1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

TECHNIQUES FOR EXAMINING NONLINEAR FLUTTER

by

ROBIN M. TARBET B.Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

MASTER OF ENGINEERING

Department of Mechanical and Aerospace Engineering
Ottawa-Carleton Institute for Mechanical and
Aerospace Engineering

Carleton University

Ottawa, Ontario

September 14, 1990

© copyright

1990, Robin Tarbet

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilm. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

ISBN 0-315-60454-9

The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis

"TECHNIQUES FOR EXAMINING NONLINEAR FLUTTER"

submitted by Robin M. Tarbet, B. Eng.

in partial fulfilment of the requirements for
the degree of Master of Engineering

Thesis Supervisor

Chair, Department of Mechanical and Aerospace Engineering

Carleton University

Abstract

Flutter damage is avoided by determining a safe operating regime. The presence of nonlinear elements can produce a unique form of flutter response known as a limit cycle. This form of response consists of neutrally stable oscillations which can cause damage through fatigue. Limit cycle flutter response can occur within the safe operating regime calculated for a linear system.

An iterative method for solving nonlinear flutter equations is used to determine flutter velocities for representative two-dimensional wing sections. The method linearizes the equations through the use of describing functions. Results obtained using the iterative procedure are compared with published results for linear and nonlinear systems with two and three degrees of freedom.

A time integration method is also used to solve the nonlinear flutter equations. The results obtained using this method are compared with the results from the iterative procedure.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. P.E. Barrington, for his guidance of my research and of the writing of this thesis. His ability to quickly solve seemingly insurmountable problems was especially appreciated. I would also like to thank him for always being there when I needed to discuss aspects of my work.

I would also like to thank my parents, both for their encouragement throughout my academic career, and for their patience during the writing of this thesis.

Table of Contents

	Page
Acceptance Sheet	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	viii
List of Figures	ix
List of Symbols	xii
 CHAPTER 1: INTRODUCTION	 1
 CHAPTER 2: BACKGROUND	 5
2.0 SOME DEFINITIONS	5
2.1 NONLINEAR FLUTTER	9
2.1.1 Nonlinear Systems	9
2.1.2 Limit Cycle Behaviour	11
2.2 LINEARIZATION TECHNIQUES	16
2.3 HISTORY OF NONLINEAR FLUTTER ANALYSIS	21
 CHAPTER 3: MODELLING LINEAR FLUTTER SYSTEMS	 25
3.0 EQUATIONS OF MOTION	25
3.1 DETERMINING THE AERODYNAMIC LOADS	30
3.1.1 Steady State Aerodynamics	30
3.1.2 Theodorsen's Circulation Function	31
3.2 SOLUTION TECHNIQUES	34
3.3 USE OF MODE SHAPES TO REDUCE SYSTEM MATRICES	37

	Page
CHAPTER 4: ITERATIVE PROCEDURE FOR NONLINEAR FLUTTER ANALYSIS	39
4.0 DETAILS OF THE METHOD	39
4.1 SOFTWARE IMPLEMENTATION	42
4.1.1 Parameter Entry Section	42
4.1.2 Matrix Element Calculation	45
4.1.3 Reduced Frequency Loop	46
4.1.4 Aerodynamic Calculations	49
4.1.5 Nonlinear Systems	50
 CHAPTER 5: TIME INTEGRATION OF FLUTTER EQUATIONS	 53
5.0 CONVERSION OF AERODYNAMIC COEFFICIENTS	53
5.1 EVALUATING THE DUHAMEL INTEGRAL	57
5.2 REARRANGING THE FLUTTER EQUATIONS OF MOTION	59
5.3 CHOICE OF TIME INTEGRATION METHOD	60
5.4 TIME INTEGRATION SOFTWARE	62
5.4.1 System Definition Section	64
5.4.2 Aerodynamic Coefficient Calculation	65
5.4.3 Matrix Calculation	65
5.4.4 Time Integration Loop	66
 CHAPTER 6: VALIDATION OF SIMULATION PROCEDURES	 68
6.0 LINEAR SYSTEMS WITH TWO DEGREES OF FREEDOM	69
6.0.1 Iterative Procedure	69
6.0.2 Time Integration Procedure	71
6.1 LINEAR SYSTEMS WITH THREE DEGREES OF FREEDOM	72
6.1.1 Iterative Procedure	72
6.1.2 Time Integration Procedure	74
6.2 NONLINEAR SYSTEMS WITH TWO DEGREES OF FREEDOM	75
6.2.1 Iterative Procedure	76

	Page
6.2.2 Time Integration Procedure	77
6.3 NONLINEAR SYSTEMS WITH THREE DEGREES OF FREEDOM	78
6.3.1 Iterative Procedure	78
6.3.2 Time Integration Procedure	79
6.4 CONCLUSIONS	79
 CHAPTER 7: RECOMMENDATIONS FOR FUTURE WORK	 116
 CHAPTER 8: CONCLUSIONS	 119
 REFERENCES	 122
 TABLES	 126
 APPENDIX A: AERODYNAMIC LOADS AND COEFFICIENTS	 132
 APPENDIX B: ITERATIVE PROCEDURE SOFTWARE	 142
 APPENDIX C: TIME INTEGRATION FORMS OF EQUATIONS OF MOTION	 181
 APPENDIX D: TIME INTEGRATION SOFTWARE	 196

List of Tables

	PAGE
6-1 Parameters Describing Systems Used in Two Degree of Freedom Linear Tests	127
6-2 Validation of Iterative Calculation Procedure For Several Two Degree of Freedom Systems	128
6-3 Validation of Time Integration Procedure For Several Two Degree of Freedom Systems	129
6-4 Parameters Describing Systems Used in Three Degree of Freedom Linear Tests	130
6-5 Validation of the Time Integration Method Using a Three Degree of Freedom Wing System	131

List of Figures

	Page
2-1 Typical Wing-Aileron System Cross-Section	6
2-2 Symmetrical Bilinear Spring Characteristics	10
2-3 Bilinear Spring with Freeplay and Preload	12
2-4 Attracting Limit Cycle on the Phase Plane	14
2-5 Unstable Limit Cycle Separating Domains of Attraction	15
3-1 Representation of a Two Dimensional Wing System	26
4-1 Information Flow Through Iterative Program	43
4-2 Information Flow Through Reduced Frequency Loop	47
5-1 Information Flow Through Time Integration Program	63
6-1 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	81
6-2 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	82
6-3 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	83
6-4 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	84
6-5 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	85
6-6 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	86
6-7 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	87
6-8 Flutter Parameter as a Function of Reduced Mass Parameter	88
6-9 Flutter Parameter as a Function of Reduced Mass Parameter	89

	Page
6-10 System Modal Response Showing Torsional Divergence	90
6-11 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	91
6-12 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	92
6-13 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	93
6-14 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	94
6-15 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	95
6-16 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	96
6-17 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	97
6-18 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	98
6-19 Flutter Parameter as a Function of Plunge-Pitch Frequency Ratio	99
6-20 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	100
6-21 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	101
6-22 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	102
6-23 Flutter Parameter as a Function of Aileron-Pitch Frequency Ratio	103
6-24 V-g Curve For Figure 6-20 With Frequency Ratio of 0.4	104
6-25 System Response to Second Flutter Mode in Figure 6-20	105
6-26 System Response to Primary Flutter Mode in Figure 6-20	106
6-27 Flutter Characteristics of System Including Effects of Structural Damping	107

	Page
6-28 Modal Response of Three Degree of Freedom System Over Short Time Period	108
6-29 Modal Response of Three Degree of Freedom System Over Long Time Period	109
6-30 Pitch Limit Cycle Amplitude as a Function of Velocity Ratio $\delta_P = .2 \quad \delta_T = .5$	110
6-31 Pitch Limit Cycle Amplitude as a Function of Velocity Ratio $\delta_P = .0 \quad \delta_T = .5$	111
6-32 Comparison of Describing Functions For Case With $\delta_P = .2$ and $\delta_T = .5$	112
6-33 Comparison of Describing Functions For Case With $\delta_P = .0$ and $\delta_T = .5$	113
6-34 Aileron Limit Cycle Amplitude Ratio as a Function of Velocity Ratio $\delta_P = \delta_T = .2$	114
6-35 Aileron Limit Cycle Amplitude Ratio as a Function of Velocity Ratio $\delta_P = .1 \quad \delta_T = .2$	115

List of Symbols

A	-	Amplitude of Motion; General Aerodynamic Load
a	-	Reduced Location of Elastic Axis; Amplitude of Motion
B	-	Duhamel Integral Function
b	-	Wing Semi-Chord Length
C_l	-	Lift Coefficient
$C(\kappa)$	-	Theodorsen Circulation Function
c	-	Reduced Location of Aileron Hinge Axis; Wing Chord Length
D	-	Dissipative (Damping) Energy
d	-	Reduced Location of Control Tab Hinge Axis; Displacement Damping Parameter
e	-	Reduced Location of Aileron Leading Edge
F	-	Restoring Force; Lifting Force; Aerodynamic Load
f	-	Reduced Location of Control Tab Leading Edge
g	-	Damping Coefficient
h	-	Vertical Translation of Wing System
I	-	Mass Moment of Inertia per Unit Length
K	-	Spring Stiffness Value; Stiffness Matrix
k	-	Inverse of Reduced Mass Parameter; Stiffness Value
L	-	Lift Force
M	-	Mass Matrix; Pitching Moment
m	-	Mass per Unit Length of Wing System
n	-	Number of Cycles
P	-	Vertical Force at Leading Edge of Aileron
Q	-	Moment About Control Tab Hinge
q	-	General Degree of Freedom
R	-	Vertical Force at Leading Edge of Control Tab
r	-	Radius of Gyration; Distance Along Mean Line of Wing
S	-	Static Unbalance
s	-	Reduced Time Parameter
T	-	Total Kinetic Energy; Moment About Aileron Hinge
t	-	Time
U	-	Potential (Strain) Energy
V	-	Velocity
w	-	Vertical Translation
X	-	Kussner's Geometric Function
x	-	Reduced Location of Centre of Gravity; Displacement
y	-	Vertical Translation of Control Tab
z	-	Vertical Translation of Aileron

Greek

α	-	Pitch Angle; Angle of Attack; Wagner Function Constant
α_R	-	Scale Factor
β	-	Aileron Deflection Angle; Wagner Function Constant
γ	-	Eigenvalue
δ	-	Control Tab Deflection Angle; Spring Displacement
δW	-	Virtual Work
δq	-	Virtual Displacement
ϵ	-	Convergence Criterion; Small Perturbation
ζ	-	Damping Coefficient
κ	-	Reduced Frequency
λ	-	Damping Coefficient
μ	-	Reduced Mass Parameter
ν	-	Circular Frequency
ξ	-	Geometric Coordinate
ρ	-	Air Density
τ	-	Reduced Leading Edge Gap Magnitude; Dummy Time Variable
Φ	-	Row Matrix of Mode Shape Values
$\Phi(s)$	-	Wagner Function
ϕ	-	General Degree of Freedom; Kussner's Geometric Function; Phase Angle
Ψ	-	Column of Modal Amplitudes
ψ	-	Kussner's Geometric Function
Ω	-	Complex System Frequency
ω	-	Natural Frequency

Superscripts

0	-	Initial Condition
1	-	Altered Condition
~	-	Instantaneous Value; Generalized Matrix Form

Subscripts

0	-	Initial Value
h	-	Plunge Equation
e	-	Effective or Equivalent Value
i	-	i th Spring
n	-	Number of Wagner Function Coefficient
R	-	Reference Degree of Freedom
SH	-	Control Tab Leading Edge Gap
SR	-	Aileron Leading Edge Gap
α	-	Pitch Equation; Pitch Degree of Freedom; Elastic Axis
β	-	Aileron Rotation Equation; Aileron Rotation Degree of Freedom; Aileron Hinge Axis
δ	-	Control Tab Rotation Equation; Control Tab Rotation Degree of Freedom; Control Tab Hinge Axis
μ	-	Number of Solution Selected for Alignment

Chapter 1

INTRODUCTION

Aeroelasticity is the term used to describe a class of problems concerned with the mutual interaction of aerodynamic, inertial and elastic forces. As aircraft structures are generally lightweight, they can be subject to problems resulting from their flexibility. The structure can deform appreciably under aerodynamic loading, and these deformations can affect both the performance and safety of the aircraft.

The interaction of aerodynamic and elastic forces is evident in the specialized case of aeroelastic deformation known as flutter. Flutter is a dynamic instability caused when the coupling of two or more degrees of freedom produces a self-excited oscillation where the motion extracts energy from the fluid stream. This type of instability occurs most frequently in bodies subjected to large lateral lifting forces.

The onset of flutter in an aircraft structure has the potential for causing great damage. This damage results from the divergent oscillations which can occur in certain flight regimes. To prevent aircraft damage, measures are taken to ensure that the aircraft will never enter a flight condition where flutter could occur. These measures usually involve using the known aircraft properties and some form of aerodynamic theory to predict onset of flutter.

The techniques which are used to solve the flutter stability equations have

been well documented for cases where the flutter system is linear. However, structural nonlinearities in the aircraft system can cause a distinct form of flutter response known as limit cycle flutter. Although a stable limit cycle response may consist of small amplitude oscillations, the aircraft structure may still be damaged through the process of fatigue. Methods must therefore be developed to examine flutter in nonlinear systems, as the flutter solution techniques cannot be used to predict the onset of flutter in these cases.

A number of methods for examining nonlinear flutter have been proposed. Many involve approximating the nonlinear elements using Fourier Series expansions to arrive at an equivalent linearized system. This approach allows the linear solution techniques to be employed. One proposed method uses single term Fourier Series expansions known as describing functions and an iterative routine to examine flutter in a complex system with multiple nonlinear elements.

The industry practice is to examine the flutter characteristics of full- or half-body aircraft systems. To perform the analysis for a complicated system, the aircraft dynamic response would first be obtained using a Finite Element Analysis (FEA) program. The aerodynamic loading upon the system would then be evaluated using a Doublet-Lattice method. Here, the lifting surface is represented as a series of acceleration doublet potentials. This application of linear aerodynamic theory has the ability to deal with three dimensional geometry, and the special cases involving the lift forces near hinge and fold lines on the wing. These methods have been shown¹

to be valid for high subsonic Mach numbers and reduced frequencies of order unity. However, the accuracy of the calculation is dependent upon the number of potentials used in the model, and the method used to sum the strengths of the potentials over the entire surface.

The effects of nonlinearities upon the flutter response of a system may be illustrated using a simplified model of a wing system. The lifting surface is represented by a two dimensional section, and the characteristics of the natural modes are modelled as resulting from the actions of springs which restrain the motion of the section. The aerodynamic loads upon the section are calculated using the unsteady theory developed by Theodorsen. This theory uses coefficients that are functions of the reduced frequency of the system, and is at least as accurate as the Doublet-Lattice method in this case. A form of the iterative solution procedure will be used to predict the onset of flutter in the two-dimensional wing model, which could include nonlinear elements.

Another method of solving the flutter stability problem involves integrating the equations of motion in the time domain. When this method is employed, any nonlinear elements present in the system can be represented exactly, but an indirect method must be used to obtain the value of the flutter speed. An Adams-Moulton predictor-corrector time integration procedure will be used to solve the flutter equations for the two-dimensional wing model described above.

The accuracy of each method will be evaluated by comparing the predictions

made with results published in the literature. The test cases consist of linear and nonlinear systems with two and three degrees of freedom.

Finally, the use of the two methods of flutter prediction will be discussed with reference to their potential for future work.

Chapter 2

BACKGROUND

2.0 SOME DEFINITIONS

Figure 2-1 illustrates a cross-section of a wing-aileron combination positioned in an airflow with freestream velocity V . The whole system is assumed to pivot about the elastic axis. This is the location on the wing where a vertical load can be placed without producing a torsional deflection of the wing. An alternate definition is that a moment about the elastic axis will not produce a bending moment in the wing. Using the position of the elastic axis as a reference, and employing the notation of Smilg and Wasserman², the wing has deflected by an angle α relative to the direction of flow, and has translated vertically by an amount h from the undisturbed position. The deflection of the aileron, β , is defined relative to the mean line of the wing, and is positive in the sense shown.

The inertial properties of the wing-aileron system are usually specified in terms of a mass per unit length, and mass moments of inertia per unit length about the elastic axis and the aileron hinge axis. The mass per unit length of the wing, m , is typically specified in terms of a reduced mass parameter (μ). The parameter

FIGURE 2-1: TYPICAL WING-AILERON SYSTEM CROSS-SECTION

expresses the mass per unit length of the wing as a fraction of the mass of a cylinder of air that has a diameter equal to the chord of the wing

$$\mu = \frac{m}{\pi \rho b^2}$$

In this expression, ρ is the density of the air, and b is the wing semi-chord value. Some authors work in terms of a reduced mass parameter which is the inverse of μ and which is denoted by k . The mass moment of inertia per unit length of the wing-aileron system about the elastic axis is denoted by I_α , and is usually specified in terms of a reduced parameter known as the radius of gyration (r_α) which is given by

$$r_\alpha^2 = \frac{I_\alpha}{mb}$$

The mass moment of inertia per unit length of the aileron (and tab if present) about the aileron hinge axis is denoted by I_β , and is expressed as the reduced parameter r_β^2 .

The key locations of the wing system are expressed as fractions of the semi-chord value. Many are defined relative to the mid-chord position of the airfoil, with a positive sense towards the trailing edge of the airfoil. The locations of the wing centre of gravity and the elastic axis are expressed as the reduced parameters x_g and a , where x_g is measured from the elastic axis. The aileron hinge axis and the aileron leading edge positions are defined by the parameters c and e . The size of the gap at

the aileron leading edge is specified by the parameter τ_{SR} which is a fraction of the whole wing chord. The aileron centre of gravity location is given by the parameter x_g , but is defined relative to the aileron hinge axis. The effect of the system masses acting through their centre of gravity locations can be expressed in terms of the static unbalance of the system. The static unbalance of the system about the elastic axis is denoted by S_α , and is defined in the following manner

$$S_\alpha = mbx_\alpha$$

The static unbalance about the aileron hinge axis is similarly denoted by S_δ .

Should the system include a control tab, the tab would be considered to deflect through an amount δ relative to the aileron. The hinge and leading edge locations would be specified by the reduced parameters d and f . The tab leading edge gap would be denoted by the fraction of the total chord τ_{SH} . All of the remaining parameters used to define the tab properties would be analogous to the equivalent aileron terms.

2.1 NONLINEAR FLUTTER

2.1.1 Nonlinear Systems

Nonlinearities can occur in the structural behaviour of an aircraft system. The nature of these nonlinearities can be divided into two groups; concentrated and distributed. The concentrated nonlinearities act locally in a lumped manner. Breitbach³ lists many of the types of nonlinearities that fall into each category. This study shall concern itself primarily with the concentrated type of nonlinearity and their effect upon the flutter characteristics of the system.

Concentrated nonlinearities are generally found in the action of control systems, or at the attachment point for external stores. As a result of a number of factors, including backlash or friction in the control linkages and limitation of the control surface deflection, the load-deflection characteristics of the connection between the control surface and the rest of the system become nonlinear. Typically, the load-deflection relationship will become a polynomial function of the displacement. However, in many instances, the curve may be accurately approximated by a bilinear curve.

Figure 2-2 illustrates a typical bilinear load-deflection curve. In this case, the curve is symmetrical about the origin of the axes, and can be seen to have one

FIGURE 2-2: SYMMETRICAL BILINEAR SPRING CHARACTERISTICS

stiffness value, K_1 over most of the range of displacement. However, over a small portion of the deflection range, the curve possesses another stiffness value K_2 , which can be greater than or less than the value of K_1 . The curve that has been illustrated is known as a hardening spring, as K_2 is less than K_1 .

Figure 2-3 shows a more extreme case of the bilinear spring, a bilinear spring with freeplay and preload. Freeplay describes the condition where the slope K_2 is equal to zero, which can arise as a result of slack in a control cable over a portion of the operating range. Preload simulates the effects of an eccentricity in the position of the freeplay region over the operating range.

2.1.2 Limit Cycle Behaviour

When a structural nonlinearity is present in the airfoil system, the response to a perturbation can be in the form of a limit cycle. In dissipative systems, any transients present will be attracted to one of three possible attractors over a long period of time. The three attractor forms are equilibrium, periodic and chaotic. In the case of periodic attractors, the transients will be attracted to stable periodic motions, which are known as attracting limit cycles. There are a number of references available which discuss limit cycles and nonlinear mechanics^{4,5}.

Having identified the nature of any attractors that are present, the next step

FIGURE 2-3: BILINEAR SPRING WITH FREEPLAY AND PRELOAD

is to associate a domain of attraction with each one. This domain represents the area on the phase $(x, \dot{x})$ plane containing all of the initial conditions which will settle to the attractor. Figure 2-4 shows the case of a single attracting limit cycle on the phase plane. In this case, the entire phase plane is the domain of attraction for the one limit cycle, and possible trajectories are shown. Here, all initial conditions will settle at the limit cycle shown. Figure 2-5 shows a more complex case of an unstable limit cycle separating two attracting limit cycles. Here the phase plane is separated into two domains of attraction, which are divided by the unstable limit cycle. Disturbances in the system originating within the locus of the unstable cycle will cause the system to limit cycle with the smaller of the two amplitudes, while disturbances outside of the unstable locus will produce limit cycles with the larger amplitude.

When the response of the system is in the form of a limit cycle, the resultant steady-state motion is oscillatory. While the amplitude of these oscillations may not be large enough to cause immediate damage as in the case of divergent flutter, even small amplitude motions have the potential for damage through the mechanism of fatigue. This form of response can occur within the flutter envelope that was calculated using the assumption of a linear system.

FIGURE 2-4: ATTRACTING LIMIT CYCLE ON THE PHASE PLANE

FIGURE 2-5: UNSTABLE LIMIT CYCLE SEPARATING DOMAINS OF ATTRACTION

2.2 LINEARIZATION TECHNIQUES

Linear techniques may still be used to solve the flutter stability equations, provided some means can be found to transform the nonlinear problem into a linear form. Many of the linearization techniques used in the study of nonlinear flutter have their origins in a method proposed by Kryloff and Bogoliuboff⁶ and modified by Bogoliuboff and Mitropolsky⁷.

Consider a system whose response is governed by the following weakly nonlinear equation

$$m \frac{d^2 x}{dt^2} + k x = \epsilon f \left(x, \frac{dx}{dt} \right)$$

The system is subject to a linear restoring force (the term kx) in addition to a nonlinear force term, whose magnitude is characterized by the parameter ϵ .

For small values of the parameter ϵ , the solution of the differential equation may be expressed in the following form

$$x = a(t) \cdot \cos(\nu t + \phi(t))$$

where

$$\nu = \sqrt{\frac{k}{m}}$$

For the remainder of the analysis, the term $v t + \phi$ will be replaced by the parameter $\psi(t)$. The system variables may be transformed on the phase plane from $(x, \dot{x})$ to (a, ϕ) using the assumed form of the solution and the relationship

$$\frac{dx}{dt} = -a \cdot v \cdot \sin \psi$$

When the assumed solution is differentiated with respect to time, the following expression results

$$\frac{dx}{dt} = \frac{da}{dt} \cos \psi - a v \sin \psi - a \frac{d\phi}{dt} \sin \psi$$

which implies that

$$\frac{da}{dt} \cos \psi - a \frac{d\phi}{dt} \sin \psi = 0$$

The differential of the assumed solution may also be differentiated with respect to time, resulting in

$$\frac{d^2x}{dt^2} = -\frac{da}{dt} v \sin \psi - a v^2 \cos \psi - a v \frac{d\phi}{dt} \cos \psi$$

Using this expression, the original nonlinear form of the differential equation may be re-written as

$$\frac{da}{dt} v \sin \psi + \frac{d\phi}{dt} a v \cos \psi + \frac{c}{m} f(a \cos \psi, -a v \sin \psi) = 0$$

The two equations in terms of the first order derivatives of the amplitude (a) and the

phase angle (ϕ) can be rearranged to express these derivatives explicitly

$$\frac{da}{dt} = -\frac{\epsilon}{mv} \cdot \sin \psi \cdot f(a \cos \psi, -av \sin \psi)$$

$$\frac{d\phi}{dt} = -\frac{\epsilon}{mav} \cdot \cos \psi \cdot f(a \cos \psi, -av \sin \psi)$$

The derivative of the phase angle ϕ is replaced by the derivative of the parameter ψ using the identity

$$\frac{d\psi}{dt} = v + \frac{d\phi}{dt}$$

Therefore, the derivative of ψ may be written as

$$\frac{d\psi}{dt} = v - \frac{\epsilon}{mav} \cdot \cos \psi \cdot f(a \cos \psi, -av \sin \psi)$$

The equations for the derivatives of a and ψ are periodic with respect to ψ , and have amplitudes of order ϵ , which suggests that the amplitude and phase angle are slowly varying functions of time. If the amplitude and phase are defined in the following manner

$$a(t) = a_0 + O(\epsilon)$$

$$\phi(t) = \phi_0 + O(\epsilon)$$

the expressions for the derivatives of the amplitude and phase angle can therefore

be averaged over one cycle

$$\frac{da}{dt} = -\frac{e}{2\pi m v} \int_0^{2\pi} \sin \psi \cdot f(a \cos \psi, -av \sin \psi) d\psi + O(e^2)$$

$$\frac{d\psi}{dt} = v - \frac{e}{2\pi m a v} \int_0^{2\pi} \cos \psi \cdot f(a \cos \psi, -av \sin \psi) d\psi + O(e^2)$$

The integral terms are equivalent to the fundamental coefficients of the Fourier Series expansion of the function f . The terms of order e^2 are neglected, and the derivative of ψ is defined as the frequency of the response of the nonlinear system, $v_e(a)$

$$v_e(a) = \frac{d\psi}{dt}$$

The square of the nonlinear frequency may be expanded to the following

$$\begin{aligned} v_e^2(a) = & v^2 - \frac{e}{\pi m a} \int_0^{2\pi} \cos \psi \cdot f(a \cos \psi, -av \sin \psi) d\psi \\ & + \frac{e^2}{4\pi^2 m^2 a^2 v^2} \left[\int_0^{2\pi} \cos \psi \cdot f(a \cos \psi, -av \sin \psi) d\psi \right]^2 \end{aligned}$$

For convenience, this expression can be written as

$$v_e^2(a) = v^2 (1 - e B_1 - e^2 B_2)$$

and the term involving e^2 will be ignored here.

An effective stiffness coefficient $k_e(a)$, which is related to the square of the nonlinear frequency, is defined in the following manner

$$k_e(a) = k - \frac{e}{\pi a} \int_0^{2\pi} \cos \psi \cdot f(a \cos \psi, -a v \sin \psi) d\psi$$

In the same manner, an effective damping coefficient can also be defined

$$\lambda_e(a) = \frac{e}{\pi a v} \int_0^{2\pi} \sin \psi \cdot f(a \cos \psi, -a v \sin \psi) d\psi$$

It can be shown that the definitions of the effective damping and stiffness can be used to express the original differential equation in the following linear manner with an error the order of magnitude e^2

$$m \frac{d^2 x}{dt^2} + \lambda_e(a) \frac{dx}{dt} + k_e(a) x = 0$$

The equivalent linearized equation of motion has been obtained by replacing the nonlinear term

$$F = e f\left(x, \frac{dx}{dt}\right)$$

by the linear term

$$F_e = - \left[k_1(a) x + \lambda_e(a) \frac{dx}{dt} \right]$$

where $k_1(a) = k_e(a) - k$.

Therefore, a nonlinear element can be replaced by an effective linear term.

The fundamental term of the Fourier Series expansion of the response of the nonlinear element to a harmonic displacement is referred to as a describing function.

2.3 HISTORY OF NONLINEAR FLUTTER ANALYSIS

Woolston and his associates^{8,9} published two papers in the early 1950's. The first paper described the use of an analog computer simulation to examine the effect of nonlinearities upon a two degree of freedom airfoil system. Three different types of concentrated nonlinearity were listed and investigated. In addition, the simulation results for the case of the freeplay nonlinearity were compared with experimental results obtained using a wind tunnel model. The second paper extended the work to include the additional degree of freedom of a control surface. The analysis was influenced by the work of Theodorsen and Garrick^{10,11}.

Shortly afterwards, Shen¹² published his work, in which he outlined a method for the approximate analysis of nonlinear flutter problems. His technique used the method of Kryloff and Bogoliuboff in the form of a harmonic balance to linearize the system. The goal of this work was an analytical re-examination of the work done previously by Woolston. Shen reported that the results of his analysis qualitatively agreed with those reported by Woolston. Shen also illustrated that the technique

could be used to handle more complex problems, and considered the case of a buckled panel in supersonic flow.

Almost twenty years after Shen's paper, Breitbach³ catalogued examples of both concentrated and distributed forms of nonlinearities. He then examined their effects upon the testing and analysis of aircraft systems. As an example of his analysis, he examined the case of a three degree of freedom wing system, and was able to obtain results which were comparable to those obtained by Shen.

Laurenson and Trn¹³ used the describing function method of equation linearization to investigate the effects of multiple nonlinearities upon missile control systems. Their aerodynamics were simplified, as the aim of their study was an investigation of the effects of the nonlinearities. The primary type of nonlinearity that was dealt with was the freeplay nonlinearity with and without preload.

McIntosh et al.^{14,15} reported on their efforts to construct a typical section model and test it in a wind tunnel. The goal of their investigation was the development of high-quality experimental data for a two degree of freedom wing model with known properties. The model possessed nonlinear springs in both the pitch and plunge modes. An additional feature of their work was the development of an accurate mathematical model of the system. This model used Theodorsen's two dimensional aerodynamics, but these were modified to include the effects of wind tunnel wall interaction, and compressibility. They report good agreement between the experimental results and the theoretical model.

Breitbart¹⁶ published a second paper in which he investigated the case of an airframe with nonlinear control characteristics. The work was an example of the application of the linearization approach to complicated systems with multiple nonlinearities. Bilinear and hysteresis nonlinearities were considered, and Breitbart was able to predict the flutter characteristics of the entire aircraft. The results of his analysis technique were compared with flight test observations.

Lee¹⁷ established a technique for analyzing a system with multiple degrees of freedom containing multiple nonlinearities. The procedure made use of describing functions to linearize the stiffness functions, and then used linear techniques to solve the flutter equation in the frequency domain. The mode shapes and natural frequencies of the system are then shifted to align the oscillatory amplitude in the nonlinear springs with the describing function predictions. Lee also examined the application and resultant error of describing functions. To validate his method, Lee used three examples of varying complexity, one of which was the two degree of freedom airfoil tested by McIntosh et al.^{14,15}

Laurenson et al.¹⁸ published a paper which reported their examination of the limit cycle response of a system with multiple discrete nonlinearities. They used an asymptotic expansion method to linearize the load-displacement relationship. This method was claimed to have greater accuracy than the describing function method, as more terms of the Fourier expansion are retained. Their test case was a aerodynamic surface in a subsonic airstream. The surface possessed two degrees of

freedom at the wing root, and simplified aerodynamics were used to determine the loads upon the system. As in the previous work by Laurenson, the aim of the research was the investigation of the system response behaviour in the presence of nonlinearities. As part of the work, the team also used a numerical integration scheme to evaluate the accuracy of the asymptotic expansion method. This, in turn, required an evaluation of the efficiency and accuracy of several numerical integration schemes.

Chapter 3

MODELLING LINEAR FLUTTER SYSTEMS

3.0 EQUATIONS OF MOTION

The equations of motion for a two dimensional airfoil can be derived using Lagrange's method. For a general degree of freedom denoted by q , Lagrange's equation gives

$$\frac{d}{dt} \left(\frac{\partial T}{\partial \dot{q}} \right) - \frac{\partial T}{\partial q} + \frac{\partial U}{\partial q} + \frac{\partial D}{\partial \dot{q}} = \frac{\partial (\delta W)}{\partial (\delta q)}$$

where T represents the total kinetic energy of the system, U is the system potential energy (virtual work done against conservative forces, such as the structural forces), and D is the dissipative or damping energy. The term δW represents the virtual work done by the system against non conservative forces, usually aerodynamic, and δq is the virtual displacement of the system in the degree of freedom q .

