Theses and Dissertations   1. Thesis and Dissertation Collection, all items

1993-09

Computer code for interactive rotorcraft preliminary design using a harmonic balance method for rotor trim

Nicholson, Robert Kirk, Jr.

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/39985

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A274 924

DTIC
S ELECTE
E JAN 2 5 1994
D

THESIS

COMPUTER CODE FOR INTERACTIVE ROTORCRAFT
PRELIMINARY DESIGN USING A
HARMONIC BALANCE METHOD FOR ROTOR TRIM

by

Robert Kirk Nicholson, Jr.

September, 1993

Thesis Advisor: E. Roberts Wood

94-02117

94 1 25 014

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 23 September 1993	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE COMPUTER CODE FOR INTERACTIVE ROTORCRAFT PRELIMINARY DESIGN USING A HARMONIC BALANCE METHOD FOR ROTOR TRIM	5. FUNDING NUMBERS
6. AUTHOR(S) Robert K. Nicholson, 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)*

The Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) computer program was developed to aid in the analysis of helicopter rotor performance, stability and control, and rotor dynamics. JANRAD is an interactive, user friendly program, capable of accurately and quickly solving helicopter design problems at the preliminary design level. The program was written as a collection of MATLAB* script and function files (M-files) using the 386-MATLAB* version 3.5 programming language. The M-file *janrad.m* invokes the user interface routines and calls various analysis modules (M-files) which contain the appropriate analysis and output routines. Each of these modules use a common routine, *trim.m*, which employs blade element theory and a harmonic balance method for rotor trim. The program is limited to conditions of steady flight with no winds and is accurate at a hover and for forward airspeeds greater than or equal to 50 knots.

14. SUBJECT TERMS Helicopter, Preliminary Design, Rotor Trim, Harmonic Balancing, Blade Element Theory	15. NUMBER OF PAGES 91
	16. PRICE CODE

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

COMPUTER CODE FOR INTERACTIVE ROTORCRAFT
PRELIMINARY DESIGN USING A
HARMONIC BALANCE METHOD FOR ROTOR TRIM

by

Robert Kirk Nicholson, Jr.
Major, United States Army
B.S., United States Military Academy, 1980
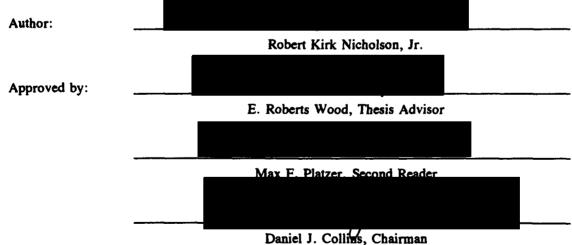
Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September, 1993

Author: _____
Robert Kirk Nicholson, Jr.

Approved by: _____
E. Roberts Wood, Thesis Advisor

_____
Max F. Platzer, Second Reader

_____
Daniel J. Collins, Chairman
Department of Aeronautics and Astronautics

ii

ABSTRACT

The Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) computer program was developed to aid in the analysis of helicopter rotor performance, stability and control, and rotor dynamics. JANRAD is an interactive, user friendly program, capable of accurately and quickly solving helicopter design problems at the preliminary design level. The program was written as a collection of MATLAB* script and function files (M-files) using the 386-MATLAB* version 3.5 programming language. The M-file *janrad.m* invokes the user interface routines and calls various analysis modules (M-files) which contain the appropriate analysis and output routines. Each of these modules use a common routine, *trim.m*, which employs blade element theory and a harmonic balance method for rotor trim. The program is limited to conditions of steady flight with no winds and is accurate at a hover and for forward airspeeds greater than or equal to 50 knots.

DTIC QUALITY INSPECTED 8

Accesion For		
NTIS CRA&I		☒
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Dist. ibution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

iii

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Special thanks goes to my advisor, Professor Wood, for his time, patience, and valuable advice. I would also like to thank the entire helicopter design class of 1993 for their patience and valuable feedback during the initial stages of this thesis.

Finally, I thank my lovely wife, Cheryl, for her understanding and support during the many hours I spent glued to the computer. With all my love, I dedicate this work to her, and the beautiful son she just gave me, Zachary.

I. INTRODUCTION

A. BACKGROUND

The development of rotorcraft design is a complicated iterative process involving the interaction of many variables. It is an especially difficult and time consuming task when the required calculations are performed by hand. There are several helicopter design software packages that perform the required calculations and automate the design analysis process. In searching for a computer program suitable for helicopter preliminary design, students found that current helicopter design codes were either too difficult to use, excessively detailed, or functionally incomplete. As a result, the need for a custom computer code was identified. Through this thesis, the Joint Army/Navy Rotorcraft Analysis and Design (JANRAD) computer program was developed to meet the specific needs of the Naval Postgraduate School for preliminary helicopter design.

JANRAD was written as an interactive, user friendly program, capable of accurately and quickly solving helicopter design problems at the preliminary design level. The program was intended primarily for analysis of helicopter rotor performance, stability and control, and rotor dynamics. It was not intended to perform comprehensive helicopter design analyses.

JANRAD was written as a collection of MATLAB* script files (M-files). It was written specifically for version 3.5 of 386-MATLAB* [Ref. 1]. This was done for two reasons. First, MATLAB* was ideally suited for the types of calculations and

1

analyses involved in the preliminary helicopter design process. Second, engineering design students had typically used MATLAB* for several projects and had a good working knowledge of the software and the programming language. The intent was to have a powerful and accurate computer aided design tool that one could master in a matter of minutes. In using JANRAD, students could occupy their time analyzing design issues rather than learning software.

B. APPROACH TO THE PROBLEM

The approach in developing JANRAD was to provide the design student with a rapid means of changing design parameters and obtaining accurate results over a wide range of helicopter configurations. To accomplish this, user input requirements were kept to a minimum, efficiency in parameter calculation routines was optimized, simplifying assumptions were made, and convergence criteria for iterative processes were carefully set to appropriate levels.

To keep user input to a minimum, airframe and rotor geometric input parameters were minimized. The rotor blade was limited to a rectangular, uniform, rigid structure with linear twist. This allowed single value inputs for setup of the rotor blade parameters. The wing and tail section inputs were limited by using area and span to determine other geometric values. User input requirements were also reduced by requiring aircraft gross weight, equivalent flat plate area, auxiliary thrust, and aerodynamic surface coefficients of lift to be precalculated.

To maximize the efficiency of the iterative processes, whenever possible, MATLAB's* built-in vector and matrix operations were used in place of repetitive

2

"for" and "while" loops. In addition, adjustments to the independent variables during the iterative processes were varied according to certain parameters such as advance ratio, density ratio, and magnitude of error. By varying the increment or decrement of the independent variables according to the appropriate parameters, the speed of convergence was greatly increased.

In making assumptions and setting convergence criteria, the trade-offs between speed and accuracy were balanced by keeping in mind that JANRAD was developed for helicopter preliminary design. Accuracy was maintained for a wide range of helicopter configurations by using blade element theory to determine resultant forces and moments in the rotor system. Overall speed was maintained by making simplifying assumptions and selecting convergence criteria that resulted in small percentage errors that were acceptable for preliminary design.

C. PROGRAM OUTLINE

Currently, JANRAD consists of a group of 22 MATLAB* script and function files. The main M-file, *janrad.m*, invokes the user interface functions and the various analysis modules. The routine allows the user to interactively input environment and design configuration parameters using the computer keyboard and screen. The user can either create or edit files containing parameters for specific designs and situations. The routine also provides the interface for selection of the various design analyses. A listing of *janrad.m* is found at Appendix A. Currently, there is a module available for rotor performance and one for stability and control. A rotor dynamics routine is under development.

3

The heart of JANRAD is the M-file *trim.m*. This M-file receives the environment and design configuration parameters from *janrad.m* and then balances the forces and moments in the rotor system for the appropriate airspeed. By trimming the forces in the rotor system, *trim.m* produces the parameters necessary for the various analysis modules. A listing of *trim.m* is found at Appendix B.

The rotor performance module is called *perf.m*. This M-file accepts the data from *trim.m* and *janrad.m*, calculates the performance parameters, and generates the rotor performance output. Appendix C contains a listing of *perf.m*. The stability and control module, *stab.m* calculates and produces the stability and control output. Each M-file, with the exception of *stab.m*, is discussed in detail in this thesis. The details of the stability and control module are addressed in another thesis [Ref. 2]. Similarly, the details of the rotor dynamics module will be discussed in a future thesis.

II. PROGRAM DETAILS

A. GENERAL

JANRAD provides accurate data for standard or compound helicopters in hovering and forward flight. Forward flight analysis is only accurate for airspeeds of 50 knots or greater. Conditions are limited to level flight with no winds. JANRAD provides data for one flight condition and one design configuration at a time. To obtain a range of data for an overall analysis, the user must make multiple runs of JANRAD. For each run, the appropriate parameters would be changed and the results could then be compared to those of previous runs.

B. THEORY

The underlying principle in helicopter rotor design is that the main rotor forces and moments must be adjusted so the rotor provides the lift and propulsive thrust required to meet the conditions of flight. The process of adjusting the rotor's forces and moments is referred to as "trimming" the rotor. At a hover with no wind and assuming centered C.G. and no rolling moment due to the anti-torque device, there is no tilt required in the rotor disk. In this case, trimming the rotor involves adjusting collective pitch until the thrust generated by the rotor equals the weight of the aircraft plus the force created by the rotor downwash on the fuselage. In level, unaccelerated forward flight, assuming a centered C.G. and no rolling moments, the rotor disk must be tilted forward to create a vertical and a horizontal thrust component. The vertical thrust component must equal the weight of the aircraft (no downwash effects) minus

lift created by any aerodynamic surfaces (assuming no lift generated by the fuselage). The horizontal component must equal the drag created by the fuselage and other aircraft components minus any auxiliary thrust. Trimming the rotor in forward flight involves the adjustment of cyclic pitch (collective pitch is the steady term of cyclic pitch) to maintain the required rotor disk tilt.

For hovering and forward flight, JANRAD uses blade element theory to determine the rotor system forces and moments, and a method for harmonically balancing the forces and moments to adjust cyclic pitch [Ref. 3]. The only difference in the process for trimming the rotor system in hovering and forward flight is in how the rotor induced velocity is calculated. In forward flight, the rotor induced velocity is assumed uniform and is calculated using the inflow parameter λ_T [Ref. 4]. For hovering flight, induced velocity is determined by using an iterative process of equating the differential thrust calculated by momentum theory and blade element theory [Ref. 5, p. 38]. This process produces an elliptical non-uniform induced velocity distribution along the rotor blade.

C. ASSUMPTIONS

The following assumptions are made in order to reduce total computation time. The assumptions produce relatively small errors which are considered acceptable for helicopter preliminary design.

1. Center of Gravity

For the purposes of trimming the rotor system, the aircraft center of gravity is located at the center of the rotor hub. In doing so, the first-harmonic terms

of thrust moment are eliminated.

2. Lateral Forces and Moments

The rotor is trimmed longitudinally and vertically. Lateral forces and moments due to the anti-torque device and aerodynamic surfaces are ignored.

3. Rigid Rotor Blade

For the purposes of the rotor trimming routine, the rotor blade is considered a rigid structure. This eliminates the need to consider blade modal responses and the subsequent affects on the rotor blade local angle of attack.

4. Tip Loss

Rotor blade tip loss is determined using a relationship involving coefficient of thrust. This results in a steady value of tip loss with respect to azimuth angle. The harmonic characteristics of tip loss are not considered.

5. Aerodynamic Surfaces

Lift and drag for the aircraft wing, horizontal tail, and vertical tail are calculated using two dimensional theory. Spanwise flow and tip losses are not accounted for. The aircraft fuselage lift characteristics are also ignored.

6. Mach Number

The airfoil lift and drag curves (coefficients of lift and drag versus angle of attack) for separate Mach numbers were reduced to one Mach independent curve.

7. Wake Interactions

Variations in angle of attack due to the interaction of rotor wake, aerodynamic surface trailing vortices, or fuselage interference were not accounted for.

D. M-FILE DETAILS

1. Janrad.m

The main module for the JANRAD program is *janrad.m*. This M-file

first queries the user to either edit or create a data file. If editing is selected, the

program asks for the file name. If the file name is not found in the current directory,

the program will tell the user to try again or instruct them on how to exit JANRAD.

When the file name is found, a list of parameters is displayed, and the user is allowed

to make changes as desired. If the user decides to create a new data file, the

parameters are displayed one at a time. In each instance, the user is prompted to

input the desired value. Each prompt includes a display of the proper units. The

parameters eligible for editing or creation are identical and are listed below:

1. pressure altitude
2. temperature
3. airspeed
4. aircraft gross weight
5. number of blades in the rotor system
6. rotor blade radius
7. rotor blade chord
8. effective hinge offset
9. rotor blade grip length
10. rotor blade angle of twist
11. rotor blade weight
12. number of rotor blade elements
13. rotor system rotational velocity
14. number of rotor disk azimuth sectors
15. average lift curve slope of rotor blade airfoil
16. rotor blade airfoil
17. collective pitch angle at 70% rotor blade radius
18. equivalent aircraft flat plate area
19. aircraft vertical projected area
20. wing area
21. wing span
22. wing lift coefficient

23. wing profile drag coefficient
24. wing efficiency factor
25. horizontal tail area
26. horizontal tail span
27. horizontal tail lift coefficient
28. horizontal tail profile drag coefficient
29. vertical tail area
30. vertical tail span
31. vertical tail lift coefficient
32. vertical tail profile drag coefficient
33. auxiliary thrust

Once the editing or creation process is complete, the program then queries the user for a file name. JANRAD saves the data to the current directory with the specified file name and a ".mat" extension. This is a binary file containing the file label and the names and values of the 33 parameters.

