PRECISION aerof(3),qs,DM,MBANG(3)
DOUBLE PRECISION RATE(3),XDUM(27)
DOUBLE PRECISION TW(3,3),TMB(3,3),RHS(3),BAQANG(3)
DOUBLE PRECISION CD,CL,MLong,Elong,VLr,VC

* PLANE CHARACTERISTICS
DOUBLE PRECISION S,B,CBAR
COMMON /PLANE/MASS,S,B,CBAR

* ATMOSPHERE CONDITION
DOUBLE PRECISION G,RHO,ASOUND,MACH
COMMON /PHYS/G,RHO,ASOUND,MACH

* LIFT DERIVATIVES
DOUBLE PRECISION CLO,CLA,CLQ,CLAD,CLM
COMMON /LIFT/CLO,CLA,CLQ,CLAD,CLM

* Y FORCE STABILITY & CONTROL DERIVATIVES
DOUBLE PRECISION CYB,CYP,CTY
COMMON /Y/CYB,CYP,CTY

138
* DRAG TERMS
  
  DOUBLE PRECISION CDO, CDM, K
  COMMON /DRAG/CDO, CDM, K

* MU CHARACTERISTICS
  
  LOGICAL MUINC
  INTEGER CYCLEN
  COMMON /LOG/MUINC, CYCLEN

* TIME QUANTITIES
  
  DOUBLE PRECISION STEP, TIME
  COMMON /CLOCK/STEP, TIME

  IF (VC.LT.1.0D-5.OR.B.LT.1.0D-5.OR.CBAR.LT.1.0D-5)
  + WRIT(101,*) 'ERROR AT TIME=', TIME
  KLONQ=CBAR/(2*VC)
  KLDIRB=(2*VC)
  WBANG(1)=-(aerof(3)/QS-CLO*KLONQ)*KLONG*Q 
  + CLAD*KLONG*ALPHAD*CLM*OM)/CLA

  CY=(aerof(1)*Dsin(BETA(27))+aerof(2)*Dcos(BETA(27)))/QS
  WBANG(2)=(CY-CYR*KLDIRB-CYP*KLDIRB)*P/CYB

  CALL SLOPE(27, STEP, A, ALPHA, XDUM, WBANG(1), ALPHAD)
  CALL SLOPE(27, STEP, A, BETA, XDUM, WBANG(2), BETAD)

  RHS(1)=FW*ALPHAD*Dsin(WBANG(2))
  RHS(2)=FW*ALPHAD*Dcos(WBANG(2))
  RHS(3)=FW*BETAD

  BAOANG(1)=-WBANG(2)
  BAOANG(2)=WBANG(1)
  BAOANG(3)=0.0

  CALL TRANS(BAOANG, TWD, TBN)
  CALL MULT(TBN, RHS, RATE)
  P=RATE(1)
  Q=RATE(2)
  R=RATE(3)

  RETURN
  END

C----------------------------------------------------------------------------------------
C subroutine mult is a simple matrix multiplication routine for
C multiplying A(3,3)*B(3)=C(3). it should never be used for anything
C more or less extensive.
C----------------------------------------------------------------------------------------
C subroutine mult(A,B,C)
DOUBLS PRECISION A(3,3), B(3), C(3), SUM
INTEGER I,J
DO 10 I=1,3
   SUM=0.0
10  SUM=SUM+A(I,J)*B(J)
10  C(I)=SUM
RETURN
END

C SUBROUTINE NC(DIM,Y,FOUND)
C NORM CHECKING FOR NEWTON METHOD SOLVER
C-----------------------------------------------
SUBROUTINE NC(DIM,Y,FOUND)
   INTEGER DIM
   DOUBLE PRECISION Y(DIM),MAX,TOL
   INTEGER I
   LOGICAL FOUND
   TOL=1.0D-12
   MAX=Y(1)
   DO 10 I=1,DIM
10   IF(DABS(Y(I)).GT.MAX) MAX=DABS(Y(I))
   IF (MAX.LT.TOL) FOUND=.TRUE.
RETURN
END