Consider the two dimensional airfoil modelled in Figure 3-1. The airfoil is considered to have two degrees of freedom defined relative to the elastic axis position: plunge and pitch. The motion of the airfoil in these two degrees of freedom is restrained by a pair of linear springs which are modelled as being connected to the

FIGURE 3-1: REPRESENTATION OF A TWO DIMENSIONAL WING SYSTEM

wing at the elastic axis. The springs are assumed to have stiffness values denoted by K_h and K_α respectively. The aerodynamic loads upon the airfoil are represented by a lifting force F and a pitching moment M , which act at the aerodynamic centre of the airfoil. For the purposes of this analysis, the aerodynamic centre is assumed to be located at the 1/4 chord point of the wing.

Let r be the distance along the mean line of the airfoil from the elastic axis, with its positive sense in the direction towards the trailing edge of the airfoil. If the angular displacement α is small, the vertical displacement of the airfoil is given by

$$w = -h - r\alpha$$

The total kinetic energy of the system, T , is found by integrating the square of the velocity term that develops when the vertical displacement function is differentiated with respect to time. Performing this integration over the chord of the airfoil results in

$$T = \frac{1}{2} m \left(\frac{dh}{dt} \right)^2 + mbx_a \frac{d\alpha}{dt} \frac{dh}{dt} + \frac{1}{2} I_a \left(\frac{d\alpha}{dt} \right)^2$$

The strain energy of the system is given by the following expression

$$U = \frac{1}{2} K_h h^2 + \frac{1}{2} K_\alpha \alpha^2$$

Finally, the virtual work performed by the wing is given by

$$\delta W = -F \cdot \left\{ \delta h - \left(\frac{1}{2} + a \right) b \cdot \delta \alpha \right\} + M \cdot \delta \alpha$$

It should be noted that although the above equations were written in terms of geometric coordinates, the method is not restricted to these coordinates. The above formulations were used in order to permit the use of existing expressions for the aerodynamic influence coefficients. As no dissipative elements were included in the model of the wing system, there is no need to evaluate an expression for the dissipative energy.

When Lagrange's equation is applied in the plunge and pitch degrees of freedom, the resulting equations are given in the following matrix form

$$\begin{bmatrix} m & S_\alpha \\ S_\alpha & I_\alpha \end{bmatrix} \cdot \begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \end{Bmatrix} + \begin{bmatrix} K_h & 0 \\ 0 & K_\alpha \end{bmatrix} \cdot \begin{Bmatrix} h \\ \alpha \end{Bmatrix} = \begin{Bmatrix} -F \\ M + F(0.5 + a)b \end{Bmatrix}$$

For convenience, equations of this type are commonly denoted by

$$[M] \{\ddot{\xi}\} + [K] \{\xi\} = \{F\}$$

where $\{\xi\}$ denotes the geometric coordinates of the system, $[M]$ is the system mass matrix, $[K]$ is the system stiffness matrix, and $\{F\}$ is the vector of aerodynamic load terms.

By extending the application of Lagrange's equation, the equations of motion for a two dimensional wing possessing four degrees of freedom can be derived. For

this case, the mass matrix [M] will be given by the following

$$\begin{bmatrix} m & S_a & S_b & S_d \\ S_a & I_a & I_b + S_b b(c-a) & I_b + S_b b(d-a) \\ S_b & I_b + S_b b(c-a) & I_b & I_b + S_b b(d-c) \\ S_d & I_b + S_b b(d-a) & I_b + S_b b(d-c) & I_b \end{bmatrix}$$

while the system stiffness matrix [K] will be given by

$$\begin{bmatrix} K_h & 0 & 0 & 0 \\ 0 & K_a & 0 & 0 \\ 0 & 0 & K_b & 0 \\ 0 & 0 & 0 & K_d \end{bmatrix}$$

These matrices contain all of the terms required to simulate systems with only two or three degrees of freedom.

Although the derivation of the equations of motion neglected the effects of dissipation within the system, the presence of structural damping may be included within the equations. By describing the effect of structural damping as a friction force that is in phase with the velocity with a magnitude proportional to the restoring force, it is possible to replace each of the diagonal terms in the structural stiffness matrix by terms of the form

$$K_q(1 + i g_q)$$

where g_q is the damping coefficient for the degree of freedom q .

3.1 DETERMINING THE AERODYNAMIC LOADS

In the derivation of the two dimensional airfoil equations of motion, the aerodynamic loads upon the wing section were represented by the vector $\{F\}$. The manner in which these loads are modelled determines the type of analysis which can be performed using these equations.

3.1.1 Steady State Aerodynamics

In this modelling technique, the forces and moments acting upon the wing are assumed to result from the steady, divergent motion of the airfoil. The expressions for the lift force and moment are derived by treating the airfoil as a flat plate in potential flow. The lift force upon the wing is typically given in terms of the additional force produced by a small change in the angle of attack of the wing.

This procedure is similar to that used by Pines¹⁹ in his study of classical flutter. The lift force acting upon the wing was defined by the following expression

$$L = \frac{1}{2} \rho V^2 \cdot c \cdot C_{l_\alpha} \cdot \tilde{\alpha}$$

where the term C_{l_α} is a coefficient describing the incremental lift force produced by a change in the angle of attack. The variable $\tilde{\alpha}$ represents the instantaneous angle

of attack, which is defined in the following manner

$$\tilde{\alpha} = \alpha + \frac{1}{V} \frac{dh}{dt}$$

The moment acting upon the airfoil was assumed to be caused by the lift force acting at the length of its moment arm about the elastic axis. No aerodynamic pitching moment was included in the theory.

The use of the instantaneous angle of attack did serve to introduce aerodynamic damping into the equations of motion. However, due to its steady-state nature, the theory does not take into account the oscillatory flow condition created by the flutter response of the wing. This fact limits the usefulness of the steady state theory in predicting the flutter response of a wing system.

3.1.2 Theodorsen's Circulation Function

In the middle 1930's, Theodorsen²⁰ presented an exact form of the unsteady aerodynamic theory describing the flow around an oscillating airfoil. Recognizing that the study of flutter is primarily concerned with the identification of the point of instability, Theodorsen assumed that the motion of the airfoil was harmonic in nature, and of small amplitude. The wing section itself was replaced by an aerodynamically equivalent mean line, and the aerodynamic loads were calculated

using thin wing theory and the theory of nonstationary potential flow.

The aerodynamic loads were assumed to result from the superposition of circulatory and steady-state components. The integration of the flow potential describing the contribution of the circulatory component gave rise to a combination of Bessel functions which were in turn functions of the reduced frequency. The reduced frequency, κ , is a parameter which describes the number of waves present in a length of the wake equal to $2\pi b$. This combination of Bessel functions is therefore a representation of the influence of the wake upon the aerodynamic loading, and come to be known as Theodorsen's circulation function, $C(\kappa)$. The presence of the wake effectively introduces a time lag into the circulatory lift terms. The steady-state component of the aerodynamic loads is created by the oscillatory motion of the airfoil, which induces fluid motion, even when the freestream velocity is equal to zero.

A similar treatment of unsteady flow was developed by Kussner and Schwarz²¹. The motion of a wing-aileron-tab combination was represented by superimposing the contributions of six basic motions. The contributions to the aerodynamic load of each of these motions could then be easily calculated and summed together. The expressions for the aerodynamic loads were presented using geometric functions of the reduced aileron and tab hinge axis locations.

Smilg and Wasserman² represented the unsteady force and moment terms given by Theodorsen and Garrick^{10,11} as coefficients of the six basic motions used by

Kussner and Schwarz. These coefficients were then combined to express the loads as functions of the geometric degrees of freedom. The coefficients quoted by Smilg and Wasserman involved both the circulation function $C(\kappa)$ as well as geometric functions given by both Theodorsen²⁰ and Kussner and Schwarz. These coefficients will be used in the present study, and are listed in Appendix A, having been re-written to involve only those geometric functions listed by Kussner and Schwarz.

The manner in which the unsteady aerodynamic theory was developed can limit its application in practical situations. The assumption that the motion is of small amplitude implies that the fluid flow normal to the mean line will be negligible, and that therefore the flow traces the mean line with constant velocity. Obviously, in situations where the assumption of small amplitude motion is not maintained, the theory will not be able to accurately predict the loading; the theory will not be valid in situations where flow separation occurs. Bisplinghoff²² cites experimental studies which suggest that above a reduced frequency of 0.4, the theoretical values of the lift and moment coefficients diverge from experimentally-determined values. This divergence is ascribed to a lag in the Kutta condition, and the fact that the wake can no longer be considered to be planar.

Similar observations are made by Smilg and Wasserman, who emphasize the fact that the derivation of the theory assumed that the flow was two dimensional in nature. Smilg and Wasserman also suggest that the theory will be unable to accurately predict flutter in cases where control tabs are involved.

A form of aerodynamic theory which combines elements of both the steady-state and oscillatory forms is the quasi-steady theory. Under this assumption, the oscillatory nature of the flow condition created by the airfoil flutter response is accounted for, but the effect of the wake vortices upon the flow is neglected. The theory uses the complex aerodynamic load terms given by Theodorsen and others, but replaces the circulation function $C(\kappa)$ with a constant value of 1.0. This effectively treats the airfoil as a flat plate oscillating in an unsteady flow with zero freestream velocity.

3.2 SOLUTION TECHNIQUES

The solution of the flutter equations begins by assuming that the response of the system is harmonic, which allows the replacement of the geometric coordinates in the following manner

$$\xi = \xi_0 \cdot e^{i\omega t}$$

where ξ_0 is a constant defining the amplitude of the motion in the degree of freedom ξ , and ω is the frequency of the motion. When the geometric coordinates are differentiated with respect to time, the following expression results

$$\frac{d^2\xi}{dt^2} = -\omega^2 \xi_0 e^{i\omega t}$$

The next stage in the solution process is to evaluate the aerodynamic loads upon the system. If the unsteady aerodynamic theory of Theodorsen is used, this typically involves assuming a value of the reduced frequency and evaluating the components of the load vector $\{F\}$ at that reduced frequency. If the aerodynamic load terms are given by the coefficient expressions developed by Smilg and Wasserman², the load vector may be replaced by the following equivalence

$$\{F\} = \omega^2 \cdot [A] \cdot \{\xi\}$$

where the ω^2 term included by Smilg and Wasserman has been excluded from the aerodynamic coefficient matrix $[A]$. Substituting the above terms into the equations of motion given in 3.0 results in

$$-\omega^2 [M] \{\xi\} + [K] \{\xi\} = \omega^2 [A] \{\xi\}$$

where the exponential term has been removed from each of the terms. The above equation can be divided by $-\omega^2$ and rearranged to give

$$(\{[M] + [A]\} - \gamma [K]) \{\xi\} = 0$$

where $\gamma = 1/\omega^2$. The stability boundary for this system is defined as the locus where the determinant of the matrix is equal to zero.

The roots of the complex equation that develops when the determinant of the combined matrix is set equal to zero may be most easily evaluated by determining the eigenvalues of a related form of the system. If the above expression is rearranged in the following manner

$$([K]^{-1}([M] + [A]) - \gamma [I])\{\xi\} = 0$$

the values of γ are the eigenvalues of the matrix

$$[K]^{-1}([M] + [A])$$

and are also the roots of the determinant equation. If a computational algorithm is used to determine the complex eigenvalues of the system, the known value of the reduced frequency can then be used to specify the frequencies and the critical air velocity of the response.

If no such computational method is available, the method of Theodorsen and Garrick¹⁰ may be used. This method involves separating the determinant equation into its real and imaginary components. Each of these new equations is re-arranged to express a parameter X as a function of the reduced frequency. The values of X obtained from the real and imaginary equations are plotted against the reduced frequency, and the solutions to the flutter equations occur where the curves intersect.

3.3 USE OF MODE SHAPES TO REDUCE SYSTEM MATRICES

The analysis of the flutter response of a system requires the use of a number of equations equal to the total number of degrees of freedom of the system. In the case of systems with multiple degrees of freedom, the matrices defining the equations of motion may become so large as to require significant computational power to determine a solution. For this reason, the system matrices are often reduced to their generalized form using the mode shapes of the system.

Essentially, the method involves approximating the true motion of the system by a limited number of component mode shapes. This is accomplished by replacing the geometric coordinate vector using the transformation

$$\{ \xi \} = [Q] \cdot \{ q \}$$

where $[Q]$ is a matrix whose columns are the mode shapes being used to approximate the system motion. The equations of motion can now be expressed as

$$[Q]^T \cdot [M] \cdot [Q] \cdot \{q\} + [Q]^T \cdot [K] \cdot [Q] \cdot \{q\} = [Q]^T \cdot \{F\}$$

or

$$[\tilde{M}] \{q\} + [\tilde{K}] \{q\} = [\tilde{A}] \{q\}$$

where the $\sim$ symbol denotes the generalized form of the system matrix, and $\{q\}$ is the vector of the generalized degrees of freedom.

If the normal modes of the system are employed, the generalized form of the system matrices will be diagonal due to the orthogonal properties of the normal modes.

Chapter 4

ITERATIVE PROCEDURE FOR NONLINEAR FLUTTER ANALYSIS

4.0 DETAILS OF THE METHOD

The method used to examine the flutter of a system with multiple degrees of freedom and multiple nonlinearities is based upon the work presented by Lee¹⁷. The method uses linear solution techniques and aligns the oscillatory amplitudes of the nonlinear degrees of freedom with predictions made using describing functions. Restricting the implementation of the iterative procedure to the solution of two dimensional systems allows a number of simplifications to be made in Lee's process.

At the start of the procedure some nominal value is assumed for each of the nonlinear spring stiffness values, and the mode shapes and natural frequencies of the system are calculated. The inertial properties of the system, as well as the aerodynamic influence coefficients are expressed in matrix form, and the mode shapes are used to reduce the system matrices to their generalized form.

The flutter equations of motion can now be expressed in the following manner

$$([\tilde{M} - \tilde{A}] + \gamma [\tilde{K}]) \{ q \} = 0$$

where γ is the complex eigenvalue of the system, and $\{q\}$ is a column vector containing the degrees of freedom of the reduced system. Each flutter solution will produce a distinct combination of amplitudes in the nonlinear degrees of freedom.

At this point, one of the solutions of the linear system may be selected for alignment with the nonlinear spring stiffness values. This produces a dynamic system tuned to a particular flutter frequency. A reference displacement, d_R is assumed in one of the system degrees of freedom, and a scale factor α_R is calculated

$$d_R = \alpha_R \cdot [\Phi_R] \cdot [\Psi_\mu]$$

In the above expression, $[\Phi_R]$ is a row matrix defining the mode shape components for the reference degree of freedom, and $[\Psi_\mu]$ is the column mode shape belonging to the μ^{th} solution which has been selected for alignment.

For any given nonlinear spring linking two degrees of freedom in a two dimensional system, the net displacement δ_i is found using the scale factor calculated earlier

$$\delta_i = \alpha_R \cdot \sum_{j=1}^n \phi_{ij} \Psi_{\mu_j}$$

By applying this equation to each of the nonlinear degrees of freedom, a vector containing the complete set of nonlinear spring displacements may be

obtained.

Once the nonlinear spring displacements are known, appropriate describing functions can be used to establish a vector containing the effective stiffness values for each of the nonlinear springs. Lee recommends that the stiffness matrix be corrected at each step through the use of an incremental stiffness matrix. However, the stiffness matrix for a two dimensional system will be diagonal, with a maximum dimension of 4×4 . This allows the system stiffness matrix to be updated more rapidly without the use of the incremental matrix. Once the spring displacements have been calculated, the new stiffness values of all of the springs are calculated and are used to replace the existing elements of the stiffness matrix.

The mode shapes of the altered system are now determined using the free vibration form of the flutter equations, which are cast in the following form

$$[[M]^{(0)} - \gamma [K]^{(1)}] \{ \xi \} = 0$$

where γ is the eigenvalue of the free vibration system, and $\{ \xi \}$ is the column matrix of mode coordinates. The superscripts 0 and 1 refer to the original and altered system states respectively. The solution of these equations will yield a new set of natural frequencies, and the new mode shape matrix is obtained from

$$[\Phi]^{(1)} = [\Phi]^{(0)} \cdot [\xi_1, \xi_2, \dots, \xi_n]$$

Using the new system mode shapes, the generalized dynamic properties of the

system are recalculated, beginning the second step of the iterative process. In general, the process is repeated until some convergence criterion is satisfied. Lee recommends a convergence check of the following form

$$\left| \frac{\delta_i^{(s)} - \delta_i^{(s-1)}}{\delta_i^{(s)}} \right| \leq \epsilon_i \quad i = 1, 2, 3, \dots, n$$

where ϵ_i is some user-defined error limit for the i^{th} nonlinear spring, and s is the iteration step number.

4.1 SOFTWARE IMPLEMENTATION

Appendix B contains the FORTRAN computer program written to implement the iterative solution technique. Figure 4-1 illustrates the information flow through the program, which can be roughly divided into five major sections.

4.1.1 Parameter Entry Section

The first section of the program loads the parameters which define the system. Subroutine ENTRY obtains the system inertial and geometric properties from an

FIGURE 4-1: INFORMATION FLOW THROUGH ITERATIVE PROGRAM

external data file named RUN.DAT which must be created by the user. The parameters that are included in the data file include an eight character name describing the test, the air density and reference semi-chord value. The program also requires a six character description of the nature of the test, linear or nonlinear. The data entry routine then obtains the values of the inertial and geometric properties for each degree of freedom.

The next stage in the solution algorithm is to use the routine SPRING to obtain the characteristics of each of the springs associated with each degree of freedom. Bilinear springs with preload are specified by giving values of the two stiffness values, the preload value and the extent of the nonlinear section. Also included is information concerning the initial displacement and the convergence criterion for each spring. The file also contains the reference displacement and degree of freedom required for the solution of nonlinear systems. Note that the springs are defined using two stiffness values, but that the linear springs are defined to have preload values on the order of $1E8$ so that the second spring stiffness value does not enter into the calculation. This manner of defining the linear springs allows the same describing function routine to be used to calculate the stiffness of both linear and nonlinear elements.

The next subroutine in the data entry section of the software is the routine THPREP. This routine loads a set of values which specify the relationship between the reduced frequency and the Theodorsen circulation function. These values were

taken from the tables published by Luke and Dengler²³, and will allow the program to use linear interpolation to obtain the value of the Theodorsen function at any value of the reduced frequency.

The final routine in the data entry section of the program is QLOAD, which obtains the mode shape matrix for the system from an external file. The routine also determines the transpose of the mode shape matrix, which is required for the calculation used to reduce the system matrices to generalized form.

4.1.2 Matrix Element Calculation

Once the parameters describing the inertial and stiffness properties of the system have been loaded, the next stage in the program is to calculate the elements of the system inertial and stiffness matrices. Routine MLOAD calculates the values of the elements of the inertial matrix, using the formulae given in Chapter 3. Once this matrix has been defined, the mode shape matrices are used to reduce the matrix to its generalized form.

Subroutine KASS then uses the properties of the springs defined earlier to calculate the stiffness values in each degree of freedom. This matrix is also reduced to its generalized form using the mode shape matrices. The generalized form of the matrix is then inverted using a Gauss-Jordan elimination routine (INVRT) so that

the stiffness matrix is in the appropriate form for use in finding the eigenvalues of the system. This arrangement limits the range of cases that can be examined using the software, as errors will result when the program attempts to invert a stiffness matrix containing elements that have zero stiffness.

4.1.3 Reduced Frequency Loop

Figure 4-2 shows the details of the program information flow through the reduced frequency loop. Having calculated the generalized forms of the inertial and stiffness matrices, the program next proceeds to the loop which will solve the flutter equation at a number of values of reduced frequency. The program steps through 200 values of the reduced frequency with a step size determined by the user. The step size determines the maximum value of the reduced frequency used by the program, which in turn affects the range of problems which can be solved by the software.

The first stage in the loop is to use the subroutine INTERP to determine the value of the Theodorsen function at the given value of the reduced frequency.

The next stage is to calculate the elements of the aerodynamic influence matrix, which is then reduced to its generalized form. The generalized aerodynamic matrix is then added to the generalized form of the mass matrix, and the sum is then multiplied by the inverse of the generalized stiffness matrix. The resulting product,

FIGURE 4-2: INFORMATION FLOW THROUGH THE REDUCED FREQUENCY LOOP

which is of the form

$$[\tilde{K}]^{-1} [\tilde{A} + \tilde{M}]$$

is then used to find the eigenvalues and eigenvectors of the system, using the sub-routine CGEEV. CGEEV is an EISPACK routine which calculates the eigenvalues and eigenvectors of an arbitrary complex matrix. The complex system frequencies are obtained by inverting the square roots of the system eigenvalues

$$\Omega = \frac{1}{\sqrt{\gamma}}$$

From the system frequency values, the modal frequency damping and velocity values can be obtained in the following manner

$$\omega = \Re(\Omega)$$

$$d = \Im(\Omega)$$

$$v = \frac{b \cdot \omega}{\kappa}$$

These values are printed in an output file.

The routine which calculates the eigenvalues of the system orders the solutions on the basis of the magnitudes of the eigenvalues. Therefore, before the solutions can be printed in the output file, the order must be changed to coincide with the manner

in which the modes were specified at the start of the program. The ordering of the modal results is accomplished by comparing the magnitudes of the eigenvalues and eigenvectors of the solutions. At the first step in the reduced frequency loop, the routine MAGORD is used to specify the initial order of the output modes. At this value of reduced frequency which most closely approximates the zero velocity case, the routine calculates and normalizes the magnitudes of the eigenvectors found by CGEEV. The modes are then ordered based upon the location of the dominant deflection in each of the geometric degrees of freedom. At each subsequent reduced frequency step, the magnitudes of the solution eigenvalues are calculated and the output solutions are ordered based upon the comparison of the eigenvalue magnitudes with those at the previous step.

4.1.4 Aerodynamic Calculations

Subroutine ALOAD is the routine which actually assigns the elements of the aerodynamic matrix. In the case of two degree of freedom systems, the calculation of the necessary aerodynamic coefficients is performed within the subroutine.

For the case of a system involving an aileron, the additional aerodynamic coefficients are calculated by the subroutine TWELVE. This routine in turn calls the subroutine PCALC which evaluates Kussner's Φ functions using the reduced location

of the aileron hinge. Subroutine TWELVE can also be used to calculate coefficients relating to the presence of a control tab. This is due to the similarity of the coefficients for aileron and tab. For the case of the control tab, the only difference is that the Φ functions are evaluated using the reduced location of the tab hinge. If the control tab degree of freedom is involved, the subroutine FOUR is used to calculate the remaining aerodynamic coefficients required by the program. Subroutine FOUR is also capable of calculating coefficients relating to both the aileron and control tab aerodynamics, and calls additional subroutines to evaluate Kussner's X functions using the appropriate geometric constants.

4.1.5 Nonlinear Systems

If the system being considered is a nonlinear one, the program uses the results obtained from the linear flutter analysis to begin the solution of the nonlinear system.

At the end of the reduced frequency loop, the routines FLTSRCH and FLTDSP find and display the points of instability of the system. FLTSRCH operates by examining the results stored in the output file for cases where the damping value becomes equal to zero or changes sign. The mode number, velocity, reduced frequency and slope of the V-g curve are stored in arrays. These arrays are used by the routine FLTDSP, which displays the results on the computer screen, in increasing

order of velocity. The user is then able to select the solution that will be modified to account for the presence of nonlinearities.

To begin the nonlinear loop, the changes in displacement of the degrees of freedom are calculated and used to determine if the solution has reached the convergence point. If this is not the case, one of the flutter solutions is selected by the user. The reduced frequency is reset to the value at the point of instability, and the spring deflection array is reset.

Using the new reduced frequency value, the system aerodynamic matrix is recalculated, and the system flutter characteristics are evaluated using the routine CGEEV. The mode undergoing flutter is identified based upon the magnitude of its damping parameter, and the column of modal amplitudes is calculated from the eigenvector of the flutter solution. The row matrix of the mode shapes for the reference degree of freedom is determined from the mode shape arrays, and the scale factor for this iteration is calculated. Finally, the net displacements of the linear and nonlinear springs are evaluated.

Knowing the values of the spring displacements, the system stiffness matrix can be recalculated, reduced and inverted. The result is then used to calculate a solution for the free vibration equations for the system which is equivalent to finding the eigenfunctions for the following product

$$[\tilde{K}]^{-1} [\tilde{M}]$$

The mode shapes of the shifted system are calculated from the eigenvectors of the free vibration solution, and the mode shape arrays are reset.

Using the new mode shape arrays, the system stiffness and inertial arrays are recast in their generalized form, and the flutter solution for the system is determined at a number of values of reduced frequency in a manner identical to that used in the linear portion of the program. At this point, the nonlinear iteration repeats.

Chapter 5

TIME INTEGRATION OF FLUTTER EQUATIONS

5.0 CONVERSION OF AERODYNAMIC COEFFICIENTS

The flutter stability problem can also be solved by integrating the equations of motion in the time domain. This method has the advantage of being able to treat nonlinear stiffness elements in an exact manner, and there is the additional benefit of being able to examine the time histories of the motion of the airfoil system.

For a general degree of freedom ϕ , the flutter equations may be expressed in the following form

$$[M] \left\{ \frac{d^2 \phi}{dt^2} \right\} + [K] \{\phi\} = \{A_\phi(\omega)\}$$

where A_ϕ is the aerodynamic load corresponding to the degree of freedom ϕ . The aerodynamic loads are more easily evaluated in the frequency domain, and therefore, the major problem in implementing the time integration scheme lies in transforming these expressions into the time domain.

The following treatment is similar to that presented by Dowell et al.²⁴. The aerodynamic terms in the frequency domain are transformed into the time domain

through the use of an inverse Fourier transform. This implies that

$$\{L(\omega)\} = L(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} L(\omega) e^{i\omega t} d\omega$$

The process of evaluating the inverse Fourier transforms can be illustrated using the lift function for a two dimensional system, as given by Smilg and Wasserman²

$$L(\omega) = \pi \rho b^2 \omega^2 L_h h(\omega) + \pi \rho b^3 \omega^2 \left\{ L_h - \left(\frac{1}{2} + a \right) L_h \right\} \alpha(\omega)$$

From Appendix A, the coefficient L_h is given by

$$L_h = 1 - i \frac{2V}{b\omega} C(\kappa)$$

Therefore, by taking the inverse transform of the first term in the expression for $L(\omega)$, the following result is obtained

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \pi \rho b^2 \omega^2 h(\omega) \left\{ 1 - i \frac{2V}{b\omega} C(\kappa) \right\} e^{i\omega t} d\omega$$

The integral can be partially evaluated using the identity given by Dowell et al.²⁴

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} (i\omega)^n \phi(\omega) e^{i\omega t} d\omega = \frac{d^n \phi}{dt^n} \quad n = 1, 2, \dots$$

Using this result, the inverse transform is reduced to

$$\pi \rho b^2 \left\{ -\frac{d^2 h}{dt^2} \right\} + \frac{1}{2\pi} \int_{-\infty}^{\infty} 2\pi \rho b \omega^2 V h(\omega) \left[\frac{1}{i\omega} C(\kappa) \right] e^{i\omega t} d\omega$$

By performing a similar series of steps with the second term in the $L(\omega)$ expression, the lift force can be expressed in the time domain

$$\begin{aligned} L(t) = & -\pi \rho b^2 \left[\frac{d^2 h}{dt^2} + \frac{1}{2} b \frac{d^2 \alpha}{dt^2} + V \frac{d\alpha}{dt} - b \left(\frac{1}{2} + a \right) \frac{d^2 \alpha}{dt^2} \right] \\ & + \frac{1}{2\pi} \int_{-\infty}^{\infty} 2\pi \rho b V \left[\omega^2 h + b \omega^2 \alpha - i\omega \alpha - \left(\frac{1}{2} + a \right) b \omega \alpha \right] \cdot \\ & \left[\frac{1}{i\omega} C(\kappa) \right] e^{i\omega t} d\omega \end{aligned}$$

However, in order to be able to implement the time integration scheme, the integral portion of the equation must be resolved.

The term

$$\frac{1}{i\omega} C(\kappa)$$

is defined as the result of a Fourier transform applied to the function $\Phi(s)$, which is known as the Wagner function. The term s is a reduced time parameter defined by

$$s = \frac{Vt}{b}$$

Now, if $F(\alpha)$ is the Fourier transform of $f(x)$, and $G(\alpha)$ is the Fourier transform of $g(x)$, the convolution theorem gives

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} F(\alpha) \cdot G(\alpha) \cdot e^{i\alpha x} d\alpha = f * g$$

where the operator $*$ denotes the convolution of f and g . Using this expression, and the definition of the Wagner function, the integral portion of the lift equation can be re-written as

$$2\pi\rho bV\Phi(s) * \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\omega^2 h + \left(\omega^2 - i\frac{\omega V}{b} \right) b\alpha \right] e^{i\omega t} d\omega$$

where the resulting integral term is easily evaluated.

Once this latest integral has been evaluated, the convolution product is usually expressed in terms of a Duhamel integral, given by

$$D(t) = \int_0^t \Phi\left[\frac{V(t-\tau)}{b}\right] \cdot \left[\frac{\partial^2 h}{\partial \tau^2} + b \frac{\partial^2 \alpha}{\partial \tau^2} + V \frac{\partial \alpha}{\partial \tau} \right] d\tau$$

When this notation is used, the resulting expression for the lift force in the time domain becomes

$$L(t) = -\pi\rho b^2 \left[\frac{d^2 h}{dt^2} + \frac{1}{2} b \frac{d^2 \alpha}{dt^2} + V \frac{d\alpha}{dt} \right] - 2\pi\rho bV \cdot D(t)$$

The aerodynamic force and moment terms can now be included in the remaining portions of the equations of motion. Appendix C lists the full equations of motion for a two dimensional wing with four degrees of freedom.

5.1 EVALUATING THE DUHAMEL INTEGRAL

One of the problems in the implementation of the time integration scheme is that the Duhamel integral must be evaluated at each time step. As part of their work, Rodden and Stahl²⁵ proposed a simple means of performing this evaluation.

The basis for this method lies in the manner in which the Wagner function $\Phi(s)$ is defined. No explicit expression for the function exists, and the function can only be determined numerically. The standard practice is to use an analytical approximation to the function, and one of the most common forms of the approximate function is an exponential expansion

$$\Phi(s) = 1 - \sum \alpha_n \cdot \exp(-\beta_n s)$$

where α_n and β_n are constants.

Rodden and Stahl themselves used a two term expression where the constants have the following values

$$\begin{aligned}\alpha_1 &= 0.165 \\ \alpha_2 &= 0.335 \\ \beta_1 &= 0.041 \\ \beta_2 &= 0.320\end{aligned}$$

These values were used in the present implementation of the numerical integration scheme.

Following the work of Rodden and Stahl, one further definition is required in order to simplify the analysis. The Duhamel integral is now re-written as

$$D(t) = \int_0^t \Phi\left[\frac{V(t-\tau)}{b}\right] Q'(\tau) d\tau$$

where the function $Q'(\tau)$ contains the differential terms. Using this definition, the Duhamel integral is approximated by

$$D(t) = Q(t) - \sum \alpha_n \cdot B_n(t)$$

In this expression, the functions $B_n(t)$ measure the lag in the induced aerodynamic loads in following the motion, and are defined by

$$B_n(t) = \exp\left(-\frac{\beta_n V t}{b}\right) \int_0^t Q'(\tau) \exp\left(\frac{\beta_n V \tau}{b}\right) d\tau$$

If both sides of the equation are differentiated with respect to time, the above equation can be expressed in differential form

$$\frac{dB_n}{dt} + \left(\frac{\beta_n V}{b}\right) B_n = Q'(t)$$

This equation will add n equations to the system of equations which must be integrated, but greatly simplifies the evaluation of the Duhamel integral.