Finally, the user is queried for a specific analysis. To date, the user can select rotor performance or aircraft stability and control. Janrad has provisions for a rotor system dynamics module. When the user selects rotor performance, the module *perf.m* is called. Similarly, for stability and control, *stab.m* is called. The user can also make selections to either change the current data (edit current parameters or create new ones) or exit the program.

2. Trim.m

The M-file *trim.m* trims the rotor system as appropriate for the environment and design parameters obtained as input in *janrad.m*. The rotor trimming process involves four steps:

1. Calculate required parameters.
2. Guess rotor drag, cyclic pitch, and location of resultant thrust vector.
3. Trim the rotor system (calculate cyclic pitch).
4. Calculate rotor drag and location of resultant thrust vector.

9

Initially, the program uses values for rotor drag, cyclic pitch, and the location of the resultant thrust vector based on established averages. The rotor system is then trimmed using those values and the other parameters previously calculated. Once the rotor is trimmed, rotor drag and the location of the resultant thrust vector are calculated. If the values for these two parameters fall within certain evaluation criteria (20% of prior rotor drag and 1.5% of prior location of resultant thrust vector), then the rotor is considered trimmed. If not, the rotor is retrimmed using new values for rotor drag and the location of the resultant thrust vector. This process is repeated until the two values fall within the evaluation criteria.

As the first step, *trim.m* calculates various parameters used throughout the routine. The parameters include ambient air density (ρ_a), rotor drag or H-force (H_{rotor}), dynamic pressure (q), rotor disk area (A_{disk}), rotor blade tip speed (V_{tip}), fuselage drag ($D_{fuselage}$), lift and drag on the aerodynamic surfaces (D_{wing}, $D_{horiz\ tail}$, $D_{vert\ tail}$, L_{wing}, $L_{horiz\ tail}$, $L_{vert\ tail}$), the rotor tip path plane angle (α_{TPP}), and the advance ratio (μ). The calculation of rotor drag at this point is a first guess based on the relationship:

$$H_{rotor} = V_\infty \frac{\rho_a}{\rho_o} \qquad (1)$$

where:

V_∞ = forward airspeed

ρ_a = ambient air density

ρ_o = sea level density

10

The calculations for lift and drag on the wing and tail sections are based on a two dimensional analysis where:

$$C_D = C_{D_o} + \frac{C_L^2}{\pi e AR} \qquad (2)$$

$$D = q C_D S \qquad (3)$$

$$L = q C_L S \qquad (4)$$

where:

C_D = drag coefficient

C_L = lift coefficient

C_{Do} = profile drag coefficient

e = wing efficiency factor

AR = wing aspect ratio

q = dynamic pressure

S = wing area

The values for the horizontal and vertical tail efficiency factors are set at 0.8. The efficiency factor for the wing, as well as C_L, and C_{Do} for the wing and other surfaces are direct input parameters obtained from *janrad.m*. The drag coefficients, aspect ratios, and areas of the aerodynamic surfaces, as well as q are calculated from input parameters obtained from *janrad.m*. The rotor tip path plan angle is calculated as:

11

$$\alpha_{TPP} = \tan^{-1}\left(\frac{D_f + H_{rotor}}{GW - L_f}\right) \qquad (5)$$

where:

$D_f = D_{fuselage} + D_{wing} + D_{horiz\ tail} + D_{vert\ tail} - T_{aux}$

T_{aux} = auxiliary thrust

H_{rotor} = H-force (rotor drag)

GW = aircraft gross weight

$L_f = L_{wing} + L_{horiz\ tail}$

Finally, the advance ratio is calculated as:

$$\mu = \frac{V_\infty \cos(\alpha_{TPP})}{V_{tip}} \qquad (6)$$

where:

V_∞ = forward velocity

α_{TPP} = tip path plane angle

V_{tip} = rotor blade tip speed

The required thrust and corresponding coefficient of thrust are also

calculated. For hovering flight, thrust required is calculated as:

$$T = \left[1 + \left(0.3\,\frac{A_{vert\ proj}}{A_{disk}}\right)\right] GW \qquad (7)$$

where:

$A_{vert\ proj}$ = vertical projected area

A_{disk} = area of rotor disk

12

GW = aircraft gross weight

This equation assumes an effective vertical drag coefficient of 0.3 for all the aircraft components in the rotor wake [Ref. 5, p. 7]. In forward flight, the thrust required is calculated as:

$$T = \frac{GW - L_f}{\cos(\alpha_{TPP})} \qquad (8)$$

where:

GW = aircraft gross weight

$L_f = L_{wing} + L_{horiz\ tail}$

α_{TPP} = tip path plane angle

The coefficient of thrust is found by:

$$C_T = \frac{T}{A_{disk}\, \rho_a\, V_{tip}^2} \qquad (9)$$

where:

T = thrust

A_{disk} = rotor disk area

ρ_a = ambient air density

V_{tip} = rotor blade tip speed

Once C_T is calculated, the rotor blade tip loss factor is determined as:

$$B = 1 - \sqrt{\frac{2\,C_T}{b}} \qquad (10)$$

13

where:

C_T = coefficient of thrust

b = number of blades

Once the required parameters are calculated, *trim.m* sets up the rotor blade radial sections and rotor disk azimuth sectors. The radial sections are determined by evenly dividing the distance from the beginning of the aerodynamic portion of the rotor blade to the effective blade radius (R_{eff} = B*R) by the number of blade elements input by the user in *janrad.m*. Azimuth is determined by dividing the 360 degree rotor disk into equally spaced areas determined by the number of azimuth sectors input by the user.

The program then calculates rotor blade geometric angle due to twist (β_t) and rotor coning angle (β_o). Blade twist is considered linear and determined by the relationship:

$$\beta_t = \theta_1 \left(0.7 - \frac{r}{R} \right) \tag{11}$$

Where:

θ_1 = blade twist

r = distance from center of hub to center of blade element

R = rotor blade radius

Equation 11 takes this form because blade twist is converted to a positive value when input by the user in *janrad.m*. This allows the user to input either a positive or negative value of blade twist without affecting the accuracy of the program's output.

14

It would be highly unlikely that blade twist would ever have positive value yet some may refer to it as positive when it is actually negative. The rotor coning angle is determined by:

$$\beta_o = \sin^{-1}\left(\frac{\frac{T}{b}\bar{R}_T - \left(\frac{1}{2}(R-e)+e\right)w_{blade}}{\left(\frac{1}{2}(R-e)+e\right)^2 \Omega^2 m_{blade}}\right) \qquad (12)$$

where:

θ_1 = blade twist

r = distance from center of hub to center of blade element

T = thrust

b = number of blades

$\bar{R}_T = (R_{eff} - e)*r_T$

R = rotor blade radius

e = effective hinge offset

r_T = location of resultant thrust vector

R_{eff} = effective radius (B*R)

w_{blade} = weight of rotor blade

Ω = rotor rotational velocity

m_{blade} = mass of rotor blade

Equation 12 is based on a uniform rectangular blade, uses small angle approximations for β_o, and initially, assumes the location of the resultant thrust vector is at 0.7 r/R.

15

The next step is to determine the induced velocity distribution (v_i). At a hover, v_i is determined by equating differential thrust from momentum theory:

$$dT = 4 \rho_a \pi v_i^2 r \, dr \qquad (13)$$

and that of blade element theory:

$$dT = b \frac{\rho_a}{2} (\Omega r)^2 a \left(\theta - \frac{v_i}{\Omega r} \right) c \, dr \qquad (14)$$

which results in a quadratic equation of the form:

$$(4\pi) v_i^2 + \left(\frac{\Omega}{2} b a c \right) v_i - \left(\frac{\Omega^2}{2} r b a c \theta \right) = 0 \qquad (15)$$

where:

ρ_a = ambient air density

Ω = rotor rotational velocity

b = number of blades

a = airfoil lift curve slope

c = rotor blade chord

θ = blade pitch angle ($\beta_t + \theta_o$)

β_t = geometric angle due to twist

θ_o = collective pitch angle

r = distance from center of hub to center of blade element

dr = blade element width

The maximum root for v_i is then taken and used in equation 14 to determine the differential thrust. These calculations are repeated, incrementing θ appropriately until

16

the sum of the dT's equals the required thrust determined from equation 7. In forward flight, induced velocity is considered uniform and is calculated using the inflow parameter (λ_T) where:

$$\lambda_T = \mu \sin(\alpha_{TPP}) + \frac{\frac{1}{2}C_T}{\sqrt{\lambda_T^2 + \mu^2}} \qquad (16)$$

The maximum root for λ_T is taken and v_i is calculated as:

$$v_i = \lambda_T \Omega R - V_\infty \sin(\alpha_{TPP}) \qquad (17)$$

where:

μ = advance ratio

α_{TPP} = tip path plane angle

C_T = coefficient of thrust

Ω = rotor rotational velocity

R = rotor blade radius

V_∞ = forward airspeed

The last step before starting the rotor trim routine is to make a first guess at cyclic pitch (θ). Cyclic pitch is a first harmonic function with the cosine term (θ_{1c}) having a small positive value dependent on airspeed and the sine term (θ_{1s}), a negative value, also dependent on airspeed. The steady term (θ_o) or collective pitch term is an input parameter obtained from the user in *janrad.m*. The cyclic pitch is calculated as:

$$\theta = \theta_o + \theta_{1c} \cos(\psi) + \theta_{1s} \sin(\psi) \qquad (18)$$

where: ψ = azimuth angle

17

The rotor system is trimmed using blade element theory in a three step iterative process that consists of first adjusting collective pitch (θ_o), then trimming cyclic pitch (θ), and finally retrimming the collective. Blade element theory is based on the determination of local angle of attack (α) for each rotor blade element. Once α is calculated, the coefficient of lift (C_L) and coefficient of drag (C_D) for each element are determined and finally, thrust and drag for each element are calculated. Blade element theory involves the use of the following equations:

$$\alpha = \theta + \beta_t - \phi \tag{19}$$

$$\phi = \tan^{-1}\left(\frac{U_p}{U_t}\right) \tag{20}$$

$$U_p = v_i \cos(\beta_o) + V_\infty \sin(\alpha_{TPP}) \cos(\beta_o) + V_\infty \cos(\alpha_{TPP}) \sin(\beta_o) \cos(\psi) \tag{21}$$

$$U_t = \Omega r + V_\infty \cos(\alpha_{TPP}) \sin(\psi) \tag{22}$$

where:

θ = cyclic pitch (eqn. 18)

β_t = rotor blade geometric angle

ϕ = inflow angle

U_p = velocity perpendicular to rotor blade

U_t = velocity parallel to rotor blade

v_i = induced velocity

β_o = coning angle

18

V_∞ = forward airspeed

α_{TPP} = tip path plane angle

ψ = azimuth angle

r = distance from center of hub to blade element

The rotor trim procedure use two M-files, *thrcalc.m* and *tmcalc.m*. The listings for these M-files are found at Appendicies D and E respectively. The first M-file, *thrcalc.m*, uses blade element theory to calculate the differential thrust along the rotor blade at each azimuth angle. The differential thrust along the rotor blade is then summed to give the total thrust at each azimuth angle (T_ψ). Differential thrust is calculated as:

$$dT_\psi = \frac{1}{2}\rho_a c\, dr \left(U_p^2 + U_t^2\right) \left[C_L \cos(\phi) - C_D \sin(\phi)\right] \qquad (23)$$

where:

ρ_a = ambient air density

c = rotor blade chord

dr = blade element width

U_p = velocity perpendicular to rotor blade

U_t = velocity parallel to rotor blade

C_L = coefficient of lift

C_D = coefficient of drag

ϕ = inflow angle

The small contribution of thrust provided by profile drag in the tip loss region of the

blade is accounted for by giving the profile drag coefficient (C_{Do}) a nominal value of .009. The equation for differential thrust in the tip loss region is as follows:

$$dT_{tip} = \frac{1}{2} \rho_a c (R - R_{eff}) \left(U_{P_{tip}}^2 + U_{t_{tip}}^2 \right) (-.009 \sin(\phi_{tip}))$$ (24)

where:

ρ_a = ambient air density

c = rotor blade chord

R = rotor blade radius

R_{eff} = effective rotor radius

$U_{p\ tip}$ = velocity perpendicular to rotor blade

$U_{t\ tip}$ = velocity parallel to rotor blade

ϕ = inflow angle

The second M-file, *tmcalc.m*, also uses blade element theory to calculate the differential thrust moments along the blade at each azimuth angle. The differential thrust moments are then summed to give the total thrust moment at each azimuth angle (M_ψ). The differential thrust moment is calculated as:

$$dM_\psi = \frac{1}{2} \rho_a c\, r\, dr \left(U_P^2 + U_t^2 \right) [C_L \cos(\phi) - C_D \sin(\phi)]$$ (25)

where:

ρ_a = ambient air density

c = rotor blade chord

r = distance from center of hub to blade element

20

dr = blade element width

U_p = velocity perpendicular to rotor blade

U_t = velocity parallel to rotor blade

C_L = coefficient of lift

C_D = coefficient of drag

ϕ = inflow angle

The small thrust moment provided by profile drag in the tip loss region of the blade is accounted for in a similar manner as the thrust in *thrcalc.m*. The equation is:

$$dM_{tip} = \frac{1}{2} \rho_a c \left(R - \frac{(R-R_{eff})}{2} \right) (R-R_{eff}) \left(U_{p_{tip}}^2 + U_{t_{tip}}^2 \right) (-.009 \sin(\phi_{tip})) \tag{26}$$

where:

ρ_a = ambient air density

c = rotor blade chord

R = rotor blade radius

R_{eff} = effective rotor radius

$U_{p\ tip}$ = velocity perpendicular to rotor blade

$U_{t\ tip}$ = velocity parallel to rotor blade

ϕ = inflow angle

The two M-files, *thrcalc.m* and *tmcalc.m*, call an M-file which contains the lift coefficient and the drag coefficient data for the appropriate rotor blade airfoil. The airfoil data are arranged as separate polynomials for various ranges of angle of attack (α). There are currently two airfoil data function files, *hh02clcd.m* and

21

vr12clcd.m. The former contains data for the McDonnell Douglas Helicopter Corporation (MDHC) HH-02 airfoil used on the AH-64A Apache attack helicopter. The latter contains data for the Boeing VR-12 airfoil planned for use on the RAH-66 Comanche. There is no Mach number dependency incorporated in either airfoil data function files. The listings for these M-files are found in Appendicies F and G respectively.

