

User's Guide for the NREL Teetering Rotor Analysis Program (STRAP)

Alan D. Wright



National Renewable Energy Laboratory
(formerly the Solar Energy Research Institute)
1617 Cole Boulevard
Golden, Colorado 80401-3393
A Division of Midwest Research Institute
Operated for the U.S. Department of Energy
under Contract No. DE-AC02-83CH10093

August 1992

NOTICE

This report was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or any agency thereof.

Printed in the United States of America
Available from:
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

Price: Microfiche A01
Printed Copy A10

Codes are used for pricing all publications. The code is determined by the number of pages in the publication. Information pertaining to the pricing codes can be found in the current issue of the following publications which are generally available in most libraries: *Energy Research Abstracts (ERA)*; *Government Reports Announcements and Index (GRA and I)*; *Scientific and Technical Abstract Reports (STAR)*; and publication NTIS-PR-360 available from NTIS at the above address.

Summary

The following report gives the reader an overview of instructions on the proper use of the National Renewable Energy Laboratory (formerly the Solar Energy Research Institute, or SERI) Teetering Rotor Analysis Program (STRAP version 2.20). STRAP is a derivative of the Force and Loads Analysis Program (FLAP). It is intended as a tool for prediction of rotor and blade loads and response for only two-bladed teetering hub wind turbines. The effects of delta-3, undersling, hub mass, and wind turbulence are accounted for.

The objectives of the report are to give an overview of the code and also show the methods of data input and correct code execution steps in order to model an example two-bladed teetering hub turbine. A large portion of the discussion (Sections 6.0, 7.0, and 8.0) is devoted to the subject of inputting and running the code for wind turbulence effects. The ability to include turbulent wind effects is perhaps the biggest change in the code since the release of FLAP version 2.01 in 1988.

This report is intended to be a user's guide. It does not contain a theoretical discussion on equations of motion, assumptions, underlying theory, etc. It is intended to be used in conjunction with Wright, Buhl, and Thresher (1988).

Hopefully, this report will assist new users in the understanding of code input preparation and correct code execution.

Table of Contents

	<u>Page</u>
Summary	iii
Nomenclature	vii
1.0 Introduction	1
2.0 Code Overview	2
3.0 Description of New Input Variables	5
4.0 Example Turbine Description	9
5.0 Preparation of Input Data	13
6.0 Preparation of Turbulent Wind Input Data	19
7.0 Input of Turbulence Data to STRAP	20
8.0 Example Interactive Run	22
8.1 Module 1—STRAP1	22
8.2 Module 2—STRAP2	22
9.0 Conclusions	25
10.0 References	26
11.0 Acknowledgments	27
Appendix A: Module 1 Input File, Interactive Run Listing, and Output Files	A-1
Appendix B: Module 2 Interactive Run Listing and Output Files	B-1
Appendix C: Module 1 Listing	C-1
Appendix D: Module 2 Listing	D-1
Appendix E: Modules 1 and 2 Flowcharts	E-1

List of Figures

	<u>Page</u>
2-1 Rotor modeshapes	2
2-2 Illustration of the teetering rotor with δ -3, undersling, and hub mass	3
3-1 δ -3 hinge	7
3-2 Illustration of the Teeter Springs and Damper Assembly	8
4-1 Illustration of example turbine	9
4-2 Turbine blade planform, thickness, and twist	10
4-3 Turbine blade stiffness distribution	10
4-4 Turbine blade weight distribution	11
4-5 Lift profile for the LS(1) airfoil	11
5-1 Illustration of rotor geometry	14

List of Tables

	<u>Page</u>
3-1 New Input Variable Description	6
4-1 Example Turbine Specifications	12

Nomenclature

A	Teeter hinge point (Figure 2-2)
C	Teeter damping (ft-lb-s/rad)
C_D	Drag coefficient
C_L	Lift coefficient
deg	Degrees
ft	Feet
m	Meters
M_{hub}	Concentrated hub mass (lb-s ² /ft)
d	Location of the hub center of gravity relative to point of blades' intersection. (see Figure 2-2) (ft)
u	Location of the teeter pin (ft) (see Figure 2-2)
K_0	Teeter hinge stiffness (ft-lb/rad)
K_1	Stiffness of first teeter stop (ft-lb/rad) (see Figure 3-2)
K_2	Stiffness of second teeter hard-stop (ft-lb/rad) (see Figure 3-2)
rad	Radians
β_1	Angle of teeter at which blade hits first teeter stop (deg) (see Figure 3-2)
β_2	Angle of teeter at which blade hits second hard stop (deg) (see Figure 3-2)
$\delta-3$	delta-3 hinge angle (see Figure 3-1) (deg)
Ω	Rotor rotation rate (RPM)

1.0 Introduction

The following report is intended to be a guide for the correct use of the National Renewable Energy Laboratory (NREL) (formerly SERI) Teetering Rotor Analysis Program (STRAP). STRAP is a derivative of the Force and Loads Analysis Program (FLAP). The original equations of motion that formed the basis for FLAP (version 2.01) have been reformulated to correctly predict loads and response for two-bladed teetering hub rotors.

The code accounts for such effects as delta-3, undersling, hub mass, and wind turbulence. Degrees of freedom include rotor teeter and three-blade elastic flap modes. A prescribed time-dependent sinusoidal yaw function can be input to the code.

The report gives a general overview of the code. A description of new input variables (compared to FLAP version 2.01 of 1988) is given. Then, an example of a two-bladed teetering hub wind turbine is described. Preparation of the basic data input file for this two-bladed turbine is described. Then, the methods of turbulent wind input preparation and input of turbulence to STRAP are explained. Finally, an interactive run session for both modules is illustrated. Code listings and output results are contained in the appendices.

2.0 Code Overview

STRAP analyzes two-bladed teetering hub rotors. It predicts blade and low-speed shaft loads that result from such effects as gravity, windshear, tower shadow, and turbulent wind fluctuations.

Degrees of freedom included in this model are rotor teeter and three elastic flap modes, as seen in Figure 2-1. Blade torsion and edgewise degrees of freedom are not included. The tower top is assumed to be fixed in space. The model assumes constant rotor speed. Although machine yaw is not considered to be a degree of freedom, a prescribed time-dependent sinusoidal yaw motion can be input to the code.

STRAP is basically a derivative of FLAP (Wright, Buhl, and Thresher, 1988). Equations of motion have been reformulated to model the type of rotor shown in Figure 2-2. This rotor consists of two blades coupled together by a teetering hinge (at point A). The hinge may be skewed at an angle, $\delta-3$. The teetering hinge may also be located a distance u (ft) downwind of the blades' apex point (0).

The total rotor mass may consist of two parts: (1) that part modeled as distributed mass of each blade and (2) that part due to the hub, not included as blade distributed mass. A description of blade mass input will be given in Section 3.0 on code input. In order to model the second part, the user can input a concentrated hub mass (M_{hub} - lb-s²/ft) located a distance d (ft) downwind of point 0. This point locates the hub center of gravity location (just the hub, with the two blades removed at the root). The blade tip masses shown in Figure 2-2 are modeled in STRAP by adding extra distributed blade mass to the last 2 ft of blade span. Provisions for blade concentrated point masses are not included in STRAP.

The model also includes the effects of teeter springs and dampers. Teeter stop impacts can be modeled by appropriate choice of the spring stiffnesses, as will be described in Section 3.0.

This version of STRAP is similar to FLAP in that a quasisteady linear aerodynamic model is used to compute blade aerodynamic forces. The lift is modeled as a linear function of angle of attack up to stall. Past stall, lift is set equal to a constant. The drag is modeled as a quadratic function involving lift coefficient. For more details, see Section 5.0 or Wright, Buhl, and Thresher (1988).

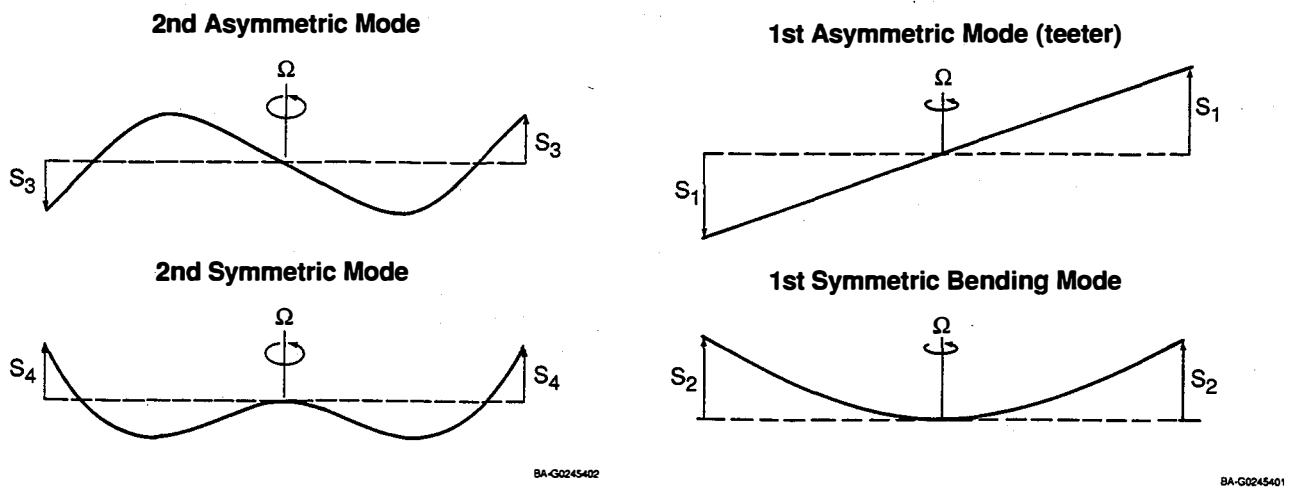


Figure 2-1. Rotor modeshapes

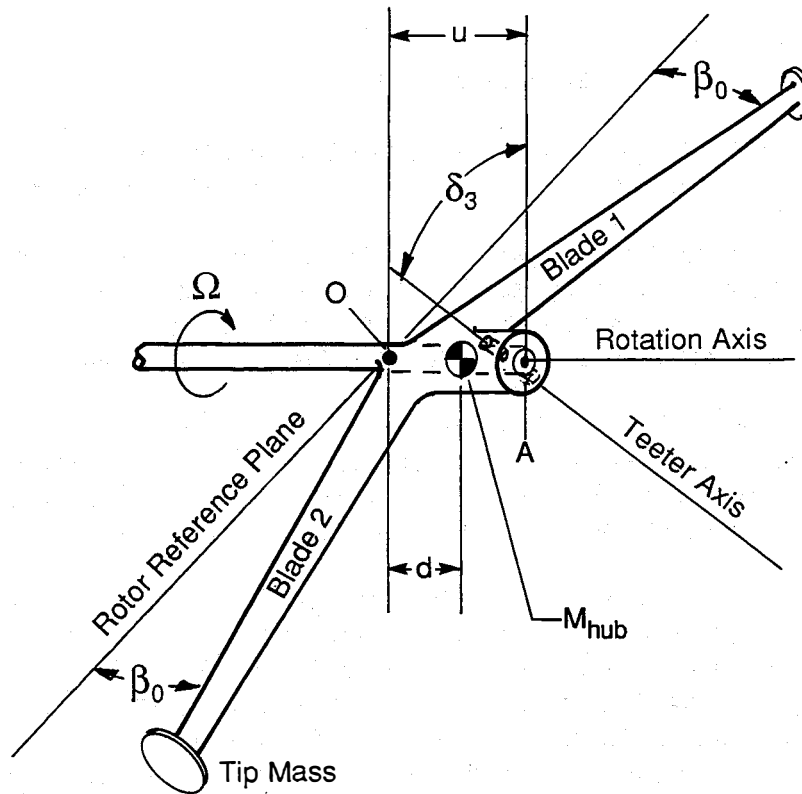


Figure 2-2. Illustration of the teetering rotor with δ -3, undersling, and hub mass

STRAP is composed of two modules (STRAP1 and STRAP2). The first module is a preprocessor that reads blade and machine property data. It computes such items as blade flapwise frequencies and modeshapes as well as stiffness, mass, coriolis, and other matrices. These quantities are variables that do not generally change from one run to the next and, thus, are computed once.

Distributed blade properties such as the blade's stiffness, weight, twist, and chord may be read into Module 1 at unevenly spaced points. The input data is interpolated to form sets of evenly spaced data. Another capability of this module is calculation of the blade's flapwise frequencies and modeshapes. This information is written to a file for examination by the user. From this information, the user can check the effects of mass and stiffness changes on blade frequencies and modeshapes.

We assume in this version of STRAP that both blades have identical mass, stiffness, twist, and chord distributions. In Module 1, mass, stiffness, and other matrices are calculated for only one blade. The effects of mass, pitch, and twist imbalances are not accounted for in this model.

The second module (STRAP2) calculates the rotor equations of motion and computes blade and low-speed shaft loads. The code calculates the rotor response to such input as gravity, windshear, tower shadow, yaw misalignment, and yaw rate. The response to these input is generally a steady-state one. The code is "started up" with initial values for the blade deflection and velocity. Once the blade reaches a steady-state trim solution, the blade loads are calculated and written to a file. The shaft loads are also calculated and written to a separate file.

The rotor response and load calculations resulting from time-dependent yaw motion or turbulent wind input is handled differently than the steady-state trim case. In those cases, a steady-state trim solution is calculated first, before the code enters the transient portion of the solution process. The results of this trim solution are then used as initial conditions for the transient analysis. After this trim solution is completed, the time clock is started, and the model is run, with the yaw function evaluated at each azimuth position.

The yaw-function values for the initial yaw-angle displacement, yaw-angle amplitude, and maximum yaw velocity are set by the user during the model-run setup. Because the yaw function is given independently of the rotor model, the model solution from one rotor revolution to the next will not be steady state.

The number of revolutions to be run in the yaw solution is set by the user during run setup. After each rotor-revolution solution in the yaw routine, loads are computed and printed out.

The procedure for calculating rotor response and loads resulting from turbulent wind fluctuations is similar to the yaw solution procedure described above. The code first calculates rotor response for a trim (steady-state) solution. The computed values for blade deflection and velocity at zero azimuth (blade straight up) are then used as initial conditions for the transient analysis.

In order to run a turbulence analysis with STRAP, a file of turbulent, rotationally sampled wind data is needed. Such a file of data can be generated by such codes as the VEERS Three-Dimensional Wind Simulation (Veers, 1988). Additional details of this file generation and code execution with turbulence will be given in Sections 6.0 and 7.0. Example inputs to Module 1 are described next, including preparation of blade property data for an example turbine. First, some new code input parameters are described.

3.0 Description of New Input Variables

For a review of old input variables to Module 1, the user is encouraged to review Table 3-15 in Wright, Buhl, and Thresher (1988). That table gives a description of all the input variables and lists the equations in which these variables were used. We will only describe herein those variables that are new to Module 1 of STRAP. Table 3-1 gives a description of these new variables.

The first new variables are HUBMAS, UNDSLGL, and HUBDIS. These variables correspond to M_{hub} , u , and d , as shown in Figure 2-2. Input of a nonzero value for UNDSLGL (ft) moves the teeter axis away from the blade apex (point 0) to point A. If a positive value of UNDSLGL (ft) is input, the teeter axis is moved by the distance u downwind of point 0 (the blade apex).

One can input a concentrated hub mass (M_{hub}) HUBMAS (lb-s²/ft) a distance HUBDIS (d-ft) downwind of point 0 (the blade apex). The hub mass may consist of the mass of the hub that is not included in the calculation of the blade's distributed mass (by the input WEIGHT). The inclusion of a concentrated hub mass may move the rotor center of gravity relative to the teeter axis, which is important in the calculation of the rotor's response and loads as a result of gravity. HUBMAS is the mass of the hub only, with the blades removed. HUBDIS locates the center of gravity of the hub relative to the blade apex point (0).

The next input is the variable DELT3, which models the effects of delta-3 in teetered rotors. The effects of delta-3 are to couple flap motion with pitch motion of the blades. Figure 3-1 shows a teetered rotor with delta-3 (δ -3).

In the convention used here, we are located upwind of the blade looking downwind. As the top blade (#1) teeters downwind (away from us), it will also pitch so that the leading edge pitches toward the wind. The blade is rotating clockwise.

It can be shown that the change in angle of attack (or pitch: $\Delta\Theta$) is related to the change in teeter $\Delta\beta$ by

$$\Delta\Theta = \Delta\beta \sin \delta$$

For most wind turbine rotors, DELT3 should be input as a positive value (in degrees). Input of a negative value will result in a destabilizing effect, resulting in poor convergence to a trim solution in Module 2.

The next six variables—BETA1, BETA2, K_0 , K_1 , K_2 , and CDAMP—allow modeling of teeter springs and damping. We will now describe these parameters.

In this analysis, there are three springs with stiffnesses K_0 , K_1 , and K_2 (ft-lb/rad) and one damper with damping constant CDAMP (ft-lb-s/rad). The angles BETA1 and BETA2 (deg) define those teeter angles at which one spring stops acting on the rotor and another one begins. All these springs are considered to be torsional springs with stiffnesses having units of ft-lb/rad. The definitions are

$$K = \begin{cases} K_0 & \text{for } \beta < \beta_1 \\ K_0 + K_1 & \text{for } \beta_1 < \beta < \beta_2 \\ K_0 + K_2 & \text{for } \beta > \beta_2 \end{cases}$$

$$C = \begin{cases} CDAMP & \text{for } \beta > \beta_1 \end{cases}$$

There may be a teeter bearing spring with stiffness K_0 that acts on the rotor for all teeter angles. Once the blade reaches a certain angle, β_1 there may be a second spring with stiffness K_1 . At some angle, β_2 ,

Table 3-1. New Input Variable Description.

Variable Name	Symbol	Description
HUBMAS	M_{hub}	Concentrated mass of only the hub ($1b\text{-s}^2/\text{ft}$) (see Figure 2-2).
UNDSLГ	u	Distance of the teeter pin downwind of the point of blades' intersection (see Figure 2-2).
HUBDIS	d	Location of the hub center of gravity downwind of the blades' intersection point (ft) (see Figure 2-2).
DELТ3	$\delta\text{-3}$	Orientation of the teetering hinge (see Figure 3-1) (deg).
BETA1	β_1	Angle of teeter (deg) at which the rotor impacts first teeter stop (see Figure 3-2).
BETA2	β_2	Angle of teeter (deg) at which rotor impacts second teeter (hard) stop (see Figure 3-2).
K0	K_0	Stiffness of a teeter hinge spring that acts on rotor for all teeter angles (ft-lb/rad) (see Figure 3-2).
K1	K_1	Stiffness of a spring (ft-lb/rad) used to simulate stiffness of the first teeter stop (see Figure 3-2).
K2	K_2	Stiffness of a spring (ft-lb/rad) used to simulate stiffness of the second hard teeter stop (see Figure 3-2).
CDAMP	C	Teeter damping, assumed to occur for teeter angles greater than BETA1 (β_1) (ft-lb-sec/rad).

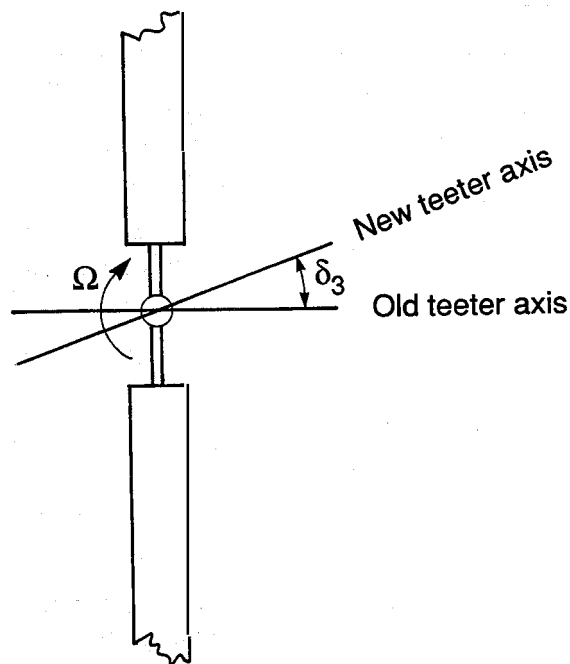


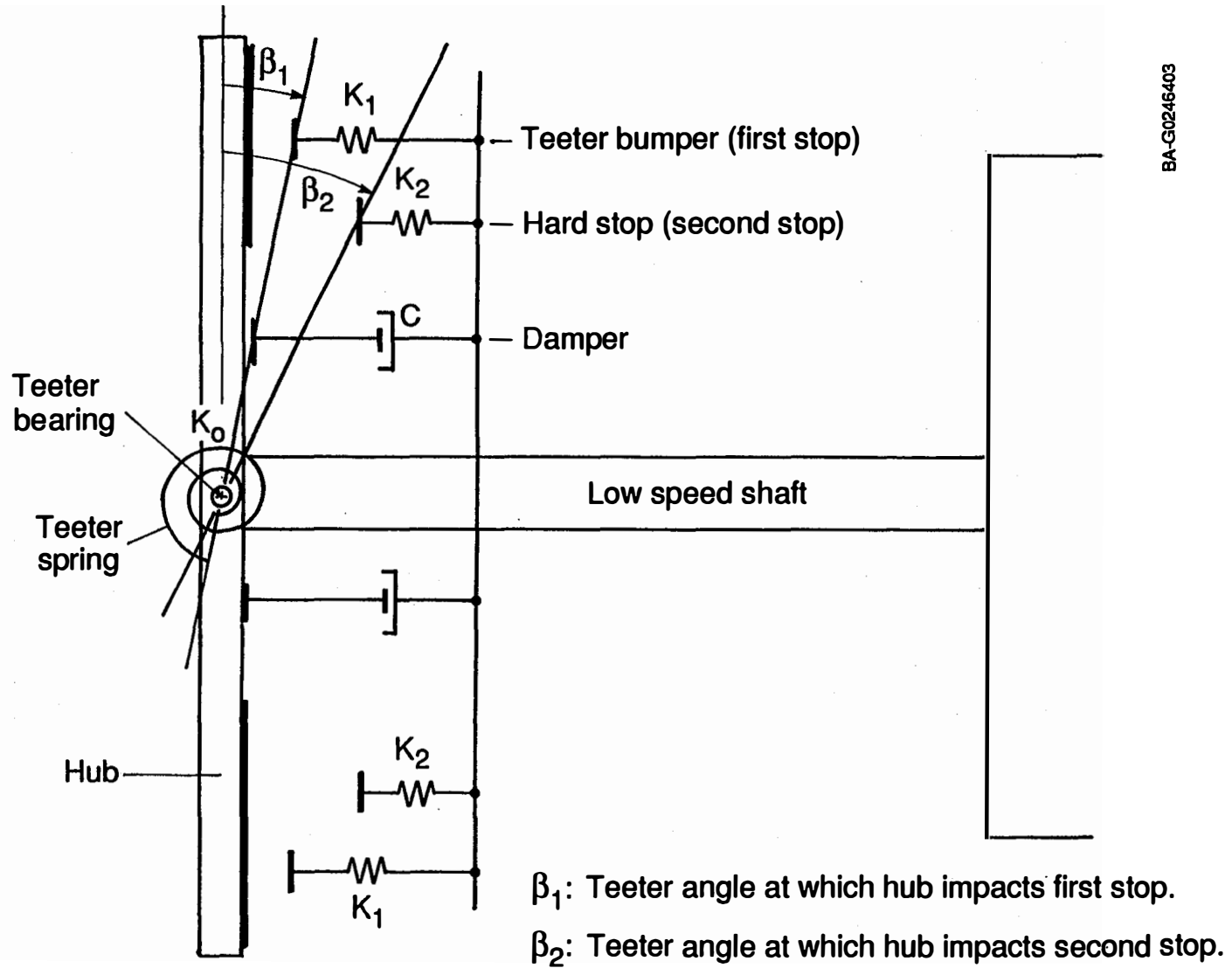
Figure 3-1. δ -3 hinge (as observed looking downwind, rotor rotating clockwise)

the blade may hit a hard stop with stiffness K_2 . The damping (CDAMP) is considered to act for any teeter angles greater than β_1 .

One note of caution is that use of a very high stiffness for K_2 may cause longer run times for Module 2. When running Module 2 with high values for K_1 or K_2 , the user should set the run parameters STEPMX, EUERR, and TRMERR in Module 2 to one. This will cause smaller step sizes in the numeric integration process and improve chances for convergence to a trim solution.

Other new code input include NUMSCN, TIMINC, MSTAT, and STA. All these variables are related to the input of turbulent windspeed fluctuations from a separate file. This file of windspeed data must be produced using a separate code, such as the VEERS model (Veers, 1988).

NUMSCN is the number of lines of windspeed data in the turbulent wind input file. For example, a windspeed file may have 8,192 lines of data. TIMINC is the time increment between each line of wind data. An example file might be sampled in the VEERS model at 24 Hz so that $TIMINC = 1/24$ sec (.041667 sec). MSTAT is the number of blade stations at which turbulent wind data are input. At the present time, only a value of $MSTAT = 2$ is input. STA represents the actual points on the rotor at which wind data are input, in this case, at 20 and 40 ft. These are the rotor 50% and 100% radial locations. It is important to note that STA represents the distance from the rotor center of rotation and not the distance from the blade root. More details will be given in Sections 6.0 and 7.0 on turbulence input.



BA-G0246403

Figure 3-2. Illustration of the Teeter Springs and Damper Assembly

4.0 Example Turbine Description

We will show the data input file preparation for an example two-bladed teetering hub test turbine. Before showing code input, the turbine will be described.

The ESI-80 wind turbine, shown in Figure 4-1, is a two-bladed, fixed-pitch, free-yaw, downwind, stall-controlled turbine. It has a rotor diameter of 24 m (80 ft) and features wood epoxy composite rotor blades. These blades use LS(1)-04XX airfoils with thickness distribution and planform shown in Figure 4-2. This blade has a chord taper ratio of 2.2 beginning at the 30% blade radial station. Figure 4-2 also describes the linear trailing-edge-drop twist distribution of 4.0 deg. Blade stiffness and mass distributions are shown in Figures 4-3 and 4-4. The lift and drag profiles for the LS(1) are shown in Figure 4-5. The blade pitch is set to zero degrees measured at the 75% blade span. The rotor has a solidity of 0.035 and a coning angle of 7.0 deg angled away from the tower. The teetered rotor has a delta-3 angle of zero degrees and rotates at a constant rotational speed of 60 RPM. Aerodynamically shaped tip vanes mounted at the blade tip perpendicular to the spanwise axis provide overspeed protection and assist in high-wind stops. Table 4-1 summarizes the major turbine specifications for the test turbine.

The distance from the yaw axis to the center of the hub is 6.79 ft. The teeter pin is 0.75-ft downwind ($u = 0.75$ ft) of the hub center. The mass of the hub, not modeled as blade mass, is 55.9 lb-s²/ft ($M_{\text{hub}} = 55.9$), located at the point where the blades intersect ($d = 0$). These inputs are important for locating the rotor's center of gravity with respect to the teeter pin.

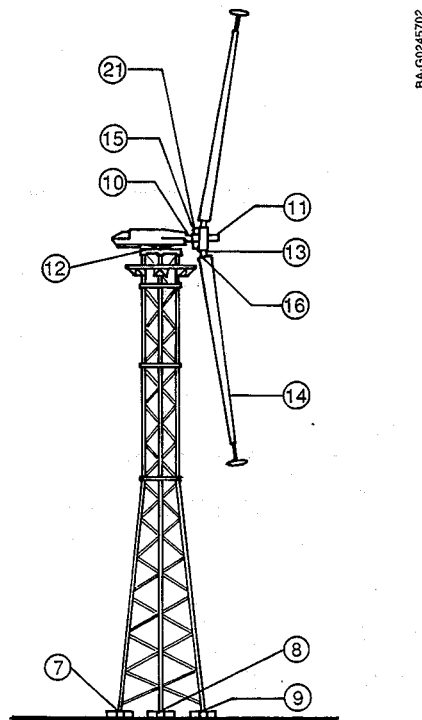


Figure 4-1. Illustration of example turbine

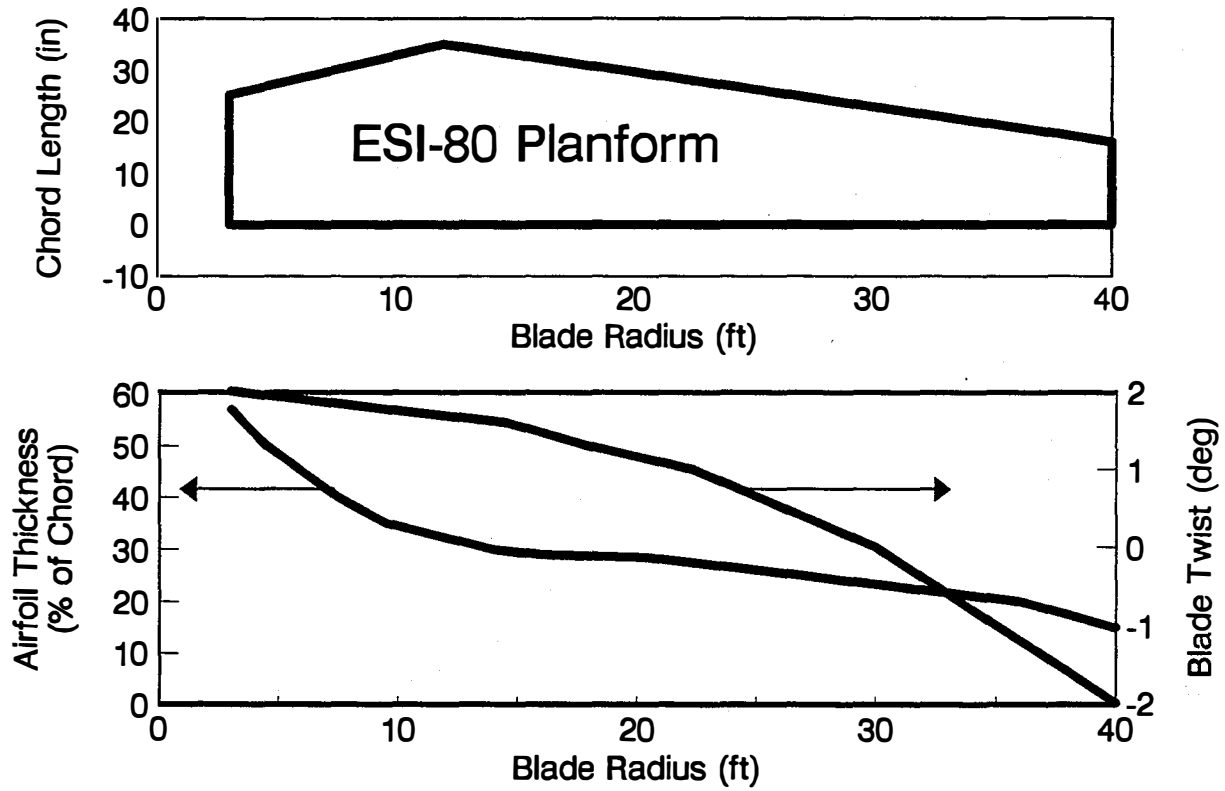


Figure 4-2. Turbine blade planform, thickness, and twist

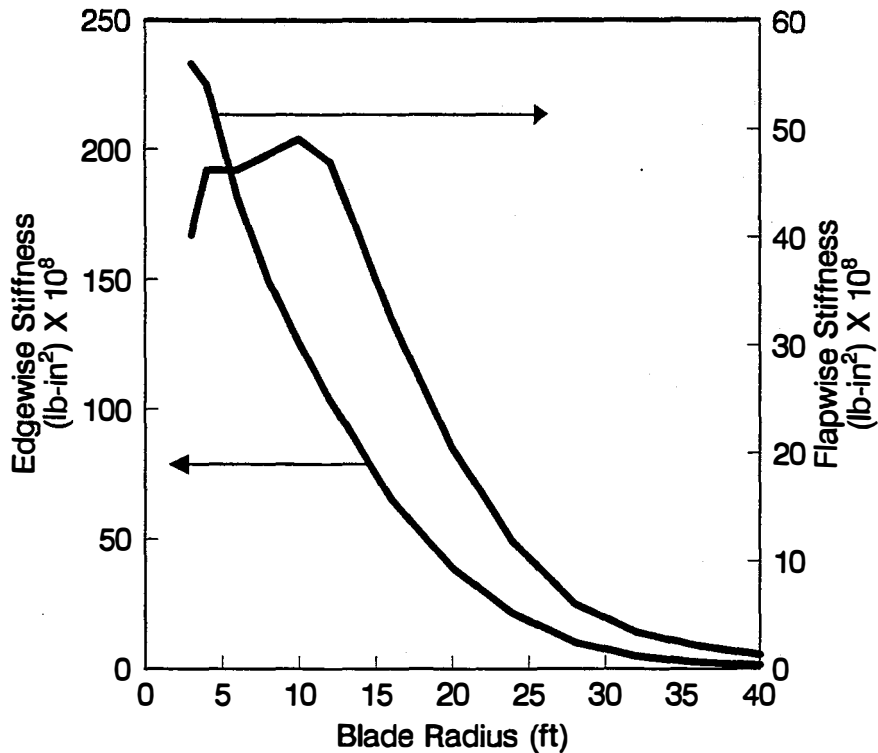


Figure 4-3. Turbine blade stiffness distribution

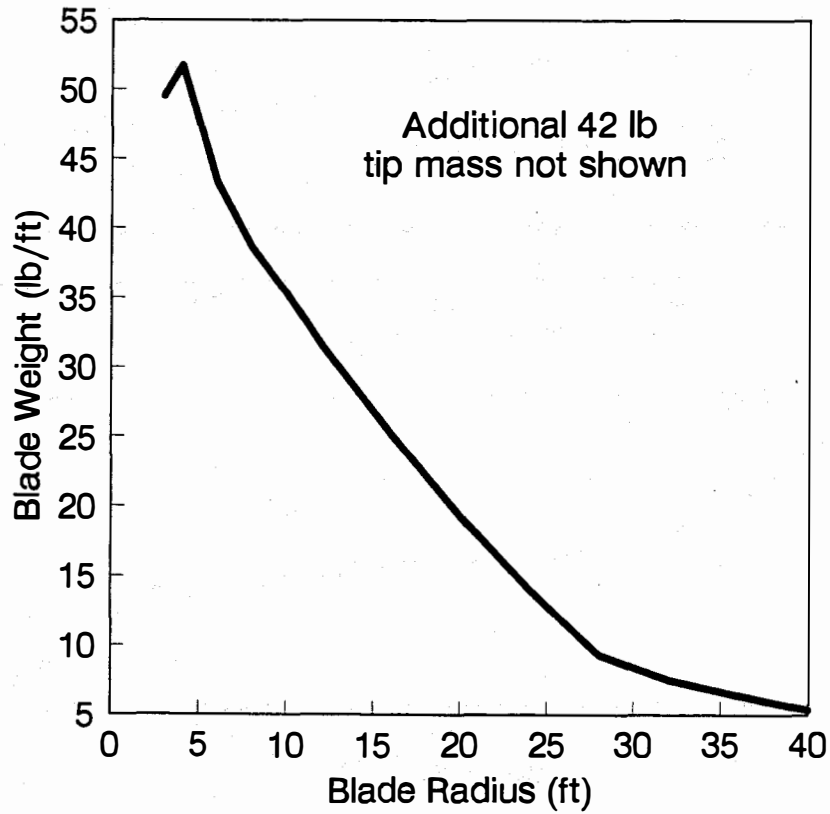


Figure 4-4. Turbine blade weight distribution

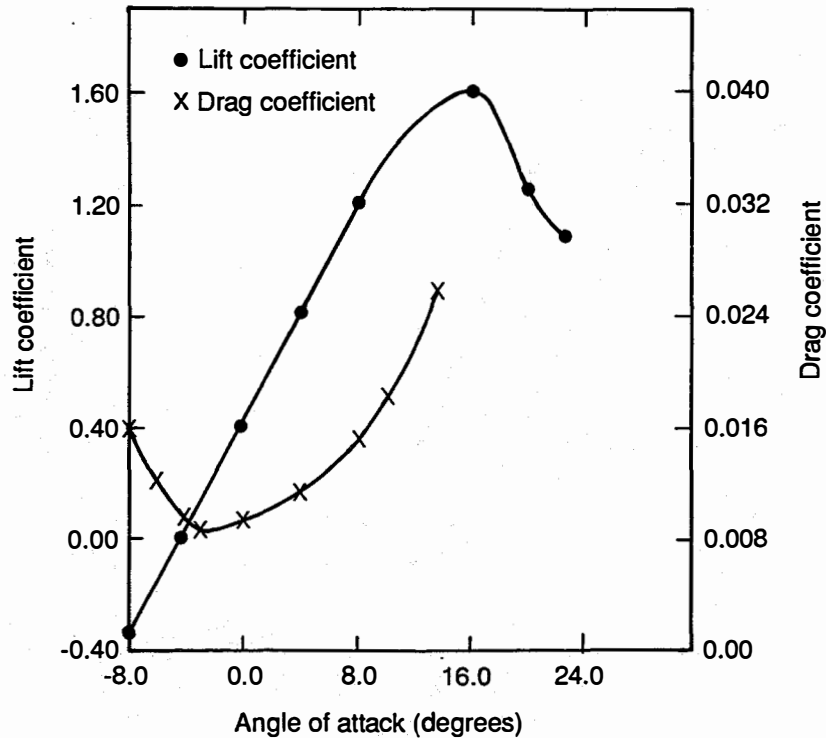


Figure 4-5. Lift profile for the LS(1) airfoil

Table 4-1. Example Turbine Specifications

Rated Power	250 kW
Rated Windspeed	20.3 m/s (45 mph) Rotor
Diameter	24.5 m (80 ft)
Rotor Type	Teetered - Stall Control
Rotor Orientation	Downwind
Blade Construction	Wood Epoxy Composite
Rotor Airfoil	NASA LS(1) 0417
Tip Speed	77.9 m/s (173 mph)
Cut-in Windspeed	5.9 m/s (13 mph)
Rotor RPM	60 RPM
Generator RPM	1,800 RPM
Generator Type	300-kW Induction, 3 Phase
Gearbox	Planetary
Hub Height	24.9 m (81.5 ft)
Tower	Open - Truss
Pitch Control	None
Yaw	Passive
Overspeed Control	Tip Vanes
Total System Weight	9,750 Kg (21,500 lbs)
Coning	7 deg

Rotating Natural Frequencies

Teeter	1 Hz
First Symmetrical Flapwise	2 Hz
Second Symmetrical Flapwise	7.8 Hz
First Edgewise	5.9 Hz
Second Antisymmetrical Flapwise	12 Hz

5.0 Preparation of Input Data

Appendix A, page 2 shows the basic input file to Module 1 for the turbine. We will now describe the basis for choosing these inputs for each variable.

ALENTH (ft)

The distance from the tower centerline or yaw axis to the center of the hub (point where the two blades intersect) is 6.79 ft. Note that this is not necessarily the distance from the yaw axis to the teeter hinge. That distance is $ALENTH + UNDLG$.

ALPHA ϕ (deg)

This variable represents the angle from the chord line to the zero lift line. This angle can be found from examination of the lift curve for the LS(1) airfoil shown in Figure 4-5 and represents the angle of attack at which C_L is equal to zero. It is equal to -4 deg for this airfoil. We assume a constant $ALPHA\phi$ for the entire blade span. Provisions for changing $ALPHA\phi$ with blade span are not provided in this version of STRAP.

CHI (deg)

The angle CHI is the rotor shaft tilt, set equal to zero for this turbine. See Wright, Buhl, and Thresher (1988) for more details.

CSUBMA

This variable represents the airfoil pitching moment coefficient, set equal to 0.015 for this airfoil. For more details, see Wright, Buhl, and Thresher (1988), page 13. This input is not very important for calculation of flap-bending moments.

DRGFRM

This input variable is used in the calculation of airfoil drag coefficient, as given in Wright, Buhl, and Thresher (1988), equation 4-7, page 85. It can be found by fitting a second-order equation of the form given in Wright, Buhl, and Thresher (1988) to the airfoil C_D versus C_L curve. For this airfoil, the value for $DRGFRM$ was calculated as 0.0032.

HUBHT (ft)

The hub height of this turbine is 80 ft.

BETA ϕ (deg)

The precone angle for this rotor is 7 deg.

BLSHNK (ft)

$BLSHNK$ is defined as the length of blade shank measured from the blade root to the point where the airfoil section begins. A nonzero value for $BLSHNK$ must always be input to this code. If the particular rotor being analyzed has a zero blade shank, set $BLSHNK$ to some small number (.01). Figure 5-1 clarifies the definition of this variable. The main purpose of this variable is to identify the portion of the blade

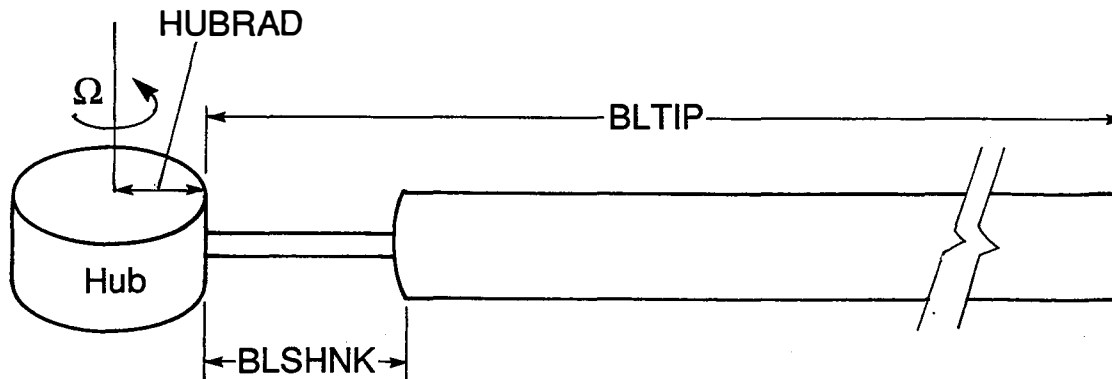


Figure 5-1. Illustration of rotor geometry

not producing aerodynamic lift. For this case, $BLSHNK = 0.8$ ft. Input of a zero for $BLSHNK$ will cause an error in code execution.

BLTIP (ft)

This variable is the blade length, as measured from the blade root to the tip. This value will not, in general, be equal to the rotor radius (see Figure 5-1). For this turbine, $BLTIP$ is set equal to 38 ft because we have a hub radius of 2 ft. The hub is not considered as part of the blade.

HUBRAD (ft)

$HUBRAD$ is the distance from the axis of rotation to the blade root. For this rotor, $HUBRAD$ is set equal to 2 ft. Figure 5-1 clarifies this input variable. In this version of the code the parameter $HUBRAD$ does not need to be set to zero as in the previous version of FLAP (see Wright, Buhl, and Thresher, 1988, p 63).

KSHADW

This variable is used in the tower shadow model (see Wright, Buhl, and Thresher, 1988, pages 7, 88). It represents the number of oscillations in the tower shadow region. For this turbine, it is set to 3 because we have a three-legged truss tower.

NBLADS

The number of blades is set to 2.

NPANEL

This variable represents the number of points on the blade, beginning at the root and ending at the tip, in which distributed blade properties are input. We input properties at 11 points along the blade. The maximum value for $NPANEL$ is 11.

OMEGA (RPM)

The rotor speed is 60 RPM.

PHIAMP (deg)

This variable is set to zero because we are not performing a yaw motion solution. See Ref. 4 for more details. This variable defines the yaw-motion amplitude in the time-dependent yaw solution.

PHIOMG (deg/sec)

This variable is also set to zero. This variable defines the steady yaw rate in the trim solution or the maximum yaw rate in a yaw-motion, time-dependent solution. (See Wright, Buhl, and Thresher, 1988).

PHI ϕ (deg)

This variable is set to +13 deg because the steady yaw error for this data case was 13 deg (see Wright, Buhl, and Thresher, 1988). See Wright and Butterfield (1992) for yaw error definition.

PSIZER

This variable represents the tower shadow half-width. The tower shadow is modeled as a pie-shaped sector centered about the tower centerline. The value of PSIZER was arbitrarily selected as 20 deg for this machine. For more details, see Wright, Buhl, and Thresher (1988), page 7.

SHERXP

The power law windshear relation given in Wright, Buhl, and Thresher (1988), page 7, had an exponent equal to 0.164 for this case. This value was determined from analysis of anemometer data at three heights, as described in Wright and Butterfield (1992).

THETAP

This variable represents the angle between the principal flapwise bending plane and the cone of rotor rotation. If the blade elastic flapping motion is considered to occur perpendicular to the rotor plane, then THETAP should be set to zero. If flapping motion is considered to occur about the section chord line at the blade root, then THETAP should be set to the angle between this chord line and the rotor cone of rotation.

For a single code run, the orientation of flapping is constant; it does not change from blade station to station because of changes in pretwist. To change the flapping direction for other stations, the code must be rerun with appropriate values of THETAP. All load and deflection results are referenced to this one axis system (the X_p , Y_p , Z_p axes [Wright, Buhl, and Thresher, 1988]). For this case, we set THETAP to 4 deg to reflect flapping about the root chord line. For more details, see Wright, Buhl, and Thresher (1988), pages 13 and 86.

THETAT

This variable represents the difference in twist between the reference station and the tip. Because the reference station was selected at the root, this twist difference is 4 deg.

WEIGHT

We wanted to include the 42-lb tip weight in the blade's distributed weight input (provisions for inputting a concentrated tip weight are not included). We placed points at 36.0, 36.2, and 38.0 ft.

Distributed property data are interpolated in Module 1 to form a set of evenly spaced data. Between each value of XLEFT, the property data are being linearly interpolated. The point 36.2 was chosen to allow the weight to change rapidly.

The other values chosen for WEIGHT were based on the blade's weight distribution, shown in Figure 4-4.

AEIARE ($\times 10^6$ lb-ft²)

Figure 4-3 shows the stiffness distributions for this airfoil. Input of values from this plot result in overprediction of the blade's flapwise bending frequencies. We deliberately reduced these values uniformly along the blade so that the predicted flapwise frequencies will agree with test data. It is known that the first symmetric flapwise frequency is approximately 2 Hz. Values taken directly from Figure 4-3 result in overprediction of the frequencies. The user can now try different mass and stiffness distributions, run Module 1, and see what effect these changes have on predicted frequencies. We deliberately reduced the blade's stiffness distribution.

We did not adjust the blade's weight distribution because this parameter is so important for correct calculation of blade centrifugal and gravitational loads.

AIEMASS, AIFMASS, AOFFST, AESBAC

These input variables are set to zero. We did not perform a thorough investigation of the effects of these parameters on blade loads and response. The variables AOFFST and AESBAC may have some effect on the torsional moments predicted by STRAP. We do not think these input parameters have a big effect on flapwise bending moments, however.

ACHORD

This variable is input directly from the information given in Figure 4-2.

ATWIST

This variable represents the blade's built-in twist distribution. It is input from the information given in Figure 4-2. It is important to note that these values must be adjusted in order to obtain zero twist at the tip; i.e., ATWIST(11) = 0. The input variable ATWIST has the blade tip as its zero reference point instead of the 75% span, as seen in Figure 4-2.

ACLALF, ACDZER

ACLALF is the lift curve slope. It is allowed to vary along the span. These values were input from information on the LS(1) airfoil.

ACDZER is the drag coefficient at zero lift. It is used in the calculation of drag coefficient (Wright, Buhl, and Thresher, 1988). It is also allowed to vary along the span.

NUMSCN, TIMINC, MSTAT, STA

These four variables provide information necessary for input of turbulent wind fluctuations to STRAP. NUMSCN was input as 8,192 to reflect 8,192 lines of data in the turbulent windspeed file. TIMINC is set to .041667 because the wind data are sampled at 24 Hz. The number of blade stations at which windspeed data are input is two. Note: The hub center windspeed data will be read in also. MSTAT represents the number of blade stations outboard of the hub center at which turbulent wind input are to be read. STA gives the positions (as measured from the center of rotor rotation) of these points, here set to 20 and 40 ft, which is 50% and 100% of the rotor radius. These inputs are seen in the input file in Appendix A.

6.0 Preparation of Turbulent Wind Input Data

If the user is not interested in running a case with turbulent windspeed input, then this section can be skipped. The STRAP code can be run for deterministic cases only by simply choosing the (Q) option after the trim solution.

In order to run a turbulent windspeed case, a separate file of turbulent windspeed data is needed for input to STRAP. It is assumed that the user must run a separate turbulence wind simulation model such as Veers (1988). Some specific details on the VEERS model will now be given as well as the structure of the turbulence input file to STRAP.

We recommend the use of Veers (1988) for generation of turbulent windspeed input for this rotor. We will henceforth refer to Veers (1988) as the VEERS Three-Dimensional Wind Simulation or, simply, the VEERS model. The reason we recommend this code is that it can be used to generate turbulent windspeed data needed for both blades of a two-bladed teetering hub rotor. With this model, a full three-component field of turbulence is not calculated, only the longitudinal, "along wind," component.

This simulation method is used to obtain rotationally sampled windspeeds (Veers, 1988). We simulate the windspeed at five points of the rotating two-bladed rotor: the 50% rotor radius location on blade 1, the 100% rotor radius for blade 1, the 50% rotor radius for blade 2, the 100% rotor radius for blade 2, and the windspeed for the center of rotor rotation.

In the VEERS code, various spectral models for the fixed-point power spectral density calculations can be chosen (Veers, 1988). We usually choose the Solari model. The coherence model is the exponential model with some modifications (Veers, 1988). Input to the model include mean windspeed, rotor speed, number of blades, turbulence intensity, terrain surface roughness, and coherence decrement. For more information, see Veers (1988).

Another input to this model is the number of equally spaced azimuth points in which windspeed data are to be simulated. This parameter will dictate the sampling rate of the generated, rotationally sampled windspeed data. From information that we have obtained from Winkelaar (1991), we usually set this parameter to a high value, in our case, 24 points around the azimuth. This will result in a turbulent windspeed file consisting of six columns of data generated at 24 Hz. The time increment between digitized data is, thus, approximately 0.0417 sec.

In the output file from the VEERS model, the first column is rotor azimuth angle (deg), the second is windspeed at 50% rotor radius on blade 1, the third is windspeed at 100% rotor radius on blade 1, the fourth is windspeed at 50% rotor radius on blade 2, the fifth is windspeed at 100% rotor radius on blade 2, and the sixth is the center of hub windspeed. The file contains 8,192 lines of data sampled at 24 Hz. This file represents 341 sec of real-time wind data. The number of lines of data in the file can be set in the VEERS model input file. We usually want this number to be some power of two (2^{13} in this case) so that we can later perform power spectral density calculations on the computed loads and response output. The STRAP code will generate corresponding loads and response output corresponding to each line of windspeed data. An example output file from VEERS is shown in Appendix F.

7.0 Input of Turbulence Data to STRAP

At this time, we will describe how these turbulent windspeed data are read into STRAP and how they are used.

In order to run a turbulent windspeed case, the STRAP code must first be run in order to determine a trim solution. The code then uses the computed values of blade deflection and velocity at zero azimuth (blade at 12 o'clock) as initial conditions for this process. An example interactive execution will be shown in the next section.

One major change included in STRAP and not included in FLAP, version 2.01, is a subroutine named TRBCLC. After the code finishes with its trim solution calculations, it enters this subroutine, if so chosen by the user. It is in this subroutine that the turbulent windspeed data generated by Veers is read into STRAP.

We must now describe the two types of interpolation of this windspeed data that are occurring in STRAP. One type is with respect to rotor azimuth, and the other is along the blade radial station locations.

To describe the first type, we must remember that the STRAP code does not increment the rotor azimuth angle in equal azimuth steps. They can vary from point to point because of the use of the Euler Predictor Corrector numeric integration procedure (Wright, Buhl, and Thresher, 1988). The windspeed data generated from VEERS (Veers, 1988) is generated at equal time steps. The main purpose of subroutine TRBCLC is to interpolate the data in order to provide approximate wind input at intermediate rotor azimuth locations. Linear interpolation is used in this subroutine.

Appendix D gives a listing of STRAP2. Subroutine TRBCLC is on page D-92. Upon entering this subroutine, the user will be prompted for the name of the windspeed input file and asked whether the windspeed data are in English (ft/sec) or metric (m/sec) units. He or she is then asked for a file name that contains load and response predictions. These predictions are not written in the RESULTS.DAT file formed previously.

The code then reads one line of wind input data at a time. For the first pass through this subroutine, before the first line of windspeed data is read, the initial azimuth angle is set to zero degrees, and the initial values for the two windspeeds on each blade and the hub center windspeed are set equal to VHUB. The first line of data is then read. The azimuth angle is then incremented, and wind input are then determined by linear interpolation for this intermediate azimuth angle. The values calculated at this intermediate azimuth are stored in the variables VYNOW1(1), VYNOW1(2), VYNOW2(1), VYNOW2(2), and VYNOWH. These variables represent the interpolated values of the two velocities on each blade and the hub center wind velocity at the present azimuth angle. The two end-point values, used in the linear interpolation process, represent the last and most recent values of windspeed read from the input file. These values are represented by the variables VYSAV1(I,J), VYSAV2(I,J), and VYSAVH(I). The integer I can be either 0 or 1, corresponding to old or new. The integer J is equal to 1 or 2, corresponding to windspeed at the 50% or 100% radial location. The variable VYSAVH(I) is the hub center windspeed.

The five interpolated windspeeds are ultimately passed to a subroutine name WINDVL. In this subroutine, values of turbulent windspeed are calculated for intermediate blade radial stations.

The STRAP code calculates blade aerodynamic forces at 21 equally spaced blade stations, beginning at the root and ending at the tip. In this subroutine, we take the five values of windspeed passed from subroutine TRBCLC and calculate interpolated values for intermediate blade radial stations. We then have

an array containing 21 equally spaced windspeed inputs for a blade. These values are saved in the array DELTVY(I); see Appendix D, page D-89 for Subroutine WINDVL.

This process of azimuth interpolation in subroutine TRBCLC and blade radial station interpolation in WINDVL is continued for each iteration in the numeric process. The code will calculate teeter and elastic flap deflections, velocity, and accelerations at every single iteration point. The code calculates and saves loads only at those azimuth positions corresponding to the azimuth values read in from the turbulent windspeed file.

Results written to the user-designated output file include azimuth angle, teeter angle, blade 1 loads and deflection information, blade 2 loads and deflection information, and low-speed rotor shaft loads (in the rotating coordinate system) at the hub center. The user can modify the write statement in subroutine TRBCLC to write out all or part of these results. This file will contain the same number of lines of output results as the number of lines of data in the windspeed input file.

Some remarks should be made on proper execution of the trim solution portion of this process. The file produced by the VEERS model contains windspeed data derived from three parts: (1) the mean hub height windspeed, (2) the variation of windspeed as a result of windshear, and (3) the variation as a result of stochastic effects. The second part may be zero if the VEERS model is run with a zero windshear exponent.

Care must be taken to run the trim solution in STRAP using the correct values of VHUB and SHERXP. The variable VHUB, in Module 2 of STRAP, should be set to the value of hub height windspeed as calculated in the VEERS model. The mean windspeed input to the VEERS model is the mean windspeed at a 10-m height (Veers, 1988). This will not necessarily be equal to the hub height windspeed. Hub height may be at a different level. The equivalent hub height windspeed from the VEERS model can be calculated from a knowledge of the windspeed input at 10 m and the power law shear exponent. Once this value is known, STRAP SHOULD BE RUN WITH VHUB SET TO THIS VALUE IN FEET/SECOND.

If the VEERS model is run with a nonzero shear exponent, then the STRAP code should be run with SHERXP set to zero. The wind data generated by the VEERS model already contain the effect of windshear. Input of a nonzero SHERXP in STRAP will result in windshear overprediction. To be safe, we set SHERXP, at the beginning of subroutine TRBCLC, to zero.

An example interactive run of Modules 1 and 2 will now be given.

8.0 Example Interactive Run

8.1 Module 1—STRAP1

Please see Appendix A for a printout of this interactive run. Upon code execution, the user is first prompted for the input file name, here referred to as TURBN.DAT. Upon typing in the input file name, the user is prompted for the name of the run-data file name. This file is written by STRAP1 and contains the data to be read by STRAP2.

All the input data are echoed out to allow the user to check all inputs. At this point, the user is not able to change input during execution of Module 1. The changes must be made to the input file and then rerun.

After all the input data have been echoed out, the user is prompted for the name of a file for writing the blade's modeshape and natural frequency data. This file is different than the run-data file. The purpose of this modeshape file is to allow the user to examine resulting flapwise natural frequencies and mode-shapes. It is not read by STRAP2, and its only purpose is to examine frequency results.

Finally, the user is asked if he or she wants to process another data file; in which case, he or she answers yes or no. The code then responds by repeating the previous steps or stating "STRAP Terminated Normally." This signals the end of STRAP1 execution. A listing of this interactive session is shown in Appendix A. A listing of the run-data file and the modeshape file is also shown.

8.2 Module 2—STRAP2

Upon executing STRAP2, the user is presented with a menu of options shown in the interactive run listing in Appendix B. The options include (R)ead in a data file, (S)et up and run the model, (D)iagnostic run, (T)urbulence run, and (Q)uit.

The user first chooses the (R) option and reads in the run-data file produced by Module 1 (STRAP1). After this step, the user selects option (S). Upon selecting (S), a list of free variables will be given. The user is free to change any of these variables. At this stage, the user selects the number of degrees of freedom to be included in the analysis. The variable NSHAPS sets the number of modes (from 1 to 4) to be used in the analysis. We suggest that NSHAPS be set to at least 2 so that the rigid-body teeter mode and the first symmetric bending mode are used in the analysis. For the example turbine described earlier, final code runs should use a value of 4 for NSHAPS because the second symmetric bending mode at a frequency of 8 Hz (8 times the rotor rotation speed) gets highly excited by effects for this rotor such as windshear, tower shadow, and turbulence (Wright and Butterfield, 1992).

After changing any of the free variables, the user is given the option of changing the run parameters. We suggest changing the variables STEPMX and EUERR under certain conditions. See Veers (1988) for the definition of these two variables. The default parameters (STPEMX = 10, EUERR = 10) can be used for initial runs, when only one or two modes are used, at low windspeeds in nonstalled conditions. When more modes are used or in cases involving high angles of attack (stalled flow), severe tower shadow input teeter stop impacts, etc., these parameters should be set to 1. Although the code will take longer to run, decreasing these input parameters will help the code to converge to a solution. The small step size is necessary when the blade is encountering highly nonlinear operating conditions, such as stalled flow, and abrupt changes because of tower shadow or teeter stop impacts.

For high-windspeed cases, the code may take 20 to 30 trim iterations before a solution is reached. This occurs because of the decrease in blade aerodynamic damping that result from high angle-of-attack conditions.

We also suggest setting the Trace Flag in the set of run parameters to TRUE. This allows printout of blade deflection, velocity, acceleration, and other items in the RESULTS.DAT file. From this information, the user can monitor problems that may be occurring during code execution.

Another variable in this list of parameters is NYAW. If the user wants to run a time-dependent yaw solution, NYAW should be set to some integer number greater than zero. The variable PHIAMP in the free variable list must also be greater than zero (see Wright, Buhl, and Thresher, 1988).

NYAW represents the number of rotor revolutions to be performed during the yaw solution. The code will first compute a trim solution and then perform a transient analysis using the time-dependent yaw solution. The solution is completed at the end of NYAW revolutions.

After reviewing and/or changing the set of run parameters, the user is prompted for the name of the RESULTS.DAT file. This file will contain the rotor teeter, blade deflection, slope, and velocity results as well as all the blade's force and moment values. These results are given as a function of blade azimuth angle as well as blade radial station.

An option we have added to STRAP is the ability to input initial values for the blade's teeter and deflection data. The number of required input will depend on the value for NSHAPS, set in the free variable list. These initial values are assumed to be for a blade azimuth angle of 270 deg.

We have found this capability useful for speeding convergence for cases involving stall or other conditions requiring a long run time. The user can first run the code with the TRACEF flag (in the run parameters) set to TRUE with the code default initial conditions. After 10 trim iterations, the user can stop execution of the code. The user can then examine the RESULTS.DAT file and find the last values for the blade deflections at the azimuth angle of 270 deg. The code can then be rerun with these values used as initial values for the rotor teeter and blade deflection values. These steps will help speed convergence for cases involving high windspeeds or other nonlinear phenomena. Once the code obtains a satisfied trim condition, the code prompts the user for a Fourier analysis of result data.

The nine result items are then listed, and the user chooses which item (one at a time) to be Fourier analyzed. The code then determines Fourier cosine and sine coefficients and writes them in table form to the RESULTS.DAT file. A RESULTS.DAT file corresponding to the interactive run is shown in Appendix B.

After this, the name of a shaft load output file is asked for. The shaft loads for the trim runs are written to a separate file. A printout of this file for this interactive run is also shown in Appendix B.

After this, the user is asked if he or she wants to make another run with these data. If the answer is yes, the previous steps are repeated. If the answer is no, the user is returned to the operations menu.

One option that has not been described is (D). This option can only be performed after a trim solution has been performed. Upon execution of this option, items such as angle of attack, lift, and drag will be written to a file named DIAGNOS.DAT. This information is presented for various blade stations as well as rotor azimuth positions.

The other option to be described is (T). This option must also be chosen only after a trim solution has been performed. In this case, we can see that a turbulence analysis was performed. The turbulent windspeed input file was named WIND-2.DAT. These windspeed data were in metric units (meters/seconds). The turbulence load output file was named LOADS-2.DAT. If the user wants to bypass the turbulence analysis, then the (Q) option should be chosen before the (T) option. This will halt code execution.

