NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

LINEAR MODELING OF ROTORCRAFT
FOR
STABILITY ANALYSIS AND PRELIMINARY DESIGN

by

Walter M. Wirth, Jr.

September, 1993

Thesis Advisor: E. Roberts Wood

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1993, September.	Master's Thesis

4. TITLE AND SUBTITLE LINEAR MODELING OF ROTORCRAFT FOR STABILITY ANALYSIS AND PRELIMINARY DESIGN

5. FUNDING NUMBERS

6. AUTHOR(S) Walter M. Wirth, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(maximum 200 words)*

This thesis investigates linear state space modeling of single main rotor helicopters culminating in a computer program that can be used for 1) stability and control analysis for any single main rotor helicopter or 2) preliminary design of a helicopter. The trim solution for a flight condition is found, the aircraft is perturbed about the nominal point, and the stability and control derivatives are determined. State space models and analysis tools are provided by the program. A notional attack helicopter designed for the 1993 American Helicopter Society Design Competition and a notional utility helicopter are used as examples.

14. SUBJECT TERMS Stability, Control, Design, Rotorcraft, Linear, Modeling	15. NUMBER OF PAGES 191
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

Linear Modeling of Rotorcraft
for
Stability Analysis and Preliminary Design

by

Walter M. Wirth, Jr.
Major, United States Army
B.S., United States Military Academy, 1981

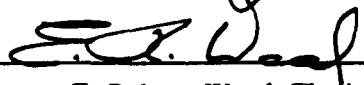
Submitted in partial fulfillment
of the requirements for the degree of

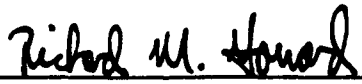
MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

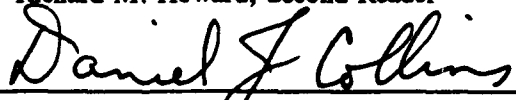
from the

NAVAL POSTGRADUATE SCHOOL
September 1993

Author: 
Walter M. Wirth, Jr.

Approved by: 
E. Roberts Wood, Thesis Advisor


Richard M. Howard, Second Reader


Daniel J. Collins, Chairman
Department of Aeronautical Engineering

ABSTRACT

This thesis investigates linear state space modeling of single main rotor helicopters culminating in a computer program that can be used for 1) stability and control analysis for any single main rotor helicopter or 2) preliminary design of a helicopter. The trim solution for a flight condition is found, the aircraft is perturbed about the nominal point, and the stability and control derivatives are determined. State space models and analysis tools are provided by the program. A notional attack helicopter designed for the 1993 American Helicopter Society Design Competition and a notional utility helicopter are used as examples.

TABLE OF CONTENTS

I. INTRODUCTION

A. NATURE OF THE PROBLEM

" Life is too short to worry about helicopter dynamics."

Louis V. Schmidt
Professor Emeritus
Naval Postgraduate School

Helicopter dynamics, because of the complexity added by the high-mass, rotational components of the aircraft, namely the main and tail rotors, is often shunned by the aerodynamicist. The fact that main and tail rotors are the dominant contributors to flight dynamics makes the situation even worse. The understanding of flight mechanics, stability, and control is difficult enough without this added twist. As a result, courses in flight mechanics focus on solutions for fixed wing aircraft.

As will be shown, the flight mechanics of a helicopter are essentially no different than those of a fixed wing aircraft. The analysis of a helicopter in flight depends on solving the equations of motion, just like for a fixed wing aircraft. The key difference is the realization of a trimmed solution for the helicopter, which is an iterative process vice a straight forward, closed form solution as for the fixed wing aircraft. Once the trim solution is found, the evaluation of stability and control derivatives for the helicopter is done the same as for a fixed wing aircraft except that the derivatives for the main and tail rotors must also be evaluated.

An attempt was made in the preparation of this paper to be as consistent with fixed wing stability notation as possible. The large number of helicopter parameters which are unique or differ from fixed wing notation are listed in Appendix A. Additionally, the symbol R is used for rolling moment instead of the more common L to be consistent with Ray Prouty's notation, common to helicopter literature.

B. BACKGROUND

This project is the result of two needs at the Naval Postgraduate School.

1. MODELING OF ROTORCRAFT

Modeling of rotorcraft needs to be performed because the school is pursuing the acquisition of aircraft for flight testing of an auto-land helicopter system. The key deficiency in the use of simple, off-the-shelf rotorcraft is the lack of design data and theoretical analysis available for control system design. The modeling presented in this thesis is linear. Non-linear modelling can be performed on Flightlab which is available at the school.

2. SOFTWARE AVAILABILITY

Software available for preliminary helicopter design, such as HESCOMP or CAMRAD, is too complicated for students to use during a one quarter Helicopter Design course. The time required for a student to learn to use available software is prohibitive. The need for user friendly software, based on easily accessible programs such as MATLAB, culminated in two software packages being prepared for preliminary helicopter design. The first package JANRAD Performance [Ref. 1] solves the helicopter trim solution and is used to perform aerodynamic analyses. The second package comes from this study and is used to perform stability and control analysis, but uses the trim solution from JANRAD. Both packages require a working knowledge of MATLAB. MATLAB was used as the base program for these software packages because it is readily available and familiar to students because of its wide use at Naval Postgraduate School.

3. ASSUMPTIONS

The following assumptions were made in the foregoing analysis.

- A person using the program has a working knowledge of MATLAB.
- The aircraft trim condition is in level flight with climb angle (γ_c) equal to zero and no sideslip.

- The main and tail rotors provide the only significant contribution to stability and control at a hover. Contributions of all other airframe components can be neglected at a hover.
- Effects of the separate components of the helicopter contribute linearly to stability and control.
- A single main rotor helicopter is used with a counter-clockwise (as viewed from above) rotating main rotor system.
- The main rotor system is articulated, bearingless or rigid in plane.
- Rotor blades are rigid.
- The tail rotor rotates counter-clockwise as viewed from the left side of the helicopter.
- The tail rotor is a teetering rotor with a delta-3 (pitch-flap coupling) hinge.
- The tail rotor mast is perpendicular to the direction of flight.
- Tail rotor coning can be approximated by using only tail rotor thrust and centrifugal forces.
- The NOTAR™ boom is circular.
- Inflow is uniform.
- The fuselage of the helicopter is a rigid body and has aerodynamic characteristics similar to the Prouty example helicopter [Ref. 2].

II. SOFTWARE OVERVIEW

A. SOFTWARE ARCHITECTURE

The Joint Army/Navy Rotorcraft Analysis and Design program, or JANRAD, consists of two major subroutines with others under development. The first routine, JANRAD Performance, calculates the trim solution and various performance parameters of a helicopter. It is described in detail in Ref. 1.

The second major component is JANRAD Stability, written by the author, which 1) calculates all the stability derivatives of each component of the helicopter, 2) determines the linear state space model at any trimmable point in the forward flight regime, 3) provides the eigenvalues of the plant matrix and plots them in the Argand plane, and 4) determines the open loop transfer functions from control inputs to state outputs of the aircraft.

The basic architecture of the program is in Figure 2.1. The opening menu has the user enter data for performance calculations, then branches to either the performance or stability and control routines as desired. When the user selects the stability routines, additional input data is requested by the program for analysis. Data required for each major aircraft component is delineated under the chapters describing the stability and control solution for the component, and is listed in Appendix F.

Upon data entry, the program performs either a hovering or forward flight stability analysis of the helicopter. For calculational purposes, hovering flight is any velocity less than 20 ft/sec or 12 knots. Each analysis requires a number of perturbations about the nominal flight condition, which is done by performing a perturbation, then by calling the TRIM subroutine to determine the new trim condition. This procedure is described in more detail in following sections.

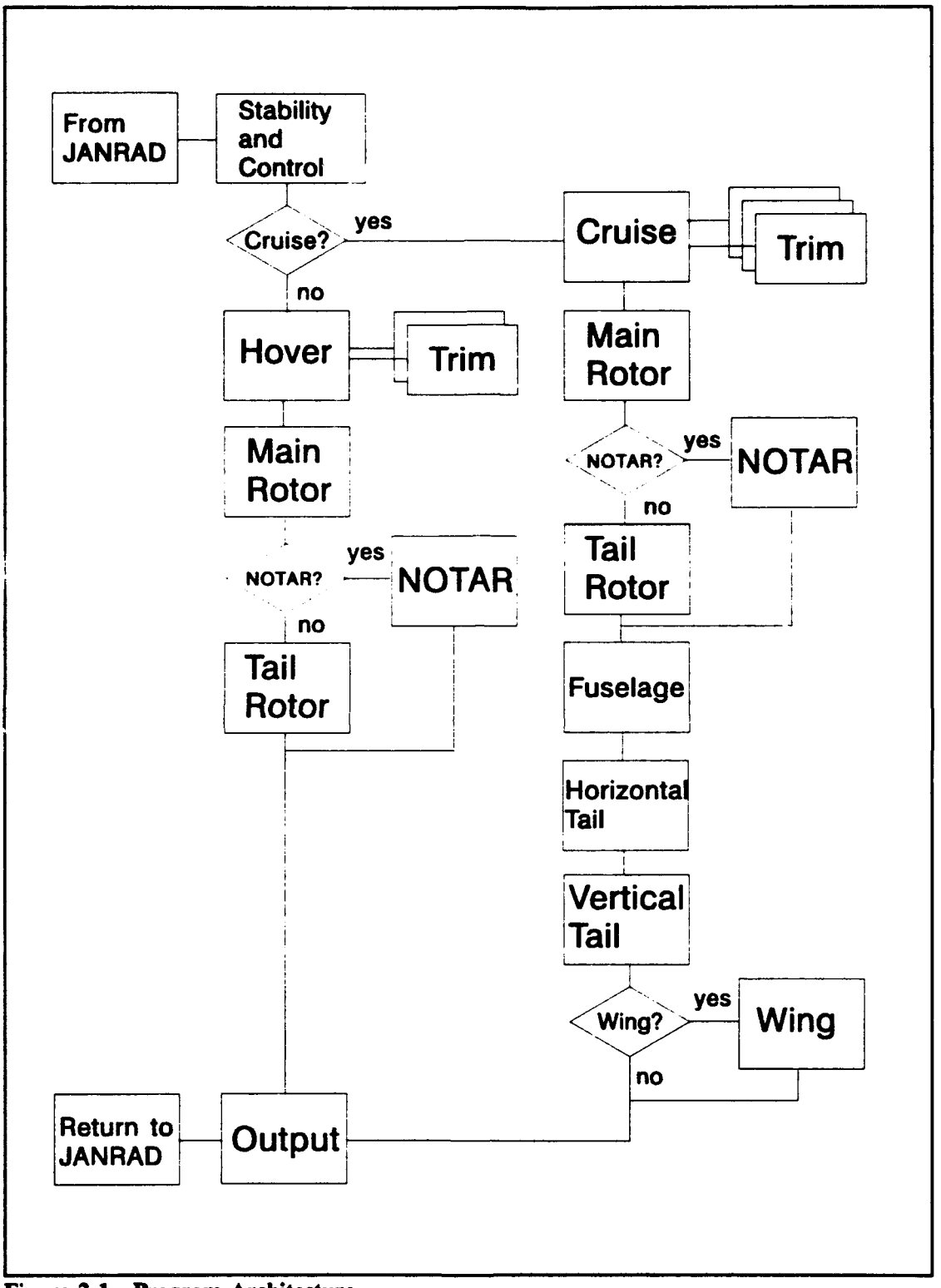


Figure 2.1. Program Architecture.

After the stability and control analysis is performed, the user can select from five different screen outputs depending on his/her needs. Information not available from the output screens can be called from the workspace as described in section III.A.

The linear models provided by the software are saved in a MATLAB data file and can be used to perform simulations of aircraft response due to particular control inputs. This is described in the Software Use, and Examples chapters.

B. MODELLING CONVENTIONS

The coordinate system used in the modelling of the helicopter is the standard set of stability and control axes (Fig. 2.2). The x axis passes through the longitudinal axis of the helicopter with the positive direction to the front of the aircraft. The y axis passes through the lateral axis of the helicopter with the positive direction to the right of the aircraft. The z axis passes through the vertical axis with the positive direction to the bottom of the aircraft, or toward the earth if the aircraft is upright. This set of coordinates is standard in flight mechanics, though it may be a bit confusing initially.

The coordinate system describing the locations of various aircraft components is the same coordinate system used to compute weight and balance for an aircraft. The fuselage station (x axis) determines the longitudinal position of the component, the buttline (y axis) determines the lateral position, and the waterline (z axis) determines the vertical component. The axes are oriented with the x axis parallel to the longitudinal axis of the aircraft with positive direction to the rear, the y axis is parallel to the lateral axis with the positive direction to the right side of the aircraft. The z direction is orthogonal to the x and y axis with the positive direction upward (Fig. 2.2).

Modelling of the helicopter was done in standard state-space format, where the dynamics of the aircraft are described by a system of differential equations, written as the following system of matrix equations.

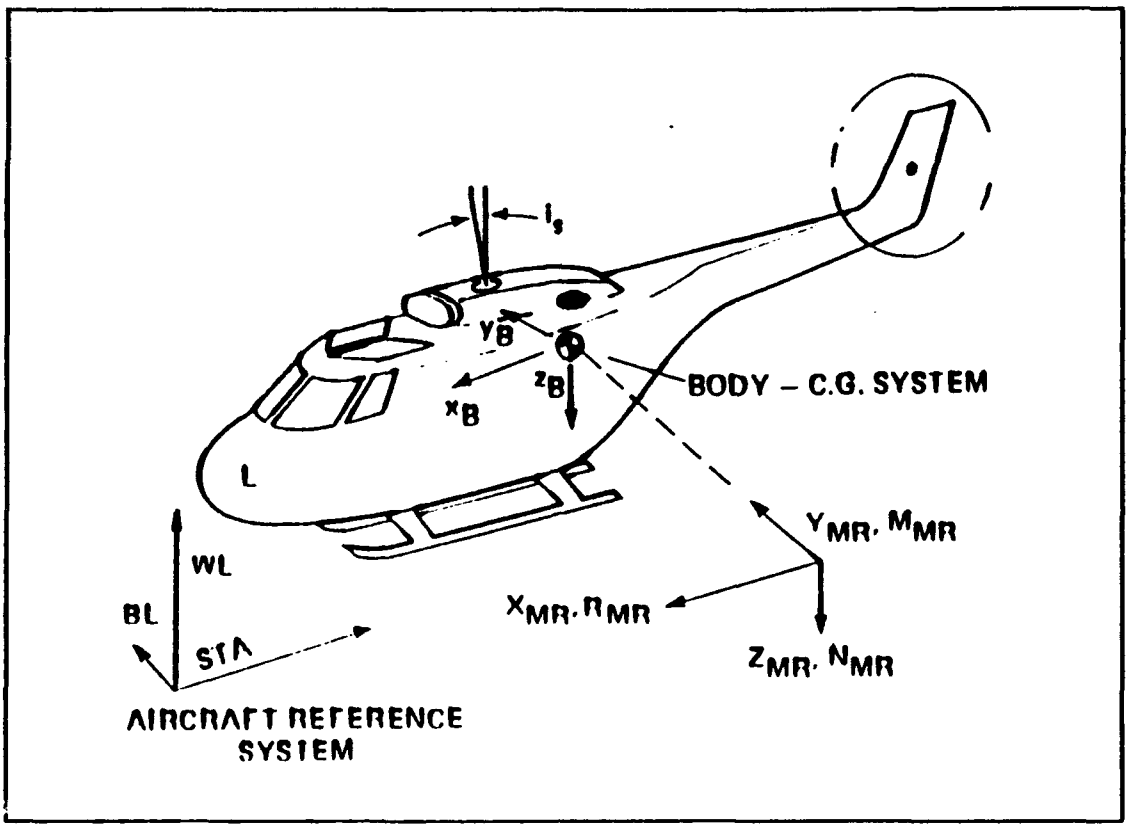


Figure 2.2. Coordinate Systems.

$$\dot{X} = [A]\ X + [B]\ u$$
$$y = [C]\ X + [D]\ u$$

X is the state vector which contains the current state of the system. States are selected which are the outputs of integrators and that describe the motion of the aircraft. For the complete coupled system, the following states were selected per Ref. 3.

u - velocity of the aircraft in the positive x direction

w - velocity of the aircraft in the positive z direction

q - pitch rate of the aircraft

θ - pitch angle of the aircraft

v - velocity of the aircraft in the positive y direction

p - roll rate of the aircraft

Φ - roll angle of the aircraft

r - yaw rate of the aircraft

Ψ - yaw angle of the aircraft

The first four states are used to describe the longitudinal characteristics of the helicopter while the last five are used to describe the lateral-directional characteristics. A fully coupled system is also described using these states.

The vector **u** contains the inputs to the system. In the case of the helicopter, the inputs are

δB1 - change in longitudinal cyclic or elevator

$\delta\theta_{oM}$ - change in collective

δA1 - change in lateral cyclic or aileron

δp - change in anti-torque pedals or rudder

[A] is the plant matrix and describes the dynamics of the aircraft. The plant matrix is determined by using Newton's laws to sum forces and moments about the helicopter at an equilibrium condition. This yields a system of differential equations. The system of equations is written in matrix form where the matrix relating the states **X** with the derivatives of the states is the plant matrix **[A]**. The plant matrix used is described in NASA Technical Memorandum 84281 [Ref. 3].

The **[B]** matrix shows the contributions of the inputs **u** to the system. **[B]** multiplies the input vector **u**, and the result is added to the system. Each element of **[B]** is the partial derivative of the corresponding state with respect to the input. For example B(1,2) would be the partial derivative of u with respect to collective, $\partial u/\partial\theta_{oM}$.

The vector **y** is the output of the system. The desired output is described by using the **[C]** and **[D]** matrices. The matrix **[C]** is used to determine the output of the system from the states **X**. If the values of the state vector are the desired output, as in the case of this paper, then the matrix is an identity matrix the size of the plant matrix; however, the **[C]** matrix does not have to be the identity

matrix and can be set up to provide a wide variety of output. For example, if the desired output is to be lateral acceleration, or $\dot{v}$, then the fifth row of the plant matrix, which corresponds to the equation

$$\dot{v} = \frac{\partial v}{\partial u}\frac{du}{dt} + \frac{\partial v}{\partial w}\frac{dw}{dt} + \frac{\partial v}{\partial \theta}\frac{d\theta}{dt} + \frac{\partial v}{\partial q}\frac{dq}{dt} + \frac{\partial v}{\partial v}\frac{dv}{dt} + \frac{\partial v}{\partial p}\frac{dp}{dt} + \frac{\partial v}{\partial \phi}\frac{d\phi}{dt} + \frac{\partial v}{\partial r}\frac{dr}{dt} + \frac{\partial v}{\partial \psi}\frac{d\psi}{dt}$$

would be used in the **[C]** matrix to provide the output.

The final matrix, **[D]**, is the direct transition matrix for the input. It can be used to provide a direct feed from the input vector **u** to the output vector **y**.

C. THE TRIM SOLUTION

Before a stability and control analysis can be performed on a helicopter, the trim solution for the main and tail rotors must be obtained. The trim solution solves for the collective pitch (θ_{oM}), the thrust moment of the blade (Tm), the first lateral harmonic of blade feathering (A_1), the first longitudinal harmonic of blade feathering (B_1), the rotor coning angle (a_{0M}) and the tip path plane angle (α_{tppM}) of the main rotor. When these values are solved for, the first coefficient of longitudinal blade flapping (a_{1sM}), and the first coefficient of lateral blade flapping (b_{1sM}), are then computed. Additionally, the trim solution for the tail rotor is needed to provide tail rotor collective (θ_{oT}), the first coefficient of longitudinal blade flapping (a_{1sT}), the first coefficient of lateral blade flapping (b_{1sT}), and the tail rotor coning angle (a_{0T}) for stability analysis.

The trim solution for the main rotor is solved by the TRIM routine in JANRAD. The values a_{1sM}, b_{1sM} are then computed in the Stability and Control Routine. The tail rotor solution is approximated using closed form solutions in the Stability and Control Routine.

D. GENERAL PROCEDURE TO EVALUATE STABILITY DERIVATIVES

Stability derivatives are determined by using closed form solutions whenever possible, and by solving multiple trim solutions about a nominal point to solve for unknowns.

By using the TRIM routine in the JANRAD program, the trim solution for the desired flight condition is determined. The helicopter is then perturbed about this nominal condition and retrimmed. From these additional solutions, the basic stability derivatives for the main and tail rotors are solved for.

By solving for the basic stability derivatives first, the helicopter can be described by using the same states as in the a standard linear model of a fixed wing aircraft [u v w p q r Ψ θ Φ]'. Without solving for the intermediate derivatives first, the helicopter would have to be described using additional states for the main rotor, [a_{1sM} b_{1sM} C_T/σ_M C_H/σ_M C_Q/σ_M μ_M λ_M']', and the tail rotor, [a_{1sT} b_{1sT} C_T/σ_T C_H/σ_T C_Q/σ_T μ_T λ_T]'. Aside from the fact that a model described in this fashion is inconsistent with the standard flight mechanic description, the size of the system model becomes large and intimidating. This is part of the reason that helicopter flight mechanics are generally avoided.

After the basic stability derivatives are determined, intermediate derivatives are evaluated. Closed form solutions are used whenever possible to solve intermediate derivatives. After an intermediate derivative is determined, the basic derivative and the intermediate derivative are combined using the chain rule to form the complete stability derivative. An example solution of a main rotor derivative follows.

Wanted: $\left(\frac{\partial X}{\partial \theta_0}\right)_M$

Find from rotor plots or numerically: $\left(\frac{\partial C_H/\sigma}{\partial \theta_{0_M}}\right), \left(\frac{\partial C_T/\sigma}{\partial \theta_{0_M}}\right), \left(\frac{\partial a_{1_s}}{\partial \theta_{0_M}}\right)$

Compute:

$$\left(\frac{\partial C_H/\sigma}{\partial a_{1_s}}\right)_M = C_T/\sigma + \frac{a}{8}\lambda'$$

$$\left(\frac{\partial X}{\partial \theta_0}\right)_M = -\rho A_b (\Omega R)^2 \left[\frac{\partial C_H/\sigma}{\partial \theta_{0_M}} + \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial \theta_{0_M}} + (a_{1_s} + i_M)\frac{\partial C_T/\sigma}{\partial \theta_{0_M}}\right]$$

III. STABILITY AND CONTROL DERIVATIVES

A. MAIN ROTOR CONTRIBUTIONS

The main rotor is the key contributor to helicopter flight dynamics. The strong coupling between the states of the rotor produce complicated responses in many of the aircraft states at the same time. This can be visualized through the following example.

If collective pitch is increased in forward flight, the increased drag in the rotor system increases the amount of torque required from the engine; this results in a yaw of the helicopter. Simultaneously, since the tip path plane is tilted forward, the forward thrust increases which increases the advanced ratio. The increase in advance ratio results in differing amounts of lift on the advancing and retreating sides of the rotor disk which results in a pitching moment because of the phase lag of the blades. At the same time, the pitching moment changes the angle of attack on the fore and aft blades, which changes lateral flapping, which changes the rolling moment. Because of the cross coupling between lateral and longitudinal flapping, there is a change in longitudinal flapping. And so on.

The basic stability derivatives of the main rotor are described in terms of the partial derivatives of the standard helicopter coefficients, a_{1sM}, b_{1sM}, C_T/σ_M, C_H/σ_M, C_Q/σ_M, μ_M, λ', A_{1M}, B_{1M}, and the main rotor blade stiffness

$$\left(\frac{dM}{da_{1_s}}\right)_M = \left(\frac{dR}{db_{1_s}}\right)_M = \frac{\frac{3}{4}\frac{e}{R}A_b\rho R(\Omega R)^2 a}{\gamma}$$

These basic main rotor derivatives are then combined as described in the prior section to form the main rotor stability derivatives. A detailed list of the basic main rotor derivatives and the stability derivatives is available in Appendix C.

To solve for the derivatives, JANRAD uses input data from the performance menu, and output computations from the TRIM section of the program. The input required for the trim solution is found in the JANRAD Performance thesis [Ref. 1]. Output from the performance routine and input from the Stability and Control Routine New File menu (Fig. 3.1) are used to solve for a_{1sM}, b_{1sM} C_T/σ_M, C_H/σ_M, C_Q/σ_M, μ_M, λ', A_1, B_1. Input data can be changed in the Stability and Control Additional Parameters Menu screen 1 (Fig. 3.2).

```
Blade flapping moment of inertia (slug ft^2):
Hub height above reference datum/waterline (ft):
Hub fuselage station (ft):
Hub position right of buttline (ft):
Mast incidence (negative forward - deg):
```

Figure 3.1. Main Rotor New File Menu.

The perturbation solution in conjunction with the equations in Appendix C solve for the partial derivatives of a_{1sM}, b_{1sM} C_T/σ_M, C_H/σ_M, C_Q/σ_M, μ_M, λ', A_1, B_1 with respect to each other, thus completing the evaluation of the basic main rotor derivatives. The CRUISE.M and HOVER.M routines then combine the basic and intermediate derivatives to give a produce a complete set of main rotor derivatives.

Main rotor output is displayed in output screen 2 (Fig. 3.3). Derivatives are stored in the workspace. Because of the large number of derivatives, it is impractical to display all of them on the screen. Instead, key derivatives and control power are displayed in the Key Control Parameters output screen (Fig. 3.4) AFTER all contributions from other aircraft components are determined.

```
          ***   STABILITY AND CONTROL MENU   ***
          *** ADDITIONAL PARAMETERS (1 of 3) ***

 Main Rotor
 1. flapping mom of inertia   2. hub height above waterline
 3. hub fuselage station      4. hub posn right of buttline
 5. mast incidence

 Tail Rotor (enter zeros (0) if using NOTAR)
 6. height above waterline    7. hub fuselage station
 8. posn right of buttline    9. number of blades
10. blade chord              11. blade radius
12. lift curve slope         13. rotational velocity
14. flap mom of inertia      15. delta-3 angle
16. blade twist

 Vertical Fin
17. height above waterline   18. fuselage station
19. posn right of buttline   20. alpha zero lift
21. CL max                   22. dynamic pressure ratio
23. lift curve slope'

0. NO CHANGES
Input the parameter to change:
```

Figure 3.2. Additional Parameters Menu, Screen 1.

```
          *** INPUT DATA CONTINUED (screen 2 of 8) ***
                          filename

                        Main Rotor

                  advance ratio = mu
          inflow parameter wrt TPP = lamp
                 Tip path angle = altpp*57.3 degs
              Rotor coning angle = ao*57.3 degs
        1st lat cyclic term-A1 = A1*57.3 degs
       1st long cyclic term-B1 = B1*57.3 degs
               lateral flapping = b1s*57.3 degs
         longitudinal flapping = a1s*57.3 degs
                   Lock number = lockno
```

Figure 3.3. Main Rotor Calculated Output.

All derivatives can be called from the workspace. This is described in the Output Data Instructions screen 1 (Fig. 3.5). The file VARLIST.TXT is found in Appendix D.

```
              *** KEY CONTROL PARAMETERS ***

      cross coupling = xcouple

                     Designed damping
               pitch = desdmdq ft-lbs/(rad/sec)
                roll = desdrdp ft-lbs/(rad/sec)
                 yaw = desdndr ft-lbs/(rad/sec)

                      Control Power
                pitch = cppitch ft-lbs/in
                roll = cproll ft-lbs/in
                 yaw = cpyaw ft-lbs/in

               Cooper Harper Pilot Ratings
                damping/moment of inertia
pitch (dM/dq)/Iyy = prpitch [ft-lbs/(rad/sec)]/(slug ft^2)
 roll (dR/dp)/Ixx = prroll [ft-lbs/(rad/sec)]/(slug ft^2)
  yaw (dN/dr)/Izz = pryaw [ft-lbs/(rad/sec)]/(slug ft^2)

           control power/moment of inertia
pitch (dM/in)/Iyy = cpipitch (ft-lbs/in)/(slug ft^2)
 roll (dR/in)/Ixx = cpiroll (ft-lbs/in)/(slug ft^2)
  yaw (dN/in)/Izz = cpiyaw (ft-lbs/in)/(slug ft^2)
```

Figure 3.4. Key Control Parameters Output Screen.

```
      *** OUTPUT DATA INSTRUCTIONS (screen 1 of 3) ***

Because this subroutine generates a large number of single
value data not shown on the output screen, a text file
VARLIST.TXT is on this disk which lists the variable names
for all the stability derivatives.  Stability derivative
contributions for all major aircraft components can be found
by reading the text file VARLIST.TXT, then asking MATLAB the
variable name corresponding to the derivative.

Press any key to continue
```

Figure 3.5. Output Data Instructions, Screen 1.

B. TAIL ROTOR CONTRIBUTIONS

Before stability contributions of the tail rotor can be determined, the tail rotor trim solution must be derived. Since only the main rotor trim solution is computed in the performance section of JANRAD, the tail rotor trim solution is solved in the following manner.

By assuming that the shaft axis of the tail rotor is perpendicular to the flight path of the helicopter, the tip path plane angle (α_{TPT}) is equal to longitudinal flapping (a_{1sT}). Additionally, no longitudinal or lateral cyclic pitch is introduced by the control system, hence $B_{1T}=0$ and $A_{1T}=0$. This reduces the inflow parameter with respect to the tip path plane from [Ref. 2:p. 189]

$$\lambda' = \mu \left[\alpha_{TPP} - (B_1 + a_{1_s})\right] - \frac{v_1}{\Omega R}$$

to

$$\lambda = - \frac{v_1}{\Omega R} = -C_T/\sigma \frac{\sigma}{2\mu}$$

Tail rotor thrust T_t is computed by using the main rotor torque from the main rotor trim solution, length of the moment arm to the tail rotor, lift of the vertical tail, and length of the moment arm to the vertical tail. The moment contributions of the tail rotor and vertical tail must equal the torque of the main rotor. This yields

$$T_T = \frac{C_{Q_M} \rho A_M (\Omega_M R_M)^2 - L_v l_v}{l_T}$$

Tail rotor coning is arrived at by using centrifugal forces and tail rotor thrust. This gives the following, where γ is the Lock number of the tail rotor blades [Ref. 2:p 171].

$$a_0 = \frac{2}{3} \gamma \frac{C_T/\sigma}{a} - \frac{\frac{3}{2} g R^2}{(\Omega R)^2}$$

Longitudinal flapping is determined by writing the flapping equations for the tail rotor in terms of a_0, C_T/σ, μ, and the tail rotor pitch-flap coupling angle δ_3. Furthermore, this allows the lateral flapping to be determined in terms of the above values [Ref. 2:p 189].

$$b_{1_s} = 2\left[\frac{\frac{4}{3}\mu a_0 + C_T/\sigma \frac{\sigma}{2\mu}}{2+\mu^2}\right] + a_{1_s}\tan\delta_3$$

$$a_{1_s} = \frac{-2\left[\frac{\frac{4}{3}\mu a_0 + C_T/\sigma \frac{\sigma}{2\mu}}{2+\mu^2}\right]\left[1-\left(\frac{4\mu}{2+3\mu^2}\right)^2\right]\tan\delta_3}{\frac{2-\mu^2}{2+3\mu^2} + \left[1-\left(\frac{4\mu}{2+3\mu^2}\right)^2\right]\tan^2\delta_3}$$

To arrive at θ_{0T}, C_T/σ, μ, tail rotor twist Θ_{1T}, and λ are used yielding [Ref. 2:p 187].

$$\theta_{0_T} = \frac{\frac{4\ C_T/\sigma}{a} - \left(\frac{1}{2} + \frac{\mu^2}{2}\right)\theta_1 + \mu b_{1_s}\tan\delta_3 - \lambda}{\frac{2}{3}+\mu^2} - a_0\tan\delta_3$$

At a hover, there is no longitudinal or lateral flapping in the tail rotor. Because of this, the tail rotor equations can be resolved in terms a_0, Θ_{0T}, Θ_{1T}, ht, lt, and the main rotor trim conditions. This results in the following set of equations for the tail rotor at a hover.

$$a_{1_s} = 0 \quad b_{1_s} = 0$$

$$\frac{C_T}{\sigma} = \frac{a}{4}\left(\frac{v_1}{\Omega R} - \sqrt{\frac{\sigma_M\,(C_T/\sigma)_M}{2}}\right)$$

$$\frac{C_Q}{\sigma} = \frac{C_T\sqrt{\frac{C_T}{2}} + \frac{c_d}{8}}{\sigma}$$

where c_d is approximated from

$$c_d = c_{d_o} + \left(\frac{6C_T/\sigma}{\pi e \frac{c_t}{R_t}}\right)$$

Once a trim solution has been determined, stability derivatives are found in the same manner as for the main rotor, by perturbing about a nominal point. At a hover, the stability derivatives are computed in terms of the main rotor stability derivatives as in Ref. 2.

These equations may produce flapping magnitudes that are lower than more exact solutions, but since the trends of the perturbations are the same in both cases, we can use these trim equations as the basis for solving the basic derivatives of the tail rotor [Ref. 2:p 189].

In addition to the tail rotor parameters input from the JANRAD opening menu, additional parameters are required for the stability and control analysis. Tail rotor inputs are entered in the Stability and Control Routine New File menu (Fig. 3.6) and changed in the Stability and Control Menu Additional Parameters menu, screen 1 (Fig. 3.7).

```
Tail rotor height above reference datum/waterline (ft):
Tail rotor fuselage station (ft):
Tail rotor position right of buttline (ft):
Number of tail rotor blades:
Blade chord (ft):
Tail rotor blade radius (ft):
Average lift curve slope of tail rotor:
Rotational velocity of tail rotor (rad/sec):
Blade flapping moment of inertia (slug ft^2):
Delta-3 angle (deg):
Blade twist (deg):
```

Figure 3.6. Tail Rotor New File Menu.

The program tests to see if an anti-torque device is installed by checking the number of tail rotor blades and the NOTAR™ fan diameter. If both are zero, an error message is displayed (Fig. 3.8).

```
          ***   STABILITY AND CONTROL MENU   ***
          *** ADDITIONAL PARAMETERS (1 of 3) ***

 Main Rotor
 1. flapping mom of inertia   2. hub height above waterline
 3. hub fuselage station      4. hub posn right of buttline
 5. mast incidence

 Tail Rotor (enter zeros (0) if using NOTAR)
 6. height above waterline    7. hub fuselage station
 8. posn right of buttline    9. number of blades
10. blade chord              11. blade radius
12. lift curve slope         13. rotational velocity
14. flap mom of inertia      15. delta-3 angle
16. blade twist

 Vertical Fin
17. height above waterline   18. fuselage station
19. posn right of buttline   20. alpha zero lift
21. CL max                   22. dynamic pressure ratio
23. lift curve slope'

0. NO CHANGES
Input the parameter to change:
```

Figure 3.7. Additional Parameters Menu, Screen 1.

```
You must have a tail rotor or NOTAR/thruster!
```

Figure 3.8. Missing Component Warning.

Tail rotor output is displayed in Input Variable output screen 3 and in the Tail Rotor Calculated Data screen (Fig 3.9).

Tail rotor derivatives are solved for in the same manner as for the main rotor. A detailed listing of tail rotor derivatives is in Appendix C.

C. FUSELAGE CONTRIBUTIONS

The fuselage contributions to stability and control are typically arrived at through empirical methods, so are best estimated initially by referring to characteristics of similar helicopters.

```
*** INPUT DATA CONTINUED (screen 3 of 8) ***
                filename

          Tail rotor (zero if NOTAR)

          Number of blades = bt
               Blade chord = cot ft
              Blade radius = Rt ft
            Lift curve slope = at
        Rotational velocity = ohmt rad/sec
Flapping moment of inertia = Ibt slug ft^2
            Delta-3 angle = delta3 deg
              Blade twist = thetalt deg
Hub height above waterline = htd ft
      Hub fuselage station = ltd ft
Hub position rt of buttline = ytd ft

             *** CALCULATED DATA ***

          Tail Rotor (zero if NOTAR)

         tail rotor thrust = Tt lbs
             advance ratio = mut
          inflow parameter = lampt
        Rotor coning angle = aot*57.3 degs
          lateral flapping = b1st*57.3 degs
longitudinal flapping = a1st*57.3 degs
            Lock number = locknot
```

Figure 3.9. Tail Rotor Output.

Lift and drag moments of the fuselage depend on the angle of attack α_F, and the sideslip angle β_F of the fuselage. Because of the downwash induced by the main rotor, α_F is estimated by

$$\alpha_F = \theta - \gamma_c - \varepsilon_{M_F}$$

where θ is the fuselage pitch attitude, γ_c is the climb angle, and

$$\varepsilon_{M_F} = \frac{v_F}{v_1} \frac{T_M}{4\, q\, A_M}$$

The value v_F/v_1, the fuselage downwash ratio, is the only fuselage stability parameter required to be input into the stability and control program. It can be estimated by methods outlined in Ref. 2. A typical value of 1.5 is recommended otherwise.

The other parameters input with fuselage data are the cg location, and moments of inertia of the aircraft.

Input for the fuselage is made in the Stability and Control Routine New File menu (Fig. 3.10) and changed in the Stability and Control Menu Additional Parameters menu, screen 3 (Fig. 3.11).

```
CG height above reference datum/waterline (ft):
CG Fuselage station (ft):
CG position right of buttline (ft):
Ixx (slug ft^2):
Iyy (slug ft^2):
Izz (slug ft^2):
Ixz (slug ft^2):
Downwash ratio for fuselage (page 513 Prouty):
```

Figure 3.10. Fuselage New File Menu.

```
          ***   STABILITY AND CONTROL MENU   ***
          *** ADDITIONAL PARAMETERS (3 of 3) ***

 CG location and Inertias/fuselage parameters
 1. cg ht. above waterline   2. cg fuselage station
 3. cg posn rt of buttline   4. Ixx
 5. Iyy                      6. Izz
 7. Ixz                      8. fuselage downwash ratio

NOTAR if available (enter zeros if using tail rotor)
 9. height above waterline  10. boom fuselage station
11. boom position left ref  12. NOTAR diameter
13. swirl angle at boom     14. NOTAR max force
15. thruster fuselage station

Rigging
16. B1 main/in defl (del e) 17. A1 main/in defl (dela)
18. theta0m/in defl (del c) 19. theta0t/pedal defl (del r or p)
20. NOTAR sleeve twist/defl 21. max rudder defl

 0. NO CHANGES
 Input the parameter to change:
```

Figure 3.11. Additional Parameters Menu, Screen 3.

Stability derivatives from the fuselage are combined with the stability derivatives of other components for display in the Key Control Parameters output screen. The individual contributions

of the fuselage to stability can be pulled from the workspace by finding the variable name in VARLIST.TXT and calling the value as described in the Main Rotor section.

The values of the basic fuselage stability contributions are set in the program to typical values found in Ref. 2. If one desires to change these values, edit the subroutine CBODYGRP.M lines 17 through 22, substituting desired values for the defaults.

D. HORIZONTAL STABILIZER CONTRIBUTIONS

Contributions of the horizontal tail are computed as for a fixed wing aircraft with one exception. In a helicopter, the downwash created by the rotor is very large, and in nearly all modes of flight, the horizontal tail is in the downwash and is affected.

The effects of this are accounted for by adjusting the dynamic pressure on the horizontal tail by determining the dynamic pressure ratio q_H/q, and downwash angles due to the main rotor ϵ_{MH}, and fuselage ϵ_{FH}. These effects are generally evaluated in some empirical fashion, but some general guidelines follow.

The dynamic pressure at the horizontal and vertical tail is generally lower than free stream due to momentum loss from the fuselage and main rotors. A first guess for q_H/q can be made by using Figure 8.9 Ref. 2. If no information is available, the author recommends a typical value of 0.6 be used.

The main rotor downwash ration for the horizontal tail has the form

$$\varepsilon_{M_H} = \frac{v_H}{v_1}\frac{v_1}{V} = \frac{v_H}{v_1}\frac{D.L.}{4q}$$

The ratio v_H/v_1 has a maximum theoretical value of two because of the maximum theoretical downwash velocity achieved by the main rotor using momentum theory, but empirical measurements

indicate that higher ratios are obtainable [Ref. 2]. The term v_H/v_i can be estimated by using Fig 8.11 Ref. 2. The author recommends a typical value of 1.5 in the absence of more precise information.

The fuselage downwash ratio has the form

$$\varepsilon_{F_H} = \varepsilon_{F_{\alpha_F=0}} + \left(\frac{d\varepsilon_{F_H}}{d\alpha_F}\right)\alpha_F$$

The fuselage downwash ratio can be estimated using Fig 8.14 Ref. 2. The fuselage downwash ratio includes the effects of any wings since most wings installed on a helicopter are small and do not warrant a separate term for downwash effects. A typical value of 0.25 is recommended when no other information is available. Detailed discussion on determination of these effects are contained in Ref. 2 page 489.

The final effect to account for is the time lag for change in rotor downwash to reach the horizontal tail. This effect is accounted for through the time lag

$$\Delta t = \frac{l_H}{V}$$

yielding the following equation for the angle of attack for the horizontal tail

$$\alpha_H = \theta + i_H - (\varepsilon_{M_H} + \varepsilon_{F_H}) - \gamma_c - \left(\frac{\partial \eta_{M_H}}{\partial \dot{z}}\right)\ddot{z}\frac{l_H}{V}$$

These effects, when combined with typical computations for horizontal stabilizer stability analysis, yield the equations found in Appendix C.

Input for the horizontal tail is made in the JANRAD Performance input screen with additional input required for stability analysis entered in Stability and Control Routine New File menu (Fig.

```
Height above reference datum/waterline (ft):
Fuselage station (ft):
Position right of buttline:
Zero lift angle for horizontal tail (deg):
Angle of incidence of horizontal tail (deg):
Lift curve slope of horizontal tail:
Dynamic pressure ratio (pg 489 Prouty):
Rotor downwash ratio (pg 489 Prouty):
Fuselage downwash ratio (pg 489 Prouty):
```

Figure 3.12. Horizontal Tail New File Menu.

3.12) and changed in the Stability and Control Menu Additional Parameters menu, screen 2 (Fig. 3.13).

```
          ***   STABILITY AND CONTROL MENU   ***
          *** ADDITIONAL PARAMETERS (2 of 3) ***

 Horizontal Tail
 1. height above waterline    2. fuselage station
 3. posn right of buttline    4. alpha zero lift
 5. angle of incidence        6. lift curve slope
 7. dynamic pressure ratio    8. rotor downwash ratio
 9. fuselage downwash ratio

 Wing
10. height above waterline   11. fuselage station
12. posn right of buttline   13. alpha zero lift
14. angle of incidence       15. lift curve slope
16. tip cord                 17. root cord
18. rotor downwash ratio     19. fuselage downwash ratio

0. NO CHANGES
Input the parameter to change:
```

Figure 3.13. Additional Parameters Menu, Screen 2.

Stability derivatives from the horizontal tail are combined with the stability derivatives of all other components for display in the Key Control Parameters output screen. The individual horizontal tail stability derivatives can be pulled from the workspace by finding the variable name in VARLIST.TXT and calling the value from the workspace.

E. VERTICAL STABILIZER CONTRIBUTIONS

The stability derivatives are evaluated in the same manner as the horizontal tail. The only differences are that the swirl from the main rotor downwash, the induced flow from the tail rotor, induced velocities from the fuselage and aircraft sideslip all create an effective sideslip (sidewash) angle on the vertical fin.

If the tail rotor is mounted close to the vertical fin, the form for the downwash effect is the same as for the main rotor downwash on the fuselage

$$\eta_T = \left[\frac{D.L.}{4\left(\frac{q_V}{q}\right)q}\right]_V$$

where

$$D.L. = \frac{1}{A_T}\left[\frac{Q_M}{l_T} - L_V\frac{l_V}{l_T}\right]$$

The parameter q_V/q can be estimated using Figure 8.9 in Ref. 2. A typical value of 0.6 is recommended otherwise.

The effect of aircraft sideslip on the vertical fin is negligible when the tail rotor is in close proximity to the vertical fin because the induced velocities of the tail rotor are much higher than the sideslip velocities typical in the linear region of this analysis. In the case where the tail rotor is far away from the vertical fin, or when a NOTAR™ is used, a typical value of sideslip effect is assigned by the program in the form

$$\frac{d\eta_f}{d\beta} = \frac{d\varepsilon_f}{d\alpha} = 0.06$$

The induced sidewash from the main rotor and other factors affecting the tail vertical fin are evaluated by Stability and Control section of JANRAD and do not need to be input by the user. A detailed explanation of these effects in available in Ref. 2.

Input for the vertical tail is made in the JANRAD Performance input screen with additional input required for stability analysis entered in Stability and Control Routine New File menu (Fig. 3.14)

```
Height above reference datum/waterline (ft):
Fuselage station (ft):
Position right of buttline (ft):
Zero lift angle for vertical tail (deg):
Maximum Cl for vertical tail:
Dynamic pressure ratio (pg 489 Prouty):
Lift curve slope of vertical tail:
```

Figure 3.14. Vertical Tail New File Menu.

and changed in the Stability and Control Menu Additional Parameters menu, screen 1 (Fig. 3.15).

As with the horizontal tail, results for the vertical tail are combined with other stability derivatives for final output, but can be called from the workspace individually as outlined previously.

F. WING CONTRIBUTIONS

Most helicopters do not have wings, and the ones that do usually have wings only in the sense that the wings are aerodynamic hard points for weapon mounting. But compound helicopters do have wings that are designed to provide a large amount of lift for the helicopter.

Stability and control analysis for a wing is the same as for a fixed wing aircraft, except that large amounts of rotor downwash must be accounted for. Rotor wash is accounted for in the same fashion as for the horizontal tail and the fuselage. The dynamic pressure at the wing is adjusted by

```
          ***   STABILITY AND CONTROL MENU   ***
          *** ADDITIONAL PARAMETERS (1 of 3) ***

 Main Rotor
 1. flapping mom of inertia   2. hub height above waterline
 3. hub fuselage station      4. hub posn right of buttline
 5. mast incidence

 Tail Rotor (enter zeros (0) if using NOTAR)
 6. height above waterline    7. hub fuselage station
 8. posn right of buttline    9. number of blades
10. blade chord              11. blade radius
12. lift curve slope         13. rotational velocity
14. flap mom of inertia      15. delta-3 angle
16. blade twist

 Vertical Fin
17. height above waterline   18. fuselage station
19. posn right of buttline   20. alpha zero lift
21. CL max                   22. dynamic pressure ratio
23. lift curve slope'

0. NO CHANGES
Input the parameter to change:
```

Figure 3.15. Additional Parameters Menu, Screen 1.

using a dynamic pressure ratio, q_w/q_l. The angle of attack is modified by down wash terms.

$$\varepsilon_{M_w} = \frac{v_w}{v_1}\frac{v_1}{V} = \frac{v_H}{v_1}\frac{D.L.}{4q}$$

$$\varepsilon_{F_w} = \varepsilon_{F_{\alpha_F=0_w}} + \left(\frac{d\varepsilon_{F_w}}{d\alpha_F}\right)\alpha_F$$

$$\alpha_w = \theta + i_w + \alpha_{0_w} - \gamma_c - \varepsilon_{M_w} - \varepsilon_{F_w}$$

Input for the wing is made in the JANRAD Performance input screen with additional input required for stability analysis entered in Stability and Control Routine New File menu (Fig. 3.16) and changed in the Stability and Control Menu Additional Parameters menu, screen 2 (Fig. 3.17).

```
Height above reference datum/waterline (ft):
Fuselage station (ft):
Position right of buttline (ft):
Zero lift angle for wing (deg):
Angle of incidence of wing (deg):
Lift curve slope of wing:
Tip chord (ft):
Root chord (ft):
Rotor downwash ratio (pg 489 Prouty):
Fuselage downwash ratio (pg 489 Prouty):
```

Figure 3.16. Wing New File Menu.

```
            ***   STABILITY AND CONTROL MENU   ***
            *** ADDITIONAL PARAMETERS (2 of 3) ***

 Horizontal Tail
 1. height above waterline    2. fuselage station
 3. posn right of buttline    4. alpha zero lift
 5. angle of incidence        6. lift curve slope
 7. dynamic pressure ratio    8. rotor downwash ratio
 9. fuselage downwash ratio

 Wing
10. height above waterline   11. fuselage station
12. posn right of buttline   13. alpha zero lift
14. angle of incidence       15. lift curve slope
16. tip cord                 17. root cord
18. rotor downwash ratio     19. fuselage downwash ratio

 0. NO CHANGES
 Input the parameter to change:
```

Figure 3.17. Additional Parameters Menu, Screen 2.