In both the initial trim and retrim portions of the collective trim procedure, the collective is adjusted by increasing or decreasing θ_o a suitable amount to match the required thrust calculated from either equation 7 or 8. The cyclic terms (θ_{1c} and θ_{1s}) are not adjusted during the collective trim procedure. Collective pitch is adjusted until the resulting thrust calculated by *thrcalc.m* is within 2% of required thrust for the initial trim procedure and 1% of required thrust for the retrim procedure.

The cyclic trim routine begins by determining the initial thrust moment ($M_{(int)}$) based on the cyclic pitch determined by the collective trim procedure. This cyclic pitch value is considered the initial pitch ($\theta_{(int)}$). The thrust moment is calculated using the M-file *tmcalc.m*. An initial partial derivative of cyclic pitch with respect to thrust moment is determined by collectively increasing cyclic pitch a very small amount and determining the corresponding thrust moment. Giving the new cyclic pitch and corresponding new thrust moment the labels $\theta_{(new)}$ and $M_{(new)}$ respectively, the derivative is determined as:

$$\frac{\partial \theta}{\partial M} = \frac{\theta_{(new)} - \theta_{(int)}}{M_{(new)} - M_{(int)}} \tag{27}$$

Next, the first harmonic terms are removed from the original thrust moment ($M_{(int)}$). Using a numerical Fourier analysis, the first harmonic terms of the thrust moment are:

$$M_{1c} = \frac{2}{n} \sum_{i=1}^{n} M_{(new)} \cos(\psi) \tag{28}$$

$$M_{1s} = \frac{2}{n} \sum_{i=1}^{n} M_{(new)} \sin(\psi) \tag{29}$$

where:

n = number of azimuth sectors

M_ψ = total thrust moment at each azimuth angle

ψ = azimuth angle

The new thrust moment ($M_{(new)}$) is then calculated as follows:

$$M_{(new)} = M_{(int)} - M_{1c} \cos(\psi) - M_{1s} \sin(\psi) \tag{30}$$

The change in thrust moment is then calculated as;

$$\Delta M = M_{(new)} - M_{(int)} \tag{31}$$

and the new cyclic pitch ($\theta_{(new)}$) is then determined by the relationship:

$$\theta_{(new)} = \theta_{(int)} + \frac{\partial \theta}{\partial M} \Delta M \tag{32}$$

A first harmonic function is then enforced on theta by using equation 18 where:

$$\theta_{1c} = \frac{2}{n} \sum_{i=1}^{n} \theta_{(new)} \cos(\psi) \qquad (33)$$

$$\theta_{1s} = \frac{2}{n} \sum_{i=1}^{n} \theta_{(new)} \sin(\psi) \qquad (34)$$

Using the new cyclic pitch from equation 32, a new thrust moment is calculated, and subsequently a new change in moment is determined. The previous values for cyclic pitch and thrust moment become the initial values, and the process is repeated (skipping the small collective increase used for the initial determination of the derivative) until there is only a small first harmonic (2% of the total thrust moment) left in the thrust moment.

At anytime during the collective or cyclic trimming routines, if the values of the respective parameters (difference in thrust or difference in thrust moment) begin to diverge, program execution will halt. Divergence of these parameters indicate that the blade is stalling over too large of an azimuth region. At this point, the user is told to use a slower airspeed or redesign the system.

Once the cyclic is trimmed and the collective retrimmed, *trim.m*, using *dmcalc.m*, then calculates the differential drag values for each azimuth angle (dD_ψ) and the rotor drag moments for each azimuth angle (DM_ψ). A listing for *dmcalc.m* is found at Appendix H. The drag moments are determined by summing the product of the differential drag values and radial distance from the center of the hub to the center of each differential blade element. The differential drag values are calculated as follows:

24

$$dD_{\psi} = \frac{1}{2}\rho_a c \, dr \left(U_P^2 + U_t^2 \right) \left[C_L \sin(\phi) + C_D \cos(\phi) \right] \qquad (35)$$

where:

ρ_a = ambient air density

c = rotor blade chord

dr = blade element width

U_p = velocity perpendicular to rotor blade

U_t = velocity parallel to rotor blade

C_L = coefficient of lift

C_D = coefficient of drag

ϕ = inflow angle

The contribution of profile drag in the tip loss region is accounted for in the drag and drag moment equations in a similar manner as in *thrcalc.m*. The equation for the differential drag in the tip loss region is:

$$dD_{tip} = \frac{1}{2}\rho_a c \, (R - R_{eff}) \left(U_{P_{tip}}{}^2 + U_{t_{tip}}{}^2 \right) (.009 \cos(\phi_{tip})) \qquad (36)$$

where:

ρ_a = ambient air density

c = rotor blade chord

R = rotor blade radius

R_{eff} = effective rotor blade radius

$U_{p\,tip}$ = velocity perpendicular to rotor blade

25

$U_{t\,tip}$ = velocity parallel to rotor blade

ϕ_{tip} = inflow angle

The equation for the drag moment in the tip loss region is:

$$dDM_{tip} = dD_{tip}\left(R - \frac{(R-R_{eff})}{2}\right) \qquad (37)$$

Once the differential drag values are determined, the H-force (rotor drag) and the location of the resultant thrust vector are then calculated. The H-force is a first harmonic function and is calculated as follows:

$$H_{rotor} = \frac{b\cos(\alpha_{TPP})}{2}\left(\sum_{r_o}^{R} H_{1s} - \sin(\beta_o)\sum_{r_o}^{R} H_{1c}\right) \qquad (38)$$

where:

b = number of blades

α_{TPP} = tip path plan angle

r_o = start of aerodynamic portion of blade

R = radius of rotor blade

β_o = coning angle

The cosine and sine terms of the rotor H-force are determined as:

$$H_{1c} = \frac{2}{n}\sum_{i=1}^{n} dT_\psi \cos(\psi) \qquad (39)$$

$$H_{1s} = \frac{2}{n}\sum_{i=1}^{n} dD_\psi \sin(\psi) \qquad (40)$$

If the difference between the initial rotor H-force and the computed value is more

26

than 20% of the initial one, then the location of the resultant thrust vector is computed and the rotor trim process is repeated. If the difference is less than 20%, then the location of the resultant thrust vector is computed and the rotor trim process is repeated only if the new location of the resultant thrust vector does not fall within its limits.

The location of the resultant thrust vector (r_T) is calculated as follows:

$$r_T = \frac{\overline{M_\downarrow}}{\overline{T_\downarrow}}$$

(41)

where:

$\overline{M_\downarrow}$ = mean of the sum of thrust moments

$\overline{T_\downarrow}$ = mean of the sum of thrust

If the difference between the initial r_T and the computed one is more than 1.5% of the original, then the rotor trim process is repeated. If the difference is less than 1.5%, then the rotor is considered trimmed.

3. Perf.m

This M-file calls the rotor trim routine, calculates various rotor performance parameters, and then displays and saves the rotor performance data. After calling *trim.m*, the routine calculates rotor power required and the resulting rotor torque using the relationships:

$$P_{rotor}(h.p.) = \frac{\frac{\Sigma DM_\downarrow}{n} b \Omega}{550}$$

(42)

27

$$Q_{rotor} = \frac{\sum DM_\psi}{n} b \qquad (43)$$

Where:

DM_ψ = total drag moment at each azimuth angle

n = number azimuth sectors

b = number of blades

Ω = rotor rotational velocity

The routine then calculates rotor solidity (σ), coefficient of thrust over solidity (C_T/σ), coefficient of torque over solidity (C_Q/σ), coefficient of H-force over solidity (C_H/σ), disk loading (D.L.), figure of merit (F.M.), and advancing blade tip Mach number (M_{tip}). Solidity is calculated as:

$$\sigma = \frac{bc}{\pi R} \qquad (44)$$

where:

b = number of blades

c = blade chord

R = blade radius

C_T was previously calculated by *trim.m* and C_Q and C_H are calculated as:

$$C_Q = \frac{Q_{rotor}}{A_{disk} \, \rho_a \, V_{tip}^2 \, R} \tag{45}$$

$$C_H = \frac{H_{rotor}}{A_{disk} \, \rho_a \, V_{tip}^2} \tag{46}$$

where:

Q_{rotor} = rotor torque

A_{disk} = area of rotor disk

ρ_a = ambient air density

V_{tip} = blade tip speed

H_{rotor} = rotor H-force

C_T, C_Q, and C_H are then divided by σ to get the C_x/σ relationships. Disk loading and figure of merit are determined by:

$$D.L. = \frac{T_{rotor}}{\pi R^2} \tag{47}$$

$$F.M. = \frac{T_{rotor} \sqrt{\dfrac{D.L.}{2 \rho_a}}}{550 \, (P_{rotor})} \tag{48}$$

where:

T_{rotor} = rotor thrust

R = rotor radius

ρ_a = ambient air density

P_{rotor} = rotor power (h.p.)

Disk loading and figure of merit are set to zero for hovering flight. The advancing

29

tip Mach number is calculated using the relationship:

$$M_{tip} = \frac{V_{tip}\cos(\alpha_{TPP}) + V_\infty}{49.05\sqrt{T_a + 460}} \qquad (49)$$

where:

V_{tip} = rotor blade tip speed

α_{TPP} = tip path plane angle

V_∞ = airspeed

T_a = ambient temperature ($^\circ$F)

Once the rotor performance parameters are calculated, *perf.m* then asks the user if they want the rotor performance parameters displayed on the screen. If the answer is yes, the data is presented as two screens of input data and two of calculated data with a user controlled pause between screens.

Finally, *perf.m* saves the rotor performance data to disk as two separate files. The first file contains the single valued input and calculated parameters and is given the file name the user defined in *janrad.m* with a ".prf" extension. This file is a MS-DOS® text file. The file is formed using the MATLAB® *DIARY* function which records a series of MATLAB® *FPRINTF* commands and then stores them to disk as a file named *diary*. A call to the operating system is then made renaming *diary* to the appropriate ".prf" file name. The second file contains the vector and matrix valued data and is given the user defined file name concatenated with an "_p" and a ".mat" extension. This file is a binary file which the user can access using the MATLAB® *LOAD* command. The file is created using the MATLAB® *SAVE* command.

30

4. Stab.m

The M-file, *stab.m*, performs the helicopter stability and control calculations, formats the stability and control output, and saves that output to disk. This module uses the following M-files: 1) *cbodygrp.m*; 2) *cmdbwplc.m*; 3) *cmdbwplh.m*; 4) *cmrgrp.m*; 5) *cruise.m*; 6) *ctrgrp.m*; 7) *dctplots.m*; 8) *dctplott.m*; 9) *deriv2.m*; 10) *hmrgrp.m*; 11) *hover.m*; 12) *htrgrp.m*; and 13) *stabout.m*. For details on *stab.m* and the various routines, See reference 2.

III. USER INSTRUCTIONS

A. GENERAL

Program installation, guidance for input requirements, and rotor performance output are discussed in the subsequent paragraphs. These instructions assume that the reader has a working knowledge of the MS-DOS* operating system and of 386-MATLAB* version 3.5.

B. INSTALLATION

To install JANRAD:

1. run MATLAB*

2. ensure you are in your working directory

3. from the MATLAB* command prompt, type »!*drive*: (A: or B:) then type »INSTALL. The M-file, *install.m*, will create a subdirectory named *JANRAD*. It will then copy the JANRAD files to that directory. Once the files are copied, the current directory is changed to *JANRAD* and the program is ready to run.