9.0 Conclusions

An overview of the NREL (formerly SERI) Teetering Rotor Analysis Program (STRAP) has been given. New input to the code have been described. An example two-bladed teetering hub wind turbine was described. Preparation of code input and an example code interactive run for this turbine were shown. Code and interactive run listings are given in the appendices.

10.0 References

Veers, P. S., 1988, *Three-Dimensional Wind Simulation*, SAND 88-0152, Sandia National Laboratories, Albuquerque, New Mexico.

Winkelaar, D., 1991, *First Three-Dimensional Wind Simulation and the Predictions of Stochastic Blade Loads*, presented at the Tenth ASME Wind Energy Symposium, Houston, Texas.

Wright, A. D., and Butterfield, C. P., 1992, *The NREL Teetering Hub Rotor Code: Final Results and Conclusions*, presented at the Eleventh ASME Wind Symposium, Houston, Texas.

Wright, A. D., Buhl, M. L., and Thresher, R. W., 1988, *FLAP Code Development and Validation*, SERI/TR-217-3125, National Renewable Energy Laboratory, Golden, Colorado.

11.0 Acknowledgments

The author would like to thank C. P. Butterfield of the National Renewable Energy Laboratory for assisting in analysis and interpretation of test results from the example turbine discussed in this report. Thanks also goes to Dr. Paul Veers of Sandia National Laboratories for many helpful suggestions on the correct use of the turbulence model. Thanks go to Lynn Starr and Rick Clyne for preparation and editing of this manuscript, respectively.

Appendix A

Module 1 Input File, Interactive Run Listing, and Output Files

TURBN.DAT

DATA INPUT FOR THE EXAMPLE TWO BLADED, DOWNWIND
80 FT DIAM. TEETERING ROTOR

ALENTH	6.79										
ALPHAO	-4.0										
BETA0	7.0										
BLSHNK	0.80										
BLTIP	38.00										
CHI	0.										
CSBMAC	0.015										
DRGFRM	0.0032										
HUBHT	80.										
HUBRAD	2.0										
KSHADW	1										
NBLADS	2										
NPANEL	11										
OMEGA	60.										
PHIAMP	0.0										
PHIOMG	0.0										
PHIO	13.0										
PSIZER	20.										
SHERXP	0.164										
THETAP	2.00										
THETAT	4.0										
TSUBP	0.05										
TSUB0	0.05										
HUBMAS	55.90										
UNDSLK	0.75										
HUBDIS	0.24										
VHUB	33.64										
DELTA3	00.0										
BETA1	0.0										
BETA2	0.0										
K0	000000.0										
K1	000000.0										
K2	000000.0										
CDAMP	0000.										
XLEFT	0.0	0.80	1.00	4.00	10.00	18.00	22.00	26.00	36.00	36.20	38.00
WEIGHT	50.0	50.00	50.00	43.3	31.6	19.3	14.0	9.35	5.39	28.30	28.20
AEIARE	38.90	38.90	38.89	22.50	13.33	4.22	2.48	1.45	0.289	0.280	0.135
AIEMAS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AIFMAS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AOFFST	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ACHORD	1.00	1.00	2.083	2.333	2.916	2.464	2.238	2.012	1.560	1.549	1.333
ATWIST	0.0	0.0	4.0	3.9	3.8	3.2	2.8	2.1	0.20	0.10	0.00
ACLALF	0.0	0.0	4.55	4.75	4.85	4.95	4.95	5.20	5.18	5.18	5.24
ACLMAX	0.0	0.0	1.0	1.1	1.2	1.3	1.5	1.6	1.6	1.6	1.6
ACDZER	0.0	0.0	0.0083	0.0083	0.0083	0.0083	0.0083	0.0083	0.0083	0.0083	0.0083
AESBAC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NUMSCN	100										
TIMINC	0.041667										
MSTAT	2										
STA	20.	40.									

Program For Analysis Of a 2-Bladed Teetering Hub Rotor
Dynamic Response and Loads Analysis

MODULE 1- STRAP1

Subroutine INPUT: Input and process new blade
and turbine data.

Enter name of input data file >
turbn.dat

Enter name of output data file >
runb.dat

TURBN.DAT

DATA INPUT FOR THE EXAMPLE TWO BLADED, DOWNWIND
80 FT DIAM. TEETERING ROTOR

Parameter values:

ALENTH =	6.79000	NPANEL =	11
ALPHAO =	-4.00000	OMEGA =	60.00000
BETA0 =	7.00000	PHIO =	13.00000
BLSHKN =	0.80000	PHIAMP =	0.00000
BLTIP =	38.00000	PHIOMG =	0.00000
CHI =	0.00000	PSIZER =	20.00000
CSUBMA =	0.01500	SHERXP =	0.16400
DRGFRM =	0.00320	THETAP =	2.00000
HUBHT =	80.00000	THETAT =	4.00000
HUBRAD =	2.00000	TSUBO =	0.05000
KSHAOW =	1	TSUBP =	0.05000
NBLADS =	2	VHUB =	33.64000
HUBMAS =	55.90000	UNDSLK =	0.75000
HUBDIS =	0.24000	DELTA3 =	0.00000
BETA1 =	0.00000	BETA2 =	0.00000
K0 =	0.00000	K1 =	0.00000
K2 =	0.00000		
CDAMP =	0.00000		
NUMSCH =	100		
TIMINC =	0.04166		
MSTAT =	2		

Hit <Enter> to continue...

XLEFT	WEIGHT	AEIARE	AIEMAS	AIFMAS	AOFFST
0.00000	50.00000	38.90000	0.00000	0.00000	0.00000
0.80000	50.00000	38.90000	0.00000	0.00000	0.00000
1.00000	50.00000	38.89000	0.00000	0.00000	0.00000
4.00000	43.30000	22.50000	0.00000	0.00000	0.00000
10.00000	31.60000	13.33000	0.00000	0.00000	0.00000
18.00000	19.30000	4.22000	0.00000	0.00000	0.00000
22.00000	14.00000	2.48000	0.00000	0.00000	0.00000
26.00000	9.35000	1.45000	0.00000	0.00000	0.00000
36.00000	5.39000	0.28900	0.00000	0.00000	0.00000

36.20000	28.30000	0.28000	0.00000	0.00000	0.00000
38.00000	28.20000	0.13500	0.00000	0.00000	0.00000

Hit <Enter> to continue...

ACHORD	ATWIST	ALCALF	ACLMAX	ACDZER	AESBAC
1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2.08300	4.00000	4.55000	1.00000	0.00830	0.00000
2.33300	3.90000	4.75000	1.10000	0.00830	0.00000
2.91600	3.80000	4.85000	1.20000	0.00830	0.00000
2.46400	3.20000	4.95000	1.30000	0.00830	0.00000
2.23800	2.80000	4.95000	1.50000	0.00830	0.00000
2.01200	2.10000	5.20000	1.60000	0.00830	0.00000
1.56000	0.20000	5.18000	1.60000	0.00830	0.00000
1.54900	0.10000	5.18000	1.60000	0.00830	0.00000
1.33300	0.00000	5.24000	1.60000	0.00830	0.00000

Hit <Enter> to continue...

STA-
20,000
40,000

Hit <Enter> to continue...

Enter name of modeshape and frequency file >
MODESb.DAT

Data for MODULE 2 have been written to RUNB.DAT.

Do you want to process another data file? (Y,=N) >

STRAP1 terminated normally.

THIS IS THE MODESHAPE AND FREQUENCY FILE PRODUCED BY MODULE 1:

FREQUENCIES = (RADIAN/SEC)

	6.28319	15.4516	30.0608	50.6140
Modeshapes:				
1	0.499915E-01	0.000000	-0.128996E-01	0.000000
11	0.974451E-01	0.479990E-03	-0.247922E-01	-0.470760E-03
21	0.144827	0.226373E-02	-0.364379E-01	-0.172893E-02
31	0.192138	0.560398E-02	-0.480739E-01	-0.376290E-02
41	0.239369	0.105679E-01	-0.594390E-01	-0.668386E-02
51	0.286509	0.171897E-01	-0.699322E-01	-0.105695E-01
61	0.333544	0.255666E-01	-0.787526E-01	-0.153675E-01
71	0.380458	0.359055E-01	-0.850203E-01	-0.208500E-01
81	0.427239	0.485312E-01	-0.878786E-01	-0.266104E-01
91	0.473877	0.638611E-01	-0.865778E-01	-0.320938E-01
101	0.520367	0.823587E-01	-0.805395E-01	-0.366515E-01
111	0.566707	0.104471	-0.694031E-01	-0.396127E-01
121	0.612903	0.130561	-0.530526E-01	-0.403623E-01
131	0.658964	0.160841	-0.316252E-01	-0.384173E-01
141	0.704903	0.195319	-0.550104E-02	-0.334936E-01
151	0.750740	0.233762	0.247261E-01	-0.255528E-01
161	0.796495	0.275687	0.582962E-01	-0.148218E-01
171	0.842191	0.320391	0.943502E-01	-0.177524E-02
181	0.887848	0.367023	0.132015	0.129271E-01
191	0.933486	0.414711	0.170507	0.285568E-01
201	0.979118	0.462746	0.209258	0.444972E-01

First Derivative of Modeshape

1	0.249957E-01	0.000000	-0.644979E-02	0.000000
11	0.249564E-01	0.560284E-03	-0.614713E-02	-0.466659E-03
21	0.249195E-01	0.133761E-02	-0.613075E-02	-0.858673E-03
31	0.248804E-01	0.218301E-02	-0.609334E-02	-0.129346E-02
41	0.248359E-01	0.304435E-02	-0.581596E-02	-0.178889E-02
51	0.247841E-01	0.393412E-02	-0.515822E-02	-0.229707E-02
61	0.247245E-01	0.490171E-02	-0.404835E-02	-0.273349E-02
71	0.246576E-01	0.601012E-02	-0.247334E-02	-0.300163E-02
81	0.245847E-01	0.731711E-02	-0.469084E-03	-0.301281E-02
91	0.245076E-01	0.886087E-02	0.188952E-02	-0.270159E-02
101	0.244289E-01	0.106501E-01	0.449843E-02	-0.203648E-02
111	0.243511E-01	0.126585E-01	0.723435E-02	-0.102597E-02
121	0.242770E-01	0.148238E-01	0.996460E-02	0.279918E-03
131	0.242092E-01	0.170511E-01	0.125570E-01	0.179279E-02
141	0.241500E-01	0.192212E-01	0.148897E-01	0.339361E-02
151	0.241012E-01	0.212022E-01	0.168611E-01	0.494507E-02
161	0.240640E-01	0.228670E-01	0.183998E-01	0.630857E-02
171	0.240384E-01	0.241143E-01	0.194743E-01	0.736578E-02
181	0.240236E-01	0.248943E-01	0.201031E-01	0.804484E-02
191	0.240176E-01	0.252389E-01	0.203642E-01	0.835122E-02
201	0.240166E-01	0.252963E-01	0.204055E-01	0.840306E-02

Second Derivative of Modeshape

1	0.000000	0.195118E-03	0.301220E-03	-0.302912E-03
11	0.000000	0.370241E-03	0.528635E-04	-0.210331E-03
21	0.000000	0.435249E-03	-0.875183E-05	-0.212279E-03
31	0.000000	0.450356E-03	0.672519E-04	-0.246663E-03
41	0.000000	0.457658E-03	0.236950E-03	-0.270386E-03
51	0.000000	0.483482E-03	0.461618E-03	-0.256911E-03
61	0.000000	0.540726E-03	0.707726E-03	-0.193842E-03
71	0.000000	0.631208E-03	0.946945E-03	-0.804900E-04
81	0.000000	0.748010E-03	0.115614E-02	0.745562E-04

91	0.000000	0.877819E-03	0.131739E-02	0.255854E-03
101	0.000000	0.100328E-02	0.141794E-02	0.443537E-03
111	0.000000	0.110532E-02	0.145026E-02	0.615745E-03
121	0.000000	0.116554E-02	0.141201E-02	0.751052E-03
131	0.000000	0.116849E-02	0.130605E-02	0.830892E-03
141	0.000000	0.110407E-02	0.114043E-02	0.841988E-03
151	0.000000	0.969868E-03	0.928408E-03	0.778785E-03
161	0.000000	0.773475E-03	0.688436E-03	0.645870E-03
171	0.000000	0.534854E-03	0.444163E-03	0.460409E-03
181	0.000000	0.288682E-03	0.224436E-03	0.254567E-03
191	0.000000	0.866866E-04	0.633004E-04	0.779450E-04
201	0.000000	0.247582E-13	0.170329E-13	0.226193E-13

TURBN.DAT

DATA INPUT FOR THE EXAMPLE TWO BLADED, DOWNWIND
80 FT DIAM. TEETERING ROTOR

ALENTH	ALPHA0	BETA0
6.7900000000E+00	-4.0000000000E+00	7.0000000000E+00
BLSHKN	BLTIP	CHI
8.0000000000E-01	3.8000000000E+01	0.0000000000E+00
CSUBMA	DRGFRM	HUBHT
1.5000000000E-02	3.2000000000E-03	8.0000000000E+01
HUBRAD	OMEGA	PHI0
2.0000000000E+00	6.0000000000E+01	1.3000000000E+01
PHIAMP	PHIOMG	PSIZER
0.0000000000E+00	0.0000000000E+00	2.0000000000E+01
SHERXP	THETAP	THETAT
1.6400000000E-01	4.0000000000E+00	4.0000000000E+00
TSUB0	TSUBP	VHUB
5.0000000000E-02	5.0000000000E-02	3.3640000000E+01

KSHADW	NBLADS	NSHAPS	NPTS
1	2	1	21

DELTA3= 0.00

UNDSLK= 0.75

HUBMAS= 55.90

HUBDIS= 0.24

BETA1 = 0.00

BETA2 = 0.00

NUMSCN= 100

TIMINC= 0.04166

MSTAT = 2

STAC(I)-

2.000E+01 4.000E+01

CLALFA-

0.0000000E+00	4.6100000E+00	4.7366670E+00	4.7783330E+00
4.8100000E+00	4.8416670E+00	4.8675000E+00	4.8912500E+00
4.9150000E+00	4.9387500E+00	4.9500000E+00	4.9500000E+00
5.0000000E+00	5.1187500E+00	5.1988000E+00	5.1950000E+00
5.1912000E+00	5.1874000E+00	5.1836000E+00	5.1800000E+00
5.2400000E+00			

CLMAX -

0.000000E+00	1.030000E+00	1.093333E+00	1.128333E+00
1.160000E+00	1.191667E+00	1.217500E+00	1.241250E+00
1.265000E+00	1.288750E+00	1.350000E+00	1.445000E+00
1.520000E+00	1.567500E+00	1.600000E+00	1.600000E+00
1.600000E+00	1.600000E+00	1.600000E+00	1.600000E+00
1.600000E+00			

CDZERO-

0.000000E+00	8.300000E-03	8.300000E-03	8.300000E-03
8.300000E-03	8.300000E-03	8.300000E-03	8.300000E-03
8.300000E-03	8.300000E-03	8.300000E-03	8.300000E-03
8.300000E-03	8.300000E-03	8.300000E-03	8.300000E-03
8.300000E-03	8.300000E-03	8.300000E-03	8.300000E-03
8.300000E-03			

CHORD-

1.000000E+00	2.158000E+00	2.316333E+00	2.498183E+00
2.682800E+00	2.867416E+00	2.836900E+00	2.729550E+00
2.622201E+00	2.514851E+00	2.407501E+00	2.300150E+00
2.192800E+00	2.085449E+00	1.984879E+00	1.898999E+00
1.813119E+00	1.727239E+00	1.641359E+00	1.554500E+00
1.333000E+00			

ECNTFN-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00			

ESUBAC-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00			

THETA0-

1.396263E-01	7.033677E-02	7.144214E-02	7.205301E-02
7.260569E-02	7.315838E-02	7.513642E-02	7.762350E-02
8.011059E-02	8.259769E-02	8.552112E-02	8.883726E-02
9.320059E-02	9.900383E-02	1.049642E-01	1.112648E-01
1.175655E-01	1.238661E-01	1.301667E-01	1.370083E-01
1.396263E-01			

CKBEND-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	1.008339E+02	0.000000E+00	-4.032225E+00
0.000000E+00	0.000000E+00	1.204145E+02	0.000000E+00
0.000000E+00	-4.032225E+00	0.000000E+00	2.843365E+01

CKTOMG-

5.108354E+00	0.000000E+00	1.135412E-01	0.000000E+00
0.000000E+00	9.480934E-01	0.000000E+00	1.021381E-01
1.135412E-01	0.000000E+00	6.232625E-01	0.000000E+00
0.000000E+00	1.021381E-01	0.000000E+00	7.142509E-02

CKTGRV-

0.000000E+00	5.1426520E-02	0.000000E+00	-7.1287890E-03
5.1426520E-02	0.000000E+00	1.4469950E-02	0.000000E+00
0.000000E+00	1.4469950E-02	0.000000E+00	4.9755630E-03
-7.1287890E-03	0.000000E+00	4.9755630E-03	0.000000E+00
CKQL0D-			
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
CMMASS-			
5.6169610E+00	0.000000E+00	-3.9048480E-07	0.000000E+00
0.000000E+00	5.7911190E-01	0.000000E+00	4.1807690E-07
-3.9048480E-07	0.000000E+00	1.6048230E-01	0.000000E+00
0.000000E+00	4.1807690E-07	0.000000E+00	1.2199980E-02
CKTCRL(1,K,L)-			
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
CKTCRL(2,K,L)-			
3.6609590E-02	0.000000E+00	3.9628800E-03	0.000000E+00
0.000000E+00	9.3012170E-03	0.000000E+00	1.3157830E-03
3.9628800E-03	0.000000E+00	5.7625750E-03	0.000000E+00
0.000000E+00	1.3157830E-03	0.000000E+00	6.7067570E-04
CKTCRL(3,K,L)-			
0.000000E+00	3.4063490E-03	0.000000E+00	8.4122110E-04
3.4063490E-03	0.000000E+00	2.4253440E-03	0.000000E+00
0.000000E+00	2.4253440E-03	0.000000E+00	5.1626760E-04
8.4122110E-04	0.000000E+00	5.1626760E-04	0.000000E+00
CKTCRL(4,K,L)-			
-2.2678710E-03	0.000000E+00	7.6378460E-04	0.000000E+00
0.000000E+00	2.8308130E-04	0.000000E+00	1.4827730E-04
7.6378460E-04	0.000000E+00	1.3073090E-04	0.000000E+00
0.000000E+00	1.4827730E-04	0.000000E+00	1.5741480E-05
CMRIGD-			
0.000000E+00	6.4966190E+01	0.000000E+00	-4.2173530E+00
CMRGD1-			
2.2779000E+02	0.000000E+00	2.3999150E-01	0.000000E+00
CMBLNC-			
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
CMGRAV-			
0.000000E+00	2.1128080E+00	0.000000E+00	-2.8395860E-01
CMGRV1-			
1.0081580E+01	0.000000E+00	-8.0574100E-01	0.000000E+00
CMHUB1-			
9.0235100E-01	0.000000E+00	0.000000E+00	0.000000E+00
KOSTIF-			

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
K1STIF-			
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
K2STIF-			
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
DAMP -			
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
TCORLS(1,1)-			
1.0081500E+01	9.8663750E+00	9.5392470E+00	9.1238200E+00
8.6388350E+00	8.1051190E+00	7.5394350E+00	6.9500780E+00
6.3529620E+00	5.7644290E+00	5.1990590E+00	4.6630750E+00
4.1681130E+00	3.7192550E+00	3.3237990E+00	2.9484800E+00
2.5839090E+00	2.2341860E+00	1.9033940E+00	1.5639160E+00
0.000000E+00			
TCORLS(2,1)-			
2.1128270E+00	2.1123900E+00	2.1090430E+00	2.0997130E+00
2.0819000E+00	2.0540900E+00	2.0154400E+00	1.9650790E+00
1.9030750E+00	1.8301280E+00	1.7474600E+00	1.6558360E+00
1.5577070E+00	1.4553350E+00	1.3524730E+00	1.2420960E+00
1.1222820E+00	9.9525940E-01	8.6397670E-01	7.1791740E-01
0.000000E+00			
TCORLS(3,1)-			
-8.0570250E-01	-7.5057150E-01	-6.6788760E-01	-5.6368380E-01
-4.4274570E-01	-3.1122080E-01	-1.7522270E-01	-3.9570120E-02
8.8817760E-02	2.0339000E-01	2.9907120E-01	3.7364430E-01
4.2563970E-01	4.5608210E-01	4.6731410E-01	4.6262870E-01
4.4329790E-01	4.1091280E-01	3.6778800E-01	3.1100700E-01
0.000000E+00			
TCORLS(4,1)-			
-2.8395640E-01	-2.8348530E-01	-2.8069860E-01	-2.7412010E-01
-2.6258720E-01	-2.4528320E-01	-2.2179010E-01	-1.9201140E-01
-1.5697290E-01	-1.1860560E-01	-7.9414780E-02	-4.1653340E-02
-7.8960160E-03	2.0126780E-02	4.1191160E-02	5.6578030E-02
6.6237290E-02	6.9901270E-02	6.7864180E-02	5.9955730E-02
0.000000E+00			
TGRAV-			
2.6713120E+01	2.3788780E+01	2.1080080E+01	1.8609220E+01
1.6357350E+01	1.4324270E+01	1.2497370E+01	1.0844640E+01
9.3644120E+00	8.0567000E+00	6.9181700E+00	5.9310100E+00
5.0908870E+00	4.3842720E+00	3.8037450E+00	3.2878390E+00
2.8163650E+00	2.3893230E+00	2.0067140E+00	1.6345970E+00
0.000000E+00			
TOMGA-			
4.0749440E+02	3.9888640E+02	3.8578960E+02	3.6914690E+02
3.4970350E+02	3.2828990E+02	3.0557420E+02	2.8188530E+02
2.5785960E+02	2.3415240E+02	2.1135070E+02	1.8970650E+02
1.6969250E+02	1.5151820E+02	1.3548440E+02	1.2024650E+02
1.0542580E+02	9.1191170E+01	7.7711350E+01	6.3862130E+01
0.000000E+00			
CIFMON(1,1)-			

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

CIFMOM(2,1)-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

CIFMOM(3,1)-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

CIFMOM(4,1)-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

DELTIM-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

OFFMAS-

0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

0.499915E-01	0.000000	-0.128996E-01	0.000000
0.974451E-01	0.479990E-03	-0.247922E-01	-0.470760E-03
0.144827	0.226373E-02	-0.364379E-01	-0.172893E-02
0.192138	0.560398E-02	-0.480739E-01	-0.376290E-02
0.239369	0.105679E-01	-0.594390E-01	-0.668386E-02
0.286509	0.171897E-01	-0.699322E-01	-0.105695E-01
0.333544	0.255666E-01	-0.787526E-01	-0.153675E-01
0.380458	0.359055E-01	-0.850203E-01	-0.208500E-01
0.427239	0.485312E-01	-0.878786E-01	-0.266104E-01
0.473877	0.638611E-01	-0.865778E-01	-0.320938E-01
0.520367	0.823587E-01	-0.805395E-01	-0.366515E-01
0.566707	0.104471	-0.694031E-01	-0.396127E-01
0.612903	0.130561	-0.530526E-01	-0.403623E-01
0.658964	0.160841	-0.316252E-01	-0.384173E-01

0.704903	0.195319	-0.550104E-02	-0.334936E-01
0.750740	0.233762	0.247261E-01	-0.255528E-01
0.796495	0.275687	0.582962E-01	-0.148218E-01
0.842191	0.320391	0.943502E-01	-0.177524E-02
0.887848	0.367023	0.132015	0.129271E-01
0.933486	0.414711	0.170507	0.285568E-01
0.979118	0.462746	0.209258	0.444972E-01
0.249957E-01	0.000000	-0.644979E-02	0.000000
0.249564E-01	0.560284E-03	-0.614713E-02	-0.466659E-03
0.249195E-01	0.133761E-02	-0.613075E-02	-0.858673E-03
0.248804E-01	0.218301E-02	-0.609334E-02	-0.129346E-02
0.248359E-01	0.304435E-02	-0.581596E-02	-0.178889E-02
0.247841E-01	0.393412E-02	-0.515822E-02	-0.229707E-02
0.247245E-01	0.490171E-02	-0.404835E-02	-0.273349E-02
0.246576E-01	0.601012E-02	-0.247334E-02	-0.300163E-02
0.245847E-01	0.731711E-02	-0.469084E-03	-0.301281E-02
0.245076E-01	0.886087E-02	0.188952E-02	-0.270159E-02
0.244289E-01	0.106501E-01	0.449843E-02	-0.203648E-02
0.243511E-01	0.126585E-01	0.723435E-02	-0.102597E-02
0.242770E-01	0.148238E-01	0.996460E-02	0.279918E-03
0.242092E-01	0.170511E-01	0.125570E-01	0.179279E-02
0.241500E-01	0.192212E-01	0.148897E-01	0.339361E-02
0.241012E-01	0.212022E-01	0.168611E-01	0.494507E-02
0.240640E-01	0.228670E-01	0.183998E-01	0.630857E-02
0.240384E-01	0.241143E-01	0.194743E-01	0.736578E-02
0.240236E-01	0.248943E-01	0.201031E-01	0.804484E-02
0.240176E-01	0.252389E-01	0.203642E-01	0.835122E-02
0.240166E-01	0.252963E-01	0.204055E-01	0.840306E-02
0.000000	0.195118E-03	0.301220E-03	-0.302912E-03
0.000000	0.370241E-03	0.528635E-04	-0.210331E-03
0.000000	0.435249E-03	-0.875183E-05	-0.212279E-03
0.000000	0.450356E-03	0.672519E-04	-0.246663E-03
0.000000	0.457658E-03	0.236950E-03	-0.270386E-03
0.000000	0.483482E-03	0.461618E-03	-0.256911E-03
0.000000	0.540726E-03	0.707726E-03	-0.193842E-03
0.000000	0.631208E-03	0.946945E-03	-0.804900E-04
0.000000	0.748010E-03	0.115614E-02	0.745562E-04
0.000000	0.877819E-03	0.131739E-02	0.255854E-03
0.000000	0.100328E-02	0.141794E-02	0.443537E-03
0.000000	0.110532E-02	0.145026E-02	0.615745E-03
0.000000	0.116554E-02	0.141201E-02	0.751052E-03
0.000000	0.116849E-02	0.130605E-02	0.830892E-03
0.000000	0.110407E-02	0.114043E-02	0.841988E-03
0.000000	0.969868E-03	0.928408E-03	0.778785E-03
0.000000	0.773475E-03	0.688436E-03	0.645870E-03
0.000000	0.534854E-03	0.444163E-03	0.460409E-03
0.000000	0.288682E-03	0.224436E-03	0.254567E-03
0.000000	0.866866E-04	0.633004E-04	0.779450E-04
0.000000	0.247582E-13	0.170329E-13	0.226193E-13

Appendix B

Module 2 Interactive Run Listing and Output Files

Program For Analysis Of Horizontal Axis Wind Turbine
Response To Dynamic Loads

MODULE 2

Operations Menu

- (R)ead in a data file
- (S)et up and run the model
- (D)iagnostic run
- (T)urbulence run
- (Q)uit

Enter Option (R,S,D,T,Q) >

R

Enter the name of the file that contains run-data >

RUNB.DAT

Reading in new data...done.

Analyzing a teetering hub.

Operations Menu

- (R)ead in a data file
- (S)et up and run the model
- (D)iagnostic run
- (T)urbulence run
- (Q)uit

Enter Option (R,S,D,T,Q) >

S

Current values for the 'free' variables:

1	ALENTH =	6.7900 feet	12	PHIAMP =	0.0000 degrees
2	ALPHA0 =	-4.0000 degrees	13	PHICMG =	0.0000 degrees/sec
3	BETA0 =	7.0000 degrees	14	PSISHD =	180.0000 degrees
4	CHI =	0.0000 degrees	15	PSIZER =	20.0000 degrees
5	GRAV =	32.1740 feet/sec ²	16	RHOAIR =	0.0020 Slugs/feet ³
6	HUBHT =	80.0000 feet	17	SHERXP =	0.1640
7	KSHADW =	1	18	THETAP =	2.0000 degrees
8	NBLADS =	2	19	THETAT =	4.0000 degrees
9	NSHAPS =	1	20	TSUBO =	0.0500
10	OMEGA =	60.0000 RPM	21	TSUBP =	0.0500
11	PHIO =	13.0000 degrees	22	VHUB =	33.6400 feet/second
23	BETA1 =	0.0000 degrees	24	BETA2 =	0.0000 degrees
25	TIMINC =	0.041660 sec	26	NUMSCN =	100

Would you like to change any values? (Y,=N) >

Y

Enter the number of the variable you would like
to change (1-26) >

9

Enter new INTEGER value for NSHAPS >

2

Current values for the 'free' variables:

1	ALENTH =	6.7900 feet	12	PHIAMP =	0.0000 degrees
2	ALPHA0 =	-4.0000 degrees	13	PHIOMG =	0.0000 degrees/sec
3	BETA0 =	7.0000 degrees	14	PSISHD =	180.0000 degrees
4	CHI =	0.0000 degrees	15	PSIZER =	20.0000 degrees
5	GRAV =	32.1740 feet/sec^2	16	RHOAIR =	0.0020 Slugs/feet^3
6	HUBHT =	80.0000 feet	17	SHERXP =	0.1640
7	KSHADW =	1	18	THETAP =	2.0000 degrees
8	NBLADS =	2	19	THETAT =	4.0000 degrees
9	NSHAPS =	2	20	TSUBO =	0.0500
10	OMEGA =	60.0000 RPM	21	TSUBP =	0.0500
11	PHIO =	13.0000 degrees	22	VHUB =	33.6400 feet/second
23	BETA1 =	0.0000 degrees	24	BETA2 =	0.0000 degrees
25	TIMINC =	0.041660 sec	26	NUMSCN =	100

Would you like to change any values? (Y,=N) >

N

The current values of the run parameters are:

1	STEPMX =	10.000 degrees	Maximum Step Size
2	STEPMN =	0.001 degrees	Minimum Step Size
3	PRINT1 =	10.000 degrees	Printout Interval in Region 1
4	PRINT2 =	10.000 degrees	Printout Interval in Region 2
5	BEGIN2 =	180.000 degrees	Beginning of Print Region 2
6	END2 =	270.000 degrees	End of Print Region 2
7	EUERR =	10.000 percent	Max value of Euler error function
8	TRMERR =	10.000 percent	Convergence Criterion for Trim Solution
9	NYAW =	0	No. of Disk Revolutions for Yawing Soln.
10	TRACEF =	.FALSE.	Trace Flag

Do you want to change any of these values? (Y,=N) >

Y

Enter number of variable you wish to change >

1

Enter new REAL value for STEPMX >

5.00000

The current values of the run parameters are:

1	STEPMX =	5.000 degrees	Maximum Step Size
2	STEPMN =	0.001 degrees	Minimum Step Size
3	PRINT1 =	10.000 degrees	Printout Interval in Region 1
4	PRINT2 =	10.000 degrees	Printout Interval in Region 2
5	BEGIN2 =	180.000 degrees	Beginning of Print Region 2
6	END2 =	270.000 degrees	End of Print Region 2
7	EUERR =	10.000 percent	Max value of Euler error function
8	TRMERR =	10.000 percent	Convergence Criterion for Trim Solution
9	NYAW =	0	No. of Disk Revolutions for Yawing Soln.
10	TRACEF =	.FALSE.	Trace Flag

Do you want to change any of these values? (Y,=N) >

Y

Enter number of variable you wish to change >

7

Enter new REAL value for EUERR >
5.00000

The current values of the run parameters are:

1 STEPMX = 5.000 degrees Maximum Step Size
2 STEP MN = 0.001 degrees Minimum Step Size
3 PRINT1 = 10.000 degrees Printout Interval in Region 1
4 PRINT2 = 10.000 degrees Printout Interval in Region 2
5 BEGIN2 = 180.000 degrees Beginning of Print Region 2
6 END2 = 270.000 degrees End of Print Region 2
7 EUERR = 5.000 percent Max value of Euler error function
8 TRMERR = 10.000 percent Convergence Criterion for Trim Solution
9 NYAW = 0 No. of Disk Revolutions for Yawing Soln.
10 TRACEF = .FALSE. Trace Flag

Do you want to change any of these values? (Y,=N) >
Y

Enter number of variable you wish to change >
8

Enter new REAL value for TRMERR >
5.00000

The current values of the run parameters are:

1 STEPMX = 5.000 degrees Maximum Step Size
2 STEP MN = 0.001 degrees Minimum Step Size
3 PRINT1 = 10.000 degrees Printout Interval in Region 1
4 PRINT2 = 10.000 degrees Printout Interval in Region 2
5 BEGIN2 = 180.000 degrees Beginning of Print Region 2
6 END2 = 270.000 degrees End of Print Region 2
7 EUERR = 5.000 percent Max value of Euler error function
8 TRMERR = 5.000 percent Convergence Criterion for Trim Solution
9 NYAW = 0 No. of Disk Revolutions for Yawing Soln.
10 TRACEF = .FALSE. Trace Flag

Do you want to change any of these values? (Y,=N) >
N

Product of CMMASS premultiplied by its inverse:

1.0000000 0.0000000
0.0000000 1.0000000

Are you ready to run the model? (=Y,N) >
Y

Enter name of results file [=RESULTS.DAT] >
>>> File already exists <<<

Do you want to overwrite the old data? (Y,N) >
Y

Do you want to input your own values for the
initial modal deflections, or do you want
SUBROUTINE STRTUP to calculate values for you?

```

If you want to input your own then input a "Y"
or a "y", If you want SUB STRTUP to do it
input an "N" or an "n" >
Y
Read in S0(1) and S0(2) <
(all must be nonzero)
-0.100000
 1.00000

Trim test #01 completed. The trim condition was not satisfied.

Trim test #02 completed. The trim condition was not satisfied.

Trim test #03 completed. The trim condition was satisfied.

Do you want to perform fourier analysis of
any of the results data? <
Y
Read in the result data item #
that you want analyzed <

1 = flapwise displacement
2 = flapwise slope
3 = flapwise segment velocity
4 = blade tension
5 = blade edgewise shear
6 = blade flapwise shear
7 = blade flapwise moment
8 = blade edgewise moment
9 = blade torsional moment

 7
Read in the highest order harmonic desired <
The highest order possible is 10P <
 10
Do you want to perform fourier analysis of
any of the results data? <
N
Read in name of shaft loads output file <
SHAFTB.DAT

Do you want to do another run with this data? (Y,=N) >
N

Operations Menu
-----

(R)ead in a data file
(S)et up and run the model
(D)iagnostic run
(T)urbulence run
(Q)uit

Enter Option (R,S,D,T,Q) >
T

Turbulence Analysis Run Set-up

Read in the name of the wind residual time series file
WIN0-2B.DAT

```

Is the wind data in metric units (meters/sec)?

Y

Read in the name of the file for loads output

turbloadb.dat

Operations Menu

(R)ead in a data file

(S)et up and run the model

(D)iagnostic run

(T)urbulence run

(Q)uit

Enter Option (R,S,D,T,Q) >

N

Invalid response. Please try again.

Enter Option (R,S,D,T,Q) >

Q

STRAP terminated normally.

Analysis of Wind Turbine Blade Loads
National Renewable Energy Laboratory

TURBN.DAT

DATA INPUT FOR THE EXAMPLE TWO BLADED, DOWNWIND

80 FT DIAM. TEETERING ROTOR

Teetering Rotor

Run Parameters, blade data and machine data used in this analysis:

ALENTH = 6.790 feet	HUBRAD = 2.000 feet	TSUBP = 0.050
ALPHA0 = -4.000 degrees	OMEGA = 60.000 RPM	TSUB0 = 0.050
BETA0 = 7.000 degrees	PHIAMP = 0.000 degrees	THETAP = 2.000 degrees
BLSHNK = 0.800 feet	PHIOMG = 0.000 degrees/sec	THETAT = 4.000 degrees
BLTIP = 38.000 feet	PHIO = 13.000 degrees	TRMERR = 5.000 percent
CHI = 0.000 degrees	PSIZER = 20.000 degrees	VHUB = 33.640 feet/second
CSUBMA = 0.015	RHOAIR = 0.002 slugs/ft ³	DELT3 = 0.000 degrees
DRGFRM = 0.003	SHERXP = 0.164	UNDSLK = 0.750 feet
EUERR = 5.000 percent	STEPMX = 5.000 degrees	HUBMAS = 55.900 lb-s ² /ft
HUBHT = 80.000 feet	STEPMN = 0.001 degrees	HUBDIS = 0.240 feet

KSHADW = 1 NBLADS = 2 NSHAPS = 2 NYAW = 0

PSI = (DEG.) TEETER = (DEG.)

0	0.176374
10	0.255745
20	0.330059
30	0.391605
40	0.438567
50	0.470043
60	0.485653
70	0.485604
80	0.470307
90	0.440561
100	0.397677
110	0.343343
120	0.279586
130	0.208652
140	0.132973
150	0.549818E-01
160	-0.228706E-01
170	-0.987628E-01
180	-0.175699
190	-0.255478
200	-0.330069
210	-0.391889
220	-0.439118
230	-0.470846
240	-0.486650
250	-0.486653
260	-0.471269
270	-0.441435
280	-0.398463
290	-0.344029
300	-0.280170
310	-0.209148
320	-0.133361
330	-0.552660E-01
340	0.227006E-01
350	0.986932E-01
360	0.175716

PSI = (DEG.)Teeter Damper Loads (ft-lb)

0	0.000000
10	0.000000
20	0.000000
30	0.000000
40	0.000000
50	0.000000
60	0.000000
70	0.000000
80	0.000000
90	0.000000
100	0.000000
110	0.000000
120	0.000000
130	0.000000
140	0.000000
150	0.000000
160	0.000000
170	0.000000
180	0.000000
190	0.000000
200	0.000000
210	0.000000
220	0.000000
230	0.000000
240	0.000000
250	0.000000
260	0.000000
270	0.000000
280	0.000000
290	0.000000
300	0.000000
310	0.000000
320	0.000000
330	0.000000
340	0.000000
350	0.000000
360	0.000000

Blade section flap displacement (feet)

Trim

Psi	Time	Phi	Phi-D	Phi-DD	X / R								
deg	sec	deg	deg/s	deg/s^2	.0	.1	.2	.3	.4	.5	.6	.7	.8
0	0.00	13.0	0.00	0.00	6.15E-3	1.91E-2	3.52E-2	5.50E-2	7.91E-2	0.11	0.15	0.19	0.25
10	0.00	13.0	0.00	0.00	8.92E-3	2.70E-2	4.82E-2	7.28E-2	0.10	0.14	0.18	0.23	0.29
20	0.00	13.0	0.00	0.00	1.15E-2	3.45E-2	6.04E-2	8.96E-2	0.12	0.16	0.21	0.26	0.32
30	0.00	13.0	0.00	0.00	1.37E-2	4.07E-2	7.07E-2	0.10	0.14	0.18	0.23	0.29	0.36
40	0.00	13.0	0.00	0.00	1.53E-2	4.55E-2	7.88E-2	0.12	0.16	0.20	0.26	0.32	0.39
50	0.00	13.0	0.00	0.00	1.64E-2	4.88E-2	8.45E-2	0.12	0.17	0.22	0.27	0.34	0.42
60	0.00	13.0	0.00	0.00	1.69E-2	5.05E-2	8.76E-2	0.13	0.17	0.23	0.29	0.36	0.44
70	0.00	13.0	0.00	0.00	1.69E-2	5.06E-2	8.80E-2	0.13	0.18	0.23	0.29	0.37	0.45
80	0.00	13.0	0.00	0.00	1.64E-2	4.91E-2	8.59E-2	0.13	0.17	0.23	0.29	0.37	0.45
90	0.00	13.0	0.00	0.00	1.54E-2	4.61E-2	8.12E-2	0.12	0.17	0.22	0.28	0.36	0.44
100	0.00	13.0	0.00	0.00	1.39E-2	4.18E-2	7.41E-2	0.11	0.15	0.20	0.27	0.34	0.42
110	0.00	13.0	0.00	0.00	1.20E-2	3.63E-2	6.50E-2	9.84E-2	0.14	0.18	0.24	0.31	0.39
120	0.00	13.0	0.00	0.00	9.75E-3	2.98E-2	5.41E-2	8.31E-2	0.12	0.16	0.21	0.27	0.35
130	0.00	13.0	0.00	0.00	7.28E-3	2.26E-2	4.20E-2	6.58E-2	9.50E-2	0.13	0.18	0.23	0.30
140	0.00	13.0	0.00	0.00	4.64E-3	1.49E-2	2.90E-2	4.74E-2	7.09E-2	0.10	0.14	0.19	0.25
150	0.00	13.0	0.00	0.00	1.92E-3	6.94E-3	1.57E-2	2.84E-2	4.61E-2	7.04E-2	0.10	0.15	0.20
160	0.00	13.0	0.00	0.00	-7.98E-4	-9.83E-4	2.38E-3	9.68E-3	2.17E-2	4.00E-2	6.68E-2	0.10	0.15
170	0.00	13.0	0.00	0.00	-3.44E-3	-8.69E-3	-1.05E-2	-8.45E-3	-1.85E-3	1.10E-2	3.20E-2	6.25E-2	0.10
180	0.00	13.0	0.00	0.00	-6.13E-3	-1.65E-2	-2.35E-2	-2.69E-2	-2.58E-2	-1.86E-2	-3.53E-3	2.07E-2	5.35E-3
190	0.00	13.0	0.00	0.00	-8.91E-3	-2.46E-2	-3.72E-2	-4.61E-2	-5.09E-2	-4.99E-2	-4.13E-2	-2.40E-2	1.54E-3
200	0.00	13.0	0.00	0.00	-1.15E-2	-3.22E-2	-4.98E-2	-6.40E-2	-7.41E-2	-7.86E-2	-7.57E-2	-6.45E-2	-4.53E-2
210	0.00	13.0	0.00	0.00	-1.37E-2	-3.85E-2	-6.01E-2	-7.83E-2	-9.24E-2	-0.10	-0.10	-9.44E-2	-7.90E-2
220	0.00	13.0	0.00	0.00	-1.53E-2	-4.32E-2	-6.78E-2	-8.87E-2	-0.11	-0.12	-0.12	-0.11	-9.89E-2
230	0.00	13.0	0.00	0.00	-1.64E-2	-4.63E-2	-7.27E-2	-9.51E-2	-0.11	-0.12	-0.13	-0.12	-0.11
240	0.00	13.0	0.00	0.00	-1.70E-2	-4.78E-2	-7.48E-2	-9.76E-2	-0.12	-0.13	-0.13	-0.12	-0.10
250	0.00	13.0	0.00	0.00	-1.70E-2	-4.77E-2	-7.43E-2	-9.65E-2	-0.11	-0.12	-0.12	-0.11	-8.94E-2
260	0.00	13.0	0.00	0.00	-1.64E-2	-4.60E-2	-7.14E-2	-9.19E-2	-0.11	-0.11	-0.11	-9.61E-2	-7.04E-2
270	0.00	13.0	0.00	0.00	-1.54E-2	-4.30E-2	-6.61E-2	-8.43E-2	-9.67E-2	-0.10	-9.48E-2	-7.65E-2	-4.69E-2
280	0.00	13.0	0.00	0.00	-1.39E-2	-3.86E-2	-5.88E-2	-7.41E-2	-8.34E-2	-8.46E-2	-7.52E-2	-5.36E-2	-2.04E-2
290	0.00	13.0	0.00	0.00	-1.20E-2	-3.31E-2	-4.98E-2	-6.16E-2	-6.75E-2	-6.53E-2	-5.27E-2	-2.80E-2	8.18E-3
300	0.00	13.0	0.00	0.00	-9.77E-3	-2.67E-2	-3.94E-2	-4.72E-2	-4.93E-2	-4.37E-2	-2.79E-2	-2.83E-4	3.84E-2
310	0.00	13.0	0.00	0.00	-7.29E-3	-1.96E-2	-2.78E-2	-3.14E-2	-2.95E-2	-2.03E-2	-1.19E-3	2.92E-2	7.01E-2
320	0.00	13.0	0.00	0.00	-4.65E-3	-1.20E-2	-1.55E-2	-1.46E-2	-8.51E-3	4.61E-3	2.70E-2	6.02E-2	0.10
330	0.00	13.0	0.00	0.00	-1.93E-3	-4.20E-3	-2.75E-3	2.81E-3	1.33E-2	3.04E-2	5.64E-2	9.25E-2	0.14
340	0.00	13.0	0.00	0.00	7.92E-4	3.62E-3	1.00E-2	2.03E-2	3.53E-2	5.66E-2	8.64E-2	0.13	0.17
350	0.00	13.0	0.00	0.00	3.44E-3	1.13E-2	2.25E-2	3.75E-2	5.70E-2	8.27E-2	0.12	0.16	0.21
360	0.00	13.0	0.00	0.00	6.13E-3	1.90E-2	3.51E-2	5.49E-2	7.90E-2	0.11	0.15	0.19	0.25

Blade section flapwise slope (feet/foot)

Trim

Psi deg	Time sec	Phi deg	Phi-D deg/s	Phi-DD deg/s^2	X / R								
					.0	.1	.2	.3	.4	.5	.6	.7	.8
0	0.00	13.0	0.00	0.00	3.08E-3	3.80E-3	4.72E-3	5.72E-3	7.03E-3	8.83E-3	1.11E-2	1.35E-2	1.55E-2
10	0.00	13.0	0.00	0.00	4.46E-3	5.14E-3	6.01E-3	6.96E-3	8.19E-3	9.89E-3	1.20E-2	1.43E-2	1.62E-2
20	0.00	13.0	0.00	0.00	5.76E-3	6.41E-3	7.24E-3	8.15E-3	9.32E-3	1.10E-2	1.30E-2	1.52E-2	1.70E-2
30	0.00	13.0	0.00	0.00	6.83E-3	7.48E-3	8.32E-3	9.22E-3	1.04E-2	1.20E-2	1.41E-2	1.63E-2	1.81E-2
40	0.00	13.0	0.00	0.00	7.65E-3	8.33E-3	9.20E-3	1.01E-2	1.14E-2	1.31E-2	1.52E-2	1.75E-2	1.94E-2
50	0.00	13.0	0.00	0.00	8.20E-3	8.93E-3	9.87E-3	1.09E-2	1.22E-2	1.40E-2	1.63E-2	1.88E-2	2.08E-2
60	0.00	13.0	0.00	0.00	8.47E-3	9.26E-3	1.03E-2	1.14E-2	1.28E-2	1.48E-2	1.73E-2	1.99E-2	2.21E-2
70	0.00	13.0	0.00	0.00	8.47E-3	9.32E-3	1.04E-2	1.16E-2	1.31E-2	1.53E-2	1.80E-2	2.08E-2	2.32E-2
80	0.00	13.0	0.00	0.00	8.20E-3	9.11E-3	1.03E-2	1.15E-2	1.32E-2	1.54E-2	1.83E-2	2.13E-2	2.38E-2
90	0.00	13.0	0.00	0.00	7.68E-3	8.62E-3	9.83E-3	1.11E-2	1.28E-2	1.52E-2	1.81E-2	2.13E-2	2.39E-2
100	0.00	13.0	0.00	0.00	6.94E-3	7.89E-3	9.11E-3	1.04E-2	1.22E-2	1.45E-2	1.75E-2	2.07E-2	2.34E-2
110	0.00	13.0	0.00	0.00	5.99E-3	6.94E-3	8.15E-3	9.47E-3	1.12E-2	1.36E-2	1.65E-2	1.97E-2	2.23E-2
120	0.00	13.0	0.00	0.00	4.88E-3	5.80E-3	6.99E-3	8.27E-3	9.95E-3	1.23E-2	1.52E-2	1.82E-2	2.08E-2
130	0.00	13.0	0.00	0.00	3.64E-3	4.53E-3	5.67E-3	6.91E-3	8.52E-3	1.07E-2	1.35E-2	1.65E-2	1.89E-2
140	0.00	13.0	0.00	0.00	2.32E-3	3.17E-3	4.26E-3	5.45E-3	6.99E-3	9.12E-3	1.18E-2	1.46E-2	1.69E-2
150	0.00	13.0	0.00	0.00	9.59E-4	1.77E-3	2.82E-3	3.95E-3	5.42E-3	7.46E-3	1.00E-2	1.27E-2	1.49E-2
160	0.00	13.0	0.00	0.00	-3.99E-4	3.87E-4	1.39E-3	2.48E-3	3.90E-3	5.86E-3	8.31E-3	1.09E-2	1.30E-2
170	0.00	13.0	0.00	0.00	-1.72E-3	-9.57E-4	1.93E-3	1.08E-3	2.47E-3	4.37E-3	6.75E-3	9.26E-3	1.13E-2
180	0.00	13.0	0.00	0.00	-3.06E-3	-2.32E-3	-1.38E-3	-3.43E-4	9.99E-4	2.85E-3	5.15E-3	7.58E-3	9.59E-3
190	0.00	13.0	0.00	0.00	-4.46E-3	-3.75E-3	-2.84E-3	-1.86E-3	-5.73E-4	1.19E-3	3.39E-3	5.70E-3	7.61E-3
200	0.00	13.0	0.00	0.00	-5.76E-3	-5.07E-3	-4.19E-3	-3.24E-3	-2.00E-3	-2.90E-4	1.84E-3	4.07E-3	5.91E-3
210	0.00	13.0	0.00	0.00	-6.83E-3	-6.14E-3	-5.26E-3	-4.29E-3	-3.04E-3	-1.32E-3	8.25E-4	3.07E-3	4.93E-3
220	0.00	13.0	0.00	0.00	-7.66E-3	-6.93E-3	-6.01E-3	-4.99E-3	-3.68E-3	-1.88E-3	3.68E-4	2.72E-3	4.67E-3
230	0.00	13.0	0.00	0.00	-8.21E-3	-7.43E-3	-6.44E-3	-5.35E-3	-3.94E-3	-2.01E-3	3.99E-4	2.92E-3	5.01E-3
240	0.00	13.0	0.00	0.00	-8.49E-3	-7.64E-3	-6.57E-3	-5.40E-3	-3.88E-3	-1.79E-3	8.14E-4	3.54E-3	5.80E-3
250	0.00	13.0	0.00	0.00	-8.49E-3	-7.58E-3	-6.43E-3	-5.18E-3	-3.54E-3	-1.30E-3	1.49E-3	4.42E-3	6.84E-3
260	0.00	13.0	0.00	0.00	-8.22E-3	-7.26E-3	-6.05E-3	-4.73E-3	-3.00E-3	-6.35E-4	2.31E-3	5.41E-3	7.97E-3
270	0.00	13.0	0.00	0.00	-7.70E-3	-6.71E-3	-5.46E-3	-4.09E-3	-2.31E-3	1.40E-4	3.19E-3	6.39E-3	9.04E-3
280	0.00	13.0	0.00	0.00	-6.95E-3	-5.95E-3	-4.68E-3	-3.30E-3	-1.50E-3	9.74E-4	4.06E-3	7.30E-3	9.98E-3
290	0.00	13.0	0.00	0.00	-6.00E-3	-5.01E-3	-3.76E-3	-2.39E-3	-6.10E-4	1.84E-3	4.89E-3	8.10E-3	1.08E-2
300	0.00	13.0	0.00	0.00	-4.89E-3	-3.93E-3	-2.71E-3	-1.38E-3	3.46E-4	2.72E-3	5.69E-3	8.81E-3	1.14E-2
310	0.00	13.0	0.00	0.00	-3.65E-3	-2.73E-3	-1.57E-3	-2.95E-4	1.36E-3	3.63E-3	6.48E-3	9.47E-3	1.19E-2
320	0.00	13.0	0.00	0.00	-2.33E-3	-1.46E-3	-3.51E-4	8.55E-4	2.42E-3	4.58E-3	7.29E-3	1.01E-2	1.25E-2
330	0.00	13.0	0.00	0.00	-9.64E-4	-1.41E-4	9.08E-4	2.05E-3	3.54E-3	5.58E-3	8.15E-3	1.08E-2	1.31E-2
340	0.00	13.0	0.00	0.00	3.96E-4	1.18E-3	2.18E-3	3.27E-3	4.69E-3	6.64E-3	9.09E-3	1.17E-2	1.38E-2
350	0.00	13.0	0.00	0.00	1.72E-3	2.48E-3	3.44E-3	4.49E-3	5.85E-3	7.74E-3	1.01E-2	1.26E-2	1.47E-2
360	0.00	13.0	0.00	0.00	3.06E-3	3.79E-3	4.71E-3	5.72E-3	7.03E-3	8.84E-3	1.11E-2	1.35E-2	1.55E-2

Blade section flap velocity (feet/second)

Trim

Psi deg	Time sec	Phi deg	Phi-D deg/s	Phi-DD deg/s^2	X / R								
					.0	.1	.2	.3	.4	.5	.6	.7	.8
0	0.00	13.0	0.00	0.00	9.86E-2	0.28	0.46	0.63	0.80	0.95	1.1	1.2	1.3
10	0.00	13.0	0.00	0.00	9.87E-2	0.28	0.46	0.63	0.80	0.95	1.1	1.2	1.3
20	0.00	13.0	0.00	0.00	8.59E-2	0.25	0.41	0.57	0.72	0.87	1.0	1.2	1.3
30	0.00	13.0	0.00	0.00	6.84E-2	0.20	0.33	0.47	0.61	0.75	0.90	1.1	1.2
40	0.00	13.0	0.00	0.00	4.94E-2	0.15	0.25	0.36	0.48	0.61	0.76	0.92	1.1
50	0.00	13.0	0.00	0.00	2.96E-2	8.93E-2	0.16	0.24	0.33	0.44	0.57	0.73	0.91
60	0.00	13.0	0.00	0.00	9.66E-3	3.18E-2	6.42E-2	0.11	0.17	0.24	0.34	0.47	0.62
70	0.00	13.0	0.00	0.00	-9.71E-3	-2.46E-2	-3.02E-2	-2.54E-2	-8.16E-3	2.59E-2	8.23E-2	0.16	0.27
80	0.00	13.0	0.00	0.00	-2.85E-2	-8.00E-2	-0.12	-0.16	-0.19	-0.20	-0.20	-0.18	-0.14
90	0.00	13.0	0.00	0.00	-4.59E-2	-0.13	-0.21	-0.29	-0.36	-0.43	-0.48	-0.53	-0.56
100	0.00	13.0	0.00	0.00	-6.14E-2	-0.18	-0.29	-0.41	-0.52	-0.64	-0.75	-0.86	-0.97
110	0.00	13.0	0.00	0.00	-7.46E-2	-0.22	-0.36	-0.51	-0.66	-0.82	-0.98	-1.1	-1.3
120	0.00	13.0	0.00	0.00	-8.50E-2	-0.25	-0.42	-0.59	-0.77	-0.96	-1.2	-1.4	-1.6
130	0.00	13.0	0.00	0.00	-9.25E-2	-0.27	-0.45	-0.65	-0.84	-1.1	-1.3	-1.5	-1.8
140	0.00	13.0	0.00	0.00	-9.69E-2	-0.28	-0.48	-0.68	-0.89	-1.1	-1.3	-1.6	-1.9
150	0.00	13.0	0.00	0.00	-9.83E-2	-0.29	-0.48	-0.68	-0.89	-1.1	-1.3	-1.6	-1.9
160	0.00	13.0	0.00	0.00	-9.66E-2	-0.28	-0.47	-0.66	-0.86	-1.1	-1.3	-1.5	-1.8
170	0.00	13.0	0.00	0.00	-9.48E-2	-0.28	-0.46	-0.65	-0.84	-1.0	-1.2	-1.5	-1.7
180	0.00	13.0	0.00	0.00	-9.91E-2	-0.29	-0.48	-0.68	-0.89	-1.1	-1.3	-1.6	-1.8
190	0.00	13.0	0.00	0.00	-9.91E-2	-0.29	-0.48	-0.69	-0.89	-1.1	-1.3	-1.6	-1.8
200	0.00	13.0	0.00	0.00	-8.63E-2	-0.25	-0.42	-0.58	-0.75	-0.93	-1.1	-1.3	-1.5
210	0.00	13.0	0.00	0.00	-6.86E-2	-0.20	-0.32	-0.45	-0.56	-0.67	-0.78	-0.87	-0.96
220	0.00	13.0	0.00	0.00	-4.97E-2	-0.14	-0.23	-0.30	-0.37	-0.42	-0.46	-0.48	-0.48
230	0.00	13.0	0.00	0.00	-2.99E-2	-8.30E-2	-0.13	-0.16	-0.18	-0.18	-0.16	-0.12	-4.49E-2
240	0.00	13.0	0.00	0.00	-9.89E-3	-2.48E-2	-2.95E-2	-2.27E-2	-2.32E-3	3.66E-2	0.10	0.19	0.31
250	0.00	13.0	0.00	0.00	9.81E-3	3.19E-2	6.33E-2	0.10	0.16	0.23	0.32	0.44	0.58
260	0.00	13.0	0.00	0.00	2.87E-2	8.57E-2	0.15	0.22	0.30	0.39	0.50	0.63	0.78
270	0.00	13.0	0.00	0.00	4.60E-2	0.13	0.23	0.32	0.42	0.53	0.65	0.77	0.91
280	0.00	13.0	0.00	0.00	6.15E-2	0.18	0.30	0.41	0.53	0.65	0.76	0.88	1.0
290	0.00	13.0	0.00	0.00	7.47E-2	0.22	0.35	0.49	0.62	0.74	0.85	0.96	1.1
300	0.00	13.0	0.00	0.00	8.51E-2	0.24	0.40	0.55	0.69	0.81	0.93	1.0	1.1
310	0.00	13.0	0.00	0.00	9.27E-2	0.27	0.43	0.59	0.74	0.87	0.99	1.1	1.2
320	0.00	13.0	0.00	0.00	9.71E-2	0.28	0.45	0.62	0.77	0.91	1.0	1.1	1.2
330	0.00	13.0	0.00	0.00	9.84E-2	0.28	0.46	0.63	0.79	0.94	1.1	1.2	1.3
340	0.00	13.0	0.00	0.00	9.67E-2	0.28	0.45	0.63	0.79	0.94	1.1	1.2	1.3
350	0.00	13.0	0.00	0.00	9.49E-2	0.27	0.45	0.62	0.78	0.94	1.1	1.2	1.3
360	0.00	13.0	0.00	0.00	9.92E-2	0.29	0.47	0.64	0.80	0.96	1.1	1.2	1.3

Blade tension (Lb)