As with the horizontal tail, results are combined with other stability derivatives for final output, but can be called from the workspace individually as outlined previously.

G. NOTAR™

NOTAR™ derivatives were obtained from proprietary information and cannot be developed here. The inputs for NOTAR™ are entered in the Stability and Control Routine New File menu (Fig. 3.18) and can be changed in the Stability and Control Additional Parameters Menu screen 3 (Fig. 3.19). NOTAR™ input is echoed in the 'Input Variable' output screen 4, but derivatives must be called from the workspace as described in prior sections.

```
Height above reference datum/waterline (ft):
Fuselage station (ft):
Position right of buttline (ft):
NOTAR boom diameter (ft):
Swirl angle at boom(deg):
Maximum thruster force of NOTAR (lbs):
Thruster fuselage station (ft):
```

Figure 3.18. NOTAR New File Menu.

```
          ***   STABILITY AND CONTROL MENU   ***
           *** ADDITIONAL PARAMETERS (3 of 3) ***

 CG location and Inertias/fuselage parameters
 1. cg ht. above waterline   2. cg fuselage station
 3. cg posn rt of buttline   4. Ixx
 5. Iyy                      6. Izz
 7. Ixz                      8. fuselage downwash ratio

NOTAR if available (enter zeros if using tail rotor)
 9. height above waterline  10. boom fuselage station
11. boom position left ref  12. NOTAR diameter
13. swirl angle at boom     14. NOTAR max force
15. thruster fuselage station

Rigging
16. B1 main/in defl (del e) 17. A1 main/in defl (dela)
18. theta0m/in defl (del c) 19. theta0t/pedal defl (del r or p)
20. NOTAR sleeve twist/defl 21. max rudder defl

 0. NO CHANGES
 Input the parameter to change:
```

Figure 3.19. Additional Parameters Menu, Screen 3.

H. RIGGING

The control input matrix [B] is automatically computed by the program, but the user must establish control rigging slopes and control limits. User inputs are entered in the Stability and Control Routine New File menu (Fig. 3.20) and can be changed in the Stability and Control Additional Parameters Menu screen 3 (Fig. 3.18).

Rigging inputs are echoed in the output listings, but the effects of the rigging are best seen in the Key Control Parameters screen, screen view selection 5, and State Matrices Representations, screen view selection 3.

```
Long cyclic pitch per inch defl (deg/in):
Lateral cyclic pitch per inch defl (deg/in):
Collective pitch per inch defl (deg/in):
Tail rotor pitch change per inch defl or percentage of twist
  Enter 0 (zero) if using NOTAR (deg/in or deg/deg of twist):
Maximum deflection of anti-torque from neutral for NOTAR,
  Enter 0 (zero) if using tail rotor (deg or inch travel):
Displacement of anti-torque control until full rudder
  deflection.  Enter 0 (zero) if rudder is fixed
  (deg or inch travel):
```

Figure 3.20. Rigging New File Menu.

IV. SOFTWARE USE WITH EXAMPLES

A. THE PROUTY HELICOPTER

The program is initiated by a program call from the JANRAD program. After entering MATLAB, the command JANRAD is entered and the opening menu appears (Fig. 4.1).

```
Do you want to edit an existing file or create a new one?
1. edit existing file   2. create new file   >>
```

Figure 4.1. Opening Menu.

If entering a new file, JANRAD will ask for the required input. When basic data entry is complete, a screen appears requesting a filename for the data. Enter a six letter filename. This allows JANRAD to concatenate additional letters to the filename for specific output files (Fig. 4.2).

If editing an old file, loading instructions appear (Fig. 4.3). The filename is entered as instructed. Data entry is performed by selecting the variable to change, then making the appropriate

```
                *** SAVE INSTRUCTIONS ***

A. Save the new data to a specified file name.
B. Do not use an extension or quotations.
C. Use letter/number combinations of 6 characters or less.
D. The file will be saved with a ".mat" extension.

ex: dsgn_2

E. If you made no changes, press < Enter >, the file will
   be saved with the original name.

save file as:
```

Figure 4.2. Save Screen.

entry. If <ENTER> is pressed without entering a value, the last number assigned to the variable is retained. This value is displayed on the screen during the procedure.

```
            *** LOAD INSTRUCTIONS ***

     A. Input the name of the file to edit.
     B. The file was saved in your previous session
        with a ".mat" extension.
     C. Do not include the extension or quotations.

     ex: dsgn1

     name of input file:
```

Figure 4.3. Load Instructions.

After pressing <ENTER>, a menu appears asking which routine to run (Fig. 4.4). When Stability and Control is selected, three additional input screens appear for further data entry. When this is complete, another save menu appears. A new filename can be entered, or <ENTER> can be pressed to save data under the last filename.

```
            *** EXECUTION MENU ***

          1.  Rotor Performance Analysis
          2.  Stability and Control Analysis
          3.  Rotor Dynamics Analysis
          4.  Change data
          5.  Quit

          Enter a 1, 2, 3, 4, or 5   >>
```

Figure 4.4. Execution Menu.

After data entry is complete, the program performs a stability and control analysis then prompts the user for desired output format (Fig. 4.5). If the user wants hard copies of the root loci or the control Bode plots, the plots must be viewed on the screen first to allow MATLAB to create the graph META files for printing.

```
Do you want the results displayed on screen?

NOTE: if you want a hard copy of the plots, you must
select (1) and view them on the screen first.

1. yes   2. no   >>
```

Figure 4.5. Data Display.

Output from the program is in six basic categories as shown in Fig. 4.6. To demonstrate the output features, the Prouty sample helicopter at a hover [Ref. 2] will be used as an example.

```
          *** STABILITY AND CONTROL PROGRAM ***
               *** SCREEN VIEW MENU ***

What do you want to see?

1. Input data.
2. Calculated data.
3. State Matrices.
4. Eigenvalues of the plants and plots of the roots.
5. Key control parameters.
6. Open loop transfer plots.

0. Exit screen view.

Enter a number;
```

Figure 4.6. Screen View Menu.

Selection 1, Input Data, is self explanatory, and will be skipped in the interest of brevity.

Selection 2, Calculated Data, has two screens. The first screen (Fig. 4.7) gives the calculated main rotor parameters of interest. Note that since the helicopter is hovering, many of the quantities are equal to or near zero. The second screen has calculated data for the anti torque system (Fig. 4.8). Again note that many parameters are zero since the aircraft has no forward velocity.

Selection 3 provides the state matrix representations of the longitudinal, lateral and coupled plants as well as the input matrices (Figs. 4.9, 4.10, and 4.11). The states are listed under each heading for convenience. These matrices can be used after exiting the program to simulate aircraft response to impulses, steps or any other variety of input. These matrices can also be used to develop

```
*** CALCULATED DATA (screen 1 of 2) ***
                    prouth

                  Main Rotor

              advance ratio =    0.0
      inflow parameter wrt TPP =  0.000
            Tip path angle =     0.0 degs
         Rotor coning angle =    7.4 degs
    1st lat cyclic term-A1 =     0.0 degs
    1st long cyclic term-B1 =   -0.0 degs
           lateral flapping =    0.00 degs
     longitudinal flapping =     0.00 degs
               Lock number =     7.2

press any key to continue...
```

Figure 4.7. Calculated Data, Screen 1.

```
*** CALCULATED DATA (screen 2 of 2)***
          prouth

           Tail Rotor (zero if NOTAR)

    tail rotor thrust = 1349.5 lbs
        advance ratio =    0.0
     inflow parameter = -0.073
    Rotor coning angle =   -0.1 degs
      lateral flapping =   0.00 degs
longitudinal flapping =    0.00 degs
          Lock number =     2.7

press any key to continue...
```

Figure 4.8. Calculated Data, Screen 2.

response to impulses, steps or any other variety of input. These matrices can also be used to develop state feedback control for the aircraft.

Selection 4 displays the eigenvalues and the root loci of the uncoupled and coupled plants. Before doing this, an instruction screen is displayed to assist in understanding the format of the output (Fig. 4.12).

The second screen displays the longitudinal and the lateral uncoupled eigenvalues, then immediately plots the roots (Fig. 4.13). The plots are META filed by MATLAB after display. To recover them on a hard copy, one must use the Graphics Post Processing (GPP) utility after exiting

```
*** CALCULATED DATA (screen 1 of 2) ***
                    prouth

                  Main Rotor

              advance ratio =    0.0
      inflow parameter wrt TPP =  0.000
             Tip path angle =    0.0 degs
        Rotor coning angle =     7.4 degs
     1st lat cyclic term-A1 =    0.0 degs
    1st long cyclic term-B1 =   -0.0 degs
          lateral flapping =    0.00 degs
     longitudinal flapping =    0.00 degs
              Lock number =      7.2

press any key to continue...
```

Figure 4.7. Calculated Data, Screen 1.

```
*** CALCULATED DATA (screen 2 of 2)***
          prouth

           Tail Rotor (zero if NOTAR)

    tail rotor thrust = 1349.5 lbs
        advance ratio =    0.0
    inflow parameter = -0.073
   Rotor coning angle =   -0.1 degs
     lateral flapping =   0.00 degs
longitudinal flapping =   0.00 degs
          Lock number =    2.7

press any key to continue...
```

Figure 4.8. Calculated Data, Screen 2.

response to impulses, steps or any other variety of input. These matrices can also be used to develop state feedback control for the aircraft.

Selection 4 displays the eigenvalues and the root loci of the uncoupled and coupled plants. Before doing this, an instruction screen is displayed to assist in understanding the format of the output (Fig. 4.12).

The second screen displays the longitudinal and the lateral uncoupled eigenvalues, then immediately plots the roots (Fig. 4.13). The plots are META filed by MATLAB after display. To recover them on a hard copy, one must use the Graphics Post Processing (GPP) utility after exiting

```
Longitudinal uncoupled plant (A or F depending on notation)
States are [u w q theta]

   -0.0106         0    2.5068  -32.2000
         0   -0.2698         0    0.0002
    0.0038         0   -0.8878         0
         0         0    0.9977         0


Longitudinal uncoupled input matrix (B or G depending on
notation)
Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]

    1.2248   -0.0000   -0.0738         0
         0   -5.8773         0         0
   -0.4338    0.0000    0.0261   -0.0238
         0         0         0         0

press any key to continue...
```

Figure 4.9. Longitudinal State Matrix Representation.

```
Lateral/directional uncoupled plant (A or F depending on
notation)
States are [v p phi r psi]

   -0.0305   -2.6260   32.1267    0.7353         0
   -0.0064   -1.0274         0    0.0783         0
         0    1.0000         0   -0.0000         0
    0.0130    0.0783         0   -3.3360         0
         0         0         0    1.0000         0

Lateral/directional uncoupled input matrix (B or G depending
on notation)
Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]
    0.1200    0.0000    0.7533   -1.9736
    0.0486    0.0000    0.3049   -0.2101
         0         0         0         0
         0    0.3001         0    1.2959
         0         0         0         0

press any key to continue...
```

Figure 4.10. Lateral/Directional State Matrix Representation.

the program as explained in the closing screen.

The first root in the lateral directional plant is an artifact of the system representation and has no physical meaning. The other modes are the Dutch-roll and spiral modes. The longitudinal modes shown are the short period and phugoid.

```
Coupled plant (A or F depending on notation)
States are [u w q theta v p phi r psi]

  Columns 1 through 7
-0.0106         0    2.5068  -32.2000   -0.0057   -0.7469         0
      0   -0.2698         0    0.0002         0         0    2.1710
 0.0038         0   -0.8878         0    0.0013    0.2645         0
      0         0    0.9977         0         0         0         0
 0.0057         0   -0.7469   -0.0000   -0.0305   -2.6260   32.1267
 0.0023         0   -0.3023         0   -0.0064   -1.0274         0
      0         0    0.0000         0         0    1.0000         0
      0   -0.0198         0         0    0.0130    0.0783         0
      0         0         0         0         0         0         0
  Columns 8 through 9
      0         0
      0         0
 0.0266         0
 0.0674         0
 0.7353         0
 0.0783         0
-0.0000         0
-3.3360         0
 1.0000         0

Coupled input matrix (B or G depending on notation)
Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]

    1.2248   -0.0000   -0.0738         0
         0   -5.8773         0         0
   -0.4338    0.0000    0.0261   -0.0238
         0         0         0         0
    0.1200    0.0000    0.7533   -1.9736
    0.0486    0.0000    0.3049   -0.2101
         0         0         0         0
         0    0.3001         0    1.2959
         0         0         0         0

press any key to continue...
```

Figure 4.11. Coupled State Matrix Representation.

Comparing the root estimates of the longitudinal and lateral plants with the roots of the coupled plant in the third screen shows close correlation.

The plots of the eigenvalues are in the Argand plane (Figs 4.15, 4.16, and 4.17). Note that the plant has some unstable roots. This is typical of helicopters. Very rarely are all the modes stable without some form of stability augmentation.

```
After you view the root loci plot, a meta file is made.
When you are done a screen will tell you the file names
of the meta files.  To get a hard copy of the plots, you
must graphics post process (GPP) the files for your
particular printer set-up then, print.

NOTE: If ALL roots are real, MATLAB will NOT plot them
      in the Argand plane, but will plot the root against
      its position in the vector (e.g. the first root would
      be plotted as (1,root))

press any key to continue...
```

Figure 4.12. Eigenvalue and Root Loci Instruction Screen.

```
        *** EIGENVALUES ***
            prouth

    Uncoupled

    Longitudinal plant

            Root                wn              damping
      -1.0155                1.0155              1.0000
       0.0585 + 0.3398i      0.3448             -0.1698
       0.0585 - 0.3398i      0.3448             -0.1698
      -0.2698                0.2698              1.0000

    Lateral/Directional plant

            Root                wn              damping
            0                     0             -1.0000
      -3.3433                3.3433              1.0000
      -1.1784                1.1784              1.0000
       0.0639 + 0.4025i      0.4075             -0.1568
       0.0639 - 0.4025i      0.4075             -0.1568

    press any key to continue...
```

Figure 4.13. Uncoupled Eigenvalues.

The final screen for this section (Fig. 4.18) tells the user the filenames for the graphs. The user can write these down, or refer to Appendix E for the filenames when hard copies are desired.

The Key Control Parameters screen 2 (Figs 4.19 and 4.20), selection 5, provides numbers that can be used to determine Cooper-Harper pilot ratings and compliance with MIL-H-8501. The screen also tells the amount of cross coupling between flapping modes of the main rotor system. Though

```
*** EIGENVALUES ***
prouth

Coupled Plant

        Root                wn              damping
           0                 0              -1.0000
     -3.3432             3.3432               1.0000
     -1.0970 + 0.2583i   1.1270               0.9734
     -1.0970 - 0.2583i   1.1270               0.9734
      0.0760 + 0.3983i   0.4054              -0.1873
      0.0760 - 0.3983i   0.4054              -0.1873
      0.0461 + 0.3582i   0.3611              -0.1276
      0.0461 - 0.3582i   0.3611              -0.1276
     -0.2689             0.2689               1.0000

press any key to continue...
```

Figure 4.14. Coupled Eigenvalues.

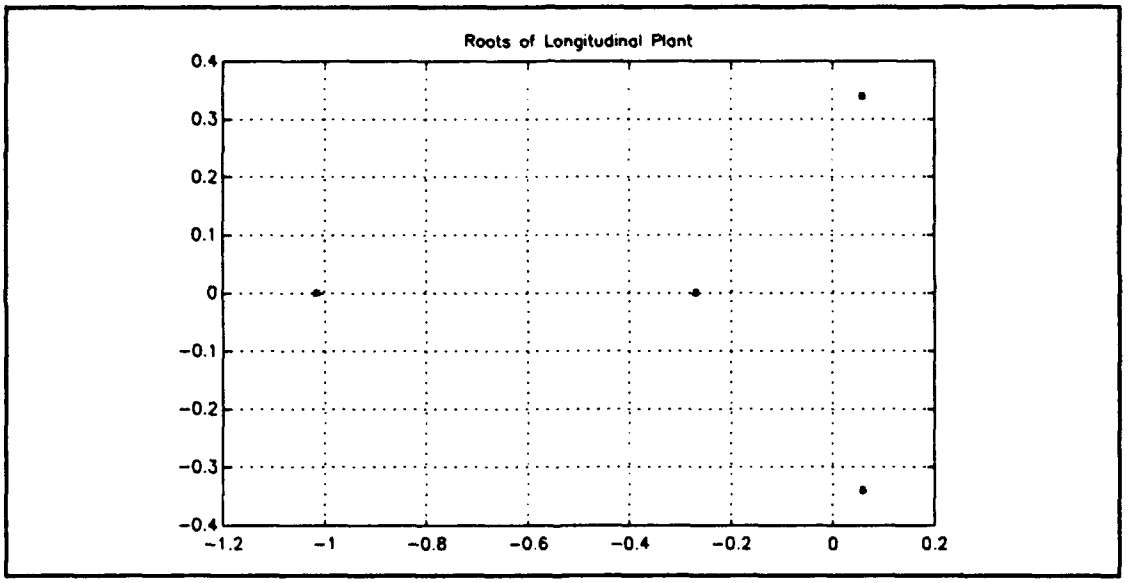


Figure 4.15. Longitudinal Roots.

```
Plots are saved under the following filenames:

Longitudinal roots - rootlon.met
Lateral/Directional roots - rootlat.met
Coupled roots - rootcoup.met

press any key to continue...
```

Figure 4.18. Eigenvalues Closing Screen.

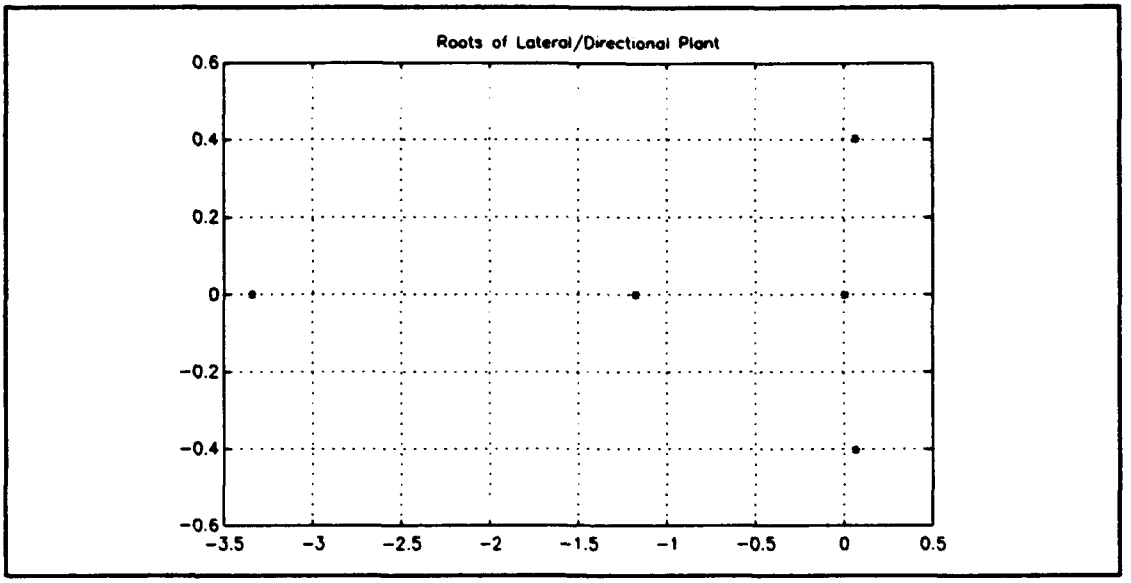


Figure 4.16. Lateral Directional Roots.

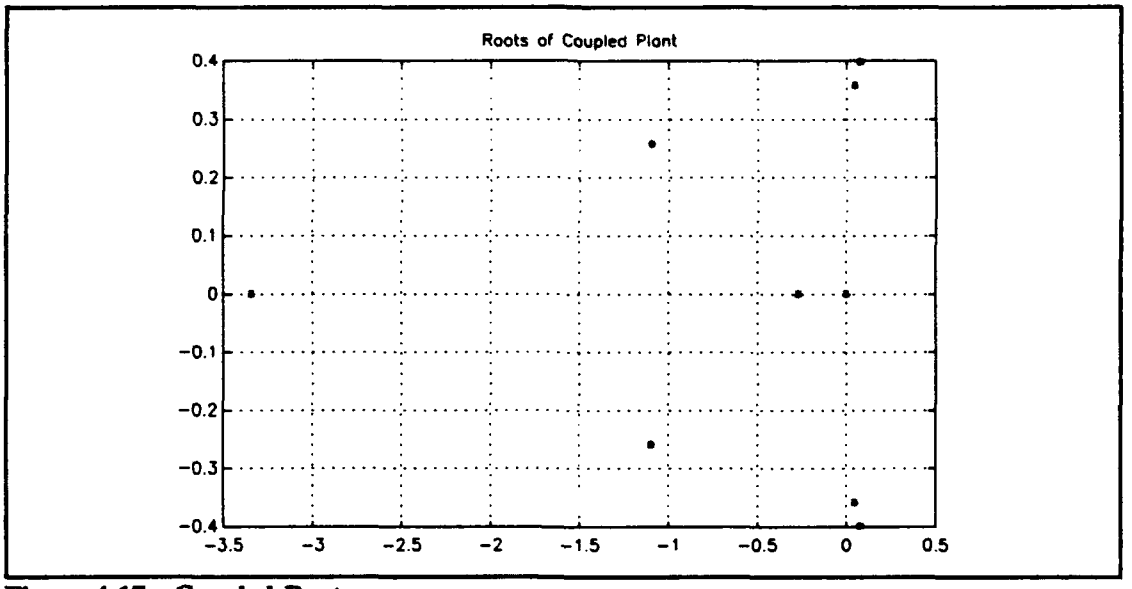


Figure 4.17. Coupled Roots.

MIL-H-8501 has been superseded by ADS-33C, the numbers still provide information on the damping characteristics and control power of the helicopter.

The last selection, open loop transfer plots, provides the transfer function from a control input to a state output. The purpose of these plots is to allow the designer to design automatic flight control systems and analyze the best techniques for controlling unstable modes. For example, to see the

```
        *** KEY CONTROL PARAMETERS (screen 1 of 2) ***
                          prouth

     cross coupling =   0.08

                        Designed damping
                pitch = -35513.5 ft-lbs/(rad/sec)
                 roll = -35957.8 ft-lbs/(rad/sec)
                  yaw = -116759.1 ft-lbs/(rad/sec)

                        Control Power
                pitch = -17351.2 ft-lbs/in
                 roll = 10672.0 ft-lbs/in
                  yaw = 45355.3 ft-lbs/in

   press any key to continue...
```

Figure 4.19. Key Control Parameters, Screen 1.

```
        *** KEY CONTROL PARAMETERS (screen 2 of 2) ***
                          prouth

                    Cooper Harper Pilot Ratings
                     damping/moment of inertia
   pitch (dM/dq)/Iyy =  -0.89 [ft-lbs/(rad/sec)]/(slug ft^2)
    roll (dR/dp)/Ixx =  -1.03 [ft-lbs/(rad/sec)]/(slug ft^2)
     yaw (dN/dr)/Izz =  -3.34 [ft-lbs/(rad/sec)]/(slug ft^2)

                  control power/moment of inertia
   pitch (dM/in)/Iyy =  -0.43 (ft-lbs/in)/(slug ft^2)
    roll (dR/in)/Ixx =   0.30 (ft-lbs/in)/(slug ft^2)
     yaw (dN/in)/Izz =   1.30 (ft-lbs/in)/(slug ft^2)

   press any key to continue...
```

Figure 4.20. Key Control Parameters, Screen 2.

effects of collective pitch on forward velocity, u, select the longitudinal set of plots from the screen (Fig. 4.21) and page through the plots until you get to "Collective Pitch to U." It can be seen as shown in Fig. 4.22, that there is little control gain from collective to forward velocity at a hover. Obviously, collective would be a poor choice for controlling fore and aft motion at a hover.

If one looks at "Longitudinal Cyclic to U," it can be seen that there is high gain from input to output, especially at the short period frequency (Fig. 4.23). This shows that longitudinal cyclic is probably a good choice for controlling fore and aft motion.

```
After you view a bode plot of the transfer function from
input to state output, a meta file is made.  When you exit,
a screen will tell you the file names of the meta files.
To get a hard copy of the plots, you must graphics post
process (GPP) the files for your particular printer set-up
then print.

While viewing a plot, press any key to go to the next plot

Do you want to see longitudinal or lateral/directional plots?

1.  Longitudinal (eight plots total).
2.  Lateral Directional (ten plots total).

Enter a number :
```

Figure 4.21. Open Loop Transfer Plots, Screen 1.

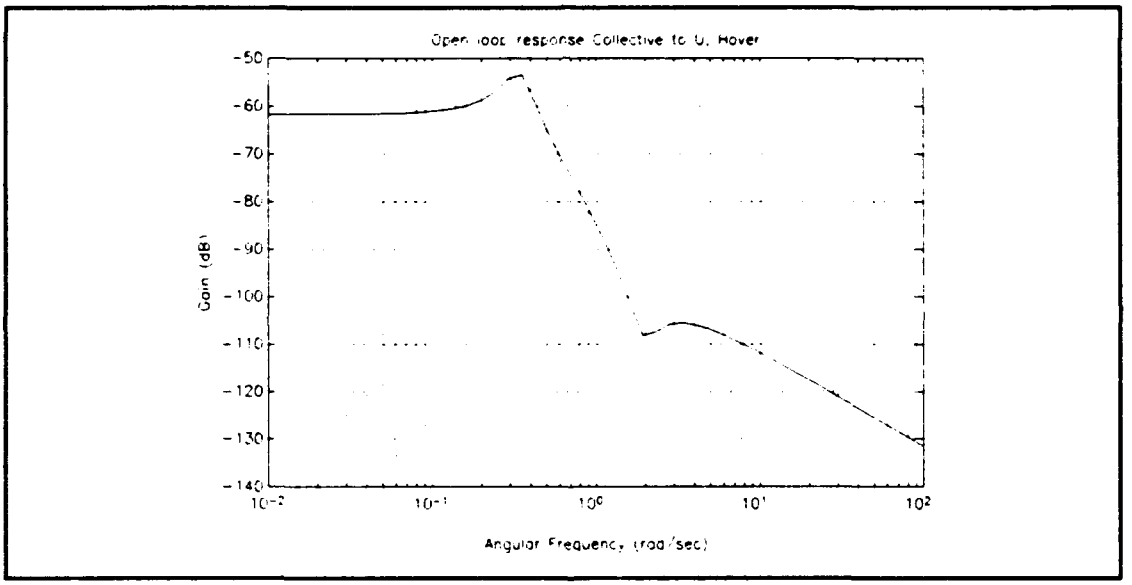


Figure 4.22. Collective to Forward Velocity.

Another observation from this plot is that at low frequencies (e.g. when you hold the cyclic steady) there is good control response. This is consistent with flight experience. When the cyclic is displaced forward, the aircraft pitches forward as the cyclic is moved, until some equilibrium pitch attitude is reached. When the cyclic is stopped, the attitude remains fixed and the aircraft moves forward. Likewise, when the cyclic is moved back and forth very rapidly, the aircraft may vibrate, but the pitch attitude remains essentially fixed because the aircraft cannot move as quickly as the

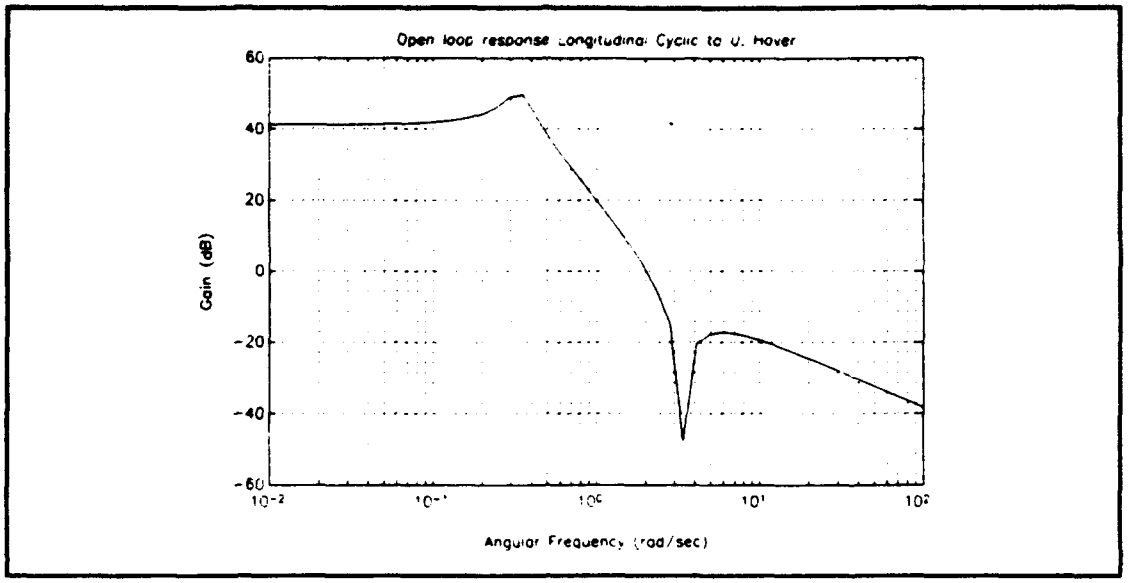


Figure 4.23. Longitudinal Cyclic to Forward Velocity.

control input, hence, the aircraft does not change its average velocity.

These effects can be seen in the control responses of all the controls. In the lateral-directional plant, the response from lateral cyclic to sideslip, v, has the same type of peak near the Dutch-roll mode, but response above the Dutch-roll frequency is reduced (Fig. 4.24).

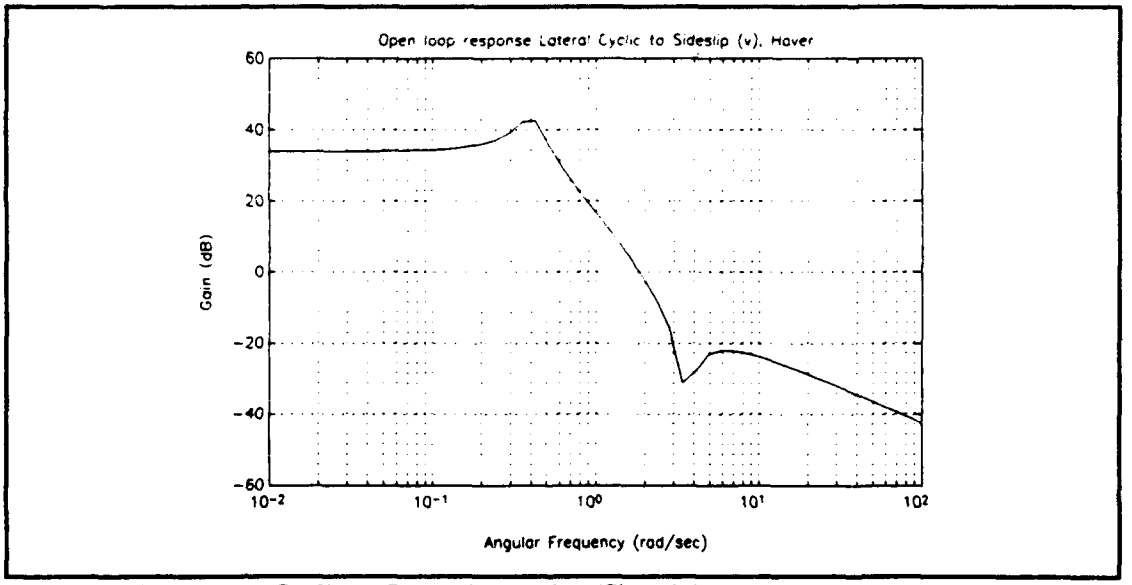


Figure 4.24. Lateral Cyclic to Lateral Velocity (Sideslip).

A complete set of Bode plots for the prouty helicopter can be found in Appendix G. The shapes of these curves is typical of helicopters with single main rotors and a tail rotor.

Hard copies of the plots are obtained in the same manner as the root locus plots. A complete list of command response META files is located in Appendix E.

The output from the screens is written to a diary after exiting the Screen View menu. The Output Data Instruction menus in Figs. 4.25 through 4.27 explain the data recovery procedures.

```
      *** OUTPUT DATA INSTRUCTIONS (screen 1 of 3) ***

Because this subroutine generates a large number of single
value data not shown on the output screen, a text file
VARLIST.TXT is on this disk which lists the variable names
for all the stability derivatives.  Stability derivative
contributions for all major aircraft components can be found
by reading the text file VARLIST.TXT, then asking MATLAB the
variable name corresponding to the derivative.
```

Figure 4.25. Output Data Instructions, Screen 1.

```
    *** OUTPUT DATA INSTRUCTIONS (screen 2 of 3) ***

A. Data from the output screen saved to a file named:
   "filename1.stb"

B. This is a text file, use the TYPE command to view the file
   or use a text editor to view/print the file.

C. Matrix and vector data saved to a default file named:
   "mstabdat.mat"

D. This is a ".mat" binary file, use the LOAD command to
   retrieve the data for plotting.

E. Rename "mstabdat.mat" to another ".mat" file.
   The file "mstabdat.mat" will be overwritten when
   the program is executed.

F. Do not rename the file as "filename1.mat"
   The file "filename1.mat" is already on disk
   and used for future editing.
```

Figure 4.26. Output Data Instructions, Screen 2.

```
        *** OUTPUT DATA INSTRUCTIONS (screen 3 of 3) ***

A. Single value data saved to a default file named:
   "vstabdat.mat"

B. This is a ".mat" binary file, use the LOAD command to
   retrieve the data for plotting.

C. Rename "vstabdat.mat" to another ".mat" file.
   The file "vstabdat.mat" will be overwritten when
   the program is executed.

D. Do not rename the file as "filename1.mat"
   The file "filename1.mat" is already on disk
   and used for future editing.

          *** END STABILITY AND CONTROL ROUTINE ***
```

Figure 4.27. Output Data Instructions, Screen 3.

B. A HELICOPTER WITH NOTAR™

A final example will be a notional helicopter designed for the 1993 annual American Helicopter Society (AHS) Design Competition, the Arapaho. The forward flight mode of the helicopter will be used. Since the procedures for entering and displaying data have been covered, only the salient points of the design will be addressed.

The helicopter incorporated a NOTAR™ tailboom. As a result, the tail rotor derivatives are unusual as are the command transfer Bode plots.

The Output file will have a zero displayed for all tail rotor computations except the tail rotor thrust. The thrust displayed in this case will be the NOTAR™ thruster force (Fig 4.28).

By comparing the longitudinal control matrices with and without a tail rotor (Fig. 4.10 and 4.29) it can be seen that a change in tail rotor thrust has a pitching moment contribution (look at the B(3,4) term). Because the tail rotor rotates about an axis that is parallel to the Y control axis, this contribution is to be expected with a change in tail rotor thrust requirement. In contrast, the NOTAR™ does not have this contribution because the thruster fan rotates on an axis parallel to the X control axis.

```
*** CALCULATED DATA (screen 2 of 2)***
arapac

Tail Rotor (zero if NOTAR)

tail rotor thrust =  264.2 lbs
advance ratio =  0.0
inflow parameter =  0.000
Rotor coning angle =  0.0 degs
lateral flapping =  0.00 degs
longitudinal flapping =  0.00 degs
Lock number =  0.0

press any key to continue...
```

Figure 4.28. Tail Rotor Output.

```
Longitudinal uncoupled plant (A or F depending on notation)
States are [u w q theta]

   -0.0363    0.0567    3.3538  -32.1459
    0.0685    0.3972   84.6890   -1.8657
   -0.0098   -0.0204   -0.5489         0
         0         0    0.9999         0

Longitudinal uncoupled input matrix (B or G depending on notation)
Inputs are [longitudinal cyclic, collective, lateral cyclic, pedals]

    7.0005   -0.5855   -2.0435         0
   26.0454  -10.9228         0         0
   -1.2115    0.5179    0.3537         0
         0         0         0         0

press any key to continue...
```

Figure 4.29. Arapaho Control Matrix.

Figure 4.30 demonstrates the difference in flight control response with airspeed. In the hover case, longitudinal cyclic has little direct effect on vertical velocity. But notice that in the forward flight regime, a longitudinal deflection has a great effect on vertical velocity. A zoom climb can be done in forward flight but not at a hover.

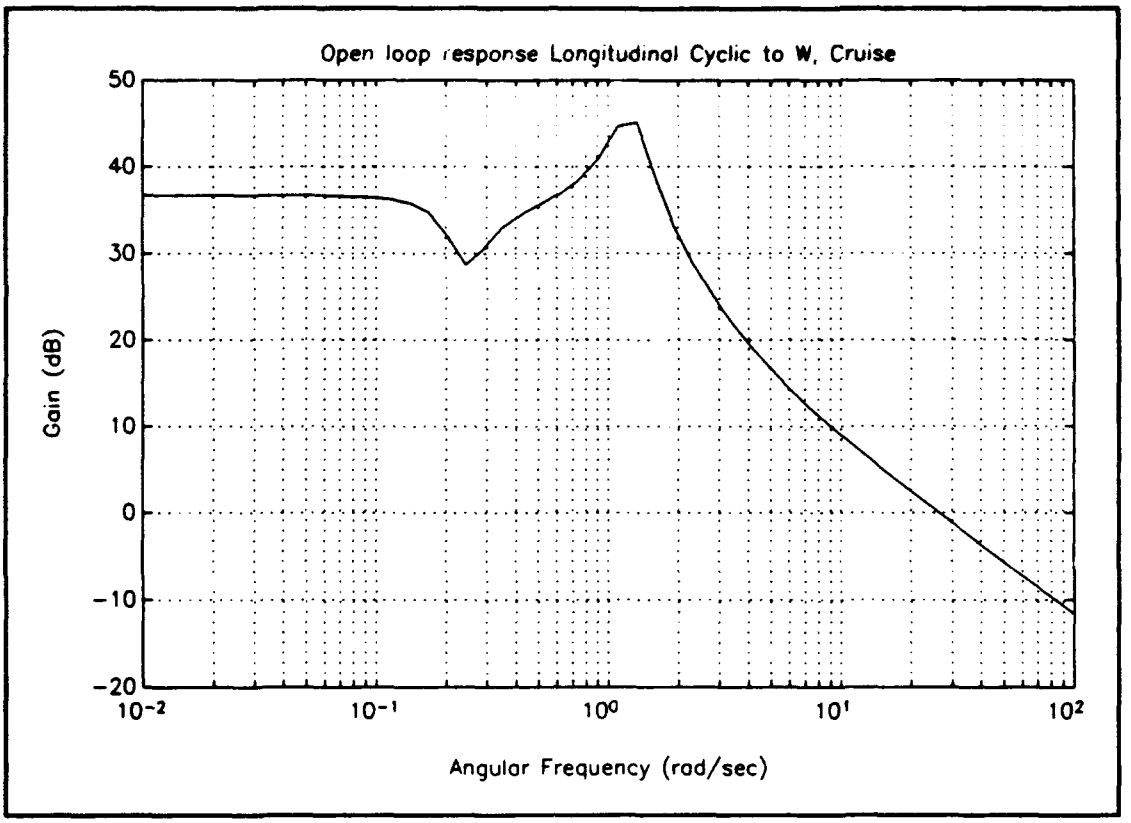


Figure 4.30. Longitudinal Cyclic to Vertical Velocity.

V. CONCLUSIONS AND RECOMMENDATIONS.

A. CONCLUSIONS

The aerodynamicist should not be intimidated by the complexity of helicopters. The complexities of determining the stability characteristics of a helicopter are formidable, but helicopter flight dynamics are essentially no different than fixed wing flight dynamics. Modern software reduces helicopter analysis to a routine procedure, making the analysis of these characteristics much less time consuming.

B. RECOMMENDATIONS

To improve the fidelity of JANRAD Stability and Control, the following actions are recommended.

a) Expand the state matrix to include a_{1sM}, b_{1sM}, C_T/σ_M, Ch/σ_M, Cq/σ_M, μ, λ' a_{1sT}, b_{1sT}, C_T/σ_T, Ch/σ_T, Cq/σ_T, μ_T and λ_T for the tail and main rotors. This representation would allow students to run simulations on MATLAB that could show rotor flapping, cyclic pitch changes, and fluctuations in the coefficients of torque, thrust and H-force.

b) Expand the input screens to allow selecting different lift and drag characteristics for the fuselage. Even though this type of information is derived empirically, studies of existing helicopter with appropriate flight test data could be done without the added artificiallity of a particular fuselage configuration. JANRAD currently uses fuselage aerodynamic characteristics of the Prouty sample helicopter as outlined in section III. C.

c) Improve the fidelity of the tail rotor trim solution. The method used to derive the tail rotor trim solution is extremely rough compared to the trim solution of the main rotor.

d) Expand the wing inputs to include dihedrally mounted wings and wing sweep. Even though most helicopters do not use wings for lift or control, compound helicopters are likely to require a higher fidelity wing solution.

APPENDIX A. NOTATION.

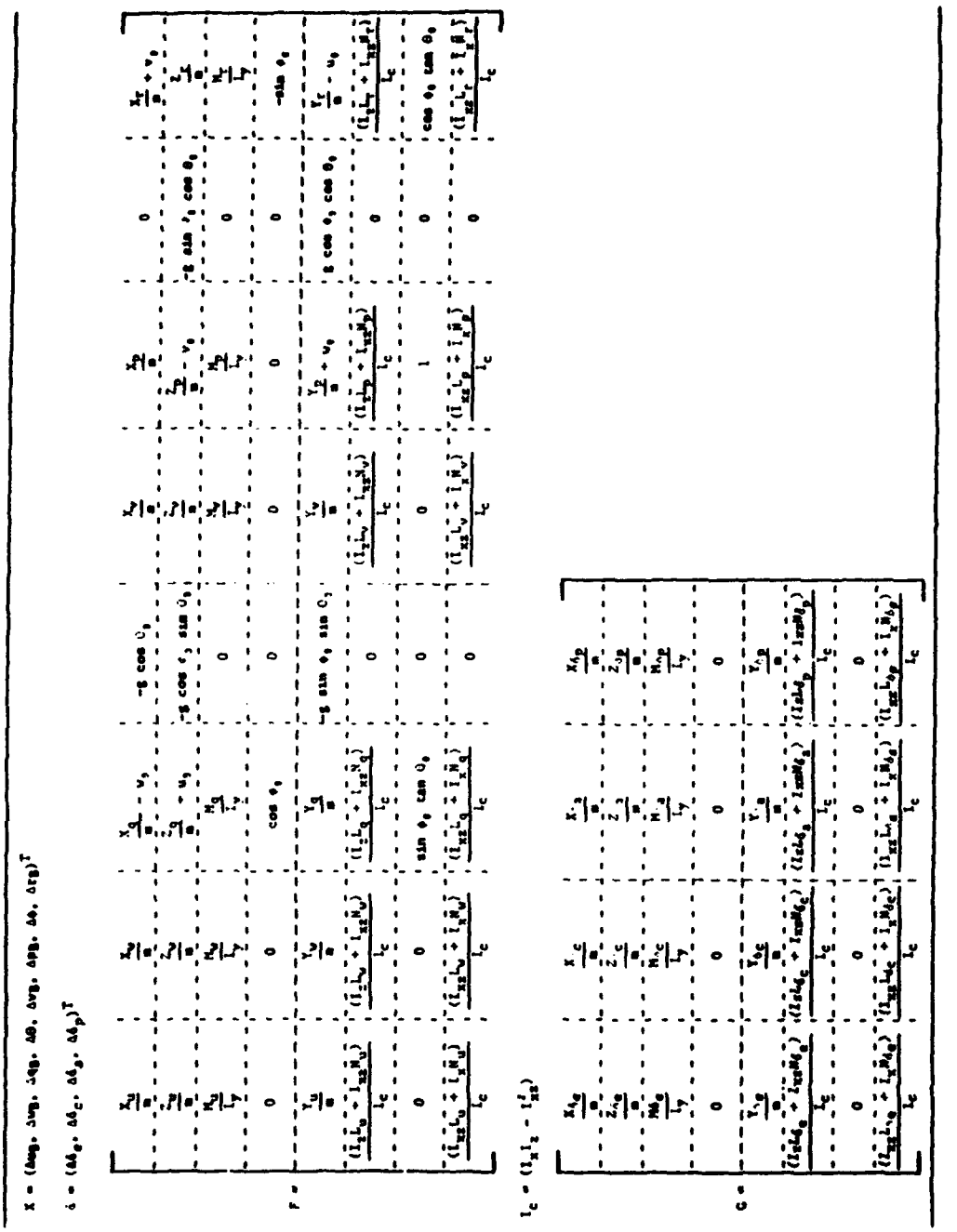
Symbol	Definition	Units
a	Lift curve slope	1/rad
a_0	Coning angle	rad, deg
a_{1s}	First harmonic of longitudinal flapping wrt shaft axis	rad, deg
b	Number of blades	
b_{1s}	First harmonic of lateral flapping wrt shaft axis	rad, deg
c	Chord	ft
c_d	Coefficient of drag	
cg	Center of gravity	
c_l	Coefficient of lift	
c_m	Coefficient of moment	
e	Hinge offset	ft
f	Equivalent flat plate area	ft^2
g	Acceleration due to gravity	ft/sec^2
i	Incidence	rad, deg
l	Fuselage station	ft
p	Roll rate	rad/sec
q	Pitch rate	rad/sec
q	Dynamic pressure	lb/ft
r	Blade element radius	ft
r	Yaw rate	rad/sec
u	forward velocity	ft/sec
$\mathbf{u}$	Input vector	
v	Induced velocity	ft/sec
v	Lateral velocity	ft/sec
w	Vertical velocity	ft/sec
y	Output vector	
y	Buttline position	ft
z	Waterline position	ft
$\mathbf{A}$	Plant matrix	
A	Area of disk	ft^2
A_b	Blade area	ft^2
A_1	First lateral harmonic of blade feathering	rad,deg
$\mathbf{B}$	Input matrix	
B_1	First longitudinal harmonic of blade feathering	rad,deg
$\mathbf{C}$	Output matrix	
C_h	Coefficient of horizontal force	
C_p	Coefficient of power	

C_q	Coefficient of torque	
C_t	Coefficient of thrust	
D	Direct transition matrix	
D.L.	Disk loading	lb/ft^2
D	Drag	lb
GW	Gross weight	lb
H	Horizontal force on rotor	lb
I_b	Flapping moment of inertia	slug ft^2
I_{xx}	Rolling moment of inertia	slug ft^2
I_{yy}	Pitching moment of inertia	slug ft^2
I_{zz}	Yawing moment of inertia	slug ft^2
L	Lift	lb
M	Pitching moment	ft-lb
N	Yawing moment	ft-lb
Q	Torque	ft-lb
R	Blade radius	ft
R	Rolling moment	ft-lb
T	Thrust	lb
X	State vector	
α	Angle of attack	rad, deg
β	Blade flapping angle	rad, deg
β	Sideslip angle	rad, deg
γ	Lock number	
γ_c	Climb angle	rad, deg
δ_3	Pitch-flap coupling angle	rad, deg
ϵ	Downwash angle	rad, deg
η	Sidewash angle	rad, deg
θ	Pitch angle	rad, deg
Θ	Blade pitch	rad, deg
Θ_1	Blade twist	rad, deg
λ	Inflow ratio wrt swashplate	
λ'	Inflow ratio wrt tip path plane	
μ	Tip speed ratio	
ρ	Density of air	$slug/ft^3$
σ	Rotor Solidity	
Φ	Roll angle	rad, deg
ϕ	Inflow angle	rad
Ψ	Yaw angle	rad, deg
Ω	Rotational velocity of rotor	rad/sec
ΩR	Rotor tip speed	ft/sec

Subscript	Definition
0 thru 1	Blade position in r/R
0	Nominal
0	Reference
b	Per blade
i	Induced
F	Fuselage
H	Horizontal stabilizer
M	Main rotor
N	NOTAR™
Q	Torque
T	Tail rotor
T	Thrust
TPP	Tip path plane
V	Vertical stabilizer

APPENDIX B. NASA STATE SPACE REPRESENTATION.

NASA state space representation from Ref. 3.



APPENDIX C. SUMMARY OF STABILITY DERIVATIVES.

A summary of stability derivatives extracted from Ref. 2.