C SUBROUTINE QUALMU USES KNOWN INFORMATION ABOUT MU TO RECONSTRUCT
C THE PROPER TRAJECTORY ANGLE MU SUCH THAT MU MAY BE BIGGER THAN
C PI OR LESS THAN -PI
C INPUT:
C CYCLEN : NUMBER OF MU ROTATION
C MUGUES : RESULT OF DATAN2(-NUM,-DEP)
C MU : LAST VALUE OF MU
C OUTPUT:
C MUTRUE : ACTUAL VALUE OF MU
C-----------------------------------------------
SUBROUTINE QUALMU(CYCLEN,MUGUES,MU,MUTRUE)
   DOUBLE PRECISION MUGUES,MU,MUTRUE,PI
   INTEGER CYCLEN
   PI=DCOS(-1.0D0)
   MUTRUE=(CYCLEN-1)*PI+MUGUES
   IF ((MUTRUE-MU).LT.-6.0D0) THEN
      CYCLEN=CYCLEN+2
      MUTRUE=(CYCLEN-1)*PI+MUGUES
   ELSE IF ((MUTRUE-MU).GT.6.0D0) THEN
      CYCLEN=CYCLEN-2
      MUTRUE=(CYCLEN-1)*PI+MUGUES
   END IF
RETURN
END
SUBROUTINE TBH1 FINDS TBH BY MULT TBH*TBH.

INPUT: CHI, GAMMA, MU, ALPHA, BETA

OUTPUT: TBH(3,3)

SUBROUTINES USED:
TRANS

SUBROUTINE TBH1(CHI, GAMMA, MU, ALPHA, BETA, TBH)
DOUBLE PRECISION CHI, GAMMA, MU, ALPHA, BETA, TBH(3,3)
DOUBLE PRECISION CC, SC, CG, SG, CM, SM, MBS(3,3), MBS(3,3),
INTEGER I, J, X

CC=DCOS(CHI)
SC=DSIN(CHI)
CG=DCOS(GAMMA)
SG=DSIN(GAMMA)
CM=DCOS(MU)
SM=DSIN(MU)

MBS(1,1)=CG*CC
MBS(1,2)=CG*SC
MBS(1,3)=-SG
MBS(2,1)=SM*SG*CC-CM*SC
MBS(2,2)=SM*SG*SC+CM*CC
MBS(2,3)=SM*CG
MBS(3,1)=CM*SG*CC+SM*SC
MBS(3,2)=CM*SG*SC-SM*CC
MBS(3,3)=CM*CG

CC=DCOS(-BETA)
SC=DSIN(-BETA)
CG=DCOS(ALPHA)
SG=DSIN(ALPHA)
CM=1.0D0
SM=0.0D0

MBS(1,1)=CG*CC
MBS(1,2)=CG*SC
MBS(1,3)=-SG
MBS(2,1)=SM*SG*CC-CM*SC
MBS(2,2)=SM*SG*SC+CM*CC
MBS(2,3)=SM*CG
MBS(3,1)=CM*SG*CC+SM*SC
MBS(3,2)=CM*SG*SC-SM*CC
MBS(3,3)=CM*CG

DO 20 I=1,3
   DO 20 J=1,3

20 CONTINUE
SUM=0.0D0
DO 30 K=1,J
  30 SUM=SUM+MBW(I,K)*MMH(K,J)
TBH(I,J)=SUM
RETURN
END

C---------------------------------------------------------------------C
C SUBROUTINE TRANS CALCULATES THE TRANSFORMATION MATRIX T14 AND THE
C INVERSE T41.
C T41 => S4 = T41 * S1
C
C
C TBH TBH TBH
C
C 4 | WIND BODY BODY
C T41 1 | HORIZ WIND HORIZ
C
C ANGLE(1)=CHI, -BETA, PSI NZ
C ANGLE(2)=GAMMA, ALPHA, THETA NY
C ANGLE(3)=MU, 0, PHI NX
C---------------------------------------------------------------------C
SUBROUTINE TRANS(ANGLE,T14,T41)
DOUBLE PRECISION ANGLE(3),T14(3,3),T41(3,3)
DOUBLE PRECISION CX,CY,CZ,SY,SZ
INTEGER I,J
C
CX=DCOS(ANGLE(1))
CY=DCOS(ANGLE(2))
CZ=DCOS(ANGLE(3))
SZ=DSIN(ANGLE(1))
SY=DSIN(ANGLE(2))
SK=DSIN(ANGLE(3))
C
T41(1,1)=CY*CZ
T41(1,2)=CY*SZ
T41(1,3)=SK
T41(2,1)=SX*SY*CZ-CX*SZ
T41(2,2)=SX*SY*SZ+CX*CZ
T41(2,3)=SX*CY
T41(3,1)=CX*SY*CZ+SX*SZ
T41(3,2)=CX*SY*SZ-SX*CZ
T41(3,3)=CX*CY
C
DO 10 I=1,J
  DO 10 J=1,J
  10 T14(I,J)=T41(J,I)
RETURN
END