C. PROGRAM EXECUTION

1. Input

To run JANRAD, at the MATLAB* command prompt, type »JANRAD. At the initial screen, enter a 1 if you want to edit a previously stored data file, otherwise enter a 2. If you chose to edit a file, you will be prompted to enter the file name without the extension. If JANRAD cannot find the file you entered in the current directory, it will allow you to try again. Check your spelling or insure you

are in the *JANRAD* directory. Once the file is loaded, the edit menu, as seen in

Figure 1, will be displayed.

```
                        *** EDIT MENU ***

         1. pressure altitude      2. temperature
         3. airspeed               4. gross weight
         5. number of blades       6. blade radius
         7. blade chord            8. hinge offset
         9. blade grip length     10. blade twist
        11. blade weight          12. # blade elements
        13. rotational velocity   14. # azimuth sectors
        15. lift curve slope      16. airfoil
        17. collective pitch      18. flat plate area
        19. vert projected area   20. wing area
        21. wing span             22. wing CL
        23. wing CDo              24. wing efficiency factor
        25. horizontal tail area  26. horizontal tail span
        27. horizontal tail CL    28. horizontal tail CDo
        29. vertical tail area    30. vertical tail span
        31. vertical tail CL      32. vertical tail CDo
        33. auxiliary thrust

         0. NO CHANGES

        Input the parameter to change:
```

Figure 1. Edit Menu

Select the number of the parameter you wish to change. The current value for that

parameter will be displayed followed by a prompt to enter the new value. If you

choose not to change the value, simply press < Enter > and the previous value will be

maintained. Once a value is entered, or < Enter > is pressed, the editing menu will

again be displayed. Choose the number of the next parameter to change, or a 0 to

exit the editing menu. If you chose to create a new data file, prompts for individual

parameters will be displayed. Enter the value for each parameter when prompted.

Enter values for the specified parameters regardless of editing or creating a

data file in the following manner:

1. PA - Enter the pressure altitude in feet.

33

2. temperature - Enter the temperature in degrees fahrenheit.

3. airspeed - Enter forward airspeed in knots. JANRAD automatically converts airspeed to feet/second.

4. GW - Enter aircraft gross weight in pounds.

5. number of blades - Enter the number of blades used in the rotor system.

6. radius - Enter the rotor blade radius in feet. Measure the radius from the center of the rotor hub to the tip of the blade.

7. chord - Enter the average rotor blade chord in feet.

8. hinge offset - Enter the effective hinge offset in feet.

9. blade grip - Enter the distance from the center of the rotor hub to the beginning of the aerodynamic portion of the rotor blade in feet.

10. blade twist - Enter rotor blade twist in degrees. You can enter blade twist as either a positive or negative value. JANRAD uses the absolute value of blade twist.

11. blade weight - Enter the weight of a single rotor blade in pounds.

12. rotor blade elements - Enter the number of rotor blade radial elements in feet. JANRAD adds one element to account for tip loss.

13. rotational velocity - Enter the rotor rotational velocity in radians/second.

14. azimuth sectors - Enter the number of rotor disk azimuth sectors.

15. lift curve slope - Enter the average slope of the linear portion of the rotor blade airfoil's lift curve, where the lift curve is C_L versus alpha.

16. rotor blade airfoil - Enter a 1 if you are using an HH-02 airfoil. Enter a 2

if you are using a VR-12 airfoil. The HH-02 airfoil data is very close to the NACA 0012 airfoil data. Therefore, enter a 1 for a generic airfoil.

17. collective pitch - Enter the estimated collective pitch at 0.7 r/R in degrees. JANRAD automatically converts the value to radians.

18. flat plate area - Enter the aircraft equivalent flat plate area in square feet.

19. vertical projected area - Enter the vertical projected area in square feet.

20. wing area - Enter the wing area in square feet. Enter a 0 if the aircraft has no wing.

21. wing span - Enter the wing span in feet. If you included the segment of the fuselage encompassed by the wing in the wing area value, include it in the span value also.

22. wing C_L - Enter the expected lift coefficient for the wing.

23. wing C_{Do} - Enter the wing profile drag coefficient.

24. wing efficiency factor - Enter the wing efficiency factor.

25. horizontal tail area - Enter the horizontal tail area in square feet.

26. horizontal tail span - Enter the horizontal tail span in feet.

27. horizontal tail C_L - Enter the expected lift coefficient for the horizontal tail.

28. horizontal tail C_{Do} - Enter the horizontal tail profile drag coefficient.

29. vertical tail area - Enter the area of the vertical tail in square feet.

30. vertical tail span - Enter the span of the vertical tail in feet.

31. vertical tail C_L - Enter the expected lift coefficient for the vertical tail.

32. vertical tail C_{Do} - Enter the vertical tail profile drag coefficient.

33. auxiliary thrust - Enter the expected auxiliary thrust in pounds.

Once the edited or new values are entered, you will be prompted to save the data under a designated file name. The file name must use a combination of six letters, numbers, or the underscore. If more than six characters are entered, you will be prompted to try again. JANRAD will save the file to the current directory (*JANRAD*) as the file name with a ".mat" extension. If you were editing a data file, press <Enter> if you want to save the file with the original file name.

The execution menu is displayed next. Enter a 1 if you want to perform a rotor performance analysis. Enter a 2 if you choose to perform a stability and control analysis. Enter a 3 to execute the rotor dynamics analysis. The rotor dynamics module is currently under development. If you enter a 3, this fact will be displayed and you will be asked to enter another selection. If you decide to edit your data further or you want to create a new file, enter a 4. To quit JANRAD, enter a 5.

From any of the selection menus, if you enter a character other than what is requested, JANRAD will request that you enter an appropriate response. If you enter a letter or special ASCII character, the MATLAB* prompt "undefined variable or function" will be displayed along with a repeat of the initial prompt. If you enter a number other than requested, the initial prompt will be repeated.

No further input is required if you select rotor performance. See Reference 2 for details on the input required if you select stability and control. Details for the rotor dynamics input requirements should be covered in a future thesis.

36

2. Output

If rotor performance is chosen from the execution menu, you will be asked if you want the rotor performance results displayed on the screen. If you respond in the affirmative, four screens will be displayed. The screens will look similar to those shown in Figures 2 through 5. The data in these figures is from the example helicopter (Prouty's helicopter) found in Reference 5, page 669. The first two screens contain the input data and the second two contain the output data.

```
                    *** INPUT DATA ***
                          prouty

         Forward velocity =      110 kts
              Temperature =       59 degs F
        Pressure altitude =        0 ft
             Gross weight =    20000 lbs
          Number of blades =        4
             Rotor radius =    30.00 ft
              Blade chord =     2.00 ft
              Blade twist =   -10.00 degs
   Blade lift curve slope =     5.73
             Blade weight =   240.00 lbs
       Rotational velocity =    21.67 rads/sec
        Blade grip length =     2.25 ft
             Hinge offset =     1.50 ft

press any key to continue...
```

Figure 2. First Output Screen (Input Data)

```
                    *** INPUT DATA CONTINUED ***
                              prouty

     Equivalent flat plate area =   30.00 ft^2
         Vertical projected area = 300.00 ft^2
                      Wing area =    0.00 ft^2
                      Wing span =    0.00 ft
                       Wing CL =     0.00
                       Wing CDo =  0.0000
        Wing efficiency factor =     0.00
          Horizontal tail area =   18.00 ft^2
          Horizontal tail span =    9.00 ft
            Horizontal tail CL =    -0.30
          Horizontal tail CDo =   0.0045
            Vertical tail area =   33.00 ft^2
            Vertical tail span =    7.70 ft
              Vertical tail CL =     0.60
             Vertical tail CDo =   0.0045
             Auxiliary thrust =   0 lbs

    press any key to continue...
```

Figure 3. Second Output Screen (Input Data Continued)

```
                      *** CALCULATED DATA ***
                              prouty

                 Fuselage drag =   1232 lbs
                    Rotor drag =    235 lbs
                     Wing lift =      0 lbs
                     Wing drag =      0 lbs
          Horizontal tail lift =   -222 lbs
          Horizontal tail drag =      9 lbs
     Vertical tail side force =    813 lbs
          Vertical tail drag =    114 lbs

                Tip path angle =   4.48 degs
            Rotor coning angle =   5.69 degs
     Location of mean thrust (r/R) =  0.63
       collective pitch at .7 r/R =   6.98 degs
     1st lat cyclic term-A1 (deg) =   2.07
    1st long cyclic term-B1 (deg) =  -4.32

    press any key to continue...
```

Figure 4. Third Output Screen (Calculated Data)

38

```
                *** CALCULATED DATA CONTINUED ***
                            prouty

                        solidity  =  0.085
                     Disk loading =    0.00  lbs/ft^2
                  Figure of Merit =    0.00
                         CT/sigma =   0.084
                         CQ/sigma =  0.0034
                         CH/sigma =  0.0010
      Tip mach of the adv. blade =   0.746
                    Advance ratio =   0.285
           Rotor thrust required (TPP) =  20284  lbs
                Rotor power required =    959  h.p.
                      Rotor torque =  24347  ft-lbs
```

Figure 5. Fourth Output Screen (Calculated Data Continued)

Each of these screens is displayed only one time. Once you press any key to go to the next screen, you cannot return to the previous one. A copy of the four screens is saved to the current (*JANRAD*) directory with the file name you entered at the "save instructions" screen and concatenated with a ".prf" extension.

The vector and matrix data generated by the rotor performance module of JANRAD is saved to the current (*JANRAD*) directory with the file name you entered at the "save instructions" screen and concatenated with a "_p" and a ".mat" extension. The following vector and matrix parameters are saved:

1. r - a vector containing the radii (in feet) to the center of each blade element. This is a row vector containing n elements, where n = # of blade elements + 1.

2. ψ = a vector containing the azimuth angles (in degrees) around the rotor disk. This is a column vector containing n elements, where n = # of azimuth sectors.

3. v_i = a vector containing the induced velocity distribution (in ft/sec) along the rotor blade. This is a row vector containing n elements, where n = # of blade

39

elements + 1.

4. θ = a vector containing cyclic pitch (in degrees) with respect to azimuth angle at 0.7 r/R. This is a column vector containing n elements, where n = # of azimuth sectors.

5. β_t = a vector containing blade pitch (in degrees) along the rotor blade. This is a row vector containing n elements, where n = # of blade elements + 1.

6. α = a matrix containing the angle of attack (in degrees) distribution. This is an m×n matrix, where m = # of azimuth sectors and n = # of blade elements +1.

7. T_ψ = a vector containing the total thrust (in pounds) at each azimuth sector. This is a column vector containing n elements, where n = # of azimuth sectors.

8. M_ψ = a vector containing the total thrust moment (in ft-lbs) at each azimuth sector. This is a column vector containing n elements, where n = # of azimuth sectors.

9. DM_ψ = a vector containing the total drag moment (in ft-lbs) at each azimuth sector. This is a column vector containing n elements, where n = # of azimuth sectors.

10. dT = a matrix containing the differential thrust (in pounds) distribution. This is an m×n matrix, where m = # of azimuth sectors and n = # of blade elements +1.

11. dM = a matrix containing the differential thrust moment (in ft-lbs) distribution. This is an m×n matrix, where m = # of azimuth sectors and n = # of blade elements +1.

40

12. dD = a matrix containing the differential drag (in pounds) distribution. This is an m×n matrix, where m = # of azimuth sectors and n = # of blade elements +1.

A final rotor performance module screen is displayed instructing you on how to access your rotor performance data. Use the MS-DOS° *TYPE* command or a DOS editor to view or print the single variable data. The vector and matrix data can be loaded into memory using the MATLAB° *LOAD* command. Some example plots, using vector and matrix data from the Prouty helicopter, are seen below. Figure 6 is a plot of hover induced velocity (ft/sec) versus fraction of blade radius (r/R).
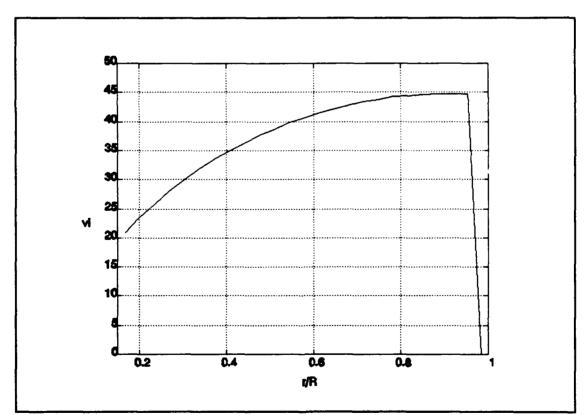

Figure 6. Hover Induced Velocity vs. Blade Radius

A plot of total thrust moment (ft-lbs) versus azimuth angle (degrees) for the Prouty

helicopter at 125 knots is shown in Figure 7.