5.2 REARRANGING THE FLUTTER EQUATIONS OF MOTION

Before the time integration procedure can be implemented, the flutter equations must be cast in a form suitable for integration. As first order differential equations are the easiest to integrate numerically, it was decided to express the system equations of motion as a system of first order differential equations. In order to do this, the equations are expressed in state variable form in the normal manner. By using the form of the Duhamel integral given by Rodden and Stahl²⁵, and including the differential terms in the main portion of the equations, the flutter system equations can be expressed in the following general form

$$[A] \left\{ \frac{d\phi}{dt} \right\} = [B] \{\phi\}$$

Although this equation can be used in the time integration procedure, the formulation will be inefficient in the case of nonlinear systems. This is due to the fact that the $[B]$ matrix will contain both linear and nonlinear terms. To solve this problem, the nonlinear terms in the $[B]$ matrix which in this case represent the restoring forces in each degree of freedom are removed, and the forces are expressed as a vector $\{F(\phi)\}$. The flutter equations now take the form

$$[A] \left\{ \frac{d\phi}{dt} \right\} = [B] \{\phi\} + \{F(\phi)\}$$

where $\{F(\phi)\}$ is the vector containing the structural restoring force terms. The $[A]$ and $[B]$ matrices are now time independent, and can be evaluated before the integration loop begins. The elements of the $[A]$ and $[B]$ matrices, as well as the force vector $\{F\}$ are listed in Appendix C. Note that there are $2n+2$ equations describing a system with n degrees of freedom. The extra two equations result from the use of the two term approximation to the Wagner function.

By inverting the matrix $[A]$ and multiplying the rest of the system by the inverse, the vector of differential terms may be given explicitly

$$\left\{ \frac{d\phi}{dt} \right\} = [A]^{-1} [B] \{\phi\} + [A]^{-1} \{F\}$$

and is now cast in a suitable form for use in a numerical time integration scheme. Note that in the case of large systems, the products $[A]^{-1} [B]$ and $[A]^{-1} \{F\}$ would be found by matrix decomposition methods to maximize the computational efficiency.

5.3 CHOICE OF TIME INTEGRATION METHOD

The paper by Laurenson et al.¹⁸, reports on their evaluation of a number of integration methods. These routines were used to analyze nonlinear flutter problems similar to the ones being considered in this thesis. The major conclusion of the work

was that the use of an unsophisticated integration routine may adversely affect the accuracy of the analysis. The authors cite the example of a trapezoidal method, which was unable to accurately predict the flutter of systems when preload nonlinearities were present in two or more degrees of freedom.

Laurenson et al. recommend three time integration methods. The method selected for use in this study is an Adams-Moulton predictor-corrector formulation, as given by Boyce and Diprima²⁶. The predictor formula is given by

$$x_{n+1} = x_n + \frac{\Delta t}{24} \left[55 \left(\frac{dx}{dt} \right)_n - 59 \left(\frac{dx}{dt} \right)_{n-1} + 37 \left(\frac{dx}{dt} \right)_{n-2} - 9 \left(\frac{dx}{dt} \right)_{n-3} \right]$$

and the estimate of the function at the next time step is corrected using the formula

$$x_{n+1} = x_n + \frac{\Delta t}{24} \left[9 \left(\frac{dx}{dt} \right)_{n+1} + 19 \left(\frac{dx}{dt} \right)_n - 5 \left(\frac{dx}{dt} \right)_{n-1} + \left(\frac{dx}{dt} \right)_{n-2} \right]$$

where n refers to the present time step, and $n+1$ is the next time step.

The predictor formula is derived by fitting a polynomial function of order p (in this case $p = 3$) which describes the derivative $x'(t)$ to the $p+1$ points at times $t_{n-p}, \dots, t_n$. This polynomial function is then integrated over the interval t_n to t_{n+1} , which results in the expression for x_{n+1} . The corrector formula is derived in a similar manner, except that the interpolation polynomial is fitted to the points at times $t_{n+1-p}, \dots, t_{n+1}$.

In their present formulation, these equations are not intended to be self-starting, as they depend upon the time histories of the derivatives of the system.

Boyce and Dprima recommend the use of another method, such as the trapezoidal or Runge-Kutta procedure to obtain the required derivative time histories. As the introduction of a second time integration routine into the software was considered an unnecessary complication, the Adams-Moulton method was used in self-starting form, with no apparent loss of accuracy. To start the simulation, the derivatives of the various degrees of freedom were set equal to zero, representing a system which has remained at rest for some time prior to the start of the simulation.

5.4 TIME INTEGRATION SOFTWARE

Appendix D contains the FORTRAN computer program written to implement the numerical integration scheme. Figure 5-1 shows the information flow paths through the program. The software design incorporated the aerodynamic coefficient calculation routines previously written for use in the iterative procedure software. Certain of the existing data entry routines were also placed directly into the new code. In this manner, the number of new subroutines that were required by the program was kept to a minimum.

FIGURE 5-1: INFORMATION FLOW THROUGH TIME INTEGRATION PROGRAM

5.4.1 System Definition Section

The first section of the time integration software concerns the specification of the wing system that is to be modelled. The subroutine WNGSPC obtains the inertial and geometric properties of the airfoil from a user prepared data file. This data file is the file RUN.DAT, which is also used by the iterative procedure. It was recognized that this mode of data entry restricted the ease with which the system parameters could be changed to begin another test, however, two benefits were perceived. By using a data file to store the system parameters, the time necessary to repeat tests of the same system with different initial conditions is reduced. The second benefit lies in the fact that both the time integration and iterative procedures use a data file with identical format. This permits the two programs to be run consecutively, once a data file has been created for one system.

The next routine used by the program is SPRING. This routine loads the specifications of the modal spring stiffness values, and was taken directly from the code written for the iterative procedure. This direct adaptation had the effect of introducing otherwise unused variables into the code. This minor problem is offset by the advantage gained from maintaining the commonality with the iterative procedure software.

The third subroutine in this section of the code is INITIAL. This routine

obtains the initial conditions of the system through an interactive session with the program user. Along with the initial modal displacements and velocities, the user also enters values of the air velocity and the time step size. The interactive nature of this data entry routine permits maximum flexibility in examining the effects of the initial conditions or velocity upon the system flutter response.

5.4.2 Aerodynamic Coefficient Calculation

If the software is intended to analyze the flutter of systems involving three or more degrees of freedom, the next stage in the program is to calculate the necessary aerodynamic coefficients. Subroutine PSICLC calls the required routines to calculate Kussner's Φ functions as they apply to aileron and tab degrees of freedom. If control tabs are present in the system, the routine then calculates the appropriate values of Kussner's X functions. All of the software used in this section of the program was taken directly from the iterative procedure program. This ensures that both the time integration and iterative routines use identical aerodynamic theory.

5.4.3 Matrix Calculation

Using the inertial properties of the system, and the aerodynamic terms

calculated by the program, the two matrices $[A]$ and $[B]$ which define the system of differential equations are filled. The elements of these two matrices are listed in Appendix C. Once the elements of the $[A]$ matrix have been defined, the matrix is inverted using a Gauss-Jordan elimination routine (RLINVRT). This inverted matrix is then multiplied by the $[B]$ matrix. Performing this multiplication before the time integration loop is started reduces the time necessary to run the software.

5.4.4 Time Integration Loop

The major portion of the time integration software is the loop which actually contains the integration predictor-corrector formulae. This loop continues for a number of steps equal to the value of the simulation duration divided by the magnitude of the time step.

The first process in the loop is the use of the subroutine FLOAD to calculate the elements of the force vector $\{F\}$. The routine uses the definition of the modal stiffness values to calculate the exact force in a linear or bilinear spring, based upon the system deflections at the present time step. The force vector is then multiplied by the inverse of the $[A]$ matrix, and the program evaluates the system derivatives at the present time step.

Using these derivatives, as well as derivatives for previous time steps, the

predictor equation is used to estimate the value of each system variable at the next time step. By calling the subroutine FLOAD and repeating the above process, a value of the derivative at the next time step is found. This value is used by the corrector formula to revise the estimate of the system variables for the next time step.

At this point, the time, variable and derivative values are written to an output data file. The variable and derivative storage arrays are then reset, ready to begin the next loop. The use of the output file allows routines to be written to manipulate the results of the time integration procedure to obtain the desired presentation form.

Chapter 6

VALIDATION OF SIMULATION PROCEDURES

The flutter of two dimensional wing systems has been extensively studied. The published results of these studies form the basis of the process that was used to validate both the time integration and iterative methods of flutter speed prediction. Several of the cases cited in the literature were selected as test cases, and the two procedures were used to predict the onset of flutter. The predictions were then compared with the published flutter characteristics. By examining the level of agreement between the predicted and published flutter speeds, an assessment of the accuracy of the two methods could be made.

The flutter velocities could be obtained directly through the use of the iterative program. However, the time integration method used the air velocity as an input parameter. For this reason, when this method was used, a value of the air velocity was selected, and the flutter response of the system was obtained using the time integration scheme. A computer program was used to calculate the amount of damping present in each mode, using the formula

$$\zeta = \frac{1}{2\pi n} \ln \left(\frac{A_0}{A} \right)$$

where the initial amplitude is A_0 , and n is the number of cycles between the cycles

with amplitudes A and A_0 . The point of instability was located as being the point where the value of the damping parameter ζ becomes equal to zero. By selecting appropriate velocity values and repeating the procedure, it would then be possible to locate the point of instability using the modal damping values.

6.0 LINEAR SYSTEMS WITH TWO DEGREES OF FREEDOM

6.0.1 Iterative Procedure

Figures 6-1 through 6-9 show the flutter speed predictions made using the iterative procedure along with the corresponding results reported by Theodorsen and Garrick¹⁰. The flutter velocities obtained were plotted as a flutter parameter, which was expressed as a function of the ratio of the modal velocities. The flutter parameter is the nondimensional form of the flutter velocity, and is given by

$$FP = \frac{V_F}{b\omega_\alpha}$$

where ω_α is the frequency of the torsional mode. Figures 6-1 through 6-7 illustrate the flutter boundary curves as functions of frequency ratio and static unbalance. Figures 6-8 and 6-9 present the system flutter boundaries as functions of reduced

mass parameter and static unbalance. Table 6-1 lists the parameters which define the system corresponding to each figure. In the majority of these cases, the iterative procedure predictions were able to closely approximate the published curves.

In Figure 6-1, the prediction of the flutter boundary for the system with a static unbalance value of 0.0 produces a curve which differs from that given in Reference 10. In this case, an additional flutter prediction technique was employed. These calculations used the flutter equations quoted by Theodorsen and Garrick to obtain the flutter speeds at several values of frequency ratio. The results obtained are plotted in Figure 6-1, and can be seen to agree with the iterative procedure prediction.

Figure 6-6 reveals two cases where the iterative predictions of the flutter boundaries differ from the published curves. In these cases, calculations were again performed using the flutter equations given by Theodorsen and Garrick for several values of frequency ratio. As with the curve in Figure 6-1, the flutter velocities predicted using these calculations agreed with the predictions of the iterative procedure.

The iterative procedure was also used to predict the flutter velocities of linear systems reported by Shien¹² and McIntosh et al.^{14,15}. Table 6-2 lists the system parameters used, as well as the comparisons of the flutter velocities, frequencies and reduced frequencies for these systems. As can be seen from the table, the iterative procedure was able to predict the flutter speed of each of these systems with a

maximum error of under 1.5 percent.

6.0.2 Time Integration Procedure

To validate the time integration procedure, several test cases were taken from the results quoted by Theodorsen and Garrick¹⁰. The flutter speed predictions made using the time integration method were then compared with the iterative procedure results for those cases. Table 6-3 lists the system parameters used in the test cases, as well as the results of the comparisons. The time integration procedure was able to predict the flutter velocities of these systems, with a maximum deviation of less than 4 percent from the iterative procedure predictions.

One of the advantages of the time integration solution of the flutter equations is the ability to predict the point of divergence for the wing. Divergence is a static instability, whose most common form is the torsional divergence of the wing. Figure 6-10 shows the response of the pitching mode of a system at a velocity just greater than the divergence speed. The figure shows the oscillatory response due to dynamic behaviour superimposed upon a curve that diverges from the zero amplitude value. The diverging system prevents the use of the modal damping calculation routine, thereby making the identification of the flutter point more difficult using the time integration method.

6.1 LINEAR SYSTEMS WITH THREE DEGREES OF FREEDOM

6.1.1 Iterative Procedure

Figures 6-11 through 6-23 compare the flutter boundary predictions made using the iterative procedure with the corresponding curves quoted by Theodorsen and Garrick.¹¹ Table 6-4 lists the system parameters which correspond to each figure. The figures illustrate that the agreement between the predicted and published values of flutter velocity is quite good in the majority of the cases. There are several gaps in the curves plotted using the predicted results. These are attributable to systems whose reduced frequency at instability is greater than the maximum value used in the iterative software.

Figure 6-17 shows the flutter boundaries of a system where a second region of instability is present at low values of frequency ratio. The iterative procedure was able to closely approximate the upper flutter boundary; the method also predicted that the lower region of instability would occur for a larger range of frequency ratios. The equations reported by Theodorsen and Garrick¹⁰ were used to predict the flutter speeds of this system with a frequency ratio of 0.5. The results of these calculations were plotted in Figure 6-17, and clearly support the results obtained using the iterative prediction method.

Figure 6-20 shows the flutter boundary of a system with a large secondary region of instability. Reference 11 reports that the upper flutter boundary extends over the entire range of frequency ratios. However, the iterative procedure predicted that this boundary was actually an extension of a portion of the secondary flutter boundary. The iterative procedure was able to closely approximate the secondary flutter boundary reported by Theodorsen and Garrick.¹¹ The iterative procedure also predicted the existence of a second flutter mode, present at low values of frequency ratio, which is not reported by Theodorsen and Garrick. Figure 6-24 shows the V-g curve obtained for the system with a frequency ratio of 0.4. This figure clearly illustrates the fact that a second mode becomes unstable over a limited range of velocities.

The time integration procedure was used to examine the flutter response of the system with a frequency ratio of 0.4 at values of the flutter parameter of 0.3 and 0.7. These points correspond to conditions where the flutter of the system is in the form of only one mode. Figure 6-25 graphically illustrates results obtained using the time integration procedure at the first value of the flutter parameter, and shows a representative cycle of the system response. The mode consists of large aileron deflections coupled with moderate plunge deflections. The pitching motion of the airfoil involved in this mode is minimal. This response can be contrasted with the motion of the system at the higher velocity. Figure 6-26 shows this mode to consist of a pronounced pitching motion coupled with moderate aileron motion. In this case,

it is the plunge motion which is negligible.

One further demonstration of the capabilities of the iterative procedure is illustrated in Figure 6-27. The figure shows the flutter boundaries for a system with varying amounts of structural damping present in the pitch degree of freedom. The iterative procedure was able to closely approximate the curves. The iterative method also predicted the existence of another flutter mode, which is most likely the aileron-plunge mode, as the flutter velocities of this mode are not affected by the increase in torsional structural damping.

6.1.2 Time Integration Procedure

A number of complications can arise when the time integration procedure is used to predict the flutter speeds of linear wing systems with three degrees of freedom. One problem reduces the accuracy of the method, and results from the complex motions which can be present in the system responses. These motions are caused by the interaction of the three modes. Figure 6-28 illustrates the aileron motion for a system at velocity below the nominal flutter point. The complex response of the system obviates the use of the computer routine to calculate the modal damping. In these cases, the system response is generated over a longer time period, as in Figure 6-29, and the system response is visually inspected to determine

the onset of instability.

The second problem concerns the length of time required in order to obtain a flutter speed prediction for a three degree of freedom system. Transient motions can be introduced into the system by the initial conditions, and therefore, the system response may require a long period of time before a determination of the stability of the system can be made.

Table 6-5 lists the flutter velocity predictions made using the time integration procedure for certain cases selected from Reference 11. These cases were also used to validate the iterative procedure, and the table lists the appropriate figure numbers. The flutter velocities were compared to the iterative predictions, and the table shows that the average difference between the two sets of results was approximately 5 percent.

6.2 NONLINEAR SYSTEMS WITH TWO DEGREES OF FREEDOM

To test the accuracy of the procedures in predicting the limit cycle response of two degree of freedom systems with nonlinear elements, two cases quoted by Shen¹² were used as a reference. The system given consists of a wing with a linear spring restraining the plunge motion, and a bilinear spring with freeplay and preload

present in the pitch degree of freedom.

6.2.1 Iterative Procedure

Figure 6-30 plots the flutter boundaries for the system when the preload is set to a value of 0.2 degrees. The figure illustrates that the flutter boundary predicted using the iterative procedure differs from the curve reported by Shen¹². The greatest difference occurs at an oscillatory amplitude of approximately 0.75 degrees. In this region, the flutter velocities predicted using the iterative method can be up to 25 percent lower than the values reported by Shen. This difference occurs despite the fact that both Shen and the iterative procedure used a single term describing function to linearize the flutter system.

The source of this difference in flutter velocities can be ascribed to errors which are present in the describing function used by Shen. The first error lies in the fact that the describing function as reported by Shen contains a discontinuity. The second error arises from the assumptions that were made in the derivation of Shen's describing functions. The limit cycle motion was assumed to consist entirely of an oscillatory component, and the presence of a steady state element of the motion was ignored. The difference that this assumption produces is evident from Figure 6-32, where the describing function derived using Shen's method is compared to the one

taken from Lee and Tron²⁶, which accounts for the presence of steady state components in the limit cycle motion.

Figure 6-31 shows the flutter boundary for the same system, with the spring preload value set to zero. The iterative procedure predictions closely approximate the curves given by Shen, but extend the flutter boundary to a velocity ratio of approximately 0.25. This is due to the difference in describing functions, illustrated in Figure 6-33.

6.2.2 Time Integration Procedure

The results for the nonlinear systems obtained using the time integration scheme are also plotted in Figures 6-30 and 6-31. As these figures show, the time integration results closely agree with the results obtained using the iterative method. These curves also reveal a difference in the solutions of the iterative procedure and the time integration scheme; the fact that the time integration scheme can only predict the stable branches of the flutter solution. The stability of a branch of the solution is determined by the response of the system to a small perturbation away from the branch. Consider the upper portion of the curve shown in Figure 6-30. An increase in the amplitude of the motion from the flutter condition will cause the local velocity to be lower than the flutter velocity required to sustain the motion, and the

amplitude will decrease to the original point. Similarly, a decrease in amplitude will lead the system into the unstable regime, and the response will increase to the previous value. In contrast, a perturbation from the unstable branch will continue to move away from the branch.

6.3 NONLINEAR SYSTEMS WITH THREE DEGREES OF FREEDOM

6.3.1 Iterative Procedure

Figures 6-34 and 6-35 show the flutter boundaries for two three degree of freedom systems reported by Shen¹². Figure 6-34 represents the response of a system with a nonlinear spring in the aileron degree of freedom whose preload and freeplay magnitudes are equal. Figure 6-35 represents the same system with the preload value halved.

The iterative procedure was able to closely approximate the lower portions of each curve (the stable branches), but differed from the reported curves on the unstable portions of the solution. These differences are presumably attributable to the differences in the describing functions used.

6.3.2 Time Integration Procedure

The results for the three degree of freedom nonlinear system as obtained using the time integration scheme are also plotted in Figures 6-34 and 6-35. As before, the results obtained were in close agreement with the predictions of the iterative method, although only the stable branches could be detected. However, the use of the time integration scheme was restricted somewhat by the fact that the three degree of freedom system required long simulation durations to reach the limit cycle condition. For this reason, simulation runs of up to 50 seconds were required to examine the behaviour of the systems.

6.4 CONCLUSIONS

Based upon the results of the validation process, a number of conclusions can be made about the use of the iterative and time integration procedures to predict the onset of flutter in a wing system.

- 1/. The iterative procedure has been shown to be a rapid and accurate method of determining the flutter speeds of both linear and nonlinear

airfoil systems.

- 2/. For the nonlinear cases that were tested, the agreement between the flutter speeds obtained using the iterative procedure and the 'exact' solutions found using time integration suggests that the one term describing function linearization technique is an acceptably accurate method of modelling nonlinear systems.
- 3/. The agreement of time integration and iterative predictions for linear systems implies that the integration procedure may be used to generate flutter results which can then be used as test cases for the iterative method. This would be useful in situations where no published data exist.
- 4/. The time integration procedure is considerably slower than the iterative method in calculating flutter speeds. This is a result of the methods which must be employed to obtain the flutter speed, as well as the nature of the system responses themselves.

FIGURE 6-1: FLUTTER PARAMETER AS A FUNCTION OF
 PLUNGE-PITCH FREQUENCY RATIO

2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

FIGURE 6-2: FLUTTER PARAMETER AS A FUNCTION OF
PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-3: FLUTTER PARAMETER AS A FUNCTION OF
PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-4: FLUTTER PARAMETER AS A FUNCTION OF PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-5: FLUTTER PARAMETER AS A FUNCTION OF
PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-6: FLUTTER PARAMETER AS A FUNCTION OF PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-7: FLUTTER PARAMETER AS A FUNCTION OF PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-8: FLUTTER PARAMETER AS A FUNCTION OF REDUCED MASS PARAMETER

FIGURE 6-9: FLUTTER PARAMETER AS A FUNCTION OF REDUCED MASS PARAMETER

FIGURE 6-10: SYSTEM MODAL RESPONSE SHOWING
TORSIONAL DIVERGENCE

FIGURE 6-11: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO

**FIGURE 6-12: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO**

FIGURE 6-13: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO

FIGURE 6-14: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO

**FIGURE 6-15: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO**

FIGURE 6-16: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO

FIGURE 6-17: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-FITCH FREQUENCY RATIO

FIGURE 6-18: FLUTTER PARAMETER AS A FUNCTION OF
PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-19: FLUTTER PARAMETER AS A FUNCTION OF PLUNGE-PITCH FREQUENCY RATIO

FIGURE 6-20: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO

**FIGURE 6-21: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO**

FIGURE 6-22: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO

**FIGURE 6-23: FLUTTER PARAMETER AS A FUNCTION OF
AILERON-PITCH FREQUENCY RATIO**

**FIGURE 6-24: V-q CURVE FOR FIGURE 6-20 WITH
FREQUENCY RATIO OF 0.4**

**FIGURE 6-25: SYSTEM RESPONSE TO SECOND FLUTTER
MODE IN FIGURE 6-20**

**FIGURE 6-26: SYSTEM RESPONSE TO PRIMARY FLUTTER
MODE IN FIGURE 6-20**

FIGURE 6-27: FLUTTER CHARACTERISTICS OF SYSTEM INCLUDING EFFECTS OF STRUCTURAL DAMPING

FIGURE 6-28: MODAL RESPONSE OF THREE DEGREE OF FREEDOM SYSTEM OVER SHORT TIME PERIOD

FIGURE 6-29: MODAL RESPONSE OF THREE DEGREE OF FREEDOM
SYSTEM OVER LONG TIME PERIOD

FIGURE 6-30: PITCH LIMIT CYCLE AMPLITUDE AS A
FUNCTION OF VELOCITY RATIO $\delta_p = .2$ $\delta_f = .5$

**FIGURE 6-31: PITCH LIMIT CYCLE AMPLITUDE AS A
FUNCTION OF VELOCITY RATIO $\delta_p=0$ $\delta_r=.5$**

FIGURE 6-32: COMPARISON OF DESCRIBING FUNCTIONS
FOR CASE WITH $\delta_p = .2$ AND $\delta_r = .5$

FIGURE 6-33: COMPARISON OF DESCRIBING FUNCTIONS
FOR CASE WITH $\delta_p = .0$ AND $\delta_r = .5$

FIGURE 6-34: AILERON LIMIT CYCLE AMPLITUDE RATIO
AS A FUNCTION OF VELOCITY RATIO $\delta_p = \delta_r = .2$

FIGURE 6-35: AILERON LIMIT CYCLE AMPLITUDE RATIO
AS A FUNCTION OF VELOCITY RATIO $\delta_p = 1.1$ $\delta_1 = 0.2$

Chapter 7

RECOMMENDATIONS FOR FUTURE WORK

The work done in this study can provide a useful base for further research into the area of wing and aircraft flutter.

One project which could be undertaken without requiring changes in the software is a parametric study of flutter in a two dimensional wing with four degrees of freedom. The aerodynamic coefficients needed to implement the four degree of freedom solution are already present in both computer programs. The task would require the systematic evaluation of each of the applicable parameters and their effects upon the flutter characteristics of the wing. As part of the work, the time integration software could be used to ensure the accuracy of the results obtained using the iterative solution procedure.

Another possible project would support the solutions obtained using the simulation techniques. One of the major limitations of the work done in studying the flutter characteristics of wings is the lack of experimental data which can be used to confirm the solutions. Although McIntosh et al.^{14,15} attempted to correct this oversight, there is still great potential for experimental studies of the flutter problem. The construction of a representative wing section with n degrees of freedom would serve three purposes. Firstly, it would provide a useful store of results which could

be used to verify the accuracy of solution techniques. Secondly, the tests of a scale model in a wind tunnel would allow the aerodynamic theory used in the simulations to be improved, as modifying factors could be found and applied to the aerodynamic coefficients. In this manner, the simulation procedures would be more closely aligned with realistic systems, improving the capabilities of those procedures. The final benefit that the experimental work would confer would be the aid to the understanding of the nature of flutter, provided through the use of a physical demonstration.

A third topic which could be considered for future work is the use of the iterative procedure to examine flutter in a three dimensional wing system. The software already contains many of the elements necessary to allow such an analysis, with one exception. In order to use the iterative technique on three dimensional systems, the procedures to calculate the stiffness values will have to be re-written, to conform more closely with Lee's¹⁷ suggested procedure. The ability to simulate the flutter characteristics of a planform wing would represent a significant increase in the capabilities of the program.

One of the requirements that must be filled before the solution of three dimensional systems can take place is the necessity for improved mode shape and stiffness data for the wing. Therefore, the work suggested above could be undertaken as part of a larger project whose goal is the integration of the of the iterative software into an existing FEA program. In this manner, the geometry and properties of the wing would be specified, and passed to the FEA program. The system stiffness

and mode shape values could then be obtained, and the flutter program started. When the flutter solution has been obtained, the sophisticated graphics capabilities present in many computer workstations could be used to animate the resulting motion in the system, thereby improving understanding of the phenomena involved. In order to achieve the minimum stage of this concept, software will have to be written to provide a link between the two existing programs. As well, a study would have to be undertaken to examine the aspects of modelling a wing structure using FEA techniques.

Chapter 8

CONCLUSIONS

Based upon the work presented in this study, a number of conclusions may be drawn concerning the nature and study of flutter in nonlinear systems.

- 1/. The presence of nonlinear elements in airfoil systems requires the development of alternate methods to predict the flutter response. These methods typically involve linearizing the equations of motion by representing the nonlinear element in terms of a Fourier Series expansion. One such method, where only the fundamental term of the Fourier Series is retained, has been shown to be an accurate method of modelling nonlinearities.
- 2/. The flutter response in nonlinear systems can be in the form of a periodic motion, known as a limit cycle, whose amplitude may depend upon the initial conditions of the system. While the amplitude of the limit cycle motion may be small, the airfoil system may still be damaged through fatigue. The limit cycle flutter response can occur at conditions within the flutter envelope predicted for a linear system.

- 3/. A method proposed by Lee¹⁷ has been adapted to the study of flutter in a two dimensional wing section. The method examined the flutter response of both two and three degree of freedom systems involving linear and nonlinear structural elements. This method made use of describing functions and an iterative solution process, and was found to be a rapid means of locating the onset of flutter in a two dimensional wing.
- 4/. The flutter envelope for the two dimensional wing was also determined using a time integration procedure. The formulation of the equations of motion made use of the simplification to the Duhamel integral given by Rodden and Stahl²⁵. This method was as accurate as the iterative procedure, but required significantly more computational time to reach a solution. This method would be most suitably used to verify selected test cases to ensure the accuracy of the iterative procedure.
- 5/. The unsteady aerodynamic theory used in the analysis may be of limited use in certain practical situations. The accuracy of the theory in predicting the aerodynamic loads in these cases may be improved if the coefficients are adjusted to coincide with results obtained through experimental studies.
- 6/. The software developed to implement the iterative solution procedure may be

adapted to the prediction of flutter in three dimensional wing systems. This adaptation would involve the revision of the stiffness calculation routines to correspond with those suggested by Lee¹⁷. The adaptation to the solution of flutter in three dimensional wings could be coupled with the development of a FEA model of a wing system.