Trim

Psi	Time	Phi	Phi-D	Phi-DD	X / R									
deg	sec	deg	deg/s	deg/s^2	.0	.1	.2	.3	.4	.5	.6	.7	.8	
0	0.00	13.0	0.00	0.00	1.52E+4	1.46E+4	1.33E+4	1.17E+4	9.88E+3	8.13E+3	6.54E+3	5.23E+3	4.07E+3	
10	0.00	13.0	0.00	0.00	1.52E+4	1.46E+4	1.33E+4	1.17E+4	9.89E+3	8.13E+3	6.54E+3	5.23E+3	4.07E+3	
20	0.00	13.0	0.00	0.00	1.53E+4	1.46E+4	1.33E+4	1.17E+4	9.90E+3	8.14E+3	6.55E+3	5.24E+3	4.08E+3	
30	0.00	13.0	0.00	0.00	1.53E+4	1.46E+4	1.34E+4	1.17E+4	9.92E+3	8.15E+3	6.56E+3	5.25E+3	4.09E+3	
40	0.00	13.0	0.00	0.00	1.54E+4	1.47E+4	1.34E+4	1.18E+4	9.95E+3	8.18E+3	6.58E+3	5.26E+3	4.09E+3	
50	0.00	13.0	0.00	0.00	1.55E+4	1.48E+4	1.35E+4	1.18E+4	9.99E+3	8.20E+3	6.60E+3	5.27E+3	4.11E+3	
60	0.00	13.0	0.00	0.00	1.57E+4	1.49E+4	1.35E+4	1.19E+4	1.00E+4	8.23E+3	6.62E+3	5.29E+3	4.12E+3	
70	0.00	13.0	0.00	0.00	1.58E+4	1.50E+4	1.36E+4	1.19E+4	1.01E+4	8.27E+3	6.64E+3	5.31E+3	4.13E+3	
80	0.00	13.0	0.00	0.00	1.59E+4	1.51E+4	1.37E+4	1.20E+4	1.01E+4	8.30E+3	6.67E+3	5.33E+3	4.15E+3	
90	0.00	13.0	0.00	0.00	1.61E+4	1.52E+4	1.38E+4	1.21E+4	1.02E+4	8.34E+3	6.70E+3	5.35E+3	4.16E+3	
100	0.00	13.0	0.00	0.00	1.62E+4	1.53E+4	1.39E+4	1.21E+4	1.02E+4	8.38E+3	6.73E+3	5.37E+3	4.18E+3	
110	0.00	13.0	0.00	0.00	1.64E+4	1.55E+4	1.40E+4	1.22E+4	1.03E+4	8.42E+3	6.75E+3	5.39E+3	4.19E+3	
120	0.00	13.0	0.00	0.00	1.65E+4	1.56E+4	1.41E+4	1.23E+4	1.03E+4	8.45E+3	6.78E+3	5.41E+3	4.20E+3	
130	0.00	13.0	0.00	0.00	1.66E+4	1.57E+4	1.41E+4	1.23E+4	1.04E+4	8.48E+3	6.80E+3	5.42E+3	4.22E+3	
140	0.00	13.0	0.00	0.00	1.67E+4	1.57E+4	1.42E+4	1.24E+4	1.04E+4	8.51E+3	6.82E+3	5.44E+3	4.23E+3	
150	0.00	13.0	0.00	0.00	1.68E+4	1.58E+4	1.43E+4	1.24E+4	1.04E+4	8.53E+3	6.84E+3	5.45E+3	4.24E+3	
160	0.00	13.0	0.00	0.00	1.69E+4	1.59E+4	1.43E+4	1.24E+4	1.05E+4	8.55E+3	6.85E+3	5.46E+3	4.24E+3	
170	0.00	13.0	0.00	0.00	1.69E+4	1.59E+4	1.43E+4	1.25E+4	1.05E+4	8.56E+3	6.86E+3	5.47E+3	4.25E+3	
180	0.00	13.0	0.00	0.00	1.69E+4	1.59E+4	1.43E+4	1.25E+4	1.05E+4	8.56E+3	6.86E+3	5.47E+3	4.25E+3	
190	0.00	13.0	0.00	0.00	1.69E+4	1.59E+4	1.43E+4	1.25E+4	1.05E+4	8.56E+3	6.86E+3	5.47E+3	4.25E+3	
200	0.00	13.0	0.00	0.00	1.69E+4	1.59E+4	1.43E+4	1.24E+4	1.05E+4	8.55E+3	6.85E+3	5.46E+3	4.25E+3	
210	0.00	13.0	0.00	0.00	1.68E+4	1.58E+4	1.43E+4	1.24E+4	1.04E+4	8.53E+3	6.84E+3	5.45E+3	4.24E+3	
220	0.00	13.0	0.00	0.00	1.67E+4	1.57E+4	1.42E+4	1.24E+4	1.04E+4	8.51E+3	6.82E+3	5.44E+3	4.23E+3	
230	0.00	13.0	0.00	0.00	1.66E+4	1.57E+4	1.41E+4	1.23E+4	1.04E+4	8.49E+3	6.80E+3	5.43E+3	4.22E+3	
240	0.00	13.0	0.00	0.00	1.65E+4	1.56E+4	1.41E+4	1.23E+4	1.03E+4	8.46E+3	6.78E+3	5.41E+3	4.21E+3	
250	0.00	13.0	0.00	0.00	1.64E+4	1.55E+4	1.40E+4	1.22E+4	1.03E+4	8.42E+3	6.76E+3	5.39E+3	4.19E+3	
260	0.00	13.0	0.00	0.00	1.62E+4	1.54E+4	1.39E+4	1.21E+4	1.02E+4	8.38E+3	6.73E+3	5.37E+3	4.18E+3	
270	0.00	13.0	0.00	0.00	1.61E+4	1.52E+4	1.38E+4	1.21E+4	1.02E+4	8.35E+3	6.70E+3	5.35E+3	4.16E+3	
280	0.00	13.0	0.00	0.00	1.59E+4	1.51E+4	1.37E+4	1.20E+4	1.01E+4	8.31E+3	6.67E+3	5.33E+3	4.15E+3	
290	0.00	13.0	0.00	0.00	1.58E+4	1.50E+4	1.36E+4	1.19E+4	1.01E+4	8.27E+3	6.65E+3	5.31E+3	4.13E+3	
300	0.00	13.0	0.00	0.00	1.57E+4	1.49E+4	1.35E+4	1.19E+4	1.00E+4	8.24E+3	6.62E+3	5.29E+3	4.12E+3	
310	0.00	13.0	0.00	0.00	1.55E+4	1.48E+4	1.35E+4	1.18E+4	9.99E+3	8.20E+3	6.60E+3	5.27E+3	4.11E+3	
320	0.00	13.0	0.00	0.00	1.54E+4	1.47E+4	1.34E+4	1.18E+4	9.95E+3	8.18E+3	6.58E+3	5.26E+3	4.09E+3	
330	0.00	13.0	0.00	0.00	1.54E+4	1.47E+4	1.34E+4	1.17E+4	9.92E+3	8.15E+3	6.56E+3	5.24E+3	4.09E+3	
340	0.00	13.0	0.00	0.00	1.53E+4	1.46E+4	1.33E+4	1.17E+4	9.90E+3	8.14E+3	6.55E+3	5.24E+3	4.08E+3	
350	0.00	13.0	0.00	0.00	1.52E+4	1.46E+4	1.33E+4	1.17E+4	9.89E+3	8.13E+3	6.54E+3	5.23E+3	4.07E+3	
360	0.00	13.0	0.00	0.00	1.52E+4	1.46E+4	1.33E+4	1.17E+4	9.88E+3	8.13E+3	6.54E+3	5.23E+3	4.07E+3	

Blade edgewise shear (Lb)

Trim

Psi	Time	Phi	Phi-D	Phi-DD	X / R								
deg	sec	deg	deg/s	deg/s^2	.0	.1	.2	.3	.4	.5	.6	.7	.8
0	0.00	13.0	0.00	0.00	3.00E+2	2.90E+2	2.71E+2	2.43E+2	2.08E+2	1.71E+2	1.34E+2	1.00E+2	68.
10	0.00	13.0	0.00	0.00	4.47E+2	4.06E+2	3.60E+2	3.10E+2	2.59E+2	2.08E+2	1.61E+2	1.20E+2	83.
20	0.00	13.0	0.00	0.00	5.92E+2	5.20E+2	4.49E+2	3.78E+2	3.10E+2	2.45E+2	1.89E+2	1.41E+2	98.
30	0.00	13.0	0.00	0.00	7.28E+2	6.28E+2	5.32E+2	4.42E+2	3.57E+2	2.81E+2	2.15E+2	1.61E+2	1.13E+2
40	0.00	13.0	0.00	0.00	8.51E+2	7.25E+2	6.08E+2	5.00E+2	4.00E+2	3.12E+2	2.38E+2	1.78E+2	1.26E+2
50	0.00	13.0	0.00	0.00	9.57E+2	8.09E+2	6.72E+2	5.49E+2	4.37E+2	3.39E+2	2.58E+2	1.93E+2	1.37E+2
60	0.00	13.0	0.00	0.00	1.04E+3	8.76E+2	7.24E+2	5.88E+2	4.66E+2	3.61E+2	2.74E+2	2.05E+2	1.46E+2
70	0.00	13.0	0.00	0.00	1.11E+3	9.25E+2	7.62E+2	6.17E+2	4.87E+2	3.76E+2	2.85E+2	2.13E+2	1.52E+2
80	0.00	13.0	0.00	0.00	1.14E+3	9.54E+2	7.84E+2	6.33E+2	4.99E+2	3.84E+2	2.91E+2	2.18E+2	1.56E+2
90	0.00	13.0	0.00	0.00	1.15E+3	9.62E+2	7.90E+2	6.37E+2	5.01E+2	3.86E+2	2.92E+2	2.18E+2	1.56E+2
100	0.00	13.0	0.00	0.00	1.14E+3	9.49E+2	7.79E+2	6.28E+2	4.94E+2	3.80E+2	2.88E+2	2.15E+2	1.54E+2
110	0.00	13.0	0.00	0.00	1.10E+3	9.16E+2	7.53E+2	6.07E+2	4.77E+2	3.67E+2	2.78E+2	2.08E+2	1.48E+2
120	0.00	13.0	0.00	0.00	1.03E+3	8.63E+2	7.11E+2	5.74E+2	4.52E+2	3.48E+2	2.63E+2	1.97E+2	1.40E+2
130	0.00	13.0	0.00	0.00	9.42E+2	7.92E+2	6.55E+2	5.31E+2	4.19E+2	3.23E+2	2.45E+2	1.83E+2	1.30E+2
140	0.00	13.0	0.00	0.00	8.34E+2	7.06E+2	5.87E+2	4.78E+2	3.79E+2	2.93E+2	2.22E+2	1.66E+2	1.17E+2
150	0.00	13.0	0.00	0.00	7.09E+2	6.07E+2	5.10E+2	4.18E+2	3.33E+2	2.59E+2	1.97E+2	1.46E+2	1.03E+2
160	0.00	13.0	0.00	0.00	5.71E+2	4.98E+2	4.25E+2	3.53E+2	2.84E+2	2.22E+2	1.69E+2	1.26E+2	88.
170	0.00	13.0	0.00	0.00	3.98E+2	3.55E+2	3.09E+2	2.60E+2	2.11E+2	1.66E+2	1.27E+2	94.	65.
180	0.00	13.0	0.00	0.00	2.23E+2	2.13E+2	1.94E+2	1.68E+2	1.39E+2	1.11E+2	86.	64.	43.
190	0.00	13.0	0.00	0.00	1.03E+2	1.24E+2	1.30E+2	1.24E+2	1.09E+2	91.	72.	53.	35.
200	0.00	13.0	0.00	0.00	-13.	38.	68.	81.	81.	72.	59.	43.	27.
210	0.00	13.0	0.00	0.00	-1.48E+2	-68.	-14.	19.	34.	38.	34.	24.	13.
220	0.00	13.0	0.00	0.00	-2.69E+2	-1.64E+2	-87.	-37.	-7.4	7.2	12.	7.9	0.20
230	0.00	13.0	0.00	0.00	-3.73E+2	-2.45E+2	-1.50E+2	-85.	-42.	-18.	-7.0	-6.0	-10.
240	0.00	13.0	0.00	0.00	-4.57E+2	-3.11E+2	-2.00E+2	-1.22E+2	-70.	-38.	-21.	-17.	-18.
250	0.00	13.0	0.00	0.00	-5.18E+2	-3.58E+2	-2.37E+2	-1.49E+2	-89.	-52.	-32.	-24.	-24.
260	0.00	13.0	0.00	0.00	-5.54E+2	-3.86E+2	-2.58E+2	-1.65E+2	-1.00E+2	-60.	-37.	-28.	-27.
270	0.00	13.0	0.00	0.00	-5.65E+2	-3.94E+2	-2.63E+2	-1.68E+2	-1.02E+2	-61.	-37.	-28.	-27.
280	0.00	13.0	0.00	0.00	-5.50E+2	-3.82E+2	-2.53E+2	-1.59E+2	-95.	-55.	-33.	-25.	-25.
290	0.00	13.0	0.00	0.00	-5.09E+2	-3.49E+2	-2.27E+2	-1.39E+2	-79.	-43.	-24.	-18.	-20.
300	0.00	13.0	0.00	0.00	-4.45E+2	-2.98E+2	-1.87E+2	-1.08E+2	-55.	-25.	-10.	-8.1	-12.
310	0.00	13.0	0.00	0.00	-3.58E+2	-2.29E+2	-1.33E+2	-66.	-24.	-1.1	7.3	5.1	-2.4
320	0.00	13.0	0.00	0.00	-2.51E+2	-1.45E+2	-67.	-15.	15.	28.	29.	21.	9.3
330	0.00	13.0	0.00	0.00	-1.28E+2	-47.	9.1	43.	59.	60.	53.	39.	23.
340	0.00	13.0	0.00	0.00	8.3	60.	93.	1.07E+2	1.07E+2	96.	79.	59.	37.
350	0.00	13.0	0.00	0.00	1.53E+2	1.74E+2	1.81E+2	1.75E+2	1.58E+2	1.34E+2	1.07E+2	80.	53.
360	0.00	13.0	0.00	0.00	3.00E+2	2.90E+2	2.71E+2	2.42E+2	2.08E+2	1.71E+2	1.34E+2	9.99E+1	68.

Blade flapwise shear (Lb)

Trim

Psi deg	Time sec	Phi deg	Phi-D deg/s	Phi-DD deg/s ²	X / R								
					.0	.1	.2	.3	.4	.5	.6	.7	.8
0	0.00	13.0	0.00	0.00	-96.	-18.	1.16E+2	2.54E+2	3.64E+2	4.17E+2	4.07E+2	3.25E+2	1.82E+2
10	0.00	13.0	0.00	0.00	-64.	9.3	1.36E+2	2.66E+2	3.68E+2	4.15E+2	4.01E+2	3.17E+2	1.75E+2
20	0.00	13.0	0.00	0.00	-27.	42.	1.62E+2	2.85E+2	3.81E+2	4.21E+2	4.04E+2	3.17E+2	1.74E+2
30	0.00	13.0	0.00	0.00	-2.4	67.	1.86E+2	3.07E+2	3.99E+2	4.37E+2	4.17E+2	3.29E+2	1.82E+2
40	0.00	13.0	0.00	0.00	26.	95.	2.14E+2	3.33E+2	4.24E+2	4.58E+2	4.36E+2	3.45E+2	1.95E+2
50	0.00	13.0	0.00	0.00	54.	1.25E+2	2.44E+2	3.62E+2	4.50E+2	4.82E+2	4.57E+2	3.62E+2	2.09E+2
60	0.00	13.0	0.00	0.00	74.	1.47E+2	2.67E+2	3.86E+2	4.72E+2	5.02E+2	4.75E+2	3.78E+2	2.22E+2
70	0.00	13.0	0.00	0.00	90.	1.66E+2	2.88E+2	4.06E+2	4.91E+2	5.19E+2	4.91E+2	3.92E+2	2.33E+2
80	0.00	13.0	0.00	0.00	92.	1.72E+2	2.95E+2	4.15E+2	4.99E+2	5.27E+2	4.99E+2	3.99E+2	2.39E+2
90	0.00	13.0	0.00	0.00	80.	1.64E+2	2.91E+2	4.12E+2	4.97E+2	5.26E+2	4.99E+2	3.99E+2	2.39E+2
100	0.00	13.0	0.00	0.00	55.	1.43E+2	2.74E+2	3.97E+2	4.84E+2	5.15E+2	4.90E+2	3.92E+2	2.34E+2
110	0.00	13.0	0.00	0.00	19.	1.12E+2	2.46E+2	3.73E+2	4.62E+2	4.96E+2	4.74E+2	3.79E+2	2.23E+2
120	0.00	13.0	0.00	0.00	-26.	72.	2.10E+2	3.41E+2	4.33E+2	4.71E+2	4.53E+2	3.62E+2	2.09E+2
130	0.00	13.0	0.00	0.00	-76.	26.	1.69E+2	3.04E+2	3.99E+2	4.42E+2	4.28E+2	3.41E+2	1.92E+2
140	0.00	13.0	0.00	0.00	-1.26E+2	-20.	1.28E+2	2.66E+2	3.66E+2	4.13E+2	4.03E+2	3.20E+2	1.76E+2
150	0.00	13.0	0.00	0.00	-1.73E+2	-63.	89.	2.31E+2	3.34E+2	3.85E+2	3.80E+2	3.01E+2	1.60E+2
160	0.00	13.0	0.00	0.00	-2.14E+2	-1.00E+2	55.	2.01E+2	3.08E+2	3.63E+2	3.61E+2	2.85E+2	1.48E+2
170	0.00	13.0	0.00	0.00	-2.48E+2	-1.33E+2	21.	1.65E+2	2.71E+2	3.31E+2	3.36E+2	2.68E+2	1.40E+2
180	0.00	13.0	0.00	0.00	-2.88E+2	-1.73E+2	-19.	1.23E+2	2.30E+2	2.97E+2	3.09E+2	2.49E+2	1.30E+2
190	0.00	13.0	0.00	0.00	-3.29E+2	-2.09E+2	-48.	1.05E+2	2.20E+2	2.88E+2	3.00E+2	2.37E+2	1.14E+2
200	0.00	13.0	0.00	0.00	-3.59E+2	-2.35E+2	-67.	94.	2.17E+2	2.86E+2	2.97E+2	2.31E+2	1.03E+2
210	0.00	13.0	0.00	0.00	-3.58E+2	-2.34E+2	-65.	97.	2.21E+2	2.90E+2	3.01E+2	2.34E+2	1.07E+2
220	0.00	13.0	0.00	0.00	-3.51E+2	-2.28E+2	-59.	1.04E+2	2.29E+2	2.98E+2	3.09E+2	2.42E+2	1.12E+2
230	0.00	13.0	0.00	0.00	-3.29E+2	-2.07E+2	-39.	1.24E+2	2.49E+2	3.17E+2	3.25E+2	2.56E+2	1.24E+2
240	0.00	13.0	0.00	0.00	-2.97E+2	-1.78E+2	-11.	1.50E+2	2.74E+2	3.40E+2	3.46E+2	2.73E+2	1.39E+2
250	0.00	13.0	0.00	0.00	-2.65E+2	-1.49E+2	16.	1.76E+2	2.99E+2	3.62E+2	3.65E+2	2.90E+2	1.52E+2
260	0.00	13.0	0.00	0.00	-2.34E+2	-1.21E+2	41.	1.98E+2	3.20E+2	3.81E+2	3.81E+2	3.04E+2	1.64E+2
270	0.00	13.0	0.00	0.00	-2.08E+2	-99.	60.	2.16E+2	3.36E+2	3.95E+2	3.93E+2	3.13E+2	1.72E+2
280	0.00	13.0	0.00	0.00	-1.85E+2	-80.	75.	2.28E+2	3.46E+2	4.04E+2	4.00E+2	3.19E+2	1.76E+2
290	0.00	13.0	0.00	0.00	-1.67E+2	-67.	84.	2.35E+2	3.52E+2	4.08E+2	4.03E+2	3.21E+2	1.78E+2
300	0.00	13.0	0.00	0.00	-1.51E+2	-56.	91.	2.38E+2	3.54E+2	4.09E+2	4.02E+2	3.20E+2	1.77E+2
310	0.00	13.0	0.00	0.00	-1.38E+2	-47.	96.	2.41E+2	3.54E+2	4.08E+2	4.00E+2	3.18E+2	1.75E+2
320	0.00	13.0	0.00	0.00	-1.24E+2	-37.	1.02E+2	2.43E+2	3.55E+2	4.08E+2	3.98E+2	3.16E+2	1.73E+2
330	0.00	13.0	0.00	0.00	-1.08E+2	-26.	1.09E+2	2.48E+2	3.57E+2	4.09E+2	3.98E+2	3.15E+2	1.72E+2
340	0.00	13.0	0.00	0.00	-90.	-11.	1.21E+2	2.56E+2	3.63E+2	4.13E+2	4.01E+2	3.17E+2	1.73E+2
350	0.00	13.0	0.00	0.00	-90.	-12.	1.21E+2	2.58E+2	3.67E+2	4.18E+2	4.07E+2	3.23E+2	1.80E+2
360	0.00	13.0	0.00	0.00	-96.	-17.	1.16E+2	2.55E+2	3.65E+2	4.18E+2	4.08E+2	3.25E+2	1.82E+2

Blade flapwise moment (ft-Lbs)

Trim

Psi	Time	Phi	Phi-D	Phi-DD	X / R								
deg	sec	deg	deg/s	deg/s^2	.0	.1	.2	.3	.4	.5	.6	.7	.8
0	0.00	13.0	0.00	0.00	-5.07E+3	-5.49E+3	-5.54E+3	-5.08E+3	-4.15E+3	-2.92E+3	-1.61E+3	-4.74E+2	2.44E+2
10	0.00	13.0	0.00	0.00	-4.88E+3	-5.27E+3	-5.30E+3	-4.84E+3	-3.93E+3	-2.73E+3	-1.47E+3	-3.81E+2	2.96E+2
20	0.00	13.0	0.00	0.00	-4.95E+3	-5.28E+3	-5.26E+3	-4.77E+3	-3.86E+3	-2.66E+3	-1.41E+3	-3.35E+2	3.25E+2
30	0.00	13.0	0.00	0.00	-5.19E+3	-5.49E+3	-5.44E+3	-4.92E+3	-3.97E+3	-2.75E+3	-1.47E+3	-3.72E+2	3.06E+2
40	0.00	13.0	0.00	0.00	-5.62E+3	-5.87E+3	-5.76E+3	-5.17E+3	-4.17E+3	-2.89E+3	-1.57E+3	-4.37E+2	2.72E+2
50	0.00	13.0	0.00	0.00	-6.16E+3	-6.33E+3	-6.14E+3	-5.48E+3	-4.40E+3	-3.07E+3	-1.94E+3	-5.13E+2	2.52E+2
60	0.00	13.0	0.00	0.00	-6.62E+3	-6.73E+3	-6.47E+3	-5.75E+3	-4.61E+3	-3.22E+3	-1.80E+3	-5.78E+2	1.98E+2
70	0.00	13.0	0.00	0.00	-7.07E+3	-7.12E+3	-6.80E+3	-6.00E+3	-4.80E+3	-3.36E+3	-1.90E+3	-6.37E+2	1.68E+2
80	0.00	13.0	0.00	0.00	-7.28E+3	-7.31E+3	-6.95E+3	-6.13E+3	-4.90E+3	-3.43E+3	-1.94E+3	-6.63E+2	1.55E+2
90	0.00	13.0	0.00	0.00	-7.27E+3	-7.31E+3	-6.96E+3	-6.13E+3	-4.89E+3	-3.43E+3	-1.94E+3	-6.58E+2	1.59E+2
100	0.00	13.0	0.00	0.00	-7.04E+3	-7.12E+3	-6.81E+3	-6.01E+3	-4.80E+3	-3.35E+3	-1.88E+3	-6.22E+2	1.80E+2
110	0.00	13.0	0.00	0.00	-6.62E+3	-6.78E+3	-6.52E+3	-5.78E+3	-4.62E+3	-3.22E+3	-1.79E+3	-5.62E+2	2.13E+2
120	0.00	13.0	0.00	0.00	-6.08E+3	-6.33E+3	-6.16E+3	-5.48E+3	-4.39E+3	-3.05E+3	-1.67E+3	-4.85E+2	2.54E+2
130	0.00	13.0	0.00	0.00	-5.45E+3	-5.81E+3	-5.73E+3	-5.14E+3	-4.12E+3	-2.85E+3	-1.53E+3	-3.99E+2	3.01E+2
140	0.00	13.0	0.00	0.00	-4.87E+3	-5.33E+3	-5.34E+3	-4.83E+3	-3.88E+3	-2.68E+3	-1.41E+3	-3.21E+2	3.43E+2
150	0.00	13.0	0.00	0.00	-4.38E+3	-4.92E+3	-5.01E+3	-4.56E+3	-3.68E+3	-2.53E+3	-1.31E+3	-2.56E+2	3.77E+2
160	0.00	13.0	0.00	0.00	-4.04E+3	-4.65E+3	-4.78E+3	-4.39E+3	-3.54E+3	-2.43E+3	-1.24E+3	-2.14E+2	3.99E+2
170	0.00	13.0	0.00	0.00	-3.63E+3	-4.29E+3	-4.47E+3	-4.14E+3	-3.38E+3	-2.34E+3	-1.22E+3	-2.37E+2	3.59E+2
180	0.00	13.0	0.00	0.00	-3.11E+3	-3.83E+3	-4.09E+3	-3.84E+3	-3.17E+3	-2.22E+3	-1.17E+3	-2.41E+2	3.32E+2
190	0.00	13.0	0.00	0.00	-2.95E+3	-3.73E+3	-4.03E+3	-3.79E+3	-3.11E+3	-2.14E+3	-1.07E+3	-1.39E+2	4.17E+2
200	0.00	13.0	0.00	0.00	-2.99E+3	-3.80E+3	-4.11E+3	-3.86E+3	-3.14E+3	-2.13E+3	-1.03E+3	-73.	4.81E+2
210	0.00	13.0	0.00	0.00	-3.45E+3	-4.19E+3	-4.44E+3	-4.13E+3	-3.35E+3	-2.29E+3	-1.14E+3	-1.44E+2	4.42E+2
220	0.00	13.0	0.00	0.00	-3.92E+3	-4.58E+3	-4.76E+3	-4.39E+3	-3.55E+3	-2.44E+3	-1.24E+3	-2.11E+2	4.06E+2
230	0.00	13.0	0.00	0.00	-4.61E+3	-5.16E+3	-5.24E+3	-4.78E+3	-3.85E+3	-2.66E+3	-1.40E+3	-3.08E+2	3.53E+2
240	0.00	13.0	0.00	0.00	-5.39E+3	-5.81E+3	-5.78E+3	-5.20E+3	-4.18E+3	-2.90E+3	-1.56E+3	-4.15E+2	2.95E+2
250	0.00	13.0	0.00	0.00	-6.04E+3	-6.36E+3	-6.22E+3	-5.56E+3	-4.46E+3	-3.10E+3	-1.70E+3	-5.02E+2	2.47E+2
260	0.00	13.0	0.00	0.00	-6.54E+3	-6.77E+3	-6.55E+3	-5.82E+3	-4.66E+3	-3.24E+3	-1.80E+3	-5.66E+2	2.13E+2
270	0.00	13.0	0.00	0.00	-6.83E+3	-6.99E+3	-6.73E+3	-5.97E+3	-4.77E+3	-3.32E+3	-1.86E+3	-6.01E+2	1.93E+2
280	0.00	13.0	0.00	0.00	-6.92E+3	-7.06E+3	-6.78E+3	-6.00E+3	-4.80E+3	-3.34E+3	-1.87E+3	-6.10E+2	1.87E+2
290	0.00	13.0	0.00	0.00	-6.84E+3	-6.98E+3	-6.71E+3	-5.95E+3	-4.75E+3	-3.31E+3	-1.85E+3	-5.97E+2	1.93E+2
300	0.00	13.0	0.00	0.00	-6.64E+3	-6.79E+3	-6.55E+3	-5.82E+3	-4.66E+3	-3.24E+3	-1.80E+3	-5.68E+2	2.08E+2
310	0.00	13.0	0.00	0.00	-6.37E+3	-6.55E+3	-6.34E+3	-5.65E+3	-4.53E+3	-3.15E+3	-1.74E+3	-5.31E+2	2.27E+2
320	0.00	13.0	0.00	0.00	-6.09E+3	-6.30E+3	-6.14E+3	-5.49E+3	-4.41E+3	-3.06E+3	-1.68E+3	-4.95E+2	2.45E+2
330	0.00	13.0	0.00	0.00	-5.86E+3	-6.10E+3	-5.97E+3	-5.35E+3	-4.31E+3	-2.99E+3	-1.63E+3	-4.67E+2	2.59E+2
340	0.00	13.0	0.00	0.00	-5.72E+3	-5.97E+3	-5.86E+3	-5.27E+3	-4.25E+3	-2.95E+3	-1.60E+3	-4.51E+2	2.65E+2
350	0.00	13.0	0.00	0.00	-5.49E+3	-5.82E+3	-5.78E+3	-5.24E+3	-4.25E+3	-2.97E+3	-1.64E+3	-4.81E+2	2.45E+2
360	0.00	13.0	0.00	0.00	-5.08E+3	-5.51E+3	-5.55E+3	-5.09E+3	-4.16E+3	-2.92E+3	-1.61E+3	-4.76E+2	2.43E+2

Average Flapwise Moment = -5764.5 (ft-Lbs) at the 20% station

Blade edgewise moment (ft-Lbs)

Trim

Psi	Time	Phi	Phi-D	Phi-DD	X / R								
deg	sec	deg	deg/s	deg/s^2	.0	.1	.2	.3	.4	.5	.6	.7	.8
0	0.00	13.0	0.00	0.00	6.37E+3	5.24E+3	4.18E+3	3.20E+3	2.34E+3	1.62E+3	1.04E+3	5.98E+2	2.79E+2
10	0.00	13.0	0.00	0.00	8.30E+3	6.67E+3	5.22E+3	3.94E+3	2.86E+3	1.98E+3	1.28E+3	7.42E+2	3.56E+2
20	0.00	13.0	0.00	0.00	1.02E+4	8.11E+3	6.26E+3	4.69E+3	3.39E+3	2.34E+3	1.51E+3	8.88E+2	4.33E+2
30	0.00	13.0	0.00	0.00	1.20E+4	9.46E+3	7.25E+3	5.40E+3	3.89E+3	2.68E+3	1.74E+3	1.03E+3	5.07E+2
40	0.00	13.0	0.00	0.00	1.37E+4	1.07E+4	8.14E+3	6.04E+3	4.33E+3	2.98E+3	1.94E+3	1.15E+3	5.72E+2
50	0.00	13.0	0.00	0.00	1.51E+4	1.17E+4	8.90E+3	6.58E+3	4.71E+3	3.24E+3	2.11E+3	1.26E+3	6.29E+2
60	0.00	13.0	0.00	0.00	1.62E+4	1.25E+4	9.51E+3	7.02E+3	5.02E+3	3.45E+3	2.25E+3	1.34E+3	6.74E+2
70	0.00	13.0	0.00	0.00	1.70E+4	1.31E+4	9.94E+3	7.32E+3	5.23E+3	3.59E+3	2.34E+3	1.40E+3	7.06E+2
80	0.00	13.0	0.00	0.00	1.75E+4	1.35E+4	1.02E+4	7.49E+3	5.35E+3	3.67E+3	2.40E+3	1.43E+3	7.24E+2
90	0.00	13.0	0.00	0.00	1.76E+4	1.36E+4	1.02E+4	7.52E+3	5.37E+3	3.69E+3	2.40E+3	1.44E+3	7.27E+2
100	0.00	13.0	0.00	0.00	1.73E+4	1.34E+4	1.01E+4	7.41E+3	5.28E+3	3.63E+3	2.37E+3	1.41E+3	7.16E+2
110	0.00	13.0	0.00	0.00	1.67E+4	1.29E+4	9.74E+3	7.16E+3	5.10E+3	3.50E+3	2.28E+3	1.37E+3	6.91E+2
120	0.00	13.0	0.00	0.00	1.58E+4	1.22E+4	9.22E+3	6.78E+3	4.83E+3	3.32E+3	2.16E+3	1.29E+3	6.51E+2
130	0.00	13.0	0.00	0.00	1.46E+4	1.13E+4	8.53E+3	6.28E+3	4.48E+3	3.07E+3	2.00E+3	1.19E+3	6.00E+2
140	0.00	13.0	0.00	0.00	1.31E+4	1.02E+4	7.70E+3	5.68E+3	4.06E+3	2.78E+3	1.81E+3	1.08E+3	5.39E+2
150	0.00	13.0	0.00	0.00	1.14E+4	8.88E+3	6.76E+3	5.00E+3	3.58E+3	2.45E+3	1.59E+3	9.42E+2	4.69E+2
160	0.00	13.0	0.00	0.00	9.53E+3	7.49E+3	5.74E+3	4.26E+3	3.06E+3	2.10E+3	1.36E+3	7.98E+2	3.93E+2
170	0.00	13.0	0.00	0.00	6.95E+3	5.52E+3	4.25E+3	3.17E+3	2.28E+3	1.56E+3	1.01E+3	5.91E+2	2.88E+2
180	0.00	13.0	0.00	0.00	4.41E+3	3.58E+3	2.80E+3	2.11E+3	1.52E+3	1.05E+3	6.74E+2	3.89E+2	1.86E+2
190	0.00	13.0	0.00	0.00	3.07E+3	2.64E+3	2.15E+3	1.67E+3	1.23E+3	8.47E+2	5.37E+2	3.00E+2	1.33E+2
200	0.00	13.0	0.00	0.00	1.81E+3	1.76E+3	1.55E+3	1.26E+3	9.52E+2	6.61E+2	4.11E+2	2.16E+2	83.
210	0.00	13.0	0.00	0.00	33.	4.37E+2	5.88E+2	5.74E+2	4.71E+2	3.32E+2	1.95E+2	83.	12.
220	0.00	13.0	0.00	0.00	-1.55E+3	-7.38E+2	-2.67E+2	-36.	45.	42.	3.9	-35.	-51.
230	0.00	13.0	0.00	0.00	-2.90E+3	-1.73E+3	-9.90E+2	-5.50E+2	-3.14E+2	-2.02E+2	-1.56E+2	-1.34E+2	-1.04E+2
240	0.00	13.0	0.00	0.00	-3.97E+3	-2.52E+3	-1.56E+3	-9.54E+2	-5.94E+2	-3.92E+2	-2.81E+2	-2.11E+2	-1.45E+2
250	0.00	13.0	0.00	0.00	-4.74E+3	-3.08E+3	-1.96E+3	-1.24E+3	-7.89E+2	-5.24E+2	-3.68E+2	-2.65E+2	-1.75E+2
260	0.00	13.0	0.00	0.00	-5.18E+3	-3.40E+3	-2.19E+3	-1.39E+3	-8.94E+2	-5.94E+2	-4.14E+2	-2.94E+2	-1.91E+2
270	0.00	13.0	0.00	0.00	-5.28E+3	-3.46E+3	-2.22E+3	-1.41E+3	-9.06E+2	-6.02E+2	-4.19E+2	-2.98E+2	-1.93E+2
280	0.00	13.0	0.00	0.00	-5.04E+3	-3.28E+3	-2.08E+3	-1.30E+3	-8.27E+2	-5.46E+2	-3.83E+2	-2.76E+2	-1.82E+2
290	0.00	13.0	0.00	0.00	-4.46E+3	-2.84E+3	-1.75E+3	-1.07E+3	-6.57E+2	-4.30E+2	-3.06E+2	-2.29E+2	-1.58E+2
300	0.00	13.0	0.00	0.00	-3.57E+3	-2.17E+3	-1.26E+3	-7.07E+2	-4.03E+2	-2.55E+2	-1.91E+2	-1.59E+2	-1.22E+2
310	0.00	13.0	0.00	0.00	-2.39E+3	-1.29E+3	-6.06E+2	-2.35E+2	-70.	-27.	-42.	-68.	-75.
320	0.00	13.0	0.00	0.00	-9.55E+2	-2.11E+2	1.85E+2	3.35E+2	3.32E+2	2.48E+2	1.39E+2	43.	-15.
330	0.00	13.0	0.00	0.00	6.98E+2	1.02E+3	1.09E+3	9.87E+2	7.91E+2	5.62E+2	3.45E+2	1.69E+2	52.
340	0.00	13.0	0.00	0.00	2.52E+3	2.38E+3	2.09E+3	1.70E+3	1.29E+3	9.06E+2	5.72E+2	3.08E+2	1.25E+2
350	0.00	13.0	0.00	0.00	4.44E+3	3.81E+3	3.13E+3	2.45E+3	1.82E+3	1.26E+3	8.08E+2	4.54E+2	2.02E+2
360	0.00	13.0	0.00	0.00	6.36E+3	5.24E+3	4.17E+3	3.20E+3	2.34E+3	1.62E+3	1.04E+3	5.97E+2	2.79E+2

Blade torsion (ft-Lbs)

Trim

Psi	Time	Phi	Phi-D	Phi-DD	X / R									
deg	sec	deg	deg/s	deg/s^2	.0	.1	.2	.3	.4	.5	.6	.7	.8	
0	0.00	13.0	0.00	0.00	-21.	-18.	-14.	-9.4	-5.3	-1.6	1.5	3.8	4.6	
10	0.00	13.0	0.00	0.00	-43.	-36.	-28.	-21.	-14.	-7.8	-2.8	1.0	3.1	
20	0.00	13.0	0.00	0.00	-69.	-57.	-45.	-33.	-23.	-15.	-7.6	-2.0	1.5	
30	0.00	13.0	0.00	0.00	-98.	-80.	-63.	-47.	-34.	-22.	-13.	-5.4	-0.32	
40	0.00	13.0	0.00	0.00	-1.27E+2	-1.03E+2	-81.	-62.	-45.	-31.	-19.	-9.1	-2.3	
50	0.00	13.0	0.00	0.00	-1.54E+2	-1.25E+2	-99.	-76.	-56.	-39.	-24.	-13.	-4.3	
60	0.00	13.0	0.00	0.00	-1.75E+2	-1.43E+2	-1.14E+2	-88.	-66.	-46.	-30.	-16.	-6.1	
70	0.00	13.0	0.00	0.00	-1.90E+2	-1.56E+2	-1.25E+2	-97.	-73.	-52.	-34.	-19.	-7.5	
80	0.00	13.0	0.00	0.00	-1.95E+2	-1.61E+2	-1.29E+2	-1.01E+2	-76.	-55.	-36.	-20.	-8.4	
90	0.00	13.0	0.00	0.00	-1.89E+2	-1.57E+2	-1.27E+2	-9.96E+1	-75.	-54.	-36.	-20.	-8.5	
100	0.00	13.0	0.00	0.00	-1.74E+2	-1.45E+2	-1.18E+2	-93.	-71.	-51.	-34.	-19.	-7.7	
110	0.00	13.0	0.00	0.00	-1.51E+2	-1.26E+2	-1.03E+2	-82.	-62.	-45.	-29.	-16.	-6.3	
120	0.00	13.0	0.00	0.00	-1.21E+2	-1.03E+2	-84.	-67.	-51.	-36.	-24.	-13.	-4.3	
130	0.00	13.0	0.00	0.00	-90.	-76.	-63.	-50.	-38.	-27.	-17.	-8.3	-2.0	
140	0.00	13.0	0.00	0.00	-58.	-50.	-42.	-33.	-25.	-17.	-9.9	-3.9	0.35	
150	0.00	13.0	0.00	0.00	-30.	-27.	-22.	-18.	-13.	-7.9	-3.4	0.31	2.6	
160	0.00	13.0	0.00	0.00	-6.7	-7.0	-6.0	-4.2	-2.1	0.10	2.3	4.0	4.6	
170	0.00	13.0	0.00	0.00	12.	9.5	8.4	7.9	7.8	7.8	7.9	7.6	6.5	
180	0.00	13.0	0.00	0.00	23.	20.	18.	17.	15.	14.	12.	10.	8.0	
190	0.00	13.0	0.00	0.00	28.	26.	23.	21.	19.	17.	15.	12.	8.9	
200	0.00	13.0	0.00	0.00	28.	28.	26.	24.	22.	19.	16.	13.	9.5	
210	0.00	13.0	0.00	0.00	23.	26.	26.	25.	23.	20.	17.	14.	9.9	
220	0.00	13.0	0.00	0.00	16.	22.	24.	25.	23.	21.	18.	14.	10.	
230	0.00	13.0	0.00	0.00	8.4	17.	22.	24.	23.	21.	18.	15.	11.	
240	0.00	13.0	0.00	0.00	2.7	14.	21.	23.	23.	22.	19.	15.	11.	
250	0.00	13.0	0.00	0.00	-0.40	13.	20.	24.	24.	22.	20.	16.	11.	
260	0.00	13.0	0.00	0.00	-0.53	13.	21.	24.	25.	23.	20.	16.	11.	
270	0.00	13.0	0.00	0.00	1.9	15.	22.	25.	25.	24.	20.	16.	12.	
280	0.00	13.0	0.00	0.00	6.2	18.	24.	26.	26.	24.	20.	16.	12.	
290	0.00	13.0	0.00	0.00	11.	20.	25.	26.	25.	23.	20.	16.	11.	
300	0.00	13.0	0.00	0.00	16.	22.	25.	25.	24.	22.	19.	15.	11.	
310	0.00	13.0	0.00	0.00	19.	23.	24.	23.	22.	20.	17.	14.	10.	
320	0.00	13.0	0.00	0.00	20.	21.	21.	20.	19.	17.	15.	13.	9.5	
330	0.00	13.0	0.00	0.00	16.	16.	16.	15.	15.	14.	12.	11.	8.5	
340	0.00	13.0	0.00	0.00	8.5	8.5	8.6	8.9	9.1	9.3	9.3	8.8	7.3	
350	0.00	13.0	0.00	0.00	-4.2	-3.0	-1.4	0.48	2.4	4.1	5.6	6.4	6.0	
360	0.00	13.0	0.00	0.00	-21.	-18.	-14.	-9.4	-5.3	-1.6	1.5	3.8	4.6	

Blade flapwise moment expansion coefficients

	X / R									
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9
A0=	-5.50E+3	-5.84E+3	-5.76E+3	-5.19E+3	-4.17E+3	-2.90E+3	-1.57E+3	-4.32E+2	2.77E+2	3.83E
B0=	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A1=	-9.53E+2	-7.53E+2	-6.11E+2	-5.00E+2	-3.95E+2	-2.89E+2	-1.99E+2	-1.27E+2	-68.	-24.
B1=	-2.04E+2	-1.38E+2	-93.	-64.	-47.	-39.	-32.	-24.	-15.	-6.0
A2=	1.45E+3	1.23E+3	1.01E+3	8.03E+2	6.07E+2	4.33E+2	2.88E+2	1.71E+2	84.	25.
B2=	3.51E+2	2.93E+2	2.38E+2	1.87E+2	1.41E+2	1.01E+2	67.	42.	22.	8.1
A3=	-17.	-29.	-36.	-36.	-29.	-18.	-7.4	1.4	5.7	4.6
B3=	18.	20.	21.	21.	18.	14.	9.0	4.5	1.3	-0.21
A4=	-1.08E+2	-95.	-83.	-73.	-66.	-59.	-51.	-41.	-28.	-13.
B4=	1.71E+2	1.47E+2	1.24E+2	1.01E+2	80.	60.	43.	28.	16.	5.7
A5=	1.5	-13.	-22.	-25.	-20.	-12.	-3.5	3.1	6.1	4.5
B5=	1.7	3.9	5.3	5.7	5.1	3.7	2.2	0.80	-0.18	-0.41
A6=	0.77	-3.3	-7.6	-13.	-19.	-23.	-24.	-23.	-17.	-8.6
B6=	55.	48.	40.	34.	27.	21.	15.	9.7	5.4	2.0
A7=	-6.8	-17.	-24.	-24.	-20.	-12.	-4.6	1.5	4.5	3.5
B7=	1.3	2.4	3.1	3.4	3.1	2.4	1.4	0.52	-6.82E-2	-0.22
A8=	20.	14.	7.4	0.88	-6.2	-12.	-15.	-15.	-12.	-6.2
B8=	36.	31.	26.	21.	17.	13.	9.2	6.0	3.4	1.2
A9=	-3.7	-12.	-17.	-17.	-14.	-8.5	-3.0	1.4	3.5	2.7
B9=	-5.1	-3.5	-2.2	-1.3	-0.81	-0.60	-0.54	-0.56	-0.51	-0.30
A*=	12.	7.9	3.9	-0.44	-5.2	-8.9	-11.	-11.	-8.7	-4.4
B*=	15.	13.	11.	9.1	7.4	5.7	4.1	2.7	1.5	0.57

Mean Rotor Torque and Power for Trim Solution

Mean Rotor Torque = 13545.72 ft-Lbs

Mean Rotor Power = 115.39 KW

No. of Data Points = 36

THIS IS THE SHAFT LOADS OUTPUT FILE

Low-Speed Shaft Loads in Rotor coordinates

Azimuth Angle	Fxr	Fyr	Fzr	Mxr	Myr	Mzr
0.000	83.224	3521.565	-1711.795	-562.079	11988.680	-228.418
10.000	353.294	3511.730	-1693.655	-568.739	12619.370	-449.934
20.000	615.885	3517.800	-1625.991	-675.134	13332.490	-667.649
30.000	887.776	3543.516	-1505.106	-473.615	13393.730	-923.316
40.000	1132.749	3578.086	-1339.407	-522.801	13472.770	-1152.804
50.000	1342.854	3627.853	-1132.409	-470.260	13569.430	-1348.364
60.000	1511.618	3679.315	-890.146	-278.797	13666.320	-1503.510
70.000	1634.316	3727.356	-621.588	-247.934	13748.860	-1613.187
80.000	1706.979	3759.993	-333.723	-151.865	13804.550	-1672.482
90.000	1727.573	3774.286	-35.804	-71.547	13829.030	-1678.620
100.000	1695.566	3771.861	263.363	28.534	13818.830	-1631.156
110.000	1611.942	3753.943	554.646	147.144	13777.180	-1531.732
120.000	1479.219	3724.870	829.066	259.385	13712.800	-1384.203
130.000	1301.426	3688.421	1078.431	399.355	13633.840	-1194.168
140.000	1084.101	3652.677	1294.824	498.634	13553.250	-968.792
150.000	833.849	3621.557	1471.772	580.412	13480.460	-715.667
160.000	558.280	3599.406	1603.791	623.337	13423.870	-442.777
170.000	239.398	3566.504	1683.807	604.637	12687.240	-102.764
180.000	-83.224	3521.565	1711.795	562.079	11988.680	228.418
190.000	-353.294	3511.730	1693.655	568.739	12619.370	449.934
200.000	-615.885	3517.800	1625.991	675.134	13332.490	667.649
210.000	-887.776	3543.516	1505.106	473.615	13393.730	923.316
220.000	-1132.749	3578.086	1339.407	522.801	13472.770	1152.804
230.000	-1342.854	3627.853	1132.409	470.260	13569.430	1348.364
240.000	-1511.618	3679.315	890.146	278.797	13666.320	1503.510
250.000	-1634.316	3727.356	621.588	247.934	13748.860	1613.187
260.000	-1706.979	3759.993	333.723	151.865	13804.550	1672.482
270.000	-1727.573	3774.286	35.804	71.547	13829.030	1678.620
280.000	-1695.566	3771.861	-263.363	-28.534	13818.830	1631.156
290.000	-1611.942	3753.943	-554.646	-147.144	13777.180	1531.732
300.000	-1479.219	3724.870	-829.066	-259.385	13712.800	1384.203
310.000	-1301.426	3688.421	-1078.431	-399.355	13633.840	1194.168
320.000	-1084.101	3652.677	-1294.824	-498.634	13553.250	968.792
330.000	-833.849	3621.557	-1471.772	-580.412	13480.460	715.667
340.000	-558.280	3599.406	-1603.791	-623.337	13423.870	442.777
350.000	-239.398	3566.504	-1683.807	-604.637	12687.240	102.764

360.000 83.224 3521.565 -1711.795 -562.079 11988.680 -228.418

Hub loads in fixed frame hub coordinates

Azimuth Angle	F _{xh}	F _{yh}	F _{zh}	M _{xh}	M _{yh}	M _{zh}
0.000	83.224	3521.565	-1711.795	-562.079	11988.680	-228.418
10.000	53.827	3511.730	-1729.274	-638.229	12619.370	-344.338
20.000	22.621	3517.800	-1738.577	-862.768	13332.490	-396.475
30.000	16.283	3543.516	-1747.348	-871.820	13393.730	-562.808
40.000	6.782	3578.086	-1754.162	-1141.497	13472.770	-547.050
50.000	-4.306	3627.853	-1756.584	-1335.184	13569.430	-506.472
60.000	-15.080	3679.315	-1754.173	-1441.476	13666.320	-510.310
70.000	-25.133	3727.356	-1748.350	-1600.698	13748.860	-318.760
80.000	-32.239	3759.993	-1738.996	-1673.445	13804.550	-140.866
90.000	-35.804	3774.286	-1727.573	-1678.620	13829.030	71.547
100.000	-35.070	3771.861	-1715.539	-1611.330	13818.830	255.147
110.000	-30.120	3753.943	-1704.430	-1489.683	13777.180	385.613
120.000	-21.617	3724.870	-1695.574	-1328.447	13712.800	467.467
130.000	-10.415	3688.421	-1690.152	-1171.486	13633.840	461.672
140.000	1.828	3652.677	-1688.740	-1004.703	13553.250	421.622
150.000	13.752	3621.557	-1691.517	-860.485	13480.460	329.580
160.000	23.918	3599.406	-1698.013	-737.183	13423.870	202.880
170.000	56.629	3566.504	-1699.797	-613.296	12687.240	-3.792
180.000	83.224	3521.565	-1711.795	-562.079	11988.680	-228.417
190.000	53.827	3511.730	-1729.274	-638.229	12619.370	-344.338
200.000	22.621	3517.800	-1738.577	-862.768	13332.490	-396.475
210.000	16.283	3543.516	-1747.348	-871.820	13393.730	-562.808
220.000	6.782	3578.086	-1754.162	-1141.497	13472.770	-547.050
230.000	-4.306	3627.853	-1756.584	-1335.184	13569.430	-506.471
240.000	-15.080	3679.315	-1754.173	-1441.476	13666.320	-510.310
250.000	-25.133	3727.356	-1748.350	-1600.698	13748.860	-318.760
260.000	-32.239	3759.993	-1738.996	-1673.445	13804.550	-140.866
270.000	-35.804	3774.286	-1727.573	-1678.620	13829.030	71.547
280.000	-35.070	3771.861	-1715.539	-1611.330	13818.830	255.147
290.000	-30.120	3753.943	-1704.430	-1489.683	13777.180	385.613
300.000	-21.617	3724.870	-1695.574	-1328.447	13712.800	467.467
310.000	-10.414	3688.421	-1690.152	-1171.487	13633.840	461.672
320.000	1.828	3652.677	-1688.740	-1004.703	13553.250	421.622
330.000	13.753	3621.557	-1691.517	-860.485	13480.460	329.579
340.000	23.918	3599.406	-1698.014	-737.183	13423.870	202.880

350.000	56.630	3566.504	-1699.797	-613.296	12687.240	-3.792
360.000	83.224	3521.565	-1711.795	-562.079	11988.680	-228.417

Harmonics of Rotor Shaft Loads

	Fxr	Fyr	Fzr	Mxr	Myr	Mzr
cos(0*PSI)	0.000	3645.596	0.000	0.000	13417.380	0.000
sin(0.*PSI)	0.000	0.000	0.000	0.000	0.000	0.000
cos(PSI)	42.884	0.000	-1717.292	-625.288	-0.001	-144.024
sin(PSI)	1726.108	0.000	-35.208	-36.940	0.000	-1666.094
cos(2.*PSI)	0.000	-121.367	0.000	0.000	-543.361	0.000
sin(2.*PSI)	0.000	-36.943	0.000	0.000	-40.345	0.000
cos(3.*PSI)	10.331	0.000	1.834	0.763	0.000	-22.828
sin(3.*PSI)	-2.286	0.000	0.792	28.029	0.001	14.056
cos(4.*PSI)	0.000	7.607	0.000	0.000	-255.006	0.000
sin(4.*PSI)	0.000	-12.744	0.000	0.000	-15.211	0.000
cos(5.*PSI)	9.647	0.000	1.363	31.431	0.000	-19.841
sin(5.*PSI)	-1.224	0.000	0.376	-1.303	0.000	2.575

Harmonics of Hub Loads in Fixed Coordinates

	Fxh	Fyh	Fzh	Mxh	Myh	Mzh
cos(0*PSI)	3.838	3645.596	-1721.700	-1145.691	13417.380	-53.542
sin(0.*PSI)	0.000	0.000	0.000	0.000	0.000	0.000
cos(PSI)	0.000	0.000	0.000	0.000	0.000	0.000
sin(PSI)	0.000	0.000	0.000	0.000	0.000	0.000
cos(2.*PSI)	44.607	-121.367	6.469	527.812	-543.363	-115.911
sin(2.*PSI)	2.348	-36.943	-33.484	-65.054	-40.345	-512.993
cos(3.*PSI)	0.000	0.000	0.000	0.000	-0.001	0.000
sin(3.*PSI)	0.000	0.000	0.000	0.000	0.001	0.000
cos(4.*PSI)	9.781	7.606	1.067	10.356	-255.007	-6.669
sin(4.*PSI)	-1.519	-12.744	0.242	11.869	-15.211	23.649
cos(5.*PSI)	0.000	0.000	0.000	0.000	-0.001	0.000
sin(5.*PSI)	0.000	0.000	0.000	0.000	0.000	0.000

THIS IS THE TURBULENT LOADS OUTPUT FILE

15.0000	0.290082	-4849.11	-2788.77	-1521.38
-547.663	13347.6	-734.451		
30.0000	0.407125	-4869.42	-3016.36	-1426.99
-500.884	12993.0	-1034.65		
45.0000	0.513719	-4950.44	-3409.65	-1524.08
-384.535	11387.6	-1313.41		
60.0000	0.599161	-4635.69	-3462.06	-1463.24
-239.181	11499.6	-1600.35		
75.0000	0.664704	-3630.25	-2913.51	-1393.49
-90.8391	10778.3	-1761.08		
90.0000	0.709339	-2814.88	-2697.22	-1163.01
39.1511	11812.2	-1723.46		
105.000	0.717143	-2379.58	-2880.73	-965.598
198.780	11880.0	-1608.53		
120.000	0.679682	-2608.66	-3668.34	-892.347
349.482	11864.4	-1422.99		
135.000	0.598025	-2945.63	-4444.34	-1021.48
466.243	11609.6	-1160.91		
150.000	0.476935	-3309.21	-5272.06	-952.531
536.781	12965.0	-715.713		
165.000	0.315479	-3681.35	-5814.07	-1321.25
617.939	13123.2	-341.400		
180.000	0.124328	-4051.85	-6340.58	-1391.53
589.503	11794.4	191.288		
195.000	-0.960621E-01	-4012.97	-6109.44	-1535.71
595.164	12959.3	544.118		
210.000	-0.316486	-4175.13	-6085.49	-1249.28
523.573	13594.5	979.355		
225.000	-0.522088	-4112.10	-5693.56	-1209.80
395.764	12281.1	1363.59		
240.000	-0.710312	-3494.74	-4581.28	-1335.89
236.239	11978.4	1673.04		
255.000	-0.874621	-3186.94	-3584.23	-1392.13
101.532	12030.6	1740.47		
270.000	-0.992994	-3101.46	-2889.71	-1267.27
-64.0715	11846.0	1779.42		
285.000	-1.05482	-3355.18	-2546.99	-1237.34
-240.000	11322.1	1718.50		
300.000	-1.05755	-3931.28	-2540.08	-1342.70
-395.122	11954.8	1555.25		
315.000	-0.999440	-5118.86	-3153.70	-1566.21
-485.228	12908.5	1203.05		
330.000	-0.873824	-6122.80	-3830.01	-1771.19
-592.406	13327.9	880.624		
345.000	-0.691812	-7068.84	-4562.66	-2042.54
-654.269	12708.8	451.860		
0.000000	-0.462763	-7457.66	-4874.85	-2204.38
-623.413	12007.4	-81.5192		
15.0000	-0.190389	-7028.93	-4618.07	-1900.44
-627.344	13214.8	-493.470		
30.0000	0.987588E-01	-5963.82	-3882.37	-1544.04
-563.545	12940.4	-893.276		
45.0000	0.380350	-4613.02	-2966.55	-1232.20
-445.280	12308.5	-1267.42		
60.0000	0.640600	-2889.45	-1812.19	-1060.22
-275.868	11280.6	-1542.84		
75.0000	0.867753	-1463.31	-1000.04	-997.492
-129.152	9730.94	-1688.56		
90.0000	1.04812	-640.104	-918.096	-675.693
39.7621	10110.6	-1716.54		
105.000	1.16363	-353.009	-1328.49	-588.716
228.823	10730.8	-1661.87		
120.000	1.20403	-793.897	-2372.28	-705.528
400.653	10553.1	-1513.66		
135.000	1.17066	-1677.02	-3740.58	-803.344
547.797	11547.1	-1331.24		
150.000	1.07367	-2593.56	-5075.21	-1173.03
637.194	11852.1	-937.132		
165.000	0.913021	-3662.34	-6457.85	-1325.70
680.690	11335.8	-452.119		
180.000	0.684709	-3620.12	-6469.05	-1481.78
658.897	10844.5	58.2567		
195.000	0.394031	-3404.65	-5979.74	-1221.29
634.640	12193.3	395.335		
210.000	0.790550E-01	-1822.75	-4045.45	-1097.25
590.171	11563.7	793.737		

225.000	-0.233243	-703.825	-2463.43	-872.113
477.264	10865.4	1169.09		
240.000	-0.526019	33.7644	-1159.01	-689.081
355.148	9872.02	1423.03		
255.000	-0.779283	-173.352	-636.544	-721.331
152.388	11126.5	1609.34		
270.000	-0.978223	-1277.17	-1125.05	-809.666
-49.2744	11001.0	1769.27		
285.000	-1.11989	-3102.66	-2146.19	-1232.14
-198.800	10505.7	1613.21		
300.000	-1.18177	-4739.16	-3160.15	-1560.01
-373.170	11304.7	1497.77		
315.000	-1.16008	-5936.85	-3829.46	-1817.33
-489.451	11779.6	1230.62		
330.000	-1.05794	-6305.16	-3835.01	-1902.58
-619.896	12352.0	921.223		
345.000	-0.886711	-6314.91	-3561.25	-2018.02
-676.901	11971.9	459.266		
0.000000	-0.650222	-5249.60	-2480.84	-1796.56
-634.560	10509.4	-60.1369		
15.0000	-0.355479	-3707.42	-1242.54	-1158.33
-650.003	11359.9	-362.736		
30.0000	-0.439645E-01	-2292.14	-124.512	-784.282
-577.154	11614.6	-791.932		
45.0000	0.260034	-1407.88	305.645	-646.202
-453.798	10989.3	-1153.27		
60.0000	0.542554	-1106.63	103.273	-771.998
-304.652	9913.58	-1489.88		
75.0000	0.791507	-1279.85	-865.234	-850.803
-168.099	10108.2	-1529.61		
90.0000	0.972950	-1696.29	-1927.19	-1053.47
25.8220	9539.03	-1637.22		
105.000	1.07684	-1973.96	-2874.12	-1036.94
197.217	10091.6	-1604.48		
120.000	1.10167	-1962.07	-3404.40	-1132.24
382.268	10654.5	-1500.60		
135.000	1.05541	-1711.40	-3666.19	-1030.41
535.889	10645.5	-1256.92		
150.000	0.946787	-1123.88	-3443.44	-1040.89
625.359	10481.3	-901.718		
165.000	0.780727	-576.995	-3160.17	-978.731
621.507	10316.1	-399.326		
180.000	0.550889	150.906	-2586.74	-681.623
642.799	9505.01	101.586		
195.000	0.263422	533.404	-1864.79	-662.127
681.428	10543.2	346.620		
210.000	-0.344865E-01	570.383	-1594.65	-503.466
567.408	10581.5	851.285		
225.000	-0.330759	241.368	-1498.35	-599.425
459.302	9410.91	1213.92		
240.000	-0.611221	-997.502	-2068.14	-755.397
354.753	11208.6	1418.88		
255.000	-0.840260	-2092.62	-2501.27	-1040.18
151.858	10633.3	1601.10		
270.000	-1.00164	-3374.82	-3131.87	-1306.04
-28.5940	9968.63	1676.50		
285.000	-1.08695	-4354.85	-3435.48	-1391.55
-174.574	11633.7	1594.57		
300.000	-1.08499	-4300.57	-2848.66	-1463.69
-329.861	11144.5	1430.30		
315.000	-0.999652	-4175.46	-2344.74	-1381.25
-490.783	11113.4	1240.45		
330.000	-0.848475	-4134.99	-1921.25	-1352.58
-568.465	11447.5	830.650		
345.000	-0.641658	-3694.44	-1388.93	-1217.67
-646.935	11064.4	498.551		
0.000000	-0.398416	-3257.31	-835.145	-1205.82
-609.674	9864.71	-78.8622		
15.0000	-0.114463	-2600.93	-296.051	-1052.58
-582.247	10532.9	-534.716		
30.0000	0.192962	-1878.21	108.195	-967.234
-549.695	10088.2	-883.685		
45.0000	0.495333	-1108.16	423.021	-872.849
-428.244	9434.06	-1245.10		
60.0000	0.773657	-518.208	468.226	-966.821
-320.146	8525.18	-1460.56		
75.0000	1.00047	165.397	415.640	-773.023
-181.561	9278.67	-1547.41		

90.0000	1.14352	394.403	-30.7065	-706.373
13.6396	8784.83	-1596.31		
105.000	1.19146	276.813	-670.636	-892.065
200.048	10070.0	-1614.54		
120.000	1.15195	-417.097	-1905.19	-922.714
363.756	9654.80	-1473.27		
135.000	1.03766	-1508.77	-3394.42	-1121.94
538.090	10328.3	-1281.36		
150.000	0.868437	-2549.85	-4751.17	-1201.02
637.469	10954.6	-946.370		
165.000	0.659184	-2828.06	-5202.72	-1434.26
646.824	11358.0	-463.798		
180.000	0.412465	-1863.28	-4270.94	-1555.39
632.218	9342.19	37.5869		
195.000	0.132063	-777.240	-3035.99	-1064.74
647.329	10312.2	402.266		
210.000	-0.149132	423.483	-1545.34	-594.580
593.966	10675.2	780.617		
225.000	-0.406365	1264.31	-345.575	-386.196
467.837	10333.0	1180.55		
240.000	-0.630098	622.589	-415.112	-384.081
318.825	10609.0	1394.93		
255.000	-0.798948	-771.819	-1200.20	-781.015
151.625	10649.6	1561.43		
270.000	-0.898365	-2977.11	-2761.18	-1297.14
-25.5090	10474.7	1607.49		
285.000	-0.918391	-5033.75	-4234.57	-1651.96
-183.601	11429.3	1548.99		
300.000	-0.860711	-6238.81	-5070.13	-1718.08
-362.582	11785.2	1511.37		
315.000	-0.749173	-6241.02	-4655.08	-1789.88
-486.804	12241.4	1234.28		
330.000	-0.600422	-6143.02	-4289.51	-1628.80
-618.926	12871.8	972.646		
345.000	-0.441627	-5745.81	-3695.00	-1504.51
-692.734	12462.7	603.578		
0.000000	-0.289338	-4995.60	-2704.20	-1594.51
-666.284	11137.8	31.7181		
15.0000	-0.134163	-4442.24	-2196.38	-1268.97
-632.350	11812.8	-368.200		
30.0000	0.136808E-01	-3533.00	-1320.49	-1271.35
-552.374	11135.4	-844.533		
45.0000	0.157206	-2360.67	-482.620	-1100.57
-462.728	10159.3	-1135.94		
60.0000	0.285412	-1222.85	173.269	-846.547
-327.099	10445.5	-1371.23		

Appendix C
Module 1 Listing

```
C *****
C *
C *                                     *
C *                               DISCLAIMER                               *
C * *
C * Neither the United States Department of Energy, the *
C * National Renewable Energy Laboratory, the Midwest *
C * Research Institute nor anyone else *
C * who has been involved in the *
C * creation, production or delivery of this program shall *
C * be liable for any direct, indirect, consequential, or *
C * incidental damages arising out of the use, the results *
C * of use, or inability to use this program even if the *
C * United States Department of Energy has been advised of *
C * the possibility of such damages or claim. Some states *
C * do not allow the exclusion or limitation of liability *
C * for consequential or incidental damages, so the above *
C * limitation may not apply to you. *
C *
C *****
```

```
C *****
C *
C *                               The STRAP Code                               *
C * *
C * STRAP calculates the forces and moments on a two bladed *
C * teetering hub rotor due to aerodynamic, inertial and gravita- *
C * tional forces. The aerodynamic effects include wind *
C * shear, tower shadow and the induced velocity due to the *
C * change in momentum of the air stream as it passes over *
C * the blade. In addition, the rotor's response to turbulent *
C * wind inputs is included in this version. The code also *
C * accounts for teetering rotors having undersling, *
C * a concentrated hubmass and delta-3. *
C *
C *****
```

```
C *****
C *
C *                               Module 1                               *
C * *
C * This program is the first of two modules that consti- *
C * tute the STRAP code. This first module interpolates the *
C * input data into a form usable by the second module *
C * which does the actual modeling. This module also com- *
C * puts the coefficient matrix used for solving the blade *
C * flaps equation of motion. *
C *
C *****
```

```
C *****
C *
C * Questions about the original formulation, theory, meth- *
C * od of solution and programming should be addressed to: *
C * *
C * Alan D. Wright *
C * Wind Program *
C * National Renewable Energy Lab *
C * 1617 Cole Blvd. *
C * Golden, CO 80401 *
```



```

C      *              (303)231-7651              *
C      *                                              *
C      *****
C
C      *****
C      *                                              *
C      *      I/O Conventions:                      *
C      *      *
C      *      Unit 1 - Input data file              *
C      *      Unit 2 - Output data file            *
C      *      Unit 5 - Keyboard                    *
C      *      Unit * - Monitor or keyboard         *
C      *
C      *****
C
C      *****
C      *      External references in this routine:  *
C      *      *
C      *      COEFFS - Subroutine that computes the K and M coef- *
C      *      ficients used in solving the equations of          *
C      *      blade motion.                                     *
C      *
C      *      INPUT - Subroutine that reads in a set of blade and *
C      *      machine data and converts them to a form          *
C      *      needed by the COEFFS subroutine.                 *
C      *
C      *      PRNCOE - Subroutine that creates the input data file *
C      *      for the second module.                          *
C      *
C      *****
C
C      *****
C      *      Local and dummy variables used in this routine:  *
C      *      *
C      *      ANS   - Used to store input responses from the key- *
C      *      board.                                           *
C      *      FILOUT - String that contains the name of the out- *
C      *      put file. It is defined in INPUT.                 *
C      *
C      *      NP    - Number of blade property values used in    *
C      *      performing composite Simpson's integration         *
C      *      for the K and M coefficient arrays. This           *
C      *      number is approximately 10 times the NPTS          *
C      *      value.                                             *
C      *
C      *      NPTS  - Number of points along the blade used to  *
C      *      perform Simpson's integration for calculat-        *
C      *      ing the moments and forces at the blade           *
C      *      root.                                             *
C      *
C      *      NSIMP - Order of the composite Simpson's integra- *
C      *      tion used in the run. Parameter is set to        *
C      *      10 in order to make 21 blade property stations.  *
C      *
C      *****

```

```

INTEGER      NP
INTEGER      NPTS
INTEGER      NSIMP

```

```

CHARACTER*1  ANS
CHARACTER*50 FILOUT

```

```

PARAMETER    ( NSIMP = 10 )

```

```

100 FORMAT (
&  '//      Program For Analysis Of a 2-Bladed Teetering Hub Rotor'
&  '/      Dynamic Response and Loads Analysis'
&  '//      MODULE 1- STRAP1' )
110 FORMAT ( / )
120 FORMAT ( A )
130 FORMAT ( / ' STRAP1 terminated normally.' / )

```

```

C      Calculate the number of blade property stations and interpola-
C      tion points.