Basic Rotor Derivatives near Hover

Derivative	Equation
$\left(\frac{\partial\mu}{\partial\dot{x}}\right)_M, \left(\frac{\partial\lambda'}{\partial\dot{z}}\right)_M$	$\left(\frac{1}{\Omega R}\right)_M$
$\left(\frac{\partial\mu}{\partial\dot{x}}\right)_T, \left(\frac{-\partial\lambda'}{\partial\dot{y}}\right)_T$	$\left(\frac{1}{\Omega R}\right)_T$
$\frac{\partial a_{1_s}}{\partial\mu}$	$\frac{8}{3}\theta_0 + 2\theta_1 - 2\frac{v_1}{\Omega R}$
$\frac{\partial b_{1_s}}{\partial\mu}$	$\frac{4}{3}a_0$
$\frac{\partial C_T/\sigma}{\partial\theta_0}$	Take from hover charts of Chapter 1
$\frac{\partial C_Q/\sigma}{\partial\theta_0}$	Take from hover charts of Chapter 1
$\frac{\partial C_T/\sigma}{\partial\lambda'}$	$\frac{1}{\frac{8}{a} + \frac{\sqrt{\frac{\sigma}{2}}}{\sqrt{C_T/\sigma}}}$
$\frac{\partial a_{1_s}}{\partial q}, \frac{\partial b_{1_s}}{\partial p}$	$-\frac{16}{\gamma\Omega\left(1-\frac{e}{R}\right)^2} - \frac{12\frac{e}{R}}{\gamma\Omega\left(1-\frac{e}{R}\right)^3}$

Derivative	Equation
$\frac{\partial C_Q/\sigma}{\partial\lambda'}$	$-\frac{a}{4}\left(\theta_{.7} - 2\frac{v_1}{\Omega R}\right)$
$\frac{\partial a_{1_s}}{\partial p}, \frac{-\partial b_{1_s}}{\partial q}$	$\frac{1}{\Omega}\left[1 - \frac{192\frac{e}{R}}{\gamma^2\left(1-\frac{e}{R}\right)^3}\right]$
$\frac{\partial a_{1_s}}{\partial A_1}, \frac{\partial b_{1_s}}{\partial B_1}$	$\frac{12\frac{e}{R}}{\gamma\left(1-\frac{e}{R}\right)^3}$
$\frac{\partial a_{1_s}}{\partial B_1}, \frac{-\partial b_{1_s}}{\partial A_1}$	$\frac{-1}{1 + \frac{144\left(\frac{e}{R}\right)^2}{\gamma^2\left(1-\frac{e}{R}\right)^6}}$
$\frac{\partial C_H/\sigma}{\partial a_{1_s}}, \frac{\partial C_y/\sigma}{\partial b_{1_s}}$	$\frac{3}{2}C_T/\sigma\left(1 - \frac{a}{18}\frac{\theta_{.7}}{C_T/\sigma}\right)$
$\frac{\partial M}{\partial a_{1_s}}, \frac{\partial R}{\partial b_{1_s}}$	$\frac{\frac{3}{4}\frac{e}{R}A_1\rho R(\Omega R)^2 a}{\gamma}$

Main Rotor Derivatives near Hover

Derivative	Equation
$\left(\frac{\partial X}{\partial \dot{x}}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial \mu} \frac{\partial \mu}{\partial \dot{x}}$
$\left(\frac{\partial X}{\partial \dot{y}}\right)_M$	$-\left(\frac{\partial Y}{\partial \dot{x}}\right)_M$
$\left(\frac{\partial X}{\partial q}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial q}$
$\left(\frac{\partial X}{\partial p}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial p}$
$\left(\frac{\partial X}{\partial \theta_0}\right)_M$	$-\rho A_b(\Omega R)^2 (\bar{a}_{1_s} + i_M) \frac{\partial C_T/\sigma}{\partial \theta_0}$
$\left(\frac{\partial X}{\partial A_1}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial A_1}$
$\left(\frac{\partial X}{\partial B_1}\right)_M$	$\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial B_1}$
$\left(\frac{\partial Y}{\partial \dot{x}}\right)_M$	$\rho A_b(\Omega R)^2 \frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial b_{1_s}}{\partial \mu} \frac{\partial \mu}{\partial \dot{x}}$
$\left(\frac{\partial Y}{\partial \dot{y}}\right)_M$	$\left(\frac{\partial X}{\partial \dot{x}}\right)_M$
$\left(\frac{\partial Y}{\partial q}\right)_M$	$\left(\frac{\partial X}{\partial p}\right)_M$
$\left(\frac{\partial Y}{\partial p}\right)_M$	$-\left(\frac{\partial X}{\partial q}\right)_M$

Derivative	Equation
$\left(\frac{\partial Y}{\partial \theta_0}\right)_M$	$\rho A_b(\Omega R)^2 \bar{b}_{1_s} \frac{\partial C_T/\sigma}{\partial \theta_0}$
$\left(\frac{\partial Y}{\partial A_1}\right)_M$	$\left(\frac{\partial X}{\partial B_1}\right)_M$
$\left(\frac{\partial Y}{\partial B_1}\right)_M$	$-\left(\frac{\partial X}{\partial A_1}\right)_M$
$\left(\frac{\partial Z}{\partial \dot{z}}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_T/\sigma}{\partial \lambda'} \frac{\partial \lambda'}{\partial \dot{z}}$
$\left(\frac{\partial Z}{\partial \theta_0}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_T/\sigma}{\partial \theta_0}$
$\left(\frac{\partial R}{\partial \dot{x}}\right)_M$	$\left(\frac{dR}{db_{1_s}}\right)_M \frac{\partial b_{1_s}}{\partial \mu} \frac{\partial \mu}{\partial \dot{x}} + \left(\frac{\partial Y}{\partial \dot{x}}\right)_M^{b_M}$
$\left(\frac{\partial R}{\partial \dot{y}}\right)_M$	$-\left(\frac{\partial M}{\partial \dot{x}}\right)_M$
$\left(\frac{\partial R}{\partial \dot{z}}\right)_M$	$\left(\frac{\partial Z}{\partial \dot{z}}\right)_M^{y_M}$
$\left(\frac{\partial R}{\partial q}\right)_M$	$\left(\frac{dR}{db_{1_s}}\right)_M \frac{\partial b_{1_s}}{\partial q} + \left(\frac{\partial Y}{\partial q}\right)_M^{b_M}$
$\left(\frac{\partial R}{\partial p}\right)_M$	$\left(\frac{dR}{db_{1_s}}\right)_M \frac{\partial b_{1_s}}{\partial p} + \left(\frac{\partial Y}{\partial p}\right)_M^{b_M}$
$\left(\frac{\partial R}{\partial \theta_0}\right)_M$	$\left(\frac{\partial Y}{\partial \theta_0}\right)_{b_M} + \left(\frac{\partial Z}{\partial \theta_0}\right)_{y_M}$

Derivative	Equation
$\left(\frac{\partial R}{\partial A_1}\right)_M$	$\left(\frac{dR}{db_{1_c}}\right)_M \frac{\partial b_{1_c}}{\partial A_1} + \left(\frac{\partial Y}{\partial A_1}\right)_M h_M$
$\left(\frac{\partial R}{\partial B_1}\right)_M$	$\left(\frac{dR}{db_{1_c}}\right)_M \frac{\partial b_{1_c}}{\partial B_1} + \left(\frac{\partial Y}{\partial B_1}\right)_M h_M$
$\left(\frac{\partial M}{\partial \dot{x}}\right)_M$	$\left(\frac{dM}{da_{1_c}}\right)_M \frac{\partial a_{1_c}}{\partial \mu} \frac{\partial \mu}{\partial \dot{x}} - \left(\frac{\partial X}{\partial \dot{x}}\right)_M h_M$
$\left(\frac{\partial M}{\partial \dot{y}}\right)_M$	$\left(\frac{\partial R}{\partial \dot{x}}\right)_M$
$\left(\frac{\partial M}{\partial \dot{z}}\right)_M$	$\left(\frac{\partial Z}{\partial \dot{z}}\right)_M I_M$
$\left(\frac{\partial M}{\partial q}\right)_M$	$\left(\frac{dM}{da_{1_c}}\right)_M \frac{\partial a_{1_c}}{\partial q} \left(\frac{\partial X}{\partial q}\right)_M h_M$
$\left(\frac{\partial M}{\partial p}\right)_M$	$\left(\frac{dM}{da_{1_c}}\right)_M \frac{\partial a_{1_c}}{\partial p} - \left(\frac{\partial X}{\partial p}\right)_M h_M$
$\left(\frac{\partial M}{\partial \theta_0}\right)_M$	$-\left(\frac{\partial X_A}{\partial \theta_0}\right)_M h_M + \left(\frac{\partial Z}{\partial \theta_0}\right)_M I_M$
$\left(\frac{\partial M}{\partial A_1}\right)_M$	$\left(\frac{dM}{da_{1_c}}\right)_M \frac{\partial a_{1_c}}{\partial A_1} - \left(\frac{\partial X}{\partial A_1}\right)_M h_M$
$\left(\frac{\partial M}{\partial B_1}\right)_M$	$\left(\frac{dM}{da_{1_c}}\right)_M \frac{\partial a_{1_c}}{\partial B_1} - \left(\frac{\partial X}{\partial B_1}\right)_M h_M$
$\left(\frac{\partial N}{\partial \dot{z}}\right)_M$	$\rho A_b (\Omega R)^2 R \frac{\partial C_Q/\sigma}{\partial \lambda'} \frac{\partial \lambda'}{\partial \dot{z}}$

Derivative	Equation
$\left(\frac{\partial N}{\partial r}\right)_M$	$2\rho A_b (\Omega R)^2 R^2 C_Q/\sigma$ (governed engine, counterclockwise rotation)
$\left(\frac{\partial N}{\partial \theta_0}\right)_M$	$\rho A_b (\Omega R)^2 R \frac{\partial C_Q/\sigma}{\partial \theta_0}$

Tail Rotor Derivatives near Hover

Derivative	Equation
$\left(\frac{\partial Y}{\partial j}\right)_T$	$\rho A_b(\Omega R)^2 \frac{\partial C_T/\sigma}{\partial \lambda'} \frac{\partial \lambda'}{\partial j}$
$\left(\frac{\partial Y}{\partial p}\right)_T$	$\left(\frac{\partial Y}{\partial j}\right)_T h_1$
$\left(\frac{\partial Y}{\partial r}\right)_T$	$-\left(\frac{\partial Y}{\partial j}\right)_T l_T$
$\left(\frac{\partial Y}{\partial \theta_0}\right)_T$	$\rho A_b(\Omega R)^2 \frac{\partial C_T/\sigma}{\partial \theta_0}$
$\left(\frac{\partial R}{\partial j}\right)_T$	$\left(\frac{\partial Y}{\partial j}\right)_T h_1$
$\left(\frac{\partial R}{\partial p}\right)_T$	$\left(\frac{\partial Y}{\partial p}\right)_T h_1$
$\left(\frac{\partial R}{\partial r}\right)_T$	$\left(\frac{\partial Y}{\partial r}\right)_T h_1$

Derivative	Equation
$\left(\frac{\partial R}{\partial \theta_0}\right)_T$	$\left(\frac{\partial Y}{\partial \theta_0}\right)_T h_T$
$\left(\frac{\partial M}{\partial j}\right)_T$	$\rho A_b(\Omega R)^2 R \frac{\partial C_Q/\sigma}{\partial \lambda'} \frac{\partial \lambda'}{\partial j}$
$\left(\frac{\partial M}{\partial r}\right)_T$	$-\left(\frac{\partial M}{\partial j}\right)_T l_T$
$\left(\frac{\partial M}{\partial \theta_0}\right)_T$	$\rho A_b(\Omega R)^2 R \frac{\partial C_Q/\sigma}{\partial \theta_0}$
$\left(\frac{\partial N}{\partial j}\right)_T$	$-\left(\frac{\partial Y}{\partial j}\right)_T l_T$
$\left(\frac{\partial N}{\partial p}\right)_T$	$-\left(\frac{\partial Y}{\partial p}\right)_T l_T$
$\left(\frac{\partial N}{\partial r}\right)_T$	$-\left(\frac{\partial Y}{\partial r}\right)_T l_T$
$\left(\frac{\partial N}{\partial \theta_0}\right)_T$	$-\left(\frac{\partial Y}{\partial \theta_0}\right)_T l_T$

Basic Main Rotor Derivatives in Forward Flight

Derivative	*Equation*
$\frac{\partial \mu}{\partial \dot{x}}$	$\frac{1}{\Omega R}$
$\frac{\partial \lambda'}{\partial \dot{x}}$	$\frac{1}{\Omega R}\left[\frac{}{\alpha_{1ff}} - \frac{\sigma}{2\mu}\left(\frac{\partial C_T/\sigma}{\partial \mu} - \frac{C_T/\sigma}{\mu}\right)\right]$
$\frac{\partial \lambda'}{\partial \dot{z}}$	$\frac{1}{\Omega R\left(1 + \frac{\partial C_T/\sigma}{\partial \lambda'}\frac{\sigma}{2\mu}\right)}$
$\frac{\partial \beta}{\partial \dot{y}}$	$\frac{1}{V}$
$\frac{\partial C_H/\sigma}{\partial a_{1_s}}, \frac{\partial C_Y/\sigma}{\partial b_{1_s}}$	$C_T/\sigma + \frac{a}{8}\lambda'$
$\frac{\partial a_{1_s}}{\partial q}$	$-\frac{16}{\gamma\Omega\left(1-\frac{e}{R}\right)^3\left(1-\frac{\mu^2}{2}\right)} - \frac{12\frac{e}{R}}{\gamma\Omega\left(1-\frac{e}{R}\right)^3\left(1-\frac{\mu^4}{4}\right)}$
$\frac{\partial a_{1_s}}{\partial p}$	$-\frac{1}{\Omega\left(1-\frac{\mu^2}{2}\right)} - \frac{192\frac{e}{R}}{\gamma^2\Omega\left(1-\frac{e}{R}\right)^3\left(1-\frac{\mu^4}{4}\right)}$
$\frac{\partial a_{1_s}}{\partial A_1}$	$\frac{12\frac{e}{R}\left(1+\frac{\mu^2}{2}\right)}{\gamma\left(1-\frac{e}{R}\right)^3\left(1-\frac{\mu^4}{4}\right)}$
$\frac{\partial a_{1_s}}{\partial B_1}$	$-\frac{\left(1+\frac{3}{2}\mu^2\right)}{1-\frac{\mu^2}{2}}$

Derivative	Equation
$\frac{\partial b_{1_s}}{\partial q}$	$-\frac{1}{\Omega\left(1+\frac{\mu^2}{2}\right)}+\frac{192\frac{\epsilon}{R}}{\gamma^2\Omega\left(1-\frac{\epsilon}{R}\right)^3\left(1-\frac{\mu^4}{4}\right)}$
$\frac{\partial b_{1_s}}{\partial p}$	$-\frac{16}{\gamma\Omega\left(1-\frac{\epsilon}{R}\right)^3\left(1+\frac{\mu^2}{2}\right)}-\frac{12\frac{\epsilon}{R}}{\gamma\Omega\left(1-\frac{\epsilon}{R}\right)^3\left(1-\frac{\mu^4}{4}\right)}$
$\frac{\partial b_{1_s}}{\partial A_1}$	1
$\frac{\partial b_{1_s}}{\partial B_1}$	$\frac{12\frac{\epsilon}{R}\left(1+\frac{3}{2}\mu^2\right)}{\gamma\left(1-\frac{\epsilon}{R}\right)^3\left(1-\frac{\mu^4}{4}\right)}$
$\frac{dM}{da_{1_s}}, \frac{dR}{db_{1_s}}$	$\frac{\frac{3}{4}\frac{\epsilon}{R}A_1\rho R(\Omega R)^2 a}{\gamma}$

Main Rotor Derivatives in Forward Flight

Derivative	Equation
$\left(\frac{\partial X}{\partial \dot{x}}\right)_M$	$-\rho A_1(\Omega R)^2\left\{\left[\frac{\partial C_H/\sigma}{\partial \mu}+\frac{\partial C_H/\sigma}{\partial a_{1_s}}\frac{\partial a_{1_s}}{\partial \mu}+(\overline{a_{1_s}}+i_M)\frac{\partial C_T/\sigma}{\partial \mu}\right]\frac{\partial \mu}{\partial \dot{x}}\right.$ $\left.+\left[\frac{\partial C_H/\sigma}{\partial \lambda'}+\frac{\partial C_H/\sigma}{\partial a_{1_s}}\frac{\partial a_{1_s}}{\partial \lambda'}+(\overline{a_{1_s}}+i_M)\frac{\partial C_T/\sigma}{\partial \lambda'}\right]\frac{\partial \lambda'}{\partial \dot{x}}\right\}$
$\left(\frac{\partial X}{\partial \dot{y}}\right)_M$	$+\rho A_1(\Omega R)^2\overline{C_T}/\sigma(A_1-\overline{b_{1_s}})\frac{\partial \beta}{\partial \dot{y}}$
$\left(\frac{\partial X}{\partial \dot{z}}\right)_M$	$-\rho A_1(\Omega R)^2\left[\frac{\partial C_H/\sigma}{\partial \lambda'}+\overline{C_T}/\sigma\frac{\partial a_{1_s}}{\partial \lambda'}+(\overline{a_{1_s}}+i_M)\frac{\partial C_T/\sigma}{\partial \lambda'}\right]\frac{\partial \lambda'}{\partial \dot{z}}$

Derivative	Equation
$\left(\frac{\partial X}{\partial q}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial q} - \left(\frac{\partial X}{\partial \dot{x}}\right)_M b_M$
$\left(\frac{\partial X}{\partial p}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial p} - \left(\frac{\partial X}{\partial \dot{y}}\right) b_M$
$\left(\frac{\partial X}{\partial \theta_0}\right)_M$	$-\rho A_b(\Omega R)^2 \left[\frac{\partial C_H/\sigma}{\partial \theta_0} + \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial \theta_0} + (\overline{a_{1_s}} + i_M) \frac{\partial C_T/\sigma}{\partial \theta_0}\right]$
$\left(\frac{\partial X}{\partial A_1}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial A_1}$
$\left(\frac{\partial X}{\partial B_1}\right)_M$	$-\rho A_b(\Omega R)^2 \frac{\partial C_H/\sigma}{\partial a_{1_s}} \frac{\partial a_{1_s}}{\partial B_1}$
$\left(\frac{\partial Y}{\partial \dot{x}}\right)_M$	$\rho A_b(\Omega R)^2 \left[\frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial b_{1_s}}{\partial \mu} + \overline{b_{1_s}} \frac{\partial C_T/\sigma}{\partial \mu}\right] \frac{\partial \mu}{\partial \dot{x}}$
$\left(\frac{\partial Y}{\partial \dot{y}}\right)_M$	$-\rho A_b(\Omega R)^2 \left[\overline{C_H}/\sigma + \overline{C_T}/\sigma(B_1 + \overline{a_{1_s}})\right] \frac{\partial \beta}{\partial \dot{y}}$
$\left(\frac{\partial Y}{\partial \dot{z}}\right)_M$	$\rho A_b(\Omega R)^2 \frac{\partial b_{1_s}}{\partial \lambda'} \frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial \lambda'}{\partial \dot{z}}$
$\left(\frac{\partial Y}{\partial q}\right)_M$	$\rho A_b(\Omega R)^2 \frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial b_{1_s}}{\partial q} + \left(\frac{\partial Y}{\partial \dot{x}}\right) b_M$
$\left(\frac{\partial Y}{\partial p}\right)_M$	$\rho A_b(\Omega R)^2 \frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial b_{1_s}}{\partial p} + \left(\frac{\partial Y}{\partial \dot{y}}\right) b_M$
$\left(\frac{\partial Y}{\partial \theta_0}\right)_M$	$\rho A_b(\Omega R)^2 \left[\frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial b_{1_s}}{\partial \theta_0} + \overline{b_{1_s}} \frac{\partial C_T/\sigma}{\partial \theta_0}\right]$
$\left(\frac{\partial Y}{\partial A_1}\right)_M$	$\rho A_b(\Omega R)^2 \frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial b_{1_s}}{\partial A_1}$
$\left(\frac{\partial Y}{\partial B_1}\right)_M$	$\rho A_b(\Omega R)^2 \frac{\partial C_Y/\sigma}{\partial b_{1_s}} \frac{\partial b_{1_s}}{\partial B_1}$

Derivative	Equation
$\left(\frac{\partial Z}{\partial \dot{x}}\right)_M$	$-\rho A_b(\Omega R)^2\left[\frac{\partial C_T/\sigma}{\partial \mu}\frac{\partial \mu}{\partial \dot{x}}+\frac{\partial C_T/\sigma}{\partial \lambda'}\frac{\partial \lambda'}{\partial \dot{x}}\right]$
$\left(\frac{\partial Z}{\partial \dot{z}}\right)_M$	$-\rho A_b(\Omega R)^2\frac{\partial C_T/\sigma}{\partial \lambda'}\frac{\partial \lambda'}{\partial \dot{z}}$
$\left(\frac{\partial Z}{\partial r}\right)_M$	$\frac{2}{\Omega}\rho A_b(\Omega R)^2\overline{C_T/\sigma}$
$\left(\frac{\partial Z}{\partial \theta_0}\right)_M$	$-\rho A_b(\Omega R)^2\frac{\partial C_T/\sigma}{\partial \theta_0}$
$\left(\frac{\partial z}{\partial B_1}\right)_M$	$-\rho A_b(\Omega R)^2\frac{\partial C_T/\sigma}{\partial \lambda'}\frac{\partial \lambda'}{\partial a_1}\frac{\partial a_1}{\partial B_1}$
$\left(\frac{\partial R}{\partial \dot{x}}\right)_M$	$\left(\frac{dR}{db_1}\right)_M\left(\frac{\partial b_1}{\partial \mu}\right)\left(\frac{\partial \mu}{\partial \dot{x}}\right)+\left(\frac{\partial Y}{\partial \dot{x}}\right)_M b_M+\left(\frac{\partial Z}{\partial \dot{x}}\right)_M y_M$
$\left(\frac{\partial R}{\partial \dot{y}}\right)_M$	$-\left(\frac{dR}{db_1}\right)_M\left(\overline{B_1+a_1}\right)\frac{\partial \beta}{\partial \dot{y}}+\left(\frac{\partial Y}{\partial \dot{y}}\right)_M b_M$
$\left(\frac{\partial R}{\partial \dot{z}}\right)_M$	$\left(\frac{dR}{db_1}\right)_M\left(\frac{\partial b_1}{\partial \lambda'}\right)\left(\frac{\partial \lambda'}{\partial \dot{z}}\right)+\left(\frac{\partial Y}{\partial \dot{z}}\right)_M b_M+\left(\frac{\partial Z}{\partial \dot{z}}\right)_M y_M$
$\left(\frac{\partial R}{\partial q}\right)_M$	$\left(\frac{dR}{db_1}\right)_M\left(\frac{\partial b_1}{\partial q}\right)+\left(\frac{\partial Y}{\partial q}\right)_M b_M$
$\left(\frac{\partial R}{\partial p}\right)_M$	$\left(\frac{dR}{db_1}\right)_M\left(\frac{\partial b_1}{\partial p}\right)+\left(\frac{\partial Y}{\partial p}\right)_M b_M$
$\left(\frac{\partial R}{\partial \theta_0}\right)_M$	$\left(\frac{dR}{db_1}\right)_M\left(\frac{\partial b_1}{\partial \theta_0}\right)+\left(\frac{\partial Y}{\partial \theta_0}\right)_M b_M+\left(\frac{\partial Z}{\partial \theta_0}\right)_M y_M$
$\left(\frac{\partial R}{\partial A_1}\right)_M$	$\left(\frac{dR}{db_1}\right)_M\left(\frac{\partial b_1}{\partial A_1}\right)+\left(\frac{\partial Y}{\partial A_1}\right)_M b_M$

Derivative	Equation
$\left(\frac{\partial R}{\partial B_1}\right)_M$	$\left(\frac{dR}{db_{1_s}}\right)_M\left(\frac{\partial b_{1_s}}{\partial B_1}\right)+\left(\frac{\partial Y}{\partial B_1}\right)_M b_M$
$\left(\frac{\partial M}{\partial \dot{x}}\right)_M$	$\left(\frac{dM}{da_{1_s}}\right)_M\left[\frac{\partial a_{1_s}}{\partial \mu}\frac{\partial \mu}{\partial \dot{x}}+\frac{\partial a_{1_s}}{\partial \lambda'}\frac{\partial \lambda'}{\partial \dot{x}}\right]-\left(\frac{\partial X}{\partial \dot{x}}\right)_M b_M+\left(\frac{\partial Z}{\partial \dot{x}}\right)_M l_M$
$\left(\frac{\partial M}{\partial \dot{y}}\right)_M$	$\left(\frac{\partial N}{\partial \dot{y}}\right)_M b_M+\left(\frac{dm}{da_{1_s}}\right)\left(A_1-\overline{b_{1_s}}\right)\frac{\partial \beta}{\partial \dot{y}}$
$\left(\frac{\partial M}{\partial \dot{z}}\right)_M$	$\left(\frac{dM}{da_{1_s}}\right)_M\frac{\partial a_{1_s}}{\partial \lambda'}\frac{\partial \lambda'}{\partial \dot{z}}-\left(\frac{\partial X}{\partial \dot{z}}\right)_M b_M+\left(\frac{\partial Z}{\partial \dot{z}}\right)_M l_M$
$\left(\frac{\partial M}{\partial q}\right)_M$	$\left(\frac{dM}{da_{1_s}}\right)_M\left(\frac{\partial a_{1_s}}{\partial q}\right)-\left(\frac{\partial X}{\partial q}\right)_M b_M$
$\left(\frac{\partial M}{\partial p}\right)_M$	$\left(\frac{dM}{da_{1_s}}\right)_M\left(\frac{\partial a_{1_s}}{\partial p}\right)-\left(\frac{\partial X}{\partial p}\right)_M b_M$
$\left(\frac{\partial M}{\partial \theta_0}\right)_M$	$\left(\frac{dM}{da_{1_s}}\right)_M\left(\frac{\partial a_{1_s}}{\partial \theta_0}\right)-\left(\frac{\partial X}{\partial \theta_0}\right)_M b_M+\left(\frac{\partial Z}{\partial \theta_0}\right)_M l_M$
$\left(\frac{\partial M}{\partial A_1}\right)_M$	$\left(\frac{dM}{da_{1_s}}\right)_M\left(\frac{\partial a_{1_s}}{\partial A_1}\right)-\left(\frac{\partial X}{\partial A_1}\right)_M b_M$
$\left(\frac{\partial M}{\partial B_1}\right)_M$	$\left(\frac{dM}{da_{1_s}}\right)_M\left(\frac{\partial a_{1_s}}{\partial B_1}\right)-\left(\frac{\partial X}{\partial B_1}\right)_M b_M$
$\left(\frac{\partial N}{\partial \dot{x}}\right)_M$	$\rho A_b(\Omega R)^2 R\left[\frac{\partial C_Q/\sigma}{\partial \mu}\frac{\partial \mu}{\partial \dot{x}}+\frac{\partial C_Q/\sigma}{\partial \lambda'}\frac{\partial \lambda'}{\partial \dot{x}}\right]$
$\left(\frac{\partial N}{\partial \dot{z}}\right)_M$	$\rho A_b(\Omega R)^2 R\left(\frac{\partial C_Q/\sigma}{\partial \lambda'}\right)\frac{\partial \lambda'}{\partial \dot{z}}$
$\left(\frac{\partial N}{\partial r}\right)_M$	$-\frac{2}{\Omega}\rho A_b(\Omega R)^2 R\overline{C_Q/\sigma}$
$\left(\frac{\partial N}{\partial \theta_0}\right)_M$	$\rho A_b(\Omega R)^2 R\left(\frac{\partial C_Q/\sigma}{\partial \theta_0}\right)$

Basic Tail Rotor Derivatives in Forward Flight

Derivative	Equation
$\frac{\partial \mu}{\partial \dot{x}}$	$\frac{1}{\Omega R}$
$\frac{\partial \lambda'}{\partial \dot{x}}$	$\frac{1}{\Omega R}\left[a_{1_s} - \frac{\sigma}{2\mu}\left(\frac{\partial C_T/\sigma}{\partial \mu} - \frac{C_T/\sigma}{\mu}\right)\right]$
$\frac{\partial \lambda'}{\partial \dot{y}}$	$\frac{1}{\Omega R}\left(1 + \frac{\partial C_T/\sigma}{\partial \lambda'}\frac{\sigma}{2\mu}\right)$
$\frac{\partial \beta}{\partial \dot{y}}$	$\frac{1}{\dot{V}}$

Tail Rotor Derivatives in Forward Flight

Derivative	Equation
$\left(\frac{\partial Y'}{\partial \dot{x}}\right)_T$	$\rho A_b(\Omega R)^2\left[\frac{\partial C_T/\sigma}{\partial \mu}\right]\frac{\partial \mu}{\partial \dot{x}}$
$\left(\frac{\partial Y'}{\partial \dot{y}}\right)_T$	$\rho A_b(\Omega R)^2\left[\frac{\partial C_T/\sigma}{\partial \lambda'}\right]\frac{\partial \lambda'}{\partial \dot{y}}$
$\left(\frac{\partial Y'}{\partial p}\right)_T$	$\left(\frac{\partial Y'}{\partial \dot{y}}\right)_T h_T$
$\left(\frac{\partial Y'}{\partial r}\right)_T$	$\left(\frac{\partial Y'}{\partial \dot{y}}\right)_T l_T$
$\left(\frac{\partial Y'}{\partial \theta_0}\right)_T$	$\rho A_b(\Omega R)^2\frac{\partial C_T/\sigma}{\partial \theta_0}$

Derivative	Equation
$\left(\frac{\partial R}{\partial \dot{x}}\right)_T$	$\left(\frac{\partial Y'}{\partial \dot{x}}\right)_T b_T$
$\left(\frac{\partial R}{\partial \dot{y}}\right)_T$	$\left(\frac{\partial Y'}{\partial \dot{y}}\right)_T b_T$
$\left(\frac{\partial R}{\partial p}\right)_T$	$\left(\frac{\partial Y'}{\partial p}\right)_T b_T$
$\left(\frac{\partial R}{\partial r}\right)_T$	$\left(\frac{\partial Y'}{\partial r}\right)_T b_T$
$\left(\frac{\partial R}{\partial \theta_0}\right)_T$	$\left(\frac{\partial Y'}{\partial \theta_0}\right)_T b_T$
$\left(\frac{\partial N}{\partial \dot{x}}\right)_T$	$-\left(\frac{\partial Y'}{\partial \dot{x}}\right)_T l_T$
$\left(\frac{\partial N}{\partial \dot{y}}\right)_T$	$-\left(\frac{\partial Y'}{\partial \dot{y}}\right)_T l_T$
$\left(\frac{\partial N}{\partial p}\right)_T$	$-\left(\frac{\partial Y'}{\partial p}\right)_T l_T$
$\left(\frac{\partial N}{\partial r}\right)_T$	$-\left(\frac{\partial Y'}{\partial r}\right)_T l_T$
$\left(\frac{\partial N}{\partial \theta_0}\right)_T$	$-\left(\frac{\partial Y'}{\partial \theta_0}\right)_T l_T$

Nondimensional Horizontal Stabilizer Derivatives

Derivative	*Equation*
$\frac{\partial \gamma}{\partial \dot{z}}$	$-\frac{1}{V}$
$\frac{\partial c_{M_H}}{\partial \dot{x}}$	$\frac{r_H}{r_1}\frac{1}{4qA_{pH}}\left[-\left(\frac{\partial Z}{\partial \dot{x}}\right)_M + \frac{2\bar{Z}_M}{V}\right]$
$\frac{\partial c_{M_H}}{\partial \dot{z}}$	$-\frac{r_H}{r_1}\frac{1}{4qA_{pH}}\left(\frac{\partial Z}{\partial \dot{z}}\right)_M$
$\frac{\partial c_{T_H}}{\partial \dot{z}}$	$\frac{dc_T}{d\alpha_T}\left[\frac{1}{4qA_{pH}}\left(\frac{\partial Z}{\partial \dot{z}}\right)_M - \frac{\partial \gamma}{\partial \dot{z}}\right]$
$\frac{\partial \alpha_H}{\partial \dot{x}}$	$\frac{\partial c_{M_H}}{\partial \dot{x}}$
$\frac{\partial \alpha_H}{\partial \dot{z}}$	$-\left[\frac{\partial c_{M_H}}{\partial \dot{z}} + \frac{\partial c_{T_H}}{\partial \dot{z}} + \frac{\partial \gamma}{\partial \dot{z}}\right]$
$\frac{\partial \alpha_H}{\partial z}$	$-\left(\frac{\partial c_{M_H}}{\partial \dot{z}}\right)\frac{l_H}{V}$

Horizontal Stabilizer Derivatives in Forward Flight

Derivative	*Equation*
$\left(\frac{\partial X}{\partial \dot{x}}\right)_H$	$\frac{2}{V}X_H + \left(\frac{q_H}{q}\right)qA_Ha_H\left\{(\bar{\alpha}_H - \alpha_{LO})\left[1 - \frac{2a_H(1+\delta_t)}{\pi A.R.}\right] + (\bar{\alpha}_H - i_H)\right\}\frac{\partial \alpha_H}{\partial \dot{x}}$
$\left(\frac{\partial X}{\partial \dot{z}}\right)_H$	$\left(\frac{q_H}{q}\right)qA_Ha_H\left\{(\bar{\alpha}_H - \alpha_{LO})\left[1 - \frac{2a_H(1+\delta_t)}{\pi A.R.}\right] + (\alpha_H - i_H)\right\}\frac{\partial \alpha_H}{\partial \dot{z}}$
$\left(\frac{\partial X}{\partial z}\right)_H$	$\left(\frac{\partial X}{\partial \dot{z}}\right)_H \frac{\frac{\partial \alpha_H}{\partial z}}{\frac{\partial \alpha_H}{\partial \dot{z}}}$

Derivative	Equation
$\left(\frac{\partial Z}{\partial \dot{x}}\right)_H$	$\frac{2}{V}\bar{Z}_H - \left(\frac{q_H}{q}\right) q A_H a_H \left\{ 1 + \frac{a_H(1+\delta_1)}{\pi A.R.} \left[2(\bar{\alpha}_H - \alpha_{LO})(\bar{\alpha}_H - i_H) + (\bar{\alpha}_H - \alpha_{LO})^2 \right] + C_{D_0} \right\} \frac{\partial \alpha_H}{\partial \dot{x}}$
$\left(\frac{\partial Z}{\partial \dot{z}}\right)_H$	$-\left(\frac{q_H}{q}\right) q A_H a_H \left\{ 1 + \frac{a_H(1+\delta_1)}{\pi A.R.} \left[2(\bar{\alpha}_H - \alpha_{LO})(\bar{\alpha}_H - i_H) + (\bar{\alpha}_H \alpha_{LO})^2 \right] + C_{D_0} \right\} \frac{\partial \alpha_H}{\partial \dot{z}}$
$\left(\frac{\partial Z}{\partial \xi}\right)_H$	$\left(\frac{\partial Z}{\partial \dot{z}}\right)_H \frac{\frac{\partial \alpha_H}{\partial \xi}}{\frac{\partial \alpha_H}{\partial \dot{z}}}$
$\left(\frac{\partial Z}{\partial q}\right)_H$	$\left(\frac{\partial Z}{\partial \dot{z}}\right)_H l_H$
$\left(\frac{\partial M}{\partial \dot{x}}\right)_H$	$-\left(\frac{\partial X}{\partial \dot{x}}\right)_H h_H + \left(\frac{\partial Z}{\partial \dot{x}}\right)_H l_H$
$\left(\frac{\partial M}{\partial \dot{z}}\right)_H$	$-\left(\frac{\partial X}{\partial \dot{z}}\right)_H h_H + \left(\frac{\partial Z}{\partial \dot{z}}\right)_H l_H$
$\left(\frac{\partial M}{\partial \xi}\right)_H$	$-\left(\frac{\partial X}{\partial \xi}\right)_H h_H + \left(\frac{\partial Z}{\partial \xi}\right) l_H$
$\left(\frac{\partial M}{\partial q}\right)_H$	$\left(\frac{\partial Z}{\partial q}\right)_H l_H$

Nondimensional Vertical Stabilizer Derivatives

Derivative	*Equation*
$\frac{\partial \beta}{\partial \dot{y}}$	$\frac{1}{V}$
$\frac{\partial M_T}{\partial \dot{x}}$	$\frac{-1}{\left(\frac{q_V}{q}\right) q A_V} \left[\left(\frac{\partial Y}{\partial \dot{x}} \right)_T - \frac{2T_T}{V} \right]$
$\frac{\partial \alpha_V}{\partial \dot{x}}$	$\frac{\partial \eta_T}{\partial \dot{x}}$
$\frac{\partial M_T}{\partial \dot{y}}$	$\frac{\left(\frac{\partial Y}{\partial \dot{y}} \right)_T}{\left(\frac{q_V}{q} \right) q A_V}$
$\frac{\partial M_\beta}{\partial \dot{y}}$	$\frac{\partial \eta_F}{\partial \beta} \frac{\partial \beta}{\partial \dot{y}}$
$\frac{\partial \alpha_V}{\partial \dot{y}}$	$-\left(\frac{\partial \beta}{\partial \dot{y}} + \frac{\partial \eta_{T_V}}{\partial \dot{y}} + \frac{\partial \eta_F}{\partial \dot{y}} \right)$

Vertical Stabilizer Derivatives in Forward Flight

Derivative	*Equation*
$\left(\frac{\partial X}{\partial \dot{x}} \right)_V$	$\frac{2}{V} \left[\bar{X}_V + 2 \Delta \bar{D}_{V_{trim}} \right]$
$\left(\frac{\partial X}{\partial \dot{y}} \right)_V$	$\frac{q_V}{q} q A_V a_V \left\{ (\bar{\alpha}_V - \alpha_{1_0}) \left[1 - \frac{2a_V(1+\delta_1)}{\pi AR} \right] + \alpha_V \right\} \frac{\partial \alpha_V}{\partial \dot{y}} + \Delta \bar{D}_{V_{trim}} \left[\frac{1}{\bar{V}_V} \left(\frac{\partial Y}{\partial \dot{y}} \right)_V + \frac{1}{\bar{V}_T} \left(\frac{\partial Y}{\partial \dot{y}} \right)_T \right]$
$\left(\frac{\partial Y}{\partial \dot{x}} \right)_V$	$\frac{2}{V} \bar{Y}_V + \left(\frac{q_V}{q} \right) q A_V a_V \frac{\partial \alpha_V}{\partial \dot{x}}$
$\left(\frac{\partial Y}{\partial \dot{y}} \right)_V$	$\frac{1}{1 - \left(\frac{\Delta \bar{D}_{V_{trim}}}{\bar{Y}_V} \right)} \left\{ \left(\frac{q_V}{q} \right) q A_V a_V \frac{\partial \alpha_V}{\partial \dot{y}} + \Delta \bar{D}_{V_{trim}} \left[-\frac{2}{V} + \frac{1}{\bar{V}_T} \left(\frac{\partial Y}{\partial \dot{y}} \right)_T \right] \right\}$

Derivative	Equation
$\left(\frac{\partial Y}{\partial p}\right)_{V}$	$\left(\frac{\partial Y}{\partial \dot{y}}\right)_{V} h_{1}$
$\left(\frac{\partial Y}{\partial r}\right)_{V}$	$-\left(\frac{\partial Y}{\partial \dot{y}}\right)_{V} l_{1}$
$\left(\frac{\partial R}{\partial \dot{x}}\right)_{V}$	$\left(\frac{\partial Y}{\partial \dot{x}}\right)_{V} h_{1}$
$\left(\frac{\partial R}{\partial \dot{y}}\right)_{V}$	$\left(\frac{\partial Y}{\partial \dot{y}}\right)_{V} h_{1}$
$\left(\frac{\partial R}{\partial p}\right)_{V}$	$\left(\frac{\partial Y}{\partial p}\right)_{V} h_{1}$
$\left(\frac{\partial R}{\partial r}\right)_{V}$	$\left(\frac{\partial Y}{\partial r}\right)_{V} h_{1}$
$\left(\frac{\partial N}{\partial \dot{x}}\right)_{V}$	$-\left(\frac{\partial Y}{\partial \dot{x}}\right)_{V} l_{1}$
$\left(\frac{\partial N}{\partial \dot{y}}\right)_{V}$	$-\left(\frac{\partial Y}{\partial \dot{y}}\right)_{V} l_{1}$
$\left(\frac{\partial N}{\partial p}\right)_{V}$	$-\left(\frac{\partial Y}{\partial p}\right)_{V} l_{1}$
$\left(\frac{\partial N}{\partial r}\right)_{V}$	$-\left(\frac{\partial Y}{\partial r}\right)_{V} l_{1}$

Nondimensional Fuselage Derivatives

Derivative	Equation
$\frac{\partial \gamma}{\partial \dot{z}}$	$-\frac{1}{V}$
$\frac{\partial \beta}{\partial \dot{y}}$	$\frac{1}{V}$
$\frac{\partial \epsilon_{M_F}}{\partial \dot{x}}$	$\frac{v_F}{v_1}\frac{1}{4q\cdot 1}\left[\left(\frac{\partial Z}{\partial \dot{x}}\right)_M + \frac{2\bar{Z}_M}{V}\right]$
$\frac{\partial \epsilon_{M_F}}{\partial \dot{z}}$	$\frac{v_F}{v_1}\frac{1}{4q\cdot 1}\left(\frac{\partial Z}{\partial \dot{z}}\right)_M$
$\frac{\partial \alpha_F}{\partial \dot{x}}$	$\frac{\partial \epsilon_{M_F}}{\partial \dot{x}}$
$\frac{\partial \alpha_F}{\partial \dot{z}}$	$\left(\frac{\partial \epsilon_{M_F}}{\partial \dot{z}} + \frac{\partial \gamma}{\partial \dot{z}}\right)$
$\frac{\partial f}{\partial \alpha_F}$, $\frac{\partial L/q}{\partial \alpha_F}$, $\frac{\partial S.F./q}{\partial \beta}$, $\frac{\partial M/q}{\partial \alpha_F}$, $\frac{\partial N/q}{\partial \beta}$, $\frac{\partial R/q}{\partial \beta}$	From curves

Fuselage Derivatives in Forward Flight

Derivative	Equation
$\left(\frac{\partial X}{\partial \dot{x}}\right)_F$	$\frac{2}{V}\overline{X_F}$
$\left(\frac{\partial X}{\partial \dot{z}}\right)_F$	$\left(\overline{L_F} - q\frac{\partial f}{\partial \alpha_F}\right)\frac{\partial \alpha_F}{\partial \dot{z}}$
$\left(\frac{\partial Y}{\partial \dot{y}}\right)_F$	$\frac{1}{V}\left(q\frac{\partial S.F./q}{\partial \beta} - \overline{D_F}\right)$
$\left(\frac{\partial Z}{\partial \dot{x}}\right)_F$	$\frac{2}{V}\overline{Z_F}$
$\left(\frac{\partial Z}{\partial \dot{z}}\right)_F$	$\left(-\overline{D_F} - q\frac{\partial L/q}{\partial \alpha_F}\right)\frac{\partial \alpha_F}{\partial \dot{z}}$
$\left(\frac{\partial R}{\partial \dot{y}}\right)_F$	$q\frac{\partial R/q}{\partial \beta}\frac{\partial \beta}{\partial \dot{y}}$
$\left(\frac{\partial M}{\partial \dot{x}}\right)_F$	$\frac{2}{V}\overline{M_F} + q\frac{\partial M/q}{\partial \alpha_F}\frac{\partial \alpha_F}{\partial \dot{x}}$
$\left(\frac{\partial M}{\partial \dot{z}}\right)_F$	$q\frac{\partial M/q}{\partial \alpha_F}\frac{\partial \alpha_F}{\partial \dot{z}}$
$\left(\frac{\partial N}{\partial \dot{y}}\right)_F$	$q\frac{\partial N/q}{\partial \beta}\frac{\partial \beta}{\partial \dot{y}}$

APPENDIX D. VARIABLE LIST.