REFERENCES

- 1 Albano, E. and Rodden, W.P.
"A Doublet-Lattice Method for Calculating Lift Distributions on Oscillating Surfaces in Subsonic Flows"
AIAA Journal Vol. 7
pp 279-285
- 2 Smilg, B. and Wasserman, L.S.
"Application of Three Dimensional Flutter Theory to Aircraft Structure"
Air Corps Technical Report 4798
July, 1942
- 3 Breitbach, E.J.
"Effects of Structural Non-Linearities on Aircraft Vibration and Flutter"
AGARD Report 665
September 1977
- 4 Nayfeh, A.H. and Mook, D.T.
Non-Linear Oscillations
John Wiley and Sons
1979
- 5 Thompson, J.M. and Stewart, H.B.
Nonlinear Dynamics and Chaos
John Wiley and Sons
1986
- 6 Kryloff, N. and Bogoliuboff, N.
Introduction to Non-Linear Mechanics
Princeton University Press
1947
- 7 Bogoliuboff, N.N. and Mitropolsky, Y.A.
Asymptotic Methods in the Theory of Non-Linear Oscillations
Hindustan Publishing Corp.
1961

- 8 Woolston, D.S., Runyan, H.L. and Byrdson, T.A.
 "Some Effects of System Nonlinearities in the Problem of Aircraft Flutter"
 NACA Technical Note 3539
 October 1955

- 9 Woolston, D.S., Runyan, H.L. and Andrews, R.E.
 "An Investigation of Effects of Certain Types of Structural Nonlinearities on
 Wing and Control Surface Flutter"
Journal of the Aeronautical Sciences
 Vol. 24, January 1957
 pp 57-63

- 10 Theodorsen T. and Garrick, I.E.
 "Mechanism of Flutter: A Theoretical and Experimental Investigation of the
 Flutter Problem"
 NACA Report 685
 1940

- 11 Theodorsen, T. and Garrick, I.E.
 "Flutter Calculations in Three Degrees of Freedom"
 NACA Technical Report 741
 1942

- 12 Shen, S.F.
 "An Approximate Analysis of Nonlinear Flutter Problems"
Journal of the Aerospace Sciences
 Vol. 26, January 1959
 pp 25-32,45

- 13 Laurenson, R.M. and Trn, R.M.
 "Flutter of Control Surfaces with Structural Non-Linearities"
AIAA Journal
 Vol. 18, No. 10
 pp 1245-1251

- 14 McIntosh, S.C., Reed, R.E. and Rodden, W.P.
 "An Experimental and Theoretical Study of Nonlinear Flutter"
 AIAA Paper 80-0791

- 15 McIntosh, S.C., Reed, R.E. and Rodden, W.P.
"Experimental and Theoretical Study of Nonlinear Flutter"
Journal of Aircraft
Vol. 18, December 1981
pp 1057-1063
- 16 Breitbach, E.J.
"Flutter Analysis of an Airplane with Multiple Structural Non-Linearities in the Control System"
NASA Technical Paper 1620
January 1980
- 17 Lee, C.L.
"An Iterative Procedure for Nonlinear Flutter Analysis"
AIAA Paper 85-0688
- 18 Laurenson, R.M., Hauenstein, A.J., Gubser, J.L. and Briley, R.P.
"Effect of Structural Nonlinearities on Limit Cycle Response of Aerodynamic Surfaces"
AIAA Paper 86-0899
- 19 Pines, S.
"An Elementary Explanation of the Flutter Mechanism"
Proceedings of the National Specialists Meeting on Dynamics and Aeroelasticity
pp 52-58
- 20 Theodorsen, T.
"General Theory of Aerodynamic Instability and the Mechanism of Flutter"
NACA Report 496
1935
- 21 Kussner, H.G. and Schwarz, I.
"The Oscillating Wing with Aerodynamically Balanced Elevator"
NACA Technical Memorandum 991
1941
- 22 Bisplinghoff, R.L.
Aeroelasticity
Addison-Wesley
1955

- 23 Luke, Y.L. and Dengler, M.A.
"Tables of the Theodorsen Circulation Function for Generalized Motion"
Journal of the Aeronautical Sciences
Vol. 18, July 1951
pp 478-483
- 24 Dowell, E.H., Curtiss, H.C., Scanlan, R.H. and Sisto, F.
A Modern Course in Aeroelasticity
Sijthoff and Noordhoff
1978
- 25 Rodden, W.P. and Stahl, B.
"A Strip Method for Prediction of Damping in Subsonic Wind Tunnel and
Flight Flutter Tests"
Journal of Aircraft
Vol. 6, January-February 1969
pp 9-17
- 26 Boyce, W.E. and DiPrima, R.C.
Elementary Differential Equations and Boundary Value Problems
John Wiley and Sons
1977
- 27 Lee, B.H.K. and Tron, A.
"Effects of Structural Nonlinearities on Flutter Characteristics of the CF-18
Aircraft"
Unpublished

TABLES

FIGURE	μ	$(r_\alpha)^2$	a	b	x_α	ω_h/ω_α
6-1	3	0.25	-0.2	1.829	0.0 0.1 0.3	var.
6-2	5	0.25	-0.3	1.829	0.0 0.1	var.
6-3	5	0.25	-0.3	1.829	0.05 0.2	var.
6-4	20	0.25	-0.4	1.829	0.0 0.1	var.
6-5	20	0.25	-0.4	1.829	0.3	var.
6-6	10	0.25	-0.45	1.829	0.025 0.04 0.1	var.
6-7	10	0.25	-0.45	1.829	0.05 0.3	var.
6-8	var.	0.25	-0.4	1.829	0.0 0.1 0.2	0.316
6-9	var.	0.25	-0.3	1.829	0.0 0.05 0.2	0.707

Table 6-1: Parameters Describing Systems Used in Two
Degree of Freedom Linear Tests

CASE	McIntosh LINEAR	McIntosh C	McIntosh D	SHEN LINEAR
Mass (kg/m)	11.83	13.24	13.00	13.98
Inertia (kg m)	0.0241	0.0241	0.0241	0.052
Semi-chord (m)	0.1257	0.1257	0.1257	0.152
Density (kg/m ³)	1.225	1.225	1.225	1.225
a	-0.297	-0.297	-0.297	-0.406
x_a	0.0795	0.0710	0.0724	0.042
K_h (N/m ²)	4.788E4	4.788E4	4.788E4	4.857E4
K_a (N/rad)	1.918E2	1.918E2	1.918E2	3.432E2
κ_F Quoted	0.2195	0.1995	0.2043	
κ_F Calculated	0.2137	0.1924	0.1937	0.1830
% Error	2.64	3.56	5.19	
V_F Quoted (m/s)	39.93	42.37	41.76	52.62
V_F Calculated	40.01	42.15	42.32	53.16
% Error	0.20	0.52	1.34	1.02
ω_F Quoted (1/s)	69.74	67.23	67.86	
ω_F Calculated	68.02	64.52	65.21	63.83
% Error	2.46	4.03	3.91	

Table 6-2: Validation of Iterative Calculation Procedure
For Several Two Degree of Freedom Systems

FIGURE	ζ_α	ω_h/ω_α	V_F Iterative	V_F Time Int.	% Diff.
6-1	0.2	0.5	163.03	164.00	0.60
6-1	0.2	1.0	107.16	110.00	2.65
6-2	0.1	0.5	247.74	245.30	0.99
6-2	0.1	1.0	150.91	154.00	2.05
6-2	0.1	0.8	202.02	200.80	0.60
6-3	0.2	0.5	204.91	204.50	0.20
6-3	0.2	1.0	129.82	132.30	1.91
6-4	0.1	1.0	169.22	168.00	0.72
6-6	0.04	0.3	702.84	707.00	0.59
6-6	0.1	0.3	458.20	452.00	1.35
6-6	0.04	0.6	632.20	623.00	1.46
6-6	0.1	0.6	339.54	332.80	1.99
6-7	0.05	0.3	627.84	628.00	0.03
6-7	0.3	0.3	315.03	310.50	1.52
6-7	0.05	0.6	507.26	496.30	2.16

Table 6-3: Validation of Time Integration Procedure
For Several Two Degree of Freedom Systems

Figure	μ	$(r_a)^2$	x_a	x_b	ω_b/ω_a	ω_b/ω_a
6-11	5	1.0	0.2	0.004	0.607	var.
6-12	5	1.0	0.0	-0.002	0.607	var.
6-13	5	1.0	-0.1	0.002	0.607	var.
6-14	5	1.0	0.2	0.002	0.607	var.
6-15	5	0.5	0.2	0.002	0.607	var.
6-16	5	0.5	0.2	-0.005	0.607	var.
6-17	5	0.5	0.2	0.002	1.000	var.
6-18	4	0.5	0.0	0.002	var.	0.50
6-19	4	0.5	0.0	0.002	var.	1.00
6-20	8	0.5	0.2	0.002	0.607	var.
6-21	8	0.5	0.2	0.002	0.316	var.
6-22	12	0.5	0.2	0.000	1.000	var.
6-23	12	0.5	0.0	0.000	1.000	var.

The following values apply to all cases:

$$b = 1.829 \text{ m}, a = -0.2, (r_b)^2 = 0.002, c = e = 0.6$$

Table 6-4: Parameters Describing Systems Used in Three
Degree of Freedom Linear Tests

Figure	$\omega_\beta/\omega_\alpha$	x_α	x_β	V_F Branch	V_F Intgrt	% ERROR
6-16	0.8	0.2	-0.005	91.48	87.00	4.89
6-17	1.5	0.2	0.002	84.68	80.00	5.53
6-18	0.6*	0.0	0.002	44.08	43.00	2.45
6-19	0.5*	0.0	0.002	166.96	158.50	5.07
6-21	1.5	0.2	0.002	270.14	271.50	0.50
6-22	0.5	0.0	0.000	165.39	161.60	2.29

Table 6-5: Validation of the Time Integration Method
Using a Three Degree of Freedom Wing System

* These frequency ratios were actually ω_h/ω_α

APPENDIX A
AERODYNAMIC LOADS AND COEFFICIENTS

The motion of the airfoil system is represented in terms of the following six component motions

α - Rotation of the wing about the elastic axis

h - Vertical translation of the wing system

β - Rotation of the aileron about the aileron hinge axis

z - Vertical translation of the aileron

δ - Rotation of the control tab about the tab hinge axis

y - Vertical translation of the control tab

The subscripts O and C refer to the condition at the leading edges of the aileron and the control tab - "OPEN" or "CLOSED"

The lift force acting upon the wing section is given by

$$L = \pi \rho b^3 \omega^2 \left[L_h h^* + \{ L_a - (0.5+a) L_h \} \alpha + \right. \\ \left. \{ L_\beta - (c-e) L_z \} \beta + \{ L_\delta - (d-f) L_y \} \delta \right]$$

The aerodynamic moment acting about the wing elastic axis is given by

$$M = \pi \rho b^4 \omega^2 \left[\{ M_h - (0.5+a) L_h \} h^* + \right. \\ \{ M_a - (0.5+a) M_h - (0.5+a) L_a + (0.5+a)^2 L_h \} \alpha + \\ \{ M_\beta - (c-e) M_z - (0.5+a) L_\beta + (c-e)(0.5+a) L_z \} \beta + \\ \left. \{ M_\delta - (d-f) M_y - (0.5+a) L_\delta + (d-f)(0.5+a) L_y \} \delta \right]$$

where

$$h^* = \frac{h}{b}$$

The aerodynamic moment about the aileron hinge line is given by

$$\begin{aligned}
 T = \pi \rho b^4 \omega^2 \bigg[& \{ T_h - (c-e) P_h \} h^* + \\
 & \{ T_a - (0.5+a) T_h - (c-e) P_a + (0.5+a)(c-e) P_h \} \alpha + \\
 & \{ T_\beta - (c-e) T_z - (c-e) P_\beta + (c-e)^2 P_z \} \beta + \\
 & \{ T_\delta - (d-f) T_y - (c-e) P_\delta + (c-e)(d-f) P_y \} \delta \bigg]
 \end{aligned}$$

The aerodynamic moment acting about the tab hinge is given by

$$\begin{aligned}
 Q = \pi \rho b^4 \omega^2 \bigg[& \{ Q_h - (d-f) R_h \} h^* + \\
 & \{ Q_a - (0.5+a) Q_h - (d-f) R_a + (0.5+a)(d-f) R_h \} \alpha + \\
 & \{ Q_\beta - (c-e) Q_z - (d-f) R_\beta + (d-f)(c-e) R_z \} \beta + \\
 & \{ Q_\delta - (d-f) Q_y - (d-f) R_\delta + (d-f)^2 R_y \} \delta \bigg]
 \end{aligned}$$

$$L_h = 1 - i \frac{2V}{b\omega} C(\kappa)$$

$$L_a = \frac{1}{2} - i \frac{V}{b\omega} \{1 + 2C(\kappa)\} - \frac{2V^2}{b^2\omega^2} C(\kappa)$$

$$L_\beta = \frac{\phi_4}{2\pi} - i \frac{V}{b\omega} \left\{ \frac{\phi_3}{\pi} + \frac{\phi_2}{\pi} C(\kappa) \right\} - \frac{2V}{b^2\omega^2} \frac{\phi_1}{\pi} C(\kappa)$$

$$L_{z_o} = \frac{\phi_3}{\pi} - i \frac{2V}{b\omega} \frac{\phi_1}{\pi} C(\kappa)$$

$$L_{z_c} = \frac{\phi_3}{\pi} - i \frac{V}{b\omega} \frac{\phi_{14}}{\pi} - 2C(\kappa) \left[\frac{V^2}{b^2\omega^2} \frac{\phi_{13}}{\pi} + i \frac{V}{b\omega} \frac{\phi_1}{\pi} \right]$$

$$L_\delta = \frac{\psi_4}{2\pi} - i \frac{V}{b\omega} \left\{ \frac{\psi_3}{\pi} + \frac{\psi_2}{\pi} C(\kappa) \right\} - \frac{2V^2}{b^2\omega^2} \frac{\psi_1}{\pi} C(\kappa)$$

$$L_{y_o} = \frac{\psi_3}{\pi} - i \frac{2V}{b\omega} \frac{\psi_1}{\pi} C(\kappa)$$

$$L_{y_c} = \frac{\psi_3}{\pi} - i \frac{V}{b\omega} \frac{\psi_{14}}{\pi} - 2C(\kappa) \left[\frac{V^2}{b^2\omega^2} \frac{\psi_{13}}{\pi} + i \frac{V}{b\omega} \frac{\psi_1}{\pi} \right]$$

PITCHING MOMENT COEFFICIENTS

$$M_h = 0.5$$

$$M_a = \frac{3}{8} - i \frac{V}{b\omega}$$

$$M_\beta = \frac{\phi_7}{4\pi} - \frac{V^2}{b^2\omega^2} \frac{\phi_5}{\pi} - i \frac{V}{b\omega} \frac{\phi_6}{2\pi}$$

$$M_{z_o} = \frac{\phi_6}{4\pi} - i \frac{V}{b\omega} \frac{\phi_5}{\pi}$$

$$M_{z_c} = \frac{\phi_6}{4\pi} - \frac{V^2}{b^2\omega^2} \frac{\phi_{15}}{\pi} - i \frac{2V}{b\omega} \frac{\phi_5}{\pi}$$

$$M_\delta = \frac{\psi_7}{4\pi} - \frac{V^2}{b^2\omega^2} \frac{\psi_5}{\pi} - i \frac{V}{b\omega} \frac{\psi_6}{2\pi}$$

$$M_{y_o} = \frac{\psi_6}{4\pi} - i \frac{V}{b\omega} \frac{\psi_5}{\pi}$$

$$M_{y_c} = \frac{\psi_6}{4\pi} - \frac{V^2}{b^2\omega^2} \frac{\psi_{15}}{\pi} - i \frac{2V}{b\omega} \frac{\psi_5}{\pi}$$

$$P_h = \frac{\phi_3}{\pi} - i \frac{2V}{b\omega} \frac{\phi_{31}}{\pi} C(\kappa)$$

$$P_a = \frac{\phi_6}{4\pi} - i \frac{V}{b\omega} \frac{\phi_{32}}{\pi} - \frac{\phi_{31}}{\pi} 2C(\kappa) \left\{ \frac{V^2}{b^2\omega^2} + i \frac{V}{b\omega} \right\}$$

$$P_\beta = \frac{\phi_{37}}{2\pi^2} - \frac{V^2}{b^2\omega^2} \frac{\phi_{35}}{\pi^2} - i \frac{V}{b\omega} \frac{\phi_{36}}{\pi^2} - 2C(\kappa) \frac{\phi_{31}}{\pi} \left\{ \frac{V^2}{b^2\omega^2} \frac{\phi_1}{\pi} + i \frac{V}{2b\omega} \frac{\phi_2}{\pi} \right\}$$

$$P_{z_o} = \frac{\phi_{17}}{\kappa^2} - i \frac{V}{b\omega} \frac{\phi_{35}}{\pi^2} - i \frac{2V}{b\omega} \frac{\phi_1 \phi_{31}}{\pi^2} C(\kappa)$$

$$P_{z_c} = \frac{\phi_{17}}{\pi^2} - i \frac{V}{b\omega} \frac{\phi_{16}}{\pi^2} - \frac{V^2}{b^2\omega^2} \left[\frac{2 \ln \tau_{sr}}{\pi^2} + \frac{\phi_{21}}{\pi^2} \right] \\ - 2C(\kappa) \left[\frac{V^2}{b^2\omega^2} \frac{\phi_{13}}{\pi} + i \frac{V}{b\omega} \frac{\phi_1}{\pi} \right] \frac{\phi_{31}}{\pi}$$

$$P_\delta = \frac{X_5(\psi, \phi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left\{ \frac{X_3(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_1(\psi, \phi)}{\pi^2} \right\} \\ - i \frac{V}{b\omega} \left\{ \frac{X_4(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_2(\psi, \phi)}{\pi^2} \right\}$$

$$P_{y_o} = \frac{X_{14}(\psi, \phi)}{\pi^2} - i \frac{V}{b\omega} \left\{ \frac{X_3(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_1(\psi, \phi)}{\pi^2} \right\}$$

$$P_{y_c} = \frac{X_{14}(\psi, \phi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left[2C(\kappa) \frac{X_{11}(\psi, \phi)}{\pi^2} + \frac{X_{12}(\psi, \phi)}{\pi^2} \right] \\ - i \frac{V}{b\omega} \left[\frac{X_{13}(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_1(\psi, \phi)}{\pi^2} \right]$$

$$T_h = \frac{\phi_4}{2\pi} - i \frac{V}{b\omega} \frac{\phi_8}{\pi} C(\kappa)$$

$$T_a = \frac{\phi_7}{4\pi} - \frac{V^2}{b^2\omega^2} \frac{\phi_8}{\pi} C(\kappa) - i \frac{V}{b\omega} \left\{ \frac{\phi_8}{\pi} C(\kappa) + \frac{\phi_9}{2\pi} \right\}$$

$$T_\beta = \frac{\phi_{12}}{4\pi^2} - \frac{V^2}{b^2\omega^2} \left\{ \frac{\phi_{10}}{\pi^2} + \frac{\phi_1\phi_8}{\pi^2} C(\kappa) \right\} - i \frac{V}{2b\omega} \left\{ \frac{\phi_2\phi_8}{\pi^2} C(\kappa) + \frac{\phi_{11}}{\pi^2} \right\}$$

$$T_{z_o} = \frac{\phi_{37}}{2\pi^2} - i \frac{V}{b\omega} \left\{ \frac{\phi_{10}}{\pi^2} + \frac{\phi_1\phi_8}{\pi^2} C(\kappa) \right\}$$

$$T_{z_c} = \frac{\phi_{37}}{2\pi^2} - i \frac{V}{b\omega} \frac{\phi_{19}}{\pi^2} - \frac{V^2}{b^2\omega^2} \frac{\phi_{18}}{\pi^2} - C(\kappa) \left[\frac{V^2}{b^2\omega^2} \frac{\phi_{13}}{\pi} + i \frac{V}{b\omega} \frac{\phi_1}{\pi} \right] \frac{\phi_8}{\pi}$$

$$T_\delta = \frac{X_{10}(\psi, \phi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left\{ \frac{X_8(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_6(\psi, \phi)}{\pi^2} \right\} - i \frac{V}{b\omega} \left\{ \frac{X_9(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_7(\psi, \phi)}{\pi^2} \right\}$$

$$T_{y_o} = \frac{X_{18}(\psi, \phi)}{\pi^2} - i \frac{V}{b\omega} \left\{ \frac{X_8(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_6(\psi, \phi)}{\pi^2} \right\}$$

$$T_{y_c} = \frac{X_{18}(\psi, \phi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left[2C(\kappa) \frac{X_{15}(\psi, \phi)}{\pi^2} + \frac{X_{16}(\psi, \phi)}{\pi^2} \right] - i \frac{V}{b\omega} \left[\frac{X_{17}(\psi, \phi)}{\pi^2} + 2C(\kappa) \frac{X_6(\psi, \phi)}{\pi^2} \right]$$

$$R_h = \frac{\Psi_3}{\pi} - i \frac{2V}{b\omega} \frac{\Psi_{31}}{\pi} C(\kappa)$$

$$R_a = \frac{\Psi_6}{4\pi} - i \frac{V}{b\omega} \frac{\Psi_{32}}{\pi} - \frac{\Psi_{31}}{\pi} 2C(\kappa) \left\{ \frac{V^2}{b^2\omega^2} + i \frac{V}{b\omega} \right\}$$

$$R_b = \frac{\Psi_{37}}{2\pi^2} - \frac{V^2}{b^2\omega^2} \frac{\Psi_{35}}{\pi^2} - i \frac{V}{b\omega} \frac{\Psi_{36}}{\pi^2} - 2C(\kappa) \frac{\Psi_{31}}{\pi} \left\{ \frac{V^2}{b^2\omega^2} \frac{\Psi_1}{\pi} + i \frac{V}{2b\omega} \frac{\Psi_2}{\pi} \right\}$$

$$R_{y_o} = \frac{\Psi_{17}}{\pi^2} - i \frac{V}{b\omega} \frac{\Psi_{35}}{\pi^2} - i \frac{2V}{b\omega} \frac{\Psi_1 \Psi_{31}}{\pi^2} C(\kappa)$$

$$R_{y_c} = \frac{\Psi_{17}}{\pi^2} - i \frac{V}{b\omega} \frac{\Psi_{16}}{\pi^2} - \frac{V^2}{b^2\omega^2} \left[\frac{2 \ln \tau_{sh}}{\pi^2} + \frac{\Psi_{21}}{\pi^2} \right] - 2C(\kappa) \left[\frac{V^2}{b^2\omega^2} \frac{\Psi_{13}}{\pi} + i \frac{V}{b\omega} \frac{\Psi_1}{\pi} \right] \frac{\Psi_{31}}{\pi}$$

$$R_\beta = \frac{X_5(\phi, \psi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left\{ \frac{X_3(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_1(\phi, \psi)}{\pi^2} \right\} - i \frac{V}{b\omega} \left\{ \frac{X_4(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_2(\phi, \psi)}{\pi^2} \right\}$$

$$R_{z_o} = \frac{X_{14}(\phi, \psi)}{\pi^2} - i \frac{V}{b\omega} \left\{ \frac{X_3(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_1(\phi, \psi)}{\pi^2} \right\}$$

$$R_{z_c} = \frac{X_{14}(\phi, \psi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left[2C(\kappa) \frac{X_{11}(\phi, \psi)}{\pi^2} + \frac{X_{12}(\phi, \psi)}{\pi^2} \right] - i \frac{V}{b\omega} \left[\frac{X_{13}(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_1(\phi, \psi)}{\pi^2} \right]$$

TAB LEADING EDGE MOMENT COEFFICIENTS

$$Q_h = \frac{\psi_6}{2\pi} - i \frac{V}{b\omega} \frac{\psi_8}{\pi} C(\kappa)$$

$$Q_a = \frac{\psi_7}{4\pi} - \frac{V^2}{b^2\omega^2} \frac{\psi_8}{\pi} C(\kappa) - i \frac{V}{b\omega} \left\{ \frac{\psi_8}{\pi} C(\kappa) + \frac{\psi_9}{2\pi} \right\}$$

$$Q_\delta = \frac{\psi_{12}}{4\pi^2} - \frac{V^2}{b^2\omega^2} \left\{ \frac{\psi_{10}}{\pi^2} + \frac{\psi_1\psi_8}{\pi^2} C(\kappa) \right\} - i \frac{V}{2b\omega} \left\{ \frac{\psi_2\psi_8}{\pi^2} C(\kappa) + \frac{\psi_{11}}{\pi^2} \right\}$$

$$Q_{y_o} = \frac{\psi_{37}}{2\pi^2} - i \frac{V}{b\omega} \left\{ \frac{\psi_{17}}{\pi^2} + \frac{\psi_1\psi_8}{\pi^2} C(\kappa) \right\}$$

$$Q_{y_c} = \frac{\psi_{37}}{2\pi^2} - i \frac{V}{b\omega} \frac{\psi_{19}}{\pi^2} - \frac{V^2}{b^2\omega^2} \frac{\psi_{18}}{\pi^2} - C(\kappa) \left[\frac{V^2}{b^2\omega^2} \frac{\psi_{13}}{\pi} + i \frac{V}{b\omega} \frac{\psi_1}{\pi} \right] \frac{\psi_8}{\pi}$$

$$Q_\beta = \frac{X_{10}(\phi, \psi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left\{ \frac{X_8(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_6(\phi, \psi)}{\pi^2} \right\} - i \frac{V}{b\omega} \left\{ \frac{X_9(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_7(\phi, \psi)}{\pi^2} \right\}$$

$$Q_{z_o} = \frac{X_{18}(\phi, \psi)}{\pi^2} - i \frac{V}{b\omega} \left\{ \frac{X_8(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_6(\phi, \psi)}{\pi^2} \right\}$$

$$Q_{z_c} = \frac{X_{18}(\phi, \psi)}{\pi^2} - \frac{V^2}{b^2\omega^2} \left[2C(\kappa) \frac{X_{15}(\phi, \psi)}{\pi^2} + \frac{X_{16}(\phi, \psi)}{\pi^2} \right] - i \frac{V}{b\omega} \left[\frac{X_{17}(\phi, \psi)}{\pi^2} + 2C(\kappa) \frac{X_6(\phi, \psi)}{\pi^2} \right]$$

APPENDIX B

ITERATIVE PROCEDURE SOFTWARE


```

C *****
C *
C *   Program BRANCH calculates the flutter velocities, frequencies
C * and mode shapes for rectangular and tapered airfoil sections
C * with or without ailerons. The aerodynamic coefficients are cal-
C * culated using strip theory and Theodorsen's unsteady theory. The
C * program requires two input files which contain the following
C * information:
C *   File RUN.DAT      Air density, reference semi-chord length,
C *                     array of elastic axis location values, array
C *                     of chord ratio values, array of moment of
C *                     inertia values, array of mass per unit length
C *                     values, array of c.g. location values,
C *                     aileron moment of inertia, aileron c.g.
C *                     location, hinge position, leading edge of
C *                     aileron position
C *   File [filename].Q  Normal mode shapes of deflection,
C *                     frequencies of the normal modes
C *
C *****
C
C   PROGRAM BRANCH
C
C *****
C *
C *   Set parameters to control the array size according to the type
C * of run that is being performed. The parameters are the following:
C *
C *       N - Number of nodal points used in describing the wing
C *       M - Number of degrees of freedom at each nodal point
C *       NM - Total number of degrees of freedom
C *       NSIGB - Total number of normal modes used in the model
C *
C *****
C
C   PARAMETER (N = 1)
C   PARAMETER (M = 2)
C   PARAMETER (NM = 2)
C   PARAMETER (NSIGB = 2)
C
C *****
C *
C *   Character, logical and integer variable declarations
C *
C *       FNAME - Name of the file containing the run information
C *       PROTYP - Character describing type of problem
C *       I,J,L - Loop counters
C *       LOC - Mode undergoing flutter
C *       INFO - Variable reporting the status of the calculation
C *              of the eigenvalues upon return from the external
C *              subroutine CGEEV
C *       AGAP,TGAP - Character variables describing the nature of

```

```

C *          aileron and tab gaps                      *
C *      DRMODE - Reference degree of freedom          *
C *      FINISH - Logical variable controlling iteration loop *
C *          LC - Flutter mode number storage array    *
C *          CT - Number of axis crossings counter     *
C *          TL - Temporary storage variable           *
C *      *****
C
C      CHARACTER*8 FNAME,AGAP*2,TGAP*2,PROTYP*6
C
C      INTEGER J,L,INFO,LC(20),LOC,DRMODE,CT,TL,PTE(NSIGB),IT,I
C
C      LOGICAL FINISH
C
C      *****
C *
C *      Real variable declarations
C *
C *      ARRAYS:
C *
C *      WORK - Array required by the external routine CGZEV
C *      EFF - Array containing values to be used in the inter-
C *            polation of the Theodorsen function
C *      MPL - Mass per unit length at each nodal point
C *      IA - Moment of inertia storage
C *      XA - Storage of c.g. position values
C *      A - Storage of elastic axis location values
C *      C - Reduced aileron hinge location values
C *      E - Reduced aileron leading edge values
C *      IB - Storage of aileron mass moment of inertia values
C *      XB - Aileron centre of gravity locations
C *      D - Reduced location of the tab hinge
C *      F - Storage for reduced location of tab leading edge
C *      ID - Mass moment of inertia of tab about its hinge
C *      XD - Tab centre of gravity location
C
C *      BRAT - Ratio of reference value to the local chord
C *      DI - Deflection of ith non-linear spring
C *      EPS - Convergence limit for iteration loop
C *      K1,K2 - Non-linear spring stiffnesses
C *      S - Point of spring non-linearity
C *      PRE - Spring preload amplitude
C *      MU - Preselected column of modal amplitudes
C *      PHI - Row matrix of eigenvector terms for reference DOF
C *      VS - Flutter velocity storage array
C *      KS - Flutter reduced frequency storage array
C *      DG - Axis crossing slope storage array
C *      G - Structural damping array
C
C *      VARIABLES:
C *

```

```

C *      RHO - Air density value *
C *      K - Reduced frequency (Strouhal Number) value *
C *      DK - Reduced frequency step size *
C *      TMP - Temporary storage variable *
C *      BR - Reference semi-chord length *
C *      W - Mode frequency value *
C *      VEL - Velocity value *
C *      DA - Damping parameter value *
C *      X,Y - Temporary storage variables *
C *      AR - Scale factor *
C *      DR - Reference spring deflection *
C *      TSR,TSH - Reduced aileron and tab gap sizes (reduced by entire *
C *                  wing chord) *
C *      TD,TW - Temporary storage variables *
C *
C *****
C
C      REAL WORK(3*NSIGB),EFF(144,3),RHO,C(N),E(N),D(N),F(N)
C      REAL K,TMP,K2(M),S(M),AR,IB(N),XB(N),ID(N),XD(N)
C      REAL MPL(N),IA(N),XA(N),A(N),BRAT(N),BR,TSR,TSH,SMALL
C      REAL W(NSIGB,2),VEL(NSIGB,2),X,DI(M,2),EPS(M),Y,K1(M),PRE(M)
C      REAL MU(NSIGB),DR,PHI(NSIGB),MAG(NSIGB,NSIGB),DA(NSIGB,2)
C      REAL DK,VS(20),KS(20),DG(20),TD,G(4,N),TW,SUM
C
C *****
C *
C *      Complex variable declarations *
C *
C *      ARRAYS: *
C *
C *      MASS - NM x NM mass matrix *
C *      GENMASS - Generalized mass matrix *
C *      Q - Normal mode array *
C *      QT - Transpose of the normal mode array *
C *      TMP1,TMP2 - Temporary storage for the multiplication routine *
C *      AERO - NM x NM aerodynamic matrix *
C *      V - Array of eigenvectors *
C *      GENAERO - Generalized aerodynamic matrix *
C *      PROD - Product of INVK and GENAERO + GENMASS *
C *      EIG - Vector of eigenvalues *
C *      FREQ - Storage vector for complex mode frequencies *
C *      INVK - Inverse of generalized stiffness matrix *
C *      SYSK - System stiffness matrix *
C *      GENK - Generalized stiffness matrix *
C *
C *      VARIABLES: *
C *
C *      CK - Theodorsen function *
C *
C *****
C

```

```

COMPLEX MASS(NM,NM),GENMASS(NSIGB,NSIGB),Q(NM,NSIGB),CK
COMPLEX QT(NSIGB,NM),TMP1(NSIGB,NM),AERO(NM,NM),V(NSIGB,NSIGB)
COMPLEX GENAERO(NSIGB,NSIGB),PROD(NSIGB,NSIGB),EIG(NSIGB)
COMPLEX FREQ(NSIGB),INVK(NSIGB,NSIGB),SYSK(NM,NM)
COMPLEX GENK(NSIGB,NSIGB),TMP2(NM,NSIGB)
C
C * Common block declarations
C
COMMON/THEODR/K,EFF,CK
COMMON/GAPTYP/AGAP,TSR,TGAP,TSH
COMMON/NOM/FNAME,RHO,BR,PROTYP
C
C * Call to routine which loads airfoil physical data
C
CALL ENTRY(N,M,A,C,D,E,F,BRAT,MPL,IA,XA,IB,XB,ID,XD,G)
C
C * Call routine to load the characteristics of the linear and/or
C * non-linear modal springs
C
CALL SPRING(DI,EPS,K1,K2,S,PRE,DR,DRMODE,M)
C
C * Call to routine to load values into the Theodorsen interpolation
C * array
C
CALL THPREP
C
C * Call routine to load in and transpose the normal mode array
C
CALL QLOAD(FNAME,NM,NSIGB,Q,QT)
C
C * Call routine to calculate the elements of the expanded mass matrix
C
CALL MLOAD(MASS,NM,N,MPL,XA,IA,XB,IB,C,M,A,D,XD,ID,BRAT)
C
C * Pre- and Post- multiply the expanded mass matrix by the normal and
C * transposed normal mode arrays to obtained the generalized mass
C * matrix
C
CALL CMLT(QT,MASS,TMP1,NSIGB,NM,NM)
CALL CMLT(TMP1,Q,GENMASS,NSIGB,NM,NSIGB)
C
C * Call to routine which uses the external stiffness function to
C * calculate the elements of the generalized stiffness matrix
C
CALL KASS(SYSK,NM,K1,K2,S,PRE,M,DI,G,N)
CALL CMLT(QT,SYSK,TMP1,NSIGB,NM,NM)
CALL CMLT(TMP1,Q,GENK,NSIGB,NM,NSIGB)
CALL INVERT(GENK,INVK,NSIGB)
C
OPEN (UNIT=1,FILE=FNAME//'.FLT')

```

```

C
C * Begin loop which will calculate increasing values of the reduced
C * frequency, and which will then determine a set of flutter values
C * at each reduced frequency
C
      DK = .025
      DO 10, I = 200,1,-1
        K = DK*I
C
C * Call the routine to interpolate and find the value of the
C * Theodorsen function C(k)
C
      CALL INTERP
C
C * Call to the routine which will calculate the expanded aerodynamic
C * matrix values
C
      CALL ALOAD(AERO,NM,N,M,A,C,D,E,F,BRAT)
C
C * Multiply to obtain the generalized aerodynamic matrix, and add the
C * result to the generalized mass matrix
C
      CALL CMLT(QT,AERO,TMP1,NSIGB,NM,NM)
      CALL CMLT(TMP1,Q,GENAERO,NSIGB,NM,NSIGB)
      DO 20, J = 1,NSIGB
        DO 30, L = 1,NSIGB
          GENAERO(J,L) = GENAERO(J,L) + GENMASS(J,L)
30      CONTINUE
20      CONTINUE
C
C * Multiply the inverse of the generalized stiffness matrix and the
C * sum of A and M, and pass the result to the external routine CGEEV
C
      CALL CMLT(INVK,GENAERO,PROD,NSIGB,NSIGB,NSIGB)
      IF (I.EQ.200) THEN
        CALL CGEEV(PROD,NSIGB,NSIGB,EIG,V,NSIGB,WORK,1,INFO)
        CALL MAGORD(V,MAG,NSIGB,PTE)
      ELSE
        CALL CGEEV(PROD,NSIGB,NSIGB,EIG,V,NSIGB,WORK,0,INFO)
      END IF
C
C * Invert the eigenvalues and take the square roots to obtain the mod
e
C * frequencies given by the flutter solution
C
      DO 40, J = 1,NSIGB
        FREQ(J) = 1.0/SQRT(EIG(J))
40      CONTINUE
C
C * Calculate the velocity, damping parameter and frequency for each
C * of the mode frequencies, and print the results in a file
C