```

```
NPTS = 2*NSIMP + 1
NP    = 20*NSIMP + 1
```

```
C      Print title.
```

```
PRINT 100
```

```
C      Get input responses, generate coefficients, and write to output
C      file.
```

```
10 PRINT 110
```

```
CALL INPUT ( NP , NPTS , FILOUT )
CALL MODES (NP)
CALL COEFFS ( NP , NPTS )
CALL PRNCOE ( FILOUT , NPTS )
```

```
C      Check to see if user wants to process another data file.
```

```
20 PRINT *, ' '
PRINT *, 'Do you want to process another data file? (Y,=N) > '
READ 120, ANS

IF ( ( ANS .EQ. 'Y' ) .OR. ( ANS .EQ. 'y' ) ) GO TO 10

IF ( ( ANS .NE. 'N' ) .AND. ( ANS .NE. 'n' )
&   .AND. ( ANS .NE. ' ' ) ) THEN
  PRINT *, 'Invalid response. Please try again...'
  GO TO 20
END IF
```

```
C      Processing complete.
```

```
PRINT 130
```

```
STOP
END
BLOCK DATA
```

```
C      *****
C      *
C      *   This module is used to initialize the COMMON blocks.   *
C      *
C      *****
C
C      *****
C      *
C      *   Named COMMON blocks used in this routine:
C      *
C      *   AERO - Aerodynamic blade properties derived from
C      *           input aerodynamic data.
C      *   BLADE - Blade position dependent values that are
C      *           computed by linear interpolation of the in-
C      *           put property data.
C      *   LITERL - Data file titles used for printed output.
C      *   LODVAL - Holds components used in calculating blade
C      *           loads and deflections.
C      *   MATRX1 - Holds some of the coefficient matrices of
C      *           the governing equation of motion.
C      *   MATRX2 - Holds some of the coefficient matrices of
C      *           the governing equation of motion.
C      *   PANELS - Used to communicate with the interpolation
C      *           subroutine INTERP.
C      *   TENSIN - Holds tension components of the stiffness
C      *           functions.
C      *   TURBN - Turbine related variables.
C      *   WIND - Wind related variables.
C      *
C      *****
C      *****
```

```

C      *
C      *   External references in this routine:
C      *
C      *   none
C      *
C      *
*****
C      *
C      *   COMMON variables used in Module 1 of STRAP:
C      *
C      *   ALENTH - Distance from the tower axis to the rotor
C      *             hub. (feet)
C      *   ALPHAO - The angle from the zero-lift line to the
C      *             section-chord line. (Generally negative)
C      *   BETAO  - Blade coning angle. The value is input in
C      *             degrees and converted to radians. (radians)
C      *   BLSHNK - Length of blade shank measured from the
C      *             blade root to the start of the airfoil section.
C      *             If airfoil begins at the root, then
C      *             BLSHNK=0. (feet)
C      *   BLTIP  - Blade length measured from the blade root
C      *             to the blade tip. Note - rotor radius is
C      *             HUBRAD + BLTIP. (feet)
C      *   CDZERO - Drag coefficient values at equidistant
C      *             points along the blade. Derived from
C      *             ACDZER values. (dimensionless)
C      *   CHI    - Rotor tilt angle. Input in degrees and
C      *             converted to radians. (radians)
C      *   CHORD  - Blade chord at equidistant points along the
C      *             blade. Derived from ACHORD. (feet)
C      *   C1FMOM - Integral of D1FMOM.
C      *   CKBEND - Bending stiffness matrix.
C      *   CKQLOD - Inertial moment stiffening matrix.
C      *   CKTCRL - Coriolis stiffening matrix.
C      *   CKTGRV - Gravity stiffening matrix.
C      *   CKTMG  - Centrifugal stiffening matrix.
C      *   CLALFA - Slope of the lift curve at equidistant
C      *             points along the blade. Derived from ACLALF
C      *             values. (radians^-1)
C      *   CLMAX  - The maximum or stall value of the lift co-
C      *             efficient. Derived from ACLMAX values.
C      *   CMBLNC - Coefficient associated with the blade mass
C      *             imbalance (OFFSET).
C      *   CMGRAV - Mass coefficients associated with gravita-
C      *             tional loads.
C      *   CMGRV1 - Submatrix of CMGRAV containing odd terms
C      *             only. Used only with teetering hubs.
C      *   CMMASS - Blade mass matrix. It is also used in the
C      *             inertia force stiffening term.
C      *   CMRGD1 - Submatrix of CMRIGD containing odd terms
C      *             only. Used only with teetering hubs.
C      *   CMRIGD - Mass coefficients associated with rigid bo-
C      *             dy motion.
C      *   CSUBMA - Pitching moment coefficient.
C      *   DELTIM - Integral of DDELTI.
C      *   DRGFRM - Drag coefficient form constant.
C      *   ECNTFN - Complicated term.
C      *   EIAREA - Moment of inertia values at equidistant
C      *             points along the blade. Derived from
C      *             AEIARE data values. (Lb-Ft**2)
C      *   ESUBAC - Distance from the blade elastic axis to the
C      *             aerodynamic center. Positive if elastic
C      *             axis is forward (toward leading edge) of
C      *             aerodynamic center. Derived from AESBAC
C      *             values. (feet)
C      *   HUBHT  - Hub reference height above the ground.
C      *             (feet)
C      *   HUBRAD - Radius of rotor hub. (feet)
C      *   IEMASS - Edgewise mass moment of inertia at equidis-
C      *             tant points along the blade section. Der-
C      *             ived from A1EMAS values. (Lb-sec^2)
C      *   IFMASS - Flapwise mass moment of inertia at equidis-
C      *             tant points along the blade section. Der-
C      *             ived from A1FMAS values. (Lb-sec^2)
C      *   KSHADW - Number of tower shadow peaks within the
C      *             tower shadow zone.
C      *   MASS   - Blade mass per unit length at equidistant

```

```

C      *           points along the blade. Input in Lbf/ft, *
C      *           via the input variable WEIGHT, and convert- *
C      *           ed to slugs/ft. (slugs/ft) *
C      * NBLADS - Number of turbine blades. *
C      * NSHAPS - Number of blade shape functions, 4 maximum. *
C      * * *
C      * OFFMAS - Integral of DOFFMS. *
C      * OFFSET - Distance from the elastic axis to the mass *
C      * axes of blade section. Positive towards *
C      * the leading edge. Derived from AOFFST values. *
C      * Called E-sub-eta in the formulation. *
C      * (feet) *
C      * OMEGA - Rotor speed. Input in RPM and converted to *
C      * radians/second. (rad/sec) *
C      * PHIO - Rotor mean yaw angle. (radians) *
C      * PHIAMP - Amplitude of periodic yaw motion about mean *
C      * yaw angle. Input as degrees and converted *
C      * to radians. (radians) *
C      * PHIOMG - Maximum yaw rate. Used internally to com- *
C      * pute the yaw period. It can be used to input *
C      * a turbine steady yaw rate. Input as *
C      * degrees/sec and converted to radians/sec- *
C      * ond. (radians/sec) *
C      * PSIZER - Half angle width of the tower shadow re- *
C      * gion. (degrees) *
C      * SHERXP - Wind shear power exponent. *
C      * STEP - Distance between the equidistant data *
C      * points along the blade. (feet) *
C      * TCORLS - Blade tension coefficient due to coriolis *
C      * effects. *
C      * TGRAV - Blade tension coefficient due to gravity *
C      * effects. *
C      * THETAO - Orientation of the zero lift line with re- *
C      * spect to the blade principal bending axis. *
C      * THETAP - The angle *
C      * from the bending axis, XP, to the cone of *
C      * rotation, X-axis, for the reference sta- *
C      * tion. THETAP establishes the orientation *
C      * of the flapping displacements. Positive *
C      * angles are toward feather. Generally, the *
C      * reference station section-chord line is *
C      * taken as the bending (flapping) axis. *
C      * (radians) *
C      * THETAT - The built-in blade twist angle from the *
C      * section-chord line at the reference station *
C      * to the section-chord line at the tip of the *
C      * rotor. Positive towards feather. *
C      * TITLE1 - First line of the data file title. *
C      * TITLE2 - Second line of the data file title. *
C      * TITLE3 - Third line of the data file title. *
C      * TOMGA - Blade tension coefficient associated with *
C      * centrifugal force effects. *
C      * TSUBO - Tower shadow wind speed offset component. *
C      * TSUBP - Tower shadow sinusoidal component. *
C      * VHUB - Air speed at the height of the hub. *
C      * (ft/sec) *
C      * XLEFT - Radial positions of the blade property data *
C      * points. These are monotonically increasing *
C      * values from 0.0 at the root to R at the *
C      * tip. (feet) *
C      * *
C      *****

```

```

INCLUDE      'C:\INCLUDE\AERO.INC'
INCLUDE      'C:\INCLUDE\BLADE.INC'
INCLUDE      'C:\INCLUDE\LITERL.INC'
INCLUDE      'C:\INCLUDE\LODVAL.INC'
INCLUDE      'C:\INCLUDE\MATRX1.INC'
INCLUDE      'C:\INCLUDE\MATRX2.INC'
INCLUDE      'C:\INCLUDE\PANELS.INC'

```

```

INCLUDE      'C:INCLUDE\TENSIN.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\WIND.INC'

```

```

END
SUBROUTINE CAPS ( STRING )

```

```

C      *****
C      *
C      * Subroutine CAPS is used to convert alphabetic charac *
C      * ters into upper case. It assumes that all the lower *
C      * case letters are in one contiguous block and all the *
C      * upper case letters are in another contiguous block. *
C      * This routine will need to be changed to work with a *
C      * noncontiguous character set like IBM's EBCDIC. It is *
C      * not needed for an upper case only character set like *
C      * CDC's Display Code.
C      *
C      *****

```

```

C      *****
C      *
C      * Named COMMON blocks used in this routine:
C      *
C      * none
C      *
C      *****

```

```

C      *****
C      *
C      * External references in this routine:
C      *
C      * none
C      *
C      *****

```

```

C      *****
C      *
C      * Local and dummy variables used in this routine:
C      *
C      * DIFF - Numerical difference between 'A' and 'a'.
C      * I - Generic index.
C      * LENGTH - The declared length of the given string.
C      * STRING - The string needing to be converted to upper
C      * case.
C      *
C      *****

```

```

INTEGER      DIFF
INTEGER      I
INTEGER      LENGTH

```

```

CHARACTER*(*) STRING

```

```

C      Compute the numerical difference between 'A' and 'a'.

```

```

DIFF = ICHAR( 'a' ) - ICHAR( 'A' )

```

```

C      Get the length of the string.

```

```

LENGTH = LEN( STRING )

```

```

C      Look for lower case letters. Convert them to upper case.

```

```

DO 100 I=1,LENGTH

```

```

IF ( ( STRING(I:I) .GE. 'a' ) .AND.
& ( STRING(I:I) .LE. 'z' ) ) THEN

```

```

STRING(I:I) = CHAR( ICHAR( STRING(I:I) ) - DIFF )

```

END IF

100 CONTINUE

RETURN
END
SUBROUTINE COEFFS (NP , NPTS)

```
C *****  
C *  
C * This subroutine computes the coefficient matrices used *  
C * in the solution of the blade equations of motion. *  
C * Since these coefficients are independent of the posi- *  
C * tion or state of the blade, and are only dependent upon *  
C * the blade properties, they are calculated independently *  
C * of the solution to the equations of motion and are made *  
C * available to the rest of the program through the COMMON *  
C * blocks MATRX1 and MATRX2. *  
C *  
C *****  
C *****  
C *  
C * Named COMMON blocks used in this routine: *  
C *  
C * AERO - Aerodynamic blade properties derived from *  
C * input aerodynamic data. *  
C * BLADE - Blade position dependent values that are *  
C * computed by linear interpolation of the in- *  
C * put property data. *  
C * LODVAL - Holds components used in calculating blade *  
C * loads and deflections. *  
C * MATRX1 - Holds some of the coefficient matrices of *  
C * the governing equation of motion. *  
C * MATRX2 - Holds some of the coefficient matrices of *  
C * the governing equation of motion. *  
C * TENSIN - Holds tension components of the stiffness *  
C * functions. *  
C *  
C *****  
C *****  
C *  
C * External references in this routine: *  
C *  
C * SIMPSN - Function that performs the composite Simp- *  
C * son's integration on the input data arrays. *  
C * TRPZOD - Function that performs composite trapezoi- *  
C * dal integration. *  
C *  
C *****  
C *****  
C *  
C * Local and dummy variables used in this routine: *  
C *  
C * BLTBLT - Square of BLTIP. (feet**2) *  
C * DDELT I - IEMASS - IFMASS. *  
C * DIFMOM - Complicated term. *  
C * DKBEND - Used to compute the integrals associated *  
C * with the coefficient matrices. *  
C * DKMASS - Used to compute the integrals associated *  
C * with the coefficient matrices. *  
C * DKQLD - Used to compute the integrals associated *  
C * with the coefficient matrices. *  
C * DKTCRL - Used to compute the integrals associated *  
C * with the coefficient matrices. *  
C * DKTGRA - Used to compute the integrals associated *  
C * with the coefficient matrices. *  
C * DKTOMG - Used to compute the integrals associated *  
C * with the coefficient matrices. *  
C * DMBALN - Used to compute the integrals associated *  
C * with the coefficient matrices. *  
C * DMGRAV - Used to compute the integrals associated *  
C * with the coefficient matrices. *
```

```

C      *      DMRIGD - Used to compute the integrals associated *
C      *      with the coefficient matrices. *
C      *      DOFFMS - OFFSET*MASS. *
C      *      DTCOR1 - Used to compute the integrals associated *
C      *      with the tension components. *
C      *      DTCOR2 - Used to compute the integrals associated *
C      *      with the tension components. *
C      *      DTCOR3 - Used to compute the integrals associated *
C      *      with the tension components. *
C      *      DTCOR4 - Used to compute the integrals associated *
C      *      with the tension components. *
C      *      DTOMGA - Used to compute the integrals associated *
C      *      with the tension components. *
C      *      I      - Generic index. *
C      *      IPT     - Array index used for trapezoidal integra- *
C      *      tion. *
C      *      K.      - Generic index. *
C      *      L      - Generic index. *
C      *      MARK1  - Used to determine which shape function to *
C      *      use. *
C      *      MARKK  - Used to determine which shape function to *
C      *      use. *
C      *      MARKK1 - Used to determine which shape function to *
C      *      use. *
C      *      MARKN  - Used to determine which shape function to *
C      *      use. *
C      *      N      - Generic index. *
C      *      NP     - Number of blade property values used in *
C      *      performing composite Simpson's integration *
C      *      for the M and K coefficient arrays. This *
C      *      number is approximately 10 times the NPTS *
C      *      value. (passed from STRAP1) *
C      *      NPTS   - Number of points along the blade used to *
C      *      perform Simpson's integration for calculat- *
C      *      ing the moments and forces at the blade *
C      *      root. (passed from STRAP1) *
C      *      SHP    - Array containing shape functions evaluated *
C      *      at regular intervals. *
C      *      SHPDD  - First derivative of SHAPE with respect to *
C      *      location along the blade. *
C      *      SHPDOT - Second derivative of SHAPE with respect to *
C      *      location along the blade. *
C      *      X      - Dummy variable used to indicate location *
C      *      along the blade for use by the internal *
C      *      shape functions. *
C      *
C      *****

```

```

REAL      DDELT1 (201)
REAL      DIFMOM (201)
REAL      DKBEND (201)
REAL      DKMASS (201)
REAL      DKQLD  (201)
REAL      DKTCRL (201)
REAL      DKTGRA (201)
REAL      DKTOMG (201)
REAL      DMBALN (201)
REAL      DMGRAV (201)
REAL      DMRIGD (201)
REAL      DOFFMS (201)
REAL      DTCOR1 (201)
REAL      DTCOR2 (201)
REAL      DTCOR3 (201)
REAL      DTCOR4 (201)
REAL      DTOMGA (201)
REAL      HUBCOF
REAL      SIMPSN
REAL      STEP
REAL      TRPZOD

```

```

INTEGER   I
INTEGER   IPT
INTEGER   K
INTEGER   L
INTEGER   MARK1
INTEGER   MARK2
INTEGER   MARKK
INTEGER   MARKK1

```

```

INTEGER      MARKN
INTEGER      N
INTEGER      NFLAG
INTEGER      NP
INTEGER      NPTS

INCLUDE      'C:INCLUDE\AERO.INC'
INCLUDE      'C:INCLUDE\BLADE.INC'
INCLUDE      'C:INCLUDE\CONST.INC'
INCLUDE      'C:INCLUDE\LODVAL.INC'
INCLUDE      'C:INCLUDE\MATRIX1.INC'
INCLUDE      'C:INCLUDE\MATRIX2.INC'
INCLUDE      'C:INCLUDE\TENSIN.INC'
INCLUDE      'C:INCLUDE\MODAL.INC'
INCLUDE      'C:INCLUDE\HUBPRP.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\SPRING.INC'

```

```

1000 FORMAT ( / ' Generating coefficient matrices and property'
&
, ' arrays...' )
8000 FORMAT ( ' &done.' )

```

```

C *****
C *
C *   Fill the intermediate function arrays which are used to *
C *   compute the tension component integrals. *
C *
C *****

```

```
STEP = BLTIP/( NP - 1.0 )
```

```
DO 200 I=1, NP
```

```

DTOMGA(I) = MASS(I)*( HUBRAD + STEP*( I - 1 ) )
DTCOR1(I) = MASS(I)*SHP(1,I)
DTCOR2(I) = MASS(I)*SHP(2,I)
DTCOR3(I) = MASS(I)*SHP(3,I)
DTCOR4(I) = MASS(I)*SHP(4,I)

```

```
200 CONTINUE
```

```

C *****
C *
C *   Compute the tension component integrals. Each tension *
C *   component is calculated by integrating from a specific *
C *   position on the blade out to the blade tip. *
C *
C *****

```

```
DO 300 I=1, NP
```

```

TOMGA(I)   = TRPZOD( DTOMGA , I , NP , BLTIP )
TGRAV(I)   = TRPZOD( MASS , I , NP , BLTIP )

TCORLS(1,I) = TRPZOD( DTCOR1 , I , NP , BLTIP )
TCORLS(2,I) = TRPZOD( DTCOR2 , I , NP , BLTIP )
TCORLS(3,I) = TRPZOD( DTCOR3 , I , NP , BLTIP )
TCORLS(4,I) = TRPZOD( DTCOR4 , I , NP , BLTIP )

```

```
300 CONTINUE
```

```
C *****
```



```

C      *
C      *   Compute the coefficient matrices. Please note that   *
C      *   only the upper triangle elements are computed.     *
C      *
C      *****

```

```

NFLAG = 1

```

```

DO 470 K=1,4

```

```

    DO 440 L=K,4

```

```

        MARK1 = 1 + NFLAG*(-1)**( K + L )
        MARK2 = 1 + NFLAG*(-1)**( K + L + 1 )

```

```

        IF ( MARK1 .NE. 0 ) THEN

```

```

C            Fill the intermediate function arrays which are used
C            to compute the stiffness and loading coefficient ma-
C            trices.

```

```

            DO 400 I=1,NP

```

```

                DKBEND(I) = EIAREA(I)*SHPDD (K,I)*SHPDD (L,I)
                DKMASS(I) = MASS (I)*SHP (K,I)*SHP (L,I)
                DKQLD (I) = IFMASS(I)*SHPDOT(K,I)*SHPDOT(L,I)
                DKTOMG(I) = TOMGA (I)*SHPDOT(K,I)*SHPDOT(L,I)

```

```

400            CONTINUE

```

```

C            Compute the K,Lth element of the coefficient matrices.

```

```

            CKBEND(K,L) = SIMPSN( 0.0 , BLTIP , NP , DKBEND )
            CKTOMG(K,L) = SIMPSN( 0.0 , BLTIP , NP , DKTOMG )
            CKQLD(K,L) = IFMASS(NP)*SHPDOT(K,NP)*SHP(L,NP)
            &          - SIMPSN( 0.0 , BLTIP , NP , DKQLD )
            CMMASS(K,L) = SIMPSN( 0.0 , BLTIP , NP , DKMASS )

```

```

            ELSE

```

```

                CKBEND(K,L) = 0.0
                CKTOMG(K,L) = 0.0
                CKQLD(K,L) = 0.0
                CMMASS(K,L) = 0.0

```

```

            END IF

```

```

            IF ( MARK2 .NE. 0 ) THEN

```

```

410            DO 410 I=1,NP
                DKTGRA(I) = TGRAV(I)*SHPDOT(K,I)*SHPDOT(L,I)

```

```

                CKTGRV(K,L) = SIMPSN( 0.0 , BLTIP , NP , DKTGRA )

```

```

            ELSE

```

```

                CKTGRV(K,L) = 0.0

```

```

            END IF

```

```

C            Index into coefficient matrix for coriolis stiffening.

```

```

DO 430 N=1,4

```

```

    MARKN = 1 + NFLAG*(-1)**( K + L + N )

```

```

    IF ( MARKN .NE. 0 ) THEN

```

```

420            DO 420 I=1,NP
                DKTCTRL(I) = TCORLS(N,I)*SHPDOT(K,I)*SHPDOT(L,I)

```

```

                CKTCTRL(N,K,L) = SIMPSN( 0.0 , BLTIP , NP , DKTCTRL )

```

```

            ELSE

```

```

                CKTCRL(N,K,L)=0.0
            ENDIF
430      CONTINUE
                CKTCRL(1,K,L)=0.
440      CONTINUE
C          Compute elements of coefficient matrices which use only a
C          single index. Fill intermediate function arrays first, then
C          compute the elements of the matrices.
MARKK = 1 + NFLAG*(-1)**K
MARKK1 = 1 + NFLAG*(-1)**( K + 1 )
IF ( MARKK .NE. 0 ) THEN
    DO 450 I=1,NP
        DMRIGD(I) = MASS(I)*( HUBRAD + STEP*( I-1 ) )*SHP(K,I)
        DMBALN(I) = MASS(I)*SHP(K,I)*OFFSET(I)
        DMGRAV(I) = MASS(I)*SHP(K,I)
450      CONTINUE
        CMRIGD(K) = SIMPSN( 0.0 , BLTIP , NP , DMRIGD )
        CMBLNC(K) = SIMPSN( 0.0 , BLTIP , NP , DMBALN )
        CMGRAV(K) = SIMPSN( 0.0 , BLTIP , NP , DMGRAV )
    ELSE
        CMRIGD(K)=0.
        CMBLNC(K)=0.
        CMGRAV(K)=0.
    ENDIF
    IF ( MARKK1 .NE. 0 ) THEN
        DO 460 I=1,NP
            DMRIGD(I) = MASS(I)*( HUBRAD + STEP*( I-1 ) )*SHP(K,I)
            DMBALN(I) = MASS(I)*SHP(K,I)*OFFSET(I)
            DMGRAV(I) = MASS(I)*SHP(K,I)
460      CONTINUE
            CMRIGD1(K) = SIMPSN( 0.0 , BLTIP , NP , DMRIGD )
            CMGRV1(K) = SIMPSN( 0.0 , BLTIP , NP , DMGRAV )
        ELSE
            CMRIGD1(K) = 0.0
            CMGRV1(K) = 0.0
        .ENDIF
470 CONTINUE
C          *****
C          *
C          *   Once the diagonal and upper triangular elements are
C          *   computed, reflect the upper triangle onto the lower
C          *   one.
C          *
C          *****
DO 520 K=1,3
    DO 510 L=K+1,4
        CKBEND(L,K) = CKBEND(K,L)

```

```

        CKQLOD(L,K) = CKQLOD(K,L)
        CKTGRV(L,K) = CKTGRV(K,L)
        CKTOMG(L,K) = CKTOMG(K,L)
        CMMASS(L,K) = CMMASS(K,L)

500      DO 500 N=1,4
          CKTCRL(N,L,K) = CKTCRL(N,K,L)

510      CONTINUE

520      CONTINUE

C        Multiply certain matrix elements by CTHP in order to
C        to take care of the difference in orientation between
C        the teetering motion frame and the flapping axes.

        CTHP = COS(THETAP*3.1415926536/180.)

        CMMASS(1,3)=CMMASS(1,3)*CTHP
        CMMASS(3,1)=CMMASS(3,1)*CTHP
        CKTGRV(1,2)=CKTGRV(1,2)*CTHP
        CKTGRV(2,1)=CKTGRV(2,1)*CTHP
        CKTGRV(1,4)=CKTGRV(1,4)*CTHP
        CKTGRV(4,1)=CKTGRV(4,1)*CTHP
        CKTOMG(1,3)=CKTOMG(1,3)*CTHP
        CKTOMG(3,1)=CKTOMG(3,1)*CTHP

C        *****
C        *
C        *   The following arrays are used to compute the loads and *
C        *   moments for completion of MODULE 2.                   *
C        *
C        *****

DO 600 I=1,NP

        DDELT I(I) = IEMASS(I) - IFMASS(I)
        DOFFMS(I) = OFFSET(I)*MASS(I)

600      CONTINUE

        DO 610 I=1,NPTS

          IPT = 10*( I - 1 ) + 1

          DELTIM(I) = TRPZOD( DDELT I , IPT , NP , BLTIP )
          OFFMAS(I) = TRPZOD( DOFFMS , IPT , NP , BLTIP )

610      CONTINUE

        DO 640 N=1,4

          DO 620 I=1,NP
820          DIFMOM(I) = IFMASS(I)*SHPDOT(N,I)

          DO 630 I=1,NPTS

            IPT = 10*( I - 1 ) + 1

            CIFMOM(N,I) = TRPZOD( DIFMOM , IPT , NP , BLTIP )

630      CONTINUE

640      CONTINUE

C        *****
C        *
C        *   These tension and property arrays will be used in *
C        *   MODULE 2 to compute the loads and moments.           *
C        *
C        *****

DO 700 I=1,NPTS

          IPT = 10*( I - 1 ) + 1

```

```

    CHORD (1) = CHORD(IPT)
    ECNTFN(1) = OFFSET(IPT)*MASS(IPT)*( HUBRAD
&      + BLTIP*( I-1 )/( NPTS-1 ) )
    TGRAV (1) = TGRAV(IPT)
    THETA0(1) = THETA0(IPT)
    TOMGA (1) = TOMGA(IPT)

    TCORLS(1,1) = TCORLS(1,IPT)
    TCORLS(2,1) = TCORLS(2,IPT)
    TCORLS(3,1) = TCORLS(3,IPT)
    TCORLS(4,1) = TCORLS(4,IPT)

```

700 CONTINUE

```

    IF ( UNDSLQ .EQ. 0.) THEN
        HUBCOF = 0.
    ELSE
        HUBCOF = 0.5*( UNDSLQ-HUBDIS)* HUBMAS/BLTIP
&      + TGRAV(1)*UNDSLQ/BLTIP
    ENDIF

```

```

    CMHUB1(1) = HUBCOF
    K0STIF(1) = K0/((BLTIP)**2)
    K1STIF(1) = K1/((BLTIP)**2)
    K2STIF(1) = K2/((BLTIP)**2)
    DAMP(1) = CDAMP/(BLTIP**2)

```

```

    DO 750 J = 2,NSHAPS
        CMHUB1(J) = 0.
        K0STIF(J) = 0.
        K1STIF(J) = 0.
        K2STIF(J) = 0.
        DAMP (J) = 0.

```

750 CONTINUE

```

C      *****
C      *
C      *   Generation of coefficient matrices and property arrays *
C      *   is complete. *
C      *
C      *****

```

RETURN
END

SUBROUTINE MODES(NP)

```

REAL CF1
REAL CF2
REAL CF2DOT
REAL CF2DD
REAL CF3
REAL CF3DOT
REAL CF3DD
REAL CF4
REAL CF4DOT
REAL CF4DD
REAL CF5
REAL CF5DOT
REAL CF5DD

```

```

REAL ASUM(8,8)
REAL BSUM(8,8)
REAL CSUM(8,8)
REAL DSUM(8,8)
REAL BLTBLT
REAL EKBEND(201)
REAL EKMASS(201)
REAL EKTOMG(201)
REAL ETOMGA(201)
REAL MAG(8)
REAL OMGA2
REAL SIMPSN
REAL STEP

```

```

REAL POLY(8,201)
REAL POLYDT(8,201)
REAL POLYDD(8,201)
REAL TTOMGA(201)
REAL TRPZOD
REAL X
REAL QPRIME(8,8)
REAL QTRANS(8,8)
REAL PRDCT1(8,8)
REAL PRDCT2(8,8)
REAL VALU(8)
REAL VECTR1(8,8)
REAL VECTR2(8,8)
REAL VECTR3(8,8)

```

```

INTEGER I
INTEGER K
INTEGER L
INTEGER NP
INTEGER NDM
INTEGER NDP

```

```

INCLUDE 'C:INCLUDE\BLADE.INC'
INCLUDE 'C:INCLUDE\MODAL.INC'
INCLUDE 'C:INCLUDE\TURBN.INC'
INCLUDE 'C:INCLUDE\CONST2.INC'
INCLUDE 'C:INCLUDE\SPRING.INC'
INCLUDE 'C:INCLUDE\MATRX1.INC'

```

CHARACTER*50 MODOUT

1100 FORMAT (A)

C Coordinate shape functions and their derivatives.
C These functions are polynomials that satisfy the
C boundary conditions for a cantilever blade, namely
C zero displacement and zero slope at the root.
C These functions will be used in linear combinations
C using a Rayleigh Ritz type procedure to determine the
C natural frequencies and modeshapes of the blade.

C Teetering shape

$$CF1(X) = HUBRAD/(BLTIP+HUBRAD)*(1.-X) + X$$

C Shape function for first symmetric bending

$$CF2(X) = X**2*(X*(X-4.0) + 6.)/3.$$

$$CF2DOT(X) = 4.*X*(X*(X-3.)+3.)/3.$$

$$CF2DD(X) = 4.*(X*(X-2.)+1.)$$

C Shape function for asymmetric bending and derivatives:

$$CF3(X) = X**3*(X*(3.*X-10.)+10.)/3.$$

$$CF3DOT(X) = 5.*X**2*(X*(3.*X-8.)+6.)/3.$$

$$CF3DD(X) = 20.*X*(X*(3.*X-6.)+3.)/3.$$

C Shape function for second symmetric bending and derivatives:

$$CF4(X) = X**4*(X*(2.*X - 6.) + 5.)$$

$$CF4DOT(X) = 2.*X**3*(X*(6.*X-15.) + 10.)$$

$$CF4DD(X) = 60.*X**2*(X*(X-2.) + 1.)$$

C Extra shape function and it's derivatives:

$$CF5(X) = X**5*(X*(10.*X - 28.) + 21.)/3.$$

$$CF5DOT(X) = 7.*X**4*(X*(10.*X - 24.) + 15.)/3.$$

$$CF5DD(X) = 14.*X**3*(X*(10.*X - 20.) + 10.)$$

C In the following set of polynomial shape function, every odd
C numbered array (i.e. POLY(1,I)) corresponds to an asymmetrical
C mode. Each even numbered member corresponds to a symmetrical mode.
C There are three functions for each subspace, corresponding to a
C total dimension of 6x6 for the matrices, to solve for the modeshapes.
C Only the lowest two modes in each subspace are retained, i.e. two

C asymmetrical modes, and two symmetrical modes. All coefficient matrices
 C passed on to module two will thus be of order 4x4.
 C Note that the first shape function for the asymmetrical modes is the
 C rigid body teeter mode, whereas the first mode for the symmetrical modes
 C is the polynomial CF2(x).

STEP = 1./(NP-1.)
 BLTBLT = BLTIP**2
 X= 0.

DO 110 I = 1, NP

POLY(1,I) = CF1(X)*CDELTA3
 POLYDT(1,I) = CDELTA3/(BLTIP+HUBRAD)
 POLYDD(1,I) = 0.

POLY(2,I) = CF2(X)
 POLYDT(2,I) = CF2DOT(X)/BLTIP
 POLYDD(2,I) = CF2DD(X)/BLTBLT

POLY(3,I) = CF2(X)
 POLYDT(3,I) = CF2DOT(X)/BLTIP
 POLYDD(3,I) = CF2DD(X)/BLTBLT

POLY(4,I) = CF3(X)
 POLYDT(4,I) = CF3DOT(X)/BLTIP
 POLYDD(4,I) = CF3DD(X)/BLTBLT

POLY(5,I) = CF3(X)
 POLYDT(5,I) = CF3DOT(X)/BLTIP
 POLYDD(5,I) = CF3DD(X)/BLTBLT

POLY(6,I) = CF4(X)
 POLYDT(6,I) = CF4DOT(X)/BLTIP
 POLYDD(6,I) = CF4DD(X)/BLTBLT

POLY(7,I) = CF4(X)
 POLYDT(7,I) = CF4DOT(X)/BLTIP
 POLYDD(7,I) = CF4DD(X)/BLTBLT

POLY(8,I) = CF5(X)
 POLYDT(8,I) = CF5DOT(X)/BLTIP
 POLYDD(8,I) = CF5DD(X)/BLTBLT

X = X + STEP
 110 CONTINUE

C Fill the intermediate function arrays which are used to
 C compute the tension component integral for the
 C centrifugal stiffening matrix.

X = 0.
 STEP = BLTIP/(NP - 1.)

DO 200 I = 1, NP

ETOMGA(I) = MASS(I)*(HUBRAD + STEP*(I-1))
 200 CONTINUE

C Compute the tension component integral.

DO 300 I = 1, NP

TTOMGA(I) = TRPZOD(ETOMGA, I, NP, BLTIP)
 300 CONTINUE

C Compute the coefficient matrices: ASUM, BSUM, CSUM.
 C ASUM is the bending stiffness matrix, BSUM is the
 C centrifugal stiffening matrix, and CSUM is the
 C mass matrix.

DO 470 K = 1, 8

DO 440 L = K, 8

```

DO 400 I = 1, NP
    EKBEND(I) = EIAREA(I)*POLYDD(K,I)*POLYDD(L,I)
    EKMASS(I) = MASS(I)*POLY(K,I)*POLY(L,I)
    EKTOMG(I) = TTOMGA(I)*POLYDT(K,I)*POLYDT(L,I)
400 CONTINUE
C      Compute the K,L th element of the coefficient matrices.
    ASUM(K,L) = SIMPSN(O., BLTIP, NP, EKBEND)
    BSUM(K,L) = SIMPSN(O., BLTIP, NP, EKTOMG)
    CSUM(K,L) = SIMPSN(O., BLTIP, NP, EKMASS)
440 CONTINUE
470 CONTINUE
C      Once the diagonal and upper triangular elements are
C      computed, reflect the upper triangle onto the lower
C      one.

```

```

ASUM(1,2)=0.
ASUM(1,4)=0.
ASUM(1,6)=0.
ASUM(1,8)=0.
ASUM(2,3)=0.
ASUM(2,5)=0.
ASUM(2,7)=0.
ASUM(3,4)=0.
ASUM(3,6)=0.
ASUM(3,8)=0.
ASUM(4,5)=0.
ASUM(4,7)=0.
ASUM(5,6)=0.
ASUM(5,8)=0.
ASUM(6,7)=0.
ASUM(7,8)=0.

```

```

BSUM(1,2)=0.
BSUM(1,4)=0.
BSUM(1,6)=0.
BSUM(1,8)=0.
BSUM(2,3)=0.
BSUM(2,5)=0.
BSUM(2,7)=0.
BSUM(3,4)=0.
BSUM(3,6)=0.
BSUM(3,8)=0.
BSUM(4,5)=0.
BSUM(4,7)=0.
BSUM(5,6)=0.
BSUM(5,8)=0.
BSUM(6,7)=0.
BSUM(7,8)=0.

```

```

CSUM(1,2)=0.
CSUM(1,4)=0.
CSUM(1,6)=0.
CSUM(1,8)=0.
CSUM(2,3)=0.
CSUM(2,5)=0.
CSUM(2,7)=0.
CSUM(3,4)=0.
CSUM(3,6)=0.
CSUM(3,8)=0.
CSUM(4,5)=0.
CSUM(4,7)=0.
CSUM(5,6)=0.
CSUM(5,8)=0.
CSUM(6,7)=0.
CSUM(7,8)=0.

```

```

DO 520 K = 1,7

```

```

    DO 510 L = K+1,8

```

```

        ASUM(L,K) = ASUM(K,L)

```

```

      BSUM(L,K) = BSUM(K,L)
      CSUM(L,K) = CSUM(K,L)

```

```

510   CONTINUE
520   CONTINUE

```

```

NDIM = 8
NDP  = 8

```

```

C     Add the Centrifugal stiffening term to
C     the bending stiffness terms.
C     Form the new matrix DSUM

```

```

OMGA2 = OMEGA * OMEGA * .010966227

```

```

DO 1820 I = 1,NDIM
DO 1830 J = 1,NDIM
      DSUM(I,J) = ASUM(I,J) + OMGA2* BSUM(I,J)

```

```

1830 CONTINUE
1820 CONTINUE

```

```

CALL JACOBI(CSUM,NDIM,VALU,VECTR1)

```

```

CALL EIGSRT(VALU,VECTR1,NDIM,NDP)

```

```

DO 880 I = 1, NDIM
DO 870 J = 1, NDIM
      SVALU = SQRT(VALU(J))
      QPRIME(I,J) = VECTR1(I,J)/SVALU

```

```

870 CONTINUE
880 CONTINUE

```

```

DO 900 I = 1,NDIM
DO 890 J = 1,NDIM
      QTRANS(J,I) = QPRIME(I,J)

```

```

890 CONTINUE
900 CONTINUE

```

```

CALL MULT(QPRIME,DSUM,PRDCT1,NDIM,NDIM)
CALL MULT(PRDCT1,QTRANS,PRDCT2,NDIM,NDIM)

```

```

CALL JACOBI(PRDCT2,NDIM,VALU,VECTR2)
CALL EIGSRT(VALU,VECTR2,NDIM,NDP)
CALL MULT(VECTR2,QPRIME,VECTR3,NDIM,NDIM)

```

```

C     The frequencies of interest are actually the square roots
C     of the calculated eigenvalues.

```

```

DO 915 J = 1,NDIM
      SUMSQ = 0.
DO 910 I = 1,NDIM
      SUMSQ = SUMSQ + VECTR3(I,J)**2

```

```

910   CONTINUE
      MAG(J) = SQRT(SUMSQ)

```

```

915 CONTINUE

```

```

DO 920 I = 2, NDIM
      FREQ(I) = SQRT( VALU(NDIM+1-I))

```

```

920 CONTINUE

```

```

FREQSQ = OMGA2+K0/(BLTIP**2*CSUM(1,1))
FREQ(1) = SQRT( FREQSQ)

```

```

DO 960 I = 1,NDIM
DO 940 J = 1,NDIM
      LAMDA(I,J) = VECTR3(I,NDIM+1-J)/MAG(NDIM+1-J)

```

```

940 CONTINUE
960 CONTINUE

```

```

PRINT*,'Enter name of modeshape and frequency file > '
READ 1100, MODOUT
PRINT *, MODOUT

```



```

OPEN(8, FILE = MODOUT, STATUS = 'UNKNOWN')

WRITE(8,*) 'FREQUENCIES = (RADIAN/SEC) '
WRITE(8,*) (FREQ(I), I=1,4)

DO 670 I = 1,4
  DO 660 J = 1,NP
    SHP(I,J) = 0.
    SHPDOT(I,J) = 0.
    SHPDD(I,J) = 0.
660 CONTINUE
670 CONTINUE

DO 700 I = 1,NP
  DO 690 J = 1,NDIM
    DO 680 K = 1,NDIM
      SHP(J,I) = SHP(J,I) + LAMDA(K,J)*POLY(K,I)
      SHPDOT(J,I) = SHPDOT(J,I) + LAMDA(K,J)*POLYDT(K,I)
      SHPDD(J,I) = SHPDD(J,I) + LAMDA(K,J)*POLYDD(K,I)

      IF( J .EQ. 1) THEN
        SHPDD(J,I) = 0.
      ENDIF
680 CONTINUE
690 CONTINUE
700 CONTINUE

DO 1400 I = 1,NDIM
  DO 1390 J = 1,NP
    SHP(I,J) = SHP(I,J)
    SHPDOT(I,J) = SHPDOT(I,J)
    SHPDD(I,J) = SHPDD(I,J)
1390 CONTINUE
1400 CONTINUE

WRITE(8,*) 'Modeshapes:'
DO 720 I = 1,NP,10
  WRITE(8,*) I, (SHP(J,I), J = 1,4)
720 CONTINUE

WRITE(8,*) ' '
WRITE(8,*) 'First Derivative of Modeshape'
DO 730 I = 1,NP,10
  WRITE(8,*) I, (SHPDOT(J,I), J=1,4)
730 CONTINUE

WRITE(8,*) ' '
WRITE(8,*) 'Second Derivative of Modeshape'
DO 740 I = 1,NP,10
  WRITE(8,*) I, (SHPDD(J,I), J=1,4)
740 CONTINUE

RETURN
END

SUBROUTINE JACOBI(A,N,D,V)
PARAMETER (NMAX=100)

REAL A(8,8)
REAL D(8)
REAL V(8,8)
REAL B(NMAX)
REAL Z(NMAX)

INTEGER N

DO 12 IP = 1,N
  DO 11 IQ = 1,N
    V(IP,IQ) = 0.
11 CONTINUE
    V(IP,IP) = 1.
12 CONTINUE

```

```

DO 13 IP = 1,N
  B(IP) = A(IP,IP)
  D(IP) = B(IP)
  Z(IP) = 0.
13 CONTINUE

NROT = 0

DO 24 I = 1,50
  SM = 0.
  DO 15 IP = 1,N-1
    DO 14 IQ = IP+1, N
      SM = SM + ABS(A(IP,IQ))
14 CONTINUE
15 CONTINUE

IF(SM .EQ. 0.) RETURN
IF(I .LT. 4) THEN
  TRESH = 0.2*SM/N**2
ELSE
  TRESH = 0.
ENDIF

DO 22 IP = 1,N-1
  DO 21 IQ = IP+1,N
    G = 100.*ABS(A(IP,IQ))
    IF( G .LT. 1.0E-08)THEN
      A(IP,IQ)=0.

    ELSEIF( ABS(A(IP,IQ)) .GT. TRESH)THEN
      H = D(IQ)-D(IP)
      IF(ABS(H) + G .EQ. ABS(H))THEN
        T = A(IP,IQ)/H
      ELSE
        THETA = 0.5*H/A(IP,IQ)
        T = 1./(ABS(THETA) + SQRT(1.+ THETA**2))
        IF(THETA .LT. 0.) T = -1.* T
      ENDIF

      C = 1./SQRT(1+T**2)
      S = T*C
      TAU = S/(1.+C)
      H = T*A(IP,IQ)
      Z(IP) = Z(IP) - H
      Z(IQ) = Z(IQ) + H
      D(IP) = D(IP) - H
      D(IQ) = D(IQ) + H
      A(IP,IQ) = 0.
      DO 16 J=1, IP-1
        G = A(J,IP)
        H = A(J,IQ)
        A(J,IP) = G-S*(H+G*TAU)
        A(J,IQ) = H+S*(G-H*TAU)
16 CONTINUE

      DO 17 J = IP+1, IQ-1
        G = A(IP,J)
        H = A(J,IQ)
        A(IP,J) = G-S*(H+G*TAU)
        A(J,IQ) = H+S*(G-H*TAU)
17 CONTINUE

      DO 18 J = IQ+1, N
        G = A(IP,J)
        H = A(IQ,J)
        A(IP,J) = G-S*(H+G*TAU)
        A(IQ,J) = H+S*(G-H*TAU)
18 CONTINUE

      DO 19 J = 1,N
        G = V(J,IP)
        H = V(J,IQ)
        V(J,IP) = G-S*(H+G*TAU)
        V(J,IQ) = H+S*(G-H*TAU)
19 CONTINUE

      NROT = NROT + 1
    ENDIF
  ENDIF

```

```

21     CONTINUE
22     CONTINUE

      DO 23 IP = 1,N
          B(IP) = B(IP) + Z(IP)
          D(IP) = B(IP)
          Z(IP) = 0.
23     CONTINUE
24     CONTINUE

```

```

RETURN
END

```

```

SUBROUTINE EIGSRT(D,V,N,NP)

```

```

DIMENSION D(NP), V(NP,NP)
DO 13 I = 1, N-1
    K=I
    P=D(I)
    DO 11 J=I+1,N
        IF(D(J) .GE. P)THEN
            K=J
            P=D(J)
        ENDIF
11     CONTINUE

    IF(K .NE. I) THEN
        D(K) = D(I)
        D(I) = P
        DO 12 J = 1,N
            P = V(J,I)
            V(J,I) = V(J,K)
            V(J,K) = P
12     CONTINUE
    ENDIF
13 CONTINUE
RETURN
END

```

```

SUBROUTINE INPUT ( NP , NPTS , FLOUT )

```

```

C *****
C *
C * Subroutine INPUT performs the following tasks: *
C *
C * 1 - Opens input and output data files. *
C * 2 - Reads in blade property and wind turbine data. *
C * 3 - Checks validity of input data. *
C * 4 - Performs a linear interpolation between data *
C * points to provide data at the proper points for *
C * the COEFFS subroutine. *
C * 5 - Performs unit conversions. *
C *
C *****
C
C *****
C *
C * External references in this routine: *
C *
C * CAPS - Converts strings to upper case. *
C * INTERP - Subroutine that performs a linear interpo- *
C * lation of the input data. *
C *
C *****
C
C *****
C *
C * Named COMMON blocks used in this routine: *
C *
C * AERO - Aerodynamic blade properties derived from *
C * input aerodynamic data. *
C * BLADE - Blade position dependent values that are *
C * computed by linear interpolation of the in- *
C * put property data. *
C * LITERL - Data file titles used for printed output. *
C * PANELS - Used to communicate with the interpolation *

```

```

C      *          subroutine INTERP.          *
C      *      TURBN  - Turbine related variables.  *
C      *      WIND   - Wind related variables.    *
C      *
C      *****
C
C      *****
C      *
C      *      Local and dummy variables used in this routine:
C      *
C      *      ACDZER - Drag coefficient at each blade station.
C      *              (dimensionless)
C      *      ACHORD - Blade chord at each data station. (feet)
C      *      ACLALF - Slope of lift curve at each blade station.
C      *              (radians**-1)
C      *      ACLMAX - The maximum or stall value of the lift co-
C      *              efficient.
C      *      AEIARE - Moment of inertia of blade at data station.
C      *              (10**6 Lb-Ft**2)
C      *      AESBAC - Distance from elastic axis to aerodynamic
C      *              center. Origin is at the elastic axis; It
C      *              is positive if the elastic axis is forward
C      *              (toward the leading edge) of the aerodynam-
C      *              ic center. (feet)
C      *      AIEMAS - Second mass moment of the blade cross sec-
C      *              tion in the edgewise direction. (Lb-sec^2)
C      *      AIFMAS - Second mass moment of the blade cross sec-
C      *              tion in the flapwise direction. (Lb-sec^2)
C      *      AOFFST - Distance from the elastic axis to the mass
C      *              axes of blade section. Positive towards
C      *              the leading edge. Called E-sub-eta in the
C      *              formulation. (feet)
C      *      ATWIST - The blade built-in twist angle as a func-
C      *              tion of position along the blade span. The
C      *              built-in twist at each of I stations.
C      *              ATWIST is the angle from the section-chord
C      *              line at the Ith station to the section-
C      *              chord line at the tip. Generally speaking,
C      *              ATWIST(Reference Station) = THETAT.
C      *      ERROR  - Error flag indicating an invalid XLEFT ar-
C      *              ray.
C      *      FILEIN - Literal containing the name of the input
C      *              data file.
C      *      FILOUT - Literal containing the name of the output
C      *              data file.
C      *      GRAV   - Constant of gravitational acceleration as
C      *              measured at sea level. (feet/second**2)
C      *      I      - Generic index.
C      *      NP     - Number of blade property values used in
C      *              performing composite Simpson's integration
C      *              for the M and K coefficient arrays. This
C      *              number is approximately 10 times the NPTS
C      *              value.
C      *      NPANEL - Number of blade property values to be read
C      *              in from the input data file.
C      *      NPTS   - Number of points along the blade used to
C      *              perform Simpson's integration for calculat-
C      *              ing the moments and forces at the blade
C      *              root.
C      *      RAD2DG - Degrees to radians conversion factor.
C      *      RTWIST - The blade twist angle relative to the zero-
C      *              lift line. (radians)
C      *      WEIGHT - Blade section weight per unit length. In-
C      *              put as blade station data. (Lbf/ft)
C      *
C      *****

```

```

REAL      ACDZER (11)
REAL      ACHORD (11)
REAL      ACLALF (11)
REAL      ACLMAX (11)
REAL      AEIARE (11)
REAL      AESBAC (11)
REAL      AIEMAS (11)
REAL      AIFMAS (11)
REAL      AOFFST (11)
REAL      ATWIST (11)

```

```

REAL          GRAV
REAL          RAD2DG
REAL          RTWIST (11)
REAL          WEIGHT (11)

INTEGER       I
INTEGER       NP
INTEGER       NPANEL
INTEGER       NPTS

LOGICAL       ERROR

CHARACTER*50  FILEIN
CHARACTER*50  FILOUT

INCLUDE       'C:INCLUDE\AERO.INC'
INCLUDE       'C:INCLUDE\BLADE.INC'
INCLUDE       'C:INCLUDE\LITERL.INC'
INCLUDE       'C:INCLUDE\PANELS.INC'
INCLUDE       'C:INCLUDE\TURBN.INC'
INCLUDE       'C:INCLUDE\TRBINF.INC'
INCLUDE       'C:INCLUDE\WIND.INC'
INCLUDE       'C:INCLUDE\HUBPRP.INC'
INCLUDE       'C:INCLUDE\SPRING.INC'

DATA          GRAV / 32.1740 /
DATA          RAD2DG / 57.29577951308233 /

```

```

1000 FORMAT ( / ' Subroutine INPUT: Input and process new blade'
& / ' and turbine data.' / )
1100 FORMAT ( A )
1200 FORMAT ( / ' *** Invalid response ***' / )
1300 FORMAT ( BN , 6X , F16.5 )
1400 FORMAT ( BN , 6X , I10 )
1500 FORMAT ( BN , 12X , 11( F10.5 , : ) )
1550 FORMAT ( BN , 12X , 11( F10.5 , : ) )
2000 FORMAT ( // 3( 1X , A / ) / ' Parameter values:' / )
2100 FORMAT ( 1X , A , F16.5 , 10X , A , I10 )
2200 FORMAT ( 1X , A , F16.5 , 10X , A , F16.5 )
2300 FORMAT ( 1X , A , I10 , 16X , A , F16.5 )
2400 FORMAT ( ' XLEFT WEIGHT AEIARE '
& ' AIEMAS AIFMAS AOFFST' / )
2500 FORMAT ( 11( F11.5 , 5F13.5 / : ) )
2600 FORMAT ( / ' ACHORD ATWIST ALCALF '
& ' ACLMAX ACDZER AESBAC' / )
2700 FORMAT ( / ' STA-' )
2750 FORMAT ( 11( F10.3/:))
3000 FORMAT ( / ' >>> Error: Blade data do not start at blade root:'
& / ' <<<'
& / ' >>>' , 50X , ' <<<'
& / ' >>> XLEFT(1) = ' , F8.3 , ' , but should be'
& / ' zero. <<<' )
3100 FORMAT ( / ' >>> Error: Blade data do not end at blade tip:'
& / ' <<<'
& / ' >>>' , 46X , ' <<<'
& / ' >>> XLEFT(' , 12.2 , ' ) = F8.3
& / ' , but should be <<<'
& / ' >>> equal to BLTIP = ' , F8.3
& / ' <<<' / )
3200 FORMAT ( / ' >>> Please correct input file and rerun job <<<'
& / ' STRAP terminated abnormally due to the error listed'
& / ' above.' / )

```

C Initialize the number of shape functions.

NSHAPS = 4

C Get input and output data file names. Open the input file.
 C Determine rotor type. Read data from input file.

```

PRINT 1000
PRINT *, 'Enter name of input data file > '
READ 1100, FILEIN
PRINT *, FILEIN

OPEN ( 1 , FILE=FILEIN , STATUS='OLD' )

PRINT *, ''
PRINT *, 'Enter name of output data file > '
READ 1100, FILOUT
PRINT *, FILOUT

CALL CAPS ( FILOUT )

```

```

110 READ (1,1100) TITLE1
    READ (1,1100) TITLE2
    READ (1,1100) TITLE3

    READ (1,1300) ALENTH
    READ (1,1300) ALPHAO
    READ (1,1300) BETA0
    READ (1,1300) BLSHNK
    READ (1,1300) BLTIP
    READ (1,1300) CHI
    READ (1,1300) CSUBMA
    READ (1,1300) DRGFRM
    READ (1,1300) HUBHT
    READ (1,1300) HUBRAD
    READ (1,1400) KSHADW
    READ (1,1400) NBLADS
    READ (1,1400) NPANEL
    READ (1,1300) OMEGA
    READ (1,1300) PHIAMP
    READ (1,1300) PHIOMG
    READ (1,1300) PHIO
    READ (1,1300) PSIZER
    READ (1,1300) SHERXP
    READ (1,1300) THETAP
    READ (1,1300) THETAT
    READ (1,1300) TSUBP
    READ (1,1300) TSUBO
    READ (1,1300) HUBMAS
    READ (1,1300) UNDSLK
    READ (1,1300) HUBDIS
    READ (1,1300) VHUB
    READ (1,1300) DELT3
    READ (1,1300) BETA1
    READ (1,1300) BETA2
    READ (1,1300) K0
    READ (1,1300) K1
    READ (1,1300) K2
    READ (1,1300) CDAMP

    READ (1,1500) ( XLEFT(I) , I=1,NPANEL )
    READ (1,1500) ( WEIGHT(I) , I=1,NPANEL )
    READ (1,1500) ( AEIARE(I) , I=1,NPANEL )
    READ (1,1500) ( AIEMAS(I) , I=1,NPANEL )
    READ (1,1500) ( AIFMAS(I) , I=1,NPANEL )
    READ (1,1500) ( AOFFST(I) , I=1,NPANEL )
    READ (1,1500) ( ACHORD(I) , I=1,NPANEL )
    READ (1,1500) ( ATWIST(I) , I=1,NPANEL )
    READ (1,1500) ( ACLALF(I) , I=1,NPANEL )
    READ (1,1500) ( ACLMAX(I) , I=1,NPANEL )
    READ (1,1500) ( ACDZER(I) , I=1,NPANEL )
    READ (1,1500) ( AESBAC(I) , I=1,NPANEL )
    READ (1,1400) NUMSCN
    READ (1,1300) TIMINC
    READ (1,1400) MSTAT
    READ (1,1550) ( STA(I) , I=1,MSTAT)

```

C Close input file.

CLOSE(1)

C Echo input data.

PRINT 2000, TITLE1 , TITLE2 , TITLE3

PRINT 2100, 'ALENTH = ' , ALENTH , 'NPANEL = ' , NPANEL
PRINT 2200, 'ALPHAO = ' , ALPHAO , 'OMEGA = ' , OMEGA
PRINT 2200, 'BETA0 = ' , BETA0 , 'PHIO = ' , PHIO
PRINT 2200, 'BLSHNK = ' , BLSHNK , 'PHIAMP = ' , PHIAMP
PRINT 2200, 'BLTIP = ' , BLTIP , 'PHIOMG = ' , PHIOMG
PRINT 2200, 'CHI = ' , CHI , 'PSIZER = ' , PSIZER
PRINT 2200, 'CSUBMA = ' , CSUBMA , 'SHERXP = ' , SHERXP
PRINT 2200, 'DRGFRM = ' , DRGFRM , 'THETAP = ' , THETAP
PRINT 2200, 'HUBHT = ' , HUBHT , 'THETAT = ' , THETAT
PRINT 2200, 'HUBRAD = ' , HUBRAD , 'TSUBO = ' , TSUBO
PRINT 2300, 'KSHADW = ' , KSHADW , 'TSUBP = ' , TSUBP
PRINT 2300, 'NBLADS = ' , NBLADS , 'VHUB = ' , VHUB
PRINT 2200, 'HUBMAS = ' , HUBMAS , 'UNDSLGL = ' , UNDSLGL
PRINT 2200, 'HUBDIS = ' , HUBDIS , 'DELTA3 = ' , DELT3
PRINT 2200, 'BETA1 = ' , BETA1 , 'BETA2 = ' , BETA2
PRINT 2200, 'K0 = ' , K0 , 'K1 = ' , K1
PRINT 2200, 'K2 = ' , K2
PRINT 2200, 'CDAMP = ' , CDAMP
PRINT 2300, 'NUMSCN = ' , NUMSCN
PRINT 2200, 'TIMINC = ' , TIMINC
PRINT 2300, 'MSTAT = ' , MSTAT

PRINT *, ' '
PRINT *, 'Hit <Enter> to continue...'

200 PRINT 2400
PRINT 2500, (XLEFT(I) , WEIGHT(I) , AEIARE(I) , AIEMAS(I)
& , AIFMAS(I) , AOFFST(I) , I=1, NPANEL)

PRINT *, ' '
PRINT *, 'Hit <Enter> to continue...'

210 PRINT 2600
PRINT 2500, (ACHORD(I) , ATWIST(I) , ACLALF(I) , ACLMAX(I)
& , ACDZER(I) , AESBAC(I) , I=1, NPANEL)

PRINT *, ' '
PRINT *, 'Hit <Enter> to continue...'