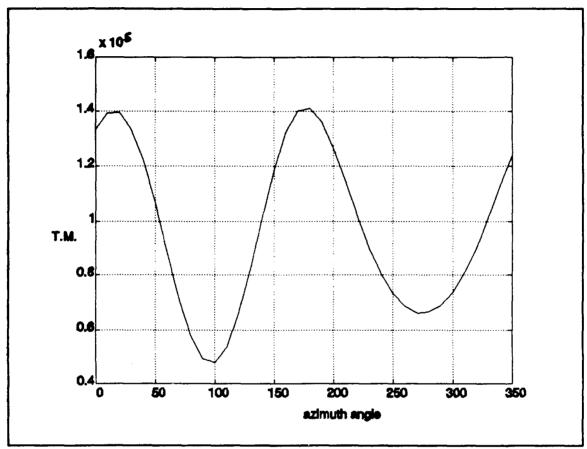**Figure 7. Total Thrust Moment vs. Azimuth Angle**

A three dimensional plot showing the differential thrust (lbs) distribution over blade

radius versus azimuth angle is at Figure 8. Figure 9 shows the angle of attack

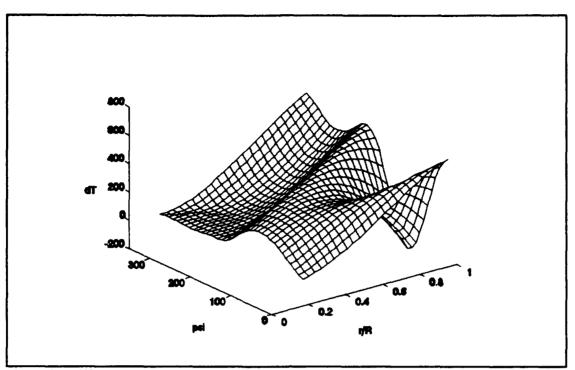(degrees) distribution. The data for both plots is from the Prouty helicopter at 125

knots.
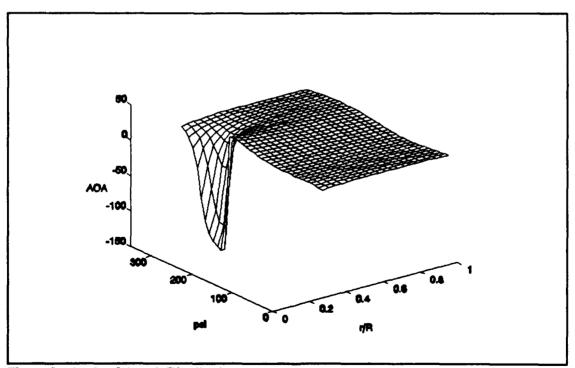
Figure 8. Differential Thrust Distribution

Figure 9. Angle of Attack Distribution

43

The plots in Figures 6 through 9 were generated from the data contained in a data file named *prouty_p.mat*. The data was plotted using MATLAB*

for WINDOWS*, version 4.0.

The preceding paragraphs discussed the details for the rotor performance output, the details for stability and control output are contained in Reference 2. The details for the blade dynamics module will be found in a future thesis.

D. PROGRAM UPDATES

JANRAD is written entirely in the 386-MATLAB* version 3.5 programming language. This version of JANRAD is designated version 1.0. The text in this document corresponds to the software loaded on a master 3.5" diskette labeled *JANRAD version 1.0, September 1993*. Subsequent changes to the software should be given a new numerical designation and loaded on a separate diskette. Each new version of MATLAB* utilizes a slightly different command set. As a result, slight modifications may be required to run JANRAD version 1.0 on later versions of MATLAB*. Additionally, JANRAD version 1.0 was developed for use on computers using the DOS type operating system. JANRAD makes some calls to the operating system using the MATLAB* "!" function. For this reason and because actual MATLAB* commands differ slightly, modifications would be required to run JANRAD version 1.0 on an operating system other than (i.e., UNIX*). The ".mat" data files created by JANRAD version 1.0 can be transferred to other operating systems using a file transfer program. Those files could then be loaded by the particular operating system's version of MATLAB* and used for plotting.

44

IV. RECOMMENDATIONS

Improvements to this program code can be made in several areas. First, the blade dynamics module should be developed as a follow-on thesis or series of AE 3402 (Helicopter Dynamics) class projects. As another thesis project, the program code should be rewritten to work with the UNIX* version of MATLAB*. The UNIX* version runs faster and has enhanced graphic capabilities. The program code should also be updated to include:

1. The capability to offset the center of gravity from the center of the rotor hub during the rotor trimming process.

2. A method of incorporating Mach number dependency in the rotor blade airfoil data. This may require the use of "look-up" tables versus the current use of parametric equations.

3. A more accurate method for determining rotor induced velocity in forward flight. This would be especially beneficial for airspeeds less than 100 knots.

4. Changes to the code to account for the effects of rotor downwash on the wing angle of attack.

5. Changes to the code to incorporate a non-rectangular, non-uniform, flexible rotor blade.

These updates to the code would expand the design students analysis capabilities as well as improve the overall accuracy of the program. This would be especially true when utilizing the enhanced capabilities of the UNIX* version of MATLAB*.

It is also recommended that this version of JANRAD (version 1.0) be kept on a master diskette and that future updates be stored on a separate disk. This will ensure that a working copy of the program is always available. Thesis level updates to JANRAD should, at a minimum, refer to this document. Updates that are not large enough to justify thesis work should be documented as attachments to this document. It is recommended that all update documentation be maintained (chronologically) in a binder along with this document.

REFERENCES

1. MATLAB*, *386-MATLAB* User's Manual* , The Math Works, Inc., 1991.

2. Wirth, W. M., *Linear Modelling of Rotorcraft for Stability and Control Analysis and Preliminary Design*, September, 1993.

3. Gerstenberger, W. and Wood, E. R., *Analysis of Helicopter Aeroelastic Characteristics in High-Speed Flight*, AIAA Journal, 1963.

4. Wheatley, J. B., *An aerodynamic analysis of the autogiro rotor with a comparison between calculated and experimental results*, NACA Report 487, 1934.

5. Prouty, R. W., *Helicopter Performance, Stability, and Control*, Reprint Edition, Robert E. Krieger, Inc., 1990.

APPENDIX A: JANRAD.M

%   Joint Army Navy Rotorcraft Analysis and Design
%   (JANRAD)
%   Version 1.0
%   September 1993

%   MAJ Bob Nicholson
%   MAJ Walter Wirth

% This program is an interactive preliminary design tool
% developed to aid the design student in determination of
% initial rotorcraft configurations and in the calculation
% of performance, stability and control, and other parameters.
% The program will work for conventional or compound rotorcraft.
% It will provide accurate data for airspeeds less than 10
% knots and greater than or equal to 50 knots.

% Variable List for janrad.m, trim.m, thrcalc.m, tmcalc.m, dmcalc.m,
% hh02clcd.m, vr12clcd.m, and perf.m

%		
%	a	lift curve slope of rotor system airfoil
%	Adisk	area of rotor disk
%	Afh	fuselage equivalent flat plate drag area
%	Afv	vertical projected area (fuselage area under disk)
%	airfoil	rotor system airfoil type (HH02/VR12)
%	alpha	angle of attack, rotor blade radial segment
%	alphaT	rotor tip path plane angle
%	b	number of rotor blades
%	B	tip loss parameter
%	betao	rotor coning angle
%	betat	geometric angle, rotor blade radial segment
%	bhoriz	span, horizontal tail
%	bvert	span, vertical tail
%	bwing	span, wing
%	cblade	chord, rotor blade
%	CD	drag coefficient, rotor blade radial segment
%	CDohoriz	profile drag coefficient, horizontal tail
%	CDovert	profile drag coefficient, vertical tail
%	CDowing	profile drag coefficient, wing
%	CDhoriz	drag coefficient, horizontal tail
%	CDvert	drag coefficient, vertical tail
%	CDwing	drag coefficient, wing
%	CH	rotor H-force coefficient
%	CH_sig	CH/solidity
%	CL	lift coefficient, rotor blade radial segment
%	CLhoriz	lift coefficient, horizontal tail
%	CLvert	lift coefficient, vertical tail
%	CLwing	lift coefficient, wing
%	CQ	rotor torque coefficient
%	CQ_sig	CQ/solidity
%	CT	rotor thrust coefficient
%	CT_sig	CT/solidity
%	dD	differential drag, rotor blade radial segment
%	ddD	differential drag, rotor blade tip
%	ddDM	differential drag moment, rotor blade tip
%	ddM	differential thrust moment, rotor blade tip
%	ddT	differential thrust, rotor blade tip
%	delM	change in total thrust moment

%	Dftotal	resultant of fuselage drag and aux thrust
%	Dfuse	total drag generated by non-rotor bodies
%	DL	disk loading
%	dM	differential thrust moment, rotor blade radial seg
%	DMpsi	total blade drag moment at specific azimuth angle
%	dr	rotor blade radial segment width
%	Drotor	rotor system drag
%	dT	differential thrust, rotor blade radial segment
%	Dhoriz	drag, horizontal tail
%	dthetadM	change in cyclic pitch with change in thrust moment
%	Dvert	drag, vertical tail
%	Dwing	drag, wing
%	e	effective hinge offset
%	ewing	wing efficiency factor
%	filename	name of input file
%	FM	figure of merit
%	grip	length of inner non-aerodynamic portion of blade
%	GW	aircraft gross weight
%	Hrotor	rotor H-force
%	lamdaT	forward flight induced velocity parameter
%	Lftotal	total lift generated by non-rotor bodies
%	Lhoriz	lift, horizontal tail
%	Lvert	lift, vertical tail
%	Lwing	lift, wing
%	M1c	first harmonic (cosine) thrust moment coefficient
%	M1s	first harmonic (sine) thrust moment coefficient
%	Machtip	Mach number at rotor blade tip
%	mblade	mass of rotor blade
%	Mpsi	total blade thrust moment at specific azimuth angle
%	mu	advance ratio
%	naz	number of azimuth sectors
%	nbe	number of blade elements
%	omega	rotor rotational velocity
%	PA	pressure altitude
%	phi	inflow angle, rotor blade radial segment
%	phitip	inflow angle, rotor blade tip
%	Protor	power required by rotor
%	psi	azimuth angle
%	q	dynamic pressure
%	Qrotor	rotor torque
%	r	radius, rotor blade radial segment
%	R	rotor blade radius
%	Rbar	Reff-e
%	RbarT	rT*Rbar
%	Reff	effective rotor blade radius (tip loss)
%	rho	ambient air density
%	rT	location of resultant thrust vector
%	solidity	solidity
%	Shoriz	area, horizontal tail
%	Svert	area, vertical tail
%	Swing	area, wing
%	T	rotor thrust
%	Taux	auxiliary thrust
%	temp	ambient air temperature
%	theta	cyclic pitch
%	theta1c	first harmonic (cosine) of cyclic pitch
%	theta1s	first harmonic (sine) of cyclic pitch
%	thetao	collective pitch at .7 r/R
%	Tpsi	total blade thrust at specific azimuth angle
%	twist	geometric rotor blade twist
%	Up	vertical component of velocity
%	Uptip	vertical component of velocity at tip

49

```
%    Ut          horizontal component of velocity
%    Uttip       horizontal component of velocity at tip
%    vi          induced velocity
%    Vinf        forward airspeed
%    Vtip        tip speed
%    wblade      weight of rotor blade

clc
disp(' ')
disp(' ')
disp('      ***********************************************************')
disp('      *                                                         *')
disp('      *   *** Joint Army/Navy Rotorcraft Analysis and Design ***   *')
disp('      *                      (JANRAD)                           *')
disp('      *                                                         *')
disp('      *                   Version 1.0                           *')
disp('      *                   September 1993                        *')
disp('      *                                                         *')
disp('      ***********************************************************')
disp(' ')

check1=1;
while check1 > 0

% clearing all variables in the MATLAB environment

clear
check1=1;

disp(' ')
disp('        Do you want to edit an existing file or create a new one?')
check=1;
while check > 0
  answer=input('      1. edit existing file   2. create new file   >> ');


%    *** 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 answer==1,
    clc
disp(' ')
disp(' ')
disp('                    *** LOAD INSTRUCTIONS *** ')
disp(' ')
disp('           A. Input the name of the file to edit.')
disp('           B. The file was saved in your previous session')
disp('              with a ".mat" extension.')
disp('           C. Do not include the extension or quotations.')
disp(' ')
disp('           ex: dsgn1')
    flag=0;
    while flag < 1
      filename1=input('          name of input file: ','s');
      eval(['flag=exist(''',filename1,'.mat'');'])
        if flag < 1,
disp(' ')
disp('           The file does not exist, try again or < Ctrl-C >')
disp('           to exit program.')
        end
```

50

```
      end
      eval(['load ',filename1])

         while check > 0

         clc
disp(' ')
disp('                        *** EDIT MENU ***')
disp(' ')
disp('            1. pressure altitude     2. temperature')
disp('            3. airspeed              4. gross weight')
disp('            5. number of blades      6. blade radius')
disp('            7. blade chord           8. hinge offset')
disp('            9. blade grip length    10. blade twist ')
disp('           11. blade weight         12. # blade elements')
disp('           13. rotational velocity  14. # azimuth sectors')
disp('           15. lift curve slope     16. airfoil')
disp('           17. collective pitch     18. flatplate area')
disp('           19. vert projected area  20. wing area')
disp('           21. wing span            22. wing CL')
disp('           23. wing CDo             24. wing efficiency factor')
disp('           25. horizontal tail area 26. horizontal tail span')
disp('           27. horizontal tail CL   28. horizontal tail CDo')
disp('           29. vertical tail area   30. vertical tail span')
disp('           31. vertical tail CL     32. vertical tail CDo')
disp('           33. auxiliary thrust')
disp(' ')
disp('            0. NO CHANGES')

         choice=input('           Input the parameter to change: ');
         if isempty(choice),
           choice=0;
         end
         if choice==1,
           clc
           disp(' ')
           PA
           tmp=PA;
           PA=input('Pressure altitude (ft): ');
           if isempty(PA),
             PA=tmp;
           end
         elseif choice==2,
           clc
           disp(' ')
           temp
           tmp=temp;
           temp=input('Temperature (deg F): ');
           if isempty(temp),
             temp=tmp;
           end
         elseif choice==3,
           clc
           disp(' ')
           Vinf=Vinf/1.69
           tmp=Vinf;
           Vinf=input('Airspeed (knots): ')*1.69;
           if isempty(Vinf),
             Vinf=tmp*1.69;
           end
         elseif choice==4,
           clc
```