```

```

      IF (I.EQ.200) THEN
        DO 45, J = 1, NSIGB
          W(J,1) = REAL(FREQ(PTE(J)))
          DA(J,1) = AIMAG(FREQ(PTE(J)))
          VEL(J,1) = BR*W(J,1)/K
45      CONTINUE
        ELSE
          DO 50, J = 1, NSIGB
            SMALL = 1000.0
            TW = REAL(FREQ(J))
            TD = AIMAG(FREQ(J))
            DO 55, L = 1, NSIGB
              X = (TW-W(L,2))**2 + (TD-DA(L,2))**2
              IF (X.LT.SMALL) THEN
                SMALL = X
                TL = L
              END IF
55      CONTINUE
            W(TL,1) = TW
            DA(TL,1) = TD
            VEL(TL,1) = BR*W(TL,1)/K
50      CONTINUE
        END IF
C
        DO 60, J = 1, NSIGB
          WRITE(1,100) W(J,1), VEL(J,1), DA(J,1)
60      CONTINUE
C
        DO 65, J = 1, NSIGB
          W(J,2) = W(J,1)
          DA(J,2) = DA(J,1)
          VEL(J,2) = VEL(J,1)
65      CONTINUE
C
100  FORMAT(5X,E10.3,5X,E10.3,5X,E10.3)
110  FORMAT(3(5X,E10.3))
120  FORMAT(//)
      CLOSE(1)
C
C * Call subroutine to determine the reduced frequencies and modes whe
re
C * flutter occurs for this case
C
      CALL FLTSRCH(FNAME/' '.FLT',DK,200,0.0,NSIGB,CT,VS,LC,KS,DG)
C
C * Call routine to print the flutter points on the screen in ascendin
g
C * order of velocity
C
      CALL FLTDSP(CT,VS,LC,KS,DG)
C

```

```

C * If the problem is non-linear, begin iteration loop to find altered
C * flutter location
C
      IF (PROTYP.EQ.'NONLIN') THEN
C
C * Determine if non-linear iteration procedure has converged, and
C * set the logical variable FINISH
C
500    CONTINUE
      DO 200, I = 1,M
        X = ABS((DI(I,1)-DI(I,2))/DI(I,1))
        IF (X.GE.EPS(I)) THEN
          FINISH = .FALSE.
        ELSE
          FINISH = .TRUE.
        END IF
      200    CONTINUE
C
C * If the procedure has not converged, begin the iteration loop
C
      IF (.NOT.FINISH) THEN
C
C * Have the user enter the choice for the flutter point of interest
C
        WRITE(*,720) CT
        WRITE(*,725)
        READ(*,730) LOC
C
720    FORMAT(//,2X,'YOU HAVE ',I2,' FLUTTER POINT(S) TO CHOOSE FROM')
725    FORMAT(2X,'ENTER THE NUMBER OF THE FLUTTER POINT OF INTEREST ',\
)
730    FORMAT(I2)
        K = KS(LOC)
C
C * Reset the non-linear deflection array
C
      DO 210, I = 1,M
        DI(I,2) = DI(I,1)
      210    CONTINUE
C
C * Calculate the generalized matrices for the aerodynamic coefficient
S
C
      CALL INTERP
      CALL ALOAD(AERO,N,M,A,C,D,E,F,BRAT)
C
      CALL CMLT(QT,AERO,TMP1,NSIGB,NM,NM)
      CALL CMLT(TMP1,Q,GENAERO,NSIGB,NM,NSIGB)
C
      DO 220, I = 1,NSIGB
        DO 230, J = 1,NSIGB
          GENAERO(I,J) = GENAERO(I,J) + GENMASS(I,J)

```

```

230          CONTINUE
220          CONTINUE
C
C * Solve the flutter equation at this point
C
          CALL CMLT(INVK,GENAERO,PROD,NSIGB,NSIGB,NSIGB)
          CALL CGEEV(PROD,NSIGB,NSIGB,EIG,V,NSIGB,WORK,1,INFO)
C
C * Determine the mode that is fluttering, based upon the magnitude of
C * the damping parameter
C
          DO 232, I = 1,NSIGB
             FREQ(I) = 1.0/SQRT(EIG(I))
232          CONTINUE
             SMALL = 1000.0
             DO 234, I = 1,NSIGB
                DA(I,1) = AIMAG(FREQ(I))
                X = ABS(DA(I,1))
                IF (X.LT.SMALL) THEN
                   SMALL = X
                   LOC = I
                END IF
234          CONTINUE
C
C * Calculate the preselected column of modal amplitudes from the
C * flutter solution
C
          DO 240, I = 1,NSIGB
             X = SQRT((REAL(V(I,LOC)))**2 + (AIMAG(V(I,LOC)))**2)
             MU(I) = X
240          CONTINUE
C
C * Obtain the row matrix of eigenvector terms for the reference
C * degree of freedom (stored in DRMODE)
C
          DO 250, I = 1,NSIGB
             PHI(I) = Q(DRMODE,I)
250          CONTINUE
C
C * Calculate the scale factor
C
          SUM = 0.0
          DO 260, I = 1,NSIGB
             SUM = SUM + MU(I)*PHI(I)
260          CONTINUE
          AR = DR/SUM
C
C * Calculate the net displacements of the linear and/or non-linear
C * springs
C
          DO 270, I = 1,M
             SUM = 0.0

```



```

DO 280, J = 1, NSIGB
    SUM = SUM + REAL(Q(I,J)) * MU(J)
280    CONTINUE
    DI(I,1) = AR*SUM
270    CONTINUE
C
C * Calculate new flutter solution
C
    CALL KASS(SYSK, NM, K1, K2, S, PRE, M, DI, C, N)
    CALL CMLT(QT, SYSK, TMP1, NSIGB, NM, NM)
    CALL CMLT(TMP1, Q, GENK, NSIGB, NM, NSIGB)
    CALL INVERT(GENK, INVK, NSIGB)
    CALL CMLT(INVK, GENMASS, PROD, NSIGB, NSIGB, NSIGB)
    CALL CGEEV(PROD, NSIGB, NSIGB, EIG, V, NSIGB, WORK, 1, INFO)
    CALL MAGORD(V, MAG, NSIGB, PTE)
C
C * Calculate the mode shapes of the shifted system
C
    DO 290, I = 1, NM
        DO 300, J = 1, NSIGB
            SUM = 0.0
            DO 310, L = 1, NSIGB
                X = REAL(V(L, PTE(J)))
                Y = AIMAG(V(L, PTE(J)))
                SUM = SUM + REAL(Q(I, L)) * SQRT(X**2 + Y**2)
310            CONTINUE
            TMP2(I, J) = CMPLX(SUM, 0.0)
300        CONTINUE
290    CONTINUE
C
    DO 320, I = 1, NM
        DO 330, J = 1, NSIGB
            Q(I, J) = TMP2(I, J)
            QT(J, I) = TMP2(I, J)
330        CONTINUE
320    CONTINUE
C
    CALL CMLT(QT, SYSK, TMP1, NSIGB, NM, NM)
    CALL CMLT(TMP1, Q, GENK, NSIGB, NM, NSIGB)
    CALL INVERT(GENK, INVK, NSIGB)
    CALL CMLT(QT, MASS, TMP1, NSIGB, NM, NM)
    CALL CMLT(TMP1, Q, GENMASS, NSIGB, NM, NSIGB)
    OPEN (UNIT=1, FILE=FNAME/' .NL')
    TMP = K
    DK = .01
    IT = 0
    DO 340, I = 200, 0, -1
        K = TMP - 1.0 + I*DK
        IF (K.GE.0) THEN
            IT = IT + 1
            CALL INTERP
            CALL ALOAD(AERO, NM, N, M, A, C, D, E, F, BRAT)

```

```

CALL CMLT(QT,AERO,TMP1,NSIGB,NM,NM)
CALL CMLT(TMP1,Q,GENAERO,NSIGB,NM,NSIGB)
DO 350, J = 1,NSIGB
  DO 360, L = 1,NSIGB
    GENAERO(J,L) = GENAERO(J,L) + GENMASS(J,L)
360    CONTINUE
350  CONTINUE
CALL CMLT(INVK,GENAERO,PROD,NSIGB,NSIGB,NSIGB)
CALL CGEEV(PROD,NSIGB,NSIGB,EIG,V,NSIGB,WORK,0,INFO)
DO 370, J = 1,NSIGB
  FREQ(J) = 1.0/SQRT(EIG(J))
370  CONTINUE
  IF (IT.EQ.1) THEN
    DO 400, J = 1,NSIGB
      W(J,1) = REAL(FREQ(J))
      DA(J,1) = AIMAG(FREQ(J))
      VEL(J,1) = BR*W(J,1)/K
400    CONTINUE
    ELSE
      DO 410, J = 1,NSIGB
        SMALL = 1000.0
        TW = REAL(FREQ(J))
        TD = AIMAG(FREQ(J))
        DO 420, L = 1,NSIGB
          X = (TW-W(L,2))**2 + (TD-DA(L,2))**2
          IF (X.LT.SMALL) THEN
            SMALL = X
            TL = L
          END IF
420        CONTINUE
        W(TL,1) = TW
        DA(TL,1) = TD
        VEL(TL,1) = BR*W(TL,1)/K
410      CONTINUE
      END IF
C
      DO 430, J = 1,NSIGB
        WRITE(1,100) W(J,1),VEL(J,1),DA(J,1)
430      CONTINUE
C
      DO 440, J = 1,NSIGB
        W(J,2) = W(J,1)
        DA(J,2) = DA(J,1)
        VEL(J,2) = VEL(J,1)
440      CONTINUE

      END IF
340    CONTINUE
    CLOSE (1)
C
    X = TMP + 1.0 - IT*DK
    CALL FLTSRCH(FNAME/' '.NL',DK,IT,X,NSIGB,CT,VS,LC,KS,DG)

```

```

      CALL FLTDSP(CT,VS,LC,KS,DG)
C
      GOTO 500
C
      END IF
C
      END IF
C
      END
C
C
C
C *****
C *
C *   Subroutine ENTRY loads the physical data concerning the airfoil*
C *   into the appropriate storage locations from the external file *
C *   RUN.DAT. The variables used in the call to this routine are as *
C *   follows:
C *
C *       FNAME - Filename describing the nature of the run
C *       N - Number of nodal points
C *       RHO - Air density
C *       A - Vector of elastic axis location values
C *       B - Vector of chord ratio values
C *       IA - Mass moment of inertia values
C *       MPL - Mass per unit length values
C *       XA - Vector of reduced c.g. location values
C *       BR - Reference semi-chord value
C *       M - Number of degrees of freedom at each node
C *       IB - Aileron mass moment of inertia
C *       XB - Aileron reduced c.g. location
C *       C - Reduced aileron hinge location
C *       E - Reduced position of the aileron leading edge
C *       G - Structural damping storage array
C *       D - Reduced tab hinge location
C *       F - Reduced position of the tab leading edge
C *       ID - Tab mass moment of inertia
C *       XD - Reduced location of the tab centre of gravity
C *
C *   Other variables:
C *
C *       I - Loop counter
C *       PROTYP - Problem type identifier
C *       AGAP - Character describing nature of aileron l.e. gap
C *       TGAP - Description of tab leading edge gap
C *       TSR - Reduced size of aileron l.e. gap
C *       TSH - Reduced size of tab leading edge gap
C *
C *****
C
      SUBROUTINE ENTRY(N,M,A,C,D,E,F,B,MPL,IA,XA,IB,XB,ID,XD,G)
C
      CHARACTER*8 FNAME,AGAP*2,TGAP*2,PROTYP*6
      INTEGER I,N,M

```

```

REAL RHO,BR,A(N),C(N),D(N),E(N),F(N),B(N),G(4,N)
REAL MPL(N),IA(N),XA(N),IB(N),XB(N),ID(N),XD(N),TSR,TSH
C
COMMON/NOM/FNAME,RHO,BR,PROTYP
COMMON/GAPTYP/AGAP,TSR,TGAP,TSH
C
OPEN (UNIT=1,FILE='RUN.DAT',STATUS='OLD')
CALL CHAREJ
READ(1,100) FNAME
C
C * Read problem type (linear or non-linear springs)
C
CALL CHAREJ
READ(1,110) PROTYP
C
C * Read the density and reference chord for this case
C
CALL CHAREJ
READ(1,120) RHO,BR
C
C * At each node, read in the relevant information
C
CALL CHAREJ
DO 10, I = 1,N
  READ(1,130) A(I),B(I),IA(I),MPL(I),XA(I),G(1,I),G(2,I)
10 CONTINUE
C
C * If the program requires the information, load the physical data
C * for the aileron into storage
C
IF (M.GE.3) THEN
  CALL CHAREJ
  DO 20, I = 1,N
    READ(1,140) IB(I),XB(I),C(I),E(I),G(3,I)
20 CONTINUE
  CALL CHAREJ
  READ(1,150) AGAP,TSR
END IF
C
IF (M.EQ.4) THEN
  CALL CHAREJ
  DO 30, I = 1,N
    READ(1,140) ID(I),XD(I),D(I),F(I),G(4,I)
30 CONTINUE
  CALL CHAREJ
  READ(1,150) TGAP,TSH
END IF
C
CLOSE(1)
C
100 FORMAT(2X,A8)
110 FORMAT(2X,A6)

```

```

120  FORMAT(2(2X,F6.3))
130  FORMAT(2X,F6.3,2X,F6.3,2X,F9.3,2X,F9.3,3(2X,F6.3))
140  FORMAT(2X,F8.5,4(2X,F6.3))
150  FORMAT(2X,A2,5X,F6.4)
C
      RETURN
      END
C
C
C
C *****
C *
C *   Subroutine SPRING loads the characteristics of the modal
C *   springs from the external user-defined file SPRING.DAT. Bilinear
C *   springs with preload and linear springs can be modelled using
C *   the variables used by this routine. The variables used by this
C *   routine are:
C *
C *           DI - Deflection of the ith non-linear spring
C *           K1,K2 - Spring stiffnesses
C *           S - Half of K2 amplitude range
C *           PRE - Spring preload amplitude
C *           EPS - Convergence limit for iteration loop
C *           I - Loop counter
C *           M - Number of degrees of freedom at each node
C *           DRMODE - Reference degree of freedom
C *           DR - Reference spring deflection/limit cycle
C *
C *****
C
      SUBROUTINE SPRING(DI, EPS, K1, K2, S, PRE, DR, DRMODE, M)
C
      INTEGER I, M, DRMODE
      REAL DI(M, 2), EPS(M), K1(M), K2(M), S(M), DR, PRE(M)
C
      OPEN(1, FILE='SPRING.DAT')
C
      CALL CHAREJ
      READ(1, 100) DR, DRMODE
      CALL CHAREJ
      DO 10, I = 1, M
         READ(1, 110) K1(I), K2(I), S(I), PRE(I), DI(I, 1), EPS(I)
         DI(I, 2) = 0.0
10    CONTINUE
C
100   FORMAT(5X, E10.3, 5X, I2)
110   FORMAT(6(2X, E10.3))
C
      CLOSE(1)
C
      RETURN
      END

```

```

C
C
C
C *****
C *
C *   Subroutine THPREP loads the array used in determining the
C * values of the Theodorsen function C(k) from the external file
C * THEODR.DAT. The first column of the F array contain the values
C * of the reduced frequency, the second column contains the F
C * values and the third column contains the G values necessary to
C * calculate  $C(k) = F + iG$ . The variables used by the routine are
C * the following:
C *
C *
C *       J1,J2 - Loop counters
C *       EFF - Value storage array
C *       K - Reduced frequency value
C *       CK - Complex Theodorsen function
C *
C *****
C
C   SUBROUTINE THPREP
C
C   INTEGER J1,J2
C   REAL EFF(144,3),K
C   COMPLEX CK
C
C   COMMON/THEODR/K,EFF,CK
C
C   OPEN (UNIT=1,FILE='THEODR.DAT',STATUS='OLD')
C   DO 10, J1 = 1,144
C       READ(1,200) (EFF(J1,J2),J2=1,3)
10  CONTINUE
200  FORMAT(2X,F7.3,5X,F7.5,5X,F7.5)
C   CLOSE(1)
C
C   RETURN
C   END
C
C *****
C *
C *   Subroutine INTERP uses linear interpolation to calculate the
C * value of the Theodorsen function C(k), given the value of the
C * reduced frequency. The variables used in the routine are given
C * below:
C *
C *       K - Reduced frequency value
C *       EFF - Value storage array
C *       X - Fractional value used in interpolation
C *       FPRIME - Interpolated value of the F function
C *       GPRIME - Interpolated value of the G function
C *       CK - Complex Theodorsen function
C *       I - Loop counter

```

```

C *
C *****
C
C     SUBROUTINE INTERP
C
C     REAL K,EFF(144,3),FPRIME,GPRIME,X
C     COMPLEX CK
C     INTEGER I
C
C     COMMON/THEODR/K,EFF,CK
C
C * Loop until the correct range of the reduced frequency is found, and
d
C * then calculate the fractional position of the given reduced
C * frequency within this range. Then calculate the interpolated value
s
C * of the F and G functions
C
C     DO 10, I = 1,143
C         IF ((K.LT.EFF(I,1)).AND.(K.GE.EFF(I+1,1))) THEN
C             X = (K-EFF(I+1,1))/(EFF(I,1)-EFF(I+1,1))
C             FPRIME = EFF(I+1,2) + X*(EFF(I,2)-EFF(I+1,2))
C             GPRIME = EFF(I+1,3) + X*(EFF(I,3)-EFF(I+1,3))
C         END IF
10    CONTINUE
C     CK = CMPLX(FPRIME,GPRIME)
C
C     RETURN
C     END
C
C
C
C *****
C *
C * Subroutine QLOAD loads the normal mode array into storage
C * within the program from the external file [filename].Q. The
C * routine also transposes the array to obtain the necessary
C * matrices for converting the expanded arrays into generalized
C * form. The variables required by the CALL statement to this
C * routine include the following:
C *
C * FNAME - Name of the file containing the mode shapes
C * NM - Total number of degrees of freedom
C * N - Number of normal modes used in the model
C * Q - Normal mode array
C * QT - Transpose of the normal mode array
C *
C * The columns of the array Q must be arranged in the following
C * order: all of the flexural modes, followed by all of the
C * torsional modes, and then the aileron modes (if any).
C *
C * Other variables used by the routine include the following:

```

```

C *
C *          I,J - Loop counters
C *          DUM - Temporary storage array
C *
C *****
C
C      SUBROUTINE QLOAD(FNAME,NM,N,Q,QT)
C
C      CHARACTER*8 FNAME
C      INTEGER NM,N,I,J
C      COMPLEX Q(NM,N),QT(N,NM)
C      REAL DUM(20)
C
C      OPEN (UNIT=1,FILE=FNAME//'.Q',STATUS='OLD')
C
C * Load and transpose elements of the normal mode array
C
C      DO 10, I = 1,NM
C          READ(1,*) (DUM(J),J=1,N)
C          DO 20, J = 1,N
C              Q(I,J) = CMPLX(DUM(J),0.0)
C          CONTINUE
C      10 CONTINUE
C      DO 40, I = 1,NM
C          DO 50, J = 1,N
C              QT(J,I) = Q(I,J)
C          CONTINUE
C      50 CONTINUE
C      40 CONTINUE
C
C      CLOSE(1)
C
C      RETURN
C      END
C
C
C *****
C *
C *      Subroutine MLOAD calculates the elements of the expanded NM x
C *      NM matrix using the physical data stored by the computer program.*
C *      The variables used by the subroutine include the following:
C *
C *      MASS - Expanded mass matrix
C *      NM - Total number of degrees of freedom
C *      N - Number of nodal points
C *      MPL - Mass per unit length values
C *      BR - Reference semi-chord length value
C *      XA - Reduced wing c.g. location values
C *      IA - Mass moment of inertia values
C *      XB - Aileron reduced c.g. location
C *      IB - Mass moment of inertia of the aileron
C *      C - Reduced aileron hinge location

```



```

C *      D - Reduced tab hinge location                      *
C *      XD - Tab reduced c.g. location value                *
C *      ID - Mass moment of inertia of tab about its hinge *
C *      BRAT - Ratio of reference to local chord value      *
C *      M - Number of degrees of freedom at each node      *
C *      A - Reduced elastic axis location values            *
C *                                                         *
C * Other variables used by the routine include:              *
C *                                                         *
C *      I,J - Loop counters                                  *
C *      Y,Z - Temporary storage variables                    *
C *                                                         *
C *****
C
C      SUBROUTINE MLOAD(MASS,NM,N,MPL,XA,IA,XB,IB,C,M,A,D,XD,ID,BRAT)
C
C      CHARACTER*8 FNAME,PROTYP*6
C      INTEGER I,J,NM,N,M
C      COMPLEX MASS(NM,NM)
C      REAL MPL(N),BR,XA(N),IA(N),XB(N),IB(N),C(N),A(N),Z,Y
C      REAL XD(N),D(N),ID(N),BRAT(N)
C
C      COMMON/NOM/FNAME,RHO,BR,PROTYP
C
C * Zero all of the elements of the mass matrix
C
C      DO 10, I = 1,NM
C        DO 20, J = 1,NM
C          MASS(I,J) = CMPLX(0.0,0.0)
C        20 CONTINUE
C      10 CONTINUE
C
C * Loop through all of the nodal points
C
C      DO 30, J = 1,N
C
C * Calculate the step size to be used in the array assignments, based
C * upon the number of degrees of freedom at each node
C
C      IF (M.EQ.2) THEN
C        I = INT(2*J-1)
C      ELSEIF (M.EQ.3) THEN
C        I = INT(3*J-2)
C      ELSEIF (M.EQ.4) THEN
C        I = INT(4*J-3)
C      END IF
C
C * Assign the values of the mass elements which are common to both
C * values of the number of degrees of freedom
C
C      Y = BR/BRAT(J)
C

```

```

      MASS(I,I) = CMPLX(MPL(J),0.0)
      MASS(I,I+1) = CMPLX((MPL(J)*Y*XA(J)),0.0)
      MASS(I+1,I) = MASS(I,I+1)
      MASS(I+1,I+1) = CMPLX(IA(J),0.0)
C
C * If the program requires it, calculate the mass matrix elements
C * relating to the aileron motion
C
      IF (M.GE.3) THEN
        MASS(I,I+2) = CMPLX((MPL(J)*Y*XB(J)),0.0)
        MASS(I+2,I) = MASS(I,I+2)
        Z = IB(J) + Y*(C(J)-A(J))*MPL(J)*Y*XB(J)
        MASS(I+1,I+2) = CMPLX(Z,0.0)
        MASS(I+2,I+1) = CMPLX(Z,0.0)
        MASS(I+2,I+2) = CMPLX(IB(J),0.0)
      END IF
C
C * If required, calculate the elements of the mass matrix that relate
C * to the motion of the tab
C
      IF (M.EQ.4) THEN
        MASS(I,I+3) = CMPLX((MPL(J)*Y*XD(J)),0.0)
        MASS(I+3,I) = MASS(I,I+3)
        Z = ID(J)+MPL(J)*Y*XD(J)*Y*(D(J)-A(J))
        MASS(I+1,I+3) = CMPLX(Z,0.0)
        MASS(I+3,I+1) = CMPLX(Z,0.0)
        Z = ID(J)+MPL(J)*Y*XD(J)*Y*(D(J)-C(J))
        MASS(I+2,I+3) = CMPLX(Z,0.0)
        MASS(I+3,I+2) = CMPLX(Z,0.0)
        MASS(I+3,I+3) = CMPLX(ID(J),0.0)
      END IF
C
30    CONTINUE
C
      RETURN
      END
C
C
C
C *****
C *
C * Subroutine CMLT multiplies two arbitrary complex arrays to
C * produce the product array. The variables used in the call to
C * this routine are the following:
C *
C *      X(I,J) - First array
C *      Y(J,L) - Second array
C *      Z(I,L) - Product array
C *      I,J,L - Array dimensions
C *
C * Other variables used are:
C *

```

```

C *          SUM - Temporary storage variable          *
C *          I1,I2,I3 - Loop counters                  *
C *          *
C *****
C
C      SUBROUTINE CMLT(X,Y,Z,I,J,L)
C
C      COMPLEX X(I,J),Y(J,L),Z(I,L),SUM
C      INTEGER I,J,L,I1,I2,I3
C
C      DO 10, I1 = 1,I
C          DO 20, I2 = 1,L
C              SUM = CMPLX(0.0,0.0)
C              DO 30, I3 = 1,J
C                  SUM = SUM + X(I1,I3)*Y(I3,I2)
C          30      CONTINUE
C              Z(I1,I2) = SUM
C      20      CONTINUE
C  10      CONTINUE
C
C      RETURN
C      END
C
C
C
C *****
C *
C *      Subroutine KASS uses an external function to calculate the
C *      values of the equivalent stiffness for the modal springs, and
C *      assigns the results to a system stiffness matrix, including
C *      the effects of structural damping, if required. The matrices
C *      containing the spring characteristics must be loaded from an
C *      external file. This particular routine is only valid for the
C *      case where NM = M, ie. for a two-dimensional case using only
C *      base modes. The arrays used by the routine are:
C *
C *      K1,K2 - Non-linear spring stiffness values
C *      S - Half of K2 amplitude range
C *      PRE - Spring pre-load amplitude
C *      DI - Deflection of the ith non-linear spring
C *      KVAL - Equivalent spring stiffness storage array
C *      SYSK - System stiffness matrix
C *      G - Structural damping coefficient array
C *
C *
C *      Other variables used in the subroutine include:
C *
C *      NM - Total number of degrees of freedom
C *      N - Total number of normal modes used
C *      M - Number of degrees of freedom at each node
C *      I,J - Loop counters
C *

```

```

C *****
C
C      SUBROUTINE KASS(SYSK,NM,K1,K2,S,PRE,M,DI,G,N)
C
C      INTEGER NM,N,M,I,J
C      REAL K1(M),K2(M),S(M),DI(M,2),KVAL(10),PRE(M),G(4,N)
C      COMPLEX SYSK(NM,NM)
C
C * Calculate the equivalent stiffness values of the linear and/or
C * non-linear springs
C
C      DO 30, I = 1,M
C          KVAL(I) = STIFF(K1,K2,S,PRE,M,DI(I,1),I)
30  CONTINUE
C
C * Assign values to system stiffness matrix
C
C      DO 40, I = 1,NM
C          DO 50, J = 1,NM
C              IF (I.EQ.J) THEN
C                  SYSK(I,J) = CMPLX(KVAL(I),G(I,1)*KVAL(I))
C              ELSE
C                  SYSK(I,J) = CMPLX(0.0,0.0)
C              END IF
50  CONTINUE
40  CONTINUE
C
C      RETURN
C      END
C
C
C *****
C *
C *      Function STIFF calculates the equivalent stiffness of a
C *      bi-linear spring with preload using the describing function
C *      equations given by Lee and Tron. The equivalent stiffness of the
C *      spring is calculated directly from the given information, and
C *      will depend upon the amplitude of displacement. The variables
C *      used by the function include:
C *
C *      K1,K2 - Actual spring stiffness values
C *      S - Half of K2 amplitude range
C *      D - Displacement of non-linear spring
C *      P - Value of the preload displacement
C *      I - Number of the spring being considered
C *      T1,T2 - Trigonometric motion functions
C *      A0,A1 - Steady state and oscillatory amplitudes
C *      KX - Difference of the spring stiffnesses
C *
C *****
C

```

```

FUNCTION STIFF(K1,K2,S,P,M,D,I)
C
PARAMETER (PI=3.1415927)
REAL K1(M),K2(M),S(M),P(M),D,KX,A0,A1,T1,T2,X
INTEGER I,M
C
IF (D.LE.P(I)) THEN
  STIFF = K1(I)
ELSEIF ((D.GT.P(I)).AND.(D.LE.(P(I)+2.0*S(I)))) THEN
  KX = K1(I) - K2(I)
  A0 = D/2.0 - 0.5*SQRT(D**2 - KX/K1(I)*(P(I)-D)*.2)
  A1 = D - A0
  T1 = ACOS((P(I)-A0)/A1)
  X = K1(I)*PI - KX*T1 + 2.0*KX*COS(T1)*SIN(T1)
  X = (X - 0.5*KX*SIN(2.0*T1))/PI
  STIFF = X
ELSEIF (D.GT.(P(I)+2.0*S(I))) THEN
  KX = K1(I) - K2(I)
  A0 = D/2.0 - 0.5*SQRT(D**2-4.0*S(I)*KX/K1(I)*(D-P(I)-S(I)))
  A1 = D - A0
  T1 = ACOS((P(I) - A0)/A1)
  T2 = ACOS((P(I) + 2.0*S(I) - A0)/A1)
  X = 2.0*KX*COS(T1)*SIN(T1) - KX*T1 - 2.0*KX*COS(T2)*SIN(T2)
  X = X + KX*T2 - 0.5*KX*(SIN(2.0*T1)-SIN(2.0*T2)) + K1(I)*PI
  STIFF = X/PI
END IF
C
RETURN
END
C
C
C
C *****
C *
C * Subroutine INVERT finds the inverse of a square complex matrix *
C * of general size. The routine accepts two arrays from the calling *
C * routine, the matrix to be inverted, and an array that will con- *
C * tain the inverted matrix. The variables used in the routine are *
C * the following. *
C *
C *          TMP - Array holding matrix to be inverted *
C *          ID - Array for inverted matrix *
C *          X - Temporary storage variable *
C *          N - Size of matrices *
C *          I,J,L - Loop counters *
C *
C *****
C
SUBROUTINE INVERT(TMP,ID,N)
C
INTEGER N,I,J,L

```

```

      COMPLEX TMP(N,N),ID(N,N),X
C
C * Initialize the inverted array as an identity matrix
C
      DO 10, I = 1,N
        DO 20, J = 1,N
          IF (I.EQ.J) THEN
            ID(I,J) = CMPLX(1.0,0.0)
          ELSE
            ID(I,J) = CMPLX(0.0,0.0)
          END IF
        20 CONTINUE
      10 CONTINUE
C
C * Invert the matrix using simple Gauss-Jordan elimination
C
      DO 70, I = 1,N
        X = TMP(I,I)
        DO 80, J = 1,N
          TMP(I,J) = TMP(I,J)/X
          ID(I,J) = ID(I,J)/X
        80 CONTINUE
        DO 90, J = 1,N
          IF (I.NE.J) THEN
            TMP(J,I) = TMP(J,I) - X*TMP(I,I)
            DO 100, L = 1,N
              TMP(J,L) = TMP(J,L) - X*TMP(I,L)
              ID(J,L) = ID(J,L) - X*ID(I,L)
            100 CONTINUE
          END IF
        90 CONTINUE
      70 CONTINUE
C
      RETURN
      END
C
C
C
C *****
C *
C * Subroutine ALOAD calculates the elements of the expanded aero- *
C * dynamic matrix, using the unsteady aerodynamic theory developed *
C * by Theodorsen, and reported by Smilg & Wasserman. The values of *
C * the aerodynamic coefficients are calculated using the formulae *
C * given by Smilg & Wasserman, with additional information coming *
C * from work presented by Theodorsen & Garrick, Kussner, and *
C * Scanlan and Rosenbaum. The variables required by the CALL to *
C * this routine are the following: *
C * *
C * AERO - Expanded aerodynamic matrix *
C * NM - Total number of degrees of freedom *