220 PRINT 2700
PRINT 2750, (STA(I) , I=1, MSTAT)

PRINT *, ' '
PRINT *, 'Hit <Enter> to continue...'

C Check validity of XLEFT array. XLEFT must start at blade root
C and end at blade tip.

300 ERROR = .FALSE.

IF (XLEFT(1) .NE. 0.0) THEN
PRINT 3000, XLEFT(1)
ERROR = .TRUE.
END IF

IF (XLEFT(NPANEL) .NE. BLTIP) THEN
PRINT 3100, NPANEL , XLEFT(NPANEL) , BLTIP
ERROR = .TRUE.
END IF

IF (ERROR) THEN
PRINT 3200
STOP
END IF

C Perform unit conversions - Lbm to slugs, degrees to radians.

```

DO 400 I=1,NPANEL
  WEIGHT(I) = WEIGHT(I)/GRAV
  AEIARE(I) = AEIARE(I)*1.E06
  AIEMAS(I) = AIEMAS(I)
  AIFMAS(I) = AIFMAS(I)
  RTWIST(I) = ( THETAT - ATWIST(I) - ALPHA0 )/RAD2DG
400 CONTINUE

```

C Calculate the cosine , sine, and tangent of the delta-3 angle

```

DELT3 = (3.141593/180.)*DELT3
CDELT3 = COS(DELT3)
SDELT3 = SIN(DELT3)
TDELT3 = TAN(DELT3)

```

C Perform linear interpolation of blade properties to set up
C arrays of blade property data at equidistant points along the
C blade. The arrays are used by the COEFFS subroutine to produce
C the coefficient matrices.

```
STEP = BLTIP/( NP - 1 )
```

```

CALL INTERP ( WEIGHT , MASS , NPANEL , NP )
CALL INTERP ( AIEMAS , IEMASS , NPANEL , NP )
CALL INTERP ( AIFMAS , IFMASS , NPANEL , NP )
CALL INTERP ( AEIARE , EIAREA , NPANEL , NP )
CALL INTERP ( AOFFST , OFFSET , NPANEL , NP )
CALL INTERP ( ACHORD , CHORD , NPANEL , NP )
CALL INTERP ( RTWIST , THETA0 , NPANEL , NP )

```

C These property arrays will be used by the RUN subroutine for
C solving the governing equations and for calculation of the
C loads and moments.

```
STEP = BLTIP/( NPTS - 1 )
```

```

CALL INTERP ( ACLALF , CLALFA , NPANEL , NPTS )
CALL INTERP ( ACDZER , CDZERO , NPANEL , NPTS )
CALL INTERP ( AESBAC , ESUBAC , NPANEL , NPTS )
CALL INTERP ( ACLMAX , CLMAX , NPANEL , NPTS )

```

```

RETURN
END
SUBROUTINE INTERP ( GIVEN , CMPUTD , NPANEL , NP )

```

```

C *****
C *
C * This subroutine performs a linear interpolation of the *
C * input data set. The input blade data contains NPANEL *
C * data points. The data are interpolated to provide data *
C * at NP evenly spaced points. *
C *
C *****

```

```

C *****
C *
C * Local and dummy variables used in this routine: *
C *
C * CMPUTD - Regularly spaced interpolated data. *
C * GIVEN - Unevenly spaced input data. *
C * IPNL - Index into the GIVEN array. *
C * JPT - Index into the CMPUTD array. *
C * MOVE - Flag to see if we've moved to the next pan- *
C * el. *
C * NP - Number of evenly spaced data points. *
C * NPANEL - Number of unevenly spaced data points. *
C * PTR - Blade position pointer. It is the location *
C * of the next interpolated value. *
C * SLOPE - Slope of the line between two GIVEN data *
C * points. *
C *
C *****

```



```

INTEGER      NP
INTEGER      NPANEL

REAL         CMPUTD (NP)
REAL         GIVEN  (NPANEL)
REAL         PTR
REAL         SLOPE

INTEGER      IPNL
INTEGER      JPT

LOGICAL      MOVE

INCLUDE      'C:\INCLUDE\PANELS.INC'

```

```

C      Initialize GIVEN index and pointer.

      IPNL = 1
      PTR  = 0.0

C      Compute slope between first two data points.

      SLOPE = ( GIVEN(2) - GIVEN(1) )/XLEFT(2)

C      Compute property at each of the NP evenly spaced points.

      DO 20 JPT=1,NP-1

          CMPUTD(JPT) = GIVEN(IPNL) + SLOPE*( PTR - XLEFT(IPNL) )

          PTR = PTR + STEP

C      Make sure that the new data point is inside the current
C      panel.  Otherwise, move over one step.

      IF ( JPT .LT. NP-1 ) THEN

          MOVE = .FALSE.

10      IF ( PTR .GT. XLEFT(IPNL+1) ) THEN
          IPNL = IPNL + 1
          MOVE = .TRUE.
          GO TO 10
          END IF

          IF ( MOVE ) SLOPE = ( GIVEN(IPNL+1) - GIVEN(IPNL) )/
&          ( XLEFT(IPNL+1) - XLEFT(IPNL) )

          END IF

20 CONTINUE

      CMPUTD(NP) = GIVEN(NPANEL)

      RETURN
      END
      FUNCTION LNTH ( STRING )

```

```

C      *****
C      *
C      *   Function LNTH returns the length of a character string.   *
C      *   When using the Lahey F77L compiler, the intrinsic func-   *
C      *   tion NBLANK can be used as we do here. This function     *
C      *   was supplied to make conversion to other compilers eas-   *
C      *   ier.                                                         *
C      *
C      *****

C      *****
C      *
C      *   Named COMMON blocks used in this routine:
C      *

```

```

C      *
C      *      none
C      *
C      *****

C      *****
C      *
C      *      Variables used in this routine:
C      *
C      *      IC      - Generic index.
C      *      LENGTH - The declared length of STRING.
C      *      LNTH   - The location of the last nonblank character
C      *                in STRING.
C      *      STRING - A character string.
C      *
C      *****

INTEGER      IC
INTEGER      LENGTH
INTEGER      LNTH

CHARACTER*(*) STRING

C      Get the declared length of STRING using the FORTRAN 77 intrinsic
C      function LEN.

LENGTH = LEN( STRING )

C      Find the location of the last nonblank character in STRING.

DO 100 IC=LENGTH,1,-1

LNTH = IC

IF ( STRING(IC:IC) .NE. ' ' ) GO TO 200

100 CONTINUE

C      STRING is all blanks.

LNTH = 0

200 RETURN
END
SUBROUTINE PRNCOE ( FILOUT , NPTS )

C      *****
C      *
C      *      This subroutine writes the data required to run MODULE
C      *      2 into the run data file. Variable names are included
C      *      in the file for informational purposes only. All vari-
C      *      ables written into the run data file have been defined
C      *      previously. Named common blocks AERO, BLADE, TURBN and
C      *      WIND first appear in subroutine INPUT. Named common
C      *      blocks MATRX1 and MATRX2 are computed in subroutine
C      *      COEFFS.
C      *
C      *****

C      *****
C      *
C      *      Named COMMON blocks used in this routine:
C      *
C      *      AERO  - Aerodynamic blade properties derived from
C      *                input aerodynamic data.
C      *      BLADE - Blade position dependent values that are
C      *                computed by linear interpolation of the in-
C      *                put property data.
C      *      LITERL - Data file titles used for printed output.
C      *      LODVAL -
C      *      MATRX1 - Holds some of the coefficient matrices of
C      *                the governing equation of motion.
C      *
C      *****

```

```

C      *      MATRIX2 - Holds some of the coefficient matrices of      *
C      *      the governing equation of motion.                        *
C      *      TENSIN - Holds tension components of the stiffness      *
C      *      functions.                                              *
C      *      TURBN - Turbine related variables.                      *
C      *      WIND - Wind related variables.                          *
C      *
C      *****
C      *****
C      *
C      *      Local and dummy variables used in this routine:        *
C      *
C      *      COL - Column index into matrices.                      *
C      *      FILOUT - Literal containing the name of the output      *
C      *      data file.                                             *
C      *      I - Generic index.                                     *
C      *      N - Generic index.                                     *
C      *      NPTS - Number of points along the blade used to        *
C      *      perform Simpson's integration for calculating           *
C      *      the moments and forces at the blade                    *
C      *      root. (passed from STRAP1)                             *
C      *      NSHPS - Number of blade shape functions. Set to 1      *
C      *      for use by MODULE 2.                                    *
C      *      ROW - Row index into matrices.                         *
C      *
C      *****

```

```

INTEGER      COL
INTEGER      I
INTEGER      LNTH
INTEGER      N
INTEGER      NPTS
INTEGER      NSHPS
INTEGER      ROW

```

```

CHARACTER*50  FILOUT

```

```

INCLUDE      'C:\INCLUDE\AERO.INC'
INCLUDE      'C:\INCLUDE\BLADE.INC'
INCLUDE      'C:\INCLUDE\LITERL.INC'
INCLUDE      'C:\INCLUDE\LODVAL.INC'
INCLUDE      'C:\INCLUDE\MATRX1.INC'
INCLUDE      'C:\INCLUDE\MATRX2.INC'
INCLUDE      'C:\INCLUDE\TENSIN.INC'
INCLUDE      'C:\INCLUDE\TRBINF.INC'
INCLUDE      'C:\INCLUDE\TURBN.INC'
INCLUDE      'C:\INCLUDE\WIND.INC'
INCLUDE      'C:\INCLUDE\MODAL.INC'
INCLUDE      'C:\INCLUDE\HUBPRP.INC'
INCLUDE      'C:\INCLUDE\SPRING.INC'

```

```

2000 FORMAT ( 3( A / ) / )
2100 FORMAT ( A )
3000 FORMAT ( / 13X , 'ALENTH' , 18X , 'ALPHA0' , 18X , ' BETA0'
& / 3( 1PE24.10 ) )
3100 FORMAT ( / 13X , 'BLSHNK' , 18X , ' BLTIP' , 18X , ' CHI '
& / 3( 1PE24.10 ) )
3200 FORMAT ( / 13X , 'CSUBMA' , 18X , 'DRGFRM' , 18X , ' HUBHT'
& / 3( 1PE24.10 ) )
3300 FORMAT ( / 13X , 'HUBRAD' , 18X , ' OMEGA' , 18X , ' PHIO '
& / 3( 1PE24.10 ) )
3400 FORMAT ( / 13X , 'PHIAMP' , 18X , 'PHIOMG' , 18X , 'PSIZER'
& / 3( 1PE24.10 ) )
3500 FORMAT ( / 13X , 'SHERXP' , 18X , 'THETAP' , 18X , 'THETAT'

```

```

&      / 3( 1PE24.10 ) )
3600 FORMAT ( / 13X , ' TSUB0' , 18X , ' TSUBP' 18X , ' VHUB '
&      / 3( 1PE24.10 ) )
3700 FORMAT ( / 10X , 'KSHADW' , 12X , 'NBLADS'
&      , 12X , 'NSHAPS' , 12X , ' NPTS ' / 114 , 3( 118 ) )
3800 FORMAT ( / 10X , 'DELTA3= ' , F6.2)
3810 FORMAT ( / 10X , 'UNDSLГ= ' , F6.2)
3820 FORMAT ( / 10X , 'HUBMAS= ' , F6.2)
3830 FORMAT ( / 10X , 'HUBDIS= ' , F6.2)
3840 FORMAT ( / 10X , 'BETA1 = ' , F6.2)
3850 FORMAT ( / 10X , 'BETA2 = ' , F6.2)
3860 FORMAT ( / 10X , 'NUMSCN= ' , I6 )
3870 FORMAT ( / 10X , 'TIMINC= ' , F10.5)
3880 FORMAT ( / 10X , 'MSTAT = ' , I3 )
4000 FORMAT ( / A , 26( / 4( 1PE18.7 , : ) ) )
5000 FORMAT ( / ' CKTCL(' , I2 , ' ,K,L)-' , 4( / 4( 1PE18.7 ) ) )
6000 FORMAT ( / A / 4( 1PE18.7 ) )
6050 FORMAT ( / ' STA(1)-' , 26( / 5( 1PE14.3 , : )))
6100 FORMAT ( / ' TCORLS(' , I1 , ' ,I)-' , 26( / 4( 1PE18.7 , : ) ) )
7000 FORMAT ( / ' CIFMOM(' , I1 , ' ,I)-' , 26( / 4( 1PE18.7 , : ) ) )
8000 FORMAT ( / ' Data for MODULE 2 have been written to ' , A , '.'
&      / )

```

C Open output data file.

```
OPEN ( 2 , FILE=FILOUT , STATUS='UNKNOWN' )
```

C Put titles and hub type in run data file.

```
WRITE (2,2000) TITLE1 , TITLE2 , TITLE3
```

C Put scalar values into run data file.

```
WRITE (2,3000) ALENTH , ALPHA0 , BETA0
WRITE (2,3100) BLSHNK , BLTIP , CHI
WRITE (2,3200) CSUBMA , DRGFRM , HUBHT
WRITE (2,3300) HUBRAD , OMEGA , PHIO
WRITE (2,3400) PHIAMP , PHIOMG , PSIZER
WRITE (2,3500) SHERXP , THETAP , THETAT
WRITE (2,3600) TSUB0 , TSUBP , VHUB

```

C Use the value 1 for NSHAPS, which is the preferred value for
C MODULE 2. Use the variable name NSHPS to avoid overwriting
C NSHAPS.

```
NSHPS = 1
```

```
WRITE (2,3700) KSHADW , NBLADS , NSHPS , NPTS
```

```
DELТ3 = DELТ3*180./3.14159
```

```
WRITE (2,3800) DELТ3
WRITE (2,3810) UNDSLГ
WRITE (2,3820) HUBMAS
WRITE (2,3830) HUBDIS
WRITE (2,3840) BETA1
WRITE (2,3850) BETA2

```

C Put turbulence calculation info into run data file.

```
WRITE (2,3860) NUMSCN
WRITE (2,3870) TIMINC
WRITE (2,3880) MSTAT
WRITE (2,6050) ( STA(1), I=1, MSTAT)

```

C Put vector values into run data file.

```
WRITE (2,4000) ' CLALFA-' , ( CLALFA(I) , I=1,NPTS )
WRITE (2,4000) ' CLMAX -' , ( CLMAX (I) , I=1,NPTS )
WRITE (2,4000) ' CDZERO-' , ( CDZERO(I) , I=1,NPTS )
WRITE (2,4000) ' CHORD-' , ( CHORD (I) , I=1,NPTS )
WRITE (2,4000) ' ECNTFN-' , ( ECNTFN(I) , I=1,NPTS )
WRITE (2,4000) ' ESUBAC-' , ( ESUBAC(I) , I=1,NPTS )

```

```
WRITE (2,4000) ' THETA0-' , ( THETA0(I) , I=1,NPTS )
```

C Put matrix coefficient values into run data file.

```
WRITE (2,4000) ' CKBEND-'  
& , ( ( CKBEND(ROW,COL) , COL=1,4 ) , ROW=1,4 )  
WRITE (2,4000) ' CKTOMG-'  
& , ( ( CKTOMG(ROW,COL) , COL=1,4 ) , ROW=1,4 )  
WRITE (2,4000) ' CKTGRV-'  
& , ( ( CKTGRV(ROW,COL) , COL=1,4 ) , ROW=1,4 )  
WRITE (2,4000) ' CKQLOD-'  
& , ( ( CKQLOD(ROW,COL) , COL=1,4 ) , ROW=1,4 )  
WRITE (2,4000) ' CMMASS-'  
& , ( ( CMMASS(ROW,COL) , COL=1,4 ) , ROW=1,4 )
```

```
DO 500 N=1,4
```

```
500 WRITE (2,5000) N , ( ( CKTCRL(N,ROW,COL), COL=1,4 ), ROW=1,4 )
```

C Put vector coefficient values into run data file.

```
WRITE (2,6000) ' CMRIGD-' , ( CMRIGD(I) , I=1,4 )  
WRITE (2,6000) ' CMRGD1-' , ( CMRGD1(I) , I=1,4 )  
WRITE (2,6000) ' CMBLNC-' , ( CMBLNC(I) , I=1,4 )  
WRITE (2,6000) ' CMGRAV-' , ( CMGRAV(I) , I=1,4 )  
WRITE (2,6000) ' CMGRV1-' , ( CMGRV1(I) , I=1,4 )  
WRITE (2,6000) ' CMHUB1-' , ( CMHUB1(I) , I=1,4 )  
WRITE (2,6000) ' KOSTIF-' , ( KOSTIF(I) , I=1,4 )  
WRITE (2,6000) ' K1STIF-' , ( K1STIF(I) , I=1,4 )  
WRITE (2,6000) ' K2STIF-' , ( K2STIF(I) , I=1,4 )  
WRITE (2,6000) ' DAMP -' , ( DAMP(I) , I=1,4 )
```

C Put matrix tension values into run data file.

```
DO 600 N=1,4
```

```
600 WRITE (2,6100) N , ( TCORLS(N,I) , I=1,NPTS )
```

C Put vector gravity and centrifugal stiffening values into run data file.

```
WRITE (2,4000) ' TGRAV-' , ( TGRAV(I) , I=1,NPTS )  
WRITE (2,4000) ' TOMGA-' , ( TOMGA(I) , I=1,NPTS )
```

C Put property values into run data file.

```
DO 700 N=1,4
```

```
700 WRITE (2,7000) N , ( CIFMOM(N,I) , I=1,NPTS )
```

```
WRITE (2,4000) ' DELTIM-' , ( DELTIM(I) , I=1,NPTS )  
WRITE (2,4000) ' OFFMAS-' , ( OFFMAS(I) , I=1,NPTS )
```

```
DO 710 I = 1, 201, 10
```

```
WRITE(2,*) (SHP(J,I), J=1,4)
```

```
710 CONTINUE
```

```
DO 720 I = 1, 201, 10
```

```
WRITE(2,*) (SHPDOT(J,I), J=1,4)
```

```
720 CONTINUE
```

```
DO 730 I = 1, 201, 10
```

```
WRITE(2,*) (SHPDD(J,I), J=1,4)
```

```
730 CONTINUE
```

C Run file generation complete. Print message and close file.

```
WRITE (*,8000) FILOUT(1:LNTH(FILOUT))
```

```
CLOSE ( 2 )
```

```
RETURN
```

```
END
```

```
FUNCTION SIMPSN ( LOWLIM , UPLIM , NPTS , FOFX )
```

```

C *****
C *
C * This function performs composite Simpson's integration *
C * on a given set of data points. The formulation appears *
C * in: *
C * Carnahan, et al, Applied Numerical Methods, John *
C * Wiley and Sons, NY, pp. 78-79. *
C *****

```

```

C *****
C *
C * Local and dummy variables used in this routine: *
C *
C * FOFX - Array of data points to be integrated. They *
C * are treated as the value of a function eval- *
C * uated at a specific point. *
C * H - Subinterval size. *
C * I - Generic index. *
C * LOWLIM - Lower limit of integration. *
C * UPLIM - Upper limit of integration. *
C * NPTS - Number of base points given by 2*N+1, where *
C * N is the number of applications of Simpson's *
C * rule. See NSIMP in the main program. Note: *
C * When calling from subroutine COEFFS, the *
C * value of NP is used instead of NPTS. *
C * SIMPSN - Value of the integral from LOWLIM to UPLIM. *
C *****

```

```

REAL FOFX (201)
REAL H
REAL LOWLIM
REAL SIMPSN
REAL UPLIM

```

```

INTEGER I
INTEGER NPTS

```

```

C Compute the subinterval size and initialize the integral.

```

```

H = ( UPLIM - LOWLIM )/( NPTS - 1 )
SIMPSN = 0.0

```

```

C Add in the intermediate points. In the formulation, all even
C numbered points have coefficient of 4. In this case, the index
C must be shifted to form the proper coefficient.

```

```

DO 10 I=2,NPTS-1,2
10 SIMPSN = SIMPSN + 4.0*FOFX(I) + 2.0*FOFX(I+1)

SIMPSN = SIMPSN + FOFX(1) - FOFX(NPTS)

SIMPSN = H*SIMPSN/3.0

```

```

RETURN
END
FUNCTION TRPZOD ( FOF , LOWLIM , NP , BLTIP )

```

```

C *****
C *
C * Function TRPZOD performs composite trapezoidal integra- *
C * tion on a set of data points transmitted from the *
C * calling routine. For derivation of the formula and *
C * limitations, see Carnahan, p. 78 (see full reference in *
C * comments for function SIMPSN). For computational effi- *
C * ciency, the interval width is not used in the formula- *
C * tion until the end when it is multiplied by the sum. *
C *
C *****

```

```

C *****
C *
C * Local and dummy variables used in this routine: *
C *
C * FOF - Array of data points to be integrated. They *
C * are treated by this function as the value *
C * of a function evaluated at specific points. *
C * (dummy argument) *
C * H - Subinterval length given by H=(B-A)/N. *
C * I - Index into the FOF array. *
C * LOWLIM Index of the lower integration limit. The *
C * blade position indexed by LOWLIM is given *
C * by BLTIP*(LOWLIM-1)/(NP-1). (dummy argu- *
C * ment) *
C * NP - Number of evenly spaced data points. (dummy *
C * argument) *
C * TRPZOD - Value of the integral from the blade posi- *
C * tion indexed by LOWLIM to BLTIP. *
C *
C *****

```

```

REAL      BLTIP
REAL      FOF      (201)
REAL      H
REAL      TRPZOD

INTEGER   I
INTEGER   LOWLIM
INTEGER   NP

```

```

C Check to see if the lower and upper limits of integration are
C the same. If so, the integral is zero.

```

```

IF ( LOWLIM .EQ. NP ) THEN

```

```

    TRPZOD = 0.0
    RETURN

```

```

END IF

```

```

C Compute the distance between data points.

```

```

H = BLTIP/( NP - 1 )

```

```

C Initialize integral to the contribution of the end points.

```

```

TRPZOD = 0.5*( FOF(LOWLIM) + FOF(NP) )

```

```

C If there are only two data points, then we are done.

```

```

IF ( LOWLIM+1 .EQ. NP ) GO TO 20

```

```

C Add in the contribution of the intermediate points.

```

```

DO 10 I=LOWLIM+1,NP-1
10 TRPZOD = TRPZOD + FOF(I)

```

```

C Multiply by the interval width and return.

```

```

20 TRPZOD = H*TRPZOD

```

```

RETURN
END
SUBROUTINE MULT ( AMATRX , BMATRX, RESULT, M , N )

```

```

C *****
C *

```

```

C      * Subroutine MULT premultiplies two incoming matrices      *
C      * together. The result is stored in the RESULT matrix.    *
C      * All matrices are considered to be square matrices up to *
C      * up to order 4. The matrix AMATRX gets multiplied        *
C      * by the matrix BMATRX.                                    *
C      *****

```

```

C      *****
C      *
C      * External references in this routine:
C      *
C      * none
C      *
C      *****

```

```

C      *****
C      *
C      * Named COMMON blocks used in this routine:
C      *
C      * NONE
C      *
C      *****

```

```

C      *****
C      *
C      * Local and dummy variables used in this routine:
C      *
C      * AMATRX - The incoming matrix to be premultiplied by
C      *           the inverse CMMASS matrix.
C      * BMATRX - The other incoming matrix which does the
C      *           multiplying.
C      * RESULT - The result matrix of the multiplications
C      * I       - Generic index.
C      * J       - Generic index.
C      * K       - Generic index.
C      * M       - Number of rows in the incoming matrix.
C      * N       - Number of columns in the incoming matrix.
C      *
C      *****

```

```

REAL      AMATRX (8,8)
REAL      BMATRX(8,8)
REAL      RESULT(8,8)

```

```

INTEGER   I
INTEGER   J
INTEGER   K
INTEGER   M
INTEGER   N

```

```

C      Multiply AMATRX by BMATRX putting the result into RESULT.

```

```

DO 30 I=1,M
  DO 20 J=1,N
    RESULT(I,J) = 0.
    DO 10 K=1,M
      10  RESULT(I,J) = RESULT(I,J) + BMATRX(I,K) * AMATRX(K,J)
    20  CONTINUE
  30  CONTINUE

RETURN
END

```


Appendix D
Module 2 Listing

```

*****
*
*           DISCLAIMER
*
*   Neither the United States Department of Energy, the
*   National Renewable Energy Laboratory,
*   the Midwest Research Institute nor anyone else
*   who has been involved in the
*   creation, production or delivery of this program shall
*   be liable for any direct, indirect, consequential, or
*   incidental damages arising out of the use, the results
*   of use, or inability to use this program even if the
*   United States Department of Energy has been advised of
*   the possibility of such damages or claim. Some states
*   do not allow the exclusion or limitation of liability
*   for consequential or incidental damages, so the above
*   limitation may not apply to you.
*
*****

```

```

*****
*
*           The STRAP Code
*
*   STRAP calculates the forces and moments on a teetering
*   hub rotor due to aerodynamic, inertial and gravita-
*   tional forces. The aerodynamic effects include wind
*   shear, tower shadow and the induced velocity due to the
*   change in momentum of the air stream as it passes over
*   the blade. In addition, the code can calculate the rotor
*   response and loads to turbulent wind inputs.
*
*****

```

```

*****
*
*           Module 2
*
*   This program is the second of two modules that consti-
*   tute the STRAP code. This second module performs the
*   actual model run, computes the loads and prints the re-
*   sults to an external data file.
*
*****

```

```

*****
*
*   Questions about the original formulation, theory, meth-
*   od of solution and programming should be addressed to:
*
*       Alan D. Wright
*       Wind Research Branch
*       National Renewable Energy Laboratory
*       1617 Cole Blvd.
*       Golden, CO 80401
*       (303)231-7651
*
*****

```

```

*****
*
*   I/O Conventions:
*
*       Unit 2 - Diagnostics file
*       Unit 3 - Input run data
*       Unit 4 - Results file
*

```

```

C      *      Unit * - Monitor or keyboard *
C      *
C      *****
C
C      *****
C      *
C      *      External references in this routine:
C      *
C      *      DATAIN - Reads in a data generated by MODULE 1.
C      *      DIAG   - Performs a test run of the aerodynamic rou-
C      *                  tines.
C      *      RUN    - Models the rotor blade motion.
C      *
C      *****
C
C      *****
C      *
C      *      Local and dummy variables used in this routine:
C      *
C      *      ANS    - Used to store input responses from key-
C      *                  board.
C      *      HAVDAT - Flag that indicates that some data has been
C      *                  read in.
C      *      HAVRUN - Flag that indicates that the model has been
C      *                  run. Used for diagnostic runs.
C      *      NEWSSET - Flag that indicates that a new data set has
C      *                  been read in.
C      *      NPTS   - Number of points along the blade used to
C      *                  perform Simpson's integration for calcu-
C      *                  lating the moments and forces at the blade
C      *                  root.
C      *      NSIMP  - Order of the composite Simpson's integra-
C      *                  tion used in the run. Parameter is set to
C      *                  10 in order to make 21 blade property sta-
C      *                  tions.
C      *      TODEGS - Logical variable that indicates the direc-
C      *                  tion of conversion between degrees and ra-
C      *                  dians.
C      *
C      *****

```

```

INTEGER      NPTS
INTEGER      NSIMP

LOGICAL      HAVDAT
LOGICAL      HAVRUN
LOGICAL      NEWSSET
LOGICAL      TODEGS

CHARACTER*1  ANS

INCLUDE      'C:INCLUDE\TRBINF.INC'

DATA        HAVDAT / .FALSE. /
DATA        HAVRUN / .FALSE. /
DATA        NSIMP  / 10 /
DATA        TODEGS / .TRUE. /

```

```

100 FORMAT (
& // ' Program For Analysis Of Horizontal Axis Wind Turbine'
& // ' Response To Dynamic Loads'
& // ' MODULE 2' )
110 FORMAT ( // ' Operations Menu'
& // ' -----'
& // ' (R)ead in a data file'
& // ' (S)et up and run the model'
& // ' (D)agnostic run'
& // ' (T)urbulence run'
& // ' (Q)uit' )
120 FORMAT ( // ' Enter Option (R,S,D,T,Q) > ' )
130 FORMAT ( A )
140 FORMAT ( // ' You must select option (R) at least once before'
& // ' invoking this option.' )
150 FORMAT ( // ' You must select option (S) at least once before'

```

```

&      / ' invoking this option.' )
160 FORMAT ( / ' Invalid response. Please try again.' )
170 FORMAT ( / ' STRAP terminated normally.' / )

```

C Calculate the number of blade property stations.

```
NPTS = 2*NSIMP + 1
```

C Print title.

```
PRINT 100
```

C Print menu of options. Ask for choice.

```
10 PRINT 110
```

```
20 PRINT 120
READ 130, ANS
PRINT *, ANS
```

C Which option was chosen?

```
IF ( ( ANS .EQ. 'R' ) .OR. ( ANS .EQ. 'r' ) ) THEN
```

C Read in a data file. Convert limits values back to degrees
C if this isn't the first time we've read in data.

```
IF ( HAVDAT ) CALL CONVRT ( TODEGS )
CALL DATAIN
HAVDAT = .TRUE.
NEWSET = .TRUE.
GO TO 10
```

```
ELSE IF ( ( ANS .EQ. 'S' ) .OR. ( ANS .EQ. 's' ) ) THEN
```

C Set up and run the model. Option (R) must have been
C previously selected.

```
IF ( HAVDAT ) THEN
CALL RUN ( NPTS , NEWSET , HAVRUN )
GO TO 10
ELSE
PRINT 140
GO TO 20
END IF
```

```
ELSE IF ( ( ANS .EQ. 'D' ) .OR. ( ANS .EQ. 'd' ) ) THEN
```

C Run diagnostics. Option (S) must have been previously
C selected.

```
IF ( HAVRUN ) THEN
CALL DIAG ( NPTS )
GO TO 10
ELSE
PRINT 150
GO TO 20
END IF
```

```
ELSE IF ( ( ANS .EQ. 'T' ) .OR. ( ANS .EQ. 't' ) ) THEN
C Run turbulence case. Option S must have been previously
C invoked.
```

```
IF ( HAVRUN ) THEN
ITURB = 1
CALL TRBCLC( NPTS)
GO TO 10
ELSE
PRINT 150
GO TO 20
ENDIF
```

```
ELSE IF ( ( ANS .NE. 'Q' ) .AND. ( ANS .NE. 'q' ) ) THEN
```

```
C      Invalid response.
```

```
      PRINT 160  
      GO TO 20
```

```
END IF
```

```
C      Processing complete.
```

```
PRINT 170
```

```
STOP  
END  
BLOCK DATA
```

```
C      *****  
C      *  
C      *   This module is used to initialize the COMMON blocks.   *  
C      *  
C      *****
```

```
C      *****  
C      *  
C      *   Named COMMON blocks used in this routine:   *  
C      *  
C      *   AER01 - Holds coefficients related to aerodynamic *  
C      *           loads calculations such as ClAlpha, CdZero, *  
C      *           etc. *  
C      *   AIRFRC - Holds values used in aerodynamic calcula- *  
C      *           tions. *  
C      *   BLADE - Holds blade property values such as stiff- *  
C      *           ness and mass distributions. *  
C      *   CONST - Turbine and other constants used in load *  
C      *           calculations. *  
C      *   DELTV - Holds turbine inputs for possible future *  
C      *           use. Not currently used. *  
C      *   FORMS - Holds blade deflections. *  
C      *   INV - Holds the inverse of the mass matrix. *  
C      *   LIMITC - Holds values used in the LIMITS routine. *  
C      *   LITERL - Holds data set titles. *  
C      *   LODVAL - Holds values used to compute blade loads. *  
C      *   MATRX1 - Holds stiffness coefficient matrices. *  
C      *   MATRX2 - Holds some of the coefficient matrices of *  
C      *           the governing equation of motion. *  
C      *   POSITN - Holds parameters related to blade position *  
C      *           such as PHI, PSI, etc. *  
C      *   SARAYS - Holds new and old values for the general- *  
C      *           ized coordinates. *  
C      *   SHAPE - Holds blade coordinate shape functions. *  
C      *   START - Holds initial blade deflection. *  
C      *   TENSIN - Holds tension components of the stiffness *  
C      *           functions. *  
C      *   TURBN - Holds turbine parameters such as number of *  
C      *           blades, rotor speed, etc. *  
C      *   VINDUC - Holds induced velocity components. *  
C      *   VREL1 - Holds blade section velocity components. *  
C      *   WIND - Holds wind shear and tower shadow parame- *  
C      *           ters. *  
C      *   WNDVEL - Holds values used in wind shear and tower *  
C      *           shadow computations. *  
C      *  
C      *****
```

```
C      *****  
C      *  
C      *   COMMON variables used in Module 2 of STRAP:   *  
C      *  
C      *   ABAR - Dimensionless form of ALENTN ( ALENTN di- *  
C      *           vided by the rotor radius, RR ). *  
C      *   AINVRS - The inverse CMMASS matrix. *  
C      *   ALENTN - Distance from the tower axis to the rotor *  
C      *  
C      *****
```

```

C      *      hub. (feet)      *
C      *      ALPHA0 - The angle from the zero-lift line to the      *
C      *      section-chord line. (Generally negative)      *
C      *      BEGIN2 - Azimuth position corresponding to the be-      *
C      *      ginning of print region 2. (degrees)      *
C      *      BETA0 - Blade coning angle. The value is input in      *
C      *      degrees and converted to radians. (radians)      *
C      *      (input data)      *
C      *      BLSHNK - Length of blade shank measured from the      *
C      *      blade root to the start of the airfoil sec-      *
C      *      tion. If airfoil begins at the root, then      *
C      *      BLSHNK=0. (feet)      *
C      *      BLTIP - Blade length measured from the blade root      *
C      *      to the blade tip. Note - rotor radius is      *
C      *      HUBRAD + BLTIP. (feet)      *
C      *      CDZERO - Drag coefficient values at equidistant      *
C      *      points along the blade. Derived from      *
C      *      ACDZER values. (dimensionless)      *
C      *      CHI - Rotor pitch angle. Input in degrees and      *
C      *      converted to radians. (radians) (input      *
C      *      data)      *
C      *      CHORD - Blade chord at equidistant points along the      *
C      *      blade. Derived from ACHORD. (feet)      *
C      *      CIFMOM - Integral of DIFMOM.      *
C      *      CKBEND - Bending stiffness matrix.      *
C      *      CKQLOD - Inertial moment stiffening matrix.      *
C      *      CKTCRL - Coriolis stiffening matrix.      *
C      *      CKTGRV - Gravity stiffening matrix.      *
C      *      CKTOMG - Centrifugal stiffening matrix.      *
C      *      CLALFA - Slope of the lift curve at equidistant      *
C      *      points along the blade. Derived from ACLALF      *
C      *      values. (radians^1)      *
C      *      CLMAX - The maximum or stall value of the lift co-      *
C      *      efficient. Derived from ACLMAX values.      *
C      *      CMBLNC - Coefficient associated with the blade mass      *
C      *      imbalance (OFFSET).      *
C      *      CMGRAV - Mass coefficients associated with gravita-      *
C      *      tional loads.      *
C      *      CMGRV1 - Submatrix of CMGRAV containing odd terms      *
C      *      only. Used only with teetering hubs.      *
C      *      CMMASS - Blade mass matrix. It is also-used in the      *
C      *      inertia force stiffening term.      *
C      *      CMRGD1 - Submatrix of CMRIGD containing odd terms      *
C      *      only. Used only with teetering hubs.      *
C      *      CMRIGD - Mass coefficients associated with rigid bo-      *
C      *      dy motion.      *
C      *      CSUBMA - Pitching moment coefficient.      *
C      *      CTHP - Cosine of the pitch angle, THETAP.      *
C      *      DAETA - Differential aerodynamic forces on a blade      *
C      *      section in the chordwise direction.      *
C      *      DAZETA - Differential aerodynamic forces on a blade      *
C      *      section in the flapwise direction.      *
C      *      DELPSI - Delta-Psi. The amount Psi will change from      *
C      *      this step to the next. Used in Euler pre-      *
C      *      dictor/corrector routine.      *
C      *      DELTAT - The incremental time change from one step      *
C      *      to the next in the yaw solution (ITRIM=0).      *
C      *      DELTIM - Integral of DDELT I.      *
C      *      DELTVX - Turbulent wind velocity fluctuations. Not      *
C      *      currently in use. (feet/second)      *
C      *      DELTVY - Turbulent wind velocity fluctuations. Not      *
C      *      currently in use. (feet/second)      *
C      *      DELTVZ - Turbulent wind velocity fluctuations. Not      *
C      *      currently in use. (feet/second)      *
C      *      DRGFRM - Drag coefficient form constant.      *
C      *      DVIND - The delta-V-induced velocity components.      *
C      *      Components are of order epsilon.      *
C      *      ECNTFN - Complicated term.      *
C      *      EIAREA - Moment of inertia values at equidistant      *
C      *      points along the blade. Derived from      *
C      *      AEIARE data values. (Lb-Ft**2)      *
C      *      END2 - Azimuth position corresponding to the end      *
C      *      of print region 2. (degrees)      *
C      *      ERROR - Difference between corrected and predicted      *
C      *      value of the blade tip displacements in the      *
C      *      Euler predictor/corrector routine.      *
C      *      ESUBAC - Distance from the blade elastic axis to the      *
C      *      aerodynamic center. Positive towards the

```

```

C      *      leading edge.  Derived from AESBAC values.  *
C      *      (feet)  *
C      *      EUERR - Convergence criterion for computing blade  *
C      *      tip deflections via the Euler predictor/  *
C      *      corrector routine. (percent)  *
C      *      FAERO - Aerodynamic force on the rotor.  *
C      *      GRAV - Constant of gravitational acceleration as  *
C      *      measured at sea level. (feet/second**2)  *
C      *      HUBHT - Hub reference height above the ground.  *
C      *      (feet)  *
C      *      HUBRAD - Radius of rotor hub. (feet) (input data)  *
C      *      HUBVEL - Wind velocity at rotor hub. Currently  *
C      *      equal to Vhub. (feet/second)  *
C      *      IEMASS - Edgewise mass moment of inertia at equidis-  *
C      *      tant points along the blade section. Der-  *
C      *      ived from AIEMAS values. (10^6 Lb-sec^2)  *
C      *      IFMASS - Flapwise mass moment of inertia at equidis-  *
C      *      tant points along the blade section. Der-  *
C      *      ived from AIFMAS values. (10^6 Lb-sec^2)  *
C      *      ITRIM - Flag indicating run status:  *
C      *      0 - Run to find yawing solution.  *
C      *      1 - Run to find trim solution.  *
C      *      2 - Trim startup run.  *
C      *      KSHADW - Number of tower shadow peaks within the  *
C      *      tower shadow zone. (input data)  *
C      *      MASS - Blade mass per unit length at equidistant  *
C      *      points along the blade. Input in Lbf/ft,  *
C      *      via the input variable WEIGHT, and convert-  *
C      *      ed to slugs/ft. (slugs/ft) (input data)  *
C      *      NBLADS - Number of turbine blades. (input data)  *
C      *      NSHAPS - Number of blade shape functions, 4 maximum.  *
C      *      (input data)  *
C      *      NYAW - Number of rotor revolutions to be run for  *
C      *      a yawing solution.  *
C      *      OFFMAS - Integral of DOFFMS.  *
C      *      OFFSET - Distance from the elastic axis to the mass  *
C      *      axes of blade section. Positive towards  *
C      *      the leading edge. Derived from AOFFST val-  *
C      *      ues. Called E-sub-eta in the formulation.  *
C      *      (feet) (input data)  *
C      *      OMEGA - Rotor speed. Input in RPM and converted to  *
C      *      radians/second. (rad/sec) (input data)  *
C      *      PHI - Rotor yaw error angle. Angle between the  *
C      *      hub axis and the mean wind. Subscript indi-  *
C      *      cates which time derivative.  *
C      *      PHIO - Rotor yaw reference angle. Measured as the  *
C      *      angle between a horizontal reference line  *
C      *      passing through the tower axis and the hor-  *
C      *      izontal projection of the line between the  *
C      *      tower axis and the rotor hub. Input as de-  *
C      *      grees and converted to radians. (radians)  *
C      *      (input data)  *
C      *      PHIAMP - Amplitude of periodic yaw motion about yaw  *
C      *      reference angle. Input as degrees and con-  *
C      *      verted to radians. (radians) (input data)  *
C      *      PHIOMG - Maximum yaw rate. Used internally to com-  *
C      *      pute the yaw period. It can be used to in-  *
C      *      put a turbine steady yaw rate. Input as  *
C      *      degrees/sec and converted to radians/sec-  *
C      *      ond. (radians/sec) (input data)  *
C      *      PI - The constant 3.141592653589793...  *
C      *      PRINT1 - Printout interval in print region 1.  *
C      *      PRINT2 - Printout interval in print region 2.  *
C      *      PSI - Blade azimuth angle. The zeroth element is  *
C      *      the past value of Psi and the first element  *
C      *      contains the present value. A value of ze-  *
C      *      ro indicates the blade is straight up.  *
C      *      PSISHD - Azimuth position of the center line of the  *
C      *      tower shadow. When it is 180 degrees, the  *
C      *      tower shadow center line is oriented verti-  *
C      *      cally downward from the hub.  *
C      *      PSIZER - Half angle width of the tower shadow re-  *
C      *      gion. (degrees) (input data)  *
C      *      RAD2DG - Radians to degrees conversion factor.  *
C      *      RHOAIR - Air density. Currently set to sea level.  *
C      *      (Lb-sec^2/feet^4 or slugs/feet^3)  *
C      *      RR - Rotor radius (BLTIP+HUBRAD). (feet)  *
C      *      RROMGA - A constant used throughout the STRAP code.  *

```

```

C      *      (RR*OMEGA*PI/30)      *
C      *      SO      - Initial static blade deflection at the tip.      *
C      *      It is set in subroutine STARTUP.      *
C      *      SHAPE - Array containing shape functions evaluated      *
C      *      at regular intervals. The second subscript      *
C      *      corresponds to Nth spatial derivative.      *
C      *      SHERXP - Wind shear power exponent. (input data)      *
C      *      SNEW  - An array of tip displacement function val-      *
C      *      ues for each of the STEPMX stations around      *
C      *      the disk. These are the results of the      *
C      *      current computation. The zeroth element      *
C      *      has the same value as the 360th element of      *
C      *      SOLD.      *
C      *      SOLD  - An array of tip displacement function val-      *
C      *      ues for each of the STEPMX stations around      *
C      *      the disk. These are the results of the      *
C      *      previous computation. The 360th element      *
C      *      has the same value as the zeroth element of      *
C      *      SNEW.      *
C      *      STEPMN - Minimum allowable azimuth angle step size.      *
C      *      (degrees)      *
C      *      STEPMX - Maximum allowable azimuth angle step size.      *
C      *      (degrees)      *
C      *      STHP  - Sine of the pitch angle THETAP.      *
C      *      TCORLS - Blade tension coefficient due to coriolis      *
C      *      effects.      *
C      *      TGRAV  - Blade tension coefficient due to gravity      *
C      *      effects.      *
C      *      THETAO - Orientation of the zero lift line with re-      *
C      *      spect to the blade principal bending axis.      *
C      *      THETAP - Blade pitch angle. The angle from the flap      *
C      *      bending axis (assumed to be the chord line)      *
C      *      at the 3/4 spanwise location. The assump-      *
C      *      tion has been made that the bending axis      *
C      *      does not vary with spanwise location. In      *
C      *      effect, THETAP sets the blade pitch angle      *
C      *      and the direction of the flapping displac-      *
C      *      ements. Input in degrees and converted to      *
C      *      radians. (radians) (input data)      *
C      *      THETAT - The built-in blade twist angle from the      *
C      *      section-chord line at the reference station      *
C      *      to the section-chord line at the tip of the      *
C      *      rotor. Positive towards feather.      *
C      *      TIME  - Total elapsed time from the beginning of      *
C      *      the yaw solution run. The zeroth element      *
C      *      holds the past value of time, and the first      *
C      *      element holds the current value. (seconds)      *
C      *      TITLE1 - First line of the data file title.      *
C      *      TITLE2 - Second line of the data file title.      *
C      *      TITLE3 - Third line of the data file title.      *
C      *      TOMGA  - Blade tension coefficient associated with      *
C      *      centrifugal force effects.      *
C      *      TRACEF - Flag that causes subroutine TRACE to print      *
C      *      the values of various variables around the      *
C      *      rotor disk.      *
C      *      TRMERR - Convergence criterion for computing the      *
C      *      trim solution. (percent)      *
C      *      TSHADW The tower shadow velocity deficit.      *
C      *      TSUBO  - Tower shadow wind speed offset component.      *
C      *      TSUBP  - Tower shadow sinusoidal component.      *
C      *      VETA   - Relative fluid velocity over the blade in      *
C      *      the edgewise direction.      *
C      *      VHUB  - Air speed at the height of the hub.      *
C      *      (ft/sec) (input data)      *
C      *      VINDO  - An induced velocity rotor term. It is uni-      *
C      *      form over the rotor disk.      *
C      *      VINDC  - An induced velocity component associated      *
C      *      with the Cosine(Psi) term.      *
C      *      VINDR  - Inflow minus uniform induced velocity.      *
C      *      VV     - The actual tip displacements computed from      *
C      *      the relation in subroutine FORM1:      *
C      *      V = Sum of S(I)*GAMMA(I)      *
C      *      and      *
C      *      VDOT = Sum of SDOT(I)*GAMMA(I),      *
C      *      where V is a function of position along the      *
C      *      blade and time, S is a function of time and      *
C      *      GAMMA is the blade coordinate shape func-      *
C      *      tion. The index I denotes summation over      *

```



```

C      *           the coordinate function shapes, DOT is the *
C      *           first derivative w.r.t. time. Thus:      *
C      *            $VV(0,I) = V$  = displacement          *
C      *           and  $VV(1,I) = V-DOT$  = velocity.        *
C      *           VWIND - Resultant wind velocity at the rotor disk. *
C      *           VZETA - Relative fluid velocity over the blade in *
C      *           the flapwise direction.                    *
C      *           WSHEAR - The harmonic or unsteady component of the *
C      *           wind shear.                                *
C      *           WSHR - Individual harmonic components of the total *
C      *           unsteady wind shear component.            *
C      *           WW - Resultant windspeed at a blade element. *
C      *
C      *****

```

```

INCLUDE      'C:\INCLUDE\AERO1.INC'
INCLUDE      'C:\INCLUDE\AIRFRC.INC'
INCLUDE      'C:\INCLUDE\BLADE2.INC'
INCLUDE      'C:\INCLUDE\CONST2.INC'
INCLUDE      'C:\INCLUDE\DELTV.INC'
INCLUDE      'C:\INCLUDE\FORMS.INC'
INCLUDE      'C:\INCLUDE\INV.INC'
INCLUDE      'C:\INCLUDE\LIMITC.INC'
INCLUDE      'C:\INCLUDE\LITERL.INC'
INCLUDE      'C:\INCLUDE\LODVAL.INC'
INCLUDE      'C:\INCLUDE\MATRX1.INC'
INCLUDE      'C:\INCLUDE\MATRX2.INC'
INCLUDE      'C:\INCLUDE\POSITN.INC'
INCLUDE      'C:\INCLUDE\SARAYS.INC'
INCLUDE      'C:\INCLUDE\SHAPE.INC'
INCLUDE      'C:\INCLUDE\START.INC'
INCLUDE      'C:\INCLUDE\TENSIN2.INC'
INCLUDE      'C:\INCLUDE\TURBN.INC'
INCLUDE      'C:\INCLUDE\VINDUC.INC'
INCLUDE      'C:\INCLUDE\VREL1.INC'
INCLUDE      'C:\INCLUDE\WIND2.INC'
INCLUDE      'C:\INCLUDE\WNDVEL.INC'

```

C Initialize COMMON blocks.

```

DATA      BEGIN2 / 180.0 /
DATA      DELPSI / 2* 0.0 /
DATA      DELTAT / 2* 0.0 /
DATA      DELTVX / 21* 0.0 /
DATA      DELTVY / 21* 0.0 /
DATA      DELTVZ / 21* 0.0 /
DATA      END2 / 270.0 /
DATA      EUERR / 10.0 /
DATA      GRAV / 32.1740 /
DATA      KSHADW / 1 /
DATA      NSHAPS / 1 /
DATA      NYAW / 0 /
DATA      PHI / 3* 0.0 /
DATA      PHI0 / 0.0 /
DATA      PI / 3.141592653589793 /
DATA      PRINT1 / 10.0 /

```

```

DATA      PRINT2 /      10.0      /
DATA      PSI    / 2*   0.0      /
DATA      PSISHD /      180.0     /
DATA      PSIZER /      15.0     /
DATA      RAD2DG /      57.29577951308233 /
DATA      RHOAIR /      0.002    /
DATA      SHERXP /      0.14286  /
DATA      STEPMN /      0.001    /
DATA      STEPMX /      10.0     /
DATA      TIME   / 2*   0.0      /
DATA      TRACEF / .FALSE.     /
DATA      TRMERR /      10.0     /
DATA      TSUBO  /      10.0     /
DATA      TSUBP  /      10.0     /
DATA      VHUB   /      27.1     /

```

END

SUBROUTINE ACCEL (DEFLTN , VELCTY , ACCELN , NPTS , STEMP)

```

C      *****
C      *
C      * Subroutine ACCEL computes the solution to the blade
C      * equation of motion. The coefficient matrices are
C      * available for all four coordinate shape functions, but
C      * the solution is only calculated for the number of func-
C      * tions specified by the value of NSHAPS.
C      *
C      *****
C
C      *****
C      *
C      * External references in this routine:
C      *
C      * AFORCE - Calculates the value of the aerodynamic
C      * force integral.
C      * FORM1  - Calculates the values of the blade dis-
C      * placement function.
C      *
C      *****
C
C      *****
C      *
C      * Named COMMON blocks used in this routine:
C      *
C      * AIRFRC - Holds values used in aerodynamic calcula-
C      * tions.
C      * CONST  - Turbine and other constants used in load
C      * calculations.
C      * MATRX1 - Holds stiffness coefficient matrices.
C      * MATRX2 - Holds some of the coefficient matrices of
C      * the governing equation of motion.
C      * POSITN - Holds parameters related to blade position
C      * such as PHI, PSI, etc.
C      * TENSIN - Holds tension components of the stiffness
C      * functions.
C      * TURBN  - Holds turbine parameters such as number of
C      * blades, rotor speed, etc.
C      *
C      *****
C
C      *****
C      *
C      * Local and dummy variables used in this routine:
C      *
C      * AA    - 2.0*Omega*Sine( ThetaP ).
C      * ACCELN - The solution to the equation of motion. It
C      * corresponds to the S-double dot value in
C      * the formulation, and is the current solu-
C      * tion for the blade tip acceleration.
C      * (feet/second^2)
C      * BB    - Gravity times cosine of the blade azimuth.
C      * BLDANG - Blade azimuth position used in the teeter-
C      * ing rotor .
C      * C1    - Beta0*Omega^2.

```

```

C      *      C2      - Omega*Phi*Cosine( Psi ).      *
C      *      C3      - dPhi/dt*Sine( Psi ).      *
C      *      C4      - -Chi*Cosine( ThetaP ).      *
C      *      C5      - Sine( ThetaP )*Sine( Psi ).      *
C      *      C6      - Beta0*Cosine( ThetaP )*Cosine( Psi ).      *
C      *      C7      - Omega^2.      *
C      *      CC      - QUANT1^2.      *
C      *      CPSI    - Cosine of the blade angle.      *
C      *      DD      - Complicated term.      *
C      *      DEFLTN  - The current value of the tip deflection,      *
C      *      corresponding to the S variable in the for-      *
C      *      mulation of the equation of motion.      *
C      *      EE      - QUANT1^2.      *
C      *      FAER01  - Aerodynamic force on blade #1 of teetering      *
C      *      rotor.      *
C      *      FAER02  - Aerodynamic force on blade #2 of teetering      *
C      *      rotor.      *
C      *      FF      - Omega*QUANT1*Cosine( ThetaP )      *
C      *      GG      - Complicated term.      *
C      *      K        - Generic DO index.      *
C      *      M        - Generic DO index.      *
C      *      N        - Generic DO index.      *
C      *      NPTS    - Number of points along the blade used to      *
C      *      perform Simpson's integration for calculat-      *
C      *      ing the moments and forces at the blade      *
C      *      root. (passed from STRAP1)      *
C      *      P1      - Array used in computing acceleration.      *
C      *      P2      - Array used in computing acceleration.      *
C      *      P3      - Array used in computing acceleration.      *
C      *      Q        - Matrix used in computing acceleration.      *
C      *      QUANT1  - Omega*Sine( ThetaP ).      *
C      *      SPSI    - Sine of the blade angle.      *
C      *      STEMP   - Temporary array holding the tip displace-      *
C      *      ment values. The third dimension repre-      *
C      *      sents the order of the time derivative.      *
C      *      VELCTY  - The tip velocity corresponding to the S-dot      *
C      *      value in the formulation of the equation of      *
C      *      motion.      *
C      *      *****

```

```

REAL      AA
REAL      ACCELN (4)
REAL      BB
REAL      BLDANG
REAL      C1
REAL      C2
REAL      C3
REAL      C4
REAL      C5
REAL      C6
REAL      C7
REAL      CC
REAL      CPSI
REAL      DEFLTN (4)
REAL      EE
REAL      FAER01 (4)
REAL      FAER02 (4)
REAL      FF
REAL      P1      (4)
REAL      P2      (4)
REAL      P3      (4)
REAL      Q        (4,4)
REAL      QUANT1
REAL      SPSI
REAL      STEMP (4,0:1,0:2)
REAL      VELCTY (4)
REAL      THETAC

```

```

INTEGER   K
INTEGER   M
INTEGER   N
INTEGER   NPTS

```

```

INCLUDE   'C:\INCLUDE\AIRFRC.INC'

```

```

INCLUDE   'C:\INCLUDE\BLADE2.INC'

```

```

INCLUDE      'C:INCLUDE\CONST2.INC'
INCLUDE      'C:INCLUDE\MATRX1.INC'
INCLUDE      'C:INCLUDE\MATRX2.INC'
INCLUDE      'C:INCLUDE\POSITN.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\TENSIN2.INC'
INCLUDE      'C:INCLUDE\FORMS.INC'
INCLUDE      'C:INCLUDE\HUBPRP2.INC'
INCLUDE      'C:INCLUDE\SPRING2.INC'

C           Calculate the aero forces. The aero subroutines need to be
C           called twice, once for each blade. The first call is for
C           blade #2 at a blade angle of Psi+180 degrees and the second
C           call is for blade #1 at a blade angle of Psi.

BLDANG = PSI(1) + PI

C           Calculate aerodynamic loads on blade #2.

CALL FORM1 ( STEMP , NPTS , NSHAPS , BLDANG )

THETAC = THETAP + BETA*SDEL3
STHC = SIN(THETAC)
CTHC = COS(THETAC)
TTHC = TAN(THETAC)

CALL AFORCE ( NPTS , BLDANG )

DO 70 M=1,NSHAPS
70 FAERO2(M) = FAERO(M)
C PRINT *, 'FAERO2 = ', FAERO2(1)

BLDANG = PSI(1)

C           Calculate aerodynamic loads on blade #1.

CALL FORM1 ( STEMP , NPTS , NSHAPS , BLDANG )

THETAC = THETAP + BETA*SDEL3
STHC = SIN(THETAC)
CTHC = COS(THETAC)
TTHC = TAN(THETAC)

CALL AFORCE ( NPTS , BLDANG )

C           Sum the aerodynamic loads for both blades. For the
C           asymmetric modes (M=1,3), the difference in the aero-
C           dynamic loads is actually found.

DO 80 M=1,NSHAPS

FAERO1(M) = FAERO(M)
FAERO (M) = 0.5*( FAERO1(M) + FAERO2(M)*(-1)**M )

80 CONTINUE

C           Calculate constant coefficients.

CTHP = COS(THETAP)
STHP = SIN(THETAP)
SPSI  = SIN( BLDANG )
CPSI  = COS( BLDANG )
QUANT1 = OMEGA*STHP

```

```

AA      = 2.0*QUANT1
BB      = GRAV*(CPSI+SPSI*TDEL3)
CC      = QUANT1*QUANT1
EE      = CC
FF      = OMEGA*QUANT1*CTHP
C1      = OMEGA*OMEGA*BETA0
C2      = OMEGA*PHI(1)*CPSI
C3      = PHI(2)*SPSI
C4      = -CHI*CTHP
C5      = STHP*SPSI
C6      = BETA0*CTHP*CPSI
C7      = OMEGA*OMEGA
C9      = CPSI*CDEL3+SPSI*SDEL3
C10     = BETA0*(CPSI + TDEL3*SPSI)
DEFL1   = BETA1*(BLTIP+HUBRAD)
DEFL2   = BETA2*(BLTIP+HUBRAD)

```

C Compute P1 (see formulation notes).

```

      DO 85 M=2,NSHAPS
          P1(M) = FAERO(M) + FF*CMBLNC(M)
&          - CTHP*( C1*CMRIGD(M) + 2*C2*CMRGD1(M)
&              + C3*CMRGD1(M) )
&          + GRAV*( C4*CMGRAV(M) + C5*CMGRV1(M)
&              + C6*CMGRV1(M))
85      CONTINUE

      P1(1) = FAERO(1)*CTHP -(2.*C2 + C3)*CMRGD1(1)
&          + GRAV*C10*CMGRV1(1) - C9*GRAV*CMHUB1(1)

```

C Compute P2 - S(M) product (see formulation notes).

```

      DO 20 M = 2, NSHAPS
          P2(M) = EE*DEFLT(M)
20      CONTINUE

      IF( DEFLT(1) .LE. 0.) THEN

          IF(ABS(DEFLT(1)) .LT. DEFL1) THEN
              P2(1) = -1.*KOSTIF(1)*DEFLT(1)

          ELSEIF(ABS(DEFLT(1)) .LT. DEFL2) THEN
              P2(1) = -1.*(KOSTIF(1)+K1STIF(1))*DEFLT(1)
&              -K1STIF(1)*DEFL1 - 1.*DAMP(1)*VELCTY(1)

          ELSE
              P2(1) = -1.*(KOSTIF(1)+K2STIF(1))*DEFLT(1)
&              -K2STIF(1)*DEFL2 - 1.*DAMP(1)*VELCTY(1)
          ENDIF

      ELSE

          IF(DEFLT(1) .LT. DEFL1) THEN
              P2(1) = -1.*KOSTIF(1)*DEFLT(1)

          ELSEIF(DEFLT(1) .LT. DEFL2) THEN
              P2(1) = -1.*(KOSTIF(1)+K1STIF(1))*DEFLT(1)
&              +K1STIF(1)*DEFL1 - 1.*DAMP(1)*VELCTY(1)

          ELSE
              P2(1) = -1.*(KOSTIF(1)+K2STIF(1))*DEFLT(1)
&              +K2STIF(1)*DEFL2 - 1.*DAMP(1)*VELCTY(1)

          ENDIF
      ENDIF

```

C Compute Q(M,K) - S(K) products (see formulation notes).

```

      DO 50 M=1,NSHAPS
          P3(M) = 0.0
      DO 40 K=1,NSHAPS

```

```

&      Q(M,K) = CC*CKQLOD(M,K) - CKBEND(M,K)
      - C7*CKTOMG(M,K) + BB*CKTGRV(M,K)

DO 30 N=1,NSHAPS
30      Q(M,K) = Q(M,K) - AA*VELCTY(N)*CKTCRL(N,M,K)

      P3(M) = P3(M) + Q(M,K)*DEFLT(N,K)

40      CONTINUE

50      CONTINUE

```

C Compute new value for tip acceleration.

```

DO 60 M=1,NSHAPS
60      ACCELN(M) = P1(M) + P2(M) + P3(M)

```

RETURN
END

SUBROUTINE AERO (NPTS , BLDANG)

```

C      *****
C      *
C      * Subroutine AERO computes the aerodynamic forces on the *
C      * blade as a function of position along the blade. The *
C      * result is an array of values for D-A(ETA) and D-A(ZETA) *
C      * for each NPTS station down the blade. *
C      *
C      *****
C
C      *****
C      *
C      * External references in this routine: *
C      *
C      * VREL - Calculates the relative velocity components *
C      * for each blade station. *
C      *
C      *****
C
C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      * AERO1 - Holds coefficients related to aerodynamic *
C      * loads calculations such as ClAlpha, CdZero, *
C      * etc. *
C      * AIRFRC - Holds values used in aerodynamic calcula- *
C      * tions. *
C      * BLADE - Holds blade property values such as stiff- *
C      * ness and mass distributions. *
C      * CONST - Turbine and other constants used in load *
C      * calculations. *
C      * TURBN - Holds turbine parameters such as number of *
C      * blades, rotor speed, etc. *
C      * VREL1 - Holds blade section velocity components. *
C      *
C      *****
C
C      *****
C      *
C      * Local and dummy variables used in this routine: *
C      *
C      * ALPHA - Angle of attack. Includes blade twist con- *
C      * tribution. (radians) *
C      * BLDANG - Blade azimuth position used in the teeter- *
C      * ing rotor . *
C      * CD2 - 0.5*AirDensity*ChordLength. *
C      * CDRAG - Drag coefficient. *
C      * CLIFT - Coefficient of lift. Equal to angle of at- *
C      * tack times the lift curve slope. *
C      * I - Generic index. *
C      * INBORD - The number of blade stations that are not *

```

```

C      *           on the airfoil section.  INBORD is used in  *
C      *           other routines to differentiate between  *
C      *           sections of the blade with and without an  *
C      *           airfoil section.  *
C      *           NPTS - Number of points along the blade used  *
C      *           to perform Simpson's integration for calculat-  *
C      *           ing the moments and forces at the blade  *
C      *           root. (passed from STRAP1)  *
C      *           *
C      *****

```

```

REAL      ALPHA
REAL      BLDANG
REAL      CD2
REAL      CDRAG
REAL      CLIFT

```

```

INTEGER   I
INTEGER   INBORD
INTEGER   NPTS

```

```

INCLUDE   'C:INCLUDE\AERO1.INC'
INCLUDE   'C:INCLUDE\AIRFRC.INC'
INCLUDE   'C:INCLUDE\BLADE2.INC'
INCLUDE   'C:INCLUDE\CONST2.INC'
INCLUDE   'C:INCLUDE\TURBN.INC'
INCLUDE   'C:INCLUDE\POSITN.INC'
INCLUDE   'C:INCLUDE\VREL1.INC'

```

```

C      Get the relative fluid velocities from each blade section.