Variable list for JANRAD Stability Routine

A - rotor disk area
Ab - area of blades
Abt - area of tail rotor blades
Ap - projected area of fuselage under rotor
A1 - lateral cyclic pitch
a1s - longitudinal flapping
a - main rotor blade lift curve slope
Ah - area of the horizontal stab
ah - lift curve slope of the horizontal stab
alph - angle of attack of horizontal stab
alphas - angle of attack WRT shaft axis
alploh - alpha-zero-lift of the horizontal stab (at the trim condition)
alplov - alpha-zero-lift of the vertical stab
alplow - alpha-zero-lift of the wing
alpv - angle of attack of vertical stab
alpw - angle of attack of wing
altpp - mean alpha TPP
ao - main rotor coning angle
aot - tail rotor coning angle
At - disc area of the tail rotor
at - tail rotor blade lift curve slope
at - tail rotor blade lift curve slope
Av - area of the vertical stab
av - lift curve slope of the vertical stab
Aw - area of the wing
aw - lift curve slope of wing
B1 - longitudinal cyclic pitch
b1s -lateral flapping
beta - trim sideslip angle
bh - span of the horizontal stab
bv - semispan of the vertical stab
bw - span of the wing
c - main rotor chord
cdoh - Cdo of the horizontal stab
cdow - Cdo of the wing
ch - coefficient of ????
chsig - Ch/sigma
clvertmax - maximum cl of the vertical stab with full rudder defl
cmu - circulation coefficient for NOTAR™
cmun - design Cmu for NOTAR™ boom in hover (.3-.6)
cot - tail rotor chord
cpipitch - Control power/inch pitch

cpiroll - Control power/inch pitch
cpiyaw - Control power/inch pitch
cppitch - Control power pitch
cproll - Control power roll
cpyaw - Control power yaw
ct - coefficient of thrust
ctsig - Ct/sigma
ctsigt - Ct/sigma of the tail rotor
ctr - wing root chord
ctw - wing tip chord
cq - coefficient of torque
cqsig - Cq/sigma
deldv - biplane effect (mutual interference of tail rotor and vertical stab)
delih - span efficiency factor of the horizontal stab
deliv - span efficiency factor of the vertical stab
deliw - span efficiency factor of the wing
delta3 - tail rotor delta-3 angle
delvmax - maximum rudder deflection on the vertical stab
desdmdq - designed pitch damping (dM/dq)
desdndr - designed yaw damping (dN/dr)
desdrdp - designed roll damping (dR/dp)
detafdalpfh - fuselage downwash ratio for the horizontal stab
detafdalpfw - fuselage downwash ratio for the wing
detafdbeta - fuselage sideslip ratio (= depsilondalpha)
dian - NOTAR™ boom diameter
e - blade offset
GW - gross weight
g - acceleration due to gravity
hh - horizontal stab vertical offset
hhd - height from waterline to horizontal tail
hm - main rotor vertical offset
hmd - height from waterline to main rotor hub
hslot - total of slot heights for NOTAR™ boom
ht - tail rotor vertical offset
htd - height from waterline to tail rotor hub
htn - NOTAR™ thruster vertical offset
hv - vertical stab vertical offset
hvd - height from waterline to vertical fin
hw - wing vertical offset
hwd - height from waterline to wing
htnd - height from waterline to NOTAR™
Ib - blade flapping inertia
Ibt - tail rotor blade flapping moment of inertia
Ixx - mass moment of inertia about x axis
Iyy - mass moment of inertia about y axis
Ixz - mass moment of inertia about xz plane
Izz - mass moment of inertia about z axis
Ic - Ixx*Izz

i - shaft incidence main rotor
ih - angle of incidence of horizontal stab
iw - angle of incidence of wing
lamp - lambda' - inflow ratio WRT TPP
lh - horizontal stab longitudinal offset
lhd - fuselage station of horizontal tail
lm - main rotor longitudinal offset
lmd - fuselage station of main rotor hub
lt - tail rotor longitudinal offset
ltd - fuselage station of tail rotor hub
ltn - NOTAR™ thruster longitudinal offset
ltnd - fuselage station of NOTAR™ boom
lttnd - fuselage station of NOTAR™ thruster
lv - vertical stab longitudinal offset
lvd - fuselage station of vertical tail
lw - wing longitudinal offset
lwd - fuselage station of wing
lockno - main rotor lock number
locknot - tail rotor lock number
lt - tail rotor longitudinal offset
lv - vertical stab longitudinal offset
maxr - maximum rudder deflection
mu - advance ratio
mut - tail rotor advance ratio
n - a counter in CTPLOTS.M
nt - number of tail rotor blades
ohm - omega
ohmt - tail rotor omega
phin - NOTAR™ thruster sleeve rotation angle
pho - roll trim attitude (euler angle)
prpitch - Pilot rating (dM/dq)/Iyy)
prroll - Pilot rating (dR/dp)/Ixx)
pryaw - Pilot rating (dN/dr)/Izz)
q - dynamic pressure, 1/2 rho V^2
qhq - horizontal tail dynamic press ratio (qh/q)
qvin - dynamic press due to downwash at the NOTAR™ slots
qvq - vertical tail dynamic press ratio (qv/q)
R - blade radius
Rt - tail rotor blade radius
rho - air density
sidearm - maximum pedal travel or twist grip deflection
sigma - solidity
sigmat - tail rotor solidity
swirl - mean swirl angle off main rotor measured at the NOTAR™ slots
T - thrust
Tt - thrust of the tail rotor
theta1 - main rotor blade twist
theta1t - tail rotor blade twist

thetao - blade pitch at root
thetat - induced angle at the tip
tho - trim pitch attitude (euler angle)
V - forward velocity
vl - induced flow velocity at a hover
vlf - approximation for vl at forward airspeed
vfvl - rotor downwash ratio for the fuselage
vhvl - rotor downwash ratio for horizontal stab
vwvl - rotor downwash ratio for wing
uo - x initial velocity
vo - y initial velocity
wo - z initial velocity
xcg - height from waterline to cg
xcouple - designed cross coupling
Ytmaxn - maximum Y force from NOTAR™ thruster
ycg - length from buttline to cg
Yv - side-force of vertical stab
yhd - length from buttline to horizontal tail
ym - main rotor lateral offset
ymd - length from buttline to main rotor hub
ytd - length from buttline to tail rotor hub
ytnd - length from buttline to NOTAR™
ywd - length from buttline to wing
yvd - length from buttline to verticle tail
zcg - fuselage station of cg
Zh - vertical force of horizontal stab
Zw - vertical force of wing

CRUISE

CRUISE -- BASIC MR Derivatives

dalda - dals/dAl
daldb - dals/dBl
daldlamp - dals/d(lambda')
daldmu - dals/dmu
daldp - dals/dp
daldq - dals/dq
daldtheto - dals/dtheta(o)
dbetdydot - d(Beta)/dYdot
dblda - dbls/dAl
dbldb - dbls/dBl
dbldlamp - dbls/d(lambda')
dbldmu - dbls/dmu
dbldp - dbls/dp
dbldq - dbls/dq
dbldtheto - dbls/dtheta(o)
dchsigda - d(Ch/sigma)/dals

dchsigdb - d(Ch/sigma)/db1s
dchsigdlamp - d(Ch/sigma)/d(lambda')
dchsigdmu - d(Ch/sigma)/dmu
dchsigdtheto - d(Ch/sigma)/dtheta(o)
dctsigdlamp - d(Ct/sigma)/d(lambda')
dctsigdmu - d(Ct/sigma)/dmu
dctsigdtheto - d(Ct/sigma)/dtheta(o)
dcqsigdlamp - d(Cq/sigma)/d(lambda')
dcqsigdmu - d(Cq/sigma)/dmu
dcqsigdtheto - d(Cq/sigma)/dtheta(o)
dcysigdb - d(Cy/sigma)/db1s
dlampdxdot - d(lambda')/dXdot
dlampdydot - d(lambda')/dYdot
dlampdzdot - d(lambda')/dZdot
dmda1sm - dM/da1s - main rotor stiffness
dmudxdot - dmu/dxdot
drdb1sm - dR/dB1s - main rotor stiffness

CRUISE -- MR derivatives

dmda1m - dM/dA1
dmdb1m - dM/dB1
dmdpm - dM/dp
dmdqm - dM/dq
dmdthetom - dM/dtheta(0)
dmdxdotm - dM/dXdot
dmdydotm - dM/dYdot
dmdzdotm - dM/dZdot
dndrm - dN/dr
dndtheom - dN/dtheta(0)
dndxdotm - dN/dXdot
dndzdotm - dN/dZdot
drda1m - dR/dA1
drdb1m - dR/dB1
drdpm - dR/dp
drdqm - dR/dq
drdthetom - dR/dtheta(0)
drdxdotm - dR/dXdot
drdydotm - dR/dYdot
drdzdotm - dR/dZdot
dxda1m - dX/dA1
dxdb1m - dX/dB1
dxdpm - dX/dp
dxdqm - dX/dq
dxdthetom - dX/dtheta0
dxdxdotm - dX/dXdot
dxdydotm - dX/dYdot
dxdzdot - dX/dZdot

dydalm - dY/dAl
dydblm - dY/dBl
dydpm - dY/dp
dydqm - dY/dq
dydthetom - dY/dtheta(0)
dydxdotm - dY/dXdot
dydydotm - dY/dYdot
dydzdotm - dY/dZdot
dzdblm - dZ/dbls
dzdrm - dZ/dr
dzdthetom - dZ/dtheta(0)
dzdxdotm - dZ/dXdot
dzdzdotm - dZ/dZdot

CRUISE -- TAIL ROTOR Derivatives

dctsigdlampt - d(Ct/sigma)/d(lambda')
dctsigdmut - d(Ct/sigma)/d(mu)
dctsigdthetot - d(Ct/sigma)/d(theta0)
dlampdydott - d(lambda')/dYdot
dydxdott - dY/dXdot
dydydott - dY/dYdot
dydpt - dY/dp
dydrt - dY/dr
dydthetot - dY/d(theta0)
drdxdott - dR/dXdot
drdydott - dR/dYdot
drdpt - dR/dp
drdrt- dR/dr
drdthetot - dR/dtheta0)
dndxdott - dN/dXdot
dndydott - dN/dYdot

CRUISE -- V[illegible]AL STAB DERIVATIVES

dalpvdxdotv - d(alpha vert)/dXdot
dalpvdydotv - d(alpha vert)/dYdot
dbetadydotv - d(beta)/dYdot
detatvdxdotv - d(eta tail rotor)/dXdot
detatvdydotv - d(eta tail rotor)/dYdot
detafdydotv - d(eta fuselage)/dYdot
drdxdotv - dR/dXdot
drdydotv - dR/dYdot
dxdxdotv - dX/dXdot
dxdydotv - dX/dYdot
dydxdotv - dY/dXdot
dydydotv - dY/dYdot
dydpv - dY/dp

dydrv - dY/dr

CRUISE -- HORIZONTAL STAB DERIVATIVES

dalphdxdoth - d(alpha horiz)/dXdot
dalphdzdbldoth - d(alpha horiz)/dZ double dot
dalphdzdoth - d(alpha horiz)/dZdot
detamhdxdoth - d(eta main rotor)/dXdot
detafhdzdoth - d(eta fuse)/dZdot
detamhdzdoth - d(eta main rotor)/dZdot
dgamdzdoth - d(gamma)/dZdot
dxdxdoth - dX/dXdot
dxdzdbldoth - dX/dZ double dot
dxdzdoth - dX/dZdot
dzdxdoth - dZ/dXdot
dzdzdbldoth - dZ/dZ double dot
dzdzdoth - dZ/dZdot

CRUISE -- FUSELAGE

the following are from the curves in Appendix A of Ref. 2:

dfdalpf - d(D/q)/d(alpha fuse)
dlqdalpf - d(L/q)/d(alpha fuse)
dmqdalpf - d(M/q)/d(alpha fuse)
dnqdbetaf - d(N/q)/d(beta)
drqdbetaf - d(R/q)/d(beta)
dsfqdbetaf - d(Y/q)/d(beta)

dalpfdxdotf - d(alpha fuse)/dXdot
dalpfdzdotf - d(alpha fuse)/dZdot
dbetadydotf - d(beta)/dYdot
detamfdxdotf - d(eta)M/dXdot
detamfdzdotf - d(eta)M/dZdot
dgamdzdotf - d(gamma)/dZdot
dmdxdotf - dM/dXdot
dmdzdotf - dM/dZdot
dndydotf - dN/dYdot
drdydotf - dR/dRdot
dxdxdotf - dX/dXdot
dxdzdotf - dX/dZdot
dydydotf - dY/dYdot
dzdxdotf - dZ/dXdot
dzdzdotf - dZ/dZdot

HOVER -- BASIC MR Derivatives

dalda - dals/dA1

daldb - dals/dBl
daldlamp - dals/d(lambda')
daldmu - dals/dmu
daldp - dals/dp
daldq - dals/dq
daldtheto - dals/dtheta(o)
dblda - dbls/dAl
dbldb - dbls/dBl
dbldlamp - dbls/d(lambda')
dbldmu - dbls/dmu
dbldp - dbls/dp
dbldq - dbls/dq
dbldtheto - dbls/dtheta(o)
dbetdydot - d(Beta)/dYdot
dchsigda - d(Ch/sigma)/dals
dchsigdb - d(Ch/sigma)/dbls
dchsigdlamp - d(Ch/sigma)/d(lambda')
dchsigdmu - d(Ch/sigma)/dmu
dchsigdtheto - d(Ch/sigma)/dtheta(o)
dctsigdlamp - d(Ct/sigma)/d(lambda')
dctsigdmu - d(Ct/sigma)/dmu
dctsigdtheto - d(Ct/sigma)/dtheta(o)
dcqsigdlamp - d(Cq/sigma)/d(lambda')
dcqsigdmu - d(Cq/sigma)/dmu
dcqsigdtheto - d(Cq/sigma)/dtheta(o)
dcysigdb - d(Cy/sigma)/dbls
dlampdxdot - d(lambda')/dXdot
dlampdydot - d(lambda')/dYdot
dlampdzdot - d(lambda')/dZdot
dmdalsm - dM/dals - main rotor stiffness
dmudxdot - dmu/dxdot
drdblsm - dR/dBls - main rotor stiffness

RIGGING -- Cockpit stick rigging gains

dalmddela - dals/d(delta aileron)
dblmddele - dbls/d(delta elev)
ddelvddelp - rudder deflection per pedal travel
dphinddelp - dphi(notar thruster sleeve angle)/d(pedal)
dthetodc - dtheta(0.7)/d(collective)
dthetomddelc - dtheta0m/d(collective)
dthetotddelp - dtheta(0.7,tail rotor)/d(pedal travel)

APPENDIX E. PLOT LISTING.

Plots are saved under the following filenames:

Longitudinal roots - rootlon.met
Lateral/Directional roots - rootlat.met
Coupled roots - rootcoup.met

Longitudinal Cyclic
Longitudinal Cyclic to U, Hover - cbe2uh.met
Longitudinal Cyclic to Theta, Hover - cbe2theh.met
Longitudinal Cyclic to Pitch Rate, Hover - cbe2qh.met
Longitudinal Cyclic to W, Hover - cbe2wh.met

Collective
Collective to U, Hover - cbc2uh.met
Collective to Pitch, Hover - cbc2theh.met
Collective to Pitch Rate, Hover - cbc2qh.met
Collective to W, Hover - cbc2wh.met

Lateral cyclic
Lateral Cyclic to Bank, Hover - cba2phih.met
Lateral Cyclic to Sideslip (v), Hover - cba2vh.met
Lateral Cyclic to Roll Rate, Hover - cba2ph.met
Lateral Cyclic to Yaw Rate, Hover - cba2rh.met
Lateral Cyclic to Yaw, Hover - cba2yh.met

Pedals
Pedals to Bank, Hover - cbp2phih.met
Pedals to Sideslip (v), Hover - cbp2vh.met
Pedals to Roll Rate, Hover - cbp2ph.met
Pedals to Yaw Rate, Hover - cbp2rh.met
Pedals to Yaw, Hover - cbp2yh.met

APPENDIX F. SAMPLE OUTPUT.

Sample output file for the Prouty helicopter at a hover.

```
                    *** RESULTS ***
                       prouth

                  *** INPUT DATA ***

                      Flight Conditions

            Forward velocity =      0 kts
                 Temperature =     90 degs F
          Pressure altitude =      0 ft
           Auxiliary thrust =      0 lbs

                          Fuselage

                  Gross weight =  20000 lbs
  Equivalent flat plate area =   19.3 ft^2
     Vertical projected area =  380.0 ft^2
   CG height above waterline =    0.0 ft
        CG fuselage station =    0.0 ft
  CG position rt of buttline =    0.0 ft
                         Ixx = 35000.0 slug ft^2
                         Iyy = 40000.0 slug ft^2
                         Izz = 35000.0 slug ft^2
                         Ixz =    0.0 slug ft^2
              Downwash ratio =   1.50

                        Main Rotor

             Number of blades =      4
                 Rotor radius =   30.0 ft
                  Blade chord =    2.0 ft
                  Blade twist =  10.00 degs
                Blade airfoil =  HH-02
      Blade lift curve slope =   5.73
                Blade weight =  207.0 lbs
           Rotational velocity =  21.67 rads/sec
            Blade grip length =    4.5 ft
                Hinge offset =    1.5 ft
  Flapping moment of inertia = 2870.0 slug ft^2
  Hub height above waterline =    7.5 ft
        Hub fuselage station =    0.0 ft
  Hub position rt of buttline =    0.0 ft
              Mast incidence =   0.00 deg

                Tail rotor (zero if NOTAR)

             Number of blades =    3.0
                  Blade chord =    1.0 ft
                  Blade radius =    6.5 ft
                Lift curve slope =   5.73
             Rotational velocity = 100.00 rad/sec
  Flapping moment of inertia =      slug ft^2
```

Delta-3 angle =	-30.00	deg
Blade twist =	-5.00	deg
Hub height above waterline =	6.0	ft
Hub fuselage station =	37.0	ft
Hub position rt of buttline =	0.0	ft

Wing

Area =	0.0	ft^2
Span =	0.0	ft
CL =	0.00	
CDo =	0.0000	
Tip cord =	0.0	ft
Root cord =	0.0	ft
Wing efficiency factor =	0.00	
Zero lift angle =	0.00	deg
Angle of incidence =	0.00	deg
Lift curve slope =	0.00	
Height above waterline =	0.0	ft
Fuselage station =	0.0	ft
Position right of buttline =	0.0	ft
Rotor downwash ratio =	0.00	
Fuselage downwash ratio =	0.00	

Horizontal tail

Area =	18.0	ft^2
Span =	9.0	ft
CL =	0.20	
CDo =	0.0045	
Zero lift angle =	0.00	deg
Angle of incidence =	0.00	deg
Lift curve slope =	5.73	
Height above waterline =	-1.5	ft
Fuselage station =	33.0	ft
Position right of buttline =	0.0	ft
Dynamic pressure ratio =	0.60	
Rotor downwash ratio =	1.50	
Fuselage downwash ratio =	1.20	

Vertical tail

Area =	33.0	ft^2
Span =	7.7	ft
CL =	0.20	
CDo =	0.0045	
Height above waterline =	3.0	ft
Fuselage station =	35.0	ft
Position right of buttline =	0.0	ft
Zero lift angle =	0.00	deg
Maximum Cl =	2.00	
Dynamic pressure ratio =	0.60	
Lift curve slope =	5.70	

Rigging

Long cyclic pitch/inch defl =	3.33	deg/in
Lat cyclic pitch/inch defl =	2.05	deg/in
Collective pitch/inch defl =	1.33	deg/in
Tail rotor pitch change/defl =	-8.70	deg/unit

```
   Max deflection of control
      from neutral for NOTAR =    0.09 units
  Displacement of anti-torque
    control until full rudder =    0.00 units

                  *** CALCULATED DATA ***

                        Main Rotor

                 advance ratio =     0.0
    inflow parameter wrt TPP =   0.000
             Tip path angle =     0.0 degs
          Rotor coning angle =     7.4 degs
     1st lat cyclic term-A1 =    0.00 degs
     1st long cyclic term-B1 =   -0.00 degs
           lateral flapping =     0.0 degs
       longitudinal flapping =     0.0 degs
                  Lock number =     7.2

                  Tail Rotor (zero if NOTAR)

            tail rotor thrust = 1349.5 lbs
                advance ratio =     0.0
             inflow parameter =  -0.073
          Rotor coning angle =    -0.1 degs
            lateral flapping =    0.00 degs
       longitudinal flapping =    0.00 degs
                  Lock number =     2.7


                       State Matrices

Longitudinal uncoupled plant (A or F depending on notation)
States are [u w q theta]

   -0.0106         0    2.5068  -32.2000
         0   -0.2698         0    0.0002
    0.0038         0   -0.8878         0
         0         0    0.9977         0


Longitudinal uncoupled input matrix (B or G depending on notation)
Inputs are [longitudinal cyclic, collective, lateral cyclic, pedals]

    1.2248   -0.0000   -0.0738         0
         0   -5.8773         0         0
   -0.4338    0.0000    0.0261   -0.0238
         0         0         0         0


Lateral/directional uncoupled plant (A or F depending on notation)
States are [v p phi r psi]

   -0.0305   -2.6260   32.1267    0.7353         0
   -0.0064   -1.0274         0    0.0783         0
         0    1.0000         0   -0.0000         0
    0.0130    0.0783         0   -3.3360         0
         0         0         0    1.0000         0
```

Lateral/directional uncoupled input matrix (B or G depending on notation)
Inputs are [longitudinal cyclic, collective, lateral cyclic, pedals]

0.1200	0.0000	0.7533	-1.9736
0.0486	0.0000	0.3049	-0.2101
0	0	0	0
0	0.3001	0	1.2959
0	0	0	0

Coupled plant (A or F depending on notation)
States are [u w q theta v p phi r psi]

Columns 1 through 7

-0.0106	0	2.5068	-32.2000	-0.0057	-0.7469	0
0	-0.2698	0	0.0002	0	0	2.1710
0.0038	0	-0.8878	0	0.0013	0.2645	0
0	0	0.9977	0	0	0	0
0.0057	0	-0.7469	-0.0000	-0.0305	-2.6260	32.1267
0.0023	0	-0.3023	0	-0.0064	-1.0274	0
0	0	0.0000	0	0	1.0000	0
0	-0.0198	0	0	0.0130	0.0783	0
0	0	0	0	0	0	0

Columns 8 through 9

0	0
0	0
0.0266	0
0.0674	0
0.7353	0
0.0783	0
-0.0000	0
-3.3360	0
1.0000	0

Coupled input matrix (B or G depending on notation)
Inputs are [longitudinal cyclic, collective, lateral cyclic, pedals]

1.2248	-0.0000	-0.0738	0
0	-5.8773	0	0
-0.4338	0.0000	0.0261	-0.0238
0	0	0	0
0.1200	0.0000	0.7533	-1.9736
0.0486	0.0000	0.3049	-0.2101
0	0	0	0
0	0.3001	0	1.2959
0	0	0	0

Eigenvalue

Uncoupled

Longitudinal plant

Root	wn	damping
-1.0155	1.0155	1.0000
0.0585 + 0.3398i	0.3448	-0.1698
0.0585 - 0.3398i	0.3448	-0.1698
-0.2698	0.2698	1.0000

Lateral/Directional plant

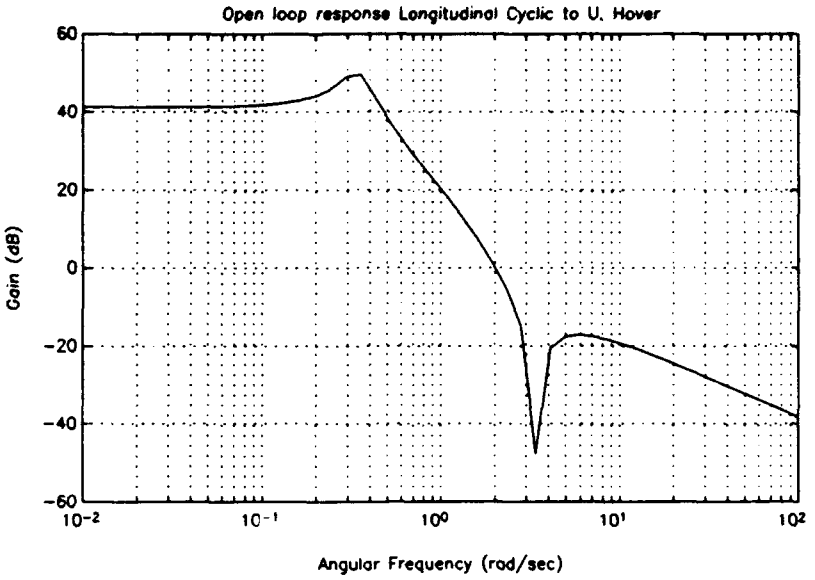
Root	wn	damping
0	0	-1.0000
-3.3433	3.3433	1.0000
-1.1784	1.1784	1.0000
0.0639 + 0.4025i	0.4075	-0.1568
0.0639 - 0.4025i	0.4075	-0.1568

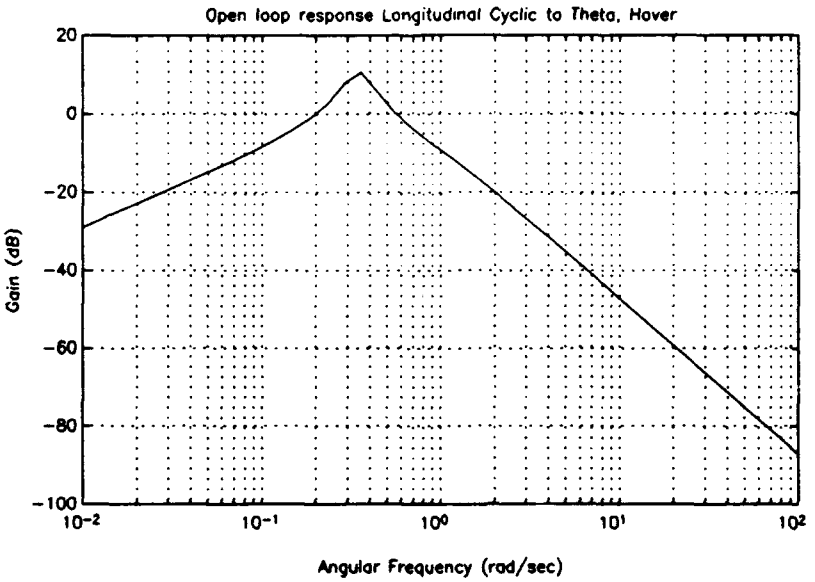
Coupled Plant

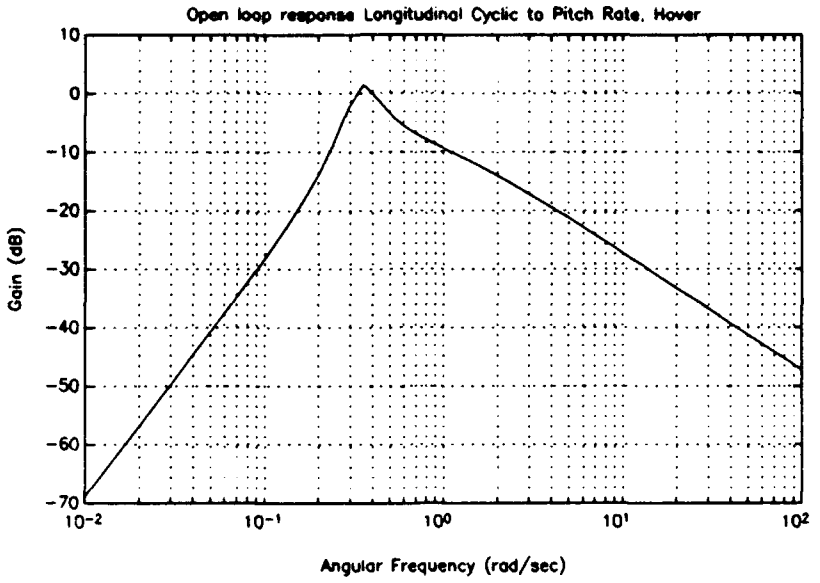
Root	wn	damping
0	0	-1.0000
-3.3432	3.3432	1.0000
-1.0970 + 0.2583i	1.1270	0.9734
-1.0970 - 0.2583i	1.1270	0.9734
0.0760 + 0.3983i	0.4054	-0.1873
0.0760 - 0.3983i	0.4054	-0.1873
0.0461 + 0.3582i	0.3611	-0.1276
0.0461 - 0.3582i	0.3611	-0.1276
-0.2689	0.2689	1.0000

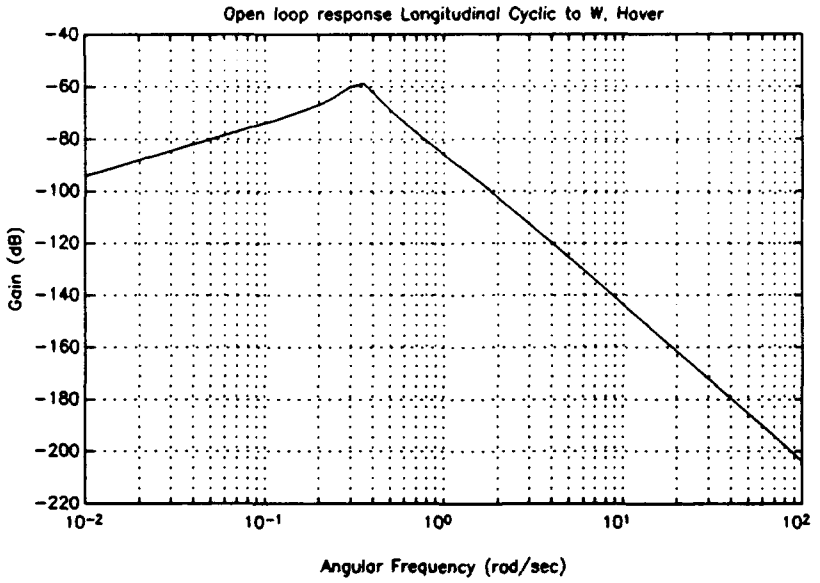
APPENDIX G. SAMPLE OPEN LOOP PLOTS.

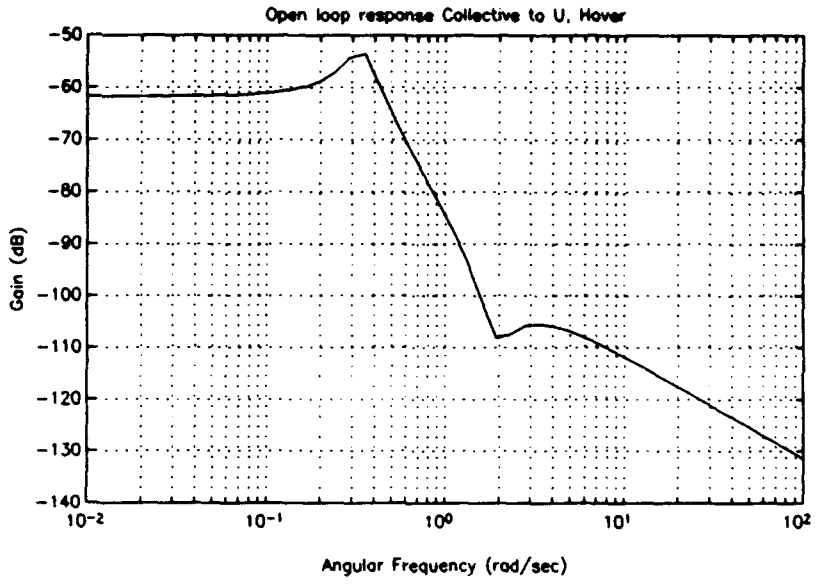
Sample open loop transfer plots

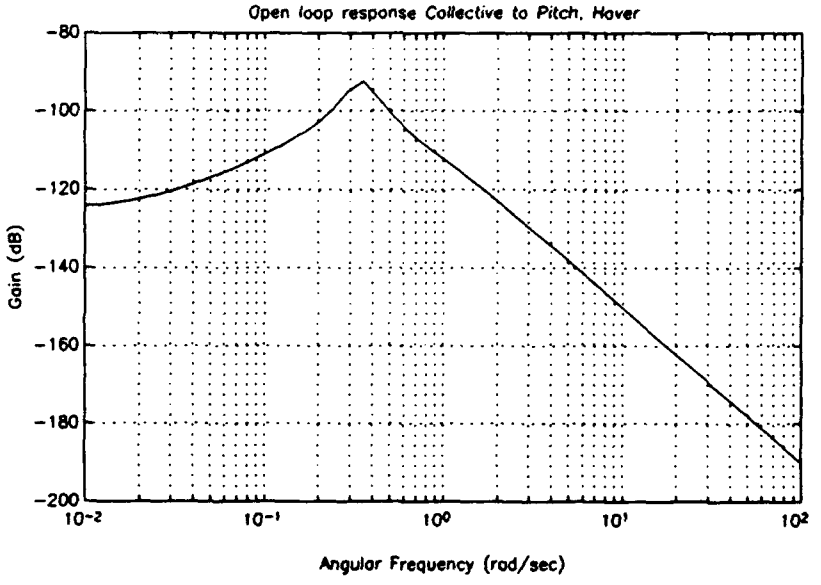


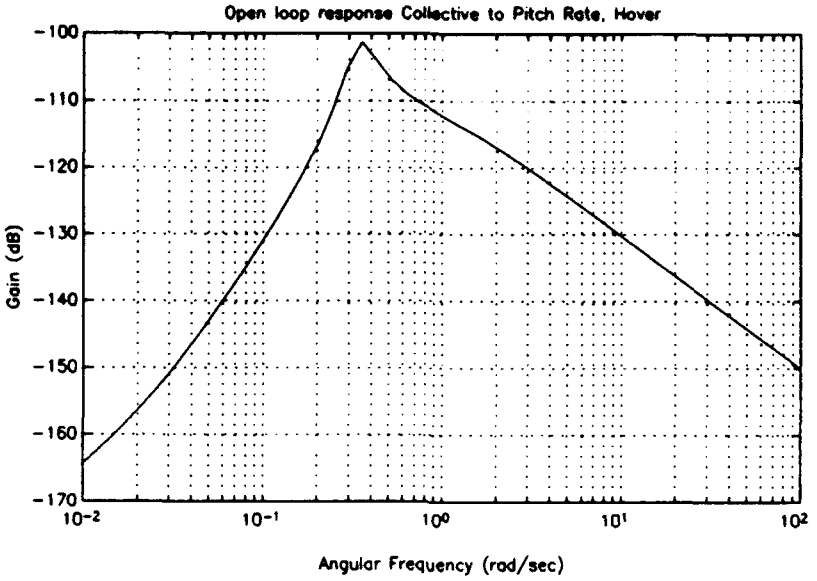


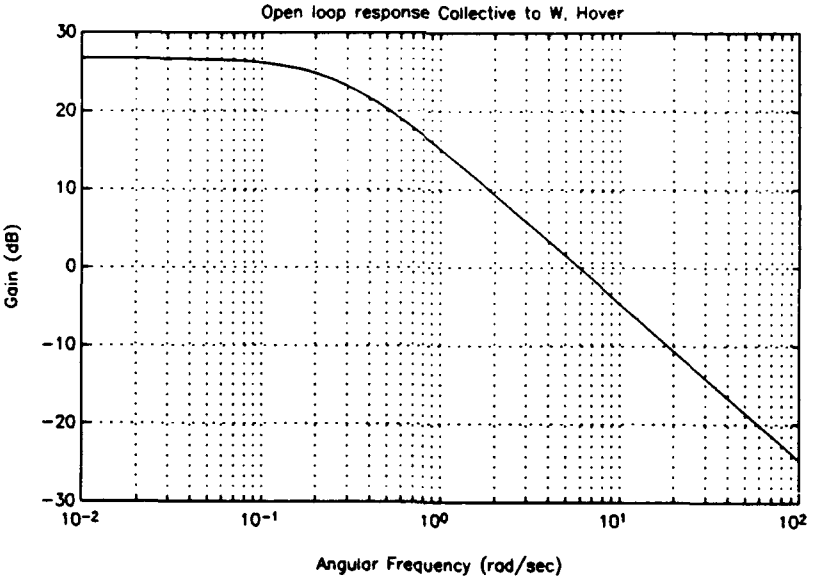

Open loop response Longitudinal Cyclic to Pitch Rate, Hover
Gain (dB)
10
0
-10
-20
-30
-40
-50
-60
-70
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

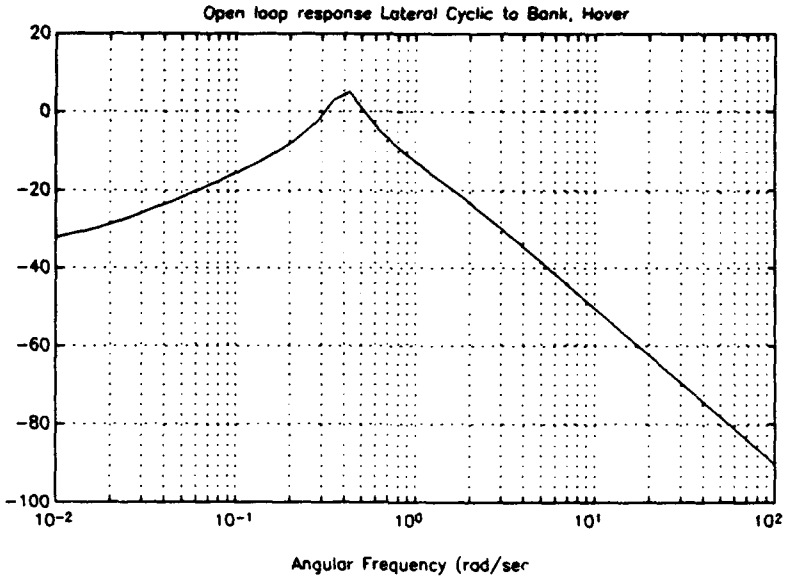

Open loop response Longitudinal Cyclic to W, Hover
Gain (dB)
-40
-60
-80
-100
-120
-140
-160
-180
-200
-220
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

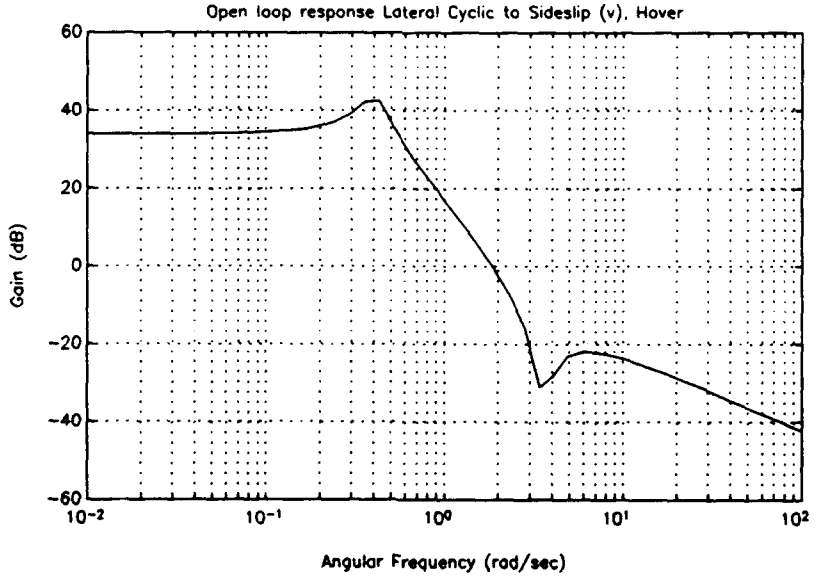

Open loop response Collective to U, Hover
-50
-60
-70
-80
-90
-100
-110
-120
-130
-140
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

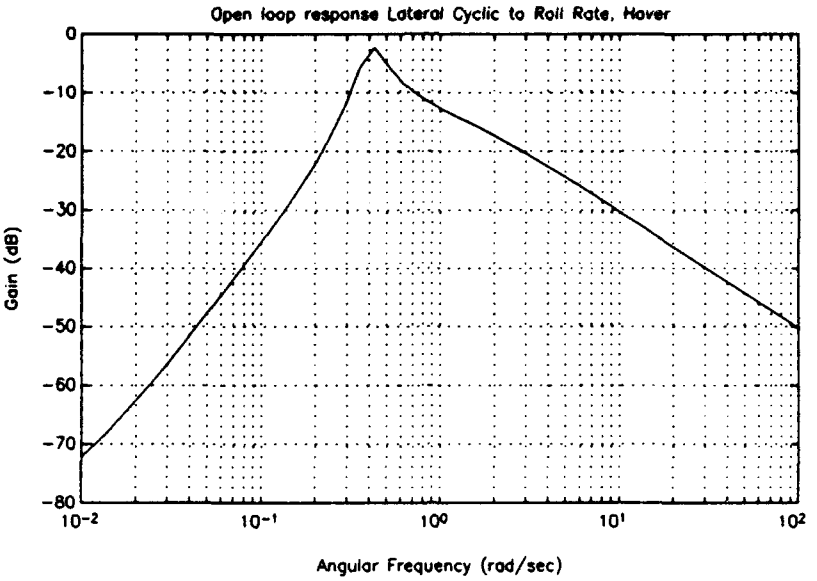

Open loop response Collective to Pitch, Hover
-80
-100
-120
-140
-160
-180
-200
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

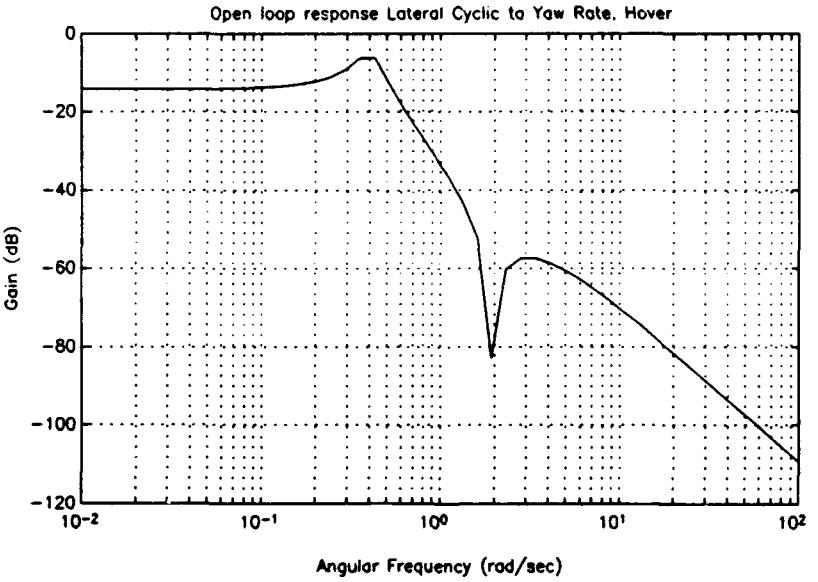

Open loop response Collective to Pitch Rate, Hover
-100
-110
-120
-130
-140
-150
-160
-170
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

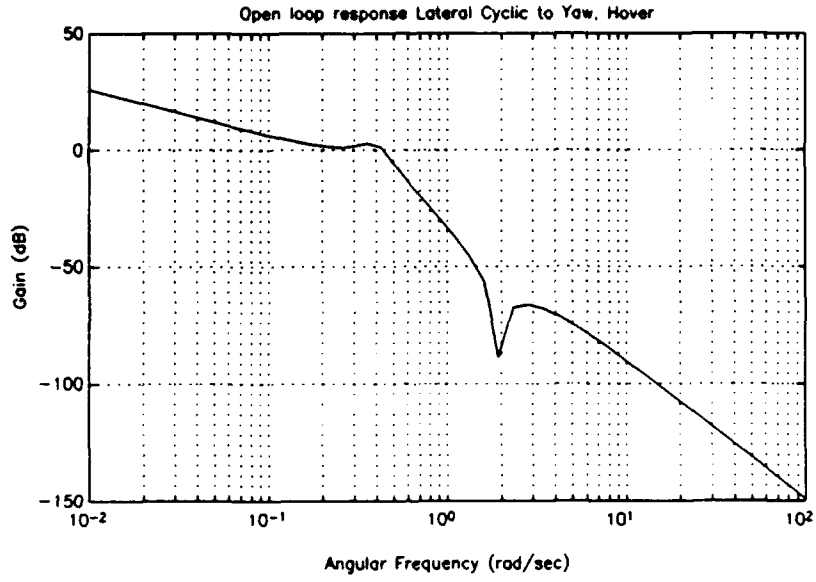

Open loop response Collective to W, Hover
30
20
10
0
-10
-20
-30
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

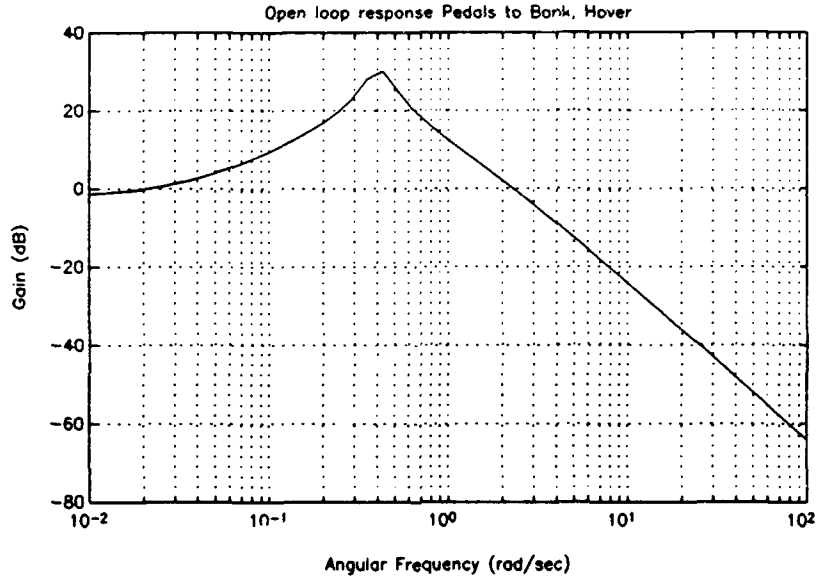

Open loop response Lateral Cyclic to Bank, Hover
20
0
-20
-40
-60
-80
-100
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec

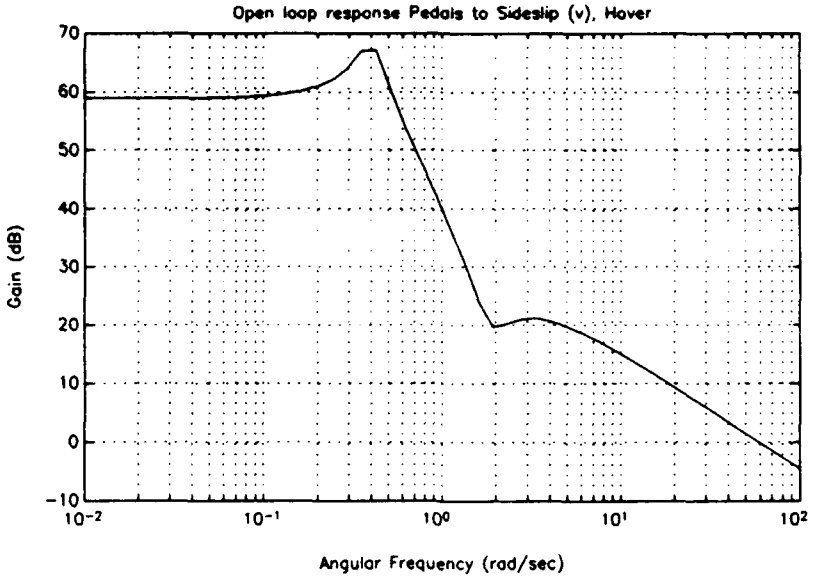

Open loop response Lateral Cyclic to Sideslip (v), Hover
60
40
20
0
-20
-40
-60
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

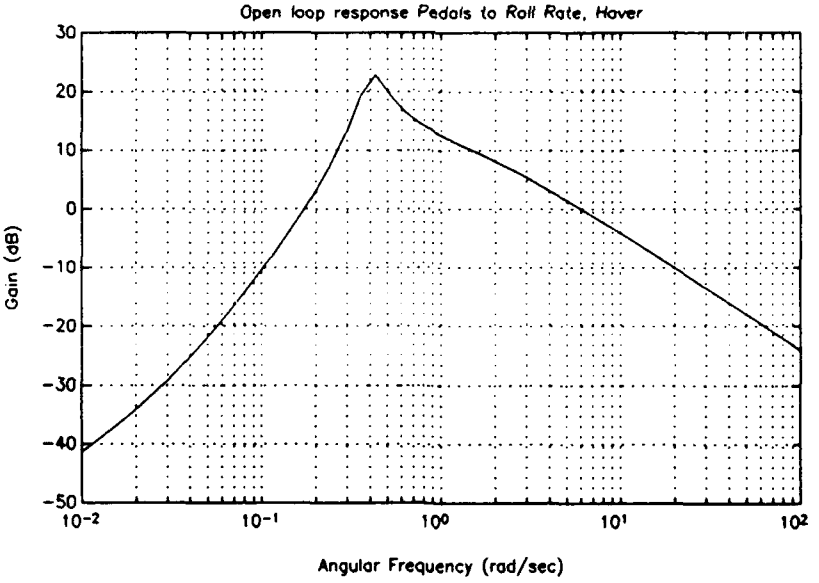

Open loop response Lateral Cyclic to Roll Rate, Hover
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Angular Frequency (rad/sec)

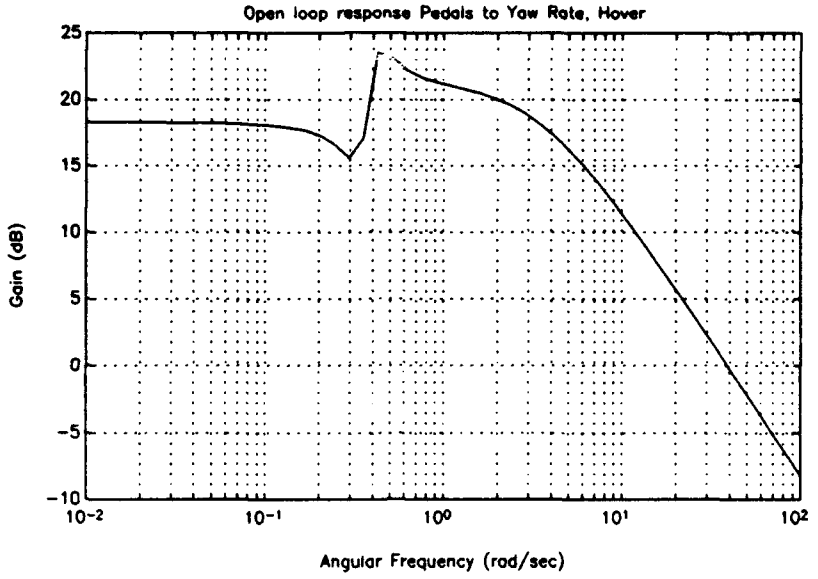

Open loop response Lateral Cyclic to Yaw Rate, Hover
Gain (dB)
0
-20
-40
-60
-80
-100
-120
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Angular Frequency (rad/sec)

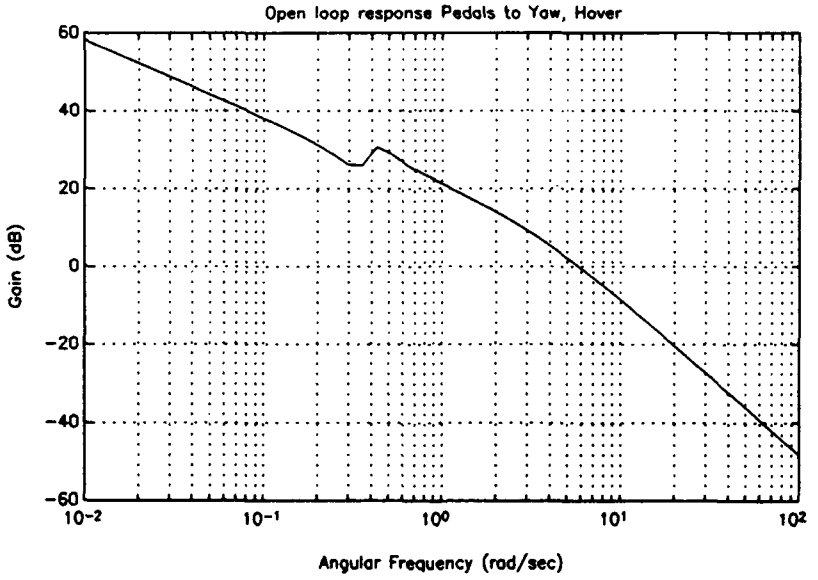

Open loop response Lateral Cyclic to Yaw, Hover
50
0
-50
-100
-150
Gain (dB)
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