```

```

C *      N - Number of nodal points used in the model      *
C *      A - Array storage of the elastic axis locations    *
C *      RHO - Air density value                             *
C *      BRAT - Array of the chord ratio values             *
C *      BR - Reference semi-chord length                   *
C *      C - Reduced aileron hinge location value           *
C *      E - Reduced position of the aileron leading edge   *
C *      D - Reduced position of the tab hinge              *
C *      F - Reduced location of the tab leading edge       *
C *      M - Number of degrees of freedom at each node      *
C *
C * Other variables used in the routine include:             *
C *
C *      I,J - Loop counters                                  *
C *      EFF - Storage array for C(k) interpolation           *
C *      K - Reduced frequency (Strouhal Number)             *
C *      U,V,X,Y,Z - Temporary storage variables            *
C *      KP - Adjusted reduced frequency value               *
C *      CP - Reduced distance between aileron hinge and l.e. *
C *      CK - Theodorsen function C(k)                       *
C *      LH,LA,LB                                             *
C *      LZ,LD,LY - Lift coefficients                        *
C *      MH,MA,MB                                             *
C *      MZ,MD,MY - Coefficients for moment about elastic axis *
C *      TH,TA,TB                                             *
C *      TZ,TD,TY - Coefficients for torque about aileron l.e. *
C *      PH,PA,PB                                             *
C *      PZ,PD,PY - Coefficients for heave force on aileron *
C *      QH,QA,QB                                             *
C *      QZ,QD,QY - Coefficients for torque about tab leading edge *
C *      RH,RA,RB                                             *
C *      RZ,RD,RY - Coefficients of vertical force at tab leading edge *
C *      LAP,MHF,MAP,                                         *
C *      TAP,TBP,LBP,                                         *
C *      MBP,THP,etc.- Combined aerodynamic coefficients including chord *
C *                   and density values                     *
C *      AGAP,TGAP - Character variables describing type of gap at *
C *                   aileron and tab leading edges           *
C *      TSR,TSH - Reduced values of aileron and tab gaps (based on *
C *                   entire wing chord)                      *
C *
C *
C *****
C
C      SUBROUTINE ALOAD(AERO,NM,N,M,A,C,D,E,F,BRAT)
C
C      PARAMETER (PI=3.1415927)
C
C      INTEGER I,NM,N,J,M
C
C      CHARACTER*8 FNAME,PROTYP*6
C

```

```

REAL A(N),EFF(144,3),RHO,BR,BRAT(N),C(N),D(N),E(N),F(N)
REAL K,KP,CP,DP,U,V,X,Y,Z,TSR,TSH
C
COMPLEX AERO(NM,NM),CK,LA,LH,LB,LZ,LD,LY,LHP,LAP,LBP,LDP
COMPLEX MH,MA,MB,MZ,MD,MY,TH,TA,TB,TZ,TD,TY,MHP,MAP,MBP,MDP
COMPLEX PH,PA,PB,PZ,PD,PY,QH,QA,QB,QZ,QD,QY,THP,TAP,TBP,TDP
COMPLEX RH,RA,RB,RZ,RD,RY,QHP,QAP,QBP,QDP
C
CHARACTER*2 AGAP,TGAP
C
COMMON/THEODR/K,EFF,CK
COMMON/GAPTYP/AGAP,TSR,TGAP,TSH
COMMON/NOM/FNAME,RHO,BR,PROTYP
C
C * Initialize the expanded aerodynamic array
C
DO 10, I = 1,NM
DO 20, J = 1,NM
AERO(I,J) = CMPLX(0.0,0.0)
20 CONTINUE
10 CONTINUE
C
C * Loop through all nodal points
C
DO 30, I = 1,N
C
C * Based upon the number of degrees of freedom at each nodal point,
C * determine the value of the step size to be used in making
C * assignments to the array
C
IF (M.EQ.2) THEN
J = INT(2*I-1)
ELSEIF (M.EQ.3) THEN
J = INT(3*I-2)
ELSEIF (M.EQ.4) THEN
J = INT(4*I-3)
END IF
C
C * Calculate the lift and moment coefficients for two degrees of
C * freedom
C
LH = 1.0 + BRAT(I)*CMPLX(0.0,-2.0/K)*CK
LA = 0.5 + BRAT(I)*(CMPLX(0.0,-1.0/K)+CMPLX(0.0,-2.0/K)*CK)
LA = LA + (BRAT(I))**2 * (-2.0/K**2 * CK)
MH = CMPLX(0.5,0.0)
MA = 0.375 + BRAT(I)*CMPLX(0.0,-1.0/K)
C
C * Calculate necessary constants
C
Z = 0.5 + A(I)
Y = BR/BRAT(I)
X = PI*RHO*Y**2

```



```

      U = X*Y
      V = X*(Y**2)
C
C * Calculate the values of the two degree of freedom array elements
C
      LAP = U*(LA-Z*LH)
      MHP = U*(MH-Z*LH)
      MAP = V*(MA-(LA+MH)*Z+LH*Z**2)
      LHP = LH*X
C
C * Assign these values to the array
C
      AERO(J,J) = LHP
      AERO(J,J+1) = LAP
      AERO(J+1,J) = MHP
      AERO(J+1,J+1) = MAP
C
C * If more than two degrees of freedom are involved, calculate the
C * necessary coefficients
C
      IF (M.GE.3) THEN
        KP = K/BRAT(I)
        CP = C(I)-E(I)
C
C * Call the routine to calculate the aerodynamic coefficients
C
        CALL TWELVE(LB,LZ,MB,MZ,TH,TA,TB,TZ,PH,PA,PB,PZ,C(I),KP,AGAP,TSR
      )
C
C * Calculate the array elements
C
      LBP = U*(LB-LZ*CP)
      MBP = V*(MB-LB*Z-MZ*CP+LZ*Z*CP)
      THP = U*(TH-PH*CP)
      TAP = V*(TA-PA*CP-TH*Z+PH*Z*CP)
      TBP = V*(TB-PB*CP-TZ*CP+PZ*CP**2)
C
C * Assign these values to the array
C
      AERO(J,J+2) = LBP
      AERO(J+1,J+2) = MBP
      AERO(J+2,J) = THP
      AERO(J+2,J+1) = TAP
      AERO(J+2,J+2) = TBP
C
      END IF
C
C * If the wing has four degrees of freedom, calculate the necessary
C * aerodynamic coefficients and assign them to the array
C
      IF (M.EQ.4) THEN
        DP = D(I)-F(I)

```

```

C
C * Call routines to calculate the necessary aerodynamic coefficients
C
      CALL TWELVE(LD,LY,MD,MY,QH,QA,QD,QY,RH,RA,RD,RY,D(I),KP,TGAP,TSH
)
C
      CALL FOUR(RB,RZ,QB,QZ,C(I),D(I),KP,AGAP)
      CALL FOUR(PD,PY,TD,TY,D(I),C(I),KP,TGAP)
C
C * Calculate array elements
C
      LDP = U*(LD-DP*LY)
      MDP = V*(MD-DP*MY-Z*LD+DP*Z*LY)
      TDP = V*(TD-DP*TY-CP*PD-CP*DP*PY)
C
      QHP = U*(QH-DP*RH)
      QAP = V*(QA-Z*QH-DP*RA+Z*DP*RH)
      QBP = V*(QB-CP*QZ-DP*RB+DP*CP*RZ)
      QDP = V*(QD-DP*QY-DP*RD-DP**2 * RY)
C
C * Assign these values to the array
C
      AERO(J,J+J) = LDP
      AERO(J+1,J+3) = MDP
      AERO(J+2,J+3) = TDP
      AERO(J+3,J) = QHP
      AERO(J+3,J+1) = QAP
      AERO(J+3,J+2) = QBP
      AERO(J+3,J+3) = QDP
C
      END IF
C
30  CONTINUE
C
      RETURN
      END
C
C
C
C *****
C *
C * Subroutine TWELVE calculates the values of twelve aerodynamic *
C * coefficients based upon the equations derived from Kussner's *
C * work. The twelve coefficients will apply to the aileron degree *
C * of freedom, or to that of the tab, depending upon the value of *
C * Z, which is the reduced distance from the wing elastic axis to *
C * the leading edge of the control surface of interest. Other *
C * variables used by the subroutine include: *
C *
C * A..J,R,L - Aerodynamic coefficients to be calculated *
C * K,KP - Reference and local values of reduced frequency *
C * KPSQ - The square of the variable KP *

```

```

C *          EFF - Array storing Theodorsen interpolation values *
C *          P - Array storing values of Kussner's PHI functions *
C *          S..V - Precalculated values *
C *          CK - Theodorsen aerodynamic function *
C *          GAP - Character describing nature of gap *
C *          TAU - Nondimensional width of gap *
C *
C *****
C
C          SUBROUTINE TWELVE(A,B,C,D,E,F,G,H,I,J,R,L,Z,KP,GAP,TAU)
C
C          PARAMETER (PI=3.1415927)
C          REAL Z,KP,EFF(144,3),K,P(37),KPSQ,TAU
C          COMPLEX A,B,C,D,E,F,G,H,I,J,R,L,S,T,U,V,CK
C          CHARACTER*2 GAP
C          COMMON/THEODR/K,EFF,CK
C
C          CALL PCALC(Z,P)
C
C          KPSQ = KP**2
C          S = 2.0*P(1)*CK
C          T = P(8)*CK
C          U = (P(10)/PI + P(1)*T)/KP
C          V = 2.0*P(31)*CK
C
C          A = P(4)/2.0 - S/KPSQ - CMPLX(0,1)*(P(3)+P(2)*CK)/KP
C          C = P(7)/4.0 - P(5)/KPSQ - CMPLX(0,1)*P(6)/(2.0*KP)
C          E = P(4)/2.0 - CMPLX(0,1)*T/KP
C          F = P(7)/4.0 - T/KPSQ - CMPLX(0,1)*(T+P(9)/2.0)/KP
C          G = P(12)/(4.0*PI) - U/KP - CMPLX(0,1)*(P(2)*T+P(11)/PI)/(2*KP)
C          I = P(3) - CMPLX(0,1)*V/KP
C          J = P(6)/4.0 - CMPLX(0,1)*P(32)/KP - V*CMPLX((1.0/KP**2),1.0/KP)
C          R = P(37)/(2.0*PI) - P(35)/(PI*KPSQ) - CMPLX(0,1)*P(36)/(PI*KP)
C          R = R - V*CMPLX((P(1)/KPSQ),(P(2)/(2.0*KP)))
C
C          IF (GAP.EQ.'OP') THEN
C              B = P(3) - CMPLX(0,1)*S/KP
C              D = P(6)/4.0 - CMPLX(0,1)*P(5)/KP
C              H = P(37)/(4.0*PI) - CMPLX(0,1)*U
C              L = P(17)/PI - CMPLX(0,1)*(P(35)/PI + V*P(1))/KP
C          ELSE
C              B = P(3)-CMPLX(0,1)*P(14)/KP-2.0*CK*CMPLX(P(13)/KPSQ,P(1)/KP)
C              D = P(6)/4-P(15)/KPSQ-CMPLX(0,1)*2.0*P(5)/KP
C              H = P(37)/(2.0*PI)-P(18)/(KPSQ*PI)-CMPLX(0,1)*P(19)/(PI*KP)
C              H = H - T*CMPLX((P(13)/KPSQ),(P(1)/KP))
C              L = P(17)/PI-CMPLX(0,1)*P(16)/(KP*PI)
C              L = L - (2*LOG(TAU)/(PI**2)+P(21)/PI)/KPSQ
C              L = L - V*CMPLX((P(13)/KPSQ),(P(1)/KP))
C          END IF
C
C          RETURN
C          END

```

```

C
C
C
C *****
C *
C * Subroutine FOUR calculates four aerodynamic coefficients which
C * Kussner lists as being functions of both PHI and PSI. These
C * variables are related to Y and Z, which are the reduced distance
C * from the wing elastic axis to the leading edge of the control
C * surface. The order in which these values are passed to the
C * routine determine whether the aerodynamic coefficients apply to
C * the aileron or the tab. Other variables used by the routine
C * include:
C *
C *      A..D - Aerodynamic coefficients
C *      CK - Theodorsen aerodynamic function
C *      Q,R - Precalculated values
C *      K,KP - Reference and local values of reduced frequency
C *      KPSQ - Square of the variable KP
C *      X - Storage array for Kussner's X values
C *      GAP - Character variable describing type of gap
C *
C *****
C
C      SUBROUTINE FOUR(A,B,C,D,Y,Z,KP,GAP)
C
C      REAL Y,Z,PHI,PSI,K,EFF(144,3),X(18),KP,KPSQ
C      COMPLEX A,B,C,D,CK,Q,R
C      CHARACTER*2 GAP
C
C      COMMON/THEODR/K,EFF,CK
C
C      PHI = ACOS(-Y)
C      PSI = ACOS(-Z)
C      KPSQ = KP**2
C
C      CALL XGET(PHI,PSI,X)
C      Q = X(3) + 2.0*X(1)*CK
C      R = X(8) + 2.0*X(6)*CK
C
C      A = X(5) - Q/KPSQ - CMPLX(0,1)*(X(4) + 2.0*X(2)*CK)/KP
C      C = X(10) - R/KPSQ - CMPLX(0,1)*(X(9) + 2.0*X(7)*CK)/KP
C
C      IF (GAP.EQ.'OP') THEN
C          B = X(14) - CMPLX(0,1)*Q/KP
C          D = X(18) - CMPLX(0,1)*R/KP
C      ELSE
C          B = X(14) - (2.0*CK*X(11) + X(12))/KPSQ
C          B = B - CMPLX(0,1)*(X(13) + 2.0*CK*X(1))/KP
C          D = X(18) - (2.0*CK*X(15) + X(16))/KPSQ
C          D = D - CMPLX(0,1)*(X(17) + 2.0*CK*X(6))/KP
C      END IF

```

```

C      RETURN
C      END
C
C
C *****
C *
C *   Subroutine PCALC calculates the values of Kussner's PHI
C *   using equations given by Kussner. The variables in the CALL
C *   statement is the reduced aileron hinge location C, and the
C *   storage array for the PHI values. Other variables that are
C *   used by the routine include:
C *
C *       I - Loop counter
C *       SP - Related to C; corresponds to Kussner's small PHI
C *       A - Temporary variable
C *       P - Storage for the values of the PHI functions
C *
C *****
C
C      SUBROUTINE PCALC(C,P)
C
C      PARAMETER (PI = 3.1415927)
C      INTEGER I
C      REAL C,SP,A,P(37),Y,Z,INPI
C
C      DO 10, I = 1,37
C         P(I) = 0.0
C 10    CONTINUE
C
C      SP = ACOS(-C)
C      A = PI - SP
C      Y = SIN(SP)
C      Z = COS(SP)
C      INPI = 1.0/PI
C
C      P(1) = A + Y
C      P(2) = A*(1.0 + 2.0*Z) + Y*(2.0+Z)
C      P(3) = A + Y*Z
C      P(4) = A*2.0*Z + 2.0*Y/3.0*(2.0 + Z**2)
C
C      P(5) = Y*(1.0 - Z)
C      P(6) = 2.0*A + 2.0*Y/3.0*(2.0 - Z)*(1.0 + 2.0*Z)
C      P(7) = A*(0.5 + 2.0*Z) - Y/6.0*(2.0*Z**3)
C      P(7) = P(7) + Y/6.0*(8.0 + 5.0*Z + 4.0*Z**2)
C      P(8) = A*(-1.0 + 2.0*Z) + Y*(2.0 - Z)
C
C      P(9) = A*(1.0 + 2.0*Z) + Y/3.0*(2.0 + 3.0*Z + 4.0*Z**2)
C      P(11) = P(2)*P(3)
C      P(12) = (A**2)*(0.5 + 4.0*Z**2)
C      P(12) = P(12) + A*Y*Z*(7.0 + 2.0*Z**2)

```

```

P(12) = P(12) + (Y**2)*(2.0 + 5.0*(Z**2)/2.0)
P(13) = TAN(SP/2.0)
C
P(14) = 2.0*Y
P(15) = P(13) - P(14)
P(16) = 2.0*P(1)*Y
P(17) = (P(3))**2 + Y**4
C
P(18) = -P(13)*(A*(1.0+2.0*Z) - Y*Z)
P(19) = P(3) * Y
P(21) = -2.0*(Z + LOG(Y**2))
P(31) = A - Y
C
P(32) = A + Y*(1.0 + 2.0*Z)
P(35) = 2.0 * Y**2
P(36) = P(32)*P(3) + 2.0 * Y**4
P(37) = P(3)*(P(2)-P(3))
P(10) = P(31)*P(5)
C
DO 20, I = 1,37
    P(I) = P(I)*INPI
20 CONTINUE
C
RETURN
END
C
C
C
C *****
C *
C * Subroutine XGET calculates the values of Kussner's X functions *
C * that are required in order to determine the values of the aero- *
C * dynamic coefficients. The routine is called with values of PHI *
C * and PSI, which are related to the distances to the leading edges *
C * of the aileron and tab, and X, which is the array that stores *
C * calculated function values. Other variables used by the routine *
C * include: *
C *
C *      A,B - Reduced angles *
C *      P1..P7 - Certain values of Kussner's Phi functions *
C *      P..T - Precalculated trigonometric functions *
C *      PISQ - Inverse of 3.1415927 squared *
C *****
C
C SUBROUTINE XGET(PHI,PSI,X)
C
C PARAMETER (PI=3.1415927)
C REAL PHI,PSI,X(18),A,B,P1,P2,P3,P4,P5,P6,P7,P,Q,R,S,T,PISQ
C INTEGER I
C
C DO 10, I = 1,18

```

```

      X(I) = 0.0
10  CONTINUE
C
  A = PI - PHI
  B = PI - PSI
  P = SIN(PHI)
  Q = COS(PHI)
  R = SIN(PSI)
  S = COS(PSI)
  T = TAN(PHI/2.0)
C
  P1 = A*(1.0+2.0*Q) + P*(2.0+Q)
  P2 = A + P*Q
  P3 = P*(1.0-Q)
  P4 = R*(1.0+S)
  P5 = B + R*S
  P6 = B*(-1.0+2.0*S) + R*(2.0-S)
  P7 = B*(1.0+2.0*S) + R*(2.0+S)
C
  X(1) = (A+P)*(B-R)
  X(2) = 0.5*(A+P)*P1
  X(3) = 2.0*P*R - 2.0*(Q-S)*ELL(PHI,PSI)
  X(4) = P2*(B+R*(1.0+2.0*S)) + 2*P3*P4 + (Q-S)*X(3)
C
  X(5) = 0.5*(P1-P2)*P5 + XNOT(PHI,PSI)
  X(6) = 0.5*(A+P)*P6
  X(7) = 0.25*P1*P6
  X(8) = P3*(B-R) - 0.5*(Q-S)*X(3)
C
  X(9) = 0.5*P2*P7 - 2.0*XNOT(PHI,PSI)
  X(10) = XTEN(PHI,PSI)
  X(11) = T*(B-R)
  X(12) = 2.0*T*R - 2.0*ELL(PHI,PSI)
C
  X(13) = 2.0*P*(B-R) + 2.0*X(3)
  X(14) = P2*P5 + P3*P4 + 0.5*(Q-S)*X(3)
  X(15) = 0.5*T*P6
  X(16) = T*P5 - 2.0*P*(B-R) + X(3)
C
  X(17) = P*P5 - (2*P*(B-R) + X(3))*(Q-S)
  X(18) = 0.5*(P7-P5)*P2 + XNOT(PSI,PHI)
C
  PISQ = 1.0/(PI**2)
  DO 20, I = 1,18
    X(I) = X(I)*PISQ
20  CONTINUE
C
  RETURN
END
C
C
C

```

```

C *****
C *
C *   Function XTEN calculates the value of Kussner's tenth X
C * constant as a function of the variables PHI and PSI, which are
C * related to the distances from the wing elastic axis to the lead-
C * edges of the aileron and tab. The variables used by the function
C * include:
C *
C *       PHI,PSI - Non-dimensional distances passed to routine
C *       X,Y - Reduced angles
C *       A,B - Pre-calculated function values
C *       XO - Value of the function XNOT
C *
C *****
C
C   FUNCTION XTEN(PHI,PSI)
C
C   PARAMETER (PI=3.1415927)
C   REAL X,Y,Z,A,B,PHI,PSI,XO
C
C   Y = PI - PHI
C   Z = PI - PSI
C   A = COS(PHI)
C   B = COS(PSI)
C   XO = XNOT(PHI,PSI)
C
C   X = Y*Z*(0.5+4.0*A*B)+Y*SIN(PSI)*(3.0*A+0.5*B+A*B**2)-(A-B)*XO
C   X = X + Z*SIN(PHI)*(3.0*B+0.5*A+B*A**2)
C   X = X + SIN(PHI)*SIN(PSI)*(2.0+5.0*A*B/2.0+(A-B)**2)
C
C   XTEN = X/4
C
C   RETURN
C   END
C
C *****
C *
C *   Function XNOT calculates the value of Kussner's zero'th X
C * constant as a function of the variables PHI and PSI. Other
C * variables used by the function include:
C *
C *       PHI,PSI - Nondimensional distances passed to function
C *       Y,Z - Reduced angles
C *       L - Value of the function ELL
C *
C *****
C
C   FUNCTION XNOT(PHI,PSI)
C
C   PARAMETER (PI=3.1415927)
C   REAL X,PHI,PSI,L,Y,Z

```



```

C
  Y = PI - PHI
  Z = PI - PSI
  L = ELL(PHI,PSI)
C
  X = Y*(SIN(PSI)**3) - Z*(SIN(PHI)**3)
  X = X + (COS(PHI)-COS(PSI))*2.0*SIN(PHI)*SIN(PSI)
  X = X - (COS(PHI)-COS(PSI))**3 * L
C
  XNOT = X/3.0
C
  RETURN
  END
C
C
C *****
C *
C *   Function ELL calculates the value of Kussner's L function      *
C *   using the values of PHI and PSI which are passed to the function.*
C *
C *****
C
  FUNCTION ELL(PHI,PSI)
C
  REAL A,B,PHI,PSI
C
  A = PHI + PSI
  B = PHI - PSI
C
  ELL = 0.5*LOG((1.0-COS(A))/(1.0-COS(B)))
C
  RETURN
  END
C
C
C *****
C *
C *   Subroutine MAGORD calculates and normalizes the magnitudes of  *
C *   the eigenvectors of the generalized system as found by the    *
C *   external routine CGEEV. The magnitudes will then be able to be *
C *   used in tracking the modes of the system.                      *
C *
C *****
C
  SUBROUTINE MAGORD(V,MAG,NSIGB,PTE)
C
  COMPLEX V(NSIGB,NSIGB)
  REAL MAG(NSIGB,NSIGB),X,Y,TMP,SMALL
  INTEGER NSIGB,I,J,CT,PTE(NSIGB),ROW
C
  DO 10, I = 1,NSIGB

```

```

      DO 20, J = 1, NSIGB
        X = REAL(V(I,J))
        Y = AIMAG(V(I,J))
        MAG(I,J) = SQRT(X**2 + Y**2)
20      CONTINUE
10      CONTINUE
C
      DO 30, I = 1, NSIGB
        TMP = -100
        DO 40, J = 1, NSIGB
          IF (MAG(I,J).GT.TMP) THEN
            TMP = MAG(I,J)
          END IF
40        CONTINUE
        DO 50, J = 1, NSIGB
          MAG(I,J) = MAG(I,J)/TMP
50        CONTINUE
30      CONTINUE
C
      DO 60, I = 1, NSIGB
        CT = 0
        DO 70, J = 1, NSIGB
          IF (MAG(J,I).EQ.1.0) THEN
            CT = CT + 1
          END IF
70        CONTINUE
        IF (CT.EQ.0) THEN
          SMALL = 100
          DO 80, J = 1, NSIGB
            X = ABS(1.0 - MAG(J,I))
            IF (X.LT.SMALL) THEN
              SMALL = X
              ROW = J
            END IF
80          CONTINUE
          TMP = MAG(ROW,I)
          DO 90, J = 1, NSIGB
            MAG(ROW,J) = MAG(ROW,J)/TMP
90          CONTINUE
        END IF
60      CONTINUE
C
      DO 100, I = 1, NSIGB
        DO 110, J = 1, NSIGB
          IF (MAG(J,I).EQ.1.0) THEN
            PTE(J) = I
          END IF
110        CONTINUE
100      CONTINUE
C
      RETURN
      END

```

```

C
C
C
C *****
C *
C * Subroutine FLTSRCH examines the output file fname.FLT to find *
C * the value of the reduced frequency at which the wing system of *
C * interest will undergo flutter, as given by the point where the *
C * value of the damping parameter crosses from a positive to a *
C * negative value. At this point, the number of the mode that is *
C * undergoing flutter is stored for later use by the program. If *
C * more than one point satisfies the damping requirement for flutter*
C * the subroutine takes the location with the lowest velocity as *
C * the critical point. Variables used by the routine include: *
C *
C * FNAME - Name of the file to be examined *
C * N - Total number of modes used *
C * NUM - Number of k values to be examined *
C * STRT - Lowest k value being considered *
C * I,J - Loop counters *
C * W,V,D - Frequency, velocity and damping storage arrays *
C * WP,WM - Upper and lower bounds of frequency range *
C * X - Variable used in linear interpolation *
C * L - Flutter mode number storage array *
C * COUNT - Number of axis crossings counter *
C * KSTO - Reduced frequency at flutter storage array *
C * VSTO - Flutter velocity storage array *
C * DK - Reduced frequency step size *
C * DGDV - Slope of damping vs. velocity curve at axis *
C * ABOVE,BELOW, *
C * PLUS,MINUS - Logical variables used in setting limits *
C *
C *****
C
C SUBROUTINE FLTSRCH(FNAME,DK,NUM,STRT,N,COUNT,VSTO,L,KSTO,DGDV)
C
C CHARACTER*12 FNAME
C INTEGER N,L(20),I,J,NUM,COUNT
C REAL KSTO(20),W(2,20),V(2,20),STRT,D(2,20),VSTO(20),DK,DGDV(20)
C REAL X,WP,WM
C LOGICAL ABOVE,BELOW,PLUS,MINUS
C
C OPEN(UNIT=1,FILE=FNAME)
C
C * Read initial frequency, velocity and damping values
C
C DO 10, I = 1,N
C READ(1,100) W(1,I),V(1,I),D(1,I)
10 CONTINUE
COUNT = 0
C
C * Loop through the remaining values of the reduced frequency

```

```

C      DO 20, I =1,NUM-1
C
C * Read second of two frequency, velocity and damping values
C
C      DO 30, J = 1,N
C        READ(1,100) W(2,J),V(2,J),D(2,J)
30    CONTINUE
C
C * At each mode, look for evidence of flutter
C
C      DO 40, J = 1,N
C
C * If the damping value is exactly equal to zero, assign that mode
C * and reduced frequency as a location of flutter
C
C        IF (D(2,J).EQ.0) THEN
C          COUNT = COUNT + 1
C          VSTO(COUNT) = V(2,J)
C          L(COUNT) = J
C          KSTO(COUNT) = (NUM-I)*DK + STRT
C          DGDV(COUNT) = (D(2,J)-D(1,J))/(V(2,J)-V(1,J))
C        ELSE
C
C * If the point is not exactly zero, look to see if the damping value
C * currently in storage are above and below zero. If they are, use
C * linear interpolation to find the velocity value
C
C          ABOVE = (D(1,J).GE.0).AND.(D(2,J).LE.0)
C          BELOW = (D(1,J).LE.0).AND.(D(2,J).GE.0)
C          IF (ABOVE.OR.BELOW) THEN
C            WP = W(1,J) + .05 * W(1,J)
C            WM = W(1,J) - .05 * W(1,J)
C            PLUS = W(2,J).LE.WP
C            MINUS = W(2,J).GE.WM
C
C * If the damping criterion was not satisfied as a result of 'mode
C * swapping', then assign the mode and reduced frequency as a
C * flutter location
C
C            IF (PLUS.AND.MINUS) THEN
C              COUNT = COUNT + 1
C              X = (0-D(2,J))/(D(1,J)-D(2,J))
C              VSTO(COUNT) = V(2,J)+X*(V(1,J)-V(2,J))
C              KSTO(COUNT) = (NUM-I+X)*DK + STRT
C              L(COUNT) = J
C              DGDV(COUNT) = (D(2,J)-D(1,J))/(V(2,J)-V(1,J))
C            END IF
C          END IF
C        END IF
40    CONTINUE

```

3

of/de

3

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

```

C
C * Reset the temporary storage arrays
C
      DO 50, J = 1,N
        D(1,J) = D(2,J)
        V(1,J) = V(2,J)
        W(1,J) = W(2,J)
50      CONTINUE
20      CONTINUE
C
100     FORMAT(5X,E10.3,5X,E10.3,5X,E10.3)
C
      CLOSE(1)
C
      RETURN
      END
C
C
C
C *****
C *
C * Subroutine FLTDSP takes the flutter points that are found by
C * the routine FLTSRCH and orders the points on the basis of
C * increasing velocity. The flutter points are then displayed on
C * the screen. The parameters passed to the routine include the
C * storage arrays DG,KS,LC,VS, and the counter CT.
C *
C *****
C
C      SUBROUTINE FLTDSP(CT,VS,LC,KS,DG)
C
C      INTEGER CT,I,J,SPT,LC(20),TL
C      REAL VS(20),TV,DG(20),TD,KS(20),TK,VL
C      CHARACTER*4 TT
C
C      IF (CT.GT.1) THEN
C        DO 70, I = 1,CT
C          TD = DG(I)
C          TV = VS(I)
C          TK = KS(I)
C          TL = LC(I)
C          VL = VS(I)
C          DO 75, J = I,CT
C            IF (VS(J).LT.VL) THEN
C              SPT = J
C              VL = VS(J)
C            END IF
75          CONTINUE
          DG(I) = DG(SPT)
          DG(SPT) = TD
          VS(I) = VS(SPT)
          VS(SPT) = TV

```

```

        KS(I) = KS(SPT)
        KS(SPT) = TK
        LC(I) = LC(SPT)
        LC(SPT) = TL
70      CONTINUE
      END IF
C
      WRITE(*,700) CT
      IF (CT.NE.0) THEN
        WRITE(*,705)
        DO 80, I = 1,CT
          IF (DG(I).LT.0) THEN
            TT = 'S->U'
          ELSE
            TT = 'U->S'
          END IF
          WRITE(*,710) TT,LC(I),KS(I),VS(I),DG(I)
80      CONTINUE
      END IF
C
700  FORMAT(//,5X,'A TOTAL OF ',I2,' CROSSING(S) FCUND:',/)
705  FORMAT(10X,'MODE',10X,'RED. FREQ.',10X,'VELOCITY',10X,'SLOPE',/)
710  FORMAT(3X,A4,4X,I2,13X,F5.3,13X,F7.2,8X,E10.3)
C
      RETURN
      END
C
C
C
C *****
C *
C *   Subroutine CHAREJ allows text of any length to be placed in
C *   the data files used by the program. Text and blank lines must
C *   be preceded by a '!' character. As soon as the subroutine finds
C *   a line that does not start with this character, it backspaces to
C *   the beginning of that line to allow the data to be read.
C *
C *****
C
C   SUBROUTINE CHAREJ
C
C   CHARACTER STRING*80
C
C400  READ(1,500) STRING
C500  FORMAT(A80)
      IF (STRING(1:1).EQ.'!') THEN
        GOTO 400
      ELSE
        BACKSPACE(1)
      END IF
C
      RETURN