```

```

CALL VREL ( INBORD , NPTS , BLDANG )

```

```

C      Clear out the aerodynamic load values for the inboard portion
C      of the blade - that is, the portion of the blade without any
C      airfoil section. This is related to BLSHMK.

```

```

DO 10 I=1,NPTS

```

```

  IF ( I .LT. INBORD ) THEN

```

```

    DAETA(I) = 0.0
    DAZETA(I) = 0.0
    WW(I)     = 0.0

```

```

  ELSE

```

```

C      See the formulation notes for the derivation of the
C      equations for DAETA, DAZETA and the other values.

```

```

    CD2      = 0.5*RHOAIR*CHORD(I)
    WW(I)    = SQRT( VETA(I)*VETA(I) + VZETA(I)*VZETA(I) )
    ALPHA    = ATAN( VZETA(I)/VETA(I) ) + THETA0(I)
    CLIFT    = ALPHA*CLALFA(I)
    CLIFT    = SIGN( AMIN1( ABS( CLIFT ) , CLMAX(I) ) , CLIFT )
    CDRAG    = CDZERO(I) + DRGFRM*CLIFT**2
    DAETA(I) = CD2*WW(I)*( CLIFT*VZETA(I) - CDRAG*VETA(I) )
    DAZETA(I)= CD2*WW(I)*( CLIFT*VETA(I) + CDRAG*VZETA(I) )

```

```

    IF( BLDANG .EQ. PSI(1)+PI) THEN
      DAET2(I) = DAETA(I)
      DAZET2(I) = DAZETA(I)

```

```

    ELSE
      DAET1(I) = DAETA(I)
      DAZET1(I) = DAZETA(I)

```

```

    ENDIF

```

END IF

10 CONTINUE

RETURN
END
SUBROUTINE AFORCE (NPTS , BLDANG)

```
C *****  
C *  
C * Subroutine AFORCE calculates the value of the aerody- *  
C * namic force integral used in solving the blade equation *  
C * of motion. The aerodynamic force, D-A(ZETA), is inte- *  
C * grated along the blade from the root of the airfoil *  
C * section to the blade tip, and then multiplied by the *  
C * inverse of the CMMASS matrix that was input in the *  
C * DATAIN routine. There is one AFORCE value for each of *  
C * the coordinate shape functions specified by the value *  
C * of NSHAPS. *  
C *  
C *****  
C *****  
C * External references in this routine: *  
C *  
C * AERO - Calculates the aerodynamic forces on the *  
C * blade. *  
C * SIMPSN - Composite Simpson's integration. *  
C *  
C *****  
C *****  
C * Named COMMON blocks used in this routine: *  
C *  
C * AIRFRC Holds values used in aerodynamic calcula- *  
C * tions. *  
C * BLADE - Holds blade property values such as stiff- *  
C * ness and mass distributions. *  
C * MATRX1 - Holds stiffness coefficient matrices. *  
C * TURBN - Holds turbine parameters such as number of *  
C * blades, rotor speed, etc. *  
C *  
C *****  
C *****  
C * Local and dummy variables used in this routine: *  
C *  
C * BLDANG - Blade azimuth position used in the teeter- *  
C * ing rotor option. *  
C * FOFZET - Product of the dA(Zeta) and the shape *  
C * function. *  
C * FTEMP - Temporary array. *  
C * I - Generic index. *  
C * L - Generic index. *  
C * M - Generic index. *  
C * NPTS - Number of points along the blade used to *  
C * perform Simpson's integration for calculat- *  
C * ing the moments and forces at the blade *  
C * root. (passed from STRAP1) *  
C *  
C *****
```

REAL BLDANG
REAL FOFZET (21)
REAL FTEMP (4)
REAL SIMPSN

INTEGER I
INTEGER L
INTEGER M
INTEGER NPTS


```

INCLUDE      'C:INCLUDE\AIRFRC.INC'
INCLUDE      'C:INCLUDE\BLADE2.INC'
INCLUDE      'C:INCLUDE\MATRX1.INC'
INCLUDE      'C:INCLUDE\SHAPE.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'

```

```

C      Compute aerodynamic forces.

```

```

CALL AERO ( NPTS , BLDANG )

```

```

C      Compute the product of the DAZETA value and the shape
C      function at each blade station for each specified shape
C      function.

```

```

DO 20 M=1,NSHAPS
    DO 10 I=1,NPTS
10  FOFZET(I) = DAZETA(I)*SHAPE(M,0,I)

```

```

C      Compute the integral of dF(Aero) along the blade.

```

```

FAERO(M) = SIMPSN( 0.0 , BLTIP , NPTS , FOFZET )

```

```

20 CONTINUE

```

```

C      Multiply the F-AERO integrals by the inverse of the CMMASS
C      coefficient matrix. This is done for the specified number
C      of shape functions.

```

```

DO 50 M=1,NSHAPS
    FTEMP(M) = 0.0
    DO 40 L=1,NSHAPS
40  FTEMP(M) = FTEMP(M) + CMMASS(M,L)*FAERO(L)

```

```

50 CONTINUE

```

```

C      Replace the F-AERO values by their corresponding values
C      multiplied by the CMMASS inverse coefficient matrix.

```

```

DO 60 M=1,NSHAPS
60 FAERO(M) = FTEMP(M)

```

```

RETURN
END
SUBROUTINE CAPS ( STRING )

```

```

C      *****
C      *
C      * Subroutine CAPS is used to convert alphabetic charac- *
C      * ters into upper case. It assumes that all the lower *
C      * case letters are in one contiguous block and all the *
C      * upper case letters are in another contiguous block. *
C      * This routine will need to be changed to work with a *
C      * noncontiguous character set like IBM's EBCDIC. It is *
C      * not needed for an upper case only character set like *
C      * CDC's Display Code. *
C      *
C      *****

```

```

C      *****
C      *
C      * Named COMMON blocks used in this routine: *

```

```

C      *
C      *      none
C      *
C      *****

C      *****
C      *
C      *      External references in this routine:
C      *
C      *      none
C      *
C      *****

C      *****
C      *
C      *      Local and dummy variables used in this routine:
C      *
C      *      DIFF - Numerical difference between 'A' and 'a'.
C      *      I - Generic index.
C      *      LENGTH - The declared length of the given string.
C      *      STRING - The string needing to be converted to upper
C      *      case.
C      *
C      *****

INTEGER      DIFF
INTEGER      I
INTEGER      LENGTH

CHARACTER*(*)  STRING

C      Compute the numerical difference between 'A' and 'a'.

DIFF = ICHAR( 'a' ) - ICHAR( 'A' )

C      Get the length of the string.

LENGTH = LEN( STRING )

C      Look for lower case letters. Convert them to upper case.

DO 100 I=1,LENGTH

  IF ( ( STRING(I:I) .GE. 'a' ) .AND.
&      ( STRING(I:I) .LE. 'z' ) ) THEN

    STRING(I:I) = CHAR( ICHAR( STRING(I:I) ) - DIFF )

  END IF

100 CONTINUE

RETURN
END
SUBROUTINE CONVRT ( TODEGS )

C      *****
C      *
C      *      Subroutine CONVRT performs the units conversion speci-
C      *      fied by the logical value of TODEGS. The calling rou-
C      *      tine sends a value of .TRUE. to indicate conversion to
C      *      degrees or .FALSE. to indicate conversion to radians.
C      *      This conversion back and forth between radians and de-
C      *      grees is to make the interactive portions of the SETUP
C      *      and LIMITS routines more 'User Friendly'.
C      *
C      *****

C      *****
C      *
C      *      External references in this routine:
C      *

```



```

PSIZER = FACT1*PSIZER
THETAP = FACT1*THETAP
THETAT = FACT1*THETAT
DELT3  = FACT1*DELT3
BETA1  = FACT1*BETA1
BETA2  = FACT1*BETA2
OMEGA  = FACT2*OMEGA

```

```

C      Convert 'limits' variables from the LIMITS routine.

```

```

BEGIN2 = FACT1*BEGIN2
END2   = FACT1*END2
PRINT1 = FACT1*PRINT1
PRINT2 = FACT1*PRINT2
STEPMX = FACT1*STEPMX
STEPMN = FACT1*STEPMN

```

```

RETURN
END
SUBROUTINE DATAIN

```

```

C      *****
C      *
C      * Subroutine DATAIN reads the data file produced by the *
C      * PRNCOE routine of Module 1. The name of the run-data *
C      * file is specified interactively. DATAIN opens the spe- *
C      * cified file, reads the data and then closes the file. *
C      *
C      * Note: If any changes have been made to the format of *
C      * the run-data file in PRNCOE of Module 1, then the *
C      * same changes MUST be made to DATAIN. *
C      *
C      *****

```

```

C      *****
C      *
C      * External references in this routine: *
C      *
C      * CONVRT - Performs units conversions. *
C      *
C      *****

```

```

C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      * AER01 - Holds coefficients related to aerodynamic *
C      * loads calculations such as ClAlpha, CdZero, *
C      * etc. *
C      * BLADE - Holds blade property values such as stiff- *
C      * ness and mass distributions. *
C      * LITERL - Holds data set titles. *
C      * LODVAL - Holds values used to compute blade loads. *
C      * MATRX1 - Holds stiffness coefficient matrices. *
C      * MATRX2 - Holds other matrices related to coriolis *
C      * stiffening, gravity, etc. *
C      * TENSIN - Holds tension component integrals. *
C      * TURBN - Holds turbine parameters such as number of *
C      * blades, rotor speed, etc. *
C      * WIND - Holds wind shear and tower shadow parame- *
C      * ters. *
C      *
C      *****

```

```

C      *****
C      *
C      * Local and dummy variables used in this routine: *
C      *
C      * COL - Column index into matrices. *
C      * I - Generic index. *
C      * N - Generic index. *
C      * NPTS - Number of points along the blade used to *
C      * perform Simpson's integration for calculat- *
C      * ing the moments and forces at the blade *
C      * root. (passed from STRAP1) *
C      *
C      *****

```

```

C      *      ROW   - Row index into matrices.          *
C      *      RUNDAT - Name of file containing run data. *
C      *      TORADS - Flag used to tell CONVRT to convert 'free' *
C      *                  variables to radians.          *
C      *
C      *****

```

```

INTEGER      COL
INTEGER      I
INTEGER      N
INTEGER      NPTS
INTEGER      ROW

CHARACTER*50  RUNDAT

LOGICAL      TORADS

INCLUDE      'C:INCLUDE\AER01.INC'
INCLUDE      'C:INCLUDE\BLADE2.INC'
INCLUDE      'C:INCLUDE\LITERL.INC'
INCLUDE      'C:INCLUDE\LODVAL.INC'
INCLUDE      'C:INCLUDE\MATRIX1.INC'
INCLUDE      'C:INCLUDE\MATRIX2.INC'
INCLUDE      'C:INCLUDE\TENSIN2.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\WIND2.INC'
INCLUDE      'C:INCLUDE\SHAPE.INC'
INCLUDE      'C:INCLUDE\TRBINF.INC'
INCLUDE      'C:INCLUDE\HUBPRP2.INC'
INCLUDE      'C:INCLUDE\SPRING2.INC'

SAVE        TORADS

DATA        TORADS / .FALSE. /

```

```

2000 FORMAT ( 3( A / ) / )
2100 FORMAT ( 9X , L7 )
3000 FORMAT ( // 3( 1PE24.7 ) )
3100 FORMAT ( // 114 , 3( I18 ) )
3200 FORMAT ( / 18X, F6.2)
3300 FORMAT ( / 18X, F10.2)
3400 FORMAT ( / 18X, I6)
3500 FORMAT ( / 18X, F10.5)
3600 FORMAT ( / 18X, I3)
3700 FORMAT ( / 2( / 5( 1PE14.3 , : ) ) )
4000 FORMAT ( / 6( / 4( 1PE18.7 , : ) ) )
4100 FORMAT ( '&done.' )

```

```

C      Get name of run-data file. Open it.

100 PRINT *, ' '
PRINT *, 'Enter the name of the file that contains run-data > '
READ (*,'(A)') RUNDAT
PRINT *, RUNDAT

OPEN ( 3 , FILE=RUNDAT , STATUS='OLD' , ERR=110 )

PRINT *, 'Reading in new data...'

GO TO 200

```

```

C      Error opening data file.

```

```
110 PRINT *, ' >>> Run-data file not found. Try again. <<<'
GO TO 100
```

C Read titles and hub type.

```
200 READ (3,2000) TITLE1 , TITLE2 , TITLE3
```

C Read scalar data.

```
READ (3,3000) ALENTH , ALPHA0 , BETA0
READ (3,3000) BLSHNK , BLTIP , CHI
READ (3,3000) CSUBMA , DRGFRM , HUBHT
READ (3,3000) HUBRAD , OMEGA , PHIO
READ (3,3000) PHIAMP , PHIOMG , PSIZER
READ (3,3000) SHERXP , THETAP , THETAT
READ (3,3000) TSUBO , TSUBP , VHUB
READ (3,3100) KSHADW , NBLADS , NSHAPS , NPTS
READ (3,3200) DELT3
READ (3,3200) UNDSLK
READ (3,3200) HUBMAS
READ (3,3200) HUBDIS
READ (3,3200) BETA1
READ (3,3200) BETA2
READ (3,3400) NUMSCN
READ (3,3500) TIMINC
READ (3,3600) MSTAT
READ (3,3700) ( STA(I), I=1,MSTAT)
```

C Read in vector values.

```
READ (3,4000) ( CLALFA(I) , I=1,NPTS )
READ (3,4000) ( CLMAX (I) , I=1,NPTS )
READ (3,4000) ( CDZERO(I) , I=1,NPTS )
READ (3,4000) ( CHORD (I) , I=1,NPTS )
READ (3,4000) ( ECNTFN(I) , I=1,NPTS )
READ (3,4000) ( ESUBAC(I) , I=1,NPTS )
READ (3,4000) ( THETA0(I) , I=1,NPTS )
```

C Read in coefficient matrices.

```
READ (3,4000) ( ( CKBEND(ROW,COL) , COL=1,4 ) , ROW=1,4 )
READ (3,4000) ( ( CKTOMG(ROW,COL) , COL=1,4 ) , ROW=1,4 )
READ (3,4000) ( ( CKTGRV(ROW,COL) , COL=1,4 ) , ROW=1,4 )
READ (3,4000) ( ( CKQLOD(ROW,COL) , COL=1,4 ) , ROW=1,4 )
READ (3,4000) ( ( CMMASS(ROW,COL) , COL=1,4 ) , ROW=1,4 )
```

```
DO 300 N=1,4
```

```
300 READ (3,4000) ( ( CKTCRL(N,ROW,COL) , COL=1,4 ) , ROW=1,4 )
```

C Read in vector coefficients.

```
READ (3,4000) ( CMRIGD(I) , I=1,4 )
READ (3,4000) ( CMRGD1(I) , I=1,4 )
READ (3,4000) ( CMBLNC(I) , I=1,4 )
READ (3,4000) ( CMGRAV(I) , I=1,4 )
READ (3,4000) ( CMGRV1(I) , I=1,4 )
READ (3,4000) ( CMHUB1(I) , I=1,4 )
READ (3,4000) ( KOSTIF(I) , I=1,4 )
READ (3,4000) ( K1STIF(I) , I=1,4 )
READ (3,4000) ( K2STIF(I) , I=1,4 )
READ (3,4000) ( DAMP(I) , I=1,4 )
```

C Read tension arrays.

```
DO 400 N=1,4
```

```
400 READ (3,4000) ( TCORLS(N,I) , I=1,NPTS )
```

```
READ (3,4000) ( TGRAV(I) , I=1,NPTS )
```

```
READ (3,4000) ( TOMGA(I) , I=1,NPTS )
```

```
DO 410 N=1,4
```

```
410 READ (3,4000) ( CIFMOM(N,I) , I=1,NPTS )
```

C Read arrays of special load computation functions.

```
READ (3,4000) ( DELTIM(I) I=1,NPTS )
READ (3,4000) ( OFFMAS(I) , I=1,NPTS )
```

C Read in modeshapes and derivatives calculated in
C module 1. Modes 1 and 3 are the rigid teeter and
C first asymmetric bending, while modes 2 and 4
C are first and second symmetric bending. For frequency
C information see the modeshape file produced as a result
C of running module 1.

C There is now no longer any need for the old SUBROUTINE
C SHAPES as the modeshapes are read in directly from this
C input file into the shape arrays, instead of re-calculating
C them as done in the "old" FLAP version.

```
DO 520 I = 1,NPTS
  READ(3,*) (SHAPE(J,0,I), J=1,4)
520 CONTINUE

DO 530 I = 1,NPTS
  READ(3,*) (SHAPE(J,1,I), J=1,4)
530 CONTINUE

DO 540 I = 1,NPTS
  READ(3,*) (SHAPE(J,2,I), J=1,4)
540 CONTINUE
```

C Processing complete. Close data file.

```
PRINT 4100
CLOSE ( 3 )
```

```
PRINT *, 'Analyzing a teetering hub.'
```

C The units are converted to radians and radians/second to permit
C compatibility between newly input data and data being used for
C consecutive model runs. Thus the DIAG and RUN routines do not
C need to know if the data they are using is new or old.

```
CALL CONVRT ( TORADS )
```

```
CDELTA3 = COS(DELTA3)
SDELTA3 = SIN(DELTA3)
TDELTA3 = TAN(DELTA3)
CDAMP = DAMP(1)*BLTIP*BLTIP
```

```
RETURN
END
SUBROUTINE DIAG ( NPTS )
```

```
C *****
C *
C * Subroutine DIAG performs a test run of the aerodynamic *
C * routines for diagnostic purposes. It steps around the *
C * rotor disk in specified increments and calculates the *
C * aerodynamic forces along the blade, the relative veloc- *
C * ity components, the induced velocity components and the *
C * windspeeds. These are printed out in chart form for *
C * each specified azimuth printout interval around the *
C * disk. DIAG can be run repetitively for sensitivity *
C * analyses. *
C *
C * The following parameters are printed for each of the *
C * specified positions around the disk: *
C *
C * DAETA DAZETA VETA VZETA VWIND *
C * WSHEAR VINDR VINDO DVIND ALPHA *
C * CDZERO CLLIFT VV *
C *****
```

```

C *****
C *
C * External references in this routine:
C *
C * AERO - Calculates the aerodynamic forces on the
C * blade.
C * CONVRT - Performs units conversions.
C * SETUP - Interactive modification of free variables.
C *
C *****
C
C *****
C *
C * Named COMMON blocks used in this routine:
C *
C * AERO1 - Holds coefficients related to aerodynamic
C * loads calculations such as ClAlpha, CdZero,
C * etc.
C * AIRFRC - Holds values used in aerodynamic calcula-
C * tions.
C * BLADE - Holds blade property values such as stiff-
C * ness and mass distributions.
C * CONST - Turbine and other constants used in load
C * calculations.
C * DELTV - Holds turbine inputs for possible future
C * use. Not currently used.
C * FORMS - Holds blade deflections.
C * POSITN - Holds parameters related to blade position
C * such as PHI, PSI, etc.
C * SARAYS - Holds new and old values for the general-
C * ized coordinates.
C * SHAPE - Holds blade coordinate shape functions.
C * TURBN - Holds turbine parameters such as number of
C * blades, rotor speed, etc.
C * VINDUC - Holds induced velocity components.
C * VREL1 - Holds blade section velocity components.
C * WNDVEL - Holds values used in wind shear and tower
C * shadow computations.
C *
C *****
C
C *****
C *
C * Local and dummy variables used in this routine:
C *
C * ALPHA - Angle of attack. Includes blade twist con-
C * tribution. (radians)
C * ANS - Used to store input responses from key-
C * board.
C * BLDANG - Blade azimuth position used in the teeter-
C * ing rotor option.
C * CLLIFT - Coefficient of lift. Equal to angle of at-
C * tack times the lift curve slope.
C * I - Generic index.
C * ICALL - Counter to save the number of calls to sub-
C * routine DIAG. Used to open output file.
C * IDPSI - Printout interval in whole degrees around
C * the rotor disk.
C * IPSI - Do loop counter for azimuth position.
C * IPT - Array index used for trapezoidal integra-
C * tion.
C * NPTS - Number of points along the blade used to
C * perform Simpson's integration for calculat-
C * ing the moments and forces at the blade
C * root. (passed from STRAP1)
C * NSHP - Counter on DO loops for the coordinate
C * shape function.
C * TODEGS - Flag used to tell CONVRT to convert 'free'
C * variables to degrees.
C * TORADS - Flag used to tell CONVRT to convert 'free'
C * variables to radians.
C *
C *****

```

```

REAL ALPHA (21)
REAL BLDANG

```



```

REAL          CLLIFT (21)
REAL          CDDRAG (21)

INTEGER       I
INTEGER       ICALL
INTEGER       IDPSI
INTEGER       IPSI
INTEGER       IPT
INTEGER       NPTS
INTEGER       NSHP

CHARACTER*1   ANS

LOGICAL       TODEGS
LOGICAL       TORADS

INCLUDE       'C:INCLUDE\AER01.INC'
INCLUDE       'C:INCLUDE\AIRFRC.INC'
INCLUDE       'C:INCLUDE\BLADE2.INC'
INCLUDE       'C:INCLUDE\CONST2.INC'
INCLUDE       'C:INCLUDE\DELT.V.INC'
INCLUDE       'C:INCLUDE\FORMS.INC'
INCLUDE       'C:INCLUDE\POSITN.INC'
INCLUDE       'C:INCLUDE\SARAYS.INC'
INCLUDE       'C:INCLUDE\SHAPE.INC'
INCLUDE       'C:INCLUDE\TURBN.INC'
INCLUDE       'C:INCLUDE\VINDUC.INC'
INCLUDE       'C:INCLUDE\VREL1.INC'
INCLUDE       'C:INCLUDE\WNDVEL.INC'

SAVE          ICALL
SAVE          TODEGS
SAVE          TORADS

DATA          ICALL / 0 /
DATA          TODEGS / .TRUE. /
DATA          TORADS / .FALSE. /

```

```

1000 FORMAT ( '1' /// 10X , 'Test Run of Aerodynamic Routines' /// )
2000 FORMAT ( // 10X , 'Azimuth position: PSI = ' , I3
&          , 10X , 'Hub velocity = ' , F10.3
&          / 48X , 'TSHADW = ' , F10.3 )
2100 FORMAT ( /// ' I DAET1(I) DAZET1(I) DAET2(I) DAZET2(I)'
&          , ' VWIND(I) WSHEAR(I) VINDR(I) VINDO(I)'
&          , ' DVIND(I) ALPHA(I) CDDRAG(I) '
&          , ' CLLIFT(I) VV(1,I)' / )
2200 FORMAT ( 14 , 13( 1PE10.3 ) )
3000 FORMAT ( A )

```

C If this is the first time DIAG is called, open the output file.

```

100 IF ( ICALL .EQ. 0 ) THEN
OPEN ( 2 , FILE='DIAGNOS.DAT' , STATUS='UNKNOWN'
&     , CARRIAGECONTROL='FORTRAN' )
ICALL = 1
END IF

```

C Print heading in diagnostic file.

```

WRITE (2,1000)

PRINT *, ' '
PRINT *, 'Subroutine DIAG: Diagnostic run and output'

```

```
PRINT *, ' '
```

C Convert values into degrees.

```
CALL CONVRT ( TODEGS )
```

C Run setup routine.

```
CALL SETUP
```

C Convert values back into radians for internal use.

```
CALL CONVRT ( TORADS )
```

C Ask for printout interval.

```
PRINT *, ' '  
PRINT *, 'Enter printout interval around rotor disk in whole '  
& READ *, 'degrees > '  
READ *, IDPSI
```

```
DELPSI(1) = IDPSI/RAD2DG  
DELTAT(1) = DELPSI(1)/OMEGA
```

C NOTE: DIAG assumes that time is frozen at time=0.

```
PHI(0) = PHIO  
PHI(1) = PHIOMG  
PHI(2) = 0.0
```

C Go around the disk in IDPSI increments.

```
DO 250 IPSI=0,360,IDPSI
```

C Clear the forms array.

```
PSI(1) = IPSI/RA02DG  
BLDANG = PSI(1)
```

C Set up the forms array.

```
203 DO 200 I=1,NPTS  
VV(0,I) = 0.0  
VV(1,I) = 0.0  
V1(0,I) = 0.0  
V1(1,I) = 0.0  
200 CONTINUE
```

```
DO 220 IPT=1,NPTS
```

```
IF( BLDANG .EQ. PSI(1)+PI) THEN  
V1(0,IPT) = V1(0,IPT) - SHAPE(1,0,IPT)*SNEW(1,0,IPSI)  
V1(1,IPT) = V1(1,IPT) - SHAPE(1,0,IPT)*SNEW(1,1,IPSI)  
ELSE  
V1(0,IPT) = V1(0,IPT) + SHAPE(1,0,IPT)*SNEW(1,0,IPSI)  
V1(1,IPT) = V1(1,IPT) + SHAPE(1,0,IPT)*SNEW(1,1,IPSI)  
ENDIF
```

```
DO 210 NSHP=1,NSHAPS
```

```
IF( BLDANG .EQ. PSI(1)+PI) THEN  
MARK = (-1.)*NSHP  
ELSE  
MARK = 1.  
ENDIF
```

```
& VV(0,IPT) = VV(0,IPT)  
+ MARK*SHAPE(NSHP,0,IPT)*SNEW(NSHP,0,IPSI)
```

```

      VV(1,IPT) = VV(1,IPT)
&      + MARK*SHAPE(NSHP,0,IPT)*SNEW(NSHP,1,IPSI)
210      CONTINUE
220      CONTINUE
      BETA = SNEW(1,0,IPSI)/BLTIP
      BETAD = SNEW(1,1,IPSI)/BLTIP
C      Compute the aerodynamic forces on the blade.
      CALL AERO ( NPTS , BLDANG )
      IF( BLDANG .EQ. PSI(1) + PI) GO TO 204
      BLDANG = PSI(1) + PI
      GO TO 203
204      DO 230 I=1,NPTS
          IF ( DAETA(I) .EQ. 0.0 ) THEN
              ALPHA(I) = 0.0
          ELSE
              ALPHA(I) = ATANC( VZETA(I)/VETA(I) ) + THETAO(I)
          END IF
          CLLIFT(I) = AMIN1( CLALFA(I)*ABS( ALPHA(I) ) , CLMAX(I) )
          CLLIFT(I) = SIGN( CLLIFT(I) , ALPHA(I) )
          CDDRAG(I) = CDZERO(I) + DRGFRM*CLLIFT(I)**2
230      CONTINUE
C      Print out results.
      WRITE (2,2000) IPSI , HUBVEL , TSHADW
      WRITE (2,2100)
240      DO 240 I=1,21
          WRITE (2,2200) I, DAET1(I), DAZET1(I), DAET2(I), DAZET2(I)
&          , VWIND(I), WSHEAR(I), VINDR(I), VINDO(I)
&          , DVIND(I), ALPHA(I), CDDRAG(I), CLLIFT(I)
&          , VV(1,I)
250      CONTINUE
C      Test run completed. See if user wants more.
      PRINT *, ' '
      PRINT *, 'Test run completed. Results written onto file '
&      , 'DIAGNOS.DAT'
300      PRINT *, ' '
      PRINT *, 'Do you want to perform another test run? (Y,=N) > '
      READ 3000, ANS
      PRINT *, ANS
      IF ( ( ANS .EQ. 'Y' ) .OR. ( ANS .EQ. 'y' ) ) GO TO 100
      IF ( ( ANS .NE. 'N' ) .AND. ( ANS .NE. 'n' )
&      .AND. ( ANS .NE. ' ' ) ) THEN
          PRINT *, ' >>> Invalid response. Please try again. <<<'
          GO TO 300
      END IF
      RETURN
      END
      SUBROUTINE DSKREV ( NPTS )

```

```

C      *****
C      *

```

```

C      * Subroutine DSKREV performs a full disk revolution solu- *
C      * tion to the blade equations of motion and saves the re- *
C      * sulting values. *
C      * *
C      *****
C      * *
C      * External references in this routine: *
C      * *
C      * EULER - Self starting modified Euler predictor/cor- *
C      * rector integration. *
C      * NXTPHI - Calculates the next values of the yaw vari- *
C      * ables. *
C      * NXTPSI - Calculates the next value of the azimuth *
C      * angle, Psi. *
C      * SAVE1 - Saves tip displacement variables. *
C      * STRTUP - Calculates the static deflection of the *
C      * blade at the startup position. *
C      * TRACE - Traces certain variables. *
C      * *
C      *****
C      * *
C      * Named COMMON blocks used in this routine: *
C      * *
C      * CONST - Turbine and other constants used in load *
C      * calculations. *
C      * LIMITC - Holds values used in the LIMITS routine. *
C      * POSITN - Holds parameters related to blade position *
C      * such as PHI, PSI, etc. *
C      * SARAYS - Holds new and old values for the general- *
C      * ized coordinates. *
C      * TURBN - Holds turbine parameters such as number of *
C      * blades, rotor speed, etc. *
C      * *
C      *****
C      * *
C      * Local and dummy variables used in this routine: *
C      * *
C      * I - Generic index. *
C      * IPSIST - The next STEPMX station in integer degrees. *
C      * ISTEP - STEPMX in integer degrees. *
C      * J - Generic index. *
C      * NEW - Array index for new data. *
C      * NPTS - Number of points along the blade used to *
C      * perform Simpson's integration for calculat- *
C      * ing the moments and forces at the blade *
C      * root. (passed from STRAP1) *
C      * NSHP - Counter on DO loops for the coordinate *
C      * shape function. *
C      * OLD - Array index for old data. *
C      * STEMP - Temporary array holding the tip displace- *
C      * ment values. The third dimension repre- *
C      * sents the order of the time derivative. *
C      * TIMNOW - Current time. *
C      * *
C      *****

```

```

REAL      STEMP (4,0:1,0:2)
REAL      TIMNOW

INTEGER   I
INTEGER   IPSIST
INTEGER   ISTEP
INTEGER   J
INTEGER   NEW
INTEGER   NPTS
INTEGER   NSHP
INTEGER   OLD

INCLUDE   'C:INCLUDE\CONST2.INC'

INCLUDE   'C:INCLUDE\LIMITC.INC'

```

```

INCLUDE      'C:INCLUDE\POSITN.INC'
INCLUDE      'C:INCLUDE\SARAYS.INC'
INCLUDE      'C:INCLUDE\START.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'

SAVE        NEW
SAVE        OLD
SAVE        STEMP

```

```

DATA        NEW    / 1 /
DATA        OLD    / 0 /

```

```

1000 FORMAT ( ' Static deflection at startup (MODE' , I2.2 , ') ='
&          , F8.4 / )

```

```

C          Initialize some variables.

```

```

ISTEP = RAD2DG*STPEMX + .001

```

```

C          IPSIST is the next STEPMX station. That is, then next azimuth
C          location at which the deflection, velocity and acceleration
C          values will be saved and the loads will be computed. Values
C          for azimuth positions intermediate to the loads will not be
C          saved.

```

```

IPSIST = RAD2DG*( PSI(1) + STEPMX ) + 0.001

```

```

C          When the ITRIM flag is set to 2, then this is the first pass
C          through DSKREV for this analysis. If it is, then compute the
C          static deflection of the blade tip for the startup condition.

```

```

IF ( ITRIM .EQ. 2 ) THEN

```

```

    CALL STRTUP ( STEMP , NPTS )

```

```

C          This section prints the computed static starting deflection
C          when the program is running in the trace mode.

```

```

IF ( TRACEF ) THEN

```

```

    DO 100 I=1,NSHAPS
100    WRITE (4,1000) I , STEMP(I,1,0)

```

```

    END IF

```

```

END IF

```

```

C          Push current values of S, SDOT and SDD onto old values.

```

```

DO 220 NSHP=1,NSHAPS

```

```

    DO 200 J=0,360,ISTEP
200    SOLD(NSHP,J) = SNEW(NSHP,0,J)

```

```

    DO 210 I=0,2
210    SNEW(NSHP,I,0) = SNEW(NSHP,I,360)

```

```

220 CONTINUE

```

```

C          Save old values of position variables and error.

```

```

300 DELPSI(0) = DELPSI(1)
    DELTAT(0) = DELTAT(1)
    PSI (0) = PSI (1)
    TIME (0) = TIME (1)

```

```

C          Get next PSI, delta PSI, etc. and next yaw values PHI, PHI dot

```

```

C      and PHI dot-dot.
400 CALL NXTPSI ( IPSIST )
      TIMNOW = TIME(1)
      CALL NXTPHI ( TIMNOW )

C      Solve the equation of motion and compute tip displacements and
C      velocities. Check error of Euler predictor-corrector solution.
C      If the error condition is satisfied, check for disk station and
C      save the new values if this azimuth is a STEPMX disk station.

      CALL EULER ( STEMP , NSHAPS , NPTS )

      IF ( (ERROR .GT. EUERR) .AND. (DELPSI(1) .GT. STEPMM) ) GO TO 400

C      The following tests check for the current azimuth location
C      between the STEPMX stations. If the current position is very
C      close to a STEPMX station, then the deflection, velocity and
C      acceleration values are saved.

      IF ( ABS( PSI(1) - IPSIST/RAD2DG ) .LT. 0.1*STEPMM ) THEN

          CALL SAVE1 ( IPSIST , NSHAPS , STEMP )

C      When the TRACEF flag is set, print out the prescribed
C      values. See subroutine TRACE for a full description.

          IF ( TRACEF ) CALL TRACE ( STEMP , NPTS )

      END IF

C      Move new temporary values to old.

      DO 510 NSHP=1,NSHAPS
          DO 500 I=0,2
500      STEMP(NSHP,OLD,I) = STEMP(NSHP,NEW,I)
510 CONTINUE

C      If we haven't gone all the way around, solve the equation of
C      motion for the next step.

      IF ( IPSIST .LE. 360 ) GO TO 300

      RETURN
      END
      SUBROUTINE EULER ( STEMP , NSHAPS , NPTS )

C      *****
C      *
C      * Subroutine EULER uses the self starting modified pre- *
C      * dictor-corrector method to compute the values of the *
C      * blade tip displacement and velocity, given the previous *
C      * values and the solution to the blade equation of mo- *
C      * tion. The solution is not iterated for a specified de- *
C      * sired accuracy, but is instead run through one computa- *
C      * tion of the predictor-corrector. The error function *
C      * compares the percentage change in tip deflection be- *
C      * tween two consecutive azimuth locations against the *
C      * static tip deflection computed by the startup routine. *
C      * This error value is sent back to the calling routine *
C      * (ie. DSKREV). If the error is too large, a smaller az- *
C      * imuth angle step size will be used and another call *
C      * will be made to this routine. *
C      *
C      *****
C      *****

```

```

C      *
C      * External references in this routine:
C      *
C      * ACCEL - Solves the blade equation of motion.
C      *
C      *****
C
C      *****
C      *
C      * Named COMMON blocks used in this routine:
C      *
C      * POSITN - Holds parameters related to blade position
C      *          such as PHI, PSI, etc.
C      *
C      * START - Holds initial blade deflection.
C      *
C      *****
C
C      *****
C      *
C      * Local and dummy variables used in this routine:
C      *
C      * ACCELN - Tip acceleration.
C      * DEFLTN - Tip displacement.
C      * NEW - Array index for new data.
C      * NPTS - Number of points along the blade used to
C      *          perform Simpson's integration for calculat-
C      *          ing the moments and forces at the blade
C      *          root. (passed from STRAP1)
C      *
C      * NSHAPS - Number of blade shape functions, 4 maximum.
C      * NSHP - Counter on DO loops for the coordinate
C      *          shape function.
C      *
C      * OLD - Array index for old data.
C      * SCORCT - Corrected values for blade tip accelera-
C      *          tion.
C      *
C      * SPRDCT - Predicted values for blade tip accelera-
C      *          tion.
C      *
C      * STEMP - Temporary array holding the tip displace-
C      *          ment values. The third dimension repre-
C      *          sents the order of the time derivative.
C      *
C      * VELCTY - Tip velocity.
C      *
C      *****

```

```

REAL          ACCELN (4)
REAL          DEFLTN (4)
REAL          SCORCT (4)
REAL          SPRDCT (4)
REAL          STEMP (4,0:1,0:2)
REAL          VELCTY (4)

```

```

INTEGER       NEW
INTEGER       NPTS
INTEGER       NSHAPS
INTEGER       NSHP
INTEGER       OLD

```

```

INCLUDE       'C:INCLUDE\POSITN.INC'

```

```

INCLUDE       'C:INCLUDE\START.INC'

```

```

SAVE         NEW
SAVE         OLD

```

```

DATA         NEW    / 1 /
DATA         OLD    / 0 /

```

```

C      Calculate predictor values.

```

```

DO 100 NSHP=1,NSHAPS

```

```

      STEMP(NSHP,NEW,1) = STEMP(NSHP,OLD,1)
&      + STEMP(NSHP,OLD,2)*DELTAT(NEW)
      STEMP(NSHP,NEW,0) = STEMP(NSHP,OLD,0)
&      + STEMP(NSHP,OLD,1)*DELTAT(NEW)

```

```

      SPRDCT(NSHP) = STEMP(NSHP,NEW,0)

```

100 CONTINUE

C Calculate predicted acceleration values by solving the blade
C equation of motion with the predicted values of the tip
C velocity and displacement.

DO 200 NSHP=1,NSHAPS

DEFLTNSHP) = STEMP(NSHP,NEW,0)
VELCTY(NSHP) = STEMP(NSHP,NEW,1)

200 CONTINUE

C Subroutine ACCEL performs the actual solution of the blade
C equation of motion. It uses the previous values of the tip
C deflection and velocity and the current values of the forces.
C It returns the new blade tip acceleration values.

CALL ACCEL (DEFLTNSHP , VELCTY , ACCELNSHP , NPTS , STEMP)

C Calculate corrector values.

DO 300 NSHP=1,NSHAPS

STEMP(NSHP,NEW,2) = ACCELNSHP)

STEMP(NSHP,NEW,0) = STEMP(NSHP,OLD,0)
& + 0.5*DELTAT(1)*(STEMP(NSHP,OLD,1)
& + STEMP(NSHP,NEW,1))

STEMP(NSHP,NEW,1) = STEMP(NSHP,OLD,1)
& + 0.5*DELTAT(1)*(STEMP(NSHP,OLD,2)
& + STEMP(NSHP,NEW,2))

SCORCT(NSHP) = STEMP(NSHP,NEW,0)

300 CONTINUE

C The error is computed from the sum of the ratios of the tip
C displacement change from the predicted to the corrected values
C relative to the initial static blade deflection computed at
C startup. Summation is over the number of coordinate shape
C functions specified in the run.

ERROR = 0.0

DO 400 NSHP=1,NSHAPS

400 ERROR = ERROR + (SCORCT(NSHP) - SPRDCT(NSHP))/S0(NSHP)

C The error value is converted into percent for use in the
C DSKREV routine.

ERROR = 100.0*ABS(ERROR)

RETURN

END

SUBROUTINE FORM1 (STEMP , NPTS , NSHAPS , BLDANG)

C *****
C *
C * Subroutine FORM1 computes the values of the blade dis- *
C * placement function for each of the NPTS stations along *
C * the blade. The values are produced solely for use in *
C * computing the blade relative velocity values that are *
C * in turn used to compute the blade aerodynamic forces *
C * and ultimately the aerodynamic force function used in *
C * the blade equation of motion. *
C *
C *****
C *****


```

C      VVSLP(I) Is the blade slope (elastic only)

DO 100 I=1,NPTS
  VV(0,I) = 0.0
  VV(1,I) = 0.0
  V1(0,I) = 0.0
  V1(1,I) = 0.0
  VE(0,I) = 0.0
  VE(1,I) = 0.0
  VVSLP(I)=0.0
100 CONTINUE

C      Compute the VV values down the blade for each of the specified
C      shape functions.

  ANGLE1 = PSI(1) + PI
  ANGLE2 = PSI(1)

C      Determine the teeter angle and velocity:
C      Note that the teeter angle is the total angle
C      about the teeter pin. The value V1 is the projection
C      of the teeter deflection and velocity onto the
C      Xp, Yp, Zp system of axes: V1 = r*beta*cos(delt3)
C      *cos(thetap).

  BETA = STEMP(1,OLD,0)/(BLTIP+HUBRAD)
  BETAD = STEMP(1,OLD,1)/(BLTIP+HUBRAD)

  IF ( BLDANG .EQ. ANGLE1 ) THEN

    BETA = -1.*BETA
    BETAD = -1.*BETAD

    DO 180 I = 1,NPTS
      V1(0,I) = V1(0,I) - SHAPE(1,0,I)*STEMP(1,OLD,0)
      V1(1,I) = V1(1,I) - SHAPE(1,0,I)*STEMP(1,OLD,1)
180  CONTINUE

    DO 210 NSHP=1,NSHAPS

      MARK=(-1)**NSHP

      DO 200 I=1,NPTS
        VV(0,I)=VV(0,I)+MARK*SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,0)
        VV(1,I)=VV(1,I)+MARK*SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,1)
        VVSLP(I)=VVSLP(I)+MARK*SHAPE(NSHP,1,I)*STEMP(NSHP,OLD,0)
200  CONTINUE
210  CONTINUE

C  DETERMINE ONLY THE ELASTIC CONTRIBUTION TO DEFLECTION
C  AND VELOCITY

  DO 650 NSHP = 2,NSHAPS

    MARK=(-1)**NSHP

    DO 640 I = 1,NPTS
      VE(0,I)=VE(0,I)+MARK*SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,0)
      VE(1,I)=VE(1,I)+MARK*SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,1)
640  CONTINUE
650  CONTINUE

  ELSE IF ( BLDANG .EQ. ANGLE2 ) THEN

    DO 215 I = 1,NPTS
      V1(0,I) = V1(0,I) + SHAPE(1,0,I)*STEMP(1,OLD,0)
      V1(1,I) = V1(1,I) + SHAPE(1,0,I)*STEMP(1,OLD,1)
215  CONTINUE

    DO 230 NSHP=1,NSHAPS

      DO 220 I=1,NPTS

```

```

        WV(0,I) = WV(0,I) + SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,0)
        WV(1,I) = WV(1,I) + SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,1)
        VVSLP(1)= VVSLP(1) + SHAPE(NSHP,1,I)*STEMP(NSHP,OLD,0)
220     CONTINUE
230     CONTINUE

DO 670 NSHP = 2,NSHAPS

    DO 660 I = 1,NPTS
        VE(0,I) = VE(0,I) + SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,0)
        VE(1,I) = VE(1,I) + SHAPE(NSHP,0,I)*STEMP(NSHP,OLD,1)
660     CONTINUE
670     CONTINUE

ELSE

    PRINT *,      >>> Error in the blade angle, BLDANG, within'
&    PRINT *,    ' subroutine FORM1 <<<'
    PRINT *,    ' '
    PRINT *,    ' '
    PRINT *,    'STRAP terminated abnormally due to the error'
&    PRINT *,    ' listed above.'
    PRINT *,    ' '

    STOP

END IF

RETURN
END
SUBROUTINE GAUSSJ ( A , N )

```

```

C      *****
C      *
C      * Subroutine GAUSSJ is used to solve linear equations by *
C      * Gauss-Jordan elimination with full pivoting. It was *
C      * transcribed from the book "Numerical Recipes" by Wil- *
C      * liam H. Press, et al. Code for right-hand-side vectors *
C      * was not implemented in this routine. This routine was *
C      * hard-wired for matrices of maximum order 4. *
C      *
C      *****

C      *****
C      *
C      * External references in this routine: *
C      *
C      * none *
C      *
C      *****

C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      * none *
C      *
C      *****

C      *****
C      *
C      * Local and dummy variables used in this routine: *
C      *
C      * A - Input matrix with NP by NP elements. After *
C      * processing it is replaced with its inverse. *
C      * BIG - The biggest element in the A matrix. *
C      * DUM - Temporary storage. *
C      * I - Generic index. *
C      * ICOL - Column. *
C      * INDXC - Used for bookkeeping on pivoting. *
C      * INDXR - Used for bookkeeping on pivoting. *
C      * IPIV - Used for bookkeeping on pivoting. *
C      * J - Generic index. *
C      * IROW - Row. *

```

```

C      *      K      - Generic index.          *
C      *      L      - Generic index.          *
C      *      LL     - Generic index.          *
C      *      N      - Order of the square A matrix. *
C      *      NMAX   - Dimension of bookkeeping arrays. *
C      *      PIVINV - Inverse of the current pivot element. *
C      *
C      *****

```

```

REAL      A      (4,4)
REAL      BIG
REAL      DUM
REAL      PIVINV

```

```

INTEGER   I
INTEGER   ICOL
INTEGER   INDXC (4)
INTEGER   INDXR (4)
INTEGER   IPIV (4)
INTEGER   J
INTEGER   IROW
INTEGER   K
INTEGER   L
INTEGER   LL
INTEGER   N

```

C Initialize the pivot matrix.

```

DO 100 J=1,N
100 IPIV(J) = 0

```

C This is the main loop over the columns to be reduced.

```

DO 260 I=1,N
BIG = 0.0

```

C This is the outer loop of the search for a pivot element.

```

DO 210 J=1,N

```

```

IF ( IPIV(J) .NE. 1 ) THEN

```

```

DO 200 K=1,N

```

```

IF ( IPIV(K) .EQ. 0 ) THEN

```

```

IF ( ABS( A(J,K) ) .GE. BIG ) THEN

```

```

BIG = ABS( A(J,K) )

```

```

IROW = J

```

```

ICOL = K

```

```

END IF

```

```

ELSE IF ( IPIV(K) .GT. 1 ) THEN

```

```

PRINT *, ' '

```

```

PRINT *, ' >>> The input matrix to GAUSSJ is '

```

```
& PRINT *, 'singular. <<<'

```

```
PRINT *, ' '

```

```
& PRINT *, 'STRAP terminated abnormally due to the '

```

```
& PRINT *, 'the error listed above.'

```

```

STOP

```

```

END IF

```

```

200 CONTINUE

```

```

END IF

```

```

210 CONTINUE

```

```
IPIV(ICOL) = IPIV(ICOL) + 1
```

```
C      We now have the pivot element, so we interchange rows.  If
C      needed, to put the pivot element on the diagonal.  The col-
C      umns are not physically interchanged, only relabeled:
C      INDXC(I), the column of the Ith pivotal element, is the Ith
C      column that is reduced, while INDXR(I) is the row in which
C      that pivot element was originally located.  If INDXR(I) is
C      not equal to INDXC(I), there is an implied column inter-
C      change.  With this form of bookkeeping, the solution Bs will
C      end up in the correct order, and the inverse matrix will be
C      scrambled by columns.
```

```
IF ( IROW .NE. ICOL ) THEN
```

```
DO 220 L=1,N
```

```
    DUM      = A(IROW,L)
    A(IROW,L) = A(ICOL,L)
    A(ICOL,L) = DUM
```

```
220    CONTINUE
```

```
END IF
```

```
C      We are now ready to divide the pivot row by the pivot
C      element, located at (IROW,ICOL).
```

```
INDXR(I) = IROW
```

```
INDXC(I) = ICOL
```

```
IF ( A(ICOL,ICOL) .EQ. 0.0 ) THEN
```

```
    PRINT *, ' '
    PRINT *, ' >>> The input matrix to GAUSSJ is '
&    PRINT *, 'singular. <<<'
    PRINT *, ' '
&    PRINT *, 'STRAP terminated abnormally due to the '
    PRINT *, 'the error listed above.'
```

```
STOP
```

```
END IF
```

```
PIVINV = 1.0/A(ICOL,ICOL)
```

```
A(ICOL,ICOL) = 1.0
```

```
230 DO 230 L=1,N
    A(ICOL,L) = PIVINV*A(ICOL,L)
```

```
C      Reduce all the rows except the pivot one.
```

```
DO 250 LL=1,N
```

```
IF ( LL .NE. ICOL ) THEN
```

```
    DUM      = A(LL,ICOL)
    A(LL,ICOL) = 0.0
```

```
240 DO 240 L=1,N
    A(LL,L) = A(LL,L) - DUM*A(ICOL,L)
```

```
END IF
```

```
250    CONTINUE
```

```
260 CONTINUE
```

```
C      Unscramble the solution in view of the column interchanges.
C      We do this by interchanging pairs of columns in the reverse
C      order that the permutation was built up.
```

```
DO 310 L=N,1,-1
```

```
IF ( INDXR(L) .NE. INDXC(L) ) THEN
```

```
DO 300 K=1,N
```

```
    DUM      = A(K,INDXR(L))  
    A(K,INDXR(L)) = A(K,INDXC(L))  
    A(K,INDXC(L)) = DUM
```

```
300    CONTINUE
```

```
END IF
```

```
310 CONTINUE
```

```
RETURN
```

```
END
```

```
SUBROUTINE GETFIL ( RESFIL )
```

```
C *****  
C *  
C * Subroutine GETFIL gets the name of the results file and *  
C * opens it. If the file already exists, the user is *  
C * asked if it should be overwritten. *  
C *  
C *****  
  
C *****  
C *  
C * External references in this routine: *  
C *  
C * LNTH - Returns the length of a string. *  
C *  
C *****  
  
C *****  
C *  
C * Named COMMON blocks used in this routine: *  
C *  
C * none *  
C *  
C *****  
  
C *****  
C *  
C * Local and dummy variables used in this routine: *  
C *  
C * ANS - Answer to user question. *  
C * EXISTS - Flag that indicates whether or not a file *  
C * already exists. *  
C * ISOPEN - Flag that indicates whether or not unit 4 *  
C * is already active. *  
C * NEDOPN - Flag that indicates whether or not we need *  
C * to open the results file. *  
C * RESFIL - Name of file to store results. *  
C * TEMP - Temporary variable used to store the name *  
C * of the results file. *  
C *  
C *****
```

```
INTEGER      LNTH  
  
LOGICAL      EXISTS  
LOGICAL      ISOPEN  
LOGICAL      NEDOPN  
  
CHARACTER*1  ANS  
CHARACTER*(*) RESFIL  
CHARACTER*50 TEMP
```

```
1000 FORMAT ( ' Enter name of results file [=', A, '] > ' )  
1100 FORMAT ( A )  
1200 FORMAT ( '1' )
```

```

C      Get name of results file.
100 PRINT 1000, RESFIL(1:LNTH(RESFIL))
   READ 1100, TEMP
      IF ( TEMP .EQ. ' ' ) THEN

C      The user hit just an <Enter>. Use the old name.
      TEMP = RESFIL
ELSE

C      Convert file name to upper case if a name was typed in.
      CALL CAPS ( TEMP )
END IF

C      Check to see if unit 4 is already attached.
INQUIRE ( 4 , OPENED=ISOPEN )
IF ( ISOPEN ) THEN
      IF ( TEMP .EQ. RESFIL ) THEN

C      Use the same file that's already open. Eject a page.
      WRITE (4,1200)
      NEDOPN = .FALSE.
ELSE

C      Use a different file. Close old one.
      CLOSE ( 4 )
      NEDOPN = .TRUE.
END IF
ELSE

C      No open file yet. We need to open one.
      NEDOPN = .TRUE.
END IF
IF ( NEDOPN ) THEN

C      We need to open the new results file. Does it already
C      exist?
      INQUIRE ( FILE=TEMP , EXIST=EXISTS )
      IF ( EXISTS ) THEN

C      File already exists. Write over it?
110 PRINT *, ' >>> File already exists <<<'
      PRINT *, ' '
      PRINT *, 'Do you want to overwrite the old data?'
      &      , ' (Y,N) > '
      READ 1100, ANS
      PRINT *, ANS
      IF ( ( ANS .EQ. 'N' ) .OR. ( ANS .EQ. 'n' ) ) GO TO 100

```

```

      IF ( ( ANS .NE. 'Y' ) .AND. ( ANS .NE. 'y' ) ) GO TO 110
END IF

```

```

C      Open the results file.

```

```

      OPEN ( 4 , FILE=TEMP , STATUS='UNKNOWN'
&          , CARRIAGECONTROL='FORTRAN' )

```

```

END IF

```

```

C      Save name of results file.