C---------------------------------------------------------------------C
C SUBROUTINE WBDOT CALCULATES THE EULER ANGLE RATES (ALPHA, BETA) OF
C THE CRAFT BASED ON THE TIME HISTORIES OF THE ANGLES OF THE CRAFT.

142
C THIS ROUTINE USES A LEAST SQUARES APPROXIMATION (3RD ORDER POLY.)
C TO FIND THE ACCELERATION AT THE LAST POINT.
C
C INPUT:
C   WBANG(1,2)=ALPHA,BETA
C OUTPUT:
C   ALPHA,BETA : UPDATED ANGLE BAKES
C   ALPHAD,BETAD : ANGLE DERIVATIVES
C
C SUBROUTINES USED:
C   ADVANC
C   SLOPE
C
C---------------------------------------------------------------------
C SUBROUTINE WBDOT(WBANG,A,ALPHA, BETA, ALPHAD, BETAD)
DOUBLE PRECISION WBANG(3)
DOUBLE PRECISION A(13,13),XDOM
DOUBLE PRECISION ALPHA(27),BETA(27)
DOUBLE PRECISION ALPHAD,BETAD

* TIME QUANTITIES
DOUBLE PRECISION STEP,TIME
COMMON /CLOCK/STEP,TIME

CALL ADVANC(ALPHA)
CALL ADVANC(BETA)

CALL SLOPE(27,STEP,A,ALPHA,XDOM,WBANG(1),ALPHAD)
CALL SLOPE(27,STEP,A,BETA, XDOM,WBANG(2),BETAD)

RETURN
END

C---------------------------------------------------------------------
C SUBROUTINE WRATE IS USED TO CALCULATE WIND RATES FROM THE WIND
C EULER ANGLE RATES AND THEN COMPUTE THE DERIVATIVES.
C
C VARIABLES:
C
C INPUT:
C   HWANG,CHI,CHI,NU
C   CHID,GAMMA,MU:DERIVATIVES
C
C OUTPUT:
C   PW,QW,RW(27) : RATE HISTORIES
C   PMID,QMD,RMD : ANGULAR ACCELERATIONS IN WIND AXES
C
C SUBROUTINES USED:
C   ADVANC
C   SLOPE
C
C---------------------------------------------------------------------
C SUBROUTINE WRATE(HWANG,CHID,GAMMA,MU,PW,QW,RW,PMID,QMD,RMD)
DOUBLE PRECISION A(13,13),XDOM(27)
DOUBLE PRECISION HWANG(3)
DOUBLE PRECISION CHID,GAMMAD,MU
DOUBLE PRECISION PW(27),QW(27),RW(27)
DOUBLE PRECISION PWD,QWD,RWD
DOUBLE PRECISION RATE(3)

* TIME QUANTITIES
DOUBLE PRECISION STEP,TIME
COMMON /CLOCK/STEP,TIME

RATE(1)=MU-DSIN(HWANG(2))*CHID

RATE(2)=DCOS(HWANG(3))*GAMMAD
  +DSIN(HWANG(3))*DCOS(HWANG(2))*CHID

RATE(3)=-DSIN(HWANG(3))*GAMMAD
  +DCOS(HWANG(3))*DCOS(HWANG(2))*CHID

CALL ADVANC(PW)
CALL ADVANC(QW)
CALL ADVANC(RW)

CALL SLOPE(27,STEP,A,PW,XDUM,RATE(1),PWD)
CALL SLOPE(27,STEP,A,QW,XDUM,RATE(2),QWD)
CALL SLOPE(27,STEP,A,RW,XDUM,RATE(3),RWD)