```

APPENDIX C
TIME INTEGRATION FORMS OF EQUATIONS OF MOTION

Plunge Equation

The equation of motion for the vertical translation of the wing is

$$\begin{aligned} & [m + \pi \rho b^2] \ddot{h} + K_h h + [S_a - \pi \rho b^3 a] \ddot{\alpha} + \pi \rho b^2 V \dot{\alpha} + \\ & \left[S_\beta + \pi \rho b^3 \left\{ \frac{\phi_4}{2\pi} - (c-e) \frac{\phi_3}{\pi} \right\} \right] \ddot{\beta} + \pi \rho b^2 V \frac{\phi_3}{\pi} \dot{\beta} + \\ & \left[S_\delta + \pi \rho b^3 \left\{ \frac{\psi_4}{2\pi} - (d-f) \frac{\psi_3}{\pi} \right\} \right] \ddot{\delta} + \pi \rho b^2 V \frac{\psi_3}{\pi} \dot{\delta} + 2\pi \rho b V D(t) = 0 \end{aligned}$$

where the Duhamel Integral $D(t)$ is given by

$$\begin{aligned} D(t) = & \int_0^t \left[\Phi \left(\frac{V(t-\tau)}{b} \right) \right] \left[\ddot{h} + b \left(\frac{1}{2} + a \right) \ddot{\alpha} + V \dot{\alpha} + \right. \\ & b \left\{ \frac{\phi_2}{2\pi} - (c-e) \frac{\phi_1}{\pi} \right\} \ddot{\beta} + V \frac{\phi_1}{\pi} \dot{\beta} + b \left\{ \frac{\psi_2}{2\pi} - (d-f) \frac{\psi_1}{\pi} \right\} \ddot{\delta} + \\ & \left. V \frac{\psi_1}{\pi} \dot{\delta} \right] d\tau \end{aligned}$$

Pitch Equation

The equation of motion describing the rotation of the wing about the elastic axis is given by

$$\left[S_\alpha - \pi \rho b^3 a \right] \ddot{h} + \left[I_\alpha + \pi \rho b^4 \left(\frac{1}{8} + a^2 \right) \right] \ddot{\alpha} + \pi \rho b^3 V \left(\frac{1}{2} - a \right) \dot{\alpha} +$$

$$K_\alpha \alpha + \left[I_\beta + S_\beta b(c-a) + \pi \rho b^3 M_\beta \right] \ddot{\beta} + \pi \rho b^3 M_\beta \dot{\beta} +$$

$$\pi \rho b^2 V^2 \frac{\phi_5}{\pi} \beta + \pi \rho b^2 V^2 \frac{\psi_5}{\pi} \delta + \left[I_\delta + S_\delta b(d-a) + \pi \rho b^3 M_\delta \right] \ddot{\delta} +$$

$$\pi \rho b^3 M_\delta \dot{\delta} - 2\pi \rho b^2 (0.5+a) V D(t) = 0$$

where

$$M_\beta = b \left\{ \frac{\phi_7}{4\pi} - (c-e) \frac{\phi_6}{4\pi} - (0.5+a) \frac{\phi_4}{2\pi} + (c-e)(0.5+a) \frac{\phi_3}{\pi} \right\}$$

$$M_\beta = V \left\{ \frac{\phi_6}{2\pi} - (c-e) \frac{\phi_5}{\pi} - (0.5+a) \frac{\phi_3}{\pi} \right\}$$

$$M_\delta = b \left\{ \frac{\psi_7}{4\pi} - (d-f) \frac{\psi_6}{4\pi} - (0.5+a) \frac{\psi_4}{2\pi} + (d-f)(0.5+a) \frac{\psi_3}{\pi} \right\}$$

$$M_\delta = V \left\{ \frac{\psi_6}{2\pi} - (d-f) \frac{\psi_5}{\pi} - (0.5+a) \frac{\psi_3}{\pi} \right\}$$

Aileron Rotation Equation

The equation of motion describing the rotation of the aileron about its hinge line is given by

$$\left[S_{\beta} + \pi \rho b^3 T_{\dot{\beta}} \right] \ddot{\beta} + \left[I_{\beta} + S_{\beta} b(c-a) + \pi \rho b^3 T_{\ddot{\alpha}} \right] \ddot{\alpha} + \pi \rho b^3 T_{\dot{\alpha}} \dot{\alpha} +$$

$$\left[I_{\beta} + \pi \rho b^3 T_{\ddot{\beta}} \right] \ddot{\beta} + \pi \rho b^3 T_{\dot{\beta}} \dot{\beta} + \left[K_{\beta} + \pi \rho b^3 T_{\beta} \right] \beta +$$

$$\left[I_{\delta} + S_{\delta} b(d-c) + \pi \rho b^3 T_{\dot{\delta}} \right] \ddot{\delta} + \pi \rho b^3 T_{\dot{\delta}} \dot{\delta} + \pi \rho b^3 T_{\delta} \delta +$$

$$2\pi \rho b^2 V \left\{ \frac{\phi_8}{2\pi} - (c-e) \frac{\phi_{31}}{\pi} \right\} D(t) = 0$$

where

$$T_{\dot{\beta}} = \frac{\phi_4}{2\pi} - (c-e) \frac{\phi_3}{\pi}$$

$$T_{\ddot{\alpha}} = b \left\{ \frac{\phi_7}{4\pi} - (0.5+a) \frac{\phi_4}{2\pi} - (c-e) \frac{\phi_6}{4\pi} + (0.5+a)(c-e) \frac{\phi_3}{\pi} \right\}$$

$$T_{\ddot{\alpha}} = V \left\{ \frac{\phi_9}{2\pi} - (c-e) \frac{\phi_{32}}{\pi} \right\}$$

$$T_{\ddot{\beta}} = b \left\{ \frac{\phi_{12}}{4\pi^2} - (c-e) \frac{\phi_{37}}{\pi^2} + (c-e)^2 \frac{\phi_{17}}{\pi^2} \right\}$$

$$T_{\mathfrak{p}} = V \left\{ \frac{\phi_{11}}{2\pi^2} - (c-e)\frac{\phi_{10}}{\pi^2} - (c-e)\frac{\phi_{36}}{\pi^2} + (c-e)^2\frac{\phi_{35}}{\pi^2} \right\}$$

$$T_{\mathfrak{p}} = \frac{V^2}{b} \left\{ \frac{\phi_{10}}{\pi^2} - (c-e)\frac{\phi_{35}}{\pi^2} \right\}$$

$$T_{\mathfrak{d}} = b \left\{ \frac{X_{10}(\psi, \phi)}{\pi^2} - (d-f)\frac{X_{18}(\psi, \phi)}{\pi^2} - (c-e)\frac{X_4(\psi, \phi)}{\pi^2} + (c-e)(d-f)\frac{X_3(\psi, \phi)}{\pi^2} \right\}$$

$$T_{\mathfrak{d}} = V \left\{ \frac{X_9(\psi, \phi)}{\pi^2} - (d-f)\frac{X_8(\psi, \phi)}{\pi^2} - (c-e)\frac{X_4(\psi, \phi)}{\pi^2} + (c-e)(d-f)\frac{X_3(\psi, \phi)}{\pi^2} \right\}$$

$$T_{\mathfrak{e}} = \frac{V^2}{b} \left\{ \frac{X_8(\psi, \phi)}{\pi^2} - (c-e)\frac{X_3(\psi, \phi)}{\pi^2} \right\}$$

Control Tab Rotation Equation

The equation of motion describing the rotation of the control tab about its hinge line is given by

$$\left[S_{\delta} + \pi \rho b^3 Q_{\dot{\delta}} \right] \ddot{h} + \left[I_{\delta} + S_{\delta} b(d-a) + \pi \rho b^3 Q_{\ddot{\alpha}} \right] \ddot{\alpha} + \pi \rho b^3 Q_{\dot{\alpha}} \dot{\alpha} +$$

$$\left[I_{\delta} + S_{\delta} b(d-c) + \pi \rho b^3 Q_{\ddot{\beta}} \right] \ddot{\beta} + \pi \rho b^3 Q_{\dot{\beta}} \dot{\beta} + \pi \rho b^3 Q_{\beta} \beta +$$

$$\left[I_{\delta} + \pi \rho b^3 Q_{\dot{\delta}} \right] \ddot{\delta} + \pi \rho b^3 Q_{\dot{\delta}} \dot{\delta} + \left[K_{\delta} + \pi \rho b^3 Q_{\delta} \right] \delta +$$

$$2\pi \rho b^2 V \left\{ \frac{\psi_8}{2\pi} - (d-f) \frac{\psi_{31}}{\pi} \right\} D(t) = 0$$

where

$$Q_{\dot{\delta}} = \frac{\psi_4}{2\pi} - (d-f) \frac{\psi_3}{\pi}$$

$$Q_{\ddot{\alpha}} = b \left\{ \frac{\psi_7}{4\pi} - (0.5+a) \frac{\psi_4}{2\pi} - (d-f) \frac{\psi_6}{4\pi} + (0.5+a)(d-f) \frac{\psi_3}{\pi} \right\}$$

$$Q_{\dot{\alpha}} = V \left\{ \frac{\psi_9}{2\pi} - (d-f) \frac{\psi_{32}}{\pi} \right\}$$

$$Q_{\delta} = b \left\{ \frac{\psi_{12}}{4\pi^2} - (d-f) \frac{\psi_{37}}{\pi^2} + (d-f)^2 \frac{\psi_{17}}{\pi^2} \right\}$$

$$Q_{\delta} = V \left\{ \frac{\psi_{11}}{2\pi^2} - (d-f) \frac{\psi_{10}}{\pi^2} - (d-f) \frac{\psi_{36}}{\pi^2} + (d-f)^2 \frac{\psi_{35}}{\pi^2} \right\}$$

$$Q_{\delta} = \frac{V^2}{b} \left\{ \frac{\psi_{10}}{\pi^2} - (d-f) \frac{\psi_{35}}{\pi^2} \right\}$$

$$Q_{\beta} = b \left\{ \frac{X_{10}(\phi, \psi)}{\pi^2} - (c-e) \frac{X_{18}(\phi, \psi)}{\pi^2} - (d-f) \frac{X_4(\phi, \psi)}{\pi^2} + (c-e)(d-f) \frac{X_3(\phi, \psi)}{\pi^2} \right\}$$

$$Q_{\beta} = V \left\{ \frac{X_9(\phi, \psi)}{\pi^2} - (c-e) \frac{X_8(\phi, \psi)}{\pi^2} - (d-f) \frac{X_4(\phi, \psi)}{\pi^2} + (c-e)(d-f) \frac{X_3(\phi, \psi)}{\pi^2} \right\}$$

$$Q_{\beta} = \frac{V^2}{b} \left\{ \frac{X_8(\phi, \psi)}{\pi^2} - (d-f) \frac{X_3(\phi, \psi)}{\pi^2} \right\}$$

State Variable Form

The state variable form of the equations of motion is represented by the following

$$[A] \{ \dot{\phi} \} = [B] \{ \phi \} + \{ F \}$$

where

$$\{ \phi \} = \begin{Bmatrix} h \\ \dot{h} \\ \alpha \\ \dot{\alpha} \\ \beta \\ \dot{\beta} \\ \delta \\ \dot{\delta} \\ B_1 \\ B_2 \end{Bmatrix}$$

and

$$\{ F \} = \begin{Bmatrix} 0 \\ F_h \\ 0 \\ F_\alpha \\ 0 \\ F_\beta \\ 0 \\ F_\delta \\ 0 \\ 0 \end{Bmatrix}$$

[A] Matrix

$$\begin{bmatrix}
 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & m+\pi\rho b^2 & 0 & S_\alpha-\pi\rho b^3a & 0 & S_\beta+\pi\rho b^3A & 0 & S_\gamma+\pi\rho b^3C & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & S_\alpha-\pi\rho b^3a & 0 & I_\alpha+\pi\rho b^4 \cdot (0.125+a^2) & 0 & I_\beta+S_\beta b(c-a) + \pi\rho b^3M_\beta & 0 & I_\gamma+S_\gamma b(d-a) + \pi\rho b^3M_\gamma & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & S_\beta+\pi\rho b^3T_\beta & 0 & I_\beta+S_\beta b(c-a) + \pi\rho b^3T_\alpha & 0 & I_\beta+\pi\rho b^3T_\beta & 0 & I_\gamma+S_\gamma b(d-c) + \pi\rho b^3T_\gamma & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & S_\gamma+\pi\rho b^3Q_\gamma & 0 & I_\gamma+S_\gamma b(d-a) + \pi\rho b^3Q_\alpha & 0 & I_\gamma+S_\gamma b(d-c) + \pi\rho b^3Q_\beta & 0 & I_\gamma+\pi\rho b^3Q_\gamma & 0 & 0 & 0 \\
 0 & -1 & 0 & -b(\frac{1}{2}-a) & 0 & -b \cdot B & 0 & -b \cdot D & 1 & 0 & 0 \\
 0 & -1 & 0 & -b(\frac{1}{2}-a) & 0 & -b \cdot B & 0 & -b \cdot D & 0 & 0 & 1
 \end{bmatrix}$$

where

$$A = \frac{\phi_4}{2\pi} - (c-e)\frac{\phi_3}{\pi}$$

$$B = \frac{\phi_2}{2\pi} - (c-e)\frac{\phi_1}{\pi}$$

$$C = \frac{\psi_4}{2\pi} - (d-f)\frac{\psi_3}{\pi}$$

$$D = \frac{\psi_2}{2\pi} - (d-f)\frac{\psi_1}{\pi}$$

[B] Matrix

The positions are given as (row, column)

$$B((2n-1), 2n) = 1 \quad n=1,2,3,4$$

$$B(2,2) = -2\pi\rho bV$$

$$B(2,3) = -2\pi\rho bV^2$$

$$B(2,4) = -2\pi\rho b^2V(1-a)$$

$$B(2,5) = -2\pi\rho bV^2\frac{\phi_1}{\pi}$$

$$B(2,6) = -\pi\rho b^2V\left\{\frac{\phi_3}{\pi} + \frac{\phi_2}{\pi} - 2(c-e)\frac{\phi_1}{\pi}\right\}$$

$$B(2,7) = -2\pi\rho bV^2\frac{\psi_1}{\pi}$$

$$B(2,8) = -\pi\rho b^2V\left\{\frac{\psi_3}{\pi} + \frac{\psi_2}{\pi} - 2(d-f)\frac{\psi_1}{\pi}\right\}$$

$$B(2,9) = 2\pi\rho bV\alpha_1$$

$$B(2,10) = 2\pi\rho b V\alpha_2$$

$$B(4,2) = 2\pi\rho b^2(0.5+a)V$$

$$B(4,3) = 2\pi\rho b^2V^2(0.5+a)$$

$$B(4,4) = \pi\rho b^3Va(1-2a)$$

$$B(4,5) = -\pi\rho b^2V^2\left\{\frac{\phi_5}{\pi} - 2(0.5+a)\frac{\phi_1}{\pi}\right\}$$

$$B(4,6) = -\pi\rho b^3\left\{M_\beta - 2V(0.5+a)\left(\frac{\phi_2}{\pi} - (c-e)\frac{\phi_1}{\pi}\right)\right\}$$

$$B(4,7) = -\pi\rho b^2V^2\left\{\frac{\psi_5}{\pi} - 2(0.5+a)\frac{\psi_1}{\pi}\right\}$$

$$B(4,8) = -\pi\rho b^3\left\{M_\delta - 2V(0.5+a)\left(\frac{\psi_2}{\pi} - (d-f)\frac{\psi_1}{\pi}\right)\right\}$$

$$B(4,9) = -2\pi\rho b^2(0.5+a)V\alpha_1$$

$$B(4,10) = -2\pi\rho b^2(0.5+a)V\alpha_2$$

$$B(6,2) = -X$$

$$B(6,3) = -X \cdot V$$

$$B(6,4) = -\pi\rho b^3T_{\alpha} - Xb(0.5-a)$$

$$B(6,5) = -\pi\rho b^3T_{\beta} - XV\frac{\phi_1}{\pi}$$

$$B(6,6) = -\left\{\pi\rho b^3T_{\beta} + Xb\left(\frac{\phi_2}{2\pi} - (c-e)\frac{\phi_1}{\pi}\right)\right\}$$

$$B(6,7) = -\pi\rho b^3T_{\delta} - XV\frac{\psi_1}{\pi}$$

$$B(6,8) = -\left\{\pi\rho b^3T_{\delta} + Xb\left(\frac{\psi_2}{2\pi} - (d-f)\frac{\psi_1}{\pi}\right)\right\}$$

$$B(6,9) = X\alpha_1$$

$$B(6,10) = X\alpha_2$$

$$B(8,2) = -Y$$

$$B(8,3) = -Y \cdot V$$

$$B(8,4) = -\pi\rho b^3 Q_a - Yb(0.5-a)$$

$$B(8,5) = -\pi\rho b^3 Q_\beta - YV\frac{\phi_1}{\pi}$$

$$B(8,6) = -\left\{\pi\rho b^3 Q_\beta + Yb\left(\frac{\phi_2}{2\pi} - (c-e)\frac{\phi_1}{\pi}\right)\right\}$$

$$B(8,7) = -\pi\rho b^3 Q_\gamma - YV\frac{\psi_1}{\pi}$$

$$B(8,8) = -\left\{\pi\rho b^3 Q_\delta + Yb\left(\frac{\psi_2}{2\pi} - (d-f)\frac{\psi_1}{\pi}\right)\right\}$$

$$B(8,9) = Y\alpha_1$$

$$B(8,10) = Y\alpha_2$$

$$B(9,4) = B(10,4) = V$$

$$B(9,6) = B(10,6) = V\frac{\phi_1}{\pi}$$

$$B(9,8) = B(10,8) = V\frac{\psi_1}{\pi}$$

$$B(9,9) = \frac{\beta_1 V}{b}$$

$$B(10,10) = \frac{\beta_2 V}{b}$$

All other entries are 0

$$X = 2\pi\rho b^2 V \left\{ \frac{\phi_8}{2\pi} - (c-e)\frac{\phi_{31}}{\pi} \right\}$$

$$Y = 2\pi\rho b^2 V \left\{ \frac{\psi_8}{2\pi} - (d-f)\frac{\psi_{31}}{\pi} \right\}$$

APPENDIX D

TIME INTEGRATION SOFTWARE

```

C *****
C *
C *   Program INTGPT implements the time integration of the flutter
C *   equations of motion for a two dimensional wing system. The time
C *   integration scheme used is a five term Adams-Moulton predictor-
C *   corrector method, as given by Boyce and Diprima. The parameter M
C *   defines the number of degrees of freedom of the system (from 2
C *   up to 4) and must be set by the user before the main portion of
C *   the program is compiled. The program uses the same input data
C *   files RUN.DAT and SPRING.DAT that are used by the iterative
C *   procedure BRANCH. However, due to the two dimensional nature of
C *   the time integration equations of motion, the parameters defining
C *   the wing system must be given at only one node.
C *   The variables used by the program are the following:
C *
C *   ARRAYS:
C *
C *   AA      - Matrix containing the A matrix of the equations of
C *             motion
C *   AB      - Array storing the product of the B matrix and the
C *             inverse of the A matrix
C *   ACTUAL  - Two column array storing the values of the variables
C *             at the present and the next time step
C *   AF      - Vector storage of the product of the inverse of the
C *             A matrix and the force vector
C *   BB      - Array storing the elements of the B matrix from the
C *             equations of motion
C *   DI      - Vector defining the initial spring displacements, as
C *             given in SPRING.DAT. Not used in this program.
C *   EPS     - Array giving the convergence criteria for the
C *             iterative procedure. Not used in this program.
C *   FORCE    - Vector storing the forces produced by the restoring
C *             springs in each of the degrees of freedom.
C *   INVA    - Array storing the inverse of the A matrix
C *   K1,K2   - Storage for the two slopes of the bilinear or other-
C *             wise springs in each of the degrees of freedom
C *   PHI     - Storage of Kussner's phi functions relating to the
C *             aileron of the system, if present.
C *   PRE     - Array storing the amount of preload present in each
C *             spring
C *   PRIME   - Five column array storing the derivative values of
C *             each variable at the next, present, and previous
C *             three time steps
C *   PSI     - Storage of Kussner's phi functions calculated using
C *             the geometric properties of the tab, if present
C *   REVX    - Kussner's functions X(PHI,PSI)
C *   S       - Array defining the extent of the nonlinear element
C *             (if present) in each spring
C *   X       - Kussner's functions X(PHI,PSI)
C *
C *   OTHER:
C *

```



```

C *      A      - Non dimensional parameter defining the location of      *
C *              the elastic axis relative to the mid chord point      *
C *      B      - Length of the wing semi-chord                        *
C *      C      - Non dimensional parameter defining the location of      *
C *              the aileron hinge relative to the wing mid chord      *
C *      D      - Non dimensional parameter defining the location of      *
C *              the tab hinge relative to the wing mid chord          *
C *      DELT    - The value of the time step used in the integration    *
C *      DR      - The amplitude of the reference degree of freedom, as  *
C *                given by the file SPRING.DAT. Not used.              *
C *      E      - Non dimensional parameter defining the location of      *
C *              the aileron leading edge relative to the mid chord     *
C *      F      - Non dimensional parameter defining the location of      *
C *              the tab leading edge relative to the wing mid chord    *
C *      FINISH  - User specified ending time for the simulation         *
C *      IA,IB,  *
C *      ID      - Variables defining the mass moment of inertia per     *
C *                unit length of the whole wing, aileron and tab,      *
C *                respectively                                           *
C *      MPL     - The mass per unit length of the wing system          *
C *      RHO     - Variable defining the air density                    *
C *      TIME    - Used to track the elapsed time during the simulation  *
C *      TOT     - Dummy summation variable                             *
C *      VEL     - Freestream air velocity                               *
C *      XA,XB,  *
C *      XD      - Non dimensional parameters, giving the location of    *
C *                the wing, aileron and tab centres of gravity,        *
C *                respectively                                           *
C *      Z      - Dummy summation variable                               *
C *
C *      INTEGERS:
C *
C *      DRMODE  - Reference degree of freedom given by SPRING.DAT. Not *
C *                used in this program.                                *
C *      I,J     - Loop counters                                         *
C *      NSTEP   - Number of steps in the time integration loop         *

```

```

C *
C *****

```

```

C
C

```

```

PROGRAM INTGRT

```

```

C

```

```

PARAMETER (M=2)

```

```

C

```

```

C * Variable declarations

```

```

C

```

```

INTEGER DRMODE,I,J,NSTEP
REAL DI(M,2),EPS(M),K1(M),K2(M),S(M),DR,PRE(M)
REAL A,C,D,E,F,MPL,IA,XA,IB,XB,ID,XD,RHO,B
REAL PRIME(2*M+2,5),ACTUAL(2*M+2,2),VEL,DELT,Z

```

```

      REAL PHI(37),PSI(37),X(18),REVS(18),FINISH,FORCE(2*M+2,1)
      REAL AA(2*M+2,2*M+2),BB(2*M+2,2*M+2),INVA(2*M+2,2*M+2)
      REAL AB(2*M+2,2*M+2),TOT,TIME,AF(2*M+2,1)
C
C * Common block definitions
C
      COMMON/PFUNC/PHI,PSI,X,REVS
      COMMON/DIMEN/A,B,C,D,E,F,XA,XB,XD
      COMMON/MASS/MPL,IA,IB,ID,RHO
C
C * Call routines to load the necessary information from the external
C * files RLJ.DAT and SPRING.DAT
C
      CALL WNGSPC(M)
      CALL SPRING(DI,EPS,K1,K2,S,PRE,DR,DRMODE,M)
C
C * Call routine to prompt user to enter the initial conditions for the
C * system, as well as the air speed and time step magnitude
C
      CALL INITIAL(ACTUAL,M,VEL,DELT,FINISH)
C
C * If the system has three or more degrees of freedom, call a routine
C * to calculate the Kussner geometric functions
C
      IF (M.GE.3) THEN
        CALL PSICLC(M)
      END IF
C
C * Initialize the PRIME array
C
      DO 10, I = 1,2*M+2
        DO 20, J = 1,5
          PRIME(I,J) = 0.0
        20 CONTINUE
      10 CONTINUE
C
C * Call subroutine to calculate the elements of the A matrix
C
      CALL AAFILL(M,AA)
C
C * Use the subroutine RLINVRT to invert the A matrix
C
      CALL RLINVRT(AA,INVA,2*M+2)
C
C * Call routine to calculate the elements of the B matrix
C
      CALL BLOAD(M,BB,VEL)
C
C * Multiply the inverted A matrix and the B matrix
C
      CALL MATMULT(INVA,BB,AB,2*M+2,2*M+2,2*M+2)

```

```

C
      NSTEP = INT(FINISH/DELT)
      OPEN(UNIT=1,FILE='INTEGRTE.OUT')
C
C * Begin time integration loop
C
      DO 40, I = 1,NSTEP
        TIME = I*DELT
C
C * Call routine to calculate the elements of the restoring force
C * vector
C
        CALL FLOAD(M,FORCE,K1,K2,S,PRE,ACTUAL,2)
C
C * Multiply the force vector by the inverted A matrix
C
        CALL MATMULT(INVA,FORCE,AF,2*M+2,2*M+2,1)
C
C * Use the arrays to calculate the derivatives at the present
C * time step
C
        DO 50, J = 1,2*M+2
          TOT = 0.0
          DO 60, L = 1,2*M+2
            TOT = TOT + AB(J,L)*ACTUAL(L,2)
          60      CONTINUE
          PRIME(J,2) = TOT - AF(J,1)
        50      CONTINUE
C
C * Use the predictor formula to estimate the value of the variables
C * at the next time step
C
        DO 70, J = 1,2*M+2
          Z = 55*PRIME(J,2)-59*PRIME(J,3)+37*PRIME(J,4)-9*PRIME(J,5)
          ACTUAL(J,1) = ACTUAL(J,2) + DELT*Z/24.0
        70      CONTINUE
C
C * Repeat the above procedure to correct the estimate using the
C * corrector formula and the estimate of the derivative at the
C * next time step
C
        CALL FLOAD(M,FORCE,K1,K2,S,PRE,ACTUAL,1)
        CALL MATMULT(INVA,FORCE,AF,2*M+2,2*M+2,1)
        DO 80, J = 1,2*M+2
          TOT = 0.0
          DO 90, L = 1,2*M+2
            TOT = TOT + AB(J,L)*ACTUAL(L,1)
          90      CONTINUE
          PRIME(J,1) = TOT - AF(J,1)
        80      CONTINUE
        DO 100, J = 1,2*M+2

```

```

      Z = 9*PRIME(J,1)+19*PRIME(J,2)-5*PRIME(J,3)+PRIME(J,4)
      ACTUAL(J,1) = ACTUAL(J,2) + DELT*Z/24.0
100    CONTINUE
C
C * Print the results to an external file
C
      WRITE(1,*) TIME,(ACTUAL(J,1),J=1,2*M)
C
C * Reset the derivative and variable arrays, ready for the next
C * time step
C
      DO 110, J = 1,2*M+2
        DO 120, L = 5,2,-1
          PRIME(J,L) = PRIME(J,L-1)
120      CONTINUE
        ACTUAL(J,2) = ACTUAL(J,1)
110    CONTINUE
C
C * Repeat loop
C
40    CONTINUE
C
      END
C
C
C
C
C *****
C *
C * Subroutine WNGSPC loads the wing system geometric parameters
C * from the data file RUN.DAT. The data file is the same as that
C * used by the program BRANCH, with the exception that the wing
C * can be specified for only one node, in order to simulate a two
C * dimensional wing. The character variables listed in the variable
C * declaration section are included only to maintain the similarity
C * between the data files; they are not used by the time integration
C * program. The other variables accessed by the subroutine are:
C *
C *   INTEGER:
C *
C *   M      - Defines number of degrees of freedom of the system
C *
C *   REAL:
C *
C *   A      - Non dimensional parameter defining the location of
C *             the elastic axis relative to the wing mid chord
C *   B      - Length of the wing semi chord
C *   C      - Non dimensional parameter defining the location of
C *             the aileron hinge relative to the wing mid chord
C *   D      - Non dimensional parameter defining the location of
C *             the tab hinge relative to the wing mid chord
C *   DUM    - Dummy variable

```

```

C *      E      - Non dimensional parameter defining the location of      *
C *              the aileron leading edge relative to the mid chord      *
C *      F      - Non dimensional parameter defining the location of      *
C *              the tab leading edge relative to the wing mid chord      *
C *      IA,IB,  - Variables giving the mass moment of inertia per unit  *
C *              ID length of the wing, aileron and tab, respectively    *
C *      MPL     - Mass per unit length of the wing system                *
C *      RHO     - Density of the air                                     *
C *      XA,XB,  - Non dimensional parameters defining the location of    *
C *              XD the centres of gravity of the wing, aileron, and     *
C *              respectively                                              *
C *      *****
C
C      SUBROUTINE WNGSPC(M)
C
C * Variable declarations
C
C      CHARACTER*8 FNAME,AGAP*2,TGAP*2,PROTYP*6
C      INTEGER M
C      REAL RHO,B,A,C,D,E,F
C      REAL MPL,IA,XA,IB,XB,ID,XD,DUM
C
C * Common block definitions
C
C      COMMON/DIMEN/A,B,C,D,E,F,XA,XB,XD
C      COMMON/MASS/MPL,IA,IB,ID,RHO
C
C * Open data file and read dummy character variables
C
C      OPEN (UNIT=1,FILE='RUN.DAT',STATUS='OLD')
C      CALL CHAREJ
C      READ(1,100) FNAME
C      CALL CHAREJ
C      READ(1,110) PROTYP
C
C * Read the density and reference chord for this case
C
C      CALL CHAREJ
C      READ(1,120) RHO,B
C      CALL CHAREJ
C      READ(1,130) A,DUM,IA,MPL,XA,DUM,DUM
C
C * If the program requires the information, load the physical data
C * for the aileron into storage
C
C      IF (M.GE.3) THEN
C          CALL CHAREJ
C          READ(1,140) IB,XB,C,E,DUM

```

```

      CALL CHAREJ
      READ(1,150) AGAP,DUM
    END IF
C
C * If the program requires the information, load the physical data
C * for the tab into storage
C
      IF (M.EQ.4) THEN
        CALL CHAREJ
        READ(1,140) ID,XD,D,F,DUM
        CALL CHAREJ
        READ(1,150) TGAP,DUM
      END IF
C
      CLOSE(1)
C
C * Format statements
C
100  FORMAT(2X,A8)
110  FORMAT(2X,A6)
120  FORMAT(2(2X,F6.3))
130  FORMAT(2X,F6.3,2X,F6.3,2X,F9.3,2X,F9.3,3(2X,F6.3))
140  FORMAT(2X,F8.5,4(2X,F6.3))
150  FORMAT(2X,A2,5X,F6.4)
C
      RETURN
      END
C
C
C
C
C *****
C *
C *   Subroutine SPRING loads the characteristics of the modal
C * springs from the external user-defined file SPRING.DAT. Bilinear
C * springs with preload and linear springs can be modelled using
C * the variables used by this routine. The variables used by this
C * routine are:
C *
C *       DI - Deflection of the ith non-linear spring
C *       K1,K2 - Spring stiffnesses
C *       S - Half of K2 amplitude range
C *       PRE - Spring preload amplitude
C *       EPS - Convergence limit for iteration loop
C *       I - Loop counter
C *       M - Number of degrees of freedom at each node
C *       DRMODE - Reference degree of freedom
C *       DR - Reference spring deflection/limit cycle
C *
C *****
C
      SUBROUTINE SPRING(DI, EPS, K1, K2, S, PRE, DR, DRMODE, M)

```

```

C      INTEGER I,M,DRMODE
C      REAL DI(M,2),EPS(M),K1(M),K2(M),S(M),DR,PRE(M)
C      OPEN(1,FILE='SPRING.DAT')
C      CALL CHAREJ
C      READ(1,100) DR,DRMODE
C      CALL CHAREJ
C      DO 10, I = 1,M
C          READ(1,110) K1(I),K2(I),S(I),PRE(I),DI(I,1),EPS(I)
C          DI(I,2) = 0.0
10      CONTINUE
C      100  FORMAT(5X,E10.3,5X,I2)
C      110  FORMAT(6(2X,E10.3))
C      CLOSE(1)
C      RETURN
C      END
C
C
C
C *****
C *
C *   Subroutine CHAREJ allows text of any length to be placed in
C *   the data files used by the program. Text and blank lines must
C *   be preceded by a '!' character. As soon as the subroutine finds
C *   a line that does not start with this character, it backspaces to
C *   the beginning of that line to allow the data to be read.
C *
C *****
C
C      SUBROUTINE CHAREJ
C      CHARACTER STRING*80
C      400  READ(1,500) STRING
C      500  FORMAT(A80)
C      IF (STRING(1:1).EQ.'!') THEN
C          GOTO 400
C      ELSE
C          BACKSPACE(1)
C      END IF
C      RETURN
C      END
C
C
C
C