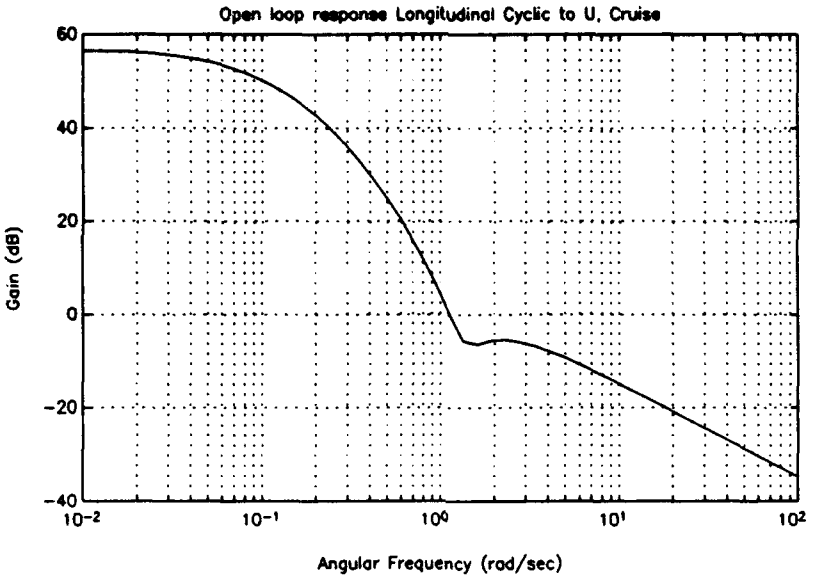

Open loop response Pedals to Bank, Hover
40
20
0
-20
-40
-60
-80
Gain (dB)
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

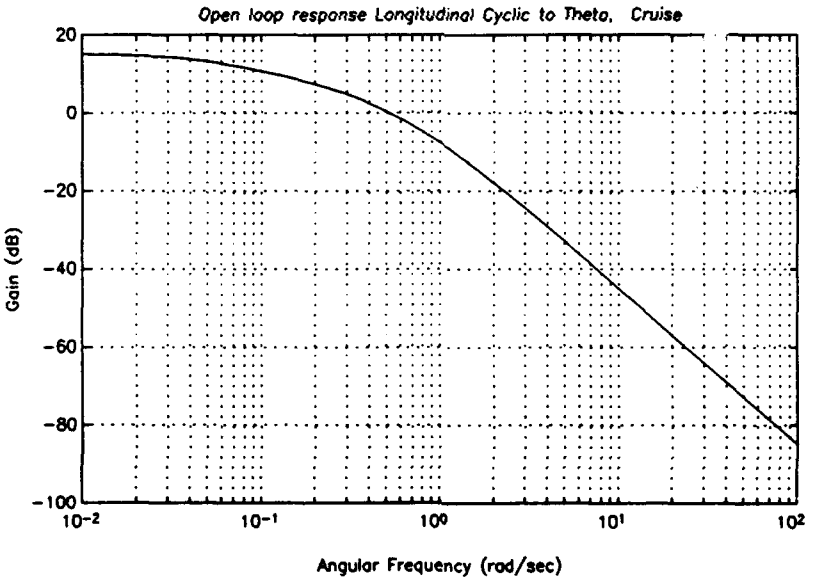

Open loop response Pedals to Sideslip (v), Hover
Gain (dB)
70
60
50
40
30
20
10
0
-10
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Angular Frequency (rad/sec)

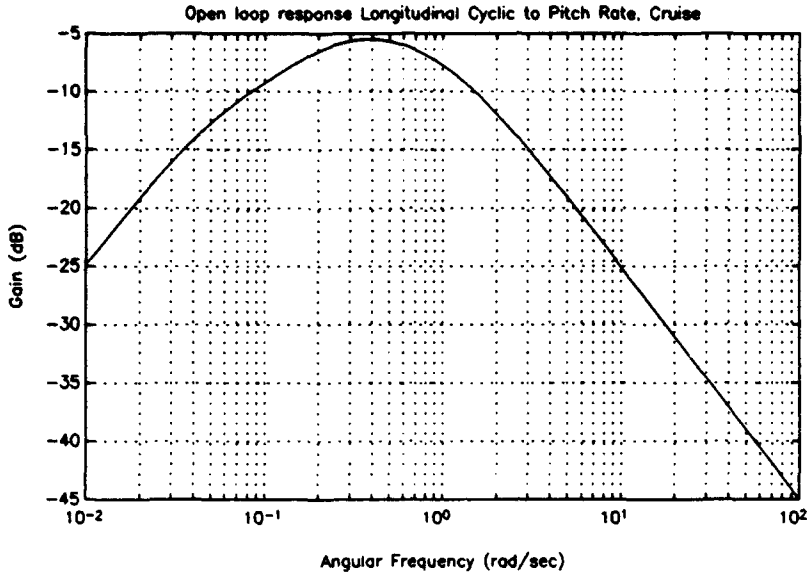

Open loop response Pedals to Roll Rate, Hover
Gain (dB)
30
20
10
0
-10
-20
-30
-40
-50
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Angular Frequency (rad/sec)

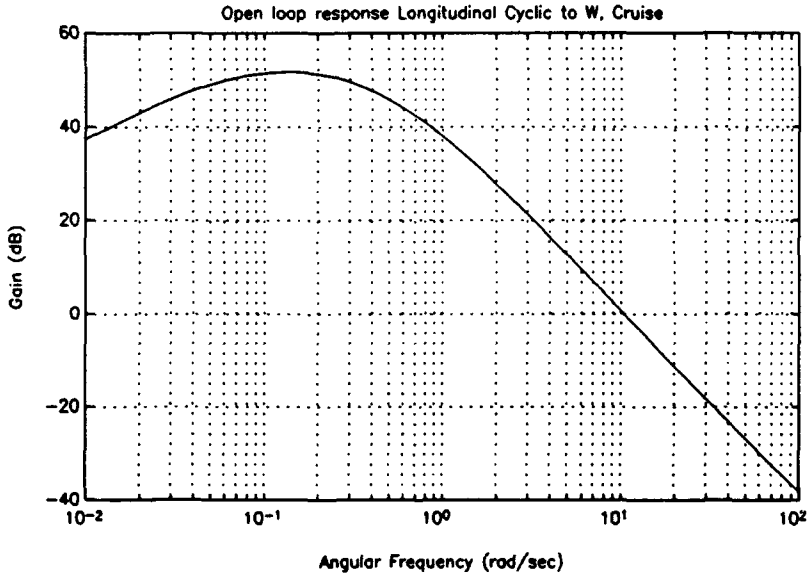

Open loop response Pedals to Yaw Rate, Hover
Gain (dB)
25
20
15
10
5
0
-5
-10
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

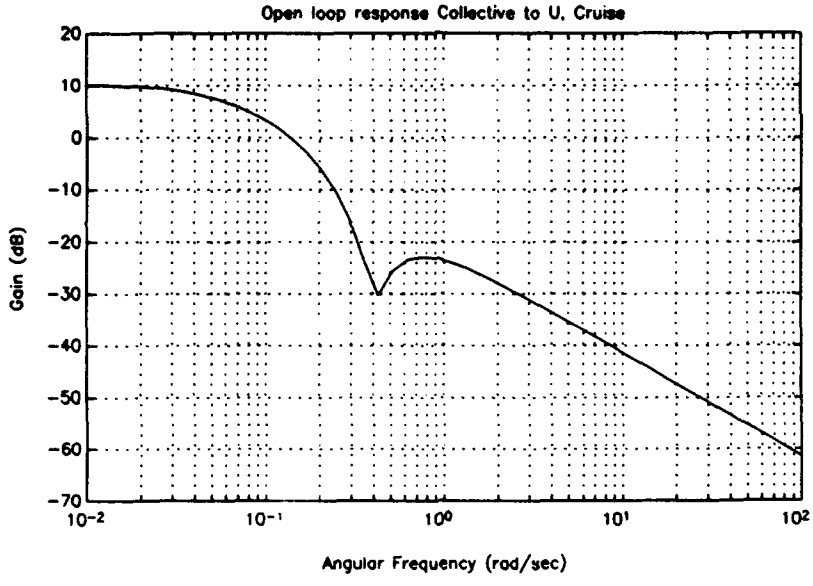

Open loop response Pedals to Yaw, Hover
Gain (dB)
60
40
20
0
-20
-40
-60
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

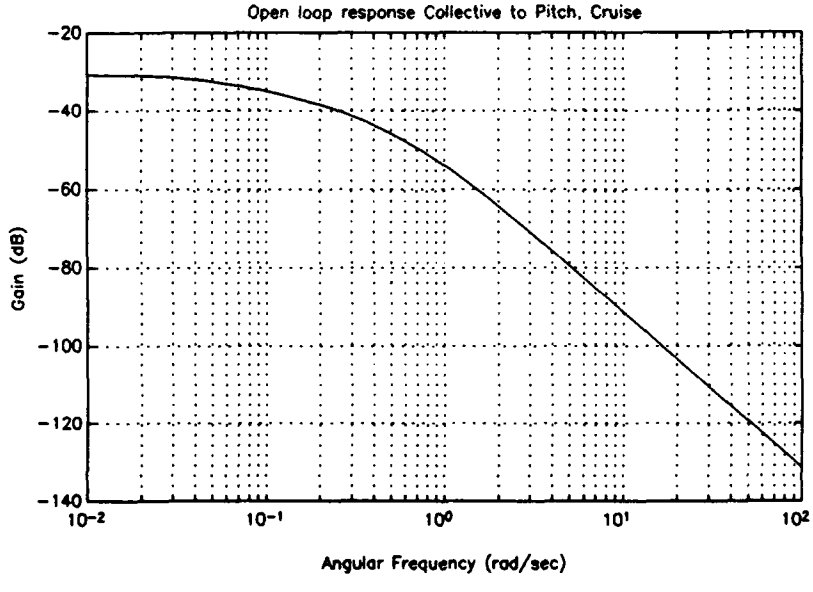

Open loop response Longitudinal Cyclic to U, Cruise
60
40
20
0
-20
-40
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

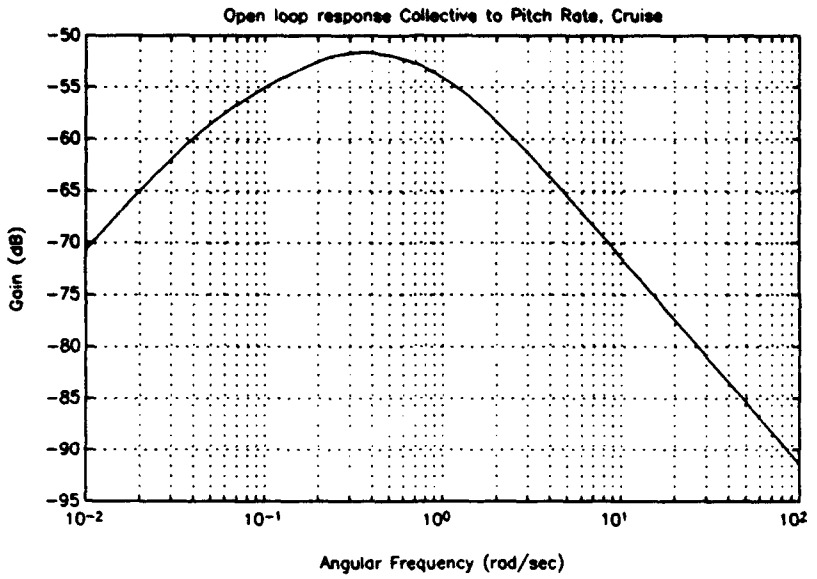

Open loop response Longitudinal Cyclic to Theta, Cruise
20
0
-20
-40
-60
-80
-100
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

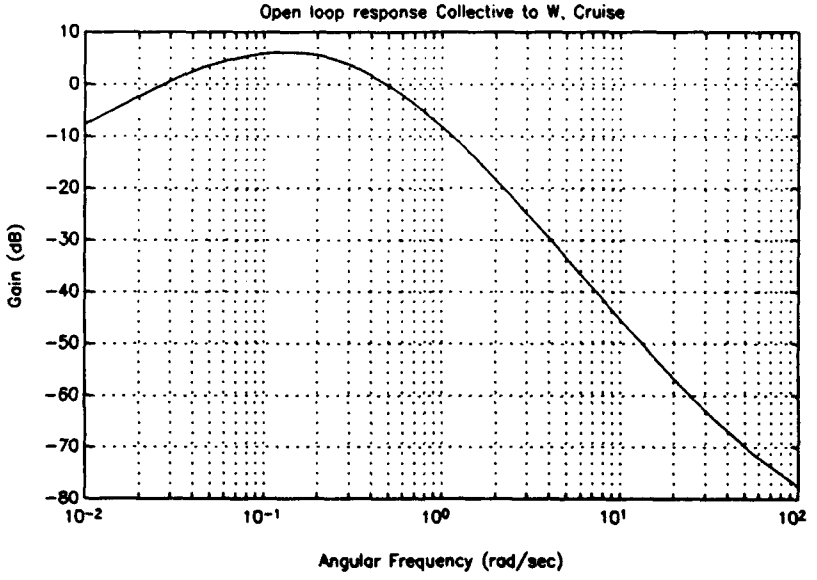

Open loop response Longitudinal Cyclic to Pitch Rate, Cruise
Gain (dB)
-5
-10
-15
-20
-25
-30
-35
-40
-45
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

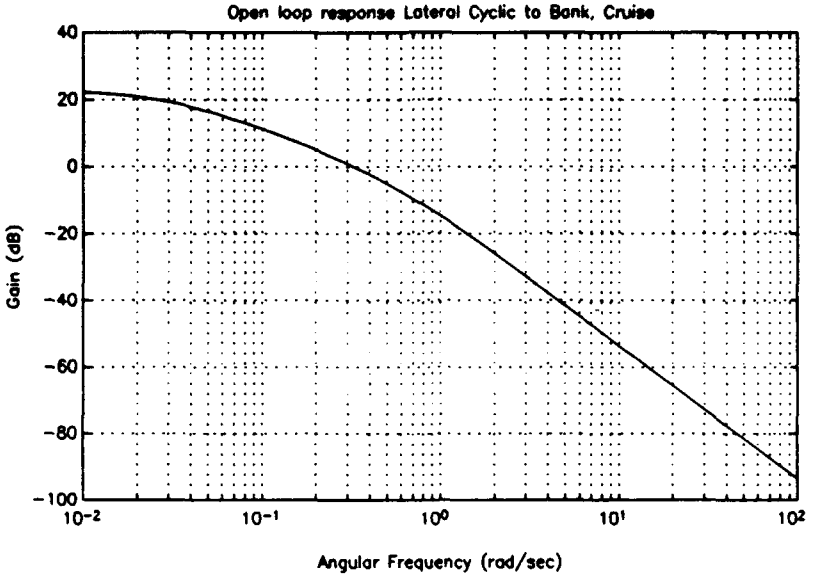

Open loop response Longitudinal Cyclic to W, Cruise
Gain (dB)
60
40
20
0
-20
-40
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

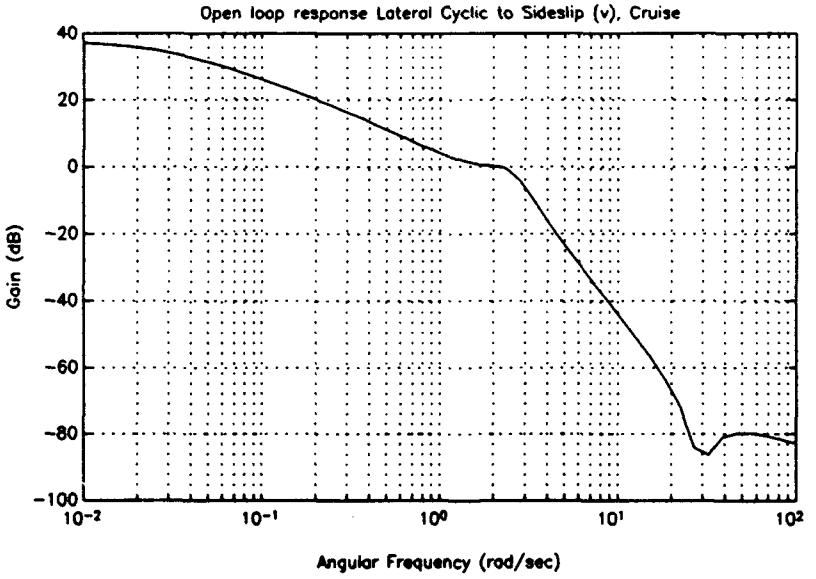

Open loop response Collective to U, Cruise
20
10
0
-10
-20
-30
-40
-50
-60
-70
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

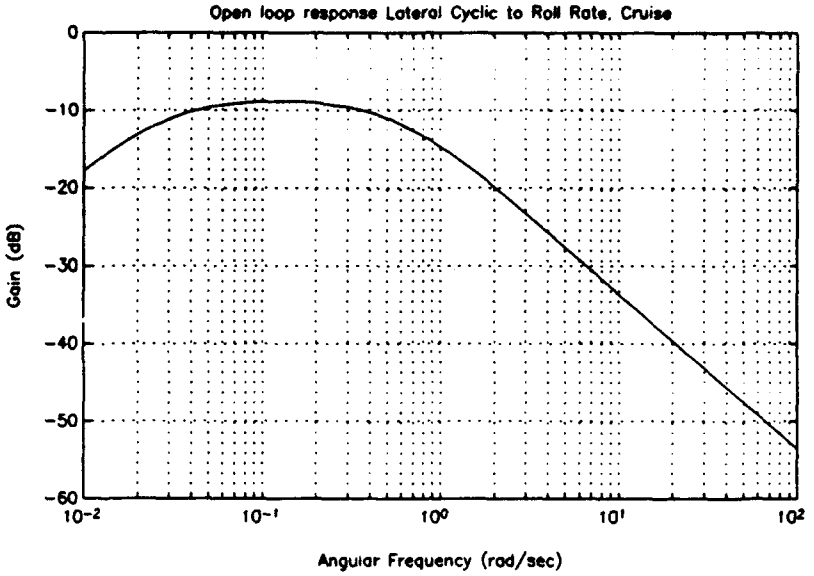

Open loop response Collective to Pitch, Cruise
-20
-40
-60
-80
-100
-120
-140
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

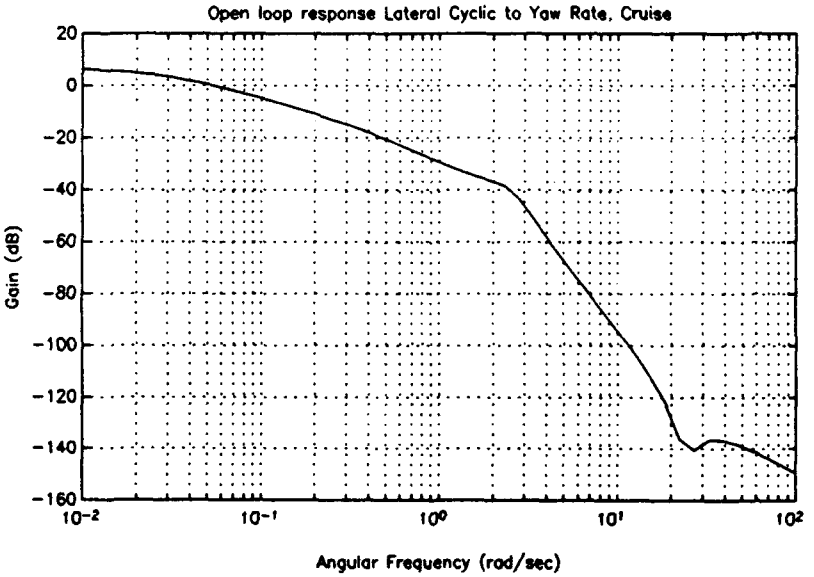

Open loop response Collective to Pitch Rate, Cruise
-50
-55
-60
-65
-70
-75
-80
-85
-90
-95
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

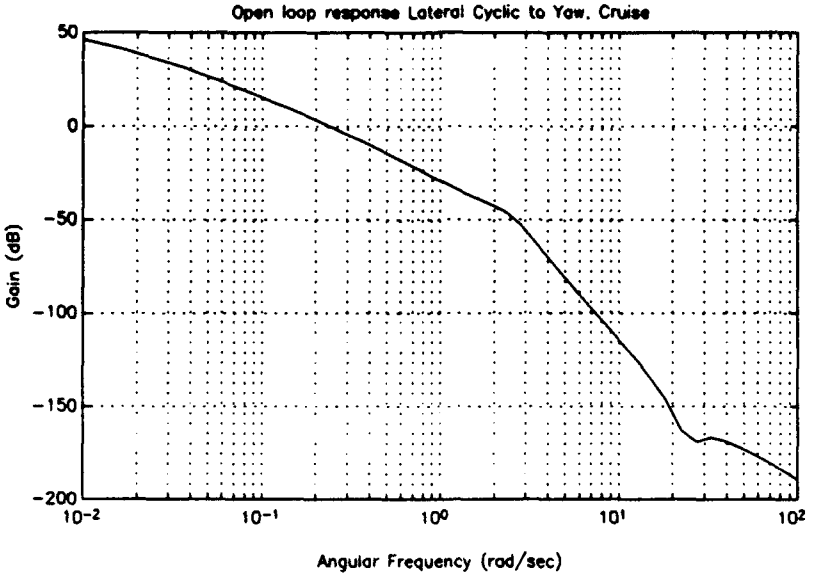

Open loop response Collective to W, Cruise
10
0
-10
-20
-30
-40
-50
-60
-70
-80
Gain (dB)
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

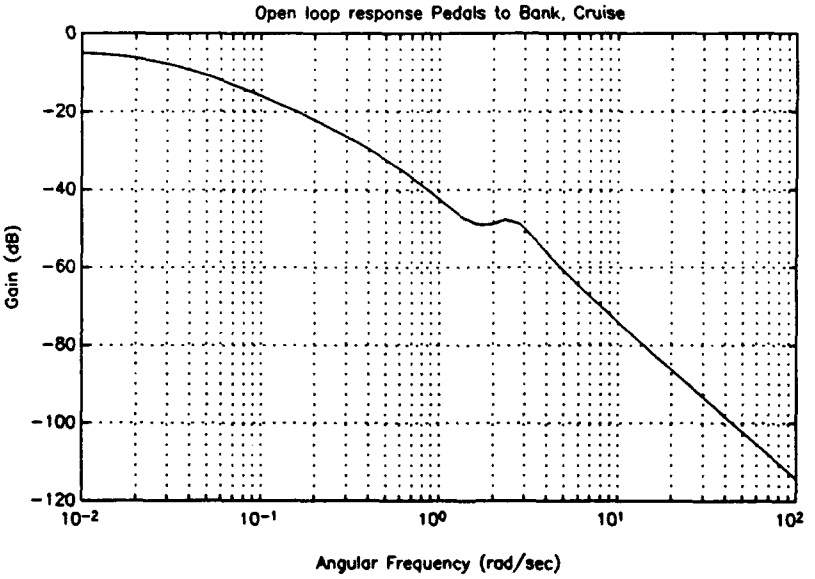

Open loop response Lateral Cyclic to Bank, Cruise
Gain (dB)
40
20
0
-20
-40
-60
-80
-100
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

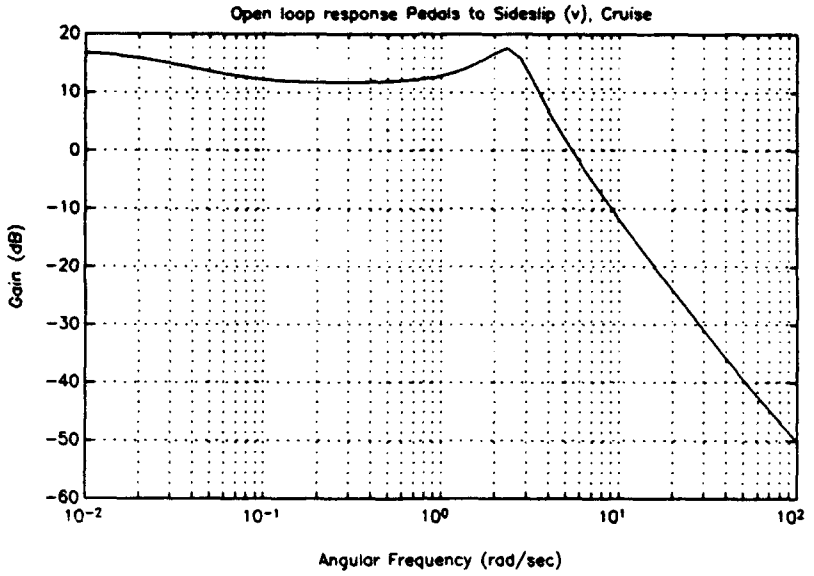

Open loop response Lateral Cyclic to Sideslip (v), Cruise
Gain (dB)
40
20
0
-20
-40
-60
-80
-100
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)

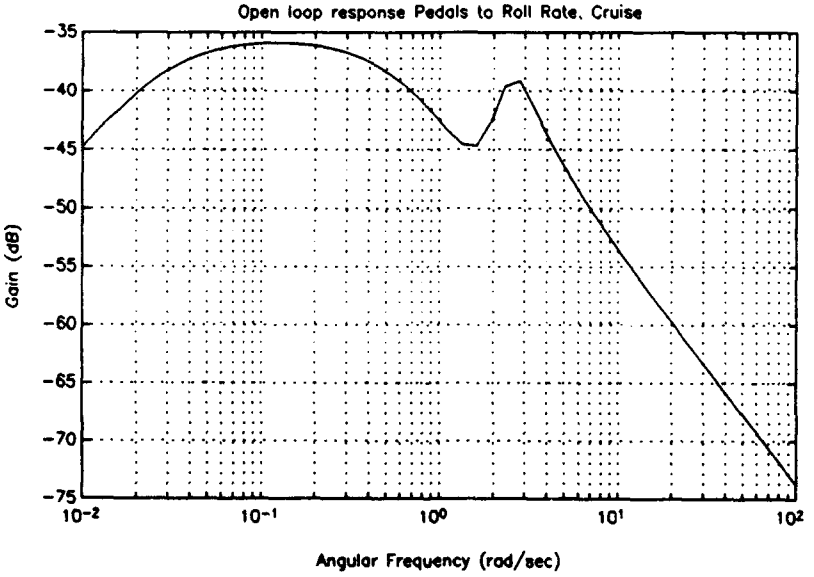

Open loop response Lateral Cyclic to Roll Rate, Cruise
Gain (dB)
0
-10
-20
-30
-40
-50
-60
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

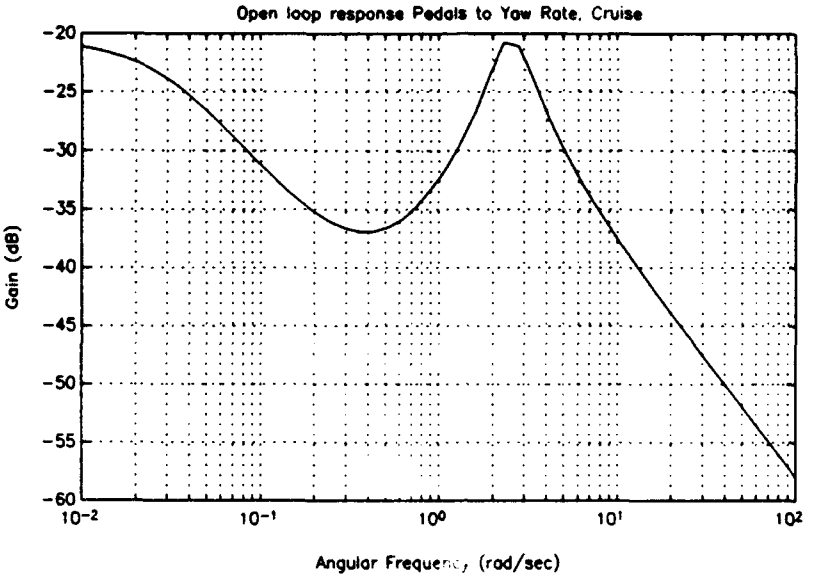

Open loop response Lateral Cyclic to Yaw Rate, Cruise
Gain (dB)
20
0
-20
-40
-60
-80
-100
-120
-140
-160
10-2
10-1
100
101
102
Angular Frequency (rad/sec)

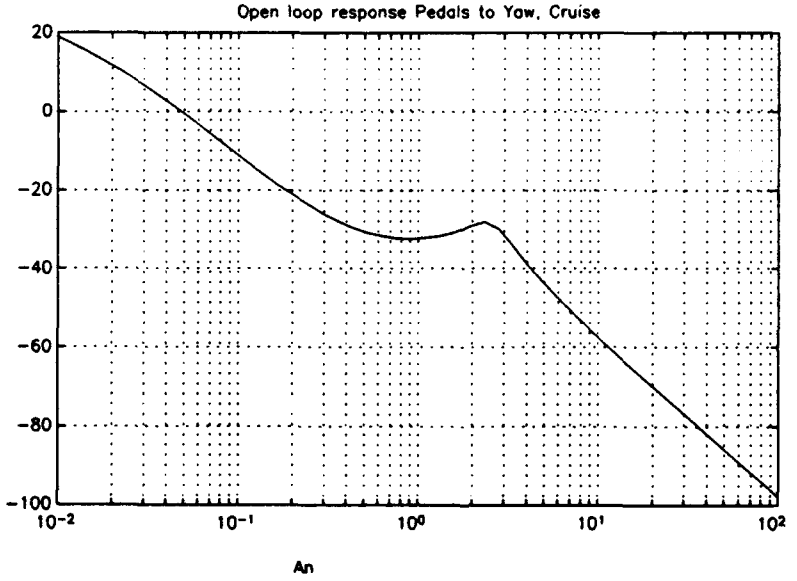

Open loop response Lateral Cyclic to Yaw, Cruise
50
0
-50
-100
-150
-200
Gain (dB)
10-2
10-1
100
101
102
Angular Frequency (rad/sec)


Open loop response Pedals to Bank, Cruise
0
-20
-40
-60
-80
-100
-120
Gain (dB)
10-2
10-1
100
101
102
Angular Frequency (rad/sec)


Open loop response Pedals to Sideslip (v), Cruise
Gain (dB)
20
10
0
-10
-20
-30
-40
-50
-60
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)


Open loop response Pedals to Roll Rate, Cruise
Gain (dB)
-35
-40
-45
-50
-55
-60
-65
-70
-75
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)


Open loop response Pedals to Yaw Rate, Cruise
Gain (dB)
-20
-25
-30
-35
-40
-45
-50
-55
-60
10^-2
10^-1
10^0
10^1
10^2
Angular Frequency (rad/sec)


Open loop response Pedals to Yaw, Cruise
20
0
-20
-40
-60
-80
-100
10^-2
10^-1
10^0
10^1
10^2
An

APPENDIX H. COMPUTER CODE.

Computer code for JANRAD Stability

A. STAB.M

```
%   STAB.M
%   NPS Helo Preliminary Design Program
%   Stability and Control Routines
%   Written by MAJ Walter M. Wirth, Jr.
%   September 1993

%   This program was designed as an interactive preliminary
%   design tool for stability and control analysis of a single
%   main rotor conventional or compound helicopter.  The
%   program provides stability derivatives, roots of the aircraft
%   plant, plots of various control parameters as well as open loop
%   control bandwidths from control inputs to aircraft reponse.
clear
load temp
eval(['load ', filename1]);
clc
disp(' ')
disp(' ')
disp('          ***   STABILITY AND CONTROL ROUTINE   ***')
disp(' ')
disp(' ')
pause(3)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   *** If editing an existing file:  get file name, display edit
%       menu, allow changes to selected variables, and save under
%       desired file name.  Loads to and saves from current
%       directory as a .mat file. ***
  if exist('Ib');
    answer0=1;
  else
    answer0=2;
  end
  if answer0==1,
    check2=1;
    while check2 > 0
      clc
disp(' ')
disp('          ***   STABILITY AND CONTROL MENU   ***')
disp('          *** ADDITIONAL PARAMETERS (1 of 3) ***')
disp(' ')
disp(' Main Rotor')
disp(' 1. flapping mom of inertia  2. hub height above waterline')
disp(' 3. hub fuselage station     4. hub posn right of buttline')
disp(' 5. mast incidence')
%
disp(' ')
disp(' Tail Rotor (enter zeros (0) if using NOTAR)')
disp(' 6. height above waterline   7. hub fuselage station')
disp(' 8. posn right of buttline   9. number of blades')
```

```
disp('10. blade chord              11. blade radius')
disp('12. lift curve slope         13. rotational velocity')
disp('14. flap mom of inertia      15. delta-3 angle')
disp('16. blade twist')
%
disp(' ')
disp(' Verticle Fin')
disp('17. height above waterline   18. fuselage station')
disp('19. posn right of buttline   20. alpha zero lift')
disp('21. CL max                   22. dynamic pressure ratio')
disp('23. lift curve slope')
%
disp(' ')
disp('0. NO CHANGES')
      choice=input('Input the parameter to change: ');
      if choice==1,
        clc
        disp(' ')
        Ib
        temp1=Ib;
        Ib=input('Blade flapping moment of inertia (slug ft^2): ');
        if isempty(Ib),
           Ib=temp1;
        end
        clear temp1
      elseif choice==2,
        clc
        disp(' ')
        hmd
        temp1=hmd;
        hmd=input('Hub height above reference datum/waterline (ft): ');
        if isempty(hmd),
           hmd=temp1;
        end
        clear temp1
      elseif choice==3,
        clc
        disp(' ')
        lmd
        temp1=lmd;
        lmd=input('Hub fuselage station (ft): ');
        if isempty(lmd),
           lmd=temp1;
        end
        clear temp1
      elseif choice==4,
        clc
        disp(' ')
        ymd
        temp1=ymd;
        ymd=input('Hub position right of buttline (ft): ');
        if isempty(ymd),
           ymd=temp1;
        end
        clear temp1
      elseif choice==5,
        clc
        disp(' ')
        im*57.3
        temp1=im*57.3;
```

```
      im=input('Mast incidence (negative forward - deg): ')/57.3;
      if isempty(im),
         im=temp1/57.3;
      end
      clear temp1
    elseif choice==6,
      clc
      disp(' ')
      htd
      temp1=htd;
      htd=input('Tail rotor height above reference datum/waterline (ft):
');
      if isempty(htd),
         htd=temp1;
      end
      clear temp1
    elseif choice==7,
      clc
      disp(' ')
      ltd
      temp1=ltd;
      ltd=input('Tail rotor fuselage station (ft): ');
      if isempty(ltd),
         ltd=temp1;
      end
      clear temp1
    elseif choice==8,
      clc
      disp(' ')
      ytd
      temp1=ytd;
      ytd=input('Tail rotor position right of buttline (ft): ');
      if isempty(ytd),
         ytd=temp1;
      end
      clear temp1
    elseif choice==9,
      clc
      disp(' ')
      bt
      temp1=bt;
      bt=input('Number of tail rotor blades: ');
      if isempty(bt),
         bt=temp1;
      end
      clear temp1
    elseif choice==10,
      clc
      disp(' ')
      cot
      temp1=cot;
      cot=input('Blade chord (ft): ');
      if isempty(cot),
         cot=temp1;
      end
      clear temp1
    elseif choice==11,
      clc
      disp(' ')
      Rt
```

```
  temp1=Rt;
  Rt=input('Tail rotor blade radius (ft): ');
  if isempty(Rt),
     Rt=temp1;
  end
  clear temp1
elseif choice==12,
  clc
  disp(' ')
  at
  temp1=at;
  at=input('Average lift curve slope of tail rotor: ');
  if isempty(at),
     at=temp1;
  end
  clear temp1
elseif choice==13,
  clc
  disp(' ')
  ohmt
  temp1=ohmt;
  ohmt=input('Rotational velocity of tail rotor (rad/sec): ');
  if isempty(ohmt),
     ohmt=temp1;
  end
  clear temp1
elseif choice==14,
  clc
  disp(' ')
  Ibt
  temp1=Ibt;
  Ibt=input('Blade flapping moment of inertia (slug ft^2): ');
  if isempty(Ibt),
     Ibt=temp1;
  end
  clear temp1
elseif choice==15,
  clc
  disp(' ')
  delta3*57.3
  temp1=delta3*57.3;
  delta3=input('Delta-3 angle (deg): ')/57.3;
  if isempty(delta3),
     delta3=temp1/57.3;
  end
  clear temp1
elseif choice==16,
  clc
  disp(' ')
  thetalt*57.3
  temp1=thetalt*57.3;
  thetalt=input('Blade twist (deg): ')/57.3;
  if isempty(thetalt),
     thetalt=temp1/57.3;
  end
  clear temp1
elseif choice==17,
  clc
  disp(' ')
  hvd
```

```
  temp1=hvd;
  hvd=input('Height above reference datum/waterline (ft): ');
  if isempty(hvd),
     hvd=temp1;
  end
  clear temp1
elseif choice==18,
  clc
  disp(' ')
  lvd
  temp1=lvd;
  lvd=input('Fuselage station (ft): ');
  if isempty(lvd),
     lvd=temp1;
  end
  clear temp1
elseif choice==19,
  clc
  disp(' ')
  yvd
  temp1=yvd;
  yvd=input('Position right of buttline (ft): ');
  if isempty(yvd),
     yvd=temp1;
  end
  clear temp1
elseif choice==20,
  clc
  disp(' ')
  alplov*57.3
  temp1=alplov*57.3;
  alplov=input('Zero lift angle for vertical tail (deg): ')/57.3;
  if isempty(alplov),
     alplov=temp1/57.3;
  end
  clear temp1
elseif choice==21,
  clc
  disp(' ')
  clvertmax
  temp1=clvertmax;
  clvertmax=input('Maximum Cl for vertical tail: ');
  if isempty(clvertmax),
     clvertmax=temp1;
  end
  clear temp1
elseif choice==22,
  clc
  disp(' ')
  qvq
  temp1=qvq;
  qvq=input('Dynamic pressure ratio (pg 489 Prouty): ');
  if isempty(qvq),
     qvq=temp1;
  end
  clear temp1
elseif choice==23,
  clc
  disp(' ')
  av
```

```
        temp1=av;
        av=input('Lift curve slope of vertical tail: ');
        if isempty(av),
           av=temp1;
        end
        clear temp1
      elseif choice==0,
        check2=0
        clc
      else
        disp(' ')
        disp('enter a displayed number ...press any key to continue')
        pause
      end
    end
%
    check2=1;
    while check2 > 0
      clc
disp(' ')
disp('          ***   STABILITY AND CONTROL MENU   ***')
disp('          *** ADDITIONAL PARAMETERS (2 of 3) ***')
%
disp(' ')
disp(' Horizontal Tail')
disp(' 1. height above waterline   2. fuselage station')
disp(' 3. posn right of buttline   4. alpha zero lift')
disp(' 5. angle of incidence       6. lift curve slope')
disp(' 7. dynamic pressure ratio   8. rotor downwash ratio')
disp(' 9. fuselage downwash ratio')
%
disp(' ')
disp(' Wing')
disp('10. height above waterline  11. fuselage station')
disp('12. posn right of buttline  13. alpha zero lift')
disp('14. angle of incidence      15. lift curve slope')
disp('16. tip cord                17. root cord')
disp('18. rotor downwash ratio    19. fuselage downwash ratio ')
%
disp(' ')
disp('0. NO CHANGES')
      choice=input('Input the parameter to change: ');
      if choice==1
        clc
        disp(' ')
        hhd
        temp1=hhd;
        hhd=input('Height above reference datum/waterline (ft): ');
        if isempty(hhd),
           hhd=temp1;
        end
        clear temp1
      elseif choice==2,
        clc
        disp(' ')
        lhd
        temp1=lhd;
        lhd=input('Fuselage station (ft): ');
        if isempty(lhd),
           lhd=temp1;
```

```
  end
  clear temp1
elseif choice==3,
  clc
  disp(' ')
  yhd
  temp1=yhd;
  yhd=input('Position right of buttline: ');
  if isempty(yhd),
     yhd=temp1;
  end
  clear temp1
elseif choice==4,
  clc
  disp(' ')
  alploh*57.3
  temp1=alploh*57.3;
  alploh=input('Zero lift angle for horizontal tail (deg): ')/57.3;
  if isempty(alploh),
     alploh=temp1/57.3;
  end
  clear temp1
elseif choice==5,
  clc
  disp(' ')
  ih*57.3
  temp1=ih*57.3;
  ih=input('Angle of incidence of horizontal tail (deg): ')/57.3;
  if isempty(ih),
     ih=temp1/57.3;
  end
  clear temp1
elseif choice==6,
  clc
  disp(' ')
  ah
  temp1=ah;
  ah=input('Lift curve slope of horizontal tail: ');
  if isempty(ah),
     ah=temp1;
  end
  clear temp1
elseif choice==7,
  clc
  disp(' ')
  qhq
  temp1=qhq;
  qhq=input('Dynamic pressure ratio (pg 489 Prouty): ');
  if isempty(qhq),
     qhq=temp1;
  end
  clear temp1
elseif choice==8,
  clc
  disp(' ')
  vhv1
  temp1=vhv1;
  vhv1=input('Rotor downwash ratio (pg 489 Prouty): ');
  if isempty(vhv1),
     vhv1=temp1;
```

```
  end
  clear temp1
elseif choice==9,
  clc
  disp(' ')
  detafdalpfh
  temp1=detafdalpfh;
  detafdalpfh=input('Fuselage downwash ratio (pg 489 Prouty): ');
  if isempty(detafdalpfh),
     detafdalpfh=temp1;
  end
  clear temp1
elseif choice==10
  clc
  disp(' ')
  hwd
  temp1=hwd;
  hwd=input('Height above reference datum/waterline (ft): ');
  if isempty(hwd),
     hwd=temp1;
  end
  clear temp1
elseif choice==11
  clc
  disp(' ')
  lwd
  temp1=lwd;
  lwd=input('Fuselage station (ft): ');
  if isempty(lwd),
     lwd=temp1;
  end
  clear temp1
elseif choice==12
  clc
  disp(' ')
  ywd
  temp1=ywd;
  ywd=input('Position right of buttline (ft): ');
  if isempty(ywd),
     ywd=temp1;
  end
  clear temp1
elseif choice==13,
  clc
  disp(' ')
  alplow*57.3
  temp1=alplow*57.3;
  alplow=input('Zero lift angle for wing (deg): ')/57.3;
  if isempty(alplow),
     alplow=temp1/57.3;
  end
  clear temp1
elseif choice==14,
  clc
  disp(' ')
  iw*57.3
  temp1=iw*57.3;
  iw=input('Angle of incidence of wing (deg): ')/57.3;
  if isempty(iw),
     iw=temp1/57.3;
```

```
  end
  clear temp1
elseif choice==15,
  clc
  disp(' ')
  aw
  temp1=aw;
  aw=input('Lift curve slope of wing: ');
  if isempty(aw),
     aw=temp1;
  end
  clear temp1
elseif choice==16,
  clc
  disp(' ')
  ctw
  temp1=ctw;
  ctw=input('Tip cord (ft): ');
  if isempty(ctw),
     ctw=temp1;
  end
  clear temp1
elseif choice==17,
  clc
  disp(' ')
  crw
  temp1=crw;
  crw=input('Root cord (ft): ');
  if isempty(crw),
     crw=temp1;
  end
  clear temp1
elseif choice==18,
  clc
  disp(' ')
  vwv1
  temp1=vwv1;
  vwv1=input('Rotor downwash ratio (pg 489 Prouty): ');
  if isempty(vwv1),
     vwv1=temp1;
  end
  clear temp1
elseif choice==19,
  clc
  disp(' ')
  detafdalpfw
  temp1=detafdalpfw;
  detafdalpfw=input('Fuselage downwash ratio (pg 489 Prouty): ');
  if isempty(detafdalpfw),
     detafdalpfw=temp1;
  end
  clear temp1
elseif choice==0,
  check2=0
  clc
else
  disp(' ')
  disp('enter a displayed number ... press any key to continue')
  pause
end
```

```
    end
%
    check2=1;
    while check2 > 0
      clc
disp(' ')
disp('          ***   STABILITY AND CONTROL MENU   ***')
disp('          *** ADDITIONAL PARAMETERS (3 of 3) ***')
%
disp(' ')
disp(' CG location and Inertias/fuselage parameters')
disp(' 1. cg ht. above waterline   2. cg fuselage station')
disp(' 3. cg posn rt of buttline   4. Ixx')
disp(' 5. Iyy                      6. Izz')
disp(' 7. Ixz                      8. fuselage downwash ratio')
%
disp(' ')
disp(' NOTAR if available (enter zeros if using tail rotor)')
disp(' 9. height above waterline  10. boom fuselage station')
disp('11. boom position left ref  12. NOTAR diameter ')
disp('13. swirl angle at boom     14. NOTAR max force')
disp('15. thruster fuselage station')
%
disp(' ')
disp(' Rigging')
disp('16. B1 main/in defl (del e) 17. A1 main/in defl (dela)')
disp('18. theta0m/in defl (del c) 19. theta0t/pedal defl (del r or p)')
disp('20. NOTAR sleeve twist/defl 21. max rudder defl')
disp(' ')
disp('0. NO CHANGES')
      choice=input('Input the parameter to change: ');
      if choice==1,
        clc
        disp(' ')
        zcg
        temp1=zcg;
        zcg=input('CG height above reference datum/waterline (ft): ');
        if isempty(zcg),
           zcg=temp1;
        end
        clear temp1
      elseif choice==2,
        clc
        disp(' ')
        xcg
        temp1=xcg;
        xcg=input('CG Fuselage station (ft): ');
        if isempty(xcg),
           xcg=temp1;
        end
        clear temp1
      elseif choice==3,
        clc
        disp(' ')
        ycg
        temp1=ycg;
        ycg=input('CG position right of buttline (ft): ');
        if isempty(ycg),
           ycg=temp1;
        end
```

```
  clear temp1
elseif choice==4,
  clc
  disp(' ')
  Ixx
  temp1=Ixx;
  Ixx=input('Ixx (slug ft^2): ');
  if isempty(Ixx),
     Ixx=temp1;
  end
  clear temp1
elseif choice==5,
  clc
  disp(' ')
  Iyy
  temp1=Iyy;
  Iyy=input('Iyy (slug ft^2): ');
  if isempty(Iyy),
     Iyy=temp1;
  end
  clear temp1
elseif choice==6,
  clc
  disp(' ')
  Izz
  temp1=Izz;
  Izz=input('Izz (slug ft^2): ');
  if isempty(Izz),
     Izz=temp1;
  end
  clear temp1
elseif choice==7,
  clc
  disp(' ')
  Ixz
  temp1=Ixz;
  Ixz=input('Ixz (slug ft^2): ');
  if isempty(Ixz),
     Ixz=temp1;
  end
  clear temp1
elseif choice==8,
  clc
  disp(' ')
  vfv1
  temp1=vfv1;
  vfv1=input('Downwash ratio for fuselage (page 513 Prouty): ');
  if isempty(vfv1),
     vfv1=temp1;
  end
  clear temp1
elseif choice==9,
  clc
  disp(' ')
  htnd
  temp1=htnd;
  htnd=input('Height above reference datum/waterline (ft): ');
  if isempty(htnd),
     htnd=temp1;
  end
```

```
  clear temp1
elseif choice==10,
  clc
  disp(' ')
  ltnd
  temp1=ltnd;
  ltnd=input('Fuselage station (ft): ');
  if isempty(ltnd),
     ltnd=temp1;
  end
  clear temp1
elseif choice==11,
  clc
  disp(' ')
  ytnd
  temp1=ytnd;
  ytnd=input('Position right of buttline (ft): ');
  if isempty(ytnd),
     ytnd=temp1;
  end
  clear temp1
elseif choice==12,
  clc
  disp(' ')
  dian
  temp1=dian;
  dian=input('NOTAR boom diameter (ft): ');
  if isempty(dian),
     dian=temp1;
  end
  clear temp1
elseif choice==13,
  clc
  disp(' ')
  swirl*57.3
  temp1=swirl*57.3;
  swirl=input('Swirl angle at boom(deg): ')/57.3;
  if isempty(swirl),
     swirl=temp1/57.3;
  end
  clear temp1
elseif choice==14,
  clc
  disp(' ')
  Ytmaxn
  temp1=Ytmaxn;
  Ytmaxn=input('Maximum NOTAR thruster force (lbs): ');
  if isempty(Ytmaxn),
     Ytmaxn=temp1;
  end
  clear temp1
elseif choice==15,
  clc
  disp(' ')
  lttnd
  temp1=lttnd;
  lttnd=input('Thruster fuselage station (ft): ');
  if isempty(lttnd),
     lttnd=temp1;
  end
```

```
        clear temp1
      elseif choice==16,
        clc
        disp(' ')
        db1mddele*57.3
        temp1=db1mddele*57.3;
        db1mddele=input('Long  cyclic  pitch  per  inch  def1  (deg/in):
')/57.3;
        if isempty(db1mddele),
           db1mddele=temp1/57.3;
        end
        clear temp1
      elseif choice==17,
        clc
        disp(' ')
        da1mddela*57.3
        temp1=da1mddela*57.3;
        da1mddela=input('Lateral cyclic pitch per inch def1 (deg/in):
')/57.3;
        if isempty(da1mddela),
           da1mddela=temp1/57.3;
        end
        clear temp1
      elseif choice==18,
        clc
        disp(' ')
        dthetomddelc*57.3
        temp1=dthetomddelc*57.3;
        dthetomddelc=input('Collective  pitch  per  inch  def1  (deg/in):
')/57.3;
        if isempty(dthetomddelc),
           dthetomddelc=temp1/57.3;
        end
        clear temp1
      elseif choice==19,
        clc
        disp(' ')
        dthetotddelp*57.3
        temp1=dthetotddelp*57.3;
        disp('Tail rotor pitch change per inch def1 or percentage of
twist')
        disp('Enter 0 (zero) if using NOTAR')
        dthetotddelp=input('(deg/in or deg/deg of twist): ')/57.3;
        if isempty(dthetotddelp),
           dthetotddelp=temp1/57.3;
        end
        clear temp1
      elseif choice==20,
        clc
        disp(' ')
        sidearm/2
        temp1=sidearm/2;
        disp('Maximum deflection of anti-torque from neutral for NOTAR,
enter')
        sidearm=input('1000 if using tail rotor (deg or inch travel):
')*2;
        if isempty(sidearm),
           sidearm=temp1*2;
        if sidearm==0,sidearm=1e3,end
        end
```

```
        clear temp1
        dphinddelp=pi/sidearm; %% pi rad sleeve twist/sidearm defl
      elseif choice==21,
        clc
        disp(' ')
        maxr
        temp1=maxr;
        disp('Displacement of anti-torque control until full rudder')
        disp('  deflection.  Enter 0 (zero) if rudder is fixed')
        maxr=input('   (deg or inch travel): ');
        if isempty(maxr),
           maxr=temp1;
        end
        clear temp1
      elseif choice==0,
        clc
        check2=0;
      else
        disp(' ')
        disp('enter a displayed number ... press any key to continue')
        pause
      end
    end
%
disp(' ')
disp(' ')
disp('                *** SAVE INSTRUCTIONS ***')
disp(' ')
disp('A. Save the new data to a specified file name.')
disp('B. Do not use an extension or quotations.')
disp('C. Use letter/number combinations of 6 characters or less.')
disp('D. The file will be saved with a ".mat" extension.')
disp(' ')
disp('E. If you made no changes or want the same name, press enter.')
disp (' ')
disp('ex: desig2')
    filename=filename1;
    filename1=input('save file as: ','s');
    if isempty(filename1),
       filename1=filename;
    end
    clear check
    eval(['save ',filename1]);
    check=0;
%  end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   *** If creating a new file:  get input for required variables
%       and save under desired file name.  Saves to current
%       directory as a .mat file. ***

  else

  check4=1;
  while check4>0;
  clc
  disp('Do you want to use a Tail Rotor or NOTAR?')
  temp=input('Tail Rotor=0, NOTAR=1: ');
    if temp==0
```

```
      notar=0;
      check4=0;
    elseif temp==1
      notar=1;
      check4=0;
    else
      disp(' ')
      disp('Enter a 0 or 1')
      disp('press any key to continue...')
      pause
    end
  end

  check4=1;
  while check4>0;
  clc
  disp('Do you want to use a controlable vertical tail?')
  temp=input('No=0, Yes=1: ');
    if temp==0
      ctail=0;
      check4=0;
    elseif temp==1
      ctail=1;
      check4=0;
    else
      disp(' ')
      disp('Enter a 0 or 1')
      disp('press any key to continue...')
      pause
    end
  end
  if Swing<.1
    wing=0;
  else
    wing=1;
  end
    clc
    disp('Main rotor')
    disp(' ')
        Ib=input('Blade flapping moment of inertia (slug ft^2): ');
        hmd=input('Hub height above reference datum/waterline (ft): ');
        lmd=input('Hub fuselage station (ft): ');
        ymd=input('Hub position right of buttline (ft): ');
        im=input('Mast incidence (negative forward - deg): ')/57.3;
    clc
  if notar==0
    disp('Tail rotor')
    disp(' ')
        htd=input('Tail rotor height above reference datum/waterline (ft):
');
        ltd=input('Tail rotor fuselage station (ft): ');
        ytd=input('Tail rotor position right of buttline (ft): ');
        bt=input('Number of tail rotor blades: ');
        cot=input('Tail rotor blade chord (ft): ');
        Rt=input('Tail rotor blade radius (ft): ');
        at=input('Average lift curve slope of tail rotor: ');
        ohmt=input('Rotational velocity of tail rotor (rad/sec): ');
        Ibt=input('Tail rotor blade flapping moment of inertia (slug
ft^2): ');
        delta3=input('Delta-3 angle (deg): ')/57.3;
```

```
        thetalt=input('Blade twist (deg): ')/57.3;
        htnd=0;ltnd=0;ytnd=0;dian=0;swirl=0;Ytmaxn=0;lttnd=0;
    elseif notar==1
      clc
      disp('NOTAR')
      disp(' ')
        htnd=input('Height above reference datum/waterline (ft): ');
        ltnd=input('Fuselage station (ft): ');
        ytnd=input('Position right of buttline (ft): ');
        dian=input('NOTAR boom diameter (ft): ');
        swirl=input('Swirl angle at boom (deg): ')/57.3;
        Ytmaxn=input('Maximum NOTAR thruster force (lbs): ');
        lttnd=input('Thruster fuselage stationerence (ft): ');
        htd=0;ltd=0;ytd=0;bt=0;cot=0;Rt=0;at=0;ohmt=0;
        Ibt=0;delta3=0;thetalt=0;
    end
      clc
      disp('Verticle tail')
      disp(' ')
        hvd=input('Height above reference datum/waterline (ft): ');
        lvd=input('Fuselage station (ft): ');
        yvd=input('Position right of buttline (ft): ');
        alplov=input('Zero lift angle for vertical tail (deg): ')/57.3;
        clvertmax=input('Maximum Cl for vertical tail: ');
        qvq=input('Dynamic pressure ratio for tail (pg 489 Prouty): ');
        av=input('Lift curve slope of vertical tail: ');
      clc
      disp('Horizontal tail')
      disp(' ')
        hhd=input('Height above reference datum/waterline (ft): ');
        lhd=input('Fuselage station (ft): ');
        yhd=input('Position right of buttline: ');
        alploh=input('Zero lift angle for horizontal tail (deg): ')/57.3;
        ih=input('Angle of incidence cf horizontal tail (deg): ')/57.3;
        ah=input('Lift curve slope of horizontal tail: ');
        qhq=input('Dynamic pressure ratio for tail (pg 489 Prouty): ');
        vhvl=input('Rotor downwash ratio for h-tail(pg 489 Prouty): ');
        detafdalpfh=input('Fuselage downwash ratio for h-tail (pg 489
Prouty): ');
    if wing==1
      clc
      disp('Wing')
      disp(' ')
        hwd=input('Height above reference datum/waterline (ft): ');
        lwd=input('Fuselage station (ft): ');
        ywd=input('Position right of buttline (ft): ');
        alplow=input('Zero lift angle for wing (deg): ')/57.3;
        iw=input('Angle of incidence of wing (deg): ')/57.3;
        aw=input('Lift curve slope of wing: ');
        ctw=input('Tip cord (ft): ');
        crw=input('Root cord (ft): ');
        vwvl=input('Rotor downwash ratio for wing (pg 489 Prouty): ');
        detafdalpfw=input('Fuselage downwash ratio for wing (pg 489
Prouty): ');
    elseif wing==0
        hwd=0;lwd=0;ywd=0;alplow=0;iw=0;aw=0;ctw=0;crw=0;
        vwvl=0;detafdalpfw=0;
    end
      clc
      disp('CG location')
```

```
    disp(' ')
        zcg=input('CG height above reference datum/waterline (ft): ');
        xcg=input('CG Fuselage station (ft): ');
        ycg=input('CG position right of buttline (ft): ');
    clc
    disp('Fuselage moments of inertia/downwash parameter')
    disp(' ')
        Ixx=input('Ixx (slug ft^2): ');
        Iyy=input('Iyy (slug ft^2): ');
        Izz=input('Izz (slug ft^2): ');
        Ixz=input('Ixz (slug ft^2): ');
        vfv1=input('Downwash ratio for fuselage (page 513 Prouty): ');
    clc
    disp(' Rigging')
    disp(' ')
        dblmddele=input('Long cyclic pitch per inch defl (deg/in):
')/57.3;
        dalmddela=input('Lateral cyclic pitch per inch defl (deg/in):
')/57.3;
        dthetomddelc=input('Collective pitch per inch defl (deg/in):
')/57.3;
    if notar==0
        disp(' ')
        disp('Tail rotor pitch change per inch defl or percentage of
twist')
        dthetotddelp=input('  (deg/in or deg/deg of twist): ')/57.3;
        dphinddelp=0;sidearm=1000;
    elseif notar==1
        disp(' ')
        disp('Max deflection of anti-torque from neutral for NOTAR, enter
')
        sidearm=input('  1000 if using tail rotor (deg or inch travel):
')*2;
        if sidearm==0,sidearm=1000,end
        dphinddelp=pi/sidearm;
    end
    if ctail==1
        disp(' ')
        disp('Displacement of anti-torque control until full rudder')
        maxr=input('  deflection (deg or inch travel): ');
    elseif ctail==0
        maxr=0
    end
    clc
    eval(['save ',filename1]);
    save stabtemp filename1
    check=0;
  end
  if dian==0;
    notar=0;
    if bt==0;
      disp('  You must have a tail rotor or NOTAR/thruster! ')
      check=1
    end
  else
    notar=1;
  end
clc
disp(' ')
disp(' ')
```

```
disp('                        *** DATA ENTRY COMPLETE ***')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp(' ')
disp('                *** EVALUATING STABILITY DERIVATIVES ***')
pause(3)

if Vinf<20
   hover   % call hover routine
elseif Vinf>=20
   cruise  % call cruise routine
end
stabout

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B. TRIM.M