RETURN
END

C----------------------------------------------------------------------------------
C----------------------------------------------------------------------------------
SUBROUTINE SLOPE(NUM,STEP,A12,XHIST,XDUM,LAST,XNEW)
INTEGER NUM,1
DOUBLE PRECISION LAST
DOUBLE PRECISION XHIST(NUM)
DOUBLE PRECISION XDUM

DOUBLE PRECISION POLY1(2),POLY2(3),POLY3(4)
DOUBLE PRECISION POLY4(5),POLY5(6),POLY6(7)
DOUBLE PRECISION POLY7(8),POLY8(9),POLY9(10)
DOUBLE PRECISION POLY10(11),POLY11(12),POLY12(13)
DOUBLE PRECISION POLYN(13)

DOUBLE PRECISION SUM1, TLAST1
DOUBLE PRECISION A12(13,13), B12(13)

DOUBLE PRECISION STEP,XDUM(NUM)
DOUBLE PRECISION PC

TLAST1=(NUM / 2)*STEP
XHIST(NUM)=LAST

CALL LEASTP13, STEP, XHIST, NUM, B12, XDIM)

CALL GAUSS13, 2, A12, B12, POLY1
CALL GAUSS13, 3, A12, B12, POLY2
CALL GAUSS13, 4, A12, B12, POLY3
CALL GAUSS13, 5, A12, B12, POLY4
CALL GAUSS13, 6, A12, B12, POLY5
CALL GAUSS13, 7, A12, B12, POLY6
CALL GAUSS13, 8, A12, B12, POLY7
CALL GAUSS13, 9, A12, B12, POLY8
CALL GAUSS13, 10, A12, B12, POLY9
CALL GAUSS13, 11, A12, B12, POLY10
CALL GAUSS13, 12, A12, B12, POLY11
CALL GAUSS13, 13, A12, B12, POLY12

POLYNEW1 = PC(1) * POLY1(1) + PC(2) * POLY2(1) + PC(3) * POLY3(1)
++ PC(4) * POLY4(1) + PC(5) * POLY5(1) + PC(6) * POLY6(1)
++ PC(7) * POLY7(1) + PC(8) * POLY8(1) + PC(9) * POLY9(1)
++ PC(10) * POLY10(1) + PC(11) * POLY11(1) + PC(12) * POLY12(1)

POLYNEW2 = PC(1) * POLY1(2) + PC(2) * POLY2(2) + PC(3) * POLY3(2)
++ PC(4) * POLY4(2) + PC(5) * POLY5(2) + PC(6) * POLY6(2)
++ PC(7) * POLY7(2) + PC(8) * POLY8(2) + PC(9) * POLY9(2)
++ PC(10) * POLY10(2) + PC(11) * POLY11(2) + PC(12) * POLY12(2)

POLYNEW3 = PC(2) * POLY2(3) + PC(3) * POLY3(3)
++ PC(4) * POLY4(3) + PC(5) * POLY5(3) + PC(6) * POLY6(3)
++ PC(7) * POLY7(3) + PC(8) * POLY8(3) + PC(9) * POLY9(3)
++ PC(10) * POLY10(3) + PC(11) * POLY11(3) + PC(12) * POLY12(3)

POLYNEW4 = PC(3) * POLY3(4)
++ PC(4) * POLY4(4) + PC(5) * POLY5(4) + PC(6) * POLY6(4)
++ PC(7) * POLY7(4) + PC(8) * POLY8(4) + PC(9) * POLY9(4)
++ PC(10) * POLY10(4) + PC(11) * POLY11(4) + PC(12) * POLY12(4)

POLYNEW5 = PC(4) * POLY4(5) + PC(5) * POLY5(5) + PC(6) * POLY6(5)
++ PC(7) * POLY7(5) + PC(8) * POLY8(5) + PC(9) * POLY9(5)
++ PC(10) * POLY10(5) + PC(11) * POLY11(5) + PC(12) * POLY12(5)

POLYNEW6 = PC(5) * POLY5(6) + PC(6) * POLY6(6)
++ PC(7) * POLY7(6) + PC(8) * POLY8(6) + PC(9) * POLY9(6)
++ PC(10) * POLY10(6) + PC(11) * POLY11(6) + PC(12) * POLY12(6)

POLYNEW7 =
PC(6)*POLY6(7)

++

PC(7)*POLY7(7)+PC(8)*POLY8(7)+PC(9)*POLY9(7)