```

```

C
C *****
C *
C *   Subroutine INITIAL prompts the user of the time integration
C *   program to enter information defining the initial conditions of
C *   the system to be considered. The parameters that are loaded give
C *   the initial displacements and velocities of each of the degrees
C *   of freedom. The variables used by the subroutine are:
C *
C *   ACTUAL - Array storing the values of the variables at the
C *             present and next time steps
C *   DELT   - Size of the time step to be used in the integration
C *   FINISH - User specified ending time for the simulation
C *   I      - Loop counter
C *   M      - Integer defining the number of degrees of freedom of
C *             the system
C *   MODE   - Character array containing descriptions of each of
C *             the degrees of freedom
C *   VEL    - Velocity of the freestream air flow
C *
C *****
C
C
C   SUBROUTINE INITIAL(ACTUAL,M,VEL,DELT,FINISH)
C
C * Variable declarations
C
C   INTEGER M,I
C   REAL ACTUAL(2*M+2,2),VEL,DELT,FINISH
C   CHARACTER*8 MODE(4)
C
C * Define the contents of the mode description array
C
C   MODE(1) = 'PLUNGE'
C   MODE(2) = 'PITCH'
C   MODE(3) = 'AILERON'
C   MODE(4) = 'TAB'
C
C * Prompt the user to enter the initial conditions for each
C * degree of freedom
C
C   DO 10, I = 1,M
C       WRITE(*,100) I,MODE(I)
C       WRITE(*,140)
C       READ(*,*) ACTUAL((2*I-1),2)
C       WRITE(*,130)
C       READ(*,*) ACTUAL((2*I),2)
10  CONTINUE
C
C * Ask the user to enter information regarding the air speed, size
C * of time step and ending time for the simulation
C

```



```

WRITE(*,150)
READ(*,*) VEL
WRITE(*,170)
READ(*,*) DELT
WRITE(*,190)
READ(*,*) FINISH
C
C * Set the initial values of the Duhamel calculation equations
C
      ACTUAL(2*M+1,2) = 0.0
      ACTUAL(2*M+2,2) = 0.0
C
C * Format statements
C
100  FORMAT(///,5X,'MODE ',I1,' : ',A8)
130  FORMAT(/,10X,'ENTER INITIAL MODAL VELOCITY : ',\ )
140  FORMAT(/,10X,'ENTER INITIAL MODAL DEFLECTION : ',\ )
150  FORMAT(///,5X,'ENTER AIR VELOCITY : ',\ )
170  FORMAT(///,5X,'ENTER THE SIZE OF TIME STEP : ',\ )
190  FORMAT(///,5X,'ENTER THE LENGTH OF TIME FOR SIMULATION : ',\ )
C
      RETURN
      END
C
C
C
C *****
C *
C *   Subroutine PSICLC calls the routines which are used to calcul- *
C * ate the necessary Kussner functions. The variables used by the *
C * routine include the geometric parameters A..F, the centre of *
C * gravity locations XA, XB and XD, as well as the following: *
C *
C *   PHI      - Kussner's phi functions evaluated using the aileron *
C *              geometric data *
C *   PSI      - Kussner's phi functions evaluated using the tab *
C *              geometric data *
C *   REVX     - Kussner's functions X(PSI,PHI) *
C *   SPHI,    *
C *   SPSI     - Angles expressing the reduced locations of the *
C *              aileron and tab hinges *
C *   X        - Kussner's functions X(PHI,PSI) *
C *
C *****
C
C
C      SUBROUTINE PSICLC(M)
C
C * Variable declarations
C
      INTEGER M

```

```

      REAL C,E,PSI(37),PHI(37),X(18),REVS(18),SPHI,SPSI
      REAL A,B,D,F,XA,XB,XD
C
C * Common block definitions
C
      COMMON/PFUNC/PHI,PSI,X,REVS
      COMMON/DIMEN/A,B,C,D,E,F,XA,XB,XD
C
C * Call routine to calculate Kussner functions for aileron degree
C * of freedom
C
      CALL PCALC(C,PHI)
C
C * If system requires the use of the tab aerodynamics, the call the
C * necessary routines to calculate the functions PSI and the X
C * functions
C
      IF (M.EQ.4) THEN
        CALL PCALC(E,PSI)
        SPHI = ACOS(-C)
        SPSI = ACOS(-E)
        CALL XGET(SPHI,SPSI,X)
        CALL XGET(SPSI,SPHI,REVS)
      END IF
C
      RETURN
      END
C
C
C
C
C *****
C *
C * Subroutine PCALC calculates the values of Kussner's PHI
C * using equations given by Kussner. The variables in the CALL
C * statement are the reduced hinge location C, and the
C * storage array for the PHI values. Other variables that are
C * used by the routine include:
C *
C * I - Loop counter
C * SP - Related to C; corresponds to Kussner's small PHI
C * A - Temporary variable
C * P - Storage for the values of the PHI functions
C *
C *****
C
      SUBROUTINE PCALC(C,P)
C
      PARAMETER (PI = 3.1415927)
      INTEGER I
      REAL C,SP,A,P(37),Y,Z,INPI
C

```

```

DO 10, I = 1,37
  P(I) = 0.0
10  CONTINUE
C
  SP = ACOS(-C)
  A = PI - SP
  Y = SIN(SP)
  Z = COS(SP)
  INPI = 1.0/PI
C
  P(1) = A + Y
  P(2) = A*(1.0 + 2.0*Z) + Y*(2.0+Z)
  P(3) = A + Y*Z
  P(4) = A*2.0*Z + 2.0*Y/3.0*(2.0 + Z**2)
C
  P(5) = Y*(1.0 - Z)
  P(6) = 2.0*A + 2.0*Y/3.0*(2.0 - Z)*(1.0 + 2.0*Z)
  P(7) = A*(0.5 + 2.0*Z) - Y/6.0*(2.0*Z**3)
  P(7) = P(7) + Y/6.0*(8.0 + 5.0*Z + 4.0*Z**2)
  P(8) = A*(-1.0 + 2.0*Z) + Y*(2.0 - Z)
C
  P(9) = A*(1.0 + 2.0*Z) + Y/3.0*(2.0 + 3.0*Z + 4.0*Z**2)
  P(11) = P(2)*P(3)
  P(12) = (A**2)*(0.5 + 4.0*Z**2)
  P(12) = P(12) + A*Y*Z*(7.0 + 2.0*Z**2)
  P(12) = P(12) + (Y**2)*(2.0 + 5.0*(Z**2)/2.0)
  P(13) = TAN(SP/2.0)
C
  P(14) = 2.0*Y
  P(15) = P(13) - P(14)
  P(16) = 2.0*P(1)*Y
  P(17) = (P(3))**2 + Y**4
C
  P(18) = -P(13)*(A*(1.0+2.0*Z) - Y*Z)
  P(19) = P(3) * Y
  P(21) = -2.0*(Z + LOG(Y**2))
  P(31) = A - Y
C
  P(32) = A + Y*(1.0 + 2.0*Z)
  P(35) = 2.0 * Y**2
  P(36) = P(32)*P(3) + 2.0 * Y**4
  P(37) = P(3)*(P(2)-P(3))
  P(10) = P(31)*P(5)
C
  DO 20, I = 1,37
    P(I) = P(I)*INPI
20  CONTINUE
C
  RETURN
END
C
C

```

```

C
C
C *****
C *
C * Subroutine XGET calculates the values of Kussner's X functions *
C * that are required in order to determine the values of the aero- *
C * dynamic coefficients. The routine is called with values of PHI *
C * and PSI, which are related to the distances to the leading edges *
C * of the aileron and tab, and X, which is the array that stores *
C * calculated function values. Other variables used by the routine *
C * include: *
C *
C *      A,B - Reduced angles *
C *      P1..P7 - Certain values of Kussner's Phi functions *
C *      P..T - Precalculated trigonometric functions *
C *      PISQ - Inverse of 3.1415927 squared *
C *****
C
C SUBROUTINE XGET(PHI,PSI,X)
C
C   PARAMETER (PI=3.1415927)
C   REAL PHI,PSI,X(18),A,B,P1,P2,P3,P4,P5,P6,P7,P,Q,R,S,T,PISQ
C   INTEGER I
C
C   DO 10, I = 1,18
C     X(I) = 0.0
10  CONTINUE
C
C   A = PI - PHI
C   B = PI - PSI
C   P = SIN(PHI)
C   Q = COS(PHI)
C   R = SIN(PSI)
C   S = COS(PSI)
C   T = TAN(PHI/2.0)
C
C   P1 = A*(1.0+2.0*Q) + P*(2.0+Q)
C   P2 = A + P*Q
C   P3 = P*(1.0-Q)
C   P4 = R*(1.0+S)
C   P5 = B + R*S
C   P6 = B*(-1.0+2.0*S) + R*(2.0-S)
C   P7 = B*(1.0+2.0*S) + R*(2.0+S)
C
C   X(1) = (A+P)*(B-R)
C   X(2) = 0.5*(A+P)*P1
C   X(3) = 2.0*P*R - 2.0*(Q-S)*ELL(PHI,PSI)
C   X(4) = P2*(B+R*(1.0+2.0*S)) + 2*P3*P4 + (Q-S)*X(3)
C
C   X(5) = 0.5*(P1-P2)*P5 + XNOT(PHI,PSI)
C   X(6) = 0.5*(A+P)*P6

```

```

      X(7) = 0.25*P1*P6
      X(8) = P3*(B-R) - 0.5*(Q-S)*X(3)
C
      X(9) = 0.5*P2*P7 - 2.0*XNOT(PHI,PSI)
      X(10) = XTEN(PHI,PSI)
      X(11) = T*(B-R)
      X(12) = 2.0*T*R - 2.0*ELL(PHI,PSI)
C
      X(13) = 2.0*P*(B-R) + 2.0*X(3)
      X(14) = P2*P5 + P3*P4 + 0.5*(Q-S)*X(3)
      X(15) = 0.5*T*P6
      X(16) = T*P5 - 2.0*P*(B-R) + X(3)
C
      X(17) = P*P5 - (2*P*(B-R) + X(3))*(Q-S)
      X(18) = 0.5*(P7-P5)*P2 + XNOT(PSI,PHI)
C
      PISQ = 1.0/(PI**2)
      DO 20, I = 1,18
        X(I) = X(I)*PISQ
20    CONTINUE
C
      RETURN
      END
C
C
C
C
C *****
C *
C *   Function XTEN calculates the value of Kussner's tenth X
C *   constant as a function of the variables PHI and PSI, which are
C *   related to the distances from the wing elastic axis to the lead-
C *   edges of the aileron and tab. The variables used by the function
C *   include:
C *
C *       PHI,PSI - Non-dimensional distances passed to routine
C *       X,Y - Reduced angles
C *       A,B - Pre-calculated function values
C *       XO - Value of the function XNOT
C *
C *****
C
      FUNCTION XTEN(PHI,PSI)
C
      PARAMETER (PI=3.1415927)
      REAL X,Y,Z,A,B,PHI,PSI,XO
C
      Y = PI - PHI
      Z = PI - PSI
      A = COS(PHI)
      B = COS(PSI)
      XO = XNOT(PHI,PSI)

```

```

C
X = Y*Z*(0.5+4.0*A*B)+Y*SIN(PSI)*(3.0*A+0.5*B+A*B**2)-(A-B)*X0
X = X + Z*SIN(PHI)*(3.0*B+0.5*A+B*A**2)
X = X + SIN(PHI)*SIN(PSI)*(2.0+5.0*A*B/2.0+(A-B)**2)
C
XTEN = X/4
C
RETURN
END
C
C
C *****
C *
C *   Function XNOT calculates the value of Kussner's zero'th X
C *   constant as a function of the variables PHI and PSI. Other
C *   variables used by the function include:
C *
C *       PHI,PSI - Nondimensional distances passed to function
C *       Y,Z - Reduced angles
C *       L - Value of the function ELL
C *
C *****
C
C   FUNCTION XNOT(PHI,PSI)
C
C   PARAMETER (PI=3.1415927)
C   REAL X,PHI,PSI,L,Y,Z
C
C   Y = PI - PHI
C   Z = PI - PSI
C   L = ELL(PHI,PSI)
C
C   X = Y*(SIN(PSI)**3) - Z*(SIN(PHI)**3)
C   X = X + (COS(PHI)-COS(PSI))*2.0*SIN(PHI)*SIN(PSI)
C   X = X - (COS(PHI)-COS(PSI))**3 * L
C
C   XNOT = X/3.0
C
C   RETURN
C   END
C
C
C *****
C *
C *   Function ELL calculates the value of Kussner's L function
C *   using the values of PHI and PSI which are passed to the function.
C *
C *****
C
C   FUNCTION ELL(PHI,PSI)
C
C   REAL A,B,PHI,PSI

```

```

C
C      A = PHI + PSI
C      B = PHI - PSI
C
C      ELL = .5*LOG((1.0-COS(A))/(1.0-COS(B)));
C
C      RETURN
C      END
C
C
C
C *****
C *
C *   Subroutine AAFILL evaluates the elements of the A matrix used *
C *   in the time integration equations of motion. The variables used *
C *   by the subroutine are the following: *
C *
C *   ARRAYS: *
C *
C *   AA      - Array storing the elements of the A matrix *
C *   PHI     - Storage of Kussner's phi functions calculated using *
C *             the aileron geometric parameters *
C *   PSI     - Storage of Kussner's phi functions calculated using *
C *             the tab geometric parameters *
C *   REVX    - Kussner's functions X(PHI,PSI) *
C *   X       - Kussner's functions X(PHI,PSI) *
C *
C *   OTHERS: *
C *
C *   A..F    - Geometric parameters *
C *   CE,DF,  *
C *   HA      - Intermediate calculation variables *
C *   IA..ID  - Mass moment of inertia values *
C *   --DOT   - Coefficients of the derivative terms, as defined in *
C *             the equations of motion *
C *   MPL     - Mass per unit length of the wing system *
C *   RHO     - Air density value *
C *   XA..XD  - Centre of gravity location parameters *
C *   Y,Z     - Dummy variables *
C *
C *   INTEGERS: *
C *
C *   I,J     - Loop counters *
C *   M       - Defines the number of degrees of freedom of the wing *
C *
C *****
C
C      SUBROUTINE AAFILL(M,AA)
C
C      PARAMETER (PI=3.1415927)

```

```

C
C * Variable declarations
C
      INTEGER M,I,J
      REAL AA(2*M+2,2*M+2),A,B,C,D,E,F,XA,XB,XD,MPL,IA,IB,ID,RHO
      REAL PHI(37),PSI(37),X(18),REVS(18),LEDDOT,MBDOT,TBDOT
      REAL LDDOT,MDDOT,TDDOT,QBDOT,QDDOT,Z,CE,HA,DF,Y
C
C * Common block definitions
C
      COMMON/PFUNC/PHI,PSI,X,REVS
      COMMON/DIMEN/A,B,C,D,E,F,XA,XB,XD
      COMMON/MASS/MPL,IA,IB,ID,RHO
C
C * Calculate the required constants to simplify the element
C * calculations
C
      Y = MPL*B
      Z = PI*RHO*B**3
      CE = C-E
      DF = D-F
      HA = 0.5 + A
C
C * Initialize the A matrix
C
      DO 10, I = 1,2*M+2
        DO 20, J = 1,2*M+2
          IF (I.EQ.J) THEN
            AA(I,J) = 1.0
          ELSE
            AA(I,J) = 0.0
          END IF
        20 CONTINUE
      10 CONTINUE
C
C * Evaluate those terms pertaining to a two degree of freedom
C * wing system
C
      AA(2,2) = MPL + Z/B
      AA(2,4) = Y*XA - Z*A
      AA(4,2) = AA(2,4)
      AA(4,4) = IA + Z*B*(.125+A**2)
      AA(2*M+1,2) = -1.0
      AA(2*M+1,4) = -1.0*B*(0.5-A)
      AA(2*M+2,2) = -1.0
      AA(2*M+2,4) = -1.0*B*(0.5-A)
C
C * For systems with three or more degrees of freedom, evaluate
C * the terms of the A matrix applying to the third degree of
C * freedom
C
      IF (M.GE.3) THEN

```



```

LBDOT = PHI(4)/2.0 - CE*PHI(3)
AA(2,6) = Y*XB + Z*LBDOT
AA(6,2) = AA(2,6)
MBDOT = B*(PHI(7)/4.0-CE*PHI(6)/4.0-HA*PHI(4)/2.0)
MBDOT = MBDOT + B*(CE*HA*PHI(3))
AA(4,6) = IB + Y*XB*B*(C-A) + Z*MBDOT
AA(6,4) = AA(4,6)
TBDOT = B*(PHI(12)/(4.0*PI)-CE*PHI(37)/PI+CE**2*PHI(17)/PI)
AA(6,6) = IB + Z*TBDOT
AA(2*M+1,6) = -1.0*B*(PHI(2)/2.0-CE*PHI(1))
AA(2*M+2,6) = -1.0*B*(PHI(2)/2.0-CE*PHI(1))

```

```

END IF

```

```

C
C * For four degree of freedom systems, evaluate the necessary
C * elements of the A matrix
C

```

```

IF (M.EQ.4) THEN
LDDOT = PSI(4)/2.0 - DF*PSI(3)
AA(2,8) = Y*XD + Z*LDDOT
AA(8,2) = AA(2,8)
MDDOT = B*(PSI(7)/4.0-DF*PSI(6)/4.0-HA*PSI(4)/2.0)
MDDOT = MDDOT + B*DF*HA*PSI(3)
AA(4,8) = ID + Y*XD*B*(D-A) + Z*MDDOT
AA(8,4) = AA(4,8)
TDDOT = B*(REX(10)-DF*REX(18)-CE*REX(5)+CE*DF*REX(14))
AA(6,8) = ID + Y*XD*B*(D-C) + Z*TDDOT
QBDOT = B*(X(10)-CE*X(18)-DF*X(5)+DF*CE*X(14))
AA(8,6) = ID + Y*XD*B*(D-C) + Z*QBDOT
QDDOT = B*(PSI(12)/4.0-DF*PSI(37)/PI+DF**2*PSI(17)/PI)
AA(8,8) = ID + Z*QDDOT
AA(M*2+1,8) = -1.0*B*(PSI(2)/2.0-DF*PSI(1))
AA(M*2+2,8) = -1.0*B*(PSI(2)/2.0-DF*PSI(1))

```

```

END IF

```

```

C
C RETURN
C END

```

```

C
C
C
C *****
C *

```

```

C * Subroutine RLINVRT finds the inverse of a square real matrix
C * of general size. The routine accepts two arrays from the calling
C * routine, the matrix to be inverted, and an array that will con-
C * tain the inverted matrix. The variables used in the routine are
C * the following:
C *

```

```

C * TMP - Array holding matrix to be inverted
C * ID - Array for inverted matrix
C * X - Temporary storage variable

```

```

C *           N - Size of matrices                      *
C *           I,J,L - Loop counters                      *
C *           *                                         *
C *****
C
C      SUBROUTINE RLINVRT(TMP,ID,N)
C
C      INTEGER N,I,J,L
C      REAL TMP(N,N),ID(N,N),X
C
C * Initialize the inverted array as an identity matrix
C
C      DO 10, I = 1,N
C        DO 20, J = 1,N
C          IF (I.EQ.J) THEN
C            ID(I,J) = 1.0
C          ELSE
C            ID(I,J) = 0.0
C          END IF
C        CONTINUE
C      20 CONTINUE
C      10 CONTINUE
C
C * Invert the matrix using simple Gauss-Jordan elimination
C
C      DO 70, I = 1,N
C        X = TMP(I,I)
C        DO 80, J = 1,N
C          TMP(I,J) = TMP(I,J)/X
C          ID(I,J) = ID(I,J)/X
C        80 CONTINUE
C        DO 90, J = 1,N
C          IF (I.NE.J) THEN
C            X = TMP(J,I)
C            DO 100, L = 1,N
C              TMP(J,L) = TMP(J,L) - X*TMP(I,L)
C              ID(J,L) = ID(J,L) - X*ID(I,L)
C            100 CONTINUE
C          END IF
C        90 CONTINUE
C      70 CONTINUE
C
C      RETURN
C      END
C
C
C
C
C *****
C *
C * Subroutine BLOAD calculates the required elements of the B
C * matrix given in the equations of motion for the time integration
C * of the two dimensional wing. The variables used by the routine

```

```

C * include the following:
C *
C *   ARRAYS:
C *
C *   BB      - Array storing the elements of the B matrix
C *   PHI     - Storage of Kussner's phi functions calculated using
C *             the aileron geometric parameters
C *   PSI     - Storage of Kussner's phi functions calculated using
C *             the tab geometric parameters
C *   REVX    - Kussner's functions X(PHI,PSI)
C *   X       - Kussner's functions X(PHI,PSI)
C *
C *   OTHERS:
C *
C *   A..F    - Geometric parameters
C *   A1,A2,
C *   B1,B2  - Constants defining the two term expansion of the
C *             Duhamel integral
C *
C *   CE,DF,
C *   HA     - Intermediate calculation variables
C *   CHI    - Intermediate storage variable
C *   IA..ID  - Mass moment of inertia values
C *   --DOT  - Coefficients of the derivative terms, as defined in
C *             the equations of motion
C *   MPL    - Mass per unit length of the wing system
C *   RHC    - Air density value
C *   UPS    - Intermediate calculation variable
C *   VEL    - Freestream air velocity
C *   W1,W2  - Dummy variables
C *   XA..XD  - Centre of gravity location parameters
C *   Y,Z    - Dummy variables
C *
C *   INTEGERS:
C *
C *   I,J     - Loop counters
C *   M       - Defines the number of degrees of freedom of the wing
C *
C *****
C
C   SUBROUTINE BLOAD(M,BB,VEL)
C
C   PARAMETER (PI=3.1415927)
C
C * Variable declarations
C
C   INTEGER M,I,J
C   REAL BB(2*M+2,2*M+2),VEL,A,B,C,D,E,F,XA,XB,XD
C   REAL MPL,IA,IB,ID,RHO,PHI(37),PSI(37),X(18),REVS(18)
C   REAL MBDOT,MDDOT,TADOT,TBDOT,TDDOT,QADOT,QBDOT,QDDOT,Z
C   REAL CE,DF,HA,TD,QB,B1,B2,A1,A2,CHI,UPS,W1,W2

```

```

C
C * Common block definitions
C
COMMON/PFUNC/PHI,PSI,X,REVS
COMMON/DIMEN/A,B,C,D,E,F,XA,XB,XD
COMMON/MASS/MPL,IA,IB,ID,RHO
C
C * Initialize the B matrix
C
DO 10, I = 1,2*M+2
DO 20, J = 1,2*M+2
BB(I,J) = 0.0
20 CONTINUE
10 CONTINUE
C
C * Set constants used to define the Duhamel integral
C
A1 = .165
A2 = .335
B1 = .041
B2 = .320
C
C * Calculate constants used in the calculation of the array elements
C
Z = PI*RHO*B**2
Y = VEL**2/B
CE = C-E
DF = D-F
HA = 0.5 + A
C
C * Calculate those array elements pertaining to a two degree of
C * freedom, two dimensional wing
C
BB(1,2) = 1.0
BB(2,2) = -2.0*PI*RHO*B*VEL
BB(2,3) = -2.0*PI*RHO*B*VEL**2
BB(2,4) = -2.0*Z*VEL*(1-A)
BB(2,2*M+1) = 2.0*PI*RHO*B*VEL*A1
BB(2,2*M+2) = 2.0*PI*RHO*B*VEL*A2
BB(3,4) = 1.0
BB(4,2) = 2.0*Z*(0.5+A)*VEL
BB(4,3) = 2.0*Z*HA*VEL**2
BB(4,4) = Z*B*VEL*A*(1.0-2.0*A)
BB(4,2*M+1) = -2.0*Z*(0.5+A)*VEL*A1
BB(4,2*M+2) = -2.0*Z*(0.5+A)*VEL*A2
BB(2*M+1,4) = VEL
BB(2*M+2,4) = VEL
BB(2*M+1,2*M+1) = -1.0*B1*VEL/B
BB(2*M+2,2*M+2) = -1.0*B2*VEL/B
C
C * For systems with three or more degrees of freedom, calculate
C * the array elements required for the third degree of freedom

```

C

```

IF (M.GE.3) THEN
  CHI = 2.0*Z*VEL*(PHI(8)/2.0-CE*PHI(31))
  W1 = PHI(2)/2.0-CE*PHI(1)
  BB(2,5) = -2.0*PI*RHO*B*VEL**2*PHI(1)
  BB(2,6) = -1.0*Z*VEL*(PHI(3)+PHI(2)-2.0*CE*PHI(1))
  BB(4,5) = -1.0*Z*VEL**2*(PHI(5)-2.0*(0.5+A)*PHI(1))
  MBDOT = VEL*(PHI(6)/2.0-CE*PHI(5)-HA*PHI(3))
  BB(4,6) = -1.0*Z*B*(MBDOT-2.0*VEL*(0.5+A)*W1)
  BB(5,6) = 1.0
  BB(6,2) = -1.0*CHI
  BB(6,3) = -1.0*CHI*VEL
  TADOT = VEL*(PHI(9)/2.0-CE*PHI(32))
  BB(6,4) = -1.0*(Z*B*TADOT+CHI*B*(0.5-A))
  TB = Y*(PHI(10)/PI-CE*PHI(35)/PI)
  BB(6,5) = -1.0*(Z*B*TB+CHI*VEL*PHI(1))
  TBDOT = VEL*(PHI(11)/(2.0*PI)-CE*PHI(10)/PI-CE*PHI(36)/PI)
  TBDOT = TBDOT + VEL*CE**2*PHI(35)/PI
  BB(6,6) = -1.0*(Z*B*TBDOT+CHI*B*W1)
  BB(6,2*M+1) = CHI*A1
  BB(6,2*M+2) = CHI*A2
  BB(2*M+1,6) = VEL*PHI(1)
  BB(2*M+2,6) = VEL*PHI(1)
END IF

```

C
 C * For a four degree of freedom system, calculate the necessary
 C * array elements
 C

```

IF (M.EQ.4) THEN
  UPS = 2.0*Z*VEL*(PSI(8)/2.0-DF*PSI(31))
  W2 = PSI(2)/2.0-DF*PSI(1)
  BB(2,7) = -2.0*PI*RHO*B*VEL**2*PSI(1)
  BB(2,8) = -1.0*Z*VEL*(PSI(3)+PSI(2)-2.0*DF*PSI(1))
  BB(4,7) = -1.0*Z*VEL**2*(PSI(5)-2.0*(0.5+A)*PSI(1))
  MDDOT = VEL*(PSI(6)/2.0-CE*PSI(5)-HA*PSI(3))
  BB(4,8) = -1.0*Z*B*(MDDOT-2.0*VEL*(0.5+A)*W2)
  TD = Y*(REVS(8)-CE*REVS(3))
  BB(6,7) = -1.0*(Z*B*TD+CHI*VEL*PSI(1))
  TDDOT = VEL*(REVS(9)-DF*REVS(8)-CE*REVS(4)+CE*DF*REVS(3))
  BB(6,8) = -1.0*(Z*B*TDDOT+CHI*B*W2)
  BB(7,8) = 1.0
  BB(8,2) = -1.0*UPS
  BB(8,3) = -1.0*UPS*VEL
  QADOT = VEL*(PSI(9)/2.0-DF*PSI(32))
  BB(8,4) = -1.0*(Z*B*QADOT+UPS*B*(0.5-A))
  QB = Y*(X(8)-DF*X(3))
  BB(8,5) = -1.0*(Z*B*QB+UPS*VEL*PHI(1))
  QBDOT = VEL*(X(9)-DF*X(4))
  BB(8,6) = -1.0*(Z*B*QBDOT+UPS*B*W1)
  QD = Y*(PSI(10)/PI-DF*PSI(35)/PI)
  BB(8,7) = -1.0*(Z*B*QD+UPS*VEL*PSI(1))
  QDDOT = VEL*(PSI(11)/(2.0*PI)-DF*PSI(10)/PI-DF*PSI(36)/PI)

```

```

      QDDOT = QDDOT + VEL*DF**2*PSI(35)/PI
      BB(8,8) = -1.0*(Z*B*QDDOT+UPS*B*W2)
      BB(8,2*M+1) = UPS*A1
      BB(8,2*M+2) = UPS*A2
      BB(2*M+1,8) = VEL*PSI(1)
      BB(2*M+2,8) = VEL*PSI(1)
    END IF

```

```

C
  RETURN
  END

```

```

C
C
C
C
C *****
C *
C * Subroutine MATMULT multiplies two arbitrary real arrays to
C * produce the product array. The variables used in the call to
C * this routine are the following:
C *
C *      X(I,J) - First array
C *      Y(J,L) - Second array
C *      Z(I,L) - Product array
C *      I,J,L - Array dimensions
C *
C * Other variables used are:
C *
C *      SUM - Temporary storage variable
C *      I1,I2,I3 - Loop counters
C *
C *****
C
C   SUBROUTINE MATMULT(X,Y,Z,I,J,L)
C
C   REAL X(I,J),Y(J,L),Z(I,L),SUM
C   INTEGER I,J,L,I1,I2,I3
C
C   DO 10, I1 = 1,I
C     DO 20, I2 = 1,L
C       SUM = 0.0
C       DO 30, I3 = 1,J
C         SUM = SUM + X(I1,I3)*Y(I3,I2)
C30      CONTINUE
C       Z(I1,I2) = SUM
C20      CONTINUE
C10     CONTINUE
C
C   RETURN
C   END
C
C
C

```

```

C
C *****
C *
C * Subroutine FLOAD calculates the elements of the force vector
C * used in the time integral equations of motion. The forces are
C * calculated explicitly, rather than by using a describing
C * function, and the equations used are valid for both linear and
C * bilinear springs. The variables used by the routine are:
C *
C * ACTUAL - Storage for the variable values at two time steps
C * D - Dummy variable representing the spring deflection
C * FORCE - Array containing the elements of the force vector
C * K1,K2 - Spring stiffness storage arrays
C * LOAD - Variable defining the force in each spring
C * PRE - Variable defining the amount of preload present in
C * each spring
C * S - Defines the extent of the nonlinearity (if present)
C * in each spring
C * M - Defines the number of degrees of freedom present in
C * the system
C * I - Sets the time step value to be used in calculating
C * the spring stiffnesses
C * J - Loop counter
C *
C *****
C
C
C SUBROUTINE FLOAD(M, FORCE, K1, K2, S, PRE, ACTUAL, I)
C
C * Variable declarations
C
C INTEGER M, I, J
C REAL FORCE(2*M+2, 1), K1(M), K2(M), S(M), PRE(M), ACTUAL(2*M+2, 2)
C REAL LOAD, D
C
C * Initialize the force vector
C
C DO 10, J = 1, 2*M+2
C FORCE(J, 1) = 0.0
10 CONTINUE
C
C * Depending upon the magnitude of the deflection of the degree
C * of freedom, calculate the restoring force generated by each
C * spring
C
C DO 20, J = 1, M
C D = ACTUAL((2*J-1), I)
C IF (D.LT.PRE(J)) THEN
C LOAD = K1(J)*D
C ELSEIF ((D.GE.PRE(J)).AND.(D.LT.(PRE(J)+2.0*S(J)))) THEN
C LOAD = K1(J)*PRE(J)+K2(J)*(D-PRE(J))
C ELSE

```

```
          LOAD = K1(J)*(D-2.0*S(J))+K2(J)*2.0*S(J)
        END IF
        FORCE((2*J),1) = LOAD
20      CONTINUE
C
      RETURN
      END
```


END

1 2 0 4 9 1

FIN