```

```

RESFIL = TEMP

```

```

RETURN

```

```

END

```

```

SUBROUTINE INDUCD ( INBORD , NPTS , BLDANG )

```

```

C      *****
C      *
C      * Subroutine INDUCD computes the induced velocities along *
C      * the blade. These are then used to compute the aerody- *
C      * namic forces. The values computed and returned by this *
C      * routine are all in non-dimensional form. That is, they *
C      * are divided by R, OMEGA or R*OMEGA where necessary. *
C      *
C      *****

```

```

C      *****
C      *
C      * External references in this routine: *
C      *
C      * none *
C      *
C      *****

```

```

C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      * AERO1 - Holds coefficients related to aerodynamic *
C      * loads calculations such as ClAlpha, CdZero, *
C      * etc. *
C      * BLADE - Holds blade property values such as stiff- *
C      * ness and mass distributions. *
C      * CONST - Turbine and other constants used in load *
C      * calculations. *
C      * POSITN - Holds parameters related to blade position *
C      * such as PHI, PSI, etc. *
C      * TURBN - Holds turbine parameters such as number of *
C      * blades, rotor speed, etc. *
C      * VINDUC - Holds induced velocity components. *
C      * WIND - Holds wind shear and tower shadow param- *
C      * eters. *
C      * WNDVEL - Holds values used in wind shear and tower *
C      * shadow computations. *
C      *
C      *****

```

```

C      *****
C      *
C      * Local and dummy variables used in this routine: *
C      *
C      * A2BVL - Temporary storage. *
C      * AA - Temporary storage. *
C      * ALMP - Temporary storage. *
C      * AM2BV - Temporary storage. *
C      * AMBV - Temporary storage. *
C      * APR - Temporary storage. *
C      * APRIME - Temporary storage. *
C      * ASTAR - Temporary storage. *
C      * BB - Temporary storage. *
C      * BLDANG - Blade azimuth position used in the teeter- *

```



```

C      *      ing rotor option.      *
C      *      BPRIME - Temporary storage.      *
C      *      BVALUE - Temporary storage.      *
C      *      CC      - Temporary storage.      *
C      *      CPRIME  Temporary storage.      *
C      *      CSS     - Temporary storage.      *
C      *      DELVSH - Temporary storage.      *
C      *      F1      - Temporary storage.      *
C      *      F2      - Temporary storage.      *
C      *      F3      - Temporary storage.      *
C      *      F4      - Temporary storage.      *
C      *      I       - Generic index.          *
C      *      INBORD - The number of blade stations that are not      *
C      *      on the airfoil section. INBORD is used in      *
C      *      other routines to differentiate between      *
C      *      sections of the blade with and without an      *
C      *      airfoil section.                    *
C      *      ISUBP  - Temporary storage.          *
C      *      ISUBW  - Temporary storage.          *
C      *      J       - Generic index.            *
C      *      LAMDAH - Inflow ratio.              *
C      *      MPRIME - Temporary storage.          *
C      *      NPTS   - Number of points along the blade used to      *
C      *      perform Simpson's integration for calculat-      *
C      *      ing the moments and forces at the blade      *
C      *      root. (passed from STRAP1)          *
C      *      P       - Temporary storage.          *
C      *      RBAR   - Temporary storage.          *
C      *      RDCL1  Temporary storage.          *
C      *      RDCL2  - Temporary storage.          *
C      *      STEADY - Flag to indicate steady flow.      *
C      *      STEP   - Distance between points along the blade.      *
C      *      TAU    - Temporary storage.          *
C      *      TAU4   - Temporary storage.          *
C      *      TAUZL  - Temporary storage.          *
C      *      THETZL - Temporary storage.          *
C      *      TLAM   - Temporary storage.          *
C      *      TSUBZR - Temporary storage.          *
C      *      VINDPC - Temporary storage.          *
C      *      VINDS  - Temporary storage.          *
C      *      Z       - Blade position pointer.      *
C      *      *      *      *      *      *      *
C      *      *****

```

```

REAL      A2BVL
REAL      AA
REAL      ALMP
REAL      AM2BV
REAL      AMBV
REAL      APR
REAL      APRIME
REAL      ASTAR
REAL      BB
REAL      BLDANG
REAL      BPRIME
REAL      BVALUE
REAL      CC
REAL      CPRIME
REAL      CSS      (5)
REAL      DELVSH
REAL      F1
REAL      F2
REAL      F3
REAL      F4
REAL      ISUBP
REAL      ISUBW
REAL      LAMDAH
REAL      MPRIME
REAL      P
REAL      RBAR
REAL      RDCL1
REAL      RDCL2
REAL      STEP
REAL      TAU
REAL      TAU4
REAL      TAUZL
REAL      THETZL
REAL      TLAM

```

```

REAL          TSUBZR
REAL          VINDPC
REAL          VINDS
REAL          Z

INTEGER       I
INTEGER       INBORD
INTEGER       J
INTEGER       NPTS

LOGICAL       STEADY

INCLUDE       'C:INCLUDE\AER01.INC'
INCLUDE       'C:INCLUDE\BLADE2.INC'
INCLUDE       'C:INCLUDE\CONST2.INC'
INCLUDE       'C:INCLUDE\POSITN.INC'
INCLUDE       'C:INCLUDE\TURBN.INC'
INCLUDE       'C:INCLUDE\VINDUC.INC'
INCLUDE       'C:INCLUDE\WIND2.INC'
INCLUDE       'C:INCLUDE\WNDVEL.INC'

2000 FORMAT ( / ' >>> Error in steady induced velocity. <<<'
&           / ' >>> Negative value in SQRT was set to 0. <<<'
&           / ' >>> Blade angle = ' , F6.2
&           / ' >>> Blade station = ' , ' , '
&           / ' >>> Blade station = ' , ' , ' 13.2
&           / )

```

C Initialize constants.

```

INBORD = 1
LAMDAH = HUBVEL/( RR*OMEGA )
STEP = BLTIP/( NPTS - 1 )

IF ( TSHADW .EQ. 0.0 ) THEN
  TSUBZR = 0.0
ELSE
  TSUBZR = TSUB0
END IF

```

```

DO 100 I=1,5
100 CSS(I) = COS( I*BLDANG )

```

C Move blade station pointer to inboard start of airfoil section
C on blade.

```

Z = 0.0

110 IF ( Z .LT. BLSHNK ) THEN
  Z = Z + STEP
  INBORD = INBORD +
  GO TO 110
END IF

```

C Initialize internal values.

```

ASTAR = 0.20
MPRIME = 2.40
BVALUE = 0.16
Z = 0.0

ALMP = LAMDAH*MPRIME
AMBV = MPRIME + BVALUE

```

```

AM2BV = LAMDAH*( MPRIME + 2.0*BVALUE )
APR = 1.0/( 8.0*PI*RR )
A2BVL = 2.0*BVALUE*LAMDAH
TLAM = LAMDAH*LAMDAH

```

C Loop through blade stations.

```
DO 220 I=1,NPTS
```

```
IF ( I .LT. INBORD ) THEN
```

```

VINDR(I) = 0.0
VINDO(I) = 0.0
DVIND(I) = 0.0

```

```
ELSE
```

```

STEADY = .TRUE.
RBAR = ( Z + HUBRAD. )/RR
THETZL = THETAP - THETAO(I)
TAU = APR*( CLALFA(I)*CHORD(I)*NBLADS )
TAUTZL = TAU*THETZL

```

```

AA = RBAR - 0.5*TAUTZL
BB = RBAR*TAU - LAMDAH*( RBAR
& - ( WSHR(0,I) - TSUBZR )*( RBAR - TAUTZL ) )
CC = -RBAR*( RBAR*TAUTZL
& + LAMDAH*( WSHR(0,I) - TSUBZR )*( LAMDAH - TAU ) )

```

C Compute the induced velocity. First check to see if the
C coefficient of the squared term is zero. If it is, then
C the equation is linear. If the radical of the root is
C less than zero, then the momentum solution is invalid,
C set the radical to zero and go on.

```
IF ( AA .NE. 0.0 ) THEN
```

```

RDCL1 = BB**2 - 4.0*AA*CC
IF ( RDCL1 .LT. 0.0 ) RDCL1 = 0.0
VINDR(I) = 0.5*( -BB + SQRT( RDCL1 ) )/AA

```

```
ELSE
```

```
VINDR(I) = -CC/BB
```

```
END IF
```

C Check to see if the induced condition is steady or
C turbulent. If it is turbulent, compute turbulent
C intermediate values and recompute the induced velocity.

```

& IF ( ( 1.0-VINDR(I)/LAMDAH .GT. ASTAR ) .OR.
& ( RDCL1 .EQ. 0.0 ) ) THEN

```

```

STEADY = .FALSE.
TAU4 = 4.0*TAU
APRIME = 2.0*TAUTZL
BPRIME = -RBAR*( ALMP + TAU4 )
& - LAMDAH*( WSHR(0,I) - TSUBZR )
& *( RBAR*MPRIME - 4*TAUTZL )
CPRIME = RBAR*( 4*RBAR*TAUTZL + AMBV*TLAM
& + LAMDAH*( WSHR(0,I) - TSUBZR )
& *( AM2BV - TAU4 ) )

```

C Compute the induced velocity. Use same logic as
C for steady flow.

```
IF ( APRIME .NE. 0.0 ) THEN
```

```

RDCL2 = BPRIME**2 - 4*APRIME*CPRIME
IF ( RDCL2 .LT. 0.0 ) THEN

```

```

        PRINT 2000, BLDANG , I
        RDCL2 = 0.0
    END IF
    VINDR(I) = -0.5*( BPRIME + SQRT( RDCL2 ) )/APRIME
ELSE
    VINDR(I) = -CPRIME/BPRIME
END IF
END IF
VINDO(I) = LAMDAH - VINDR(I)
C      Calculate the intermediate components for forming induced
C      velocity terms.
F1 = TAU*VINDR(I)*( VINDR(I) - 2*RBAR*THETZL )
F2 = TAU*( RBAR - THETZL*VINDR(I) )
F3 = RBAR*( VINDR(I) - VINDO(I) ) + F2
IF ( STEADY ) THEN
    ISUBP = F1/F3
    ISUBW = LAMDAH*( F2 - RBAR*VINDO(I) )/F3
ELSE
    F4 = RBAR*ALMP + 4*F2
    ISUBP = 4*F1/F4
    ISUBW = LAMDAH*( 4*F2 - RBAR*( A2BVL
&                                     + MPRIME*VINDO(I) ) )/F4
END IF
C      Form induced velocity components of order epsilon, etc.
VINDC(1) = -PHI(0)*ISUBP + WSHR(1,I)*ISUBW
VINDS = -CHI*ISUBP
200 DO 200 J=2,5
    VINDC(J) = WSHR(J,I)*ISUBW
C      Compute the tower shadow effect if in the tower shadow
C      region.
IF ( TSHADW .NE. 0.0 ) THEN
    VINDPC = TSUBP*ISUBW
    P = KSHADW*PI/PSIZER
    DELVSH = VINDPC*COS( P*( BLDANG - PSISHD ) )
ELSE
    VINDPC = 0.0
    DELVSH = 0.0
END IF
C      Compute the delta-V-induced term. i.e., the sum of the
C      order epsilon components.
DVIND(I) = VINDS*SIN( BLDANG ) - DELVSH
210 DO 210 J=1,5
    DVIND(I) = DVIND(I) + VINDC(J) * CSS( J )
END IF

```

```
C      Increment the blade position pointer and repeat the induced
C      velocity procedure out to the blade tip.
```

```
      Z = Z + STEP
```

```
220 CONTINUE
```

```
RETURN
END
SUBROUTINE INVERT ( AMAT , AINV , N )
```

```
C      *****
C      *
C      * Subroutine INVERT is a transparent interface between *
C      * the calling routine, SOLVE, and GAUSSJ, the routine *
C      * that performs the actual inversion. The incoming ma- *
C      * trix is unaffected by the inversion process. *
C      *
C      *****
```

```
C      *****
C      *
C      * External references in this routine: *
C      *
C      * GAUSSJ - Matrix inversion using Gauss-Jordan elimi- *
C      * nation with full pivoting. *
C      *
C      *****
```

```
C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      * none *
C      *
C      *****
```

```
C      *****
C      *
C      * Local and dummy variables used in this routine: *
C      *
C      * AINV - The resulting inverse matrix. *
C      * AMAT - The incoming matrix to be inverted. *
C      * I - Generic index. *
C      * J - Generic index. *
C      * N The order of the incoming square matrix. *
C      *
C      *****
```

```
REAL AINV (4,4)
REAL AMAT (4,4)
```

```
INTEGER I
INTEGER J
INTEGER N
```

```
C      Load the incoming matrix into the inverse matrix AINV.
```

```
      DO 20 I=1,N
      DO 10 J=1,N
      AINV(I,J) = AMAT(I,J)
10 CONTINUE
20 CONTINUE
```

```
C      Invert the matrix.
```

```
CALL GAUSSJ ( AINV , N )
```

```
RETURN
```

END
SUBROUTINE LIMITS

```

C *****
C *
C * Subroutine LIMITS inputs the run limit values through *
C * interactive dialog. A set of default values are init- *
C * ialized in the BLOCK DATA subprogram. *
C *
C *****
C
C *****
C *
C * External references in this routine: *
C *
C * none *
C *
C *****
C
C *****
C *
C * Named COMMON blocks used in this routine: *
C *
C * LIMITC - Holds values used in the LIMITS routine. *
C *
C *****
C
C *****
C *
C * Local and dummy variables used in this routine: *
C *
C * ANS Answer to user question. *
C * ERROR - IOSTAT error value on OPEN statement. Used *
C * to check for existing files. *
C * ICASE - Number of variable to change. *
C * P1 - Temporary variable used for testing to see *
C * if PRINT1 is a multiple of STEPMX. *
C * P2 - Temporary variable used for testing to see *
C * if PRINT2 is a multiple of STEPMX. *
C * TEMP - Temporary variable used to store the name *
C * of the results file. *
C *
C *****

```

```

REAL P1
REAL P2

INTEGER ICASE

CHARACTER*1 ANS

INCLUDE 'C:INCLUDE\LIMITC.INC'

```

```

1000 FORMAT ( // ' The current values of the run parameters are:' / )
1100 FORMAT ( 1X , A , F8.3 , A )
1200 FORMAT ( 1X , A , I4 , 4X , A )
2000 FORMAT ( A )

```

```

C Print out the current limit values.

100 PRINT 1000

PRINT 1100, ' 1 STEPMX = ' , STEPMX
& ' degrees Maximum Step Size'

PRINT 1100, ' 2 STEP MN = ' , STEP MN
& ' degrees Minimum Step Size'

PRINT 1100, ' 3 PRINT1 = ' , PRINT1
& ' degrees Printout Interval in Region 1'

PRINT 1100, ' 4 PRINT2 = ' , PRINT2
& ' degrees Printout Interval in Region 2'

```

```

PRINT 1100, ' 5 BEGIN2 = ', BEGIN2
&          , ' degrees Beginning of Print Region 2'

PRINT 1100, ' 6 END2   = ', END2
&          , ' degrees End of Print Region 2'

PRINT 1100, ' 7 EUERR  = ', EUERR
&          , ' percent Max value of Euler error function'

PRINT 1100, ' 8 TRMERR = ', TRMERR
&          , ' percent Convergence Criterion for Trim Solution'

PRINT 1200, ' 9 NYAW   = ', NYAW
&          , '          No. of Disk Revolutions for Yawing Soln.'

IF ( TRACEF ) THEN
  PRINT *, '10 TRACEF = .TRUE.          Trace Flag'
ELSE
  PRINT *, '10 TRACEF = .FALSE.         Trace Flag'
END IF

```

C Ask user for changes to present values.

```

200 PRINT *, ' '
PRINT *, 'Do you want to change any of these values? (Y,N) > '
READ 2000, ANS
PRINT *, ANS

IF ( ( ANS .EQ. 'Y' ) .OR. ( ANS .EQ. 'y' ) ) THEN

210 PRINT *, ' '
PRINT *, 'Enter number of variable you wish to change > '
READ *, ICASE
PRINT *, ICASE

IF ( ( ICASE .LE. 0 ) .OR. ( ICASE .GT. 10 ) ) THEN

  PRINT *, ' >>> Invalid response. Please try again. <<<'
  GO TO 210

ELSE

```

C Change one of the variables.

```

GO TO (300,310,320,330,340,350,360,370,380,390), ICASE

300 PRINT *, 'Enter new REAL value for STEPMX > '
READ *, STEPMX
PRINT *, STEPMX
GO TO 100

310 PRINT *, 'Enter new REAL value for STEPMM > '
READ *, STEPMM
PRINT *, STEPMM
GO TO 100

320 PRINT *, 'Enter new REAL value for PRINT1 > '
READ *, PRINT1
PRINT *, PRINT1
GO TO 100

330 PRINT *, 'Enter new REAL value for PRINT2 > '
READ *, PRINT2
PRINT *, PRINT2
GO TO 100

340 PRINT *, 'Enter new REAL value for BEGIN2 > '
READ *, BEGIN2
PRINT *, BEGIN2
GO TO 100

350 PRINT *, 'Enter new REAL value for END2 > '
READ *, END2
PRINT *, END2

```

```

        GO TO 100
360    PRINT *, 'Enter new REAL value for EUERR > '
        READ *, EUERR
        PRINT *, EUERR
        GO TO 100

370    PRINT *, 'Enter new REAL value for TRMERR > '
        READ *, TRMERR
        PRINT *, TRMERR
        GO TO 100

380    PRINT *, 'Enter new INTEGER value for NYAW > '
        READ *, NYAW
        PRINT *, NYAW
        GO TO 100

390    PRINT *, 'Enter new LOGICAL value for TRACEF (T,F) > '
        READ *, TRACEF
        PRINT *, TRACEF
        GO TO 100

    END IF

    ELSE IF ( ( ANS .NE. 'N' ) .AND. ( ANS .NE. 'n' )
&          & .AND. ( ANS .NE. ' ' ) ) THEN

        PRINT *, ' >>> Invalid response. Please try again. <<<'

        GO TO 200

    END IF

C      Check to make sure the values of PRINT1 and PRINT2 are integer
C      multiples of STEPMX. If they are not, print error message and
C      go back to the change values section.

P1 = STEPMX*INT( PRINT1/STPEMX + 0.00001 )
P2 = STEPMX*INT( PRINT2/STPEMX + 0.00001 )

IF ( ABS( P1 - PRINT1 ) .GT. 0.001 ) THEN

    PRINT *, ' '
    PRINT *, ' >>> PRINT1 must be a multiple of STEPMX <<<'

    GO TO 100

END IF

IF ( ABS( P2 - PRINT2 ) .GT. 0.001 ) THEN

    PRINT *, ' '
    PRINT *, ' >>> PRINT2 must be a multiple of STEPMX <<<'

    GO TO 100

END IF

RETURN
END
FUNCTION LNTH ( STRING )

```

```

C      *****
C      *
C      *   Function LNTH returns the length of a character string. *
C      *   When using the Lahey F77L compiler, the intrinsic func- *
C      *   tion NBLANK can be used as we do here. This function *
C      *   was supplied to make conversion to other compilers eas- *
C      *   ier. *
C      *
C      *****

C      *****
C      *
C      *   Named COMMON blocks used in this routine: *

```



```

C      *
C      *      none
C      *
C      *****
C
C      *****
C      *
C      *      Variables used in this routine:
C      *
C      *      IC      - Generic index.
C      *      LENGTH - The declared length of STRING.
C      *      LNTH   - The location of the last nonblank character
C      *                in STRING.
C      *      STRING - A character string.
C      *
C      *****
C
C      INTEGER      IC
C      INTEGER      LENGTH
C      INTEGER      LNTH
C
C      CHARACTER*(*) STRING
C
C      Get the declared length of STRING using the FORTRAN 77 intrinsic
C      function LEN.
C
C      LENGTH = LEN( STRING )
C
C      Find the location of the last nonblank character in STRING.
C
C      DO 100 IC=LENGTH,1,-1
C
C          LNTH = IC
C
C          IF ( STRING(IC:IC) .NE. ' ' ) GO TO 200
C
C      100 CONTINUE
C
C      STRING is all blanks.
C
C      LNTH = 0
C
C      200 RETURN
C      END
C      SUBROUTINE LODOUT ( NCALLS )
C
C      *****
C      *
C      *      Subroutine LODOUT is used to compute blade loads and
C      *      print them out.
C      *
C      *****
C
C      *****
C      *
C      *      External references in this routine:
C      *
C      *      AERO    Calculates the aerodynamic forces on the
C      *                blade.
C      *      CONVRT - Performs units conversions.
C      *      LNTH   - Returns the length of a string.
C      *      NXTPHI - Calculates the next values of the yaw vari-
C      *                ables.
C      *      SIMPSN - Composite Simpson's integration.
C      *      TRPZOD - Composite trapezoidal integration.
C      *
C      *****
C
C      *****
C      *
C      *      Named COMMON blocks used in this routine:
C      *
C      *

```

```

C      *      AERO1 - Holds coefficients related to aerodynamic *
C      *      loads calculations such as ClAlpha, CdZero, *
C      *      etc. *
C      *      AIRFRC - Holds values used in aerodynamic calcula- *
C      *      tions. *
C      *      BLADE - Holds blade property values such as stiff- *
C      *      ness and mass distributions. *
C      *      CONST - Turbine and other constants used in load *
C      *      calculations. *
C      *      FORMS - Holds blade deflections. *
C      *      LIMITC - Holds values used in the LIMITS routine. *
C      *      LITERL - Holds data set titles. *
C      *      LODVAL - Holds values used to compute blade loads. *
C      *      POSITN - Holds parameters related to blade position *
C      *      such as PHI, PSI, etc. *
C      *      SARAYS - Holds new and old values for the general- *
C      *      ized coordinates. *
C      *      SHAPE - Holds blade coordinate shape functions. *
C      *      TENSIN - Holds tension component integrals. *
C      *      TURBN - Holds turbine parameters such as number of *
C      *      blades, rotor speed, etc. *
C      *      WIND - Holds wind shear and tower shadow parame- *
C      *      ters. *
C      *
C      *****

```

```

C      *****
C      *
C      *      Local and dummy variables used in this routine: *
C      *
C      *      AVE MOM - Mean flapwise bending moment at 0.2*R. *
C      *      BLDANG - Blade azimuth position used in the teeter- *
C      *      ing rotor . *
C      *      COEF1 - Complicated term. *
C      *      COEF2 - Complicated term. *
C      *      COEF3 - Complicated term. *
C      *      COEF4 - Complicated term. *
C      *      COEF5 - Complicated term. *
C      *      COEF6 - Complicated term. *
C      *      COEF7 - Complicated term. *
C      *      COEF8 - Complicated term. *
C      *      CPSI - Cosine of the blade angle. *
C      *      DMSBAC - Complicated term. *
C      *      ECENGR - Integral of ECNTFN. *
C      *      EDGEFN - Edgewise shear force distribution. *
C      *      EINTGR - Integral of EDGEFN. *
C      *      FACTR1 - Complicated term. *
C      *      FINTGR - Integral of FLAPFN. *
C      *      FLAPFN - Flapwise shear force distribution. *
C      *      FPS2KW - Factor for converting Ft-Lbs/sec to KW. *
C      *      I - Generic index. *
C      *      IBEGIN - Azimuth position for beginning of print re- *
C      *      gion 1. *
C      *      IEND - Azimuth position for end of print region 1. *
C      *      ILABEL - Strings used for printout. *
C      *      IPRNT1 - Printout interval for results in print re- *
C      *      gion 1. *
C      *      IPRNT2 - Printout interval for results in print re- *
C      *      gion 2. *
C      *      IPSI - Integerized disk station azimuth positions. *
C      *      IPSIST - The next STEPMX station in integer degrees. *
C      *      IPT - Data point index. *
C      *      IST - Station index. *
C      *      IVALUE - Temporary storage. *
C      *      J - Generic index. *
C      *      LABEL1 - String to hold variable names. *
C      *      LABEL2 - String to hold units for variables. *
C      *      LABEL3 - String to hold variable descriptions. *
C      *      LABEL4 - String. *
C      *      LABEL5 - String. *
C      *      LL - Generic index. *
C      *      M - Generic index. *
C      *      NCALLS - Number of calls to LODOUT. *
C      *      NOWPS1 - Current value of azimuth position. *
C      *      NPTS - Number of points along the blade used to *
C      *      perform Simpson's integration for calculat- *
C      *      ing the moments and forces at the blade *
C      *      root. (passed from STRAP1) *
C      *

```

```

C      *      NSHP   - Counter on DO loops for the coordinate *
C      *      shape function. *
C      *      NSTN   - Number of stations around rotor disk. *
C      *      NTRQGS - Number of values used to compute TRQSUM. *
C      *      NTOT   - NUmber of values used to compute AVE MOM. *
C      *      OMGASQ - Omega^2. *
C      *      POWER  - Power output of wind turbine. *
C      *      PRDCT0 - Temporary storage. *
C      *      PRDCT1 - Temporary storage. *
C      *      PRDCT2 - Temporary storage. *
C      *      PRODC  - Temporary storage. *
C      *      PRODD  - Temporary storage. *
C      *      PXAERO - Aerodynamic loads in the edgewise direc- *
C      *      tion. *
C      *      PYAERO - Aerodynamic loads in the flapwise direc- *
C      *      tion. *
C      *      QZAERO - Integral of QZFCN. *
C      *      QZFCN  - Aerodynamic moment per unit length about *
C      *      the elastic axis. *
C      *      RDIST  - Distance out along blade. *
C      *      RESLTS - Temporary storage. *
C      *      RPO    - Phi in degrees. *
C      *      RP1    - Phi-dot in degrees. *
C      *      RP2    - Phi-dot-dot in degrees. *
C      *      RPHI   - Phi and its derivatives at this disk sta- *
C      *      tion. *
C      *      RTIME  - Time values in yaw solution. *
C      *      SINTGR - The integral of VXFCN. *
C      *      SPSI   - Sine of the blade angle. *
C      *      STEP1  - Distance between stations along the blade. *
C      *      TIMNOW - Current time. *
C      *      TINTGR - Integral of TSNFCN. *
C      *      TODEGS - Flag used to tell CONVRT to convert 'free' *
C      *      variables to degrees. *
C      *      TORADS - Flag used to tell CONVRT to convert 'free' *
C      *      variables to radians. *
C      *      TORQFN - The torque produced by an incremental aero- *
C      *      dynamic force. *
C      *      TRQSUM - Total torque. Integral of TORQFN. *
C      *      TSNFCN - Product of the tension force and the slope *
C      *      at a blade station. *
C      *      VALUE  - Temporary storage. *
C      *      VXFCN  - Product of the edgewise shear force and the *
C      *      blade slope at a blade station. *
C      *      *****

```

```

REAL      AVE MOM
REAL      BLDANG
REAL      COEF1
REAL      COEF2
REAL      COEF3
REAL      COEF4
REAL      COEF5
REAL      COEF6
REAL      COEF7
REAL      COEF8
REAL      CPSI
REAL      DAMCOF
REAL      DMSBAC
REAL      ECENGR
REAL      EDGEFN (21)
REAL      EINTGR
REAL      FACTR1
REAL      FINTGR
REAL      FLAPFN (21)
REAL      FDAMP (360)
REAL      FPS2KW
REAL      OMGASQ
REAL      POWER
REAL      PRDCT0
REAL      PRDCT1
REAL      PRDCT2
REAL      PRODC (21)
REAL      PRODD
REAL      PXAERO (21)
REAL      PYAERO (21)

```

REAL	QZFCN (21)
REAL	RDIST
REAL	TEETER (360)
REAL	RPO
REAL	RP1
REAL	RP2
REAL	RPHI (360,0:2)
REAL	RTIME (360)
REAL	SIMPSN
REAL	SINTGR
REAL	SPSI
REAL	STEP1
REAL	TIMNOW
REAL	TINTGR
REAL	TORQFN (21)
REAL	TRPZOD
REAL	TRQSUM
REAL	TSNFCN (21)
REAL	VALUE (30)
REAL	VXFCN (21)
REAL	ACOF (0:10,21)
REAL	BCOF (0:10,21)
INTEGER	H
INTEGER	I
INTEGER	IPRNT1
INTEGER	IPSI (360)
INTEGER	IPSIST
INTEGER	IPT
INTEGER	IST
INTEGER	IVALUE (4)
INTEGER	J
INTEGER	LL
INTEGER	LNTH
INTEGER	M
INTEGER	MITEM
INTEGER	NHARM
INTEGER	NCALLS
INTEGER	NOWPSI
INTEGER	NPTS
INTEGER	NSHP
INTEGER	NSTN
INTEGER	NTORQS
INTEGER	NTOT
CHARACTER*10	I LABEL (4)
CHARACTER*9	LABEL1 (30)
CHARACTER*11	LABEL2 (30)
CHARACTER*45	LABEL3 (9)
CHARACTER*50	LABEL6 (9)
CHARACTER*19	LABEL4
CHARACTER*2	LABEL5
CHARACTER*1	ANS1
LOGI CAL	TODEGS
LOGI CAL	TORADS
INCLUDE	'C:\INCLUDE\AERO1.INC'
INCLUDE	'C:\INCLUDE\AIRFRC.INC'
INCLUDE	'C:\INCLUDE\BLADE2.INC'
INCLUDE	'C:\INCLUDE\CONST2.INC'
INCLUDE	'C:\INCLUDE\FORMS.INC'
INCLUDE	'C:\INCLUDE\LIMITC.INC'
INCLUDE	'C:\INCLUDE\LITERL.INC'
INCLUDE	'C:\INCLUDE\LODVAL.INC'
INCLUDE	'C:\INCLUDE\POSITN.INC'
INCLUDE	'C:\INCLUDE\SARAYS.INC'

```

INCLUDE      'C:INCLUDE\SHAPE.INC'
INCLUDE      'C:INCLUDE\TENSIN2.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\RESLTS.INC'
INCLUDE      'C:INCLUDE\WIND2.INC'
INCLUDE      'C:INCLUDE\SPRING2.INC'
INCLUDE      'C:INCLUDE\HUBPRP2.INC'

SAVE         I LABEL
SAVE         LABEL1
SAVE         LABEL2
SAVE         LABEL3
SAVE         NPTS
SAVE         TODEGS
SAVE         TORADS
SAVE         IPSI

DATA         FPS2KW / 0.0013558 /

DATA         I LABEL / 'KSHADW = ' , 'NBLADS = '
&           , 'NSHAPS = ' , 'NYAW = ' /

DATA         LABEL1
& / 'ALENTH = ' , 'ALPHA0 = ' , 'BETA0 = ' , 'BLSHNK = '
& , 'BLTIP = ' , 'CHI = ' , 'CSUBMA = ' , 'DRGFRM = '
& , 'EUERR = ' , 'HUBHT = ' , 'HUBRAD = ' , 'OMEGA = '
& , 'PHIAMP = ' , 'PHIOMG = ' , 'PHIO = ' , 'PSIZER = '
& , 'RHOAIR = ' , 'SHERXP = ' , 'STEPMX = ' , 'STEPMN = '
& , 'TSUBP = ' , 'TSUB0 = ' , 'THETAP = ' , 'THETAT = '
& , 'TRMERR = ' , 'VHUB = ' , 'DELT3 = ' , 'UNDSLGL = '
& , 'HUBMAS = ' , 'HUBDIS = ' /

DATA         LABEL2
& / 'feet ' , 'degrees ' , 'degrees ' , 'feet '
& , 'feet ' , 'degrees ' , 'degrees ' , 'feet '
& , 'percent ' , 'feet ' , 'feet ' , 'RPM '
& , 'degrees ' , 'degrees/sec' , 'degrees ' , 'degrees '
& , 'slugs/ft^3 ' , 'degrees ' , 'degrees '
& , 'percent ' , 'feet/second' , 'degrees ' , 'feet '
& , 'lb-s**2/ft ' , 'feet ' /

DATA         LABEL3
& / 'Blade section flap displacement ( feet ) '
& , 'Blade section flapwise slope ( feet/foot ) '
& , 'Blade section flap velocity ( feet/second ) '
& , 'Blade tension ( Lb ) '
& , 'Blade edgewise shear ( Lb ) '
& , 'Blade flapwise shear ( Lb ) '
& , 'Blade flapwise moment ( ft-Lbs ) '
& , 'Blade edgewise moment ( ft-Lbs ) '
& , 'Blade torsion ( ft-Lbs ) ' /

DATA         LABEL6
& / 'Blade flap-displacement expansion coefficients '
& , 'Blade flapwise slope expansion coefficients '
& , 'Blade flap-velocity expansion coefficients '
& , 'Blade tension force expansion coefficients '
& , 'Blade edgewise shear expansion coefficients '
& , 'Blade flapwise shear expansion coefficients '
& , 'Blade flapwise moment expansion coefficients '
& , 'Blade edgewise moment expansion coefficients '
& , 'Blade torsional moment expansion coefficients ' /

DATA         NPTS / 21 /

DATA         TODEGS / .TRUE. /

DATA         TORADS / .FALSE. /

```

```

4000 FORMAT ( /// 37X , 'Analysis of Wind Turbine Blade Loads' ,
&           / 39X , 'National Renewable Energy Lab' //// )

```

```

4100 FORMAT ( 3( 10X , A // ) )
4200 FORMAT ( 10X , A , ' Rotor'
&      ' Run Parameters, blade data and machine data used in'
&      ' this analysis: ' / )
4300 FORMAT ( 9X , 3( 1X , A , F7.3 , 1X , A , 5X , : ) // )
4400 FORMAT ( /// 9X , 4( 1X , A , I3 , 12X ) / )
4500 FORMAT ( '1' /// 1X , A , 65X , A , A / )
4600 FORMAT ( ' Psi Time Phi Phi-D Phi-DD' , 38X , 'X / R '
&      / 1X , 131( '--' )
&      / ' deg sec deg deg/s deg/s^2 .0 .1'
&      , ' .2 .3 .4 .5 .6'
&      , ' .7 .8 .9 1.0'
&      / 1X , 131( '--' ) / )
4700 FORMAT( 14 , F6.2 , F6.1 , F7.2 , F7.2 , 3X , 11(1X , 1PG8.2E1) )
4800 FORMAT ( // ' Average Flapwise Moment = ' , F9.1
&      , ' (ft-Lbs) at the 20% station' / )
5000 FORMAT ( '1' /// ' Mean Rotor Torque and Power for ' , 2A // )
5100 FORMAT ( 10X , 'Mean Rotor Torque = ' , F13.2 , ' ft-Lbs'
&      // 10X , 'Mean Rotor Power = ' , F13.2 , ' KW'
&      // 10X , 'No. of Data Points = ' , I13 )

5200 FORMAT ('1' /// 1X , A , 65X , A , A/)

5300 FORMAT (69X , 'X / R ' / 1X , 131( '--' ) / 131( '--' ) /
&      12X , '.0 .1 .2 .3 .4'
&      , '.5 .6 .7 .8 .9'
&      , '1.' /
&      1X , 131( '--' ) / 131( '--' ) / )

5400 FORMAT (1X , 'A' , I1 , '=' , 3X , 11(3X , 1PG8.2E1))
5500 FORMAT (1X , 'B' , I1 , '=' , 3X , 11(3X , 1PG8.2E1))
5600 FORMAT (1X , 131( '--' ))

```

C Initialize torque count and sum.

```

NTORQS = 0
TRQSUM = 0.0

```

C Fill the IPSI array with PSI disk station values. This section
C of code is executed only the first time this routine is called.

```

IF( NCALLS .EQ. 0 ) THEN

  IPRNT1 = PRINT1 * RAD2DG + 0.001
  NSTN = 1
  IPSIST = 0
  IPSI(NSTN) = IPSIST

```

```

120 IF ( IPSIST .LT. 360 ) THEN

  IPSIST = IPSIST + IPRNT1
  NSTN = NSTN + 1
  IPSI(NSTN) = IPSIST

  GO TO 120

ENDIF

```

```

ENDIF

IPSI(NSTN) = 360

```

C Initialize the time values in the RTIME array for the trim run.

```

IF ( ( NCALLS .EQ. 0 ) .OR. ( PHIAMP .EQ. 0.0 ) ) THEN

```

```

  DO 200 I=1, NSTN
200 RTIME(I) = 0.0

```

```

ELSE

```

C Calculate the time values around the disk stations for the

```

C      yaw runs. .
      RTIME(1) = 2*PI*( NCALLS - 1 )/OMEGA
      DO 210 IST=2,NSTN
210    RTIME(IST) = RTIME(1) + IPSI(IST)/( RAD2DG*OMEGA )
      END IF

C      Initialize outer computation loop used for going around the disk.
      PRODD = 0.5*RHOAIR*CSUBMA
      DO 370 IST=1,NSTN

C      Compute the PHI values at this disk azimuth.
      TIMNOW = RTIME(IST)
      CALL NXTPHI(TIMNOW)
      RPHI(IST,0) = PHI(0)
      RPHI(IST,1) = PHI(1)
      RPHI(IST,2) = PHI(2)

C      Compute constants used in computing the shear and moment
C      and tension values.
      OMGASQ = OMEGA**2
      PSI(1) = IPSI(IST)/RAD2DG
      CPSI   = COS( PSI(1) )
      SPSI   = SIN( PSI(1) )
      NOWPSI = IPSI(IST)

      TEETER(IST) = SNEW(1,0,NOWPSI)/(BLTIP+HUBRAD)
      THETAC = THETAP + TEETER(IST)*SDELTA3
      CTHC = COS(THETAC)
      STHC = SIN(THETAC)

      FACTR1 = OMGASQ*BETA0+
&      2*OMEGA*CPSI*PHI(1) + SPSI*PHI(2)

      COEF2 = GRAV*( -CHI*CTHC + STHC*SPSI + BETA0*CTHC*CPSI )
      COEF3 = GRAV*( CHI*STHC + CTHC*SPSI - BETA0*STHC*CPSI )
      COEF5 = GRAV*CPSI
      COEF8 = GRAV*SPSI*STHC

      COEF1 = 2.0*OMEGA*STHC
      COEF4 = OMGASQ*STHP*CTHC
      COEF6 = OMGASQ*CTHP*CTHC
      COEF7 = OMGASQ*STHP*STHC

C      Fill the V arrays.
C      V      = RESLTS( , ,1)
C      V-PRIME = RESLTS( , ,2)
C      V-DOT  = RESLTS( , ,3)

      DO 305 IPT=1,NPTS
      V1(0,IPT) = 0.
      V1(1,IPT) = 0.
305    CONTINUE

      DO 308 IPT = 1,NPTS
&      V1(0,IPT) = V1(0,IPT) + SHAPE(1,0,IPT)*
&      SNEW(1,0,NOWPSI)*CTHP
&      V1(1,IPT) = V1(1,IPT) + SHAPE(1,0,IPT)*
&      SNEW(1,1,NOWPSI)*CTHP
308    CONTINUE

      DO 310 IPT=1,NPTS
      RESLTS(IST,IPT,1) = 0.0

```

```
RESLTS(IST,IPT,2) = 0.0
RESLTS(IST,IPT,3) = 0.0
```

```
DO 300 NSHP=1,NSHAPS
```

```
      IF(NSHP .EQ. 1) THEN
&         RESLTS(IST,IPT,1) = RESLTS(IST,IPT,1)
&           + SHAPE(NSHP,0,IPT)*SNEW(NSHP,0,NOWPSI)*CTHP
&
&         RESLTS(IST,IPT,2) = RESLTS(IST,IPT,2)
&           + SHAPE(NSHP,1,IPT)*SNEW(NSHP,0,NOWPSI)*CTHP
&
&         RESLTS(IST,IPT,3) = RESLTS(IST,IPT,3)
&           + SHAPE(NSHP,0,IPT)*SNEW(NSHP,1,NOWPSI)*CTHP
&
&         ELSE
&
&         RESLTS(IST,IPT,1) = RESLTS(IST,IPT,1)
&           + SHAPE(NSHP,0,IPT)*SNEW(NSHP,0,NOWPSI)
&
&         RESLTS(IST,IPT,2) = RESLTS(IST,IPT,2)
&           + SHAPE(NSHP,1,IPT)*SNEW(NSHP,0,NOWPSI)
&
&         RESLTS(IST,IPT,3) = RESLTS(IST,IPT,3)
&           + SHAPE(NSHP,0,IPT)*SNEW(NSHP,1,NOWPSI)
&
&         ENDIF
```

```
300    CONTINUE
```

```
C      The values for WV(0,IPT) and WV(1,IPT) are needed for the
C      AERO and VREL routines.
```

```
      WV(0,IPT) = RESLTS(IST,IPT,1)
      WV(1,IPT) = RESLTS(IST,IPT,3)
```

```
310    CONTINUE
```

```
C      Compute the aerodynamic force components and the Qz compo-
C      nent form integrals and save them for use in computing the
C      shears and moments.
```

```
C      Compute loads on the teeter damper
```

```
      IF( ABS(TEETER(IST)) .GE. BETA1) THEN
&         DAMCOF= CDAMP
&
&         ELSE
&         DAMCOF = 0.
&
&         ENDIF
&
&         FDAMP(IST) = DAMCOF * SNEW(1,1,NOWPSI)/(HUBRAD+BLTIP)
```

```
      BLDANG = PSI(1)
```

```
      CALL AERO ( NPTS , BLDANG )
```

```
C      This section of code computes the rotor torque component
C      that produces the turbine shaft torque. The torque is
C      averaged over the disk to give average rotor torque for one
C      rotor revolution. This does not use weighted values. All
C      values have equal weight regardless of the azimuth spacing.
C      Thus these results will only be accurate when PRINT1 and
C      PRINT2 have the same value.
```

```
      STEP1 = BLTIP/( NPTS - 1 )
```

```
      IF ( NOWPSI .LT. 360 ) THEN
```

```
        DO 320 I=1,NPTS
```



```

RDIST      = HUBRAD + STEP1*( I - 1 )
TORQFN(I) = RDIST*( DAETA(I)*CTHC + DAZETA(I)*STHC )

320  CONTINUE

TRQSUM = TRQSUM + SIMPSN( 0 , BLTIP , NPTS , TORQFN )
NTORQS = NTORQS + 1

END IF

DO 330 IPT=1,NPTS

DMSBAC      = PRODD*CHORD(IPT)*WW(IPT)**2
QZFCN(IPT) = DMSBAC + ESUBAC(IPT)*DAZETA(IPT)

330  CONTINUE

DO 350 IPT=1,NPTS

PXAERO(IPT) = TRPZOD( IPT , BLTIP , DAETA , NPTS )
PYAERO(IPT) = TRPZOD( IPT , BLTIP , DAZETA , NPTS )

C      Compute the constants needed to compute tension
C      and shear.

PRDCT0 = 0.0
PRDCT1 = 0.0
PRDCT2 = 0.0

PRODCT(IPT) = 0.0

DO 340 NSHP=1,NSHAPS

IF( NSHP .EQ. 1 ) THEN
&      PRDCT0 = PRDCT0 + SNEW(NSHP,0,NOWPSI)*TCORLS(NSHP,
&      IPT)*CTHP
&      PRDCT1 = PRDCT1 + SNEW(NSHP,1,NOWPSI)*TCORLS(NSHP,
&      IPT)*CTHP
&      PRDCT2 = PRDCT2 + SNEW(NSHP,2,NOWPSI)*TCORLS(NSHP,
&      IPT)*CTHP
&      PRODCT(IPT) = PRODCT(IPT)+SNEW(NSHP,0,NOWPSI)
&      *CIFMOM(NSHP,IPT)*CTHP
ELSE
&      PRDCT0 = PRDCT0 + SNEW(NSHP,0,NOWPSI)*
&      TCORLS(NSHP,IPT)
&      PRDCT1 = PRDCT1 + SNEW(NSHP,1,NOWPSI)*
&      TCORLS(NSHP,IPT)
&      PRDCT2 = PRDCT2 + SNEW(NSHP,2,NOWPSI)*
&      TCORLS(NSHP,IPT)
&      PRODCT(IPT) = PRODCT(IPT)+SNEW(NSHP,0,NOWPSI)
&      *CIFMOM(NSHP,IPT)
ENDIF

340  CONTINUE

&      RESULTS(IST,IPT,4) = OMGASQ*TOMGA(IPT) + COEF1*PRDCT1
&      - COEF5*TGRAV(IPT)

&      RESULTS(IST,IPT,6) = PYAERO(IPT) - FACTR1*CTHC*TOMGA(IPT)
&      + COEF7*PRDCT0 - PRDCT2
&      + COEF2*TGRAV(IPT) + COEF4*OFFMAS(IPT)

&      RESULTS(IST,IPT,5) = PXAERO(IPT) + FACTR1*STHC*TOMGA(IPT)
&      + COEF4*PRDCT0 + COEF3*TGRAV(IPT)
&      + COEF6*OFFMAS(IPT)

EDGEFN(IPT) = RESULTS(IST,IPT,5)
VXFCN (IPT) = EDGEFN(IPT)*RESULTS(IST,IPT,2)
TSNFCN(IPT) = RESULTS(IST,IPT,4)*RESULTS(IST,IPT,2)
FLAPFN(IPT) = RESULTS(IST,IPT,6)

350  CONTINUE

C      Compute the torsion and moment array values for this
C      azimuth position.

```

DO 360 IPT=1,NPTS

EINTGR = TRPZOD(IPT , BLTIP , EDGEFN , NPTS)
SINTGR = TRPZOD(IPT , BLTIP , VXFCN , NPTS)
TINTGR = TRPZOD(IPT , BLTIP , TSNFCN , NPTS)
QZAERO = TRPZOD(IPT , BLTIP , QZFCN , NPTS)
FINTGR = TRPZOD(IPT , BLTIP , FLAPFN , NPTS)
ECENGR = TRPZOD(IPT , BLTIP , ECNTFN , NPTS)

RESLTS(IST,IPT,7) = TINTGR - FINTGR + COEF7*PRODCT(IPT)

&
RESLTS(IST,IPT,8) = EINTGR - OMGASQ*ECENGR
- COEF4*PRODCT(IPT) + COEF5*OFFMAS(IPT)

&
RESLTS(IST,IPT,9) = QZAERO - SINTGR + COEF4*DELTIM(IPT)
+ COEF8*OFFMAS(IPT)

360 CONTINUE

370 CONTINUE

TRQSUM = TRQSUM*NBLADS/NTORQS
POWER = TRQSUM*OMEGA*FPS2KW

C Print out the resulting values with a heading on the first
C pass only. Convert values to degrees for the printout.

CALL CONVRT (TODEGS)

IF (MOD(NCALLS , NYAW+1) .EQ. 0) THEN

VALUE(1) = ALENTH
VALUE(2) = ALPHAO
VALUE(3) = BETAO
VALUE(4) = BLSHNK
VALUE(5) = BLTIP
VALUE(6) = CHI
VALUE(7) = CSUBMA
VALUE(8) = DRGFRM
VALUE(9) = EUERR
VALUE(10) = HUBHT
VALUE(11) = HUBRAD
VALUE(12) = OMEGA
VALUE(13) = PHIAMP
VALUE(14) = PHIOMG
VALUE(15) = PHIO
VALUE(16) = PSIZER
VALUE(17) = RHOAIR
VALUE(18) = SHERXP
VALUE(19) = STEPMX
VALUE(20) = STEPMN
VALUE(21) = TSUBP
VALUE(22) = TSUBO
VALUE(23) = THETAP
VALUE(24) = THETAT
VALUE(25) = TRMERR
VALUE(26) = VHUB
VALUE(27) = DELT3
VALUE(28) = UNDSL
VALUE(29) = HUBMAS
VALUE(30) = HUBDIS

IVALUE(1) = KSHADW
IVALUE(2) = NBLADS
IVALUE(3) = NSHAPS
IVALUE(4) = NYAW

WRITE (4,4000)
WRITE (4,4100) TITLE1(1:LNTH(TITLE1))
& , TITLE2(1:LNTH(TITLE2))
& , TITLE3(1:LNTH(TITLE3))

WRITE (4,4200) 'Teetering'

DO 400 J=1,10
400 WRITE (4,4300) (LABEL1(I), VALUE(I), LABEL2(I), I=J,30,10)

```

WRITE (4,4400) ( I LABEL(I) , I VALUE(I) , I=1,4 )
END IF

C      Convert values back to radians for use in internal
C      calculations.

CALL CONVRT ( TORADS )

C      This section of code computes the mean flapwise moment at the
C      20% blade station for NPTS=21. If NPTS is changed, the second
C      subscript of RESLTS( , ,7) must be changed also.

C      Programmer note:
C      Changed to what?

NTOT = 0
AVEMOM = 0.0

DO 410 LL=1,NSTN-1

    AVEMOM = AVEMOM + RESLTS(LL,5,7)
    NTOT = NTOT + 1

410 CONTINUE

AVEMOM = AVEMOM/NTOT

C      Form the proper chart headings and print out the nine charts
C      of the results from this run.

IF ( NCALLS .EQ. 0 ) THEN

    LABEL4 = 'Trim Solution      '
    LABEL5 = ' '

ELSE

    LABEL4 = 'Yaw Solution, Rev #'
    WRITE (LABEL5,'(12.2)') NCALLS

END IF

WRITE(4,*) ' '
WRITE(4,*) ' ', 'PSI = (DEG.) ', 'TEETER = (DEG.) '
DO 450 II = 1, NSTN
    WRITE(4,*) IPSI(II), TEETER(II)*RAD2DG
450 CONTINUE

WRITE(4,*) ' '
WRITE(4,*) ' ', 'PSI = ( DEG.)', 'Teeter Damper Loads (ft-lb)'
DO 460 II = 1, NSTN
    WRITE(4,*) IPSI(II), FDAMP(II)
460 CONTINUE

DO 430 M=1,9

    WRITE (4,4500) LABEL3(M) , LABEL4 , LABEL5
    WRITE (4,4600)

    DO 420 I=1,NSTN

        RPO = RPHI(I,0)*RAD2DG
        RP1 = RPHI(I,1)*RAD2DG
        RP2 = RPHI(I,2)*RAD2DG

        WRITE (4,4700) IPSI(I) , RTIME(I) , RPO , RP1 , RP2
    &      , ( RESLTS(I,J,M) , J=1,NPTS,2 )

420 CONTINUE

```



```

C      *   inverse CMMASS matrix.  The inverse CMMASS matrix is a   *
C      *   square symmetric matrix of order 4.  The incoming ma-   *
C      *   trix can be of any size but must have exactly 4 rows.   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   External references in this routine:   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   none   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   Named COMMON blocks used in this routine:   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   INV   - Holds the inverse of the mass matrix.   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   Local and dummy variables used in this routine:   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   AMATRX - The incoming matrix to be premultiplied by   *
C      *   the inverse CMMASS matrix.   *
C      *   I     - Generic index.   *
C      *   J     - Generic index.   *
C      *   K     - Generic index.   *
C      *   M     - Number of rows in the incoming matrix.   *
C      *   N     - Number of columns in the incoming matrix.   *
C      *   TEMP  - Temporary work matrix.   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
C      *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *

```

```

REAL      AMATRX (4,4)
REAL      TEMP   (4,4)

INTEGER   I
INTEGER   J
INTEGER   K
INTEGER   M
INTEGER   N

INCLUDE   'C:INCLUDE\INV.INC'

```

```

C      Multiply AINVRs by AMATRX putting the result into TEMP.

```

```

DO 30 I=1,M
  DO 20 J=1,N
    TEMP(I,J) = 0.0
    DO 10 K=1,M
      TEMP(I,J) = TEMP(I,J) + AINVR(I,K) * AMATRX(K,J)
    10 CONTINUE
  20 CONTINUE
30 CONTINUE

```

```

C      Move the resulting temporary matrix into the original
C      incoming matrix.

```

```

DO 50 I=1,M
  DO 40 J=1,N
    AMATRX(I,J) = TEMP(I,J)
  40 CONTINUE
50 CONTINUE

```

```

RETURN

```

END
SUBROUTINE NXTPHI (TIMNOW)

```

C
C *****
C *
C * Subroutine NXTPHI computes the next values of the yaw *
C * variables. If this is a trim run, then the yaw varia- *
C * bles do not change. If this is a yawing run, then the *
C * yaw variables are computed based on the next value of *
C * the time variable. *
C *
C * The basic yaw function used here is: *
C *
C *   Phi = Phi-sub-0 + SIN( Omega-sub-Phi*Time ). *
C *
C * The two derivatives with respect to time, Phi-dot and *
C * Phi-double-dot are derived in a straightforward manner *
C * via differentiation with respect to time. *
C *
C * The indices of the PHI array correspond to the order of *
C * the derivative of Phi with respect to time. *
C *
C *****

```

```

C *****
C *
C * External references in this routine: *
C *
C *   none *
C *
C *****

```

```

C *****
C *
C * Named COMMON blocks used in this routine: *
C *
C *   POSITN - Holds parameters related to blade position *
C *           such as PHI, PSI, etc. *
C *   TURBN  - Holds turbine parameters such as number of *
C *           blades, rotor speed, etc. *
C *
C *****

```

```

C *****
C *
C * Local and dummy variables used in this routine: *
C *
C *   OMPHIT - Phase angle of the sinusoidal yaw function. *
C *           It corresponds to the Omega-sub-Phi*T term *
C *           of the formulation notes. The Omega-sub-Phi *
C *           component is the yaw velocity amplitude *
C *           divided by the yaw angle amplitude. *
C *   SOMPHT - Sine of Omega-sub-Phi*T. *
C *   TIMNOW - Current time. TIMNOW is identically zero *
C *           for the trim solution computation. *
C *
C *****

```

```

REAL      OMPHIT
REAL      SOMPHT
REAL      TIMNOW

```

```

INCLUDE    'C:INCLUDE\POSITN.INC'

```

```

INCLUDE    'C:INCLUDE\TURBN.INC'

```

```

C      Check to see if this is time zero.

```

```

IF ( TIMNOW .EQ. 0.0 ) THEN

```

```

C      Set the yaw function and its derivatives for TIME=0.

```

```

PHI(0) = PHI0
PHI(1) = PHIOMG

```

PHI(2) = 0.0

ELSE

C Compute the yaw function and its derivatives for times
C greater than time zero.

OMPHT = TIMNOW*PHIOMG/PHIAMP
SOMPHT = SIN(OMPHT)
PHI(0) = PHI0 + PHIAMP*SOMPHT
PHI(1) = PHIOMG*COS(OMPHT)
PHI(2) = -PHIOMG*PHIOMG*SOMPHT/PHIAMP

END IF

RETURN

END

SUBROUTINE NXTPSI (IPSIST)

C *****
C *
C * Subroutine NXTPSI computes the next value of the azi- *
C * muth angle, Psi, based on the current value of the Eu- *
C * ler error and the previous delta-Psi value. If the er- *
C * ror is too large, a smaller delta-Psi will be used to *
C * compute the value of Psi, unless the delta-Psi is al- *
C * ready at the STEPMN lower limit. Likewise, if the Eu- *
C * ler error is below a certain limit, the delta-Psi value *
C * is increased for the next Psi computation. *
C *
C *****

C *****
C *
C * External references in this routine: *
C * *
C * none *
C * *
C *****

C *****
C *
C * Named COMMON blocks used in this routine: *
C * *
C * CONST - Turbine and other constants used in load *
C * calculations. *
C * LIMITC - Holds values used in the LIMITS routine. *
C * POSITN - Holds parameters related to blade position *
C * such as PHI, PSI, etc. *
C * TURBN - Holds turbine parameters such as number of *
C * blades, rotor speed, etc. *
C * *
C *****

C *****
C *
C * Local and dummy variables used in this routine: *
C * *
C * IPSIST - The next STEPMX station in integer degrees. *
C * NEW - Array index for new data. *
C * OLD - Array index for old data. *
C * PSIST - The next STEPMX station in radians. *
C * *
C *****

REAL PSIST

INTEGER IPSIST

INTEGER NEW

INTEGER OLD

INCLUDE 'C:\INCLUDE\CONST2.INC'

INCLUDE 'C:\INCLUDE\LIMITC.INC'

```

INCLUDE      'C:INCLUDE\POSITN.INC'

INCLUDE      'C:INCLUDE\TURBN.INC'

SAVE        NEW
SAVE        OLD

DATA        NEW    / 1 /
DATA        OLD    / 0 /

```

```

C      Check the error.  If it is too large, decrease delta-Psi by
C      50%.  If it is very small, increase delta-Psi by 50%.  The
C      value of delta-Psi must always be greater than STEPMN.

```

```

IF ( ERROR .GT. EUERR ) THEN

    DELPSI(NEW) = AMAX1( 0.5*DELPSI(NEW) , STEPMN )

ELSE IF ( ERROR .LT. 0.1*EUERR ) THEN

    DELPSI(NEW) = 1.5*DELPSI(NEW)

END IF

```

```

C      This section of code checks the position of the new Psi against
C      the next STEPMX station.  If the next Psi will go past the
C      STEPMX station, the delta-Psi is adjusted to bring the next Psi
C      directly onto the STEPMX point.  If the next Psi will fall just
C      short of a STEPMX station, the delta-Psi is adjusted to assure
C      that it will not be necessary to use a delta-Psi smaller than
C      STEPMN to reach the next STEPMX station on the next iteration.

```

```

C      Programmer note:

```

```

C      The following logic can produce a delta-Psi that is less
C      than STEPMN for two steps.  It would then move back within
C      tolerances.  This case comes up when we are within slightly
C      less than two STEPMS of the next STEPMX station.  This
C      algorithm cannot produce a delta-Psi that is less than
C      STEPMN/2.
C      MLB

```

```

PSIST = IPSIST/RAD2DG

IF ( PSI(OLD)+DELPSI(NEW) .GT. PSIST ) THEN

    DELPSI(NEW) = PSIST - PSI(OLD)

ELSE IF ( PSI(OLD)+DELPSI(NEW)+STEPMN .GT. PSIST ) THEN

    IF ( (DELPSI(NEW) .NE. STEPMN) .OR. (ERROR .LE. EUERR) ) THEN

        DELPSI(NEW) = 0.5*( PSIST - PSI(OLD) )

    END IF

END IF

```

```

C      Set new Psi and delta-Time.

```

```

PSI(NEW)    = PSI(OLD) + DELPSI(NEW)
DELTAT(NEW) = DELPSI(NEW)/OMEGA

```

```

C      Get time values if we're working on a yawing solution.

```

```

IF ( ITRIM .EQ. 0 ) TIME(NEW) = TIME(OLD) + DELTAT(NEW)

```

```

RETURN
END
SUBROUTINE RUN ( NPTS , NEWSET , HAVRUN )

```

```

C      *****

```



```

C      *
C      * Subroutine RUN performs the modeling of the rotor blade *
C      * motion. The coefficients and data read in by subrou- *
C      * tine DATAIN are used along with the run limit and RUN *
C      * setup values input in subroutines LIMITS and SETUP. *
C      *
C      * There are two basic steps to the run solution - the *
C      * trim solution and the yawing solution. In the trim so- *
C      * lution, the yaw position and yaw rate are kept at a *
C      * fixed value, and the time is kept fixed at zero. The *
C      * equations are solved for successive revolutions around *
C      * the rotor disk until the tip displacement functions no *
C      * longer change significantly from one revolution to the *
C      * next. This is the trim solution. Once the trim solu- *
C      * tion is achieved, the trim solution is completed, with *
C      * the yaw angle, yaw angular velocity and acceleration *
C      * computed from a given function and the time elapsed *
C      * since the start of the yaw solution. After each rotor *
C      * disk revolution of the yaw solution, the loads are cal- *
C      * culated and printed out. *
C      *
C      *****
C
C      *****
C      *
C      * External references in this routine: *
C      *
C      *   CONVRT - Performs units conversions. *
C      *   DSKREV - Performs a full disk revolution solution to *
C      *             the blade equations of motion. *
C      *   GETFIL - Get the name of the results file and open *
C      *             it. *
C      *   LIMITS - Interactive input of run limits. *
C      *   LODOUT - Calculates and prints blade loads. *
C      *   SETUP  - Interactive modification of free variables. *
C      *   SHAPES - Calculates the four coordinate shape func- *
C      *             tions. *
C      *   SOLVE  - Calculates the inverse CMMASS matrix for *
C      *             premultiplying other coefficient matrices. *
C      *   TRMTST - Compares the results of two consecutive ro- *
C      *             tor revolutions. *
C      *
C      *****
C
C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      *   CONST - Turbine and other constants used in load *
C      *             calculations. *
C      *   LIMITC - Holds values used in the LIMITS routine. *
C      *   POSITN - Holds parameters related to blade position *
C      *             such as PHI, PSI, etc. *
C      *   SARAYS - Holds new and old values for the general- *
C      *             ized coordinates. *
C      *   TURBN  - Holds turbine parameters such as number of *
C      *             blades, rotor speed, etc. *
C      *
C      *****
C
C      *****
C      *
C      * Local and dummy variables used in this routine: *
C      *
C      *   ANS   - Used to store input responses from key- *
C      *             board. *
C      *   HAVRUN - Flag that indicates that the model has been *
C      *             run. Used for diagnostic runs. *
C      *   I     - Generic index. *
C      *   ITCNT - The count of the number of disk revolutions *
C      *             required to achieve a trim solution. *
C      *   J     - Generic index. *
C      *   NCALLS - Number of calls to LODOUT. *
C      *   NEWSSET - Flag to control calls to SOLVE. *
C      *   NPTS  - Number of points along the blade used to *
C      *             perform Simpson's integration for calculat- *
C      *             ing the moments and forces at the blade *
C      *             root. (passed from STRAP1) *

```

```

C      *      NSHP - Counter on DO loops for the coordinate *
C      *      shape function. *
C      *      RESFIL - Name of file to which results are written. *
C      *      TODEGS - Flag used to tell CONVRT to convert 'free' *
C      *      variables to degrees. *
C      *      TORADS - Flag used to tell CONVRT to convert 'free' *
C      *      variables to radians. *
C      *
C      *****

```

```

INTEGER      I
INTEGER      ITCNT
INTEGER      J
INTEGER      NCALLS
INTEGER      NPTS
INTEGER      NSHP

```

```

LOGICAL      HAVRUN
LOGICAL      NEWSET
LOGICAL      TODEGS
LOGICAL      TORADS

```

```

CHARACTER*1  ANS
CHARACTER*50 RESFIL

```

```

INCLUDE      'C:INCLUDE\CONST2.INC'

```

```

INCLUDE      'C:INCLUDE\LIMITC.INC'

```

```

INCLUDE      'C:INCLUDE\POSITN.INC'

```

```

INCLUDE      'C:INCLUDE\SARAYS.INC'

```

```

INCLUDE      'C:INCLUDE\TURBN.INC'

```

```

SAVE        RESFIL
SAVE        TODEGS
SAVE        TORADS

```

```

DATA        RESFIL / 'RESULTS.DAT' /
DATA        TODEGS / .TRUE. /
DATA        TORADS / .FALSE. /

```

```

2000 FORMAT ( A )
4000 FORMAT ( / ' The trim condition is not satisfied after' , I3.2
&          , ' rotor'
&          / ' revolutions. Shall we continue? (=Y,N) > ' )
4100 FORMAT ( '& The trim condition was not satisfied.' )
4200 FORMAT ( / ' Trim test #' , I2.2 , ' completed.' )
4300 FORMAT ( '& The trim condition was satisfied.' )
6000 FORMAT ( / ' Do you want to do another run with this data? (Y,N)'
&          , ' > ' )

```

```

C      The main run loop begins here. The interactive program se-
C      quence starts.

```

```

100 CONTINUE

```

```

C      Clear NCALLS which represents the number of calls to LODOUT.
C      It is used to control printing of the title page. It is
C      incremented in LODOUT just before returning.

```

```

NCALLS = 0

```

```

C      Set up a run by first calling the SETUP and LIMITS routines.
C      Temporarily convert the units of the 'free' variables to de-
C      grees and RPM for interactive use.

```

```

CALL CONVRT ( TODEGS )
CALL SETUP
CALL LIMITS
CALL CONVRT ( TORADS )

```

C Before starting the model run, check for potential error in the
C yaw function. A value of zero for PHIAMP in a yaw solution
C will cause the program to abort.

```
IF ( ( NYAW .GT. 0 ) .AND. ( PHIAMP .EQ. 0.0 ) ) THEN
```

```
PRINT *, ' '  
PRINT *, 'Error in input data. PHIAMP cannot be zero'  
PRINT *, 'for a yaw solution. Please review the setup'  
PRINT *, 'and limits values.'
```

```
GO TO 600
```

```
END IF
```

C On the first pass through the RUN routine, invert the mass
C coefficient matrix and premultiply all the coefficient ma-
C trices by the result. This is done just once per data set.

```
IF ( NEWSET ) THEN
```

```
CALL SOLVE  
NEWSET = .FALSE.
```

```
END IF
```

```
200 PRINT *, ' '  
PRINT *, 'Are you ready to run the model? (=Y,N) > '  
READ 2000, ANS  
PRINT *, ANS
```

```
IF ( ( ANS .EQ. 'N' ) .OR. ( ANS .EQ. 'n' ) ) RETURN
```

```
IF ( ( ANS .NE. 'Y' ) .AND. ( ANS .NE. 'y' )  
& .AND. ( ANS .NE. ' ' ) ) THEN
```

```
PRINT *, ' >>> Invalid response. Please try again. <<<'  
GO TO 200
```

```
END IF
```

C Open the file that is to contain the results.

```
CALL GETFIL ( RESFIL )
```

C Zero the blade deflection, velocity and acceleration for the
C start of the trim solution run.

```
DO 320 NSHP=1,NSHAPS  
DO 310 I=0,2  
DO 300 J=0,360
```

```
SNEW(NSHP,I,J) = 0.0  
SOLD(NSHP,J) = 0.0
```

```
300 CONTINUE  
310 CONTINUE  
320 CONTINUE
```

C Start the trim run. When the ITRIM flag is set to 2, the first
C pass through DSKREV will also execute the STARTUP startup rou-
C tine. STARTUP computes the static deflection of the blade at
C the 270 degree azimuth position, under the given wind condi-
C tions. This static deflection is used to start the actual mo-
C del run, and helps to assure rapid convergence to the trim so-
C lution by removing many of the startup transients associated
C with beginning execution without a realistic initial blade tip
C deflection.

```
ITRIM = 2  
PSI(1) = 270/RAD2DG  
DELPSI(1) = STEPMM  
ERROR = 0.0
```

```

CALL DSKREV ( NPTS )

ITRIM = 1

C      Iterate for the trim solution. Keep count of the rotor re-
C      volutions and give the user a chance to bail out every ten
C      revolutions.

ITCNT = 0

400 IF ( ITRIM .EQ. 1 ) THEN
      IF ( ITCNT .NE. 0 ) THEN
        IF ( MOD( ITCNT , 10 ) .EQ. 0 ) THEN
          PRINT 4000, ITCNT
          READ 2000, ANS
          PRINT *, ANS

          IF ( ( ANS .EQ. 'N' ) .OR. ( ANS .EQ. 'n' ) ) THEN
            PRINT *, ' '
            PRINT *, 'STRAP terminated by user.'
            PRINT *, ' '

            STOP

          END IF

        ELSE

          PRINT 4100

        END IF

      END IF

C      Initialize the azimuth angle and run another revolution.
C      Test the last set of solutions against this set for stabil-
C      ity. When TRMTST is satisfied, ITRIM will be set to zero
C      the loop will be terminated.

      PSI(1) = 0.0

      CALL DSKREV ( NPTS )

      ITCNT = ITCNT + 1

      CALL TRMTST ( ITRIM )

      PRINT 4200, ITCNT

      GO TO 400

    END IF

C      Trim solution completed.

    PRINT 4300

C      Calculate results of the trim solution.

    CALL LODOUT ( NCALLS )

C      Initialize the yaw run loop by starting time at zero. Run yaw
C      solution NYAW times. Calculate the loads and print out the
C      results after each disk revolution.

    IF ( NYAW .GT. 0 ) THEN
      TIME(1) = 0.0

```

```

DO 500 I=1,NYAW
    PSI(1) = 0.0
    CALL DSKREV ( NPTS )
    CALL LODOUT ( NCALLS )
500 CONTINUE
END IF

C      Run complete.

600 PRINT 6000
READ 2000, ANS
PRINT *, ANS

IF ( ( ANS .EQ. 'Y' ) .OR. ( ANS .EQ. 'y' ) ) THEN
    PRINT *, ' '
    PRINT *, 'You must not change the number of shapes, NSHAPS!!!'
    PRINT *, ' '

    GO TO 100

ELSE IF ( ( ANS .NE. 'N' ) .AND. ( ANS .NE. 'n' )
&      .AND. ( ANS .NE. ' ' ) ) THEN
    PRINT *, ' >>> Invalid response. Please try again. <<<'

    GO TO 600

END IF

C      Set the HAVRUN flag to true to enable diagnostic runs.

HAVRUN = .TRUE.

RETURN
END
SUBROUTINE SAVE1 ( IPSIST , NSHAPS , STEMP )

C      *****
C      *
C      * Subroutine SAVE1 makes a permanent copy of the tip dis- *
C      * placement variables at each STEPMX station around the *
C      * rotor disk. These values are used by the loads routine *
C      * to calculate the blade loads based upon the tip dis- *
C      * placements. Only the values corresponding to the spec- *
C      * ified number of shape functions are saved. *
C      *
C      *****

C      *****
C      *
C      * External references in this routine: *
C      *
C      * none *
C      *
C      *****

C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      * CONST - Turbine and other constants used in load *
C      * calculations. *
C      * LIMITC - Holds values used in the LIMITS routine. *
C      * POSITN - Holds parameters related to blade position *
C      * such as PHI, PSI, etc. *
C      * SARAYS - Holds new and old values for the general- *
C      * ized coordinates. *
C      *
C      *****

```

```

C *****
C *****
C *
C * Local and dummy variables used in this routine: *
C * *
C * IPSIST - The next STEPMX station in integer degrees. *
C * NEW - Array index for new data. *
C * NSHAPS - Number of blade shape functions, 4 maximum. *
C * NSHP - Counter on DO loops for the coordinate *
C * shape function. *
C * STEMP - Temporary array holding the tip displace- *
C * ment values. The third dimension repre- *
C * sents the order of the time derivative. *
C * *
C *****

```

```

REAL STEMP (4,0:1,0:2)

```

```

INTEGER IPSIST
INTEGER NEW
INTEGER NSHAPS
INTEGER NSHP

```

```

INCLUDE 'C:\INCLUDE\CONST2.INC'

```

```

INCLUDE 'C:\INCLUDE\LIMITC.INC'

```

```

INCLUDE 'C:\INCLUDE\POSITN.INC'

```

```

INCLUDE 'C:\INCLUDE\SARAYS.INC'

```

```

SAVE NEW

```

```

DATA NEW / 1 /

```

```

C Move the values stored in STEMP into SNEW.