```
%                          TRIM.M
%  Rotor trim subroutine
disp(' ')
disp('                    *** EXECUTING ROTOR TRIM ROUTINE ***')
%   *** calculation of required parameters ***

rho=.002377*(-.000031*PA+(-.002*temp+1.118));

%   *** first guess at rotor profile drag ( H force) ***

if Vinf < 16.9,
  Drotor=0;
else
  Drotor=Vinf*(rho/.002377);
end

q=0.5*rho*Vinf^2;
Adisk=pi*R^2;
Vtip=omega*R;
Dfuse=q*Afh;
CDwing=CDowing+(CLwing^2/(ewing*pi*(bwing^2/Swing)));
CDhoriz=CDohoriz+(CLhoriz^2/(.8*pi*(bhoriz^2/Shoriz)));
CDvert=CDovert+(CLvert^2/(.8*pi*(bvert^2/Svert)));
Dwing=q*CDwing*Swing;
Dhoriz=q*CDhoriz*Shoriz;
Dvert=q*CDvert*Svert;
Dftotal=(Dfuse+Dwing+Dhoriz+Dvert)-Taux;
Lwing=q*CLwing*Swing;
Lhoriz=q*CLhoriz*Shoriz;
Lvert=q*CLvert*Svert;
Lftotal=Lwing+Lhoriz;
alphaT=atan2((Dftotal+Drotor),(GW-Lftotal));
mu=Vinf*cos(alphaT)/Vtip;


%   *** thrust calculation ***

  if Vinf < 16.9,
    T=(1+(0.3*Afv/Adisk))*GW;
    CT=T/(Adisk*rho*Vtip^2);
  else
    T=(GW-Lftotal)/cos(alphaT);
    CT=T/(Adisk*rho*Vtip^2);
  end


%   *** setup blade radius elements, azimuth elements,
%       induced velocity distributions, and determination
%       of coning angle and tip loss parameter ***

B=1-(sqrt(2*CT)/b);
Reff=B*R;
Rbar=Reff-e;
dr=(Reff-grip)/nbe;
r=grip:dr:Reff-dr;,r=r+dr/2;
rT1=0.7;,% *** first guess at rT ***
RbarT=rT1*Rbar;
mblade=wblade/32.17;
```

```
betao=asin((T/b*RbarT-(.5*(R-e)+e)*wblade)/((.5*(R-e)+e)^2*omega^2*mblad
e));
betat=twist*(0.7-(r/R));
psi=0:360/naz:360-360/naz;,psi=psi'/57.3;


%   *** induced velocity determination ***

  if Vinf < 16.9,
    A=4*pi;
    Bv=(b/2)*omega*a*cblade;
    Tv=0;
    delT=T-Tv;
    while abs(delT) > .01*T
      thetav=twist*(0.7-(r/R))+thetao;
      C=(-b/2)*cblade*omega^2*r*a.*thetav;
      vi=(-Bv+sqrt(Bv^2-(4*A*C)))/(2*A);
      dTv=(b/2)*rho*((omega*r).^2)*a.*(thetav-(vi./(omega*r)))*cblade*dr;
      Tv=sum(dTv);
      delT=T-Tv;
        if delT < 0,
          thetao=thetao-0.5*thetao*abs(delT/T);
        else
          thetao=thetao+0.5*thetao*abs(delT/T);
        end
    end
  else
    lamdaT=[1 -2*mu*sin(alphaT) mu^2*(sin(alphaT)^2+1)-2*mu^3*sin(alphaT)
mu^4*sin(alphaT)^2-CT^2/4];
    lamdaT=max(real(roots(lamdaT)));
    vi=lamdaT*Vtip-Vinf*sin(alphaT);
    vi=vi*ones(r);
  end


%   *** first guess at theta ***

theta1c=0.035*((0.0006e-3*Vinf^2+0.244e-3*Vinf)/0.105);
theta1s=-0.087*((0.0006e-3*Vinf^2+0.244e-3*Vinf)/0.105);
theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);



%   *** rotor trimming routine ***

disp(' ')
disp(' ')
disp('                          *** TRIMMING COLLECTIVE ***')

k=1;
error0=(T*.02)+1;

  while abs(error0) > T*.02
    Tpsi=zeros(psi);
    thrcalc
    error0=T-(mean(Tpsi)*b);
      if error0 < -T*.02,
        thetao=thetao-0.35*thetao*abs(1.5*error0/T)*(1-mu);
      elseif error0 > T*.02,
        thetao=thetao+0.35*thetao*abs(1.5*error0/T)*(1-mu);
```

```
    end
  theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
  if k > 1,
    if abs(error0) > abs(error1),
      clc
      disp(' ')
      disp(' ')
      disp('This configuration will not trim')
      disp('Try a lower airspeed or a new design')
      disp(' ')
      disp('Program execution terminated')
      disp(' ')
      error('*** END OF PROGRAM ***')
    end
  end
  error1=error0;
  k=k+1;
end


disp(' ')
disp('                          *** TRIMMING CYCLIC ***')

t0=clock;
k=1;
error0=(((T/b)*rT1*(R-grip))*.04)+1;

  while error0 > ((T/b)*rT1*(R-grip))*.04

    time=etime(clock,t0);
    if time > 15,
      disp(' ')
      disp('still trimming ...')
      t0=clock;
    end

    Mpsi(:,k)=zeros(psi);
    tmcalc
    theta=[theta theta(:,k)];
    Mpsi=[Mpsi Mpsi(:,k)];

%   *** calculation of initial dthetadM ***

      if k < 2,
        theta(:,k+1)=theta(:,k)+0.25/57.3;
        Mpsi(:,k+1)=zeros(psi);
        k=k+1;
        tmcalc
        k=k-1;
        dthetadM=(theta(:,k+1)-theta(:,k))./(Mpsi(:,k+1)-Mpsi(:,k));
      end

%   *** calculation of M first harmonic parameters ***

    M1c=2*sum(Mpsi(:,k).*cos(psi))/naz;
    M1s=2*sum(Mpsi(:,k).*sin(psi))/naz;

%   *** removal of first harmonic terms from Mpsi ***

    Mpsi(:,k+1)=Mpsi(:,k)-M1c.*cos(psi)-M1s.*sin(psi);
```

```
    delM=Mpsi(:,k+1)-Mpsi(:,k);
    error0=max(delM)-min(delM);
    if k > 1,
      if error0 > error1,
        clc
        disp(' ')
        disp(' ')
        disp('This configuration will not trim')
        disp('Try a lower airspeed or a new design')
        disp(' ')
        disp('Program execution terminated')
        disp(' ')
        error('*** END OF PROGRAM ***')
      end
    end
    error1=error0;

%   *** calculation of new theta ***

    delM=0.5*(1-mu)*delM;
    theta(:,k+1)=theta(:,k)+(dthetadM.*delM);

    if error0 <= ((T/b)*rT1*(R-grip))*.04,
      theta1c=2*sum(theta(:,k).*cos(psi))/naz;
      theta1s=2*sum(theta(:,k).*sin(psi))/naz;
    else
      theta1c=2*sum(theta(:,k+1).*cos(psi))/naz;
      theta1s=2*sum(theta(:,k+1).*sin(psi))/naz;
    end

    theta(:,k+1)=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);

%   *** calculation of new dthetadM ***

    theta=[theta theta(:,k+1)];
    Mpsi=[Mpsi Mpsi(:,k+1)];
    theta(:,k+2)=theta(:,k)+0.25/57.3;
    Mpsi(:,k+2)=zeros(Mpsi(:,k+1));
    k=k+2;
    tmcalc
    k=k-2;
    dthetadM=(theta(:,k+2)-theta(:,k))./(Mpsi(:,k+2)-Mpsi(:,k));
    k=k+1;
  end


disp(' ')
disp('                          *** ADJUSTING COLLECTIVE ***')
disp(' ')

theta=theta(:,k);
k=1;
error0=(T*.01)+1;

  while abs(error0) > T*.01
    Tpsi=zeros(psi);
    thrcalc
    error0=T-(mean(Tpsi)*b);
      if error0 < -T*.01,
        thetao=thetao-0.25*thetao*abs(1.25*error0/T)*(1-mu);
```

```
      elseif error0 > T*.01,
        thetao=thetao+0.25*thetao*abs(1.25*error0/T)*(1-mu);
      end
    theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
    if k > 1,
      if abs(error0) > abs(error1),
        clc
        disp(' ')
        disp(' ')
        disp('This configuration will not trim')
        disp('Try a lower airspeed or a new design')
        disp(' ')
        disp('Program execution terminated')
        disp(' ')
        error('*** END OF PROGRAM ***')
      end
    end
    error1=error0;
    k=k+1;
  end


% *** calculating drag moments ***

DMpsi=zeros(psi);
dmcalc


%   *** calculating rotor H force ***

  if Vinf < 16.9,
    Hrotor=0;
    dT=[dT ddT];
    dD=[dD ddD];
  else
    dT=[dT ddT];
    dD=[dD ddD];
      for i=1:length(r)+1,
        H1c(i)=2*sum(dT(:,i).*cos(psi))/naz;
        H1s(i)=2*sum(dD(:,i).*sin(psi))/naz;
      end
    Hrotor=(((b*cos(alphaT)/2)*(sum(H1s)-sin(betao)*sum(H1c)))+Drotor)/2;
end


%   *** calculating new rT ***

rT2=(((mean(Mpsi(:,length(Mpsi(1,:))-1))/mean(Tpsi))/R)+rT1)/2;


%   *** check rotor drag and rT, retrim rotor if required ***

while abs(Drotor-Hrotor) > 0.2*Hrotor | abs(rT1-rT2) > 0.015*rT1

  if abs(Drotor-Hrotor) > 0.2*Hrotor,
    disp(' ')
    disp('                        *** ADJUSTING ROTOR DRAG ***')
  end
  Drotor=Hrotor;
```

```
  if abs(rT1-rT2) > 0.015*rT1,
    disp(' ')
    disp('                    *** ADJUSTING MEAN THRUST LOCATION ***')
  end


  disp(' ')
  disp('                          *** RETRIMMING ROTOR ***')
  disp(' ')
  dT=dT(:,1:nbe);
  dD=dD(:,1:nbe);


%   *** recalculating parameters ***

  alphaT=atan((Dftotal+Drotor)/(GW-Lftotal));
  mu=Vinf*cos(alphaT)/Vtip;

    if Vinf >= 16.9,
      T=(GW-Lftotal)/cos(alphaT);
      CT=T/(Adisk*rho*Vtip^2);
            l a m d a T = [ 1    - 2 * m u * s i n ( a l p h a T )
mu^2*(sin(alphaT)^2+1)-2*mu^3*sin(alphaT) mu^4*sin(alphaT)^2-CT^2/4];
      lamdaT=max(real(roots(lamdaT)));
      vi=    'aT*Vtip-Vinf*sin(alphaT);
      vi=v..ones(r);
    end

  B=1-(sqrt(2*CT)/b);
  Reff=B*R;
  Rbar=Reff-e;
  dr=(Reff-grip)/nbe;
  r=grip:dr:Reff-dr;,r=r+dr/2;
  RbarT=rT2*Rbar;

betao=asin((T/b*RbarT-(.5*(R-e)+e)*wblade)/((.5*(R-e)+e)^2*omega^2*mblad
e));


%   *** trimming collective ***

  t0=clock;
  k=1;
  error0=(T*.02)+1;

    while abs(error0) > T*.02
      Tpsi=zeros(psi);
      thrcalc
      error0=T-(mean(Tpsi)*b);
      if error0 < -T*.02,
        thetao=thetao-0.35*thetao*abs(1.5*error0/T)*(1-mu);
      elseif error0 > T*.02,
        thetao=thetao+0.35*thetao*abs(1.5*error0/T)*(1-mu);
      end
      theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
      if k > 1,
        if abs(error0) > abs(error1),
          clc
          disp(' ')
          disp(' ')
```

```
        disp('This configuration will not trim')
        disp('Try a lower airspeed or a new design')
        disp(' ')
        disp('Program execution terminated')
        disp(' ')
        error('*** END OF PROGRAM ***')
      end
    end
    error1=error0;
    k=k+1;
  end


%   *** trimming cyclic ***

  k=1;
  error0=(((T/b)*rT2*(R-grip))*.04)+1;

  while error0 > ((T/b)*rT2*(R-grip))*.04

    time=etime(clock,t0);
    if time > 15,
      disp('still trimming ...')
      disp(' ')
      t0=clock;
    end

    Mpsi(:,k)=zeros(psi);
    tmcalc
    theta=[theta theta(:,k)];
    Mpsi=[Mpsi Mpsi(:,k)];

%   *** calculation of initial dthetadM ***

    if k < 2,
      theta(:,k+1)=theta(:,k)+0.25/57.3;
      Mpsi(:,k+1)=zeros(psi);
      k=k+1;
      tmcalc
      k=k-1;
      dthetadM=(theta(:,k+1)-theta(:,k))./(Mpsi(:,k+1)-Mpsi(:,k));
    end

%   *** calculation of M first harmonic parameters ***

    M1c=2*sum(Mpsi(:,k).*cos(psi))/naz;
    M1s=2*sum(Mpsi(:,k).*sin(psi))/naz;

%   *** removal of first harmonic terms from Mpsi ***

    Mpsi(:,k+1)=Mpsi(:,k)-M1c.*cos(psi)-M1s.*sin(psi);
    delM=Mpsi(:,k+1)-Mpsi(:,k);
    error0=max(delM)-min(delM);
    if k > 1,
      if error0 > error1,
        clc
        disp(' ')
        disp(' ')
        disp('This configuration will not trim')
        disp('Try a lower airspeed or a new design')
```

```
        disp(' ')
        disp('Program execution terminated')
        disp(' ')
        error('*** END OF PROGRAM ***')
      end
    end
    error1=error0;

%   *** calculation of new theta ***

    delM=0.5*(1-mu)*delM;
    theta(:,k+1)=theta(:,k)+(dthetadM.*delM);
    if error0 <= ((T/b)*rT2*(R-grip))*.04,
      theta1c=2*sum(theta(:,k).*cos(psi))/naz;
      theta1s=2*sum(theta(:,k).*sin(psi))/naz;
    else
      theta1c=2*sum(theta(:,k+1).*cos(psi))/naz;
      theta1s=2*sum(theta(:,k+1).*sin(psi))/naz;
    end
    theta(:,k+1)=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);

%   *** calculation of new dthetadM ***

    theta=[theta theta(:,k+1)];
    Mpsi=[Mpsi Mpsi(:,k+1)];
    theta(:,k+2)=theta(:,k)+0.25/57.3;
    Mpsi(:,k+2)=zeros(Mpsi(:,k+1));
    k=k+2;
    tmcalc
    k=k-2;
    dthetadM=(theta(:,k+2)-theta(:,k))./(Mpsi(:,k+2)-Mpsi(:,k));
    k=k+1;
  end


%   *** retrimming collective ***

  theta=theta(:,k);
  k=1;
  error0=(T*.01)+1;

  while abs(error0) > T*.01
    Tpsi=zeros(psi);
    thrcalc
    error0=T-(mean(Tpsi)*b);
      if error0 < -T*.01,
        thetao=thetao-0.25*thetao*abs(1.25*error0/T)*(1-mu);
      elseif error0 > T*.01,
        thetao=thetao+0.25*thetao*abs(1.25*error0/T)*(1-mu);
      end
    theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
    if k > 1,
      if abs(error0) > abs(error1),
        clc
        disp(' ')
        disp(' ')
        disp('This configuration will not trim')
        disp('Try a lower airspeed or a new design')
        disp(' ')
        disp('Program execution terminated')
```

```
        disp(' ')
        error('*** END OF PROGRAM ***')
      end
    end
    error1=error0;
    k=k+1;
  end    .

%   *** recalculating rotor H force ***

  if Vinf < 16.9,
    Hrotor=0;
    dT=[dT ddT];
    dD=[dD ddD];
  else
    dT=[dT ddT];
    dD=[dD ddD];
      for i=1:length(r)+1,
        H1c(i)=2*sum(dT(:,i).*cos(psi))/naz;
        H1s(i)=2*sum(dD(:,i).*sin(psi))/naz;
      end
    Hrotor=(((b*cos(alphaT)/2)*(sum(H1s)-sin(betao)*sum(H1c)))+Drotor)/2;
  end

%   *** recalculating rT ***

  rT1=rT2;
  rT2=(((mean(Mpsi(:,length(Mpsi(1,:))-1))/mean(Tpsi))/R)+rT1)/2;
end

% *** recalculating drag moments ***

dT=dT(:,1:nbe);
dD=dD(:,1:nbe);
DMpsi=zeros(psi);
dmcalc
dT=[dT ddT];
dD=[dD ddD];
clc
disp(' ')
disp(' ')
disp('                              *** ROTOR TRIMMED ***')
pause(3)
```

C. CRUISE.M

```
% CRUISE.M
% CALLED BY STAB.M
% Computes the stability derivatives in cruise flight.
% calls the following subroutines
%
% CMRGRP
% CTRGRP
% CFUSEGRP
% TRIM
% DCTPLOTS or DCTMATS
%
% evaluate basic stability derivatives
%
% MAIN ROTOR
% dctsigdmu    dctsigdtheto    dctsigdlamp
% dchsigdmu    dchsigdtheto    dchsigdlamp
% dcqsigdmu    dcqsigdtheto    dcqsigdlamp
% da1dmu       da1dtheto       da1dlamp
% db1dmu       db1dtheto       db1dlamp
%
% TAIL ROTOR
% dctsigdlampt dctsigdmut dctsigdthetot
%
vctsig=ones(1,3);
vcqsig=ones(1,3);
vchsig=ones(1,3);
valtpp=ones(1,3);
vA1=ones(1,3);
vB1=ones(1,3);
va1s=ones(1,3);
vb1s=ones(1,3);
vmu=ones(1,3);
vtheta7=ones(1,3);
vao=ones(1,3);
vv1=ones(1,3);
vlamp=ones(1,3);
vthetao=ones(1,3);
vctsigt=ones(1,3);
vmut=ones(1,3);
vlampt=ones(1,3);
vthetaot=ones(1,3);
%
theta1=-twist;
eval(['load ',filename1])
Vinf=Vinf*1.0005;  % .05% higher
disp(' ')
disp('                *** FIRST OF THREE TRIM ITERATIONS ***')
disp(' ')
pause(3)
trim
Qrotor=mean(DMpsi)*b;
solidity=b*cblade/(pi*R);
CQ=Qrotor/(Adisk*rho*Vtip^2*R);
CH=Hrotor/(Adisk*rho*Vtip^2);
vctsig(1,3)=CT/solidity;
vcqsig(1,3)=CQ/solidity;
vchsig(1,3)=CH/solidity;
valtpp(1,3)=-alphaT;
```

```
vA1(1,3)=-theta1c;
vB1(1,3)=-theta1s;
vmu(1,3)=mu;
thetasave=thetao;
vthetao(1,3)=thetasave-.7*theta1;
vtheta7(1,3)=vthetao(1,3)+.7*theta1;
vao(1,3)=betao;
vv1(1,3)=mean(vi);
vlamp(1,3)=vmu(1,3)*valtpp(1,3)-vv1(1,3)/omega/R;
lockno=rho*a*cblade*R^4/Ib;
num1=vthetao(1,3)*(8/3+32/45*vmu(1,3)^3/pi);
num2=theta1*(2+vmu(1,3)^4/12);
num3=vlamp(1,3)*(2-vmu(1,3)*vmu(1,3)/2);
den1=1+3/2*vmu(1,3)*vmu(1,3)-5*vmu(1,3)^4/24;
va1s(1,3)=vmu(1,3)*(num1+num2+num3)/den1-vB1(1,3);
vb1s(1,3)=va1s(1,3)*12/lockno*e/R/(1+e/R/3);
cqsig=CQ/solidity;
ohm=omega;
sigma=solidity;
lv=lvd-xcg;
g=32.2;
A=pi*R*R;
%
% tail rotor
%
if notar==0
  Abt=bt*Rt*cot;
  At=pi*Rt*Rt;
  mut=Vinf/ohmt/Rt;
  sigmat=Abt/At;
  lt=ltd-xcg;
  ht=htd-zcg;
  yt=ytd-ycg;
  locknot=rho*at*cot*Rt^4/Ibt;
  Tt=(cqsig*sigma*rho*A*(ohm*R)^2*R-Lvert*lv)/lt;
  ctsigt=Tt/(rho*At*(ohmt*Rt)^2);
  lampt=-ctsigt*sigma/2/mu;
  aot=2/3*locknot*ctsigt/at-3/2*g*Rt^2/(ohmt*Rt)^2;

a1st=(-2*((4/3*mut*aot+ctsigt*sigmat/2/mut)/(2+mut*mut))*(1-(4*mut/(2+3*
mut*mut))^2)*...

tan(delta3))/(((2-mut*mut)/(2+3*mut*mut))+(1-(4*mut/(2+3*mut*mut))^2)*ta
n(delta3)^2);
  b1st=2*((4/3*mu*aot*ctsigt*sigma/2/mu)/(2+mu*mu))+a1st*tan(delta3);

thetaot=((4*ctsigt/at)-(.5+mut*mut/2)*theta1t+mut*b1st*tan(delta3)-lampt
)/...
   (2/3+mut*mut)-aot*tan(delta3);
  theta75t=thetaot+.75*theta1t;
  v1t=sqrt(Tt/2/rho/At);
  vctsigt(1,3)=ctsigt;
  vmut(1,3)=mut;
  vthetaot(1,3)=thetaot;
  vlampt(1,3)=lampt;
end
%
save stabtemp vctsig vcqsig vchsig valtpp vA1 vB1 vmu vthetao theta1 ...
 vtheta7 vao vv1 vlamp lockno va1s vb1s rho filename1 vctsigt vmut ...
 vthetaot vlampt g
```

```
clear
load stabtemp
eval(['load ',filename1])
load stabtemp
Vinf=Vinf*.9995;  % .05% lower
V=Vinf;
disp(' ')
disp('                *** SECOND OF THREE TRIM ITERATIONS ***')
disp(' ')
pause(3)
trim
Qrotor=mean(DMpsi)*b;
solidity=b*cblade/(pi*R);
CQ=Qrotor/(Adisk*rho*Vtip^2*R);
CH=Hrotor/(Adisk*rho*Vtip^2);
vctsig(1,1)=CT/solidity;
vcqsig(1,1)=CQ/solidity;
vchsig(1,1)=CH/solidity;
valtpp(1,1)=-alphaT;
vA1(1,1)=-theta1c;
vB1(1,1)=-theta1s;
vmu(1,1)=mu;
thetasave=thetao;
vthetao(1,1)=thetasave-.7*theta1;
vtheta7(1,1)=vthetao(1,1)+.7*theta1;
vao(1,1)=betao;
vv1(1,1)=mean(vi);
vlamp(1,1)=vmu(1,1)*valtpp(1,1)-vv1(1,1)/omega/R;
lockno=rho*a*cblade*R^4/Ib;
num1=vthetao(1,1)*(8/3+32/45*vmu(1,1)^3/pi);
num2=theta1*(2+vmu(1,1)^4/12);
num3=vlamp(1,1)*(2-vmu(1,1)*vmu(1,1)/2);
den1=1+3/2*vmu(1,1)*vmu(1,1)-5*vmu(1,1)^4/24;
va1s(1,1)=vmu(1,1)*(num1+num2+num3)/den1-vB1(1,1);
vb1s(1,1)=va1s(1,1)*12/lockno*e/R/(1+e/R/3);
cqsig=CQ/solidity;
ohm=omega;
sigma=solidity;
lv=lvd-xcg;
A=pi*R*R;
%
%tail rotor
%
if notar==0
  Abt=bt*Rt*cot;
  At=pi*Rt*Rt;
  mut=Vinf/ohmt/Rt;
  sigmat=Abt/At;
  lt=ltd-xcg;
  ht=htd-zcg;
  yt=ytd-ycg;
  locknot=rho*at*cot*Rt^4/Ibt;
  Tt=(cqsig*sigma*rho*A*(ohm*R)^2*R-Lvert*lv)/lt;
  ctsigt=Tt/(rho*At*(ohmt*Rt)^2);
  lampt=-ctsigt*sigma/2/mu;
  aot=2/3*locknot*ctsigt/at-3/2*g*Rt^2/(ohmt*Rt)^2;

a1st=(-2*((4/3*mut*aot+ctsigt*sigmat/2/mut)/(2+mut*mut))*(1-(4*mut/(2+3*
mut*mut))^2)*...
```

```
tan(delta3))/(((2-mut*mut)/(2+3*mut*mut))+(1-(4*mut/(2+3*mut*mut))^2)*ta
n(delta3)^2);
  blst=2*((4/3*mu*aot*ctsigt*sigma/2/mu)/(2+mu*mu))+alst*tan(delta3);

thetaot=((4*ctsigt/at)-(.5+mut*mut/2)*thetalt+mut*blst*tan(delta3)-lampt
)/...
  (2/3+mut*mut)-aot*tan(delta3);
  theta75t=thetaot+.75*thetalt;
  vlt=sqrt(Tt/2/rho/At);
  vctsigt(1,1)=ctsigt;
  vmut(1,1)=mut;
  vthetaot(1,1)=thetaot;
  vlampt(1,1)=lampt;
end
%
save stabtemp vctsig vcqsig vchsig valtpp vA1 vB1 vmu vthetao theta1 ...
 vtheta7 vao vv1 vlamp lockno va1s vb1s rho filename1 vctsigt vmut ...
 vthetaot vlampt g
clear
load stabtemp
eval(['load ',filename1])
load stabtemp
disp(' ')
disp('                    *** THIRD OF THREE TRIM ITERATIONS ***')
disp(' ')
pause(3)
trim
Qrotor=mean(DMpsi)*b;
solidity=b*cblade/(pi*R);
CQ=Qrotor/(Adisk*rho*Vtip^2*R);
CH=Hrotor/(Adisk*rho*Vtip^2);
disp(' ')
disp('                          *** PERTURBATIONS COMPLETE ***')
disp(' ')
disp('                    *** EVALUATING STABILITY DERIVATIVES ***')
disp(' ')
vctsig(1,2)=CT/solidity;
vcqsig(1,2)=CQ/solidity;
vchsig(1,2)=CH/solidity;
valtpp(1,2)=-alphaT;
vA1(1,2)=-theta1c;
vB1(1,2)=-theta1s;
vmu(1,2)=mu;
thetasave=thetao;
vthetao(1,2)=thetasave-.7*theta1;
vtheta7(1,2)=vthetao(1,2)+.7*theta1;
vao(1,2)=betao;
vv1(1,2)=mean(vi);
vlamp(1,2)=vmu(1,2)*valtpp(1,2)-vv1(1,2)/omega/R;
V=Vinf;
lockno=rho*a*cblade*R^4/Ib;
num1=vthetao(1,2)*(8/3+32/45*vmu(1,2)^3/pi);
num2=theta1*(2+vmu(1,2)^4/12);
num3=vlamp(1,2)*(2-vmu(1,2)*vmu(1,2)/2);
den1=1+3/2*vmu(1,2)*vmu(1,2)-5*vmu(1,2)^4/24;
va1s(1,2)=vmu(1,2)*(num1+num2+num3)/den1-vB1(1,2);
vb1s(1,2)=va1s(1,2)*12/lockno*e/R/(1+e/R/3);
cqsig=CQ/solidity;
ohm=omega;
```

```
sigma=solidity;
lv=lvd-xcg;
A=pi*R*R;
%
% tail rotor
%
if notar==0
  Abt=bt*Rt*cot;
  At=pi*Rt*Rt;
  mut=Vinf/ohmt/Rt;
  sigmat=Abt/At;
  lt=ltd-xcg;
  ht=htd-zcg;
  yt=ytd-ycg;
  locknot=rho*at*cot*Rt^4/Ibt;
  Tt=(cqsig*sigma*rho*A*(ohm*R)^2*R-Lvert*lv)/lt;
  ctsigt=Tt/(rho*At*(ohmt*Rt)^2);
  lampt=-ctsigt*sigma/2/mu;
  aot=2/3*locknot*ctsigt/at-3/2*g*Rt^2/(ohmt*Rt)^2;

a1st=(-2*((4/3*mut*aot+ctsigt*sigmat/2/mut)/(2+mut*mut))*(1-(4*mut/(2+3*
mut*mut))^2)*...

tan(delta3))/(((2-mut*mut)/(2+3*mut*mut))+(1-(4*mut/(2+3*mut*mut))^2)*ta
n(delta3)^2);
  b1st=2*((4/3*mu*aot*ctsigt*sigma/2/mu)/(2+mu*mu))+a1st*tan(delta3);

thetaot=((4*ctsigt/at)-(.5+mut*mut/2)*theta1t+mut*b1st*tan(delta3)-lampt
)/...
  (2/3+mut*mut)-aot*tan(delta3);
  theta75t=thetaot+.75*theta1t;
  v1t=sqrt(Tt/2/rho/At);
  vctsigt(1,2)=ctsigt;
  vmut(1,2)=mut;
  vthetaot(1,2)=thetaot;
  vlampt(1,2)=lampt;
end
%
save stabtemp vctsig vcqsig vchsig valtpp vA1 vB1 vmu vthetao theta1 ...
 vtheta7 vao vv1 vlamp lockno va1s vb1s T rho filename1
%
% Solving for unknowns
%
DCTPLOTS
%
dcysigdb=inv(T*sin(.1));
%
if notar==0
dctplott
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONFIGURATION CALCULATION
%
% converting data from crank
theta7=vthetao(1,2);
thetao=theta7-.7*theta1;
ctsig=vctsig(1,2);
chsig=vchsig(1,2);
cqsig=vcqsig(1,2);
altpp=valtpp(1,2);
```

```
A1=vA1(1,2);
B1=vB1(1,2);
ao=vao(1,2);
v1=vv1(1,2);
V=Vinf;
% Main rotor
hm=hmd-zcg;
ym=ymd-ycg;
lm=lmd-xcg;
c=cblade;
Ab=c*R*4;
lamp=vlamp(1,2);
mu=vmu(1,2);
ct=CT;
f=Afv;
thetat=4/a*ctsig+sqrt(sigma*ctsig/2);
lockno=rho*a*c*R^4/Ib;
theta75=thetao+.75*theta1;
a1s=va1s(1,2);
b1s=vb1s(1,2);
m=GW/g;
Ic=Ixx*Izz;
lh=lhd-xcg;
hh=hhd-zcg;
yh=yhd-ycg;
hv=hvd-zcg;
yv=yvd-ycg;
delvmax=pi/4;    %est max verticle fin defl before stall
if maxr==0
  ddelvddelp=0;
else
  ddelvddelp=delvmax/maxr;   % verticle fin deflection/maxr
end
if notar==1;
  % NOTAR
  v1max=1.2*v1;      %  downwash at the boom slots (NOTAR)
  ltn=lttnd-xcg;
  ln=ltnd-xcg;
  yn=ytnd-xcg;
  hn=htnd-zcg;
  htn=hn;
  Tt=(cqsig*sigma*rho*A*(ohm*R)^2*R-Lvert*lv)/ltn;
  cmun=.55;  % a typical value
  bn=.5*R;
  hslot=.028*dian/2;
  arn=bn/dian;
  At=pi*dian*dian/4;
  Rt=dian/2;
end
if Swing<.1
  wing=0;
else
  Aw=Swing;
  lw=lwd-xcg;
  hw=hwd-zcg;
  yw=ywd-ycg;
  alpw=CLwing/aw;
  deliw=ewing;
  Zw=Lwing;
  bw=bwing;
```

```
  cdow=CDowing;
  wing=1;
end
Av=Svert;
bv=bvert;
Ah=Shoriz;
bh=bhoriz;
cdov=CDovert;
cdoh=CDohoriz;
deliv=ewing;
delih=ewing;
alph=CLhoriz/ah;
Zh=Lhoriz;
Yv=Lvert;
if Yv<.1;Yv=.1;end;
deldv=1.72*(1-b  (20+Rt))*(2*Yv/(2*Rt+bv));
% trim conditions
tho=-T*(im+als)/GW;
pho=-(T*bls+Tt)/GW;
wo=0;
vo=0;
uo=Vinf;
gamc=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% STABILITY CALCULATION
%
cmrgrp
ctrgrp
cbodygrp
%
% computation of A,B,C,D matrices
%
%  CHECK THE WING CALCULATIONS
%
A11= (dxdxdotm+dxdxdoth+dxdxdotv+dxdxdotf+dxdxdotw)/m;
A12= (dxdzdotm+dxdzdoth+dxdzdotf+dxdzdotw)/m;
A13= (dxdqm)/m-wo;
A14= -g*cos(tho);
A15= (dxdydotm+dxdydotv)/m;
A16= dxdpm/m;
A17= 0;
A18= 0;%(dxdrw)/m+vo; recheck this one
%
A21= (dzdxdotm+dzdxdoth+dzdxdotf+dzdxdotw)/m;
A22= (dzdzdotm+dzdzdoth+dzdzdotf+dzdzdotw)/m;
A23= (dzdqh+dzdqw)/m+uo;
A24= -g*cos(pho)*sin(tho);
A25= 0;%      (dzdydotw)/m;
A26= 0;%      (dzdpw)/m-vo;
A27= -g*sin(pho)*cos(tho);
A28= dzdrm/m;%  (dzdrm+dzdrw)/m;  % recheck this one
%
A31= (dmdxdotm+dmdxdoth+dmdxdotf+dmdxdotw)/Iyy;
A32= (dmdzdotm+dmdzdoth+dmdzdotf+dmdzdotw)/Iyy;
A33= (dmdqm+dmdqh+dmdqw)/Iyy;
A34= 0;
A35= dmdydotm/Iyy;
A36= (dmdpm)/Iyy;
A37= 0;
```

```
A38= 0;%  dmdrf/Iyy;
%
A41= 0;
A42= 0;
A43= cos(pho);
A44= 0;
A45= 0;
A46= 0;
A47= 0;
A48= -sin(pho);
%
A51= (dydxdotm+dydxdott+dydxdotn+dydxdotv)/m;
A52= (dydzdotm+dydzdotn)/m;
A53= dydqm/m;
A54= -g*sin(pho)*sin(tho);
A55= (dydydotm+dydydott+dydydotn+dydydotv+dydydotf)/m;
A56= (dydpm+dydpt+dydpn+dydpv)/m+wo;
A57= g*cos(pho)*cos(tho);
A58= (dydrt+dydrn+dydrv)/m-uo;
%
A61= (Izz*(drdxdotm+drdxdott+drdxdotn+drdxdotv)...
+Ixz*(dndxdotm+dndxdott+dndxdotn+dndxdotv))/Ic;
A62= (Izz*(drdzdotm+drdzdotn)+Ixz*(dndzdotm+dndzdotn))/Ic;
A63= (Izz*(drdqm)+Ixz*(0))/Ic; % recheck this one
A64= 0;
A65= (Izz*(drdydotm+drdydott+drdydotn+drdydotv+drdydotf)...
+Ixz*(dndydott+dndydotn+dndydotv+dndydotf))/Ic;
A66= (Izz*(drdpm+drdpt+drdpn+drdpv+drdpw)+Ixz*(dndpt+dndpn+dndpv))/Ic;
A67= 0;
A68= (Izz*(drdrt+drdrn+drdrv+drdrw)+Ixz*(dndrm+dndrt+dndrn+dndrv))/Ic;
%
A71= 0;
A72= 0;
A73= sin(pho)*tan(tho);
A74= 0;
A75= 0;
A76= 1;
A77= 0;
A78= cos(pho)*tan(tho);
%
A81= (Ixz*(drdxdotm+drdxdott+drdxdotn+drdxdotv)...
+Ixx*(dndxdotm+dndxdott+dndxdotn+dndxdotv))/Ic;
A82= (Ixz*(drdzdotm+drdzdotn)+Ixx*(dndzdotm+dndzdotn))/Ic;
A83= (Ixz*(drdqm)+Ixx*(0))/Ic;%  recheck
A84= 0;
A85= (Ixz*(drdydotm+drdydott+drdydotn+drdydotv+drdydotf)...
+Ixx*(dndydott+dndydotn+dndydotv+dndydotf))/Ic;
A86= (Ixz*(drdpm+drdpt+drdpn+drdpv+drdpw)+Ixx*(dndpt+dndpn+dndpv))/Ic;
A87=  0;
A88=(Ixz*(drdrt+drdrv+drdrw)+Ixx*(dndrt+dndrv))/Ic;
%
% longitudinal plant augmented is  X=[u w q theta]'
Flonaug=[A11 A12 A13 A14;A21 A22 A23 A24;A31 A32 A33 A34;
A41 A42 A43 A44];
Plonaug=poly(Flonaug);
Rlonaug=roots(Plonaug);
% lateral plant augmented is  X=[v p phi r psi]'
Flataug=[A55 A56 A57 A58 0;A65 A66 A67 A68 0;
A75 A76 A77 A78 0;A85 A86 A87 A88 0;0 0 0 1 0];
Plataug=poly(Flataug);
```

```
Rlataug=roots(Plataug);
% coupled plant
Amat=[A11 A12 A13 A14 A15 A16 A17 A18 0;
A21 A22 A23 A24 A25 A26 A27 A28 0;
A31 A32 A33 A34 A35 A36 A37 A38 0;
A41 A42 A43 A44 A45 A46 A47 A48 0;
A51 A52 A53 A54 A55 A56 A57 A58 0;
A61 A62 A63 A64 A65 A66 A67 A68 0;
A71 A72 A73 A74 A75 A76 A77 A78 0;
A81 A82 A83 A84 A85 A86 A87 A88 0;
0 0 0 0 0 0 0 1 0];
Pcoup=poly(Amat);
Rcoup=roots(Pcoup);
%
B11= dxdb1m*db1mddele/m;
B12= dxdthetom*dthetomddelc/m;
B13= dxda1m*da1mddela/m;
B14= 0;
%
B21= dzdb1m*db1mddele/m;
B22= dzdthetom*dthetomddelc/m;
B23= 0;
B24= 0;
%
B31= dmdb1m*db1mddele/Iyy;
B32= dmdthetom*dthetomddelc/Iyy;
B33= dmda1m*da1mddela/Iyy;
B34= 0;
%
B41= 0;
B42= 0;
B43= 0;
B44= 0;
%
B51= dydb1m*db1mddele/m;
B52= dydthetom*dthetomddelc/m;
B53= dyda1m*da1mddela/m;
B54= (dydphin*dphinddelp+dyddelv*ddelvddelp+dydthetot*dthetotddelp)/m;
%
B61= Izz*drdb1m*db1mddele/Ic;
B62= (Izz*drdthetom*dthetomddelc+Ixz*dndthetom*dthetomddelc)/Ic;
B63= Izz*drda1m*da1mddela/Ic;
B                          6                                  4                                  =
(Izz*(drdphin*dphinddelp+drddelv*ddelvddelp+drdthetot*dthetotddelp)+...