++

PC(10)*POLY10(7)+PC(11)*POLY11(7)+PC(12)*POLY12(7)

POLNEW(8)=

++

PC(7)*POLY7(8)+PC(8)*POLY8(8)+PC(9)*POLY9(8)

++

PC(10)*POLY10(8)+PC(11)*POLY11(8)+PC(12)*POLY12(8)

POLNEW(9)=

++

PC(8)*POLY8(9)+PC(9)*POLY9(9)

++

PC(10)*POLY10(9)+PC(11)*POLY11(9)+PC(12)*POLY12(9)

POLNEW(10)=

++

PC(9)*POLY9(10)

++

PC(10)*POLY10(10)+PC(11)*POLY11(10)+PC(12)*POLY12(10)

POLNEW(11)=

++

PC(10)*POLY10(11)+PC(11)*POLY11(11)+PC(12)*POLY12(11)

POLNEW(12)=

++

PC(11)*POLY11(12)+PC(12)*POLY12(12)

POLNEW(13)=PC(12)*POLY12(13)

SUM1=0.0DO

DO 60 I=1,12

60 SUM1=SUM1+DBLE(I)*POLNEW(I+1)*TLAST1**DBLE(I-1)

XNEW=SUM1

RETURN

END

C==================================================================================================
C SUBROUTINE LEASTA IS USED TO INITIALLY CALCULATE THE LEAST SQUARES
C A MATRIX. AFTER THE INITIAL CALCULATION, A
C IS CONSTANT AND THE DATA IS SHIFTED IN TIME.
C IN A FORTRAN COMPILER THAT ALLOWS 0.0**0.0 THE IF STATEMENT
C IS NOT NECESSARY.
C==================================================================================================
C
SUBROUTINE LEASTA(OP1,STEP,NUM,A,X)

INTEGER OP1

DOUBLE PRECISION STEP,A(OP1,OP1),X(NUM)

INTEGER NUM

INTEGER I,J,L

DO 5 I=1,NUM

5 X(I)=(NUM / 2)*STEP+DBLE(I-NUM)*STEP

DO 10 I=0,OP1-1

DO 10 J=0,OP1-1

10 A(I+1,J+1)=0.0DO

146
DO 10 I=0,OP1-1
DO 10 J=0,OP1-1
DO 20 L=1,NUM
IF((I+J).EQ.0) THEN
   A(I+1,J+1)=A(I+1,J+1)+1.0D0
ELSE
   A(I+1,J+1)=A(I+1,J+1)*X(L)**(I+J)
END IF
20 CONTINUE
RETURN
END

C==============================================================================

C SUBROUTINE LEASTB IS USED TO CALCULATE THE LEAST SQUARES
C B VECTOR.
C IN A FORTRAN COMPILER THAT ALLOWS 0.0**0.0 THE IF STATEMENT
C IS NOT NECESSARY.
C===============================================================================

SUBROUTINE LEASTB(OP1,STEP,Y,NUM,B,X)
INTEGER OP1
DOUBLE PRECISION X(NUM,,Y(NUM),B(OP1),STEP
INTEGER NUM
INTEGER I,L
DO 5 I=1,NUM
5 X(I)=(NUM / 2)+STEP+DBLE(I-NUM)*STEP

DO 10 I=0,OP1-1
10 B(I+1)=0
DO 20 I=0,OP1-1
DO 20 L=1,NUM
IF(I.EQ.0) THEN
   B(I+1)=B(I+1)+Y(L)
ELSE
   B(I+1)=B(I+1)+X(L)**(I)*Y(L)
END IF
20 CONTINUE
RETURN
END

C===============================================================================

DOUBLE PRECISION FUNCTION PC(N)
INTEGER N,I,N2
DOUBLE PRECISION SUM
SUM=0.0D0
N2=N
IF (N.GT.12) N2=12
IF (N.LE.1) N2=2
DO 10 I=12,N2,-1
10 SUM=SUM+1.0D0/DBLE(I)
PC=SUM/11.0D0
RETURN
END
C GAUSS ELIMINATION WITH SCALED COLUMN PIVOTING P.330
C "NUMERICAL ANALYSIS", BURDEN, R.L. AND J.D. FAIRES, 4TH ED. PWS-KENT
C BOSTON, 1989.
C
C "SUBROUTINE GAUSS", MUNRO, BRUCE C. 1991
C
C AMAIN, B ORIGINAL MATRIX AND VECTOR PASSED FOR AX=b
C
A = (A|B)