```

```

DO 10 NSHP=1,NSHAPS

```

```

    SNEW(NSHP,0,IPSIST) = STEMP(NSHP,NEW,0)
    SNEW(NSHP,1,IPSIST) = STEMP(NSHP,NEW,1)
    SNEW(NSHP,2,IPSIST) = STEMP(NSHP,NEW,2)

```

```

10 CONTINUE

```

```

C Set the STEPMX station pointer, IPSIST, for the next STEPMX
C location on the rotor disk.

```

```

IPSIST = IPSIST + STEPMX*RAD2DG + 0.001

```

```

RETURN
END
SUBROUTINE SETUP

```

```

C *****
C *
C * Subroutine SETUP allows the user to interactively *
C * change the 'free' variables of STRAP. The 'free' vari- *
C * ables are those that do not affect the computation of *
C * the coefficient matrices or property arrays and which *
C * are not specifically machine dependent. *
C * *
C *****

```

```

C *****
C *
C * External references in this routine: *
C * *
C * none *
C * *
C *****

```

```

C *****
C *
C *   Named COMMON blocks used in this routine:
C *
C *   BLADE - Holds blade property values such as stiff-
C *           ness and mass distributions.
C *   CONST - Turbine and other constants used in load
C *           calculations.
C *   LITERL - Holds data set titles.
C *   TURBN - Holds turbine parameters such as number of
C *           blades, rotor speed, etc.
C *   WIND - Holds wind shear and tower shadow paramete-
C *           rs.
C *
C *****
C
C *****
C *
C *   Local and dummy variables used in this routine:
C *
C *   ANS - Used to store responses to questions.
C *   ICASE - Number of variable to be changed.
C *
C *****

```

```

INTEGER      ICASE

CHARACTER*1  ANS

INCLUDE      'C:INCLUDE\BLADE2.INC'
INCLUDE      'C:INCLUDE\CONST2.INC'
INCLUDE      'C:INCLUDE\LITERL.INC'
INCLUDE      'C:INCLUDE\TRBINF.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\WIND2.INC'
INCLUDE      'C:INCLUDE\HUBPRP2.INC'
INCLUDE      'C:INCLUDE\SPRING2.INC'

```

```

1000 FORMAT ( // ' Current values for the "free" variables:'
&           // ' 1 ALENTH = ', F10.4, ' feet '
&           // ' 12 PHIAMP = ', F10.4, ' degrees'
&           // ' 2 ALPHA0 = ', F10.4, ' degrees '
&           // ' 13 PHIOMG = ', F10.4, ' degrees/sec'
&           // ' 3 BETA0 = ', F10.4, ' degrees '
&           // ' 14 PSISHD = ', F10.4, ' degrees'
&           // ' 4 CHI = ', F10.4, ' degrees '
&           // ' 15 PSIZER = ', F10.4, ' degrees'
&           // ' 5 GRAV = ', F10.4, ' feet/sec^2 '
&           // ' 16 RHOAIR = ', F10.4, ' Slugs/feet^3'
&           // ' 6 HUBHT = ', F10.4, ' feet '
&           // ' 17 SHERXP = ', F10.4
&           // ' 7 KSHADW = ', I10, ' 12X
&           // ' 18 THETAP = ', F10.4, ' degrees'
&           // ' 8 NBLADS = ', I10, ' 12X
&           // ' 19 THETAT = ', F10.4, ' degrees'
&           // ' 9 NSHAPS = ', I10, ' 12X
&           // ' 20 TSUBO = ', F10.4
&           // ' 10 OMEGA = ', F10.4, ' RPM '
&           // ' 21 TSUBP = ', F10.4
&           // ' 11 PHIO = ', F10.4, ' degrees '
&           // ' 22 VHUB = ', F10.4, ' feet/second'
&           // ' 23 BETA1 = ', F10.4, ' degrees '
&           // ' 24 BETA2 = ', F10.4, ' degrees '
&           // ' 25 TIMINC = ', F10.6, ' sec '
&           // ' 26 NUMSCN = ', I10 )
2000 FORMAT ( A )

```

C Print the values of the free variables.

```

100 PRINT 1000, ALENTH , PHIAMP
&      , ALPHAO , PHIOMG
&      , BETA0 , PSISHD
&      , CHI , PSIZER
&      , GRAV , RHOAIR
&      , HUBHT , SHERXP
&      , KSHADW , THETAP
&      , NBLADS , THETAT
&      , NSHAPS , TSUB0
&      , OMEGA , TSUBP
&      , PHIO , VHUB
&      , BETA1 , BETA2
&      , TIMINC , NUMSCN

```

C Allow user to change values of the 'free' variables. Display
C values of all variables after each change.

```

PRINT * , ' '
200 PRINT * , 'Would you like to change any values? (Y,=N) > '
READ 2000, ANS
PRINT * , ANS

IF ( ( ANS .EQ. 'Y' ) .OR. ( ANS .EQ. 'y' ) ) THEN

210 PRINT * , 'Enter the number of the variable you would like'
PRINT * , 'to change (1-26) > '
READ * , ICASE
PRINT * , ICASE

GO TO (300,302,305,310,315,320,325,330,335,340,345
&      ,350,355,360,365,370,375,380,385,390,392,395
&      ,400,402,403,404) ICASE

PRINT * , ' >>> Invalid response. Please try again. <<<'
PRINT * , ' '

GO TO 210

300 PRINT * , 'Enter new REAL value for ALENTH > '
READ * , ALENTH
PRINT * , ALENTH
GO TO 100

302 PRINT * , 'Enter new REAL value for ALPHAO > '
READ * , ALPHAO
PRINT * , ALPHAO
GO TO 100

305 PRINT * , 'Enter new REAL value for BETA0 > '
READ * , BETA0
PRINT * , BETA0
GO TO 100

310 PRINT * , 'Enter new REAL value for CHI > '
READ * , CHI
PRINT * , CHI
GO TO 100

315 PRINT * , 'Enter new REAL value for GRAV > '
READ * , GRAV
PRINT * , GRAV
GO TO 100

320 PRINT * , 'Enter new REAL value for HUBHT > '
READ * , HUBHT
PRINT * , HUBHT
GO TO 100

325 PRINT * , 'Enter new INTEGER value for KSHADW > '
READ * , KSHADW
PRINT * , KSHADW
GO TO 100

330 PRINT * , 'Enter new INTEGER value for NBLADS > '
READ * , NBLADS
PRINT * , NBLADS
GO TO 100

```



```

335 PRINT *, 'Enter new INTEGER value for NSHAPS > '
    READ *, NSHAPS
    PRINT *, NSHAPS
    GO TO 100

340 PRINT *, 'Enter new REAL value for OMEGA > '
    READ *, OMEGA
    PRINT *, OMEGA
    GO TO 100

345 PRINT *, 'Enter new REAL value for PHIO > '
    READ *, PHIO
    PRINT *, PHIO
    GO TO 100

350 PRINT *, 'Enter new REAL value for PHIAMP > '
    READ *, PHIAMP
    PRINT *, PHIAMP
    GO TO 100

355 PRINT *, 'Enter new REAL value for PHIOMG > '
    READ *, PHIOMG
    PRINT *, PHIOMG
    GO TO 100

360 PRINT *, 'Enter new REAL value for PSISHD > '
    READ *, PSISHD
    PRINT *, PSISHD
    GO TO 100

365 PRINT *, 'Enter new REAL value for PSIZER > '
    READ *, PSIZER
    PRINT *, PSIZER
    GO TO 100

370 PRINT *, 'Enter new REAL value for RHOAIR > '
    READ *, RHOAIR
    PRINT *, RHOAIR
    GO TO 100

375 PRINT *, 'Enter new REAL value for SHERXP > '
    READ *, SHERXP
    PRINT *, SHERXP
    GO TO 100

380 PRINT *, 'Enter new REAL value for THETAP > '
    READ *, THETAP
    PRINT *, THETAP
    GO TO 100

385 PRINT *, 'Enter new REAL value for THETAT > '
    READ *, THETAT
    PRINT *, THETAT
    GO TO 100

390 PRINT *, 'Enter new REAL value for TSUBO > '
    READ *, TSUBO
    PRINT *, TSUBO
    GO TO 100

392 PRINT *, 'Enter new REAL value for TSUBP > '
    READ *, TSUBP
    PRINT *, TSUBP
    GO TO 100

395 PRINT *, 'Enter new REAL value for VHUB > '
    READ *, VHUB
    PRINT *, VHUB
    GO TO 100

400 PRINT *, ' Enter new REAL value for BETA1>'
    READ *, BETA1
    PRINT *, BETA1
    GO TO 100

402 PRINT *, ' Enter new REAL value for BETA2>'
    READ *, BETA2

```

```

PRINT *, BETA2
GO TO 100

403 PRINT *, ' Enter new REAL value for TIMINC>'
READ *, TIMINC
PRINT *, TIMINC
GO TO 100

404 PRINT *, ' Enter new INTEGER value for NUMSCN>'
READ *, NUMSCN
PRINT *, NUMSCN
GO TO 100

ELSE IF ( ( ANS .NE. 'N' ) .AND. ( ANS .NE. 'n' )
& .AND. ( ANS .NE. ' ' ) ) THEN

PRINT *, ' >>> Invalid response. Please try again. <<<'
PRINT *, ' '

GO TO 200

END IF

```

C Compute commonly used constants.

```

RR = BLTIP + HUBRAD
ABAR = ALENT/RR
RROMGA = RR*OMEGA*PI/30.0

```

RETURN
END

FUNCTION SIMPSN (LOWLIM , UPLIM , NPTS , FOFX)

```

C *****
C *
C * Function SIMPSN performs a composite Simpson's integra- *
C * tion of a given data set. This routine is identical to *
C * the SIMPSN routine used by COEFFS in Module 1. *
C *
C *****

C *****
C *
C * External references in this routine: *
C *
C * none *
C *
C *****

C *****
C *
C * Named COMMON blocks used in this routine: *
C *
C * none *
C *
C *****

C *****
C *
C * Local and dummy variables used in this routine: *
C *
C * FOFX - Array of data points to be integrated. *
C * They are treated as the value of a function *
C * evaluated at a specific point. *
C * H - Subinterval size. *
C * I - Generic index. *
C * LOWLIM - Lower limit of integration. *
C * NPTS - Number of points along the blade used to *
C * perform Simpson's integration for calculat- *
C * ing the moments and forces at the blade *
C * root. *
C * SIMPSN - Value of the integral from LOWLIM to UPLIM. *
C * UPLIM - Upper limit of integration. *
C *
C *****

```

```

REAL          FOFX  (21)
REAL          H
REAL          LOWLIM
REAL          SIMPSN
REAL          UPLIM

INTEGER       I
INTEGER       NPTS

```

C Compute the subinterval size and initialize the integral.

```

H          = ( UPLIM - LOWLIM )/( NPTS - 1 )
SIMPSN = 0.0

```

C Add in the intermediate points. In the formulation, all even
C numbered points have coefficient of 4. In this case, the index
C must be shifted to form the proper coefficient.

```

DO 10 I=2,NPTS-1,2
10 SIMPSN = SIMPSN + 4.0*FOFX(I) + 2.0*FOFX(I+1)

SIMPSN = SIMPSN + FOFX(1) - FOFX(NPTS)

SIMPSN = H*SIMPSN/3.0

```

```

RETURN
END
SUBROUTINE SOLVE

```

```

C        *****
C        *
C        * Subroutine SOLVE computes the inverse of coefficient *
C        * matrix CMMASS and then premultiplies all the other co- *
C        * efficient matrices by the inverse CMMASS matrix. The *
C        * result of multiplying the inverse CMMASS matrix by the *
C        * CMMASS matrix is printed on the screen for diagnostic *
C        * purposes. It should be very close to the Identity mat- *
C        * rix. *
C        *
C        *****

```

```

C        *****
C        *
C        * External references in this routine: *
C        *
C        *     INVERT - Provides an interface to the GAUSS routine. *
C        *     MULT  - Premultiplies the incoming matrix by the *
C        *            inverse CMMASS matrix. *
C        *
C        *****

```

```

C        *****
C        *
C        * Named COMMON blocks used in this routine: *
C        *
C        *     INV   - Holds the inverse of the mass matrix. *
C        *     MATRX1 - Holds stiffness coefficient matrices. *
C        *     MATRX2 - Holds other matrices related to coriolis *
C        *            stiffening, gravity, etc. *
C        *     TURBN - Holds turbine parameters such as number of *
C        *            blades, rotor speed, etc. *
C        *
C        *****

```

```

C        *****
C        *
C        * Local and dummy variables used in this routine: *
C        *
C        *     I     - Generic index. *
C        *     J     - Generic index. *
C        *     K     - Generic index. *
C        *     TEMP  - A temporary storage array used for matrix *
C        *            manipulation. *
C        *
C        *****

```

```

C      *
C      *****
REAL          TEMP   (4,4)

INTEGER       I
INTEGER       J
INTEGER       K

INCLUDE       'C:INCLUDE\INV.INC'

INCLUDE       'C:INCLUDE\MATRIX1.INC'

INCLUDE       'C:INCLUDE\MATRIX2.INC'

INCLUDE       'C:INCLUDE\TURBN.INC'

INCLUDE       'C:INCLUDE\SPRING2.INC'

4000 FORMAT ( 1X , 4F12.7 / )

C      Invert the CMMASS matrix.

CALL INVERT ( CMMASS , AINVRS , NSHAPS )

C      Premultiply the other coefficient matrices by the inverse
C      CMMASS matrix.

CALL MULT ( CKBEND , NSHAPS , NSHAPS )
CALL MULT ( CKTOMG , NSHAPS , NSHAPS )
CALL MULT ( CKTGRV , NSHAPS , NSHAPS )
CALL MULT ( CKQLOD , NSHAPS , NSHAPS )

C      The coefficient matrix CKTCRL must be separated into individual
C      2-dimensional matrices for multiplication by the inverse CMMASS
C      matrix. Then the individual matrices are recombined into the
C      original 3-dimensional form.

DO 140 I=1,NSHAPS
  DO 110 J=1,NSHAPS
    DO 100 K=J,NSHAPS
      TEMP(J,K) = CKTCRL(I,J,K)
      TEMP(K,J) = CKTCRL(I,K,J)
100    CONTINUE
110    CONTINUE
      CALL MULT ( TEMP , NSHAPS , NSHAPS )
      DO 130 J=1,NSHAPS
        DO 120 K=J,NSHAPS
          CKTCRL(I,J,K) = TEMP(J,K)
          CKTCRL(I,K,J) = TEMP(K,J)
120        CONTINUE
130      CONTINUE
140    CONTINUE

C      Load three of the 1-dimensional matrices into a single matrix
C      of order 4 by 3 and premultiply it by the inverse CMMASS ma-
C      trix. Then move the resulting columns back into the original
C      vectors.

DO 200 J=1,NSHAPS
  TEMP(J,1) = CMRIGD(J)

```

```
TEMP(J,2) = CMBLNC(J)
TEMP(J,3) = CMGRAV(J)
```

200 CONTINUE

```
CALL MULT ( TEMP , NSHAPS , 3 )
```

```
DO 210 J=1,NSHAPS
```

```
CMRIGD(J) = TEMP(J,1)
CMBLNC(J) = TEMP(J,2)
CMGRAV(J) = TEMP(J,3)
```

210 CONTINUE

C Repeat the same process for the special teetering vectors.

```
DO 300 J=1,NSHAPS
```

```
TEMP(J,1) = CMRGD1(J)
TEMP(J,2) = CMGRV1(J)
TEMP(J,3) = CMHUB1(J)
```

300 CONTINUE

```
CALL MULT ( TEMP , NSHAPS , 3 )
```

```
DO 310 J=1,NSHAPS
```

```
CMRGD1(J) = TEMP(J,1)
CMGRV1(J) = TEMP(J,2)
CMHUB1(J) = TEMP(J,3)
```

310 CONTINUE

```
DO 330 J=1,NSHAPS
```

```
TEMP(J,1) = KOSTIF(J)
TEMP(J,2) = K1STIF(J)
TEMP(J,3) = K2STIF(J)
TEMP(J,4) = DAMP (J)
```

330 CONTINUE

```
CALL MULT( TEMP, NSHAPS, 4)
```

```
DO 340 J=1,NSHAPS
```

```
KOSTIF(J) = TEMP(J,1)
K1STIF(J) = TEMP(J,2)
K2STIF(J) = TEMP(J,3)
DAMP(J) = TEMP(J,4)
```

340 CONTINUE

C Premultiply the CMMASS matrix by its inverse and save for diagnostic purposes. Print out the result.

```
CALL MULT ( CMMASS , NSHAPS , NSHAPS )
```

```
PRINT *, ' '
PRINT *, 'Product of CMMASS premultiplied by its inverse:'
PRINT *, ' '
```

```
DO 400 I=1,NSHAPS
```

```
400 PRINT 4000, ( CMMASS(I,J) , J=1,NSHAPS )
```

C CMMASS is replaced by its inverse for use in the RUN routine
C in computing the solution for the aerodynamic forces.

```
DO 500 I=1,NSHAPS
```

```
DO 510 J=1,NSHAPS
```

510 CMMASS(I,J) = AINVR(I,J)

500 CONTINUE

RETURN
END
SUBROUTINE STRTUP (STEMP , NPTS)

```
C *****
C *
C * Subroutine STRTUP computes the static deflection of the *
C * blade at the startup position (Psi=270). This is then *
C * used to compute further values based on an initial de- *
C * flection. The initial static deflection is computed *
C * based on the blade azimuth and the static (instantane- *
C * ous) aerodynamic loads on the blade. *
C *****
C *****
C *
C * External references in this routine: *
C *
C *   AFORCE - Calculates the value of the aerodynamic *
C *             force integral. *
C *   FORM1 - Calculates the values of the blade dis- *
C *             placement function. *
C *   INVERT - Provides an interface to the GAUSS routine. *
C *   NXTPHI - Calculates the next values of the yaw vari- *
C *             ables. *
C *****
C *****
C *
C * Named COMMON blocks used in this routine: *
C *
C *   AIRFRC - Holds values used in aerodynamic calcula- *
C *             tions. *
C *   CONST - Turbine and other constants used in load *
C *             calculations. *
C *   MATRX1 - Holds stiffness coefficient matrices. *
C *   MATRX2 - Holds other matrices related to coriolis *
C *             stiffening, gravity, etc. *
C *   POSITN - Holds parameters related to blade position *
C *             such as PHI, PSI, etc. *
C *   SARAYS - Holds new and old values for the general- *
C *             ized coordinates. *
C *   START - Holds initial blade deflection. *
C *   TURBN - Holds turbine parameters such as number of *
C *             blades, rotor speed, etc. *
C *****
C *****
C *
C * Local and dummy variables used in this routine: *
C *
C *   BLDANG - Blade azimuth position used in the teeter- *
C *             ing rotor option. *
C *   FAER01 - Aerodynamic force on blade #1. *
C *   FAER02 - Aerodynamic force on blade #2. *
C *   FSTAR - Complicated term. *
C *   I - Generic index. *
C *   IDENT - The identity matrix. *
C *   K - Generic index. *
C *   KINVR - Inverse of KSTAR matrix. *
C *   KSTAR - Complicated term. *
C *   L - Generic index. *
C *   M - Generic index. *
C *   NPTS - Number of points along the blade used to *
C *             perform Simpson's integration for calculat- *
C *             ing the moments and forces at the blade *
C *             root. (passed from STRAP1) *
C *   OMSGQR - Omega^2. *
C *   STEMP - Temporary array holding the tip displace- *
C *             *
```

```

C      *          ment values. The third dimension repre- *
C      *          sents the order of the time derivative. *
C      *          TIMNOW - Current time. *
C      *          *
C      *****

```

```

REAL      BLDANG
REAL      CPSI
REAL      SPSI
REAL      C1
REAL      C7
REAL      FAER01 (4)
REAL      FAER02 (4)
REAL      FAEROT (4)
REAL      FSTAR (4)
REAL      KINVR5 (4,4)
REAL      KSTAR (4,4)
REAL      STEMP (4,0:1,0:2)
REAL      TIMNOW

```

```

INTEGER   I
INTEGER   K
INTEGER   L
INTEGER   M
INTEGER   NPTS

```

```

INCLUDE   'C:\INCLUDE\AIRFRC.INC'
INCLUDE   'C:\INCLUDE\CONST2.INC'
INCLUDE   'C:\INCLUDE\MATRX1.INC'
INCLUDE   'C:\INCLUDE\MATRX2.INC'
INCLUDE   'C:\INCLUDE\POSITN.INC'
INCLUDE   'C:\INCLUDE\SARAYS.INC'
INCLUDE   'C:\INCLUDE\START.INC'
INCLUDE   'C:\INCLUDE\TURBN.INC'
INCLUDE   'C:\INCLUDE\FORMS.INC'

```

```

CHARACTER *2  ANS

```

```

CPSI = COS(PSI(1))
SPSI = SIN(PSI(1))
CTHP = COS(THETAP)
STHP = SIN(THETAP)
C1   = OMEGA*OMEGA*BETA0
C7   = OMEGA*OMEGA
C10  = CTHP*BETA0*(CPSI+TDELTA3*SPSI)

```

```

PRINT *, '
PRINT *, ' Do you want to input your own values for the'
PRINT *, ' initial modal deflections, or do you want'
PRINT *, ' SUBROUTINE STRTUP to calculate values for you?'
PRINT *, '
PRINT *, ' If you want to input your own then input a "Y"'
PRINT *, ' or a "y", If you want SUB STRTUP to do it'
PRINT *, ' input an "N" or an "n" > '
READ *, ANS
PRINT *, ANS

```

```

IF( ANS .EQ. 'N' .OR. ANS .EQ. 'n') THEN

```

```

C      Initialize some arrays.

```

```

DO 120 M=1,NSHAPS

```

```

    DO 100 I=0,2

```

```

        STEMP(M,0,I) = 0.0

```

```

        STEMP(M,1,I) = 0.0
100  CONTINUE
120  CONTINUE

C      Compute the FAERO values for use in calculating the FSTAR
C      values.  These are actually FAERO multiplied by the inverse
C      CMMASS matrix.

TIMNOW = TIME(1)
CALL NXTPHI ( TIMNOW )
BLDANG = PSI(1) + PI
      CALL FORM1 ( STEMP , NPTS , NSHAPS , BLDANG )
      THETAC = THETAP + BETA*SDEL3
      STHP = SIN(THETAC)
      CTHP = COS(THETAC)
      TTHP = TAN(THETAC)
      CALL AFORCE ( NPTS , BLDANG )
200  DO 200 M=1,NSHAPS
      FAERO2(M) = FAERO(M)
      BLDANG = PSI(1)
      CALL FORM1 ( STEMP , NPTS , NSHAPS , BLDANG )
      THETAC = THETAP + BETA*SDEL3
      STHP = SIN(THETAC)
      CTHP = COS(THETAC)
      TTHP = TAN(THETAC)
      CALL AFORCE ( NPTS , BLDANG )
210  DO 210 M=1,NSHAPS
      FAERO1(M) = FAERO(M)
220  DO 220 M=1,NSHAPS
      FAEROT(M) = 0.5*( FAERO1(M) + FAERO2(M)*(-1)**M )

C      Compute the FSTAR values.

      FSTAR(1) = GRAV*C10*CMGRV1(1)
      + FAEROT(1)
C
      DO 300 M=2,NSHAPS
      &      FSTAR(M) = FAEROT(M)
      &      - CTHP*0.1*( C1*CMRIGD(M))
300  CONTINUE

C      Compute the KSTAR values.

      DO 410 M=1,NSHAPS
      DO 400 K=1,NSHAPS
400  KSTAR(M,K) = CKBEND(M,K) + C7*CKTOMG(M,K)

410  CONTINUE

C      Invert the KSTAR matrix.

      CALL INVERT ( KSTAR , KINVRS , NSHAPS )

C      Compute S(K) = S(L) = ( KSTAR - INVERSE(L,M) )*FSTAR(M).

```



```

DO 510 L=1,NSHAPS
    SO(L) = 0.0
    DO 500 M=1,NSHAPS
500    SO(L) = SO(L) + KINVR(L,M)*FSTAR(M)
510 CONTINUE

SO(1) = -.8
SO(3) = -.1

PRINT *, 'Calculated values of SO:'
PRINT *, 'SO(1) = ', SO(1)
PRINT *, 'SO(2) = ', SO(2)
PRINT *, 'SO(3) = ', SO(3)
PRINT *, 'SO(4) = ', SO(4)

ELSE
    IF( NSHAPS .EQ. 1) THEN
        PRINT *, 'Read in SO(1) ( must be nonzero!) < '
        READ *, SO(1)
        PRINT *, SO(1)
        SO(2) = 0.
        SO(3) = 0.
        SO(4) = 0.
    ELSEIF( NSHAPS .EQ. 2) THEN
        PRINT *, 'Read in SO(1) and SO(2) < '
        PRINT *, ' (all must be nonzero)'
        READ *, SO(1), SO(2)
        PRINT *, SO(1)
        PRINT *, SO(2)
        SO(3) = 0.
        SO(4) = 0.
    ELSEIF( NSHAPS .EQ. 3) THEN
        PRINT *, 'Read in SO(1), SO(2), AND SO(3) < '
        PRINT *, ' (all must be nonzero)'
        READ *, SO(1), SO(2), SO(3)
        PRINT *, SO(1)
        PRINT *, SO(2)
        PRINT *, SO(3)
        SO(4) = 0.
    ELSEIF ( NSHAPS .EQ. 4) THEN
        PRINT *, 'Read in SO(1), SO(2), SO(3), AND SO(4) < '
        PRINT *, ' ( all must be nonzero)'
        READ *, SO(1), SO(2), SO(3), SO(4)
        PRINT *, SO(1)
        PRINT *, SO(2)
        PRINT *, SO(3)
        PRINT *, SO(4)
    ENDIF

ENDIF

C      Load the resulting static deflection values into the appropriate
C      tip data array and set all other values to zero.

DO 610 K=1,NSHAPS
    DO 600 I=0,1
        STEMP(K,I,0) = SO(K)
        STEMP(K,I,1) = 0.0
        STEMP(K,I,2) = 0.0
    600 CONTINUE
610 CONTINUE

RETURN
END
SUBROUTINE TRACE ( STEMP , NPTS )

```

```

C *****
C *
C * Subroutine TRACE prints out the values of certain vari- *
C * ables at each printout interval around the disk. The *
C * printout includes the current values of DELPSI, the *
C * blade tip acceleration, velocity and deflection, the *
C * error value and the values of the relative fluid veloc- *
C * ity in the edgewise and flapwise directions. This is *
C * done for each STEPMX station and is part of the diag- *
C * nostic features of the code. It is controlled by the *
C * TRACEF flag set in the subroutine LIMITS. *
C *
C *****
C
C *****
C *
C * External references in this routine: *
C *
C * none *
C *
C *****
C
C *****
C *
C * Named COMMON blocks used in this routine: *
C *
C * AIRFRC - Holds values used in aerodynamic calcula- *
C * tions. *
C * CONST - Turbine and other constants used in load *
C * calculations. *
C * POSITN - Holds parameters related to blade position *
C * such as PHI, PSI, etc. *
C * TURBN - Holds turbine parameters such as number of *
C * blades, rotor speed, etc. *
C * VREL1 - Holds blade section velocity components. *
C *
C *****
C
C *****
C *
C * Local and dummy variables used in this routine: *
C *
C * JCALLS - Print counter. *
C * K - Generic index. *
C * NEW - Array index for new data. *
C * NPTS - Number of points along the blade used to *
C * perform Simpson's integration for calculat- *
C * ing the moments and forces at the blade *
C * root. (passed from STRAP1) *
C * STEMP - Temporary array holding the tip displace- *
C * ment values. The third dimension repre- *
C * sents the order of the time derivative. *
C *
C * X - Psi in degrees. *
C * XX - DeltaPsi in degrees. *
C *
C *****

```

```

REAL      STEMP (4,0:1,0:2)
REAL      X
REAL      XX
REAL      STO
REAL      ST1
REAL      ST2

INTEGER   JCALLS
INTEGER   K
INTEGER   NEW
INTEGER   NPTS

INCLUDE   'C:INCLUDE\BLADE2.INC'

INCLUDE   'C:INCLUDE\AIRFRC.INC'

INCLUDE   'C:INCLUDE\CONST2.INC'

INCLUDE   'C:INCLUDE\POSITN.INC'

INCLUDE   'C:INCLUDE\TURBN.INC'

```

```

INCLUDE      'C:INCLUDE\VREL1.INC'

SAVE        JCALLS
SAVE        NEW

DATA        JCALLS / -1 /
DATA        NEW   / 1 /

1000 FORMAT( / ' Psi d-Psi   Accel  Velocity Deflect   Error
&              FAero VEta-Tip VZeta-Tip', ' Alpha', / )
1100 FORMAT( F5.0 , F6.2 , 2F10.3 , F8.3 , F11.6 , F10.3 , 2F9.3,
&   F9.4 )

```

C Get Psi and delta-Psi in degrees.

```

X = PSI(1)*RAD2DG
XX = DELPSI(1)*RAD2DG

```

C JCALLS is used to print column headings every ten lines.

```

JCALLS = JCALLS + 1

IF ( MOD( JCALLS , 10 ) .EQ. 0 ) WRITE (4,1000)

```

```

ST2 = STEMP(1,NEW,2)*CDELTA3*CTHP
ST1 = STEMP(1,NEW,1)*CDELTA3*CTHP
ST0 = STEMP(1,NEW,0)*CDELTA3*CTHP

```

```

ALPHA = ATAN(VZETA(NPTS)/VETA(NPTS))-THETA0(NPTS)

```

```

DO 100 K=1,NSHAPS

```

```

IF( K .EQ. 1 ) THEN
WRITE(4,1100) X , XX, ST2, ST1, ST0, ERROR, FAERO(K),
&   VETA(NPTS), VZETA(NPTS), ALPHA

```

```

ELSE
WRITE (4,1100) X , XX , STEMP(K,NEW,2) , STEMP(K,NEW,1)
&   , STEMP(K,NEW,0) , ERROR , FAERO(K) , VETA(NPTS)
&   , VZETA(NPTS), ALPHA

```

```

ENDIF
100 CONTINUE

```

```

RETURN
END

```

```

SUBROUTINE TRMTST ( ITRIM )

```

```

C        *****
C        *
C        * Subroutine TRMTST compares the results of two consecu- *
C        * tive rotor revolutions. The trim error condition is *
C        * satisfied if the value of QUANT2 times the trim error *
C        * fraction is less than the QUANT1 value. If any one az- *
C        * imuth location fails the trim test criterion, control *
C        * is returned to the calling routine and another disk *
C        * revolution is performed. If all the STEPMX stations *
C        * pass the test, then the trim solution exists and the *
C        * loads are computed. *
C        *
C        *****

```

```

C        *****
C        *
C        * External references in this routine: *
C        *
C        * none *
C        *
C        *****

```

```

C *****
C *
C * Named COMMON blocks used in this routine:
C *
C *   CONST - Turbine and other constants used in load
C *           calculations.
C *   LIMITC - Holds values used in the LIMITS routine.
C *   SARAYS - Holds new and old values for the general-
C *             ized coordinates.
C *   START - Holds initial blade deflection.
C *   TURBN - Holds turbine parameters such as number of
C *             blades, rotor speed, etc.
C *
C *****

```

```

C *****
C *
C * Local and dummy variables used in this routine:
C *
C *   I - Generic index.
C *   ISTPMX - STEPMX integerized.
C *   ITRIM - Flag that indicates that the trim solution
C *             has been completed when set to 0.
C *   NSHP - Counter on DO loops for the coordinate
C *             shape function.
C *   QUANT1 - This contains the sum of the differences
C *             between the blade tip deflections computed
C *             for this rotor revolution and the last rev-
C *             olution. This summation is also over the
C *             coordinate shapes used in this run.
C *   QUANT2 - This contains the sum of the blade tip de-
C *             flections computed by subroutine STRTUP for
C *             the static blade deflection of each coordi-
C *             nate function used in this run.
C *
C *****

```

```

REAL          QUANT1
REAL          QUANT2

INTEGER       I
INTEGER       ISTPMX
INTEGER       ITRIM
INTEGER       NSHP

INCLUDE      'C:INCLUDE\CONST2.INC'

INCLUDE      'C:INCLUDE\LIMITC.INC'

INCLUDE      'C:INCLUDE\SARAYS.INC'

INCLUDE      'C:INCLUDE\START.INC'

INCLUDE      'C:INCLUDE\TURBN.INC'

```

```

C Integerize the STEPMX interval.

```

```

ISTPMX = RAD2DG*STPEMX + 0.001

```

```

C Compute the sum of the tip deflections that were computed in
C subroutine STRTUP.

```

```

QUANT2 = 0.0

```

```

DO 100 NSHP=1,NSHAPS
100 QUANT2 = QUANT2 + SO(NSHP)

```

```

QUANT2 = ABS( QUANT2 )

```

```

C Check each STEPMX station around the disk.

```

```

DO 210 I=ISTPMX,360,ISTPMX

```

```

QUANT1 = 0.0

```

C Compute the sum of the differences between tip deflections
C for this rotor revolution and the last.

DO 200 NSHP=1,NSHAPS
200 QUANT1 = QUANT1 + SNEW(NSHP,0,I) - SOLD(NSHP,I)

C Compare the trim error fraction to the current deflection
C differences. If any station fails the test, return for
C another revolution.

IF(ABS(QUANT1) .GT. (0.01*QUANT2*TRMERR)) RETURN

210 CONTINUE

C All stations satisfied the trim error criterion. The trim solution
C is now complete. Set the ITRIM flag to zero to signal it. Loads
C can now be computed and the yaw run started.

ITRIM = 0

RETURN

END

FUNCTION TRPZOD (LOWLIM , BLTIP , FOF , NPTS)

```
C           *****  
C           *  
C           *   Function TRPZOD performs composite trapezoidal integra-   *  
C           *   tion on a set of data points transmitted from the   *  
C           *   calling routine. For derivation of the formula and   *  
C           *   limitations, see Carnahan, p. 78 (see full reference in   *  
C           *   comments for function SIMPSN). For computational effi-   *  
C           *   ciency, the interval width is not used in the formula-   *  
C           *   tion until the end when it is multiplied by the sum.   *  
C           *   *  
C           *   This function is similar to function TRAP in Module 1   *  
C           *   of the STRAP code.   *  
C           *   *  
C           *****  
  
C           *****  
C           *  
C           *   External references in this routine:   *  
C           *   *  
C           *   none   *  
C           *   *  
C           *****  
  
C           *****  
C           *  
C           *   Named COMMON blocks used in this routine:   *  
C           *   *  
C           *   none   *  
C           *   *  
C           *****  
  
C           *****  
C           *  
C           *   Local and dummy variables used in this routine:   *  
C           *   *  
C           *   BLTIP - Blade length. All integrations using this   *  
C           *   function are performed from the lower limit   *  
C           *   indexed by LOWLIM to the upper limit BLTIP.   *  
C           *   *  
C           *   FOF - Array of data points to be integrated. They   *  
C           *   are treated by this function as the value   *  
C           *   of a function evaluated at specific points.   *  
C           *   *  
C           *   H - Subinterval length given by H=(B-A)/N.   *  
C           *   I - Index into the FOF array.   *  
C           *   *  
C           *   LOWLIM - Index of the lower integration limit. The   *  
C           *   blade position indexed by LOWLIM is given   *  
C           *   by BLTIP*(LOWLIM-1)/(NPTS-1).   *  
C           *   *  
C           *   NPTS - Number of points along the blade used to   *  
C           *   perform Simpson's integration for calculat-   *
```

```

C      *           ing the moments and forces at the blade *
C      *           root. (passed from STRAP1) *
C      *      TRPZOD - Value of the integral from the blade posi- *
C      *           tion indexed by LOWLIM to BLTIP. *
C      * *
C      *****

```

```

REAL      BLTIP
REAL      FOF (201)
REAL      H
REAL      TRPZOD

INTEGER   I
INTEGER   LOWLIM
INTEGER   NPTS

```

```

C      Check to see if the lower and upper limits of integration are
C      the same. If so, the integral is zero.

```

```

IF ( LOWLIM .EQ. NPTS ) THEN

```

```

    TRPZOD = 0.0
    RETURN

```

```

END IF

```

```

C      Compute the distance between data points.

```

```

H = BLTIP/( NPTS - 1 )

```

```

C      Initialize integral to the contribution of the end points.

```

```

TRPZOD = 0.5*( FOF(LOWLIM) + FOF(NPTS) )

```

```

C      If there are only two data points, then we are done.

```

```

IF ( LOWLIM+1 .EQ. NPTS ) GO TO 20

```

```

C      Add in the contribution of the intermediate points.

```

```

DO 10 I=LOWLIM+1,NPTS-1
10 TRPZOD = TRPZOD + FOF(I)

```

```

C      Multiply by the interval width and return.

```

```

20 TRPZOD = H*TRPZOD

```

```

RETURN
END
SUBROUTINE VREL ( INBORD , NPTS , BLDANG )

```

```

C      *****
C      *
C      *      Subroutine VREL computes the relative velocity compo- *
C      *      nents for each of the blade stations. These values are *
C      *      needed to compute the aerodynamic forces. *
C      * *
C      *****

```

```

C      *****
C      *
C      *      External references in this routine: *
C      * *
C      *      INDUCD - Calculates the induced velocities along the *
C      *      blade. *
C      *      WINDVL - Computes wind velocity components. *
C      * *
C      *****

```

```

C *****
C
C *****
C *
C * Named COMMON blocks used in this routine: *
C *
C * BLADE - Holds blade property values such as stiff- *
C * ness and mass distributions. *
C * CONST - Turbine and other constants used in load *
C * calculations. *
C * DELTV - Holds turbine inputs for possible future *
C * use. Not currently used. *
C * FORMS - Holds blade deflections. *
C * POSITN - Holds parameters related to blade position *
C * such as PHI, PSI, etc. *
C * TURBN - Holds turbine parameters such as number of *
C * blades, rotor speed, etc. *
C * VINDUC - Holds induced velocity components. *
C * VREL1 - Holds blade section velocity components. *
C * WNDVEL - Holds values used in wind shear and tower *
C * shadow computations. *
C *****
C
C *****
C *
C * Local and dummy variables used in this routine: *
C *
C * BLDANG - Blade azimuth position used in the teeter- *
C * ing rotor option. *
C * CPSI - Cosine of the blade angle. *
C * I - Generic index. *
C * INBORD - The number of blade stations that are not *
C * on the airfoil section. INBORD is used in *
C * other routines to differentiate between *
C * sections of the blade with and without an *
C * airfoil section. *
C * NPTS - Number of points along the blade used to *
C * perform Simpson's integration for calculat- *
C * ing the moments and forces at the blade *
C * root. (passed from STRAP1) *
C * PHIDBR - Temporary storage. *
C * PHIDCT - Temporary storage. *
C * PHIDST - Temporary storage. *
C * QUANT1 - Temporary storage. *
C * QUANT2 - Temporary storage. *
C * QUANT3 - Temporary storage. *
C * QUANT4 - Temporary storage. *
C * R - Distance from hub axis. *
C * ROMEGA - Temporary storage. *
C * SPSI - Sine of the blade angle. *
C * SPSI2 - Temporary storage. *
C * SPSI3 - Temporary storage. *
C * STEP - Distance between points along the blade. *
C * X - The radial location along the blade (nondi- *
C * mensional). *
C * Z - Blade station pointer. Z goes from zero to *
C * the tip. *
C *****

```

```

REAL BLDANG
REAL CPHI
REAL CPSI
REAL PHIDBR
REAL PHIDCT
REAL PHIDST
REAL QUANT1
REAL QUANT2
REAL QUANT3
REAL QUANT4
REAL R
REAL ROMEGA
REAL SPSI
REAL SPSI2
REAL SPSI3
REAL STEP
REAL X
REAL Z

```

```

INTEGER      I
INTEGER      INBORD
INTEGER      NPTS

INCLUDE      'C:INCLUDE\BLADE2.INC'
INCLUDE      'C:INCLUDE\CONST2.INC'
INCLUDE      'C:INCLUDE\DELT.V.INC'
INCLUDE      'C:INCLUDE\FORMS.INC'
INCLUDE      'C:INCLUDE\POSITN.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\VINDUC.INC'
INCLUDE      'C:INCLUDE\VREL1.INC'
INCLUDE      'C:INCLUDE\WINDVEL.INC'

```

```

C      Compute the wind velocity components and then the induced
C      velocity components.

```

```

CALL WINDVL ( NPTS , BLDANG )
CALL INDUCD ( INBORD , NPTS , BLDANG )

```

```

C      Set up constants used in this computation.

```

```

CPSI  = COS( BLDANG )
SPSI  = SIN( BLDANG )
CPHI  = COS(PHI(0))
SPHI  = SIN(PHI(0))
PHIDBR = PHI(1)/OMEGA
PHIDST = PHIDBR*STHP*SPSI
PHIDCT = PHIDBR*CTHP*SPSI
ROMEGA = 1.0/RRROMGA
STEP  = BLTIP/( NPTS - 1 )
Z     = 0.0

```

```

C      Compute the velocity components VEta and VZeta.

```

```

DO 100 I=1,NPTS
  R = HUBRAD + Z

```

```

C      Clear the velocity components on the nonaerodynamic portion
C      of the blade.

```

```

IF ( I .LT. INBORD ) THEN
  VETA(I) = 0.0
  VZETA(I) = 0.0
ELSE
  X      = R/RR
  VETA(I) = -X*CTHC
  &      + V1(1,I)*ROMEGA*STHP
  VZETA(I) = -X*STHC - VE(1,I)*ROMEGA
  &      - V1(1,I)*ROMEGA*CTHP

  QUANT1 = VINDR(I) + ROMEGA*( VWIND(I) - HUBVEL )
  &      + ROMEGA*DELT.VY(I) - DVIND(I)
  QUANT2 = DELTVX(I)*ROMEGA*CPHI
  &      + SPHI*QUANT1
  QUANT3 = PHIDBR*( ABAR + BETA0*X )
  QUANT4 = -DELT.VZ(I)*ROMEGA
  &      + CHI*VINDR(I)*CPHI

  VETA(I) = VETA(I) - QUANT1*STHC*CPHI

```



```

&      + CPSI*( CTHP *( QUANT2 + QUANT3 )
&      + PHIDBR*VV(0,1)/RR )
&      + SPSI*(QUANT4*CTHP) + DELTVX(I)*ROMEGA*SPHI*STHP
&      + X*PHIDST

      VZETA(I) = VZETA(I) + QUANT1*CTHC*CPHI
&      + CPSI*(QUANT2 + QUANT3)*STHP
&      + SPSI*QUANT4*STHP - X*PHIDCT
&      - VINDR(I)*(VVSLP(I)+CTHP*(BETA0
&      + BETA*CELT3))*SPHI*SPSI
&      - DELTVX(I)*ROMEGA*CTHP*SPHI

C      Convert the velocity components into dimensional form
C      before returning to the calling routine. The VEta term
C      is negated to account for the sign convention that pos-
C      itive velocities are from the leading edge to the trail-
C      ing edge.

      VETA(I) = -VETA(I)*RROMGA
      VZETA(I) = VZETA(I)*RROMGA

      END IF

C      Increment the location along the blade and make another pass.

      Z = Z + STEP

100 CONTINUE

RETURN
END

SUBROUTINE WINDVL ( NPTS , BLDANG )

C      *****
C      *
C      * Subroutine WINDVL computes the wind velocity components *
C      * along the blade for a given time and azimuth position. *
C      * Provision has been made to include a time dependent *
C      * function for the hub velocity, but none is included *
C      * now. Likewise, time dependent functions deltaX, deltaY *
C      * and deltaZ can be included in the future in the section *
C      * set aside for them here. They are now set to zero. *
C      *
C      *****

C      *****
C      *
C      * External references in this routine: *
C      *
C      * none *
C      *
C      *****

C      *****
C      *
C      * Named COMMON blocks used in this routine: *
C      *
C      * BLADE - Holds blade property values such as stiff- *
C      * ness and mass distributions. *
C      * CONST - Turbine and other constants used in load *
C      * calculations. *
C      * DELTV - Holds turbine inputs for possible future *
C      * use. Not currently used. *
C      * POSITN - Holds parameters related to blade position *
C      * such as PHI, PSI, etc. *
C      * TURBN - Holds turbine parameters such as number of *
C      * blades, rotor speed, etc. *
C      * WIND - Holds wind shear and tower shadow parame- *
C      * ters. *
C      * WNDVEL - Holds values used in wind shear and tower *
C      * shadow computations. *
C      *
C      *****
C      *****

```

```

C      *
C      * Local and dummy variables used in this routine:
C      *
C      * BLDANG - Blade azimuth position used in the teeter-
C      *           ing rotor option.
C      * HH - Hub radius divided by its height.
C      * I - Generic index.
C      * J - Generic index.
C      * NPTS - Number of points along the blade used to
C      *           perform Simpson's integration for calculat-
C      *           ing the moments and forces at the blade
C      *           root.
C      * P - Complicated term.
C      * R - Location of the current point along the
C      *           blade.
C      * STEP - Distance between points along the blade.
C      *
C      *****

```

```

REAL      BLDANG
REAL      HH
REAL      P
REAL      R
REAL      STEP

INTEGER   I
INTEGER   J
INTEGER   NPTS

INCLUDE   'C:INCLUDE\BLADE2.INC'
INCLUDE   'C:INCLUDE\CONST2.INC'
INCLUDE   'C:INCLUDE\DELT.V.INC'
INCLUDE   'C:INCLUDE\POSITN.INC'
INCLUDE   'C:INCLUDE\TURBN.INC'
INCLUDE   'C:INCLUDE\WIND2.INC'
INCLUDE   'C:INCLUDE\WINDVEL.INC'
INCLUDE   'C:INCLUDE\TRBIN.F.INC'

```

```

C      Compute the wind velocity at the hub. A time dependent function
C      may be added in the future.

```

```

C      IF( ITURB .EQ. 1) THEN
C      HUBVEL = VHUB + VYNOWH
C      ELSE
C      HUBVEL = VHUB
C      ENDIF

```

```

C      Compute the wind shear components.

```

```

R      = HUBRAD
STEP = BLTIP/( NPTS - 1 )

DO 110 I=1,NPTS

  HH = R/HUBHT

  WSHR(0,I) = 0.25*SHERXP*( SHERXP - 1.0 )*HH**2
  WSHR(1,I) = SHERXP*HH
  WSHR(2,I) = 0.
  WSHR(3,I) = WSHR(0,I)
  WSHR(4,I) = 0.0
  WSHR(5,I) = 0.0

  R = R + STEP

  WSHEAR(I) = WSHR(0,I)

DO 100 J=1,5

```

```

100  WSHEAR(I) = WSHEAR(I) + WSHR(J,I)*COS( J*BLDANG )
110  CONTINUE

C      Compute tower shadow effects. When blade #1 is close to 360
C      degrees azimuth, the second blade must know that it is in the
C      tower shadow.

      IF ( ABS( BLDANG - PSISHD ) .LE. PSIZER ) THEN

          P      = KSHADW*PI/PSIZER
          TSHADW = TSUBO + TSUBP*COS( P*( BLDANG - PSISHD ) )

      ELSE IF ( ABS( BLDANG - PSISHD - 2*PI ) .LE. PSIZER ) THEN

          P      = KSHADW*PI/PSIZER
          TSHADW = TSUBO + TSUBP*COS( P*( BLDANG - PSISHD - 2*PI ) )

      ELSE

          TSHADW = 0.0

      END IF

      DO 200 I=1,NPTS
200    VWIND(I) = HUBVEL*( 1.0 + WSHEAR(I) - TSHADW )

C      *****
C      *
C      *   This place is reserved for future time dependent functions. *
C      *
C      *****

C      Input the time dependent windspeed fluctuations if
C      working on a turbulent analysis: ITURB .EQ. 1

      R = HUBRAD/(HUBRAD+BLTIP)
      STEP = (1.-R)/(NPTS-1)

C      Flag for deciding if working on a turbulence analysis.

      IF( ITURB .EQ. 1 ) THEN

C      Work on blade #2 first.

          IF( BLDANG .EQ. PSI(1)+PI) THEN

              DO 500 I = 1,NPTS

C      Linearly interpolate to get windspeed values at
C      intermediate blade radial stations.

                  IF(R .LE. 0.5) THEN
                      DELTVY(I)=(VYNOW2(1)-VYNOWH)*2.*R+VYNOWH
                  ELSE
                      DELTVY(I)=(VYNOW2(2)-VYNOW2(1))*2.*(R-.5)
                      &      + VYNOW2(1)
                  ENDIF
                  R=R+STEP
500          CONTINUE

C      Same procedure for blade #1.

          ELSEIF ( BLDANG .EQ. PSI(1)) THEN

              DO 600 I = 1,NPTS

                  IF(R .LE. 0.5) THEN
                      DELTVY(I)=(VYNOW1(1)-VYNOWH)*2.*R+VYNOWH
                  ELSE
                      DELTVY(I)=(VYNOW1(2)-VYNOW1(1))*2.*(R-.5)
                      &      + VYNOW1(1)

```

```

        ENDIF
        R=R+STEP
600    CONTINUE

    ENDIF

ENDIF

RETURN
END

SUBROUTINE SERIES(F,M,N,A,B,NST,IPS,IPR)
REAL F(361,21,9)
REAL A(0:10,21)
REAL B(0:10,21)
REAL FACTOR
REAL FI

INTEGER IPS(360)
INTEGER M
INTEGER N
INTEGER NST
INTEGER IPR

FACTOR = FLOAT(IPR)/180.

DO 100 I = 0,N

    FI = FLOAT(I)

    DO 80 J = 1,21

        SUMA2 = 0.
        SUMB2 = 0.

        DO 75 IST = 2,NST-1
            PSI = IPS(IST)* .017453293
            SUMA2 = SUMA2 + COS(FI*PSI)*F(IST,J,M)
            SUMB2 = SUMB2 + SIN(FI*PSI)*F(IST,J,M)
75        CONTINUE

        PSI1 = 0.
        PSIN = 360.0*.017453293

        SUMA = 2.*SUMA2 + COS(FI*PSI1)*F(1,J,M) +
&        COS(FI*PSIN)*F(NST,J,M)

        SUMB = 2.*SUMB2 + SIN(FI*PSI1)*F(1,J,M) +
&        SIN(FI*PSIN)*F(NST,J,M)

        IF( I .EQ. 0) THEN
            A(I,J) = 0.25*FACTOR*SUMA
            B(I,J) = 0.25*FACTOR*SUMB
        ELSE
            A(I,J) = 0.5*FACTOR*SUMA
            B(I,J) = 0.5*FACTOR*SUMB
        ENDIF

80    CONTINUE
100 CONTINUE

RETURN
END

```

```

SUBROUTINE TRBCLC( NPTS )

```

```

C      Subroutine for calculation of turbulence. We will assume that
C      turbulence is being modeled from rotationally sampled wind data
C      as predicted from the VEERS Wind Simulation model.
C      The file of data must contain 6 columns ( as evidenced in the
C      read statement below.
C      The first column is blade azimuth angle, the second column
C      is the windspeed for blade #1 at the 50% rotor radius station,
C      the third column is the windspeed for blade #1 at the tip,
C      the fourth column is the windspeed for blade #2 at the 50%
C      rotor radius location, the fourth column is for blade #2 at
C      the tip, and finally the fifth column is the windspeed at the

```

C center of the hub.
 C Before this windspeed data is read in, the user is asked for the
 C name of the file containing the turbulent windspeed data, is asked
 C whether this data is in metric (meters/sec) or english units
 C (ft/s) and is asked for the name of the file to contain the
 C predicted turbulent loads data. For the structure of this file,
 C see the block of comments near the WRITE statement toward the
 C end of this subroutine.

C BE SURE TO RUN THE TRIM SOLUTION CORRESPONDING WITH THIS
 C TURBULENCE ANALYSIS WITH A VALUE OF SHERXP EQUAL TO ZERO
 C IF THE VEERS GENERATED WINDSPEED DATA ALREADY HAS THE
 C EFFECTS OF WINDSHEAR. IF YOU DON'T THEN THE PREDICTIONS
 C WILL HAVE TOO MUCH CYCLIC CONTENT, DUE TO TWO TIMES THE
 C DESIRED WINDSHEAR VALUES.

```
REAL STEM (4, 0:1, 0:2)
REAL ANGLE
REAL AZIDAT
REAL AZIINC
REAL AZIOLD
REAL BLDANG
REAL DELVY1 (10)
REAL DELVY2 (10)
REAL HUBLOD ( 6)
REAL PSIDIF
REAL RESLTS (2,21,9)
REAL SHFLOD ( 6)
REAL TEETER
REAL VY1 (10)
REAL VY2 (10)
REAL VYHUB
REAL VYSAV1 (0:1,10)
REAL VYSAV2 (0:1,10)
REAL VYSAVH (0:1)
```

```
INTEGER I
INTEGER J
INTEGER NEW
INTEGER NPTS
INTEGER NSHP
INTEGER OLD
```

```
CHARACTER *50 WDNFIL
CHARACTER *50 LODFIL
CHARACTER * 2 ANS
```

```
INCLUDE 'C:\INCLUDE\BLADE2.INC'
INCLUDE 'C:\INCLUDE\LIMTC.INC'
INCLUDE 'C:\INCLUDE\POSITN.INC'
INCLUDE 'C:\INCLUDE\SARAYS.INC'
INCLUDE 'C:\INCLUDE\TURBN.INC'
INCLUDE 'C:\INCLUDE\TRBINF.INC'
INCLUDE 'C:\INCLUDE\CONST2.INC'
INCLUDE 'C:\INCLUDE\DELTV.INC'
INCLUDE 'C:\INCLUDE\WIND2.INC'
```

```
SAVE NEW
SAVE OLD
SAVE STEM
```

```
DATA NEW / 1 /
DATA OLD / 0 /
```

```
PRINT *, ' '
PRINT *, 'Turbulence Analysis Run Set-up'
PRINT *, ' '
```

C Read in the name of the file containing the VEERS generated
 C windspeed data.

```
PRINT *, 'Read in the name of the wind residual time series file'
READ *, WDNFIL
PRINT *, WDNFIL
```

C See if the windspeed data is in metric or english units.

```

PRINT *, 'Is the wind data in metric units (meters/sec)?'
READ *, ANS
PRINT *, ANS
PRINT *, ' '

```

C Give the name of the file you want to write rotor and blade
C teeter, loads, shaft-loads, etc. out to.

```

PRINT *, 'Read in the name of the file for loads output'
READ *, LODFIL
PRINT *, LODFIL

```

```

OPEN (15, FILE = WNDFIL, STATUS = 'UNKNOWN')
OPEN (16, FILE = LODFIL, STATUS = 'UNKNOWN')

```

```

NSCN = 1
AZIDAT = 0.

```

C AZIDAT is the azimuth angle read from data,
C VY1(J) is the windspeed data
C for blade number 1, and VY2(J), is the windspeed data
C for blade number 2, and VYHUB is the hubcenter windspeed.
C Currently MSTAT is limited to 2. We usually read in windspeed
C data at the 50% and 100% rotor radius locations, for each
C blade and then the hubcenter windspeed.

C Set the shear exponent from the trim run to
C zero, since the file generated from the
C VEERS model already has the effects of windshear
C in it. Failure to do so will result in too much
C windshear. Other parameters such as tower shadow
C inputs are not set to zero unless the user
C reruns the trim solution with zero values
C for Tp, T0 - the tower shadow parameters.

C Initialize values for SHEREXP, PSI(0),
C AZIOLD, ERROR, DELPSI(1),
C STEMP, etc.

```

SHERXP = 0.
PSI(0) = 0.
AZIOLD = -0.0001
ERROR = 0.
DELPSI(1) = STEPMN
AZIINC = TIMINC*OMEGA

```

c set the initial blade deflection, velocity,
c and acceleration values to what was calculated
c in the trim solution at zero azimuth angle. This
c serves to start the solution process.

```

DO 260 NSHP = 1, NSHAPS
DO 250 I = 0, 2
    STEMP(NSHP,OLD,I) = SNEW(NSHP, I , 360)
250 CONTINUE
260 CONTINUE

```

C Set the left-hand endpoint windspeed value
C needed in the linear interpolation process
C equal to VHUB since we have not read in
C a windspeed data line from the VEERS file yet.
C The right-hand endpoint value will be the
C new line of windspeed data read in.
C These values are necessary in order to have
C endpoints for interpolation, since the blade
C azimuth in the numerical integration process
C will lie between azimuth values read from
C the data (AZIDAT). Setting this first left-
C hand point to VHUB starts the process. I=0
C stands for the left-hand endpoint value,
C while I=1 is the right-hand endpoint.
C VYSAV1 is for blade 1, VYSAV2 is for blade
C 2, and VYSAVH is the hub-center. J is equal
C to 1 or 2 depending on whether it's the 50%
C station or the blade tip.

```

        DO 300 J = 1,MSTAT
          VYSAV1(0,J) = 0.
          VYSAV2(0,J) = 0.
300    CONTINUE
        VYSAVH(0) = 0.

1000 READ (15,*) AZIDAT, (VY1(J), J=1,MSTAT)
      & , (VY2(J), J=1,MSTAT), VYHUB

      IF (( ANS .EQ. 'Y') .OR. ( ANS .EQ. 'y')) THEN
        DO 220 J = 1,MSTAT
          VY1(J) = 3.28*VY1(J)
          VY2(J) = 3.28*VY2(J)
220    CONTINUE
          VYHUB = 3.28*VYHUB
        ENDIF

C      Convert AZIDAT to radians

      AZIDAT = AZIDAT/RAD2DG
      IF( AZIDAT .EQ. 0.) THEN
        PSI(0)=6.283185-AZIINC
      ENDIF

C      Be sure that the windspeed data produced by the VEERS model
C      contains the hubheight or mean windspeed. If it doesn't then
C      the next 4 lines of code should be commented out.
C      The purpose of the next 4 lines of code is
C      to subtract out the mean windspeed from the data input.

      DO 230 J = 1,MSTAT
        VY1(J) = VY1(J)-VHUB
        VY2(J) = VY2(J)-VHUB
230    CONTINUE
        VYHUB = VYHUB-VHUB

C      We read in one line of windspeed data at a time.
C      We interpolate the windspeed data to get turbulent
C      windspeed fluctuations when the blade azimuth angle
C      lies between successive values in the input file.

C      Set the right-hand endpoint windspeed in the
C      interpolation process equal to the windspeeds
C      just read in.

      DO 350 J = 1, MSTAT
        VYSAV1(1,J) = VY1(J)
        VYSAV2(1,J) = VY2(J)
350    CONTINUE
        VYSAVH(1) = VYHUB

C      If we have passed through 360 degrees
C      then subtract out 360 degrees from
C      PSI. We are not able to deal with
C      azimuth angles greater than 360 deg.
C      Be sure to convert to radians.

      IF( (AZIDAT - AZIOLD) .LT. 0.) THEN
        PSI(0) = PSI(0) - 360./RAD2DG
        AZIOLD = AZIOLD - 360./RAD2DG
      ENDIF

C      Find the next azimuth angle in the
C      code numerical solution process ( not
C      the azimuth angle of the next line
C      of wind data. PSI(1) represents
C      this new azimuth angle (radians).
C      PSIDIF represents the difference between
C      this new azimuth location and the last
C      azimuth angle read in from the data.

400    CALL NXTAZI ( AZIDAT)

        PSIDIF = PSI(1) - AZIOLD

```

C Interpolate in order to determine turbulent windspeed
 C data for azimuth angles lying between the previously
 C read in line of data and the new line of data.
 C VYNOW1(J) is interpolated windspeed for blade 1 at
 C this new azimuth angle. VYNOW2(J) is the interpolated
 C windspeed for blade 2. VYNOWH is the interpolated
 C hub center windspeed for this azimuth position.
 C These windspeed values are the ones that ultimately
 C get passed into the WINDVL subroutine.
 C The windspeed values for blade radial stations lying
 C between the hub, 50% radial station and blade tip
 C get formed in WINDVL by linear interpolation along
 C the blade span. If the user wants to read windspeed
 C data at more than
 C these stations then SUBROUTINE WINDVL will have
 C to be changed and more columns of data would have
 C to be read in from the VEERS file.

```

DO 450 J = 1, MSTAT
  DELVY1(J) = VYSAV1(1,J) - VYSAV1(0,J)
  DELVY2(J) = VYSAV2(1,J) - VYSAV2(0,J)
  VYNOW1(J) = (DELVY1(J)/AZIINC)*PSIDIF + VYSAV1(0,J)
  VYNOW2(J) = (DELVY2(J)/AZIINC)*PSIDIF + VYSAV2(0,J)
450 CONTINUE
DELVYO = VYSAVH(1) - VYSAVH(0)
VYNOWH = (DELVYO/AZIINC)*PSIDIF + VYSAVH(0)

CALL EULER ( STEMP, NSHAPS, NPTS)

IF(( ERROR .GT. EUERR) .AND. ( DELPSI(1) .GT. STEPMM))
&   GO TO 400

```

```

DO 510 NSHP = 1, NSHAPS
DO 500 I = 0,2
  STEMP ( NSHP, OLD, I) = STEMP(NSHP, NEW, I)
500 CONTINUE
510 CONTINUE