Ixz*(dndphin*dphinddelp+dnddelv*ddelvddelp+dndthetot*dthetotddelp))/Ic;
%
B71= 0;
B72= 0;
B73= 0;
B74= 0;
%
B81= Ixz*drdb1m*db1mddele/Ic;
B82= (Ixz*drdthetom*dthetomddelc+Ixx*dndthetom*dthetomddelc)/Ic;
B83= Ixz*drda1m*da1mddela/Ic;
B                          8                                  4                                  =
(Ixz*(drdphin*dphinddelp+drddelv*ddelvddelp+drdthetot*dthetotddelp)+...

Ixx*(dndphin*dphinddelp+dnddelv*ddelvddelp+dndthetot*dthetotddelp))/Ic;
%
```

```
Glonaug=[B11 B12 B13 B14;
B21 B22 B23 B24;
B31 B32 B33 B34;
B41 B42 B43 B44];
Glataug=[B51 B52 B53 B54;
B61 B62 B63 B64;
B71 B72 B73 B74;
B81 B82 B83 B84;
0 0 0 0];
% coupled input matrix
Bmat=[B11 B12 B13 B14;B21 B22 B23 B24;
B31 B32 B33 B34;B41 B42 B43 B44;
B51 B52 B53 B54;B61 B62 B63 B64;
B71 B72 B73 B74;B81 B82 B83 B84;0 0 0 0];
%
xcouple=12/lockno*e/R/(1+e/3/R);
% designed damping
desdmdq=dmdqm+dmdqh+dmdqw;
desdrdp=drdpm+drdpt+drdpw+drdpv+drdpn;
desdndr=dndrm+dndrt+dndrv+dndrn;
% cooper harper pilot rating
prpitch=desdmdq/Iyy;
prroll=desdrdp/Ixx;
pryaw=desdndr/Izz;
% control power
cppitch=B31*Iyy;
cproll=B63*Ixx;
cpyaw=B84*Izz;
cpipitch=B31;
cpiroll=B63;
cpiyaw=B84;
%
theta0=theta7;
```

D. DCTPLOTS.M

```
% DCTPLOTS.M
%
% plots ct/sigma, ch/sigma, cq/sigma, a1s and b1s
%                    VERSES
% mu, thetao and lamp
% fits a polynomial to the curve, takes a derivative of the curves
% then evaluates the appropriate parameters for CRUISE.M
%
% d(ct/sigma), d(ch/sigma), d(cq/sigma), da1s and db1s
%                    VERSES
% dmu, dthetao and dlamp
% uses the pertubation points from CRANK.M about nominal
%
% polyfitting data to obtain plots for n=8
n=2;
ectsigmu=polyfit(vmu,vctsig,n);
ectsigtheto=polyfit(vthetao,vctsig,n);
ectsiglamp=polyfit(vlamp,vctsig,n);
%
echsigmu=polyfit(vmu,vchsig,n);
echsigtheto=polyfit(vthetao,vchsig,n);
echsiglamp=polyfit(vlamp,vchsig,n);
%
ecqsigmu=polyfit(vmu,vcqsig,n);
ecqsigtheto=polyfit(vthetao,vcqsig,n);
ecqsiglamp=polyfit(vlamp,vcqsig,n);
%
ea1smu=polyfit(vmu,va1s,n);
ea1stheto=polyfit(vthetao,va1s,n);
ea1slamp=polyfit(vlamp,va1s,n);
%
eb1smu=polyfit(vmu,vb1s,n);
eb1stheto=polyfit(vthetao,vb1s,n);
eb1slamp=polyfit(vlamp,vb1s,n);
%
%taking derivatives
%
edctsigdmu=deriv2(ectsigmu);
edctsigdtheto=deriv2(ectsigtheto);
edctsigdlamp=deriv2(ectsiglamp);
%
edchsigdmu=deriv2(echsigmu);
edchsigdtheto=deriv2(echsigtheto);
edchsigdlamp=deriv2(echsiglamp);
%
edcqsigdmu=deriv2(ecqsigmu);
edcqsigdtheto=deriv2(ecqsigtheto);
edcqsigdlamp=deriv2(ecqsiglamp);
%
eda1sdmu=deriv2(ea1smu);
eda1sdtheto=deriv2(ea1stheto);
eda1sdlamp=deriv2(ea1slamp);
%
edb1sdmu=deriv2(eb1smu);
edb1sdtheto=deriv2(eb1stheto);
edb1sdlamp=deriv2(eb1slamp);
%
% evaluating derivatives
```

```
%
dctsigdmu=polyval(edctsigdmu,vmu(1,2));
dctsigdtheto=polyval(edctsigdtheto,vthetao(1,2));
dctsigdlamp=polyval(edctsigdlamp,vlamp(1,2));
%
dchsigdmu=polyval(edchsigdmu,vmu(1,2));
dchsigdtheto=polyval(edchsigdtheto,vthetao(1,2));
dchsigdlamp=polyval(edchsigdlamp,vlamp(1,2));
%
dcqsigdmu=polyval(edcqsigdmu,vmu(1,2));
dcqsigdtheto=polyval(edcqsigdtheto,vthetao(1,2));
dcqsigdlamp=polyval(edcqsigdlamp,vlamp(1,2));
%
da1dmu=polyval(eda1sdmu,vmu(1,2));
da1dtheto=polyval(eda1sdtheto,vthetao(1,2));
da1dlamp=polyval(eda1sdlamp,vlamp(1,2));
%
db1dmu=polyval(edb1sdmu,vmu(1,2));
db1dtheto=polyval(edb1sdtheto,vthetao(1,2));
db1dlamp=polyval(edb1sdlamp,vlamp(1,2));
%
```

E. DCTPLOTT.M

```
% DCTPLOTT.M
%
% plots ct/sigmat
%       VERSES
% mut, thetaot and lampt
% fits a polynomial to the curve, takes a derivative of the curves
% then evaluates the appropriate parameters for CRUISE.M
%
% d(ct/sigma)
%           VERSES
% dmut, dthetaot and dlampt
% uses the pertubation points from JANRAD about nominal
%
% polyfitting data to obtain plots for n=2
n=2;
ectsigmut=polyfit(vmut,vctsigt,n);
ectsigthetot=polyfit(vthetaot,vctsigt,n);
ectsiglampt=polyfit(vlampt,vctsigt,n);
%
%taking derivatives
%
edctsigdmut=deriv2(ectsigmut);
edctsigdthetot=deriv2(ectsigthetot);
edctsigdlampt=deriv2(ectsiglampt);
%
% evaluating derivatives
%
dctsigdmut=polyval(edctsigdmu,vmu(1,2));
dctsigdthetot=polyval(edctsigdtheto,vthetao(1,2));
dctsigdlampt=polyval(edctsigdlamp,vlamp(1,2));
```

F. CMRGRP.M

```
% CMRGRP.M
% CALLED BY CRUISE.M
% Computes the basic MR derivatives in cruise flight
% Computes the stability derivatives of the main rotor in cruise flight.
% Uses data loaded in the workspace by JANRAD.M and STAB.M
%
% Compute the basic mainrotor derivatives.
%
dmudxdot=1/ohm/R;
dlampdxdot=1/ohm/R*(altpp-sigma/2/mu*(dctsigdmu-ctsig/mu));
dlampdzdot=1/(ohm*R*(1+dctsigdlamp*sigma/2/mu));
dbetdydot=1/V;
dchsigda=ctsig+a/8*lamp;
dchsigdb=dchsigda;
da1dq=-16/(lockno*ohm*(1-e/R)^2*(1-mu*mu/2))-...
12*e/R/(lockno*ohm*(1-e/R)^3*(1-mu^4/4));
da1dp=1/(ohm*(1-mu^2/2))-192*e/R/(lockno^2*ohm*(1-e/R)^5*(1-mu^4/4));
da1da=12*e/R*(1+mu^2/2)/(lockno*(1-e/R)^3*(1-mu^4/4));
da1db=-1*(1+3/2*mu^2)/(1-mu^2/2);
db1dq=-inv(ohm*(1+mu^2/2))+192*e/R/(lockno^2*ohm*(1-e/R)^5*(1-mu^4/4));
db1dp=-16/(lockno*ohm*(1-e/R)^2*(1+mu*mu/2))-...
12*e/R/(lockno*ohm*(1-e/R)^3*(1-mu^4/4));
db1da=1;
db1db=12*e/R*(1+3/2*mu^2)/(lockno*(1-e/R)^3*(1-mu^4/4));
dmda1sm=3/4*e/R*Ab*rho*R*(ohm*R)^2*a/lockno;
drdb1sm=dmda1sm;
%
% Compute the mainrotor stability derivatives.
%
dxdxdotm=-rho*Ab*(ohm*R)^2*((dchsigdmu+dchsigda*da1dmu+(a1s+im)*dctsigdm
u)...
*dmudxdot+((dchsigdlamp+dchsigda*da1dlamp+(a1s+im)*dctsigdlamp)*dlampdxd
ot));
dxdydotm=rho*Ab*(ohm*R)^2*ctsig*(A1-b1s)*dbetdydot;
dxdzdotm=-rho*Ab*(ohm*R)^2*(dchsigdlamp+ctsig*da1dlamp+(a1s+im)*dctsigdl
amp)...
*dlampdzdot;
dxdqm=-rho*Ab*(ohm*R)^2*dchsigda*da1dq-dxdxdotm*hm;
dxdpm=-rho*Ab*(ohm*R)^2*dchsigda*da1dp-dxdydotm*hm;
dxdthetom=-rho*Ab*(ohm*R)^2*(dchsigdtheto+dchsigda*da1dtheto+...
(a1s+im)*dctsigdtheto);
dxda1m=-rho*Ab*(ohm*R)^2*dchsigda*da1da;
dxdb1m=-rho*Ab*(ohm*R)^2*dchsigda*da1db;
dydxdotm=rho*Ab*(ohm*R)^2*(dcysigdb*db1dmu+b1s*dctsigdmu)*dmudxdot;
dydydotm=-rho*Ab*(ohm*R)^2*(chsig+ctsig*(B1+a1s))*dbetdydot;
dydzdotm=rho*Ab*(ohm*R)^2*db1dlamp*dcysigdb*dlampdzdot;
dydqm=rho*Ab*(ohm*R)^2*dcysigdb*db1dq+dydxdotm*hm;
dydpm=rho*Ab*(ohm*R)^2*dcysigdb*db1dp+dydydotm*hm;
dydthetom=rho*Ab*(ohm*R)^2*(dcysigdb*db1dtheto+b1s*dctsigdtheto);
dyda1m=rho*Ab*(ohm*R)^2*dcysigdb*db1da;
dydb1m=rho*Ab*(ohm*R)^2*dcysigdb*db1db;
dzdxdotm=-rho*Ab*(ohm*R)^2*(dctsigdmu*dmudxdot+dctsigdlamp*dlampdxdot);
dzdzdotm=-rho*Ab*(ohm*R)^2*dctsigdlamp*dlampdzdot;
dzdrm=2/ohm*rho*Ab*(ohm*R)^2*ctsig;
dzdthetom=-rho*Ab*(ohm*R)^2*dctsigdtheto;
dzdb1m=-rho*Ab*(ohm*R)^2*dctsigdlamp*inv(da1dlamp)*da1db;
drdxdotm=drdb1sm*db1dmu*dmudxdot+dydxdotm*hm+dzdxdotm*ym;
drdydotm=-drdb1sm*(B1+a1s)*dbetdydot+dydydotm*hm;
```

```
drdzdotm=drdb1sm*db1dlamp*dlampdzdot+dydzdotm*hm+dzdzdotm*ym;
drdqm=drdb1sm*db1dq+dydqm*hm;
drdpm=drdb1sm*db1dp+dydpm*hm;
drdthetom=drdb1sm*db1dtheto+dydthetom*hm+dzdthetom*ym;
drda1m=drdb1sm*db1da+dyda1m*hm;
drdb1m=drdb1sm*db1db+dydb1m*hm;
dmdxdotm=dmda1sm*(da1dmu*dmudxdot+da1dlamp*dlampdxdot)+dxdxdotm*hm+dzdxd
otm*lm;
dmdydotm=-dxdydotm*hm+dmda1sm*(A1-b1s)*dbetdydot;
dmdzdotm=dmda1sm*da1dlamp*dlampdzdot-dxdzdotm*hm+dzdzdotm*lm;
dmdqm=dmda1sm*da1dq-dxdqm*hm;
dmdpm=dmda1sm*da1dp+dxdpm*hm;
dmdthetom=dmda1sm*da1dtheto-dxdthetom*hm+dzdthetom*lm;
dmda1m=dmda1sm*da1da-dxda1m*hm;
dmdb1m=dmda1sm*da1db-dxdb1m*hm;
dndxdotm=rho*Ab*(ohm*R)^2*R*(dcqsigdmu*dmudxdot+dcqsigdlamp*dlampdxdot);
dndzdotm=rho*Ab*(ohm*R)^2*R*dcqsigdlamp*dlampdzdot;
dndrm=-2/ohm*rho*Ab*(ohm*R)^2*R*cqsig;
dndthetom=rho*Ab*(ohm*R)^2*R*dcqsigdtheto;
%
% return to CRUISE.M
```

G. CTRGRP.M

```
% CTRGRP.M
% CALLED BY CRUISE.M
% Computes the stability derivatives of the tail rotor in cruise flight.
% Computes the stability derivatives of the NOTAR in cruise flight.
% Uses data loaded in the workspace by JANRAD.M and STAB.M.
%
if notar==0
%
%    FIX THIS
%
% Compute the stability derivatives of the tail rotor in cruise flight.
%
%  dctsigdlampt=1.04                          %  table lookup, Prouty table
9.5
%  dctsigdmut=-0.07                          %  table lookup, Prouty chap 3
%  dctsigdthetot=.659;                        %  table 9.5
  dlampdydott=-1/((ohmt*Rt)*(1+dctsigdlampt*sigmat/(2*mut)));
  dydxdott=rho*Abt*(ohmt*Rt)^2*dctsigdmut*dmudxdot;
  dydydott=rho*Abt*(ohmt*Rt)^2*dctsigdlampt*dlampdydott;
  dydpt=dydydott*ht;
  dydrt=-dydydott*lt;
  dydthetot=rho*Abt*(ohmt*Rt)^2*dctsigdthetot;
  drdxdott=dydxdott*ht;
  drdydott=dydydott*ht;
  drdpt=dydpt*ht;
  drdrt=dydrt*ht;
  drdthetot=dydthetot*ht;
  dndxdott=-dydxdott*lt;
  dndydott=-dydydott*lt;
  dndpt=-dydpt*lt;
  dndrt=-dydrt*lt;
  dndthetot=-dydthetot*lt;
  %
  %  If tail rotor is to be used, zero out NOTAR derivatives:
  %
  dydxdotn=0;dydydotn=0;dydzdotn=0;dydpn=0;dydrn=0;drdxdotn=0;
  drdydotn=0;drdzdotn=0;drdpn=0;drdrn=0;dndxdotn=0;dndydotn=0;
  dndzdotn=0;dndpn=0;dndrn=0;
  dndphin=0;drdphin=0;dydphin=0;
  dnddelv=0;drddelv=0;dyddelv=0;

elseif notar==1

  %
  % Compute the stability derivatives of the NOTAR in cruise flight.
  %
  qvin=.5*rho*v1max^2;
  dydxdotn=0;
  dydcsoar=qvin*bn^2*max([(-V/125+1),0]);
  % downwash blown off circulation control section above 125 ft/sec (75
kts)
  dcsoardcloar=1;
  dcloardpslot=2/pi;
  dpslotdcmu=dian/(2*bn)*sqrt(cmun*dian/(2*hslot))*1.5;
  dcmudvimax=-64*hslot/(dian*v1max);
  dvimaxdzdot=1;
  dcsoardswirl=-1/pi;
  dswirldydot=1/v1;
```

```
  dydydotn=dydcsoar*dcsoardswirl*dswirldydot;

dydzdotn=dydcsoar*dcsoardcloar*dcloardpslot*dpslotdcmu*dcmudvimax*dvimax
dzdot;
  dydpn=dydydotn*hn;
  dydrn=-dydydotn*ln;
  drdxdotn=dydxdotn*hn;
  drdydotn=dydydotn*hn;
  drdzdotn=dydzdotn*ln;
  drdpn=dydpn*hn;
  drdrn=dydrn*hn;
  dndxdotn=-dydxdotn*ln;
  dndydotn=-dydydotn*ln;
  dndzdotn=-dydzdotn*ln;
  dndpn=-dydpn*ln;
  dndrn=-dydrn*ln;
  dydphin=Ytmaxn/(pi/2);
  drdphin=dydphin*htn;
  dndphin=-dydphin*ltn;
  %
  %  If NOTAR is to be used, zero out tail rotor derivatives:
  %
  dydxdott=0;dydydott=0;dydpt=0;dydrt=0;dydthetot=0;drdxdott=0;
  drdydott=0;drdpt=0;drdrt=0;drdthetot=0;dndxdott=0;dndydott=0;
  dndpt=0;dndrt=0;dndthetot=0;
  mut=0;lampt=0;aot=0;blst=0;alst=0;locknot=0;
  drdthetot=0;dndthetot=0;dydthetot=0;
else
disp(' ')
disp('                          ERROR IN CTRGRP.M')
disp('                NO TAIL ROTOR OR NOTAR INSTALLED')
disp('                          CHECK INPUT DATA')
disp(' ')
disp(' Press any key to continue...')
pause
end
%
% return to CRUISE.M
```

H. CBODYGRP.M

```
% CBODYGRP.M
% CALLED BY CRUISE.M
% Computes the stability derivatives of the fuselage, wing, verticle
% fin and horizontal stabalizer in cruise flight.
% Uses data loaded in the workspace by JANRAD.M STAB.M CRUISE.M
% CMRGRP.M CTRGRP.M
%
% THIS SUBROUTINE MUST FOLLOW CMRGRP.M AND CTRGRP.M
%
% Compute the stability derivatives of the fuselage
%
dgamdzdotf=-1/V;
dbetadydotf=1/V;
detamfdxdotf=vfv1/(4*q*pi*R^2)*(-dzdxdotm-2*T/V);
detamfdzdotf=-vfv1/(4*q*pi*R^2)*dzdzdotm;
dalpfdxdotf=-detamfdxdotf;
dalpfdzdotf=-(detamfdzdotf+dgamdzdotf);
%
%  the folowing are from the curves in appendix A of Prouty:
%  and can be modified if necessary
dfdalpf=-0.5;
dlqdalpf=19.2;
dsfqdbetaf=-55;
dmqdalpf=445;
dnqdbetaf=-205;
drqdbetaf=58;
%
dxdxdotf=-2/V*f*q;
dxdzdotf=-(-f*q-q*dfdalpf)*dalpfdzdotf;
dydydotf=1/V*(q*dsfqdbetaf-f*q);
dzdxdotf=0;
dzdzdotf=(-f*q-q*dlqdalpf)*dalpfdzdotf;
drdydotf=q*drqdbetaf*dbetadydotf;
dmdxdotf=q*dmqdalpf*dalpfdxdotf;
dmdzdotf=q*dmqdalpf*dalpfdzdotf;
dndydotf=q*dnqdbetaf*dbetadydotf;
%
% Compute the stability derivatives of the wing
%
if wing==1
  dgamdzdotw=-1/V;
  detamhdxdotw=vwv1/(4*q*pi*R^2)*(-dzdxdotm-2*T/V);
  detamhdzdotw=-vwv1/(4*q*pi*R^2)*dzdzdotm;;
  detafhdzdotw=detafdalpfw*(1/(4*q*pi*R^2)*dzdzdotm-dgamdzdotw);
  dalphdxdotw=-detamhdxdotw;
  dalphdzdotw=-(detamhdzdotw+detafhdzdotw+dgamdzdotw);
  dalphdzdbldotw=-detamhdzdotw*lw/V;
  %
  dxdxdotw=-2*cdow;

dxdzdotw=q*Aw*aw*((alpw-alplow)*(1-2*aw*(1+deliw)*Aw/pi/bw^2)+alpw-iw)*...
    dalphdzdotw;
  dxdzdbldotw=dxdzdotw*dalphdzdbldotw/dalphdzdotw;
  dzdxdotw=2/V*Zw-q*Aw*aw*(1+aw*(1+deliw)*Aw/pi/bw^2*(2*(alpw-alplow)*...
    (alpw-iw)+(alpw-alplow)^2)+cdow)*dalphdxdotw;
  dzdzdotw=-q*Aw*aw(1+aw*(1+deliw)*Aw/pi/bw^2*(2*(alpw-alplow)*...
    (alpw-iw)+(alpw*alplow)^2)+cdow)*dalphdzdotw;
  dzdzdbldotw=dzdzdotw*dalphdzdbldotw/dalphdzdotw;
```

```
  %
  dzdqw=dzdzdotw*lw;
  dzdqw=dzdzdotw*lw;
  dmdxdotw=-dxdxdotw*hw+dzdxdotw*lw;
  dmdzdotw=-dxdzdotw*hw+dzdzdotw*lw;
  dmdzdbldotw=-dxdzdbldotw*hw+dzdzdbldotw*lw;
  dmdqw=dzdqw*lh;
  %
  drdrw=q*Aw*bw*bw*aw*alpw/4/V/V/Ixx;
  drdpw=-q*Aw*bw*bw/4/V/V/Ixx*pi*(1+3*ctw/crw)/(6*(1+ctw/crw));
else
  %
  % Zero stability derivatives of the wing when no wing is installed
  %
  dxdxdotw=0;dxdzdotw=0;dxdzdbldotw=0;dzdxdotw=0;
  dzdzdotw=0;dzdzdbldotw=0;
  dzdqw=0;dzdqw=0;dmdxdotw=0;dmdzdotw=0;
  dmdzdbldotw=0;dmdqw=0;
  drdrw=0;drdpw=0;
end
%
% Compute the stability derivatives of the verticle fin
%
detafdbeta=.06; % assumed to be .06 because of little study of effect
dbetadydotv=1/V;
detatvdxdotv=-1/(4*qvq*q*Av)*(dydxdott-2*Tt/V);
dalpvdxdotv=detatvdxdotv;
detatvdydotv=dydydott/(4*qvq*q*At);
detafdydotv=detafdbeta*dbetadydotv;
dalpvdydotv=-(dbetadydotv+detatvdydotv+detafdydotv);
%
dxdxdotv=0;
dxdydotv=0;
dydxdotv=2/V*Yv+qvq*q*Av*av*dalpvdxdotv;
dydydotv=1/(1-deldv/Yv)*...
  (qvq*q*Av*av*dalpvdydotv+deldv*(-2/V+1/Tt*dydydott));
%
dydpv=dydydotv*hv;
dydrv=-dydydotv*lv;
drdxdotv=dydxdotv*hv;
drdydotv=dydydotv*hv;
drdpv=dydpv*hv;
drdrv=dydrv*hv;
dndxdotv=-dydxdotv*lv;
dndydotv=-dydydotv*lv;
dndpv=-dydpv*lv;
dndrv=-dydrv*lv;
%
dyddelv=q*qvq*Av*clvertmax/delvmax;
drddelv=dyddelv*hv;
dnddelv=-dyddelv*lv;
%
% Compute the stability derivatives of the horizontal stabalizer
%
dgamdzdoth=-1/V;
detamhdxdoth=vhv1/(4*q*pi*R^2)*(-dzdxdotm-2*T*cos(altpp)/V);
detamhdzdoth=-vhv1/(4*q*pi*R^2)*dzdzdotm;
detafhdzdoth=detafdalpfh*(1/(4*q*pi*R^2)*dzdzdotm-dgamdzdoth);
dalphdxdoth=-detamhdxdoth;
dalphdzdoth=-abs(detamhdzdoth+detafhdzdoth+dgamdzdoth);
```

```
dalphdzdbldoth=-detamhdzdoth*lh/V;
%
etamh=vhv1*v1/V;  %fix this
etafh=detafdalpfh*tho;
Xh=Lhoriz*sin(tho-(etamh+etafh+gamc))-Dhoriz*cos(tho-(etafh+etafh+gamc));
dxdxdoth=2/V*Xh+qhq*q*Ah*ah*((alph-alploh)*(1-2*ah*(1+delih)*Ah/pi/bh^2)
+alph-ih)*...
  dalphdxdoth;
dxdzdoth=qhq*q*Ah*ah*((alph-alploh)*(1-2*ah*(1+delih)*Ah/pi/bh^2)+alph-i
h)*...
  dalphdzdoth;
dxdzdbldoth=dxdzdoth*dalphdzdbldoth/dalphdzdoth;
dzdxdoth=2/V*Zh-qhq*q*Ah*ah*(1+ah*(1+delih)*Ah/pi/bh^2*(2*(alph-alploh)*
...
  (alph-ih)+(alph-alploh)^2)+cdoh)*dalphdxdoth;
dzdzdoth=-abs(-qhq*q*Ah*ah(1+ah*(1+delih)*Ah/pi/bh^2*(2*(alph-alploh)*...
  (alph-ih)+(alph*alploh)^2)+cdoh)*dalphdzdoth);
dzdzdbldoth=dzdzdoth*dalphdzdbldoth/dalphdzdoth;
%
dzdqh=dzdzdoth*lh;
dzdqh=dzdzdoth*lh;
dmdxdoth=-dxdxdoth*hh+dzdxdoth*lh;
dmdzdoth=-dxdzdoth*hh+dzdzdoth*lh;
dmdzdbldoth=-dxdzdbldoth*hh+dzdzdbldoth*lh;
dmdqh=dzdqh*lh;
%
% return to CRUISE.M
```

I. HOVER.M

```
% HOVER.M
% CALLED BY STABDER.M
% Computes the stability derivatives at a hover.
% calls the FOLLOWING subroutines to compute stability derivatives
%
% HMRGRP
% HTRGRP
% TRIM
%
% computation of stability derivatives
% the only derivatives important at hover are main and tail rotor
format compact
% evaluate dctsigdtheto dcqsigdtheto dctsigdthetot and dcqsigdthetot
%
vctsig=zeros(1,2);
vcqsig=zeros(1,2);
vthetao=zeros(1,2);
vctsigt=ones(1,2);
vcqsigt=ones(1,2);
vthetaot=ones(1,2);
eval(['load ',filename1])
theta1=-twist;
GW=GW*1.0005;   % .5% higher
disp(' ')
disp('                    *** FIRST OF TWO TRIM ITERATIONS ***')
disp(' ')
pause(3)
trim
%   *** Calculation of output parameters from trim ***
Qrotor=mean(DMpsi)*b;
solidity=b*cblade/(pi*R);
CQ=Qrotor/(Adisk*rho*Vtip^2*R);
CH=Hrotor/(Adisk*rho*Vtip^2);
CH_sig=CH/solidity;
vctsig(1,2)=CT/solidity;
vcqsig(1,2)=CQ/solidity;
thetasave=thetao;
vtheta(1,2)=thetasave-.7*theta1;
ohmt=ohmt;
sigma=solidity;
g=32.174;
A=pi*R*R;
if notar==0
  Abt=bt*Rt*cot;
  At=pi*Rt*Rt;
  mut=Vinf/ohmt/Rt;
  sigmat=Abt/At;
  lt=ltd-xcg;
  ht=htd-zcg;
  yt=ytd-ycg;
  locknot=rho*at*cot*Rt^4/Ibt;
  Tt=Qrotor/lt;
  ctsigt=Tt/(rho*At*(ohmt*Rt)^2);
  v1t=sqrt(Tt/2/rho/At);
  lampt=-v1t/ohmt/Rt;
  aot=2/3*locknot*ctsigt/at-3/2*g*Rt^2/(ohmt*Rt)^2;
  thetaot=3/2*(4/at*ctsigt+sqrt(CQ/2))-3/4*theta1t;
  a1st=0;
```

```
  blst=0;
  theta75t=thetaot+.75*thetalt;
  vlt=sqrt(Tt/2/rho/At);
  aoat=theta75t+atan(vlt/ohmt/Rt/.7);
  cdt=.006+(aoat*at)^2/pi/Rt/cot;
  phit=sqrt(ctsigt*sigmat/2);
  thet=4/at*ctsigt+sqrt(phit);
  cqsigt=sigmat/2*(at/2*(thet-phit)*phit+cdt/4);
end
vctsigt(1,2)=ctsigt;
vcqsigt(1,2)=cqsigt;
vthetaot(1,2)=thetaot;
%
save stabtemp vctsig vcqsig vthetao theta1 filename1 vctsigt vcqsigt
vthetaot g
clear
load stabtemp
eval(['load ',filename1])
load stabtemp
disp(' ')
disp('                    *** SECOND OF TWO TRIM ITERATIONS ***')
disp(' ')
pause(3)
trim
disp(' ')
disp('                        *** PERTURBATIONS COMPLETE ***')
disp(' ')
disp('                    *** EVALUATING STABILITY DERIVATIVES ***')
disp(' ')
Qrotor=mean(DMpsi)*b;
solidity=b*cblade/(pi*R);
CQ=Qrotor/(Adisk*rho*Vtip^2*R);
CH=Hrotor/(Adisk*rho*Vtip^2);
vctsig(1,1)=CT/solidity;
vcqsig(1,1)=CQ/solidity;
thetasave=thetao;
vthetao(1,1)=thetasave-.7*theta1;
dctsigdtheto=(vctsig(1,1)-vctsig(1,2))/(vthetao(1,1)-vthetao(1,2));
dcqsigdtheto=(vcqsig(1,1)-vcqsig(1,2))/(vthetao(1,1)-vthetao(1,2));
ohm=omega;
sigma=solidity;
A=pi*R*R;
if notar==0
  Abt=bt*Rt*cot;
  At=pi*Rt*Rt;
  mut=Vinf/ohmt/Rt;
  sigmat=Abt/At;
  lt=ltd-xcg;
  ht=htd-zcg;
  yt=ytd-ycg;
  locknot=rho*at*cot*Rt^4/Ibt;
  Tt=Qrotor/lt;
  ctsigt=Tt/(rho*At*(ohmt*Rt)^2);
  vlt=sqrt(Tt/2/rho/At);
  lampt=-vlt/ohmt/Rt;
  aot=2/3*locknot*ctsigt/at-3/2*g*Rt^2/(ohmt*Rt)^2;
  thetaot=3/2*(4/at*ctsigt+sqrt(CQ/2))-3/4*thetalt;
  alst=0;
  blst=0;
  theta75t=thetaot+.75*thetalt;
```

```
  v1t=sqrt(Tt/2/rho/At);
  aoat=theta75t+atan(v1t/ohmt/Rt/.7);
  cdt=.006+(aoat*at)^2/pi/Rt/cot;
  phit=sqrt(ctsigt*sigmat/2);
  thet=4/at*ctsigt+sqrt(phit);
  cqsigt=sigmat/2*(at/2*(thet-phit)*phit+cdt/4);
end
vctsigt(1,1)=ctsigt;
vcqsigt(1,1)=cqsigt;
vthetaot(1,1)=thetaot;
dctsigdthetot=(vctsigt(1,2)-vctsigt(1,1))/(vthetaot(1,2)-vthetaot(1,1));
dcqsigdthetot=(vcqsigt(1,2)-vcqsigt(1,1))/(vthetaot(1,2)-vthetaot(1,1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONFIGURATION CALCULATIONS
% converting data from crank
theta7=thetao;
thetao=theta7-.7*theta1;
ctsig=CT/solidity;
chsig=CH/solidity;
cqsig=CQ/solidity;
altpp=-alphaT;
A1=-theta1c;
B1=-theta1s;
ao=betao;
v1=mean(vi);
V=Vinf;
% Main rotor
g=32.2;
hm=hmd-zcg;
ym=ymd-ycg;
lm=lmd-xcg;
c=cblade;
Ab=c*R*4;
lamp=0;
mu=0;
ct=CT;
f=Afv;
thetat=4/a*ctsig+sqrt(sigma*ctsig/2);
lockno=rho*a*c*R^4/Ib;
theta75=thetao+.75*theta1;
num1=thetao*(8/3+32/45*mu^3/pi);
num2=theta1*(2+mu^4/12);
num3=lamp*(2-mu*mu/2);
den1=1+3/2*mu*mu-5*mu^4/24;
a1s=mu*(num1+num2+num3)/den1-B1;
b1s=a1s*12/lockno*e/R/(1+e/R/3);
m=GW/g;
Ic=Ixx*Izz;
if notar==1;
  % NOTAR
  v1max=1.2*v1;      %  downwash at the boom slots (NOTAR)
  delvmax=pi/4;
  ddelvddelp=delvmax/maxr;
  ltn=lttnd-xcg;
  ln=ltnd-xcg;
  yn=ytnd-xcg;
  hn=htnd-zcg;
  htn=hn;
  Tt=cqsig*sigma*rho*A*(ohm*R)^2*R/ltn;
  cmun=.55;    % typical value
```

```
  bn=.5*R;
  hslot=.028*dian/2;
  arn=bn/dian;
end
% trim conditions
tho=-T*(im+als)/GW;
pho=-(T*bls+Tt)/GW;
wo=0;
vo=0;
uo=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STABILITY CALCULATION
%
hmrgrp
htrgrp
%
% computation of A,B,C,D matrices
%
A11= (dxdxdotm)/m;
A12= (dxdzdotm)/m;
A13= (dxdqm)/m-wo;
A14= -g*cos(tho);
A15= (dxdydotm)/m;
A16= (dxdpm)/m;
A17= 0;
A18= (dxdrm)/m+vo;
%
A21= (dzdxdotm)/m;
A22= (dzdzdotm)/m;
A23= (dzdqm)/m+uo;
A24= -g*cos(pho)*sin(tho);
A25= (dzdydotm+dzdydott)/m;
A26= (dzdpm)/m-vo;
A27= -g*sin(pho)*cos(tho);
A28= (dzdrm)/m;
%
A31= (dmdxdotm)/Iyy;
A32= (dmdzdotm)/Iyy;
A33= (dmdqm)/Iyy;
A34= 0;
A35= (dmdydotm+dmdydott+dmdydotn)/Iyy;
A36= (dmdpm)/Iyy;
A37= 0;
A38= (dmdrm+dmdrt+dmdrn)/Iyy;
%
A41= 0;
A42= 0;
A43= cos(pho);
A44= 0;
A45= 0;
A46= 0;
A47= 0;
A48= -sin(pho);
%
A51= (dydxdotm)/m;
A52= (dydzdotm+dydzdotn)/m;
A53= (dydqm)/m;
A54= -g*sin(pho)*sin(tho);
A55= (dydydotm+dydydott+dydydotn)/m;
A56= (dydpm+dydpt+dydpn)/m+wo;
```

```
A57= g*cos(pho)*cos(tho);
A58= (dydrm+dydrt+dydrn)/m-uo;
%
A61= (Izz*(drdxdotm+drdxdotn)+Ixz*(dndxdotm+dndxdotn))/Ic;
A62= (Izz*(drdzdotm+drdzdotn)+Ixz*(dndzdotm+dndzdotn))/Ic;
A63= (Izz*(drdqm)+Ixz*(dndqm))/Ic;
A64= 0;
A                          6                          5                          =
(Izz*(drdydotm+drdydott+drdydotn)+Ixz*(dndydotm+dndydott+drdydotn))/Ic;
A66= (Izz*(drdpm+drdpt+drdpn)+Ixz*(dndpm+dndpt+dndpn))/Ic;
A67= 0;
A68= (Izz*(drdrm+drdrt+drdrn)+Ixz*(dndrm+dndrt+dndrn))/Ic;
%
A71= 0;
A72= 0;
A73= sin(pho)*tan(tho);
A74= 0;
A75= 0;
A76= 1;
A77= 0;
A78= cos(pho)*tan(tho);
%
A81= (Ixz*(drdxdotm+drdxdotn)+Ixx*(dndxdotm+dndxdotn))/Ic;
A82= (Ixz*(drdzdotm+drdzdotn)+Ixx*(dndzdotm+dndzdotn))/Ic;
A83= (Ixz*(drdqm)+Ixx*(dndqm))/Ic;
A84= 0;
A                          8                          5                          =
(Ixz*(drdydotm+drdydott+drdydotn)+Ixx*(dndydotm+dndydott+dndydotn))/Ic;
A86= (Ixz*(drdpm+drdpt+drdpn)+Ixx*(dndpm+dndpt+dndpn))/Ic;
A87=  0;
A88= (Ixz*(drdrm+drdrt+drdrn)+Ixx*(dndrm+dndrt+dndrn))/Ic;
%
% longitudinal plant augmented X=[u w q theta]
Flonaug=[A11 A12 A13 A14;A21 A22 A23 A24;A31 A32 A33 A34;
A41 A42 A43 A44];
Plonaug=poly(Flonaug);
Rlonaug=roots(Plonaug);
%
% Lateral plant augmented with X=[v p phi r psi]'
Flataug=[A55 A56 A57 A58 0;A65 A66 A67 A68 0;
A75 A76 A77 A78 0;A85 A86 A87 A88 0;0 0 0 1 0];
Plataug=poly(Flataug);
Rlataug=roots(Plataug);
% coupled plant
Amat=[A11 A12 A13 A14 A15 A16 A17 A18 0;
A21 A22 A23 A24 A25 A26 A27 A28 0;
A31 A32 A33 A34 A35 A36 A37 A38 0;
A41 A42 A43 A44 A45 A46 A47 A48 0;
A51 A52 A53 A54 A55 A56 A57 A58 0;
A61 A62 A63 A64 A65 A66 A67 A68 0;
A71 A72 A73 A74 A75 A76 A77 A78 0;
A81 A82 A83 A84 A85 A86 A87 A88 0;
0 0 0 0 0 0 0 1 0];
Pcoup=poly(Amat);
Rcoup=roots(Pcoup);
%
B11= dxdb1m*db1mddele/m;
B12= dxdthetom*dthetomddelc/m;
B13= dxda1m*da1mddela/m;
B14= 0;
```

```
%
B21= dzdb1m*db1mddele/m;
B22= dzdthetom*dthetomddelc/m;
B23= 0;
B24= 0;
%
B31= dmdb1m*db1mddele/Iyy;
B32= dmdthetom*dthetomddelc/Iyy;
B33= dmda1m*da1mddela/Iyy;
B34= (dmdphin*dphinddelp+dmdthetot*dthetotddelp)/Iyy;
%
B41= 0;
B42= 0;
B43= 0;
B44= 0;
%
B51= dydb1m*db1mddele/m;
B52= dydthetom*dthetomddelc/m;
B53= dyda1m*da1mddela/m;
B54= (dydphin*dphinddelp+dydthetot*dthetotddelp)/m;
%
B61= Izz*drdb1m*db1mddele/Ic;
B62= (Izz*drdthetom*dthetomddelc+Ixz*dndthetom*dthetomddelc)/Ic;
B63= Izz*drda1m*da1mddela/Ic;
B64= (Izz*(drdphin*dphinddelp+drdthetot*dthetotddelp)+...
Ixz*(dndphin*dphinddelp+dndthetot*dthetotddelp))/Ic;
%
B71= 0;
B72= 0;
B73= 0;
B74= 0;
%
B81= Ixz*drdb1m*db1mddele/Ic;
B82= (Ixz*drdthetom*dthetomddelc+Ixx*dndthetom*dthetomddelc)/Ic;
B83= Ixz*drda1m*da1mddela/Ic;
B84=  (Ixz*(drdphin*dphinddelp+drdthetot*dthetotddelp)+...
Ixx*(dndphin*dphinddelp+dndthetot*dthetotddelp))/Ic;
%
Glonaug=[B11 B12 B13 B14;
B21 B22 B23 B24;
B31 B32 B33 B34;
B41 B42 B43 B44];
%
Glataug=[B51 B52 B53 B54;
B61 B62 B63 B64;
B71 B72 B73 B74;
B81 B82 B83 B84;
0 0 0 0];
% coupled input matrix
Bmat=[B11 B12 B13 B14;B21 B22 B23 B24;
B31 B32 B33 B34;B41 B42 B43 B44;
B51 B52 B53 B54;B61 B62 B63 B64;
B71 B72 B73 B74;B81 B82 B83 B84;0 0 0 0];
%
% cross coupling
xcouple=12/lockno*e/R/(1+e/3/R);
% designed damping
desdmdq=dmdqm;
desdrdp=drdpm+drdpt+drdpn;
desdndr=dndrm+dndrt+dndrn;
```

```
% now cooper harper pilot rating
prpitch=desdmdq/Iyy;
prroll=(drdpm+drdpt+drdpn)/Ixx;
pryaw=desdndr/Izz;
% control power
cppitch=B31*Iyy;
cproll=B63*Ixx;
cpyaw=B84*Izz;
cpipitch=B31;
cpiroll=B63;
cpiyaw=B84;
%
thetao=theta7;
```

J. HMRGRP.M

```
% HMRGRP.M
% CALLED BY HOVER.M
% Computes the basic main rotor derivatives at a hover
% Computes the stability derivatives of the main rotor at a hover
% Uses data loaded in the workspace by JANRAD.M and STAB.M
%
% Compute the basic mainrotor derivatives at a hover
%
dmudxdot=1/ohm/R;
dlampdzdot=dmudxdot;
da1dmu=8/3*thetao+2*theta1-2*v1/ohm/R;
db1dmu=4/3*ao;
dctsigdlamp=inv(8/a+(sqrt(sigma/2)/(sqrt(ctsig))));
dcqsigdlamp=-a/4*(theta75-2*v1/ohm/R);
da1dq=-16/(lockno*ohm*(1-e/R)^2)-12*e/R/(lockno*ohm*(1-e/R)^3);
db1dp=da1dq;
da1dp=1/ohm*(1-(192*e/R/(lockno^2*(1-e/R)^5)));
db1dq=-da1dp;
da1da=12*e/R/(lockno*(1-e/R)^3);
db1db=da1da;
da1db=-1/(1+((144*(e/R)^2)/(lockno^2*(1-e/R)^6)));
db1da=-da1db;
dchsigda=3/2*ctsig*(1-a/18*theta75/ctsig);
dcysigdb=dchsigda;
dmda1s=3/4*e/R*Ab*rho*R*(ohm*R)^2*a/lockno;
drdb1s=dmda1s;
%
% Compute the mainrotor stability derivatives at a hover
%
dxdxdotm=-rho*Ab*(ohm*R)^2*dchsigda*da1dmu*dmudxdot;
dxdydotm=-rho*Ab*(ohm*R)^2*dcysigdb*db1dmu*dmudxdot;
dxdzdotm= 0;
dxdqm=-rho*Ab*(ohm*R)^2*dchsigda*da1dq;
dxdpm=-rho*Ab*(ohm*R)^2*dchsigda*da1dp ;
dxdrm=0;
dxdthetom=-rho*Ab*(ohm*R)^2*(a1s+im)*dctsigdtheto;
dxda1m=-rho*Ab*(ohm*R)^2*dchsigda*da1da;
dxdb1m=-rho*Ab*(ohm*R)^2*dchsigda*da1db;
dydxdotm=rho*Ab*(ohm*R)^2*dcysigdb*db1dmu*dmudxdot;
dydydotm=dxdxdotm;
dydzdotm=0;
dydqm=dxdpm;
dydrm=0;
dydpm=-dxdqm;
dydthetom=rho*Ab*(ohm*R)^2*(2*b1s)*dctsigdtheto;
dyda1m=dxdb1m;
dydb1m=-dxda1m;
dzdxdotm=0;
dzdydotm=0;
dzdzdotm=-rho*Ab*(ohm*R)^2*dctsigdlamp*dlampdzdot;
dzdpm=0;
dzdqm=0;
dzdrm=0;
dzdthetom=-rho*Ab*(ohm*R)^2*dctsigdtheto;
dzdb1m=0;
drdxdotm=drdb1s*db1dmu*dmudxdot+dydxdotm*hm+dzdxdotm*hm;
drdydotm=-dmda1s*da1dmu*dmudxdot+dxdxdotm*hm;
drdzdotm=dzdzdotm*ym;
```

```
drdqm=drdb1s*db1dq+dydqm*hm;
drdrm=0;
drdpm=drdb1s*db1dp+dydpm*hm;
drdthetom=dydthetom*hm+dzdthetom*ym;
drda1m=drdb1s*db1da+dyda1m*hm;
drdb1m=drdb1s*db1db+dydb1m*hm;
dmdxdotm=dmda1s*da1dmu*dmudxdot-dxdxdotm*hm;
dmdydotm=drdxdotm;
dmdzdotm=dzdzdotm*lm;
dmdqm=dmda1s*da1dq-dxdqm*hm;
dmdrm=0;
dmdpm=dmda1s*da1dp-dxdpm*hm;
dmdthetom=-dxdthetom*hm+dzdthetom*lm;
dmda1m=dmda1s*da1da-dxda1m*hm;
dmdb1m=dmda1s*da1db-dxdb1m*hm;
dndxdotm=0;
dndzdotm=rho*Ab*(ohm*R)^2*R*dcqsigdlamp*dlampdzdot;
dndydotm=0;
dndqm=0;
dndpm=0;
dndrm=-2*rho*Ab*(ohm*R)^2*R*cqsig;
dndthetom=rho*Ab*(ohm*R)^2*R*dcqsigdtheto;
%
% return to HOVER.M
```

K. HTRGRP.M

```
% HTRGRP.M
% CALLED BY HOVER.M
% Computes the basic tail rotor or NOTAR derivatives at a hover
% Computes the stability derivatives of the tail rotor or NOTAR at a hover
% OR
% Computes the stability derivatives of the NOTAR at a hover
%
% Uses data loaded in the workspace by JANRAD.M and STAB.M
%
if notar==0
  %
  % Compute the basic tail rotor derivatives at a hover.
  %
  dmudxdott=1/ohmt/Rt;
  dlampdydott=-dmudxdott;
  da1dmut=8/3*thetao+2*theta1-2*v1/ohmt/Rt;
  db1dmut=4/3*aot;
  dctsigdlampt=inv(8/at+(sqrt(sigmat/2)/(sqrt(ctsig))));
  dcqsigdlampt=-at/4*(theta75t-2*v1t/ohmt/Rt);
  da1dqt=0;
  db1dpt=0;
  da1dpt=0;
  db1dqt=0;
  da1dat=0;
  db1dbt=0;
  da1dbt=0;
  db1dat=0;
  dchsigdat=3/2*ctsig*(1-at/18*theta75t/ctsig);
  dcysigdbt=dchsigdat;
  dmda1st=0;
  drdb1st=0;
  %
  % Compute the tail rotor stabitity derivatives at a hover
  %
  dydydott=rho*Abt*(ohmt*Rt)^2*dctsigdlampt*dlampdydott;
  dydpt=dydydott*ht;
  dydrt=-dydydott*lt;
  dydthetot=rho*Abt*(ohmt*Rt)^2*dctsigdthetot;
  drdydott=dydydott*ht;
  drdrt=dydrt*ht;
  drdpt=dydpt*ht;
  drdthetot=dydthetot*ht;
  dmdydott=rho*Abt*(ohmt*Rt)^2*Rt*dcqsigdlampt*dlampdydott;
  dmdrt=-dmdydott*lt;
  dmdthetot=rho*Abt*(ohmt*Rt)^2*Rt*dcqsigdthetot;
  dndydott=-dydydott*lt;
  dndpt=-dydpt*lt;
  dndrt=-dydrt*lt;
  dndthetot=-dydthetot*lt;
  %
  % If tail rotor is used, zero out NOTAR derivatives:
  %
  dydxdotn=0;dydydotn=0;dydzdotn=0;dydpn=0;dydrn=0;
  dzdydott=0;
  drdxdotn=0;drdydotn=0;drdzdotn=0;drdpn=0;drdrn=0;
  dmdydotn=0;dmdrn=0;
  dndxdotn=0;dndydotn=0;dndzdotn=0;dndpn=0;dndrn=0;
  dmdphin=0;drdphin=0;dndphin=0;dydphin=0;
```