C---------------------------------------------------------------C
SUBROUTINE GAUSS(DIMAT,DIM,AMAIN,B,X)
  INTEGER DIM,DIMAT,P,NROW(15),NCPY,I,J,K
  DOUBLE PRECISION A(15,16),AMAIN(DIMAT,DIMAT),B(DIMAT),X(DIM)
  DOUBLE PRECISION S(15),Q1,Q2,M(15,15)
  DO 5 I=1,DIM
    DO 3 J=1,DIM
      3 A(I,J)=AMAIN(I,J)
      5 A(I,DIM+1)=B(I)

  DO 10 I=1,DIM
    S(I)=DABS(A(I,1))
    DO 20 J=2,DIM
      IF (DABS(A(I,J)) .GT. S(I)) THEN
        S(I)=DABS(A(I,J))
      END IF
      20 CONTINUE
  
  IF (S(I).EQ.0.0D0) THEN
    WRITE(*,*) 'PROGRAM ERROR GAUSS SCHEME: S(I)=0'
    STOP
  END IF
  NROW(I)=I
  10 CONTINUE

  DO 30 I=1,DIM-1
    P=I
    Q1=DABS(A(NROW(P),I)/S(NROW(P)))
  
    DO 40 J=I,DIM
      Q2=DABS(A(NROW(J),I)/S(NROW(J)))
      IF (Q2.GT.Q1) THEN
        Q1=Q2
        P=J
      END IF
      40 CONTINUE

  IF(A(NROW(P),I).EQ.0.0D0) THEN
    WRITE(*,*) 'PROGRAM ERROR GAUSS SCHEME: NO SOLUTION 1'
    STOP

148
END IF
IF(NROW(I).NE.NROW(P)) THEN
   NCOPY=NROW(I)
   NROW(I)=NROW(P)
   NROW(P)=NCOPY
END IF

C
DO 50 J=I+1,DIM
   M(NROW(J),I)=A(NROW(J),I)/A(NROW(I),I)
   DO 50 K=1,DIM+1
      A(NROW(J),K)=A(NROW(J),K)-M(NROW(J),I)*A(NROW(I),K)
   CONTINUE
50 CONTINUE
30 CONTINUE

C
IF (A(NROW(DIM),DIM).EQ.0.0D0) THEN
   WRITE(*,*) 'PROGRAM ERROR GAUSS SCHEME: NO SOLUTION 2'
   STOP
END IF

C
X(DIM)=A(NROW(DIM),DIM+1)/A(NROW(DIM),DIM)
DO 70 I=DIM-1,1,-1
   Q1=0.0D0
   DO 80 J=I+1,DIM
      Q1=Q1+A(NROW(I),J)*X(J)
   70 X(I)=(A(NROW(I),DIM+1)-Q1)/A(NROW(I),I)
RETURN
END

C-----------------------------------------------------------------------------------C
C---End Program---------------------------------------------------------------------C
C-----------------------------------------------------------------------------------C
C-----------------------------------------------------------------------------------C
C-----------------------------------------------------------------------------------C
VITA

The author was born in Dallas, Texas but has lived most of his life in Wilmington, Delaware. His parents are Carl Munro, retired Corporate Director in the Quality Area for the E. I. DuPont de Nemours Company, and Helen Munro, a high school computer programming teacher. They supported and inspired the author’s interest in art, science, and mathematics through his high school career at Brandywine High School. He graduated with honors and enrolled at Virginia Polytechnic Institute & State University in order to pursue a degree in Aerospace Engineering. During his undergraduate years, the author had been fortunate to participate in the Co-op program with Martin Marietta Aero & Naval Systems, a summer internship with DuPont Space Structures, and the Virginia Tech NASP Mockup Team. In May of 1990, he graduated cum laude and decided to continue in the masters program at Virginia Tech concentrating in aircraft dynamics and control. The author is currently under research contract at NASA Langley Research Center participating in a joint NASA, U. S. Navy and Virginia Tech control specification study for fighter aircraft. Upon graduation, he hopes to continue his career in advanced controls development.

Bruce C. Munro