51

```
      disp(' ')
      GW
      tmp=GW;
      GW=input('Aircraft gross weight (lbs): ');
      if isempty(GW),
        GW=tmp;
      end
    elseif choice==5,
      clc
      disp(' ')
      b
      tmp=b;
      b=input('Number of blades: ');
      if isempty(b),
        b=tmp;
      end
    elseif choice==6,
      clc
      disp(' ')
      R
      tmp=R;
      R=input('Blade radius: center of hub to blade tip (ft): ');
      if isempty(R),
        R=tmp;
      end
    elseif choice==7,
      clc
      disp(' ')
      cblade
      tmp=cblade;
      cblade=input('Blade chord (ft): ');
      if isempty(cblade),
        cblade=tmp;
      end
    elseif choice==8,
      clc
      disp(' ')
      e
      tmp=e;
      e=input('Hinge offset (ft): ');
      if isempty(e),
        e=tmp;
      end
    elseif choice==9,
      clc
      disp(' ')
      grip
      tmp=grip;
      grip=input('Non-aerodyn inboard portion of blade (ft): ');
      if isempty(grip),
        grip=tmp;
      end
      if grip < 1e-10,
        grip=1e-10;
      end
    elseif choice==10,
      clc
      disp(' ')
      twist=-twist*57.3
      tmp=twist;
      twist=input('Blade twist (deg): ');
      if isempty(twist),
```

```
        twist=abs(tmp)/57.3;
    else
        twist=abs(twist)/57.3;
    end
elseif choice==11,
    clc
    disp(' ')
    wblade
    tmp=wblade;
    wblade=input('Weight of aero portion of blade (lbs): ');
    if isempty(PA),
        wblade=tmp;
    end
elseif choice==12,
    clc
    disp(' ')
    nbe
    tmp=nbe;
    nbe=input('Number of blade elements: ');
    if isempty(nbe),
        nbe=tmp;
    end
elseif choice==13,
    clc
    disp(' ')
    omega
    tmp=omega;
    omega=input('Rotor rotational velocity (rad/sec): ');
    if isempty(omega),
        omega=tmp;
    end
elseif choice==14,
    clc
    disp(' ')
    naz
    tmp=naz;
    naz=input('Number of azimuth sectors: ');
    if isempty(naz),
        naz=tmp;
    end
elseif choice==15,
    clc
    disp(' ')
    a
    tmp=a;
    a=input('Average lift curve slope (CL vs alpha): ');
    if isempty(a),
        a=tmp;
    end
elseif choice==16,
    clc
    disp(' ')
    airfoil
    tmp=airfoil;
    flag=1;
    while flag > 0
        airfoil=input('Airfoil  1. HH-02  2. VR-12   >> ');
        if isempty(airfoil),
            airfoil=tmp;
        end
        if airfoil==1,
            flag=0;
```

53

```
          elseif airfoil==2,
            flag=0;
          else
            disp(' ')
            disp('  *** Enter a 1 or 2 ***')
          end
    end
  elseif choice==17,
    clc
    disp(' ')
    thetao=thetao*57.3
    tmp=thetao;
    thetao=input('Collective pitch at .7 r/R (deg): ')/57.3;
    if isempty(thetao),
      thetao=tmp/57.3;
    end
  elseif choice==18,
    clc
    disp(' ')
    Afh
    tmp=Afh;
    Afh=input('Aircraft equivalent flatplate area (ft^2): ');
    if isempty(Afh),
      Afh=tmp;
    end
  elseif choice==19,
    clc
    disp(' ')
    Afv
    tmp=Afv;
    Afv=input('Vertical projected area (ft^2): ');
    if isempty(Afv),
      Afv=tmp;
    end
  elseif choice==20,
    clc
    disp(' ')
    Swing
    tmp=Swing;
    Swing=input('Wing area (ft^2): ');
    if isempty(Swing),
      Swing=tmp;
    end
    if Swing < 1e-10,
      Swing=1e-10;
    end
  elseif choice==21,
    clc
    disp(' ')
    bwing
    tmp=bwing;
    bwing=input('Wing span (ft): ');
    if isempty(bwing),
      bwing=tmp;
    end
    if bwing < 1e-10,
      bwing=1e-10;
    end
  elseif choice==22,
    clc
    disp(' ')
    CLwing
```

```matlab
    tmp=CLwing;
    CLwing=input('Expected CL for the wing: ');
    if isempty(CLwing),
      CLwing=tmp;
    end
  elseif choice==23,
    clc
    disp(' ')
    CDowing
    tmp=CDowing;
    CDowing=input('Wing profile drag coef (CDo): ');
    if isempty(CDowing),
      CDowing=tmp;
    end
  elseif choice==24,
    clc
    disp(' ')
    ewing
    tmp=ewing;
    ewing=input('Wing efficiency factor (e): ');
    if isempty(ewing),
      ewing=tmp;
    end
    if ewing < 1e-10,
      ewing=1e-10;
    end
  elseif choice==25,
    clc
    disp(' ')
    Shoriz
    tmp=Shoriz;
    Shoriz=input('Horizontal tail area (ft^2): ');
    if isempty(Shoriz),
      Shoriz=tmp;
    end
    if Shoriz < 1e-10,
      Shoriz=1e-10;
    end
  elseif choice==26,
    clc
    disp(' ')
    bhoriz
    tmp=bhoriz;
    bhoriz=input('Horizontal tail span (ft): ');
    if isempty(bhoriz),
     bhoriz=tmp;
    end
    if bhoriz < 1e-10,
      bhoriz=1e-10;
    end
  elseif choice==27,
    clc
    disp(' ')
    CLhoriz
    tmp=CLhoriz;
    CLhoriz=input('Expected CL for the horizontal tail: ');
    if isempty(CLhoriz),
      CLhoriz=tmp;
    end
  elseif choice==28,
    clc
    disp(' ')
```

```
        CDohoriz
        tmp=CDohoriz;
        CDohoriz=input('Horizontal tail profile drag coef (CDo): ');
        if isempty(CDohoriz),
           CDohoriz=tmp;
        end
      elseif choice==29,
        clc
        disp(' ')
        Svert
        tmp=Svert;
        Svert=input('Vertical tail area (ft^2): ');
        if isempty(Svert),
           Svert=tmp;
        end
        if Svert < 1e-10,
           Svert=1e-10;
        end
      elseif choice==30,
        clc
        disp(' ')
        bvert
        tmp=bvert;
        bvert=input('Vertical tail span (ft): ');
        if isempty(bvert),
           bvert=tmp;
        end
        if bvert < 1e-10,
           bvert=1e-10;
        end
      elseif choice==31,
        clc
        disp(' ')
        CLvert
        tmp=CLvert;
        CLvert=input('Expected CL for the vertical tail: ');
        if isempty(CLvert),
           CLvert=tmp;
        end
      elseif choice==32,
        clc
        disp(' ')
        CDovert
        tmp=CDovert;
        CDovert=input('Vertical tail profile drag coef (CDo): ');
        if isempty(CDovert),
           CDovert=tmp;
        end
      elseif choice==33,
        clc
        disp(' ')
        Taux
        tmp=Taux;
        Taux=input('Auxiliary thrust (lbs): ');
        if isempty(Taux),
           Taux=tmp;
        end
      elseif choice==0,

        clc
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('          ex: dsgn_2')
disp(' ')
disp('          E. If you made no changes, press < Enter >, the file will')
disp('             be saved with the original name.')
       flag=1;
       while flag > 0
         filename0=filename1;
         filename1=input('          save file as: ','s');
           if isempty(filename1),
             filename1=filename0;
           end
           clear filename0
           if length(filename1) > 6,
             disp(' ')
             disp('          use 6 characters or less')
             flag=1;
           else
             flag=0;
           end
       end
       eval(['save ',filename1])
       check=0;
     else
disp(' ')
disp('          Enter a displayed number ...press any key to continue')
       pause
     end
   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. ***

  elseif answer==2,
    clc
    PA=input('Pressure altitude (ft): ');
    while isempty(PA),
      disp(' ')
      disp('You must enter a numerical value')
      PA=input('Pressure altitude (ft): ');
    end
    temp=input('Temperature (deg F): ');
    while isempty(temp),
      disp(' ')
      disp('You must enter a numerical value')
      temp=input('Temperature (deg F): ');
    end
    Vinf=input('Airspeed (knots): ')*1.69;
    while isempty(Vinf),
      disp(' ')
      disp('You must enter a numerical value')
      Vinf=input('Airspeed (knots): ')*1.69;
    end
```

57

```
GW=input('Aircraft gross weight (lbs): ');
while isempty(GW),
  disp(' ')
  disp('You must enter a numerical value')
  GW=input('Aircraft gross weight (lbs): ');
end
b=input('Number of blades: ');
while isempty(b),
  disp(' ')
  disp('You must enter a numerical value')
  b=input('Number of blades: ');
end
R=input('Blade radius: center of hub to blade tip (ft): ');
while isempty(R),
  disp(' ')
  disp('You must enter a numerical value')
  R=input('Blade radius: center of hub to blade tip (ft): ');
end
cblade=input('Blade chord (ft): ');
while isempty(cblade),
  disp(' ')
  disp('You must enter a numerical value')
  cblade=input('Blade chord (ft): ');
end
e=input('Hinge offset (ft): ');
while isempty(e),
  disp(' ')
  disp('You must enter a numerical value')
  e=input('Hinge offset (ft): ');
end
grip=input('Non-aerodynamic inboard portion of blade (ft): ');
while isempty(grip),
  disp(' ')
  disp('You must enter a numerical value')
  grip=input('Non-aerodynamic inboard portion of blade (ft): ');
end
while grip < 1e-10,
  grip=1e-10;
end
twist=input('Blade twist (deg): ')/57.3;,twist=abs(twist);
while isempty(twist),
  disp(' ')
  disp('You must enter a numerical value')
  twist=input('Blade twist (deg): ')/57.3;,twist=abs(twist);
end
wblade=input('Weight of aero portion of one blade (lbs): ');
while isempty(wblade),
  disp(' ')
  disp('You must enter a numerical value')
  wblade=input('Weight of aero portion of one blade (lbs): ');
end
nbe=input('Number of blade elements: ');
while isempty(nbe),
  disp(' ')
  disp('You must enter a numerical value')
  nbe=input('Number of blade elements: ');
end
omega=input('Rotor rotational velocity (rad/sec): ');
while isempty(omega),
  disp(' ')
  disp('You must enter a numerical value')
  omega=input('Rotor rotational velocity (rad/sec): ');
```

```
end
naz=input('Number of azimuth sectors: ');
while isempty(naz),
  disp(' ')
  disp('You must enter a numerical value')
  naz=input('Number of azimuth sectors: ');
end
a=input('Lift curve slope of rotor airfoil (CL vs alpha): ');
while isempty(a),
  disp(' ')
  disp('You must enter a numerical value')
  a=input('Lift curve slope of rotor airfoil (CL vs alpha): ');
end
flag=1;
while flag > 0
  airfoil=input('Airfoil  1. HH-02  2. VR-12   >> ');
    while isempty(airfoil),
      disp(' ')
      disp('You must enter a numerical value')
      airfoil=input('Airfoil  1. HH-02  2. VR-12   >> ');
    end
    if airfoil~=1 & airfoil~=2,
      disp(' ')
      disp('  *** Enter a 1 or 2 ***')
    else
      flag=0;
    end
end
thetao=input('Collective pitch at .7 r/R (deg): ')/57.3;
while isempty(thetao),
  disp(' ')
  disp('You must enter a numerical value')
  thetao=input('Collective pitch at .7 r/R (deg): ')/57.3;
end
Afh=input('Aircraft equivalent flatplate area (ft^2): ');
while isempty(Afh),
  disp(' ')
  disp('You must enter a numerical value')
  Afh=input('Aircraft equivalent flatplate area (ft^2): ');
end
Afv=input('Vertical projected area (ft^2): ');
while isempty(Afv),
  disp(' ')
  disp('You must enter a numerical value')
  Afv=input('Vertical projected area (ft^2): ');
end
Swing=input('Wing area, 0 if no wing (ft^2): ');
  while isempty(Swing),
    disp(' ')
    disp('You must enter a numerical value')
    Swing=input('Wing area, 0 if no wing (ft^2): ');
  end
if Swing~=0,
  bwing=input('Wing span (ft): ');
    while isempty(bwing),
      disp(' ')
      disp('You must enter a numerical value')
      bwing=input('Wing span (ft): ');
    end
    if bwing < 1e-10,
      bwing=1e-10;
    end
```

```matlab
    CLwing=input('Expected CL for the wing: ');
      while isempty(CLwing),
        disp(' ')
        disp('You must enter a numerical value')
        CLwing=input('Expected CL for the wing: ');
      end
    CDowing=input('Wing profile drag coef (CDo): ');
      while isempty(CDowing),
        disp(' ')
        disp('You must enter a numerical value')
        CDowing=input('Wing profile drag coef (CDo): ');
      end
    ewing=input('Wing efficiency factor (e): ');
      while isempty(ewing),
        disp(' ')
        disp('You must enter a numerical value')
        ewing=input('Wing efficiency factor (e): ');
      end
      if ewing < 1e-10,
        ewing=1e-10;
      end
  else
    Swing=1e-10;
    bwing=1e-10;
    CLwing=0;
    CDowing=0;
    ewing=1e-10;
  end
Shoriz=input('Horizontal tail area, 0 if none (ft^2): ');
  while isempty(Shoriz),
    disp(' ')
    disp('You must enter a numerical value')
    Shoriz=input('Horizontal tail area, 0 if none (ft^2): ');
  end
if Shoriz~=0,
  bhoriz=input('Horizontal tail span (ft): ');
    while isempty(bhoriz),
      disp(' ')
      disp('You must enter a numerical value')
      bhoriz=input('Horizontal tail span (ft): ');
    end
  if bhoriz < 1e-10,
    bhoriz=1e-10;
  end
  CLhoriz=input('Expected CL for the horizontal tail: ');
    while isempty(CLhoriz),
      disp(' ')
      disp('You must enter a numerical value')
      CLhoriz=input('Expected CL for the horizontal tail: ');
    end
  CDohoriz=input('Horizontal tail profile drag coef (CDo): ');
    while isempty(CDohoriz),
      disp(' ')
      disp('You must enter a numerical value')
      CDohoriz=input('Horizontal tail profile drag coef (CDo): ');
    end
else
  Shoriz=1e-10;
  bhoriz=1e-10;
  CLhoriz=0;
  CDohoriz=0;
end
```

```
      Svert=input('Vertical tail area, 0 if none (ft^2): ');
        while isempty(Svert),
          disp(' ')
          disp('You must enter a numerical value')
          Svert=input('Vertical tail area, 0 if none (ft^2): ');
        end
      if Svert~=0,
        bvert=input('Vertical tail span (ft): ');
          while isempty(bvert),
            disp(' ')
            disp('You must enter a numerical value')
            bvert=input('Vertical tail span (ft): ');
          end
          if bvert < 1e-10,
            bvert=1e-10;
          end
        CLvert=input('Expected CL for the vertical tail: ');
          while isempty(CLvert),
            disp(' ')
            disp('You must enter a numerical value')
            CLvert=input('Expected CL for the vertical tail: ');
          end
        CDovert=input('Vertical tail profile drag coef (CDo): ');
          while isempty(CDovert),
            disp(' ')
            disp('You must enter a numerical value')
            CDovert=input('Vertical tail profile drag coef (CDo): ');
          end
      else
        Svert=1e-10;
        bvert=1e-10;
        CLvert=0;
        CDovert=0;
      end
      Taux=input('Auxiliary thrust (lbs): ');
        while isempty(Taux),
          disp(' ')
          disp('You must enter a numerical value')
          Taux=input('Auxiliary thrust (lbs): ');
        end

      clc
disp(' ')
disp(' ')
disp('                        *** SAVE INSTRUCTIONS ***')
disp(' ')
disp('          A. Save the 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('          ex: dsgn_1')
disp(' ')
disp('          E.  If you do not enter a name, the default is "dsgn_1" ')
      flag=1;
      while flag > 0
        filename1=input('          save file as: ','s');
          if isempty(filename1),
            filename1='dsgn_1';
          end
          if length(filename1) > 6,
            disp(' ')
```

```
                disp('              use 6 characters or less')
                flag=1;
             else
                flag=0;
             end
      end
      eval(['save ',filename1])
      check=0;
   else
      disp(' ')
      disp('    Enter a 1 or 2')
   end
end
clc
disp(' ')
disp(' ')
disp('                      *** DATA ENTRY COMPLETE ***')
pause(3)


check2=1;
while check2 > 0

clc
disp(' ')
disp('                      *** EXECUTION MENU ***')
disp(' ')
disp('                1.  Rotor Performance Analysis')
disp('                2.  Stability and Control Analysis')
disp('                3.  Rotor Dynamics Analysis')
disp('                4.  Change data')
disp('                5.  Quit')
   check3=1;
   while check3 > 0
      answer=input('              Enter a 1, 2, 3, 4, or 5   >> ');
      save temp check1 check2 check3 filename1

      if answer == 1,
         perf
         load temp
         check3=0;

      elseif answer == 2,
         stab
         load temp
         check3=0;

      elseif answer == 3,
%     insert name of blade dynamics M-file here
         load temp
         check3=0;
disp(' ')
disp('              Blade dynamics module is currently under development')
disp(' ')
disp('              press any key to continue...')
pause

      elseif answer == 4,
         clc
         check2=0;
         check3=0;
```

62

```
    elseif answer == 5,
       disp(' ')
       check1=0;
       check2=0;
       check3=0;
    else
       disp(' ')
    end
  end
end
end
```

```
%  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+thetalc.*cos(psi)+thetals.*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 ***
```

66

```
      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,
        theta..  ·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,
```

67

```
              thetao=thetao+0.25*thetao*abs(1.25*error0/T)*(1-mu);
           end
        theta=thetao+thetalc.*cos(psi)+thetals.*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);
      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

 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
```

70

```
       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)
```

APPENDIX C: PERF.M

```
%    *** Performance output subroutine ***

%    *** Clearing all variables in the Matlab environment ***

clear

%    *** Loading input and check parameters ***

load temp
eval(['load ',filename1])

clc
disp(' ')
disp('                  *** ROTOR PERFORMANCE ROUTINE ***')
disp('                               *******')
disp(' ')

trim

%    *** Calculation of output parameters ***

Protor=mean(DMpsi)*b*omega/550;
Qrotor=mean(DMpsi)*b;
solidity=b*cblade/(pi*R);
CQ=Qrotor/(Adisk*rho*Vtip^2*R);
CH=Hrotor/(Adisk*rho*Vtip^2);
CT_sig=CT/solidity;
CQ_sig=CQ/solidity;
CH_sig=CH/solidity;
Machtip=(Vtip*cos(alphaT)+Vinf)/(49.05*sqrt(temp+460));
  if Vinf < 16.9,
    DL=T/(pi*R^2);
    FM=(T*sqrt(DL/(2*rho)))/(550*Protor);
  else
    DL=0;
    FM=0;
  end

clc
disp(' ')
disp('                  *** PERFORMANCE CALCULATIONS COMPLETE ***')
disp(' ')
disp('         Do you want the performance results displayed on
screen?')
flag=1;
while flag > 0
  answer=input('          1. yes   2. no    >> ');
  if answer == 2,
    flag=0;
  elseif answer == 1,


%    *** output to screen ***

clc
disp(' ')
disp('                *** INPUT DATA ***')
```

73

```
eval(['disp(''                            ',filename1,''')'])

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('                       Gross weight = %6.0f lbs\n',GW)
fprintf('                   Number of blades = %6.0f \n',b)
fprintf('                       Rotor radius = %6.2f ft\n',R)
fprintf('                        Blade chord = %6.2f ft\n',cblade)
fprintf('                        Blade twist = %6.2f degs\n',-1*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 = %6.2f lbs\n',wblade)
fprintf('               Rotational velocity = %6.2f rads/sec\n',omega)
fprintf('                  Blade grip length = %6.2f ft\n',grip)
fprintf('                        Hinge offset = %6.2f ft\n',e)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause


clc
disp(' ')
disp('        *** INPUT DATA CONTINUED ***')
eval(['disp(''                    ',filename1,''')'])
disp(' ')
fprintf('    Equivalent flat plate area = %6.2f ft^2\n',Afh)
fprintf('        Vertical projected area = %6.2f ft^2\n',Afv)
fprintf('                      Wing area = %6.2f ft^2\n',Swing)
fprintf('                      Wing span = %6.2f ft\n',bwing)
fprintf('                        Wing CL = %6.2f \n',CLwing)
fprintf('                       Wing CDo = %6.4f \n',CDowing)
fprintf('        Wing efficiency factor = %6.2f \n',ewing)
fprintf('           Horizontal tail area = %6.2f ft^2\n',Shoriz)
fprintf('           Horizontal tail span = %6.2f ft\n',bhoriz)
fprintf('             Horizontal tail CL = %6.2f \n',CLhoriz)
fprintf('             Horizontal tail CDo = %6.4f \n',CDohoriz)
fprintf('             Vertical tail area = %6.2f ft^2\n',Svert)
fprintf('             Vertical tail span = %6.2f ft\n',bvert)
fprintf('               Vertical tail CL = %6.2f \n',CLvert)
fprintf('               Vertical tail CDo = %6.4f \n',CDovert)
fprintf('               Auxiliary thrust = %6.0f lbs\n',Taux)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause


clc
disp(' ')
disp('            *** CALCULATED DATA ***')
eval(['disp(''                    ',filename1,''')'])
disp(' ')
fprintf('                   Fuselage drag = %6.0f lbs\n',Dfuse)
fprintf('                      Rotor drag = %6.0f lbs\n',Hrotor)
fprintf('                       Wing lift = %6.0f lbs\n',Lwing)
fprintf('                       Wing drag = %6.0f lbs\n',Dwing)
fprintf('             Horizontal tail lift = %6.0f lbs\n',Lhoriz)
```

74

```
fprintf('          Horizontal tail drag = %6.0f lbs\n',Dhoriz)
fprintf('     Vertical tail side force = %6.0f lbs\n',Lvert)
fprintf('          Vertical tail drag = %6.0f lbs\n',Dvert)
fprintf('             Tip path angle = %6.2f degs\n',alphaT*57.3)
fprintf('         Rotor coning angle = %6.2f degs\n',betao*57.3)
fprintf('Location of mean thrust (r/R) = %6.2f \n',rT2)
fprintf('    Collective pitch at .7 r/R = %6.2f degs\n',thetao*57.3)
fprintf(' 1st lat cyclic term-A1 (deg) = %6.2f \n',theta1c*57.3)
fprintf('1st long cyclic term-B1 (deg) = %6.2f \n',theta1s*57.3)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause

clc
disp(' ')
disp('     *** CALCULATED DATA CONTINUED ***')
eval(['disp(''                    ',filename1,''')'])
disp(' ')
fprintf('           solidity (sigma) = %6.3f \n',solidity)
fprintf('               Disk loading = %6.2f lbs/ft^2\n',DL)
fprintf('            Figure of Meri' = %6.2f \n',FM)
fprintf('                   CT/sigma = %6.3f \n',CT_sig)
fprintf('                   CQ/sigma = %6.4f \n',CQ_sig)
fprintf('                   CH/sigma = %6.4f \n',CH_sig)
fprintf('    Tip mach of the adv. blade = %6.3f \n',Machtip)
fprintf('              Advance ratio = %6.3f \n',mu)
fprintf('   Rotor thrust required (TPP) = %6.0f lbs\n',T)
fprintf('        Rotor power required = %6.0f h.p.\n',Protor)
fprintf('              Rotor torque = %6.0f ft-lbs\n',Qrotor)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause

clc
    flag=0;
  else
    disp(' ')
    disp('          Enter a 1 or 2')
  end
end


%   *** output to disk (text file) ***
disp(' ')
disp('          *** Saving data to disk ***')
pause(2)
diary on
diary off
delete diary
diary on

disp('              *** RESULTS ***')
eval(['disp(''                    ',filename1,''')'])
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('               Gross weight = %6.0f lbs\n',GW)
fprintf('           Number of blades = %6.0f \n',b)
fprintf('               Rotor radius = %6.2f ft\n',R)
```

```
fprintf('                        Blade chord = %6.2f ft\n',cblade)
fprintf('                        Blade twist = %6.2f degs\n',-1*twist*57.3)
fprintf('             Blade lift curve slope = %6.2f \n',a)
fprintf('                       Blade weight = %6.2f lbs\n',wblade)
fprintf('                Rotational velocity = %6.2f rads/sec\n',omega)
fprintf('                  Blade grip length = %6.2f ft\n',grip)
fprintf('                       Hinge offset = %6.2f ft\n',e)
fprintf('        Equivalent flat plate area = %6.2f ft^2\n',Afh)
fprintf('            Vertical projected area = %6.2f ft^2\n',Afv)
fprintf('                          Wing area = %6.2f ft^2\n',Swing)
fprintf('                          Wing span = %6.2f ft\n',bwing)
fprintf('                            Wing CL = %6.2f \n',CLwing)
fprintf('                           Wing CDo = %6.4f \n',CDowing)
fprintf('             Wing efficiency factor = %6.2f \n',ewing)
fprintf('               Horizontal tail area = %6.2f ft^2\n',Shoriz)
fprintf('               Horizontal tail span = %6.2f ft\n',bhoriz)
fprintf('                 Horizontal tail CL = %6.2f \n',CLhoriz)
fprintf('                Horizontal tail CDo = %6.4f \n',CDohoriz)
fprintf('                 Vertical tail area = %6.2f ft^2\n',Svert)
fprintf('                 Vertical tail span = %6.2f ft\n',bvert)
fprintf('                   Vertical tail CL = %6.2f \n',CLvert)
fprintf('                  Vertical tail CDo = %6.4f \n',CDovert)
fprintf('                      Fuselage drag = %6.0f lbs\n',Dfuse)
fprintf('                         Rotor drag = %6.0f lbs\n',Hrotor)
fprintf('                          Wing lift = %6.0f lbs\n',Lwing)
fprintf('                          Wing drag = %6.0f lbs\n',Dwing)
fprintf                Horizontal tail lift = %6.0f lbs\n',Lhoriz)
fprintf,                Horizontal tail drag = %6.0f lbs\n',Dhoriz)
fprintf('            Vertical tail side force = %6.0f lbs\n',Lvert)
fprintf('                 Vertical tail drag = %6.0f lbs\n',Dvert)
fprintf('                   Auxiliary thrust = %6.0f lbs\n',Taux)
fprintf('                     Tip path angle = %6.2f degs\n',alphaT*57.3)
fprintf('                 Rotor coning angle = %6.2f degs\n',betao*57.3)
fprintf('Location of mean thrust (r/R) = %6.2f \n',rT2)
fprintf('    Collective pitch at .7 r/R = %6.2f degs\n',thetao*57.3)
fprintf(' 1st lat cyclic term-A1 (deg) = %6.2f \n',theta1c*57.3)
fprintf('1st long cyclic term-B1 (deg) = %6.2f \n',theta1s*57.3)
fprintf('                           solidity = %6.3f \n',solidity)
fprintf('                       Disk loading = %6.2f lbs/ft^2\n',DL)
fprintf('                    Figure of Merit = %6.2f \n',FM)
fprintf('                           CT/sigma = %6.3f \n',CT_sig)
fprintf('                           CQ/sigma = %6.4f \n',CQ_sig)
fprintf('                           CH/sigma = %6.4f \n',CH_sig)
fprintf('    Tip mach of the adv. blade = %6.3f \n',Machtip)
fprintf('                       Advance ratio = %6.3f \n',mu)
fprintf('    Rotor thrust required (TPP) = %6.0f lbs\n',T)
fprintf('               Rotor power required = %6.0f h.p.\n',Protor)
fprintf('                        Rotor torque = %6.0f ft-lbs\n',Qrotor)
diary off

clc
disp(' ')
disp('                *** PERFORMANCE OUTPUT DATA INSTRUCTIONS ***')
disp(' ')
disp('        A. Single value data saved to a file named:')
eval(['disp(''            "',filename1,'.prf"'')'])
disp(' ')
disp('        B. This is a text file, use the TYPE command to view the')
disp('            file or use a text editor to view/print the file.')
eval(['flag=exist('''',filename1,'.prf'');'])
  if flag < 1,
```

```
      eval(['!rename diary ',filename1,'.prf''']))
    else
      eval(['!del ',filename1,'.prf''']))
      eval(['!rename diary ',filename1,'.prf''']))
    end


%    *** Output to disk (.mat file containing matrix variables
%        r, psi, vi, theta, betat, alpha, Tpsi, Mpsi, DMpsi,
%        dT, dM, dD) ***

%    *** Configuring variables for output ***

theta=theta*57.3;
betat=[betat twist*(0.7-(Reff+(R-Reff)/2)/R)]*57.3;
alpha=alpham*57.3;,alpha=[alpha zeros(psi)];
Mpsi=Mpsi(:,length(Mpsi(1,:))-1);
dM=[dM ddM];
psi=psi*57.3;
r=[r (R-(R-Reff)/2)];
vi=[vi 0];


filename2=[filename1 '_p'];
disp(' ')
disp('            C. Matrix and vector data saved to a file named:')
eval(['disp(''            "',filename2,'.mat"'')'])
disp(' ')
disp('            D. This is a ".mat" binary file, use the LOAD command
to')
disp('               retrieve the data for plotting.')
eval(['save ',filename2,' r psi vi theta betat alpha Tpsi Mpsi DMpsi dT
dM dD'])


disp(' ')
disp('                      *** END OF PERFORMANCE ***')
disp(' ')
disp(' ')
disp('            press any key to continue ...')
pause
```

APPENDIX D: THRCALC.M

```
%thrcalc calculates the total thrust along a blade at
%each azimuth (psi) location

Up=zeros(psi*r);
Ut=zeros(Up);
dT=zeros(Up);
ddT=zeros(psi);

for i=1:length(psi),

Up(i,:)=vi.*cos(betao)+Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(
betao)*cos(psi(i));
   Ut(i,:)=r.*omega+Vinf*cos(alphaT)*sin(psi(i));
   phi=atan2(Up(i,:),Ut(i,:));
   alpha=theta(i)+betat-phi;
      if airfoil==1,
        [CL,CD]=hh02clcd(alpha);
      else
        [CL,CD]=vr12clcd(alpha);
      end

dT(i,:)=0.5*rho*cblade*dr*(Up(i,:).^2+Ut(i,:).^2).*(CL.*cos(phi)-CD.*sin
(phi));
   Tpsi(i)=sum(dT(i,:));

%   *** calculations for tip loss area ***


Uptip=Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i)
);
   Uttip=(R-(R-Reff)/2)*omega+Vinf*cos(alphaT)*sin(psi(i));
   phitip=atan2(Uptip,Uttip);
   ddT(i)=0.5*rho*cblade*(R-Reff)*(Uptip^2+Uttip^2)*(-.009*sin(phitip));
   Tpsi(i)=Tpsi(i)+ddT(i);
end
```

```
%tmcalc calculates the total thrust moment along a blade
%at each azimuth (psi) location

Up=zeros(psi*r);
Ut=zeros(Up);
dM=zeros(Up);
ddM=zeros(psi);

for i=1:length(psi),

Up(i,:)=vi.*cos(betao)+Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(
betao)*cos(psi(i)));
  Ut(i,:)=r.*omega+Vinf*cos(alphaT)*sin(psi(i)));
  phi=atan2(Up(i,:),Ut(i,:));
  alpha=theta(i,k)+betat-phi;
    if airfoil==1,
      [CL,CD]=hh02clcd(alpha);
    else
      [CL,CD]=vr12clcd(alpha);
    end

dM(i,:)=0.5*rho*cblade.*r*dr.*(Up(i,:).^2+Ut(i,:).^2).*(CL.*cos(phi)-CD.
*sin(phi));
  Mpsi(i,k)=sum(dM(i,:));

%  *** calculations for tip loss areas ***


Uptip=Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i)
);
  Uttip=(R-(R-Reff)/2)*omega+Vinf*cos(alphaT)*sin(psi(i)));
  phitip=atan2(Uptip,Uttip);

ddM(i)=0.5*rho*cblade*(R-(R-Reff)/2)*(R-Reff)*(Uptip^2+Uttip^2)*(-.009*s
in(phitip));
  Mpsi(i,k)=Mpsi(i,k)+ddM(i);
end
```

APPENDIX F: DMCALC.M

```
%dmcalc calculates the total drag along a blade at
%each azimuth (psi) location

Up=zeros(psi*r);
Ut=zeros(Up);
alpham=zeros(Up);
dD=zeros(Up);
ddD=zeros(psi);
ddDM=zeros(psi);

for i=1:length(psi),

Up(i,:)=vi.*cos(betao)+Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(
betao)*cos(psi(i)));
   Ut(i,:)=r.*omega+Vinf*cos(alphaT)*sin(psi(i));
   phi=atan2(Up(i,:),Ut(i,:));
   alpha=theta(i)+betat-phi;
   alpham(i,:)=alpha;
     if airfoil==1,
       [CL,CD]=hh02clcd(alpha);
     else
       [CL,CD]=vr12clcd(alpha);
     end

dD(i,:)=0.5*rho*cblade*dr*(Up(i,:).^2+Ut(i,:).^2).*(CL.*sin(phi)+CD.*cos
(phi));
   dDM=dD(i,:).*r;
   DMpsi(i)=sum(dDM);

%   *** calculations for tip loss area ***


Uptip=Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i)
);
   Uttip=(R-(R-Reff)/2)*omega+Vinf*cos(alphaT)*sin(psi(i));
   phitip=atan2(Uptip,Uttip);
   ddD(i)=0.5*rho*cblade*(R-Reff)*(Uptip^2+Uttip^2)*(.009*cos(phitip));
   ddDM(i)=ddD(i)*(R-(R-Reff)/2);
   DMpsi(i)=DMpsi(i)+ddDM(i);
end
```

80

```
function [CL,CD]=hh02clcd(alpha,Mach)
%hh02clcd calculates CL and CD for an HH-02 airfoil
%given angle of attack (alpha) in radians and Mach number (Mach)
%[CL,CD]=hh02clcd(alpha,Mach)
CL=zeros(alpha);
CD=zeros(alpha);
a=alpha*180/pi;

chk1=(a>=20 & a<=180);

CL=CL+chk1.*(0.42541+0.026863*a+5.5988e-4*a.^2-2.1493e-5*a.^3+1.5932e-7*
a.^4-3.4659e-10*a.^5);

CD=CD+chk1.*(-0.7179+0.061213*a-5.9861e-4*a.^2+7.3708e-6*a.^3-6.6605e-8*
a.^4+1.913e-10*a.^5);

chk1=(a>=-180 & a<=-50);

CL=CL+chk1.*(-4.6183-0.1923*a-3.5554e-3*a.^2-3.3273e-5*a.^3-1.4528e-7*a.
^4-2.3003e-10*a.^5);

CD=CD+chk1.*(2.7093e-2-2.1309e-2*a+2.0335e-4*a.^2+3.47e-7*a.^3-3.0586e-8
*a.^4-1.2584e-10*a.^5);

chk1=(a>-50 & a<-20);

CL=CL+chk1.*(-2.5519-0.22847*a-9.5667e-3*a.^2-1.7051e-4*a.^3-1.0909e-6*a
.^4);

CD=CD+chk1.*(2.7093e-2-2.1309e-2*a+2.0335e-4*a.^2+3.47e-7*a.^3-3.0586e-8
*a.^4-1.2584e-10*a.^5);

chk1=(a>=-20 & a<=-10);
 CL=CL+chk1.*(-0.2+0.089*a+0.0034*a.^2);

CD=CD+chk1.*(2.7093e-2-2.1309e-2*a+2.0335e-4*a.^2+3.47e-7*a.^3-3.0586e-8
*a.^4-1.2584e-10*a.^5);

chk1=(a<20 & a>-10);

CL=CL+chk1.*(5.8766e-2+1.3131e-1*a+2.4742e-3*a.^2-5.303e-4*a.^3-1.5818e-
5*a.^4+1.28e-6*a.^5);
   chk2=a<-4;
   chk2=chk2.*chk1;

CD=CD+chk2.*(1.3786+0.916*a+0.21396*a.^2+2.0371e-2*a.^3+7.0076e-4*a.^4);
   chk2=(a>=-4 & a<=7);
    chk2=chk2.*chk1;

CD=CD+chk2.*(9.732e-3+3.2326e-4*a+1.4392e-4*a.^2-8.5073e-5*a.^3+1.1826e-
6*a.^4+1.5271e-6*a.^5);
   chk2=a>7;
   chk2=chk2.*chk1;
    CD=CD+chk2.*(1.842e-1-5.7532e-2*a+5.8043e-3*a.^2-1.2803e-4*a.^3);
```

81

APPENDIX H: VR12CLCD.M

```
function [CL,CD]=vr12clcd(alpha)
%vr12clcd calculates CL and CD for the VR-12 airfoil
%given angle of attack (alpha) in radians
%[CL,CD]=vr12clcd(alpha)
CL=zeros(alpha);
CD=zeros(alpha);
a=alpha*180/pi;

chk=(a>=20 & a<=180);

CL=CL+chk.*(1.1733-0.018879*a+1.5762e-3*a.^2-3.1925e-5*a.^3+2.0949e-7*a.
^4-4.3807e-10*a.^5);

chk=(a>=-180 & a<=-50);

CL=CL+chk.*(-4.6183-0.1923*a-3.5554e-3*a.^2-3.3273e-5*a.^3-1.4528e-7*a.^
4-2.3003e-10*a.^5);

chk=(a>-50 & a<-30);
 CL=CL+chk.*(-0.22114+0.020857*a+2.8571e-4*a.^2);

chk=(a>=-30 & a<=-10);
 CL=CL+chk.*(-1.11-0.12383*a-0.01515*a.^2-6.8667e-4*a.^3-1e-5*a.^4);

chk=(a<20 & a>-10);
 CL=CL+chk.*(0.11976+0.12341*a+5.5841e-4*a.^2-2.0652e-4*a.^3);

chk=(a>=17 & a<=180);

CD=CD+chk.*(-0.26376+0.017917*a+6.9927e-4*a.^2-9.1137e-6*a.^3+2.6277e-8*
a.^4);

chk=(a>=-180 & a<=-10);

CD=CD+chk.*(-0.17486-0.034463*a-1.0233e-4*a.^2-2.8958e-6*a.^3-4.6577e-8*
a.^4-1.5557e-10*a.^5);

chk=(a>-10 & a<=0);

CD=CD+chk.*(9.8678e-3+3.4934e-3*a+1.4844e-3*a.^2-1.3564e-4*a.^3-1.0936e-
5*a.^4);

chk=(a>0 & a<=15);

CD=CD+chk.*(9.8e-3+7.0457e-4*a+5.6104e-5*a.^2-4.1151e-5*a.^3+3.8695e-6*a
.^4);

chk=(a>15 & a<17);
 CD=CD+chk.*(-1.33+1.325e-1*a-2.5e-3*a.^2);
```

INITIAL DISTRIBUTION LIST

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