```
elseif notar==1

  % Compute the NOTAR derivatives at a hover.
  %
  qvin=.5*rho*v1max^2;
  dydxdotn=0;
  dydcsoar=qvin*bn^2;
  dcsoardcloar=1;
  dcloardpslot=2/pi;
  dpslotdcmu=dian/(2*bn)*sqrt(cmun*dian/(2*hslot))*1.5;
  dcmudvimax=-64*hslot/(dian*v1max);
  dvimaxdzdot=1;
  dcsoardswirl=-1/pi;
  dswirldydot=1/v1;
  dydydotn=dydcsoar*dcsoardswirl*dswirldydot;

dydzdotn=dydcsoar*dcsoardcloar*dcloardpslot*dpslotdcmu*dcmudvimax*dvimax
dzdot;
  dydpn=dydydotn*hn;
  dydrn=-dydydotn*ln;
  drdxdotn=dydxdotn*hn;
  drdydotn=dydydotn*hn;
  drdzdotn=dydzdotn*ln;
  drdpn=dydpn*hn;
  drdrn=dydrn*hn;
  dmdydotn=0;
  dmdrn=-dmdydotn*ln;
  dndxdotn=-dydxdotn*ln;
  dndydotn=-dydydotn*ln;
  dndzdotn=-dydzdotn*ln;
  dndpn=-dydpn*ln;
  dndrn=-dydrn*ln;
  dydphin=Ytmaxn/(pi/2);
  drdphin=dydphin*htn;
  dmdphin=0;
  dndphin=-dydphin*ltn;
  %
  %  If NOTAR is to be used, zero out tail rotor derivatives:
  %
  dydydott=0;dydpt=0;dydrt=0;dydthetot=0;drdydott=0;
  dzdydott=0;
  drdrt=0;drdpt=0;drdthetot=0;
  dmdydott=0;dmdrt=0;dmdthetot=0;
  dndydott=0;dndpt=0;dndrt=0;dndthetot=0;
  lampt=0;mut=0;aot=0;b1st=0;a1st=0;locknot=0;
  dmdthetot=0;dndthetot=0;drdthetot=0;dydthetot=0;
else
disp(' ')
disp('                          ERROR IN HTRGRP.M')
disp('              NO TAIL ROTOR OR NOTAR INSTALLED')
disp('                          CHECK INPUT DATA')
disp(' ')
disp(' Press any key to continue...')
pause
end
%
% return to HOVER.M
```

L. STABOUT.M

```
%  STABOUT.M
%
%  calls
%  CMDBWPLH
%  CMDBWPLC
%
%  *** Stability and Control Output Subroutine ***
pack
format compact
clc
disp(' ')
disp(' ')
disp('Do you want the results displayed on screen?')
disp(' ')
disp('  NOTE: if you want a hard copy of the plots, you must')
disp('  select (1) and view them on the screen first.')
flag=1;
answer=input('1. yes   2. no   >> ');
while flag>0
  if answer == 2,
    flag=0;
  elseif answer == 1,
%
% output to screen
%
  clc
  disp(' ')
  disp('         *** STABILITY AND CONTROL PROGRAM ***')
  disp('               *** SCREEN VIEW MENU ***')
  disp(' ')
  disp('What do you want to see?')
  disp(' ')
  disp('1. Input data.')
  disp('2. Calculated data.')
  disp('3. State Matrices.')
  disp('4. Eigenvalues of the plants and plots of the roots.')
  disp('5. Key control parameters.')
  disp('6. Open loop transfer plots.')
  disp(' ')
  disp('0. Exit screen view.')
  disp(' ')
    choice=input('Enter a number; ');
    if choice==1,
clc
disp(' ')
disp('              *** INPUT DATA (screen 1 of 8) ***')
eval(['disp('''                          ',filename1,''')'])
disp(' ')
disp('                       Flight Conditions')
disp(' ')
fprintf('             Forward velocity = %6.0f kts\n',Vinf/1.69)
fprintf('                  Temperature = %6.0f degs F\n',temp)
fprintf('            Pressure altitude = %6.0f ft\n',PA)
fprintf('             Auxiliary thrust = %6.0f lbs\n',Taux)
disp(' ')
disp('                          Fuselage')
disp(' ')
fprintf('                Gross weight = %6.0f lbs\n',GW)
```

```
fprintf('   Equivalent flat plate area = %6.1f ft^2\n',Afh)
fprintf('      Vertical projected area = %6.1f ft^2\n',Afv)
fprintf('    CG height above waterline = %6.1f ft\n',zcg)
fprintf('          CG fuselage station = %6.1f ft\n',xcg)
fprintf('CG position right of buttline = %6.1f ft\n',ycg)
fprintf('                          Ixx = %6.1f slug ft^2\n',Ixx)
fprintf('                          Iyy = %6.1f slug ft^2\n',Iyy)
fprintf('                          Izz = %6.1f slug ft^2\n',Izz)
fprintf('                          Ixz = %6.1f slug ft^2\n',Ixz)
fprintf('               Downwash ratio = %6.2f \n',vfv1)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('         *** INPUT DATA CONTINUED (screen 2 of 8) ***')
eval(['disp('' ',filename1,''')'])
disp(' ')
disp('                        Main Rotor')
disp(' ')
fprintf('             Number of blades = %6.0f \n',b)
fprintf('                 Rotor radius = %6.1f ft\n',R)
fprintf('                  Blade chord = %6.1f ft\n',cblade)
fprintf('                  Blade twist = %6.2f degs\n',twist*57.3)
if airfoil==1,
   disp('                Blade airfoil =  HH-02')
else
   disp('                Blade airfoil =  VR-12')
end
fprintf('       Blade lift curve slope = %6.2f \n',a)
fprintf('                 Blade weight = %6.1f lbs\n',wblade)
fprintf('          Rotational velocity = %6.2f rads/sec\n',omega)
fprintf('            Blade grip length = %6.1f ft\n',grip)
fprintf('                 Hinge offset = %6.1f ft\n',e)
fprintf('   Flapping moment of inertia = %6.1f slug ft^2\n',Ib)
fprintf('   Hub height above waterline = %6.1f ft\n',hmd)
fprintf('         Hub fuselage station = %6.1f ft\n',lmd)
fprintf('  Hub position rt of buttline = %6.1f ft\n',ymd)
fprintf('               Mast incidence = %6.2f deg\n',im*57.3)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('         *** INPUT DATA CONTINUED (screen 3 of 8) ***')
eval(['disp('' ',filename1,''')'])
disp(' ')
disp('          Tail rotor (zeros if using NOTAR)')
disp(' ')
fprintf('             Number of blades = %6.1f \n',bt)
fprintf('                  Blade chord = %6.1f ft\n',cot)
fprintf('                 Blade radius = %6.1f ft\n',Rt)
fprintf('                Lift curve slope = %6.2f \n',at)
fprintf('           Rotational velocity = %6.2f rad/sec\n',ohmt)
fprintf('   Flapping moment of inertia = %6.1f slug ft^2\n',Ibt)
fprintf('                Delta-3 angle = %6.2f deg\n',delta3*57.3)
fprintf('                  Blade twist = %6.2f deg\n',theta1t*57.3)
fprintf('   Hub height above waterline = %6.1f ft\n',htd)
fprintf('         Hub fuselage station = %6.1f ft\n',ltd)
```

```
fprintf('  Hub position rt of buttline = %6.1f ft\n',ytd)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('          *** INPUT DATA CONTINUED (screen 4 of 8) ***')
eval(['disp('''                          ',filename1,''')'])
disp(' ')
disp('              NOTAR (zeros if using tail rotor)')
disp(' ')
fprintf('       Height above waterline = %6.1f ft^2\n',htnd)
fprintf('             Fuselage station = %6.1f ft^2\n',ltnd)
fprintf('   Position right of buttline = %6.1f ft^2\n',ytnd)
fprintf('           NOTAR boom diameter = %6.1f ft^2\n',dian)
fprintf('           Swirl angle at boom = %6.2f deg\n',swirl*57.3)
fprintf('       Maximum thruster force = %6.1f lbs\n',Ytmaxn)
fprintf('      Thrust fuselage station = %6.1f ft^2\n',lttnd)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('          *** INPUT DATA CONTINUED (screen 5 of 8) ***')
eval(['disp('''                          ',filename1,''')'])
disp(' ')
disp('                              Wing')
disp(' ')
fprintf('                             Area = %6.1f ft^2\n',Swing)
fprintf('                             Span = %6.1f ft\n',bwing)
fprintf('                               CL = %6.2f \n',CLwing)
fprintf('                              CDo = %6.4f \n',CDowing)
fprintf('                         Tip cord = %6.1f ft\n',ctw)
fprintf('                        Root cord = %6.1f ft\n',crw)
fprintf('       Wing efficiency factor = %6.2f \n',ewing)
fprintf('              Zero lift angle = %6.2f deg\n',alplow*57.3)
fprintf('            Angle of incidence = %6.2f deg\n',iw*57.3)
fprintf('              Lift curve slope = %6.2f \n',aw)
fprintf('       Height above waterline = %6.1f ft\n',hwd)
fprintf('             Fuselage station = %6.1f ft\n',lwd)
fprintf('   Position right of buttline = %6.1f ft\n',ywd)
fprintf('         Rotor downwash ratio = %6.2f \n',vwv1)
fprintf('      Fuselage downwash ratio = %6.2f \n',detafdalpfw)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('          *** INPUT DATA CONTINUED (screen 6 of 8) ***')
eval(['disp('''                          ',filename1,''')'])
disp(' ')
disp('                         Horizontal tail')
disp(' ')
fprintf('                             Area = %6.1f ft^2\n',Shoriz)
fprintf('                             Span = %6.1f ft\n',bhoriz)
fprintf('                               CL = %6.2f \n',CLhoriz)
fprintf('                              CDo = %6.4f \n',CDohoriz)
fprintf('                 Zero lift angle = %6.2f deg\n',alploh*57.3)
fprintf('               Angle of incidence = %6.2f deg\n',iw*57.3)
fprintf('                 Lift curve slope = %6.1f \n',ah)
```

```
fprintf('       Height above waterline = %6.1f ft\n',hhd)
fprintf('             Fuselage station = %6.1f ft\n',lhd)
fprintf('   Position right of buttline = %6.1f ft\n',yhd)
fprintf('       Dynamic pressure ratio = %6.2f \n',qhq)
fprintf('         Rotor downwash ratio = %6.2f \n',vhv1)
fprintf('      Fuselage downwash ratio = %6.2f \n',detafdalpfh)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('         *** INPUT DATA CONTINUED (screen 7 of 8) ***')
eval(['disp('''                          ',filename1,''')'])
disp(' ')
disp('                       Vertical tail')
disp(' ')
fprintf('                            Area = %6.1f ft^2\n',Svert)
fprintf('                            Span = %6.1f ft\n',bvert)
fprintf('                              CL = %6.2f \n',CLvert)
fprintf('                             CDo = %6.4f \n',CDovert)
fprintf('       Height above waterline = %6.1f ft\n',hvd)
fprintf('             Fuselage station = %6.1f ft\n',lvd)
fprintf('   Position right of buttline = %6.1f ft\n',yvd)
fprintf('              Zero lift angle = %6.2f deg\n',alplov*57.3)
fprintf('                   Maximum Cl = %6.1f \n',clvertmax)
fprintf('       Dynamic pressure ratio = %6.2f \n',qvq)
fprintf('             Lift curve slope = %6.2f \n',av)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('         *** INPUT DATA CONTINUED (screen 8 of 8) ***')
eval(['disp('''                          ',filename1,''')'])
disp(' ')
disp('                          Rigging')
disp(' ')
fprintf('  Long cyclic pitch/inch defl = %6.2f deg/in\n',db1mddele*57.3)
fprintf('   Lat cyclic pitch/inch defl = %6.2f deg/in\n',da1mddela*57.3)
fprintf('            Collective   pitch/inch   defl    =    %6.2f
deg/in\n',dthetomddelc*57.3)
fprintf('    Tail    rotor    pitch    change/defl    =    %6.2f
deg/unit\n',dthetotddelp*57.3)
   disp('    Max deflection of control')
fprintf('       from neutral for NOTAR = %6.2f units\n',dphinddelp*57.3)
   disp('  Displacement of anti-torque')
fprintf('    control until full rudder = %6.2f units\n',maxr)
disp(' ')
disp('press any key to continue...')
pause
    elseif choice==2, % calculated data
clc
disp(' ')
disp('         *** CALCULATED DATA (screen 1 of 2) ***')
eval(['disp('''                  ',filename1,''')'])
disp(' ')
disp('                         Main Rotor')
disp(' ')
fprintf('                          thrust = %6.1f lbs\n',T)
fprintf('                          torque = %6.1f ft-lbs\n',Qrotor)
```

```
fprintf('                advance ratio = %6.1f \n',mu)
fprintf('     inflow parameter wrt TPP = %6.3f \n',lamp)
fprintf('                Tip path angle = %6.1f degs\n',altpp*57.3)
fprintf('           Rotor coning angle = %6.1f degs\n',ao*57.3)
fprintf('      1st lat cyclic term-A1 = %6.1f degs\n',A1*57.3)
fprintf('      1st long cyclic term-B1 = %6.1f degs\n',B1*57.3)
fprintf('             lateral flapping = %6.2f degs\n',b1s*57.3)
fprintf('       longitudinal flapping = %6.2f degs\n',a1s*57.3)
fprintf('                  Lock number = %6.1f \n',lockno)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('          *** CALCULATED DATA (screen 2 of 2)***')
eval(['disp('' ',filename1,''')'])
disp(' ')
disp('                    Tail Rotor (zero if NOTAR)')
disp(' ')
fprintf('            tail rotor thrust = %6.1f lbs\n',Tt)
fprintf('                advance ratio = %6.1f \n',mut)
fprintf('             inflow parameter = %6.3f \n',lampt)
fprintf('           Rotor coning angle = %6.1f degs\n',aot*57.3)
fprintf('             lateral flapping = %6.2f degs\n',b1st*57.3)
fprintf('       longitudinal flapping = %6.2f degs\n',a1st*57.3)
fprintf('                  Lock number = %6.1f \n',locknot)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause
    elseif choice==3, % state matrices
clc
disp(' ')
disp('Longitudinal uncoupled plant (A or F depending on notation)')
disp('States are [u w q theta]')
disp(' ')
disp(Flonaug)
disp(' ')
disp(' ')
disp('Longitudinal uncoupled input matrix (B or G depending on notation)')
disp('Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]')
disp(' ')
disp(Glonaug)
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('Lateral/directional uncoupled plant (A or F depending on notation)')
disp('States are [v p phi r psi]')
disp(' ')
disp(Flataug)
disp(' ')
disp(' ')
disp('Lateral/directional uncoupled input matrix (B or G depending')
disp('on notation)')
disp('Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]')
disp(Glataug)
```

```
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('Coupled plant (A or F depending on notation)')
disp('States are [u w q theta v p phi r psi]')
disp(' ')
disp(Amat)
disp('press any key to continue...')
pause
clc
disp(' ')
disp('Coupled input matrix (B or G depending on notation)')
disp('Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]')
disp(' ')
disp(Bmat)
disp(' ')
disp('press any key to continue...')
pause
clc
    elseif choice==4, % eigenvalues and root loci
clc
disp(' ')
disp('After you view the root loci plot, a meta file is made.')
disp('When you are done a screen will tell you the file names')
disp('of the meta files.  To get a hard copy of the plots, you')
disp('must graphics post process (GPP) the files for your')
disp('particular printer set-up then, print.')
disp(' ')
disp('NOTE: If ALL roots are real, MATLAB will NOT plot them')
disp('      in the Argand plane, but will plot the root against')
disp('      its position in the vector (e.g. the first root would')
disp('      be ploted as (1,root))')
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
disp('            *** EIGENVALUES ***')
eval(['disp('''                ',filename1,''')'])
disp(' ')
disp('Uncoupled')
disp(' ')
disp('Longitudinal plant')
disp(' ')
disp('        Root              wn            damping')
[wn,z]=damp(Rlonaug);
disp([Rlonaug,wn,z])
disp(' ')
disp('Lateral/Directional plant')
[wn,z]=damp(Rlataug);
disp(' ')
disp('        Root              wn            damping')
disp([Rlataug,wn,z])
disp(' ')
disp('press any key to continue...')
pause
plot(Rlonaug,'*'),grid,title('Roots of Longitudinal Plant')
```

```
pause
!del rootlon.*
meta rootlon
plot(Rlataug,'*'),grid,title('Roots of Lateral/Directional Plant')
pause
!del rootlat.*
meta rootlat
clc
disp('            *** EIGENVALUES ***')
eval(['disp('' ',filename1,''')'])
disp(' ')
disp('Coupled Plant')
disp(' ')
[wn,z]=damp(Rcoup);
disp('        Root              wn            damping')
disp([Rcoup wn z])
disp(' ')
disp('press any key to continue...')
pause
plot(Rcoup,'*'),grid,title('Roots of Coupled Plant')
pause
!del rootcoup.*
meta rootcoup
clc
disp(' ')
disp('Plots are saved under the following filenames:')
disp(' ')
disp('Longitudinal roots - rootlon.met')
disp('Lateral/Directional roots - rootlat.met')
disp('Coupled roots - rootcoup.met')
disp(' ')
disp('press any key to continue...')
pause
    elseif choice==5, % key control parameters
clc
disp(' ')
   disp('      *** KEY CONTROL PARAMETERS (screen 1 of 2) ***')
eval(['disp('' ',filename1,''')'])
disp(' ')
fprintf('   cross coupling = %6.2f \n',xcouple)
   disp(' ')
   disp('                        Designed damping')
fprintf('             pitch = %6.1f ft-lbs/(rad/sec)\n',desdmdq)
fprintf('              roll = %6.1f ft-lbs/(rad/sec)\n',desdrdp)
fprintf('               yaw = %6.1f ft-lbs/(rad/sec)\n',desdndr)
   disp(' ')
   disp('                          Control Power')
fprintf('             pitch = %6.1f ft-lbs/in\n',cppitch)
fprintf('              roll = %6.1f ft-lbs/in\n',cproll)
fprintf('               yaw = %6.1f ft-lbs/in\n',cpyaw)
   disp(' ')
disp(' ')
disp('press any key to continue...')
pause
clc
disp(' ')
   disp('      *** KEY CONTROL PARAMETERS (screen 2 of 2) ***')
eval(['disp('' ',filename1,''')'])
disp(' ')
   disp('               Cooper Harper Pilot Ratings')
```

```
   disp('                damping/moment of inertia')
fprintf('pitch    (dM/dq)/Iyy    =    %6.2f    [ft-lbs/(rad/sec)]/(slug
ft^2)\n',prpitch)
fprintf('    roll    (dR/dp)/Ixx    =    %6.2f    [ft-lbs/(rad/sec)]/(slug
ft^2)\n',prroll)
fprintf('      yaw    (dN/dr)/Izz    =    %6.2f    [ft-lbs/(rad/sec)]/(slug
ft^2)\n',pryaw)
   disp(' ')
   disp('              control power/moment of inertia')
fprintf('pitch (dM/in)/Iyy = %6.2f (ft-lbs/in)/(slug ft^2)\n',cpipitch)
fprintf(' roll (dR/in)/Ixx = %6.2f (ft-lbs/in)/(slug ft^2)\n',cpiroll)
fprintf('  yaw (dN/in)/Izz = %6.2f (ft-lbs/in)/(slug ft^2)\n',cpiyaw)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause
    elseif choice==6,  % command bandwidth plots
clc
disp(' ')
disp('After you view a bode plot of the transfer function from')
disp('input to state output, a meta file is made.  When you exit,')
disp('a screen will tell you the file names of the meta files.')
disp('To get a hard copy of the plots, you must graphics post')
disp('process (GPP) the files for your particular printer set-up')
disp('then print.')
disp(' ')
      if Vinf<20
        cmdbwplh
      else,
        cmdbwplc
      end
    elseif choice==0,
      flag=0;
    else
      disp(' ')
      disp('Enter a number on the menu')
      pause(3)
    end
  end
end
%
%   *** output to disk (text file) ***
%
diary on
diary off
delete diary
diary on
disp('                      *** RESULTS ***')
eval(['disp('''                          ',filename1,''')'])
disp(' ')
disp('                    *** INPUT DATA ***')
disp(' ')
disp('                       Flight Conditions')
disp(' ')
fprintf('             Forward velocity = %6.0f kts\n',Vinf/1.69)
fprintf('                  Temperature = %6.0f degs F\n',temp)
fprintf('            Pressure altitude = %6.0f ft\n',PA)
fprintf('             Auxiliary thrust = %6.0f lbs\n',Taux)
disp(' ')
disp('                            Fuselage')
```

```
disp(' ')
fprintf('                    Gross weight = %6.0f lbs\n',GW)
fprintf('   Equivalent flat plate area = %6.1f ft^2\n',Afh)
fprintf('      Vertical projected area = %6.1f ft^2\n',Afv)
fprintf('    CG height above waterline = %6.1f ft\n',zcg)
fprintf('          CG fuselage station = %6.1f ft\n',xcg)
fprintf('   CG position rt of buttline = %6.1f ft\n',ycg)
fprintf('                          Ixx = %6.1f slug ft^2\n',Ixx)
fprintf('                          Iyy = %6.1f slug ft^2\n',Iyy)
fprintf('                          Izz = %6.1f slug ft^2\n',Izz)
fprintf('                          Ixz = %6.1f slug ft^2\n',Ixz)
fprintf('               Downwash ratio = %6.2f \n',vfv1)
disp(' ')
disp('                        Main Rotor')
disp(' ')
fprintf('             Number of blades = %6.0f \n',b)
fprintf('                 Rotor radius = %6.1f ft\n',R)
fprintf('                  Blade chord = %6.1f ft\n',cblade)
fprintf('                  Blade twist = %6.2f degs\n',twist*57.3)
if airfoil==1,
   disp('                Blade airfoil =  HH-02')
else
   disp('                Blade airfoil =  VR-12')
end
fprintf('       Blade lift curve slope = %6.2f \n',a)
fprintf('                  Blade weight = %6.1f lbs\n',wblade)
fprintf('          Rotational velocity = %6.2f rads/sec\n',omega)
fprintf('            Blade grip length = %6.1f ft\n',grip)
fprintf('                  Hinge offset = %6.1f ft\n',e)
fprintf('   Flapping moment of inertia = %6.1f slug ft^2\n',Ib)
fprintf('   Hub height above waterline = %6.1f ft\n',hmd)
fprintf('         Hub fuselage station = %6.1f ft\n',lmd)
fprintf('  Hub position rt of buttline = %6.1f ft\n',ymd)
fprintf('                Mast incidence = %6.2f deg\n',im*57.3)
disp(' ')
if notar==0
  disp('                  Tail rotor (zero if NOTAR)')
  disp(' ')
  fprintf('             Number of blades = %6.1f \n',bt);
  fprintf('                  Blade chord = %6.1f ft\n',cot)
  fprintf('                 Blade radius = %6.1f ft\n',Rt)
  fprintf('             Lift curve slope = %6.2f \n',at)
  fprintf('           Rotational velocity = %6.2f rad/sec\n',ohmt)
  fprintf('    Flapping moment of inertia = %6.1f slug ft^2\n',Ibt)
  fprintf('                 Delta-3 angle = %6.2f deg\n',delta3*57.3)
  fprintf('                   Blade twist = %6.2f deg\n',thetalt*57.3);
  fprintf('    Hub height above waterline = %6.1f ft\n',htd)
  fprintf('          Hub fuselage station = %6.1f ft\n',ltd)
  fprintf('   Hub position rt of buttline = %6.1f ft\n',ytd)
  disp(' ')
elseif notar==1
  disp('                              NOTAR')
  disp(' ')
  fprintf('       Height above waterline = %6.1f ft^2\n',htnd)
  fprintf('             Fuselage station = %6.1f ft^2\n',ltnd)
  fprintf('    Position right of buttline = %6.1f ft^2\n',ytnd)
  fprintf('           NOTAR boom diameter = %6.1f ft^2\n',dian)
  fprintf('           Swirl angle at boom = %6.2f deg\n',swirl*57.3)
  fprintf('        Maximum thruster force = %6.1f lbs\n',Ytmaxn)
  fprintf(' Thrust fuselage station = %6.1f ft^2\n',lttnd)
```

```
  disp(' ')
end
disp('                              Wing')
disp(' ')
fprintf('                              Area = %6.1f ft^2\n',Swing)
fprintf('                              Span = %6.1f ft\n',bwing)
fprintf('                                CL = %6.2f \n',CLwing)
fprintf('                               CDo = %6.4f \n',CDowing)
fprintf('                          Tip cord = %6.1f ft\n',ctw)
fprintf('                         Root cord = %6.1f ft\n',crw)
fprintf('         Wing efficiency factor = %6.2f \n',ewing)
fprintf('                Zero lift angle = %6.2f deg\n',alplow*57.3)
fprintf('             Angle of incidence = %6.2f deg\n',iw*57.3)
fprintf('               Lift curve slope = %6.2f \n',aw)
fprintf('         Height above waterline = %6.1f ft\n',hwd)
fprintf('               Fuselage station = %6.1f ft\n',lwd)
fprintf('    Position right of buttline = %6.1f ft\n',ywd)
fprintf('          Rotor downwash ratio = %6.2f \n',vwv1)
fprintf('       Fuselage downwash ratio = %6.2f \n',detafdalpfw)
disp(' ')
disp('                          Horizontal tail')
disp(' ')
fprintf('                              Area = %6.1f ft^2\n',Shoriz)
fprintf('                              Span = %6.1f ft\n',bhoriz)
fprintf('                                CL = %6.2f \n',CLhoriz)
fprintf('                               CDo = %6.4f \n',CDohoriz)
fprintf('                Zero lift angle = %6.2f deg\n',alploh*57.3)
fprintf('             Angle of incidence = %6.2f deg\n',iw*57.3)
fprintf('               Lift curve slope = %6.2f \n',ah)
fprintf('         Height above waterline = %6.1f ft\n',hhd)
fprintf('               Fuselage station = %6.1f ft\n',lhd)
fprintf('    Position right of buttline = %6.1f ft\n',yhd)
fprintf('         Dynamic pressure ratio = %6.2f \n',qhq)
fprintf('           Rotor downwash ratio = %6.2f \n',vhv1)
fprintf('       Fuselage downwash ratio = %6.2f \n',detafdalpfh)
disp(' ')
disp('                         Vertical tail')
disp(' ')
fprintf('                              Area = %6.1f ft^2\n',Svert)
fprintf('                              Span = %6.1f ft\n',bvert)
fprintf('                                CL = %6.2f \n',CLvert)
fprintf('                               CDo = %6.4f \n',CDovert)
fprintf('         Height above waterline = %6.1f ft\n',hvd)
fprintf('               Fuselage station = %6.1f ft\n',lvd)
fprintf('    Position right of buttline = %6.1f ft\n',yvd)
fprintf('                Zero lift angle = %6.2f deg\n',alplov*57.3)
fprintf('                     Maximum Cl = %6.2f \n',clvertmax)
fprintf('         Dynamic pressure ratio = %6.2f \n',qvq)
fprintf('               Lift curve slope = %6.2f \n',av)
disp(' ')
disp('                               Rigging')
disp(' ')
fprintf('  Long cyclic pitch/inch defl = %6.2f deg/in\n',db1mddele*57.3)
fprintf('   Lat cyclic pitch/inch defl = %6.2f deg/in\n',da1mddela*57.3)
fprintf('             Collective    pitch/inch    defl    =    %6.2f
deg/in\n',dthetomddelc*57.3)
fprintf('    Tail    rotor    pitch    change/defl    =    %6.2f
deg/unit\n',dthetotddelp*57.3)
disp('    Max deflection of control')
fprintf('       from neutral for NOTAR = %6.2f units\n',dphinddelp*57.3)
```

```
disp('  Displacement of anti-torque')
fprintf('    control until full rudder = %6.2f units\n',maxr)
disp(' ')
disp('                    *** CALCULATED DATA ***')
disp(' ')
disp('                         Main Rotor')
disp(' ')
fprintf('                          thrust = %6.1f lbs\n',T)
fprintf('                          torque = %6.1f ft-lbs\n',Qrotor)
fprintf('                  advance ratio = %6.1f \n',mu)
fprintf('     inflow parameter wrt TPP = %6.3f \n',lamp)
fprintf('                Tip path angle = %6.1f degs\n',altpp*57.3)
fprintf('           Rotor coning angle = %6.1f degs\n',ao*57.3)
fprintf('       1st lat cyclic term-A1 = %6.2f degs\n',A1*57.3)
fprintf('      1st long cyclic term-B1 = %6.2f degs\n',B1*57.3)
fprintf('             lateral flapping = %6.1f degs\n',b1s*57.3)
fprintf('        longitudinal flapping = %6.1f degs\n',a1s*57.3)
fprintf('                  Lock number = %6.1f \n',lockno)
disp(' ')
disp('                   Tail Rotor (zero if NOTAR)')
disp(' ')
fprintf('             tail rotor thrust = %6.1f lbs\n',Tt)
fprintf('                 advance ratio = %6.1f \n',mut)
fprintf('              inflow parameter = %6.3f \n',lampt)
fprintf('            Rotor coning angle = %6.1f degs\n',aot*57.3)
fprintf('              lateral flapping = %6.2f degs\n',b1st*57.3)
fprintf('        longitudinal flapping = %6.2f degs\n',a1st*57.3)
fprintf('                  Lock number = %6.1f \n',locknot)
disp(' ')
disp(' ')
disp('                       State Matrices')
disp(' ')
disp('Longitudinal uncoupled plant (A or F depending on notation)')
disp('States are [u w q theta]')
disp(' ')
disp(Flonaug)
disp(' ')
disp(' ')
disp('Longitudinal uncoupled input matrix (B or G depending on notation)')
disp('Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]')
disp(' ')
disp(Glonaug)
disp(' ')
disp(' ')
disp('Lateral/directional uncoupled plant (A or F depending on notation)')
disp('States are [v p phi r psi]')
disp(' ')
disp(Flataug)
disp(' ')
disp(' ')
disp('Lateral/directional uncoupled input matrix (B or G depending')
disp('on notation)')
disp('Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]')
disp(Glataug)
disp(' ')
disp(' ')
disp('Coupled plant (A or F depending on notation)')
disp('States are [u w q theta v p phi r psi]')
```

```
disp(' ')
disp(Amat)
disp(' ')
disp(' ')
disp('Coupled input matrix (B or G depending on notation)')
disp('Inputs are [longitudinal cyclic, collective, lateral cyclic,
pedals]')
disp(' ')
disp(Bmat)
disp(' ')
disp(' ')
disp('                        Eigenvalue')
disp(' ')
disp('Uncoupled')
disp(' ')
disp('Longitudinal plant')
disp(' ')
disp('        Root              wn              damping')
[wn,z]=damp(Rlonaug);
disp([Rlonaug,wn,z])
disp(' ')
disp('Lateral/Directional plant')
[wn,z]=damp(Rlataug);
disp(' ')
disp('        Root              wn              damping')
disp([Rlataug,wn,z])
disp(' ')
disp(' ')
disp('Coupled Plant')
disp(' ')
[wn,z]=damp(Rcoup);
disp('        Root              wn              damping')
disp([Rcoup wn z])
disp(' ')
disp(' ')
disp('          *** KEY CONTROL PARAMETERS ***')
disp(' ')
fprintf('   cross coupling = %6.2f \n',xcouple)
   disp(' ')
   disp('                          Designed damping')
fprintf('             pitch = %6.1f ft-lbs/(rad/sec)\n',desdmdq)
fprintf('              roll = %6.1f ft-lbs/(rad/sec)\n',desdrdp)
fprintf('               yaw = %6.1f ft-lbs/(rad/sec)\n',desdndr)
   disp(' ')
   disp('                            Control Power')
fprintf('             pitch = %6.1f ft-lbs/in\n',cppitch)
fprintf('              roll = %6.1f ft-lbs/in\n',cproll)
fprintf('               yaw = %6.1f ft-lbs/in\n',cpyaw)
   disp(' ')
   disp('                Cooper Harper Pilot Ratings')
   disp('                 damping/moment of inertia')
fprintf('pitch    (dM/dq)/Iyy    =    %6.2f    [ft-lbs/(rad/sec)]/(slug
ft^2)\n',prpitch)
fprintf('    roll    (dR/dp)/Ixx    =    %6.2f    [ft-lbs/(rad/sec)]/(slug
ft^2)\n',prroll)
fprintf('       yaw    (dN/dr)/Izz    =    %6.2f    [ft-lbs/(rad/sec)]/(slug
ft^2)\n',pryaw)
   disp(' ')
   disp('              control power/moment of inertia')
fprintf('pitch (dM/in)/Iyy = %6.2f (ft-lbs/in)/(slug ft^2)\n',cpipitch)
```

```
fprintf(' roll (dR/in)/Ixx = %6.2f (ft-lbs/in)/(slug ft^2)\n',cpiroll)
fprintf('  yaw (dN/in)/Izz = %6.2f (ft-lbs/in)/(slug ft^2)\n',cpiyaw)
disp(' ')
disp(' ')
diary off
%
clc
disp(' ')
disp('     *** OUTPUT DATA INSTRUCTIONS (screen 1 of 3) ***')
disp(' ')
disp('Because this subroutine generates a large number of single')
disp('value data not shown on the output screen, a text file')
disp('VARLIST.TXT is on this disk which lists the variable names')
disp('for all the stability derivatives.  Stability derivative')
disp('contributions for all major aircraft components can be found')
disp('by reading the text file VARLIST.TXT, then asking MATLAB the')
disp('variable name corresponding to the derivative.')
disp(' ')
disp('Press any key to continue')
pause
 clc
disp(' ')
disp('     *** OUTPUT DATA INSTRUCTIONS (screen 2 of 3) ***')
disp(' ')
disp('A. Data from the output screen saved to a file named:')
eval(['disp(''   ',filename1,'.stb'')'])
disp(' ')
disp('B. This is a text file, use the TYPE command to view the file')
disp('   or use a text editor to view/print the file.')
eval(['flag=exist(''',filename1,'.stb'');']);
  if flag < 1,
    eval(['!rename diary ',filename1,'.stb''']);
  else
    eval(['!del ',filename1,'.stb''']);
    eval(['!rename diary ',filename1,'.stb''']);
  end
disp(' ')
disp('C. Matrix and vector data saved to a default file named:')
disp('   mstabdat.mat')
disp(' ')
disp('D. This is a ".mat" binary file, use the LOAD command to')
disp('   retrieve the data for plotting.')
disp(' ')
disp('E. Rename "mstabdat.mat" to another ".mat" file.')
disp('   The file "mstabdat.mat" will be overwritten when')
disp('   the program is executed.')
disp(' ')
eval(['disp(''F. Do not rename the file as "',filename1,'.mat"'')'])
eval(['disp(''   The file "',filename1,'.mat" is already on disk'')'])
disp('   and used for future editing.')
disp(' ')
disp('Press any key to continue')
pause
%
%   *** Output to disk (.mat file containing matrix variables
%   Amat Bmat Rcoup Flataug Glataug Rlataug Plataug Flataug
%   Glataug Rlonaug Plonaug
%
%   *** Configuring variables for output ***
%
```

```
save mstabdat Amat Bmat Rcoup Flataug Glataug Rlataug Plataug Flataug ...
 Glataug Rlonaug Plonaug
%
clc
disp(' ')
disp('      *** OUTPUT DATA INSTRUCTIONS (screen 3 of 3) ***')
disp(' ')
disp('A. Single value data saved to a default file named:')
disp('   vstabdat.mat')
disp(' ')
disp('B. This is a ".mat" binary file, use the LOAD command to')
disp('   retrieve the data for plotting.')
disp(' ')
disp('C. Rename "vstabdat.mat" to another ".mat" file.')
disp('   The file "vstabdat.mat" will be overwritten when')
disp('   the program is executed.')
disp(' ')
eval(['disp(''D. Do not rename the file as "',filename1,'.mat"'')'])
eval(['disp(''   The file "',filename1,'.mat" is already on disk'')'])
disp('   and used for future editing.')
%
%   *** Configuring variables for output ***
%
clear Amat Bmat Rcoup Flataug Glataug Rlataug Plataug Flataug ...
 Glataug Rlonaug Plonaug num den vA1 vB1 va1s vb1s vmu vtheta7 ...
 vao vv1 vlamp vthetao vctsig vcqsig vchsig valtpp
%
save vstabdat
clear
clc
disp(' ')
disp('              *** END STABILITY AND CONTROL ROUTINE ***')
disp(' ')
disp('press any key to continue...')
pause
format loose
%
% return to JANRAD.M
```

M. CMDBWPLH.M

```
% CMDBWPLH.M
% open loop response plots for longitudinal and lateral plants %
disp('While viewing a plot, press any key to go to the next plot')
disp(' ')
disp('Do you want to see longitudinal or lateral/directional plots?')
disp(' ')
disp('1.  Longitudinal (eight plots total).')
disp('2.  Lateral Directional (ten plots total).')
disp(' ')
pview=input('Enter a number : ');
clc
w=logspace(-2,2);
if pview==1
% open loop response plots for longitudinal and lateral plants
w=logspace(-2,2);
Du=[0 0 0 0];
Cu=[1 0 0 0];
Cthet=[0 0 0 1];
Cqrat=[0 0 1 0];
Cw=[0 1 0 0];
disp('longitudinal cylic')
% command bw e to u
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cu,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to U, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2uh.*
meta cbe2uh
% command bw e to theta
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cthet,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to Theta, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2theh.*
meta cbe2theh
% command bw e to q
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cqrat,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to Pitch Rate, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2qh.*
meta cbe2qh
% command bw e to w
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cw,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to W, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2wh.*
meta cbe2wh
%
% now collective
disp('collective')
% command bw c to u
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cu,Du,2);
```

```
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Collective to U, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2uh.*
meta cbc2uh
% command bw c to theta
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cthet,Du,2);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Collective to Pitch, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2theh.*
meta cbc2theh
% command bw c to q
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cqrat,Du,2);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Collective to Pitch Rate, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2qh.*
meta cbc2qh
% command bw c to w
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cw,Du,2);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Collective to W, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2wh.*
meta cbc2wh
clc
disp(' ')
disp('Plots are saved under the following filenames:')
disp(' ')
disp('Longitudinal Cyclic')
disp('Longitudinal Cyclic to U, Hover - cbe2uh.met')
disp('Longitudinal Cyclic to Theta, Hover - cbe2theh.met')
disp('Longitudinal Cyclic to Pitch Rate, Hover - cbe2qh.met')
disp('Longitudinal Cyclic to W, Hover - cbe2wh.met')
disp('')
disp('Collective')
disp('Collective to U, Hover - cbc2uh.met')
disp('Collective to Pitch, Hover - cbc2theh.met')
disp('Collective to Pitch Rate, Hover - cbc2qh.met')
disp('Collective to W, Hover - cbc2wh.met')
disp(' ')
disp('press any key to continue ...')
pause
%
% now for lateral directional plant
elseif pview==2
%
% now for lateral directional plant
Cphi=[0 0 1 0 0];
Cv=[1 0 0 0 0];
Cp=[0 1 0 0 0];
Cr=[0 0 0 1 0];% yaw rate
Cy=[0 0 0 0 1];% yaw angle
% lateral cyclic
disp('lateral cyclic')
```

```
% command bw lateral cyclic to bank
[NUM,DEN]=ss2tf(Flataug,Glataug,Cphi,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Lateral Cyclic to Bank, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cba2phih.*
meta cba2phih
% command bw lateral cyclic to sideslip (lateral velocity)
[NUM,DEN]=ss2tf(Flataug,Glataug,Cv,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Lateral Cyclic to Sideslip (v), Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cba2vh.*
meta cba2vh
% command bw lateral cyclic to roll rate
[NUM,DEN]=ss2tf(Flataug,Glataug,Cp,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Lateral Cyclic to Roll Rate, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cba2ph.*
meta cba2ph
% command bw lateral cyclic to yaw rate
[NUM,DEN]=ss2tf(Flataug,Glataug,Cr,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Lateral Cyclic to Yaw Rate, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cba2rh.*
meta cba2rh
% command bw lateral cyclic to yaw angle
[NUM,DEN]=ss2tf(Flataug,Glataug,Cy,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Lateral Cyclic to Yaw, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cba2yh.*
meta cba2yh
%
% pedals
% command bw pedals to bank
disp('pedals')
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cphi,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Bank, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2phih.*
meta cbp2phih
% command bw pedals to sideslip
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cv,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Sideslip (v), Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2vh.*
meta cbp2vh
% command bw pedals to roll rate
```

```
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cp,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Roll Rate, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2ph.*
meta cbp2ph
% command bw pedals to yaw rate
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cr,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Yaw Rate, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2rh.*
meta cbp2rh
% command bw pedals to yaw
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cy,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Yaw, Hover')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2yh.*
meta cbp2yh
clc
disp(' ')
disp('Plots are saved under the following filenames:')
disp(' ')
disp('Lateral cyclic')
disp('Lateral Cyclic to Bank, Hover - cba2phih.met')
disp('Lateral Cyclic to Sideslip (v), Hover - cba2vh.met')
disp('Lateral Cyclic to Roll Rate, Hover - cba2ph.met')
disp('Lateral Cyclic to Yaw Rate, Hover - cba2rh.met')
disp('Lateral Cyclic to Yaw, Hover - cba2yh.met')
disp('Pedals')
disp('Pedals to Bank, Hover - cbp2phih.met')
disp('Pedals to Sideslip (v), Hover - cbp2vh.met')
disp('Pedals to Roll Rate, Hover - cbp2ph.met')
disp('Pedals to Yaw Rate, Hover - cbp2rh.met')
disp('Pedals to Yaw, Hover - cbp2yh.met')
disp(' ')
disp('press any key to continue ...')
pause
%
else
end
```

N. CMDBWPLC.M

```
% CMDBWPLC.M
% open loop response plots for longitudinal and lateral plants
%
disp('While viewing a plot, press any key to go to the next plot')
disp(' ')
disp('Do you want to see longitudinal or lateral/directional plots?')
disp(' ')
disp('1.  Longitudinal (eight plots total).')
disp('2.  Lateral Directional (ten plots total).')
disp(' ')
pview=input('Enter a number : ');
clc
w=logspace(-2,2);
if pview==1
Du=[0 0 0 0];
Cu=[1 0 0 0];
Cthet=[0 0 0 1];
Cqrat=[0 0 1 0];
Cw=[0 1 0 0];
disp('longitudinal cylic')
% command bw e to u
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cu,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to U, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2uc.*
meta cbe2uc
% command bw e to theta
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cthet,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to Theta,  Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2thec.*
meta cbe2thec
% command bw e to q
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cqrat,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to Pitch Rate, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2qc.*
meta cbe2qc
% command bw e to w
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cw,Du,1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Longitudinal Cyclic to W, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbe2wc.*
meta cbe2wc
%
% now collective
disp('collective')
% command bw c to u
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cu,Du,2);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
```

```
title('Open loop response Collective to U, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2uc.*
meta cbc2uc
% command bw c to theta
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cthet,Du,2);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Collective to Pitch, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2thec.*
meta cbc2thec
% command bw c to q
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cqrat,Du,2);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Collective to Pitch Rate, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2qc.*
meta cbc2qc
% command bw c to w
[NUM,DEN]=ss2tf(Flonaug,Glonaug,Cw,Du,2);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Collective to W, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbc2wc.*
meta cbc2wc
clc
disp(' ')
disp('Plots are saved under the following filenames:')
disp(' ')
disp('Longitudinal Cyclic')
disp('Longitudinal Cyclic to U, Cruise - cbe2uc.met')
disp('Longitudinal Cyclic to Theta, Cruise - cbe2thec.met')
disp('Longitudinal Cyclic to Pitch Rate, Cruise - cbe2qc.met')
disp('Longitudinal Cyclic to W, Cruise - cbe2wc.met')
disp('')
disp('Collective')
disp('Collective to U, Cruise - cbc2uc.met')
disp('Collective to Pitch, Cruise - cbc2thec.met')
disp('Collective to Pitch Rate, Cruise - cbc2qc.met')
disp('Collective to W, Cruise - cbc2wc.met')
disp(' ')
disp('press any key to continue ...')
pause
%
% now for lateral directional plant
elseif pview==2
Cphi=[0 0 1 0 0];
Cv=[1 0 0 0 0];
Cp=[0 1 0 0 0];
Cr=[0 0 0 1 0];% yaw rate
Cy=[0 0 0 0 1];% yaw angle
% lateral cyclic
disp('lateral cyclic')
% command bw lateral cyclic to bank
[NUM,DEN]=ss2tf(Flataug,Glataug,Cphi,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
```

```
title('Open loop response Lateral Cyclic to Bank, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cba2phic.*
meta cba2phic
% command bw lateral cyclic to sideslip (lateral velocity)
[NUM,DEN]=ss2tf(Flataug,Glataug,Cv,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
title('Open loop response Lateral Cyclic to Sideslip (v), Cruise')
pause
!del cba2vc.*
meta cba2vc
% command bw lateral cyclic to roll rate
[NUM,DEN]=ss2tf(Flataug,Glataug,Cp,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Lateral Cyclic to Roll Rate, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cba2pc.*
meta cba2pc
% command bw lateral cyclic to yaw rate
[NUM,DEN]=ss2tf(Flataug,Glataug,Cr,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
title('Open loop response Lateral Cyclic to Yaw Rate, Cruise')
pause
!del cba2rc.*
meta cba2rc
% command bw lateral cyclic to yaw angle
[NUM,DEN]=ss2tf(Flataug,Glataug,Cy,Du,3);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
title('Open loop response Lateral Cyclic to Yaw, Cruise')
pause
!del cba2yc.*
meta cba2yc
%
% pedals
% command bw pedals to bank
disp('pedals')
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cphi,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Bank, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2phic.*
meta cbp2phic
% command bw pedals to sideslip
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cv,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Sideslip (v), Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2vc.*
meta cbp2vc
% command bw pedals to roll rate
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cp,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Roll Rate, Cruise')
```

```
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2pc.*
meta cbp2pc
% command bw pedals to yaw rate
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cr,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Yaw Rate, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2rc.*
meta cbp2rc
% command bw pedals to yaw
[NUM,DEN]=ss2tf(Flataug,Glataug(:,4)*30,Cy,[0],1);
semilogx(w,20*log10(bode(NUM,DEN,w))),grid
title('Open loop response Pedals to Yaw, Cruise')
xlabel('Angular Frequency (rad/sec)'),ylabel('Gain (dB)')
pause
!del cbp2yc.*
meta cbp2yc
%
clc
disp(' ')
disp('Plots are saved under the following filenames:')
disp(' ')
disp('Lateral cyclic')
disp('Lateral Cyclic to Bank, Cruise - cba2phic.met')
disp('Lateral Cyclic to Sideslip (v), Cruise - cba2vc,met')
disp('Lateral Cyclic to Roll Rate, Cruise - cba2pc.met')
disp('Lateral Cyclic to Yaw Rate, Cruise - cba2rc.met')
disp('Lateral Cyclic to Yaw, Cruise - cba2yc.met')
disp('Pedals')
disp('Pedals to Bank, Cruise - cbp2phic.met')
disp('Pedals to Sideslip (v), Cruise - cbp2vc.met')
disp('Pedals to Roll Rate, Cruise - cbp2pc.met')
disp('Pedals to Yaw Rate, Cruise - cbp2rc.met')
disp('Pedals to Yaw, Cruise - cbp2yc.met')
disp(' ')
disp('press any key to continue ...')
pause
%
else
end
```

```
0.   DERIV2.M
% DERIV2.M
% computes the derivative of a polynomial
function[deriv]=deriv(polyno);
der=[2 1 0];
d=der.*polyno;
deriv=[d(1,1) d(1,2)];
```

LIST OF REFERENCES

1. Nicholson, Robert K. Jr., Computer Code for Interactive Rotorcraft Preliminary Design Using a Harmonic Balance Method for Rotor Trim, Naval Postgraduate School, Monterey, CA, 1993.

2. Prouty, Raymond W., Helicopter Performance, Stability, and Control, Krieger Publishing, Malabar, FL, 1990.

3. Talbot, Peter D., Bruce E. Tinling, William A. Decker, and Robert T. N. Chen, A Mathematical Model of a Single Main Rotor Helicopter for Piloted Simulation, NASA Technical Memorandum 84281, Ames Research Center, Moffett Field, CA, 1982.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Professor Daniel J. Collins, Code AA/Co Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor E. Roberts Wood, code AA/Wd Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5000	2
5.	Professor Richard M. Howard AA/Ho Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5000	1
6.	MAJ Robert K. Nicholson, USA 1116 Walker Road Great Falls, Virginia 22066	1
7.	MAJ Walter M. Wirth, USA P.O. Box 201 456 North Monroe Street Monticello, Wisconsin 53570	1
8.	Dr. Dev Banerjee Bldg. 530/B325 McDonnell Douglas Helicopter Co. 5000 E. McDowell road Mesa, AZ 85215	1
9.	Mr. Dean Carico RW04B NAWCAD Patuxent River, MD 20670-5304	1
10.	Mr. Dean E. Cooper Sikorsky Aircraft Div., UTC 6900 Main Street Stratford, CT 06601	1

11. Mr. Andrew W. Kerr 1
Aeroflightdynamics Directorate
NASA Ames Research Center
Moffett Field, CA 94035

12. Mr. John C. McKeown 1
Naval Air Systems Command
Code AIR-530A
1421 Jefferson Highway
Arlington, VA 22243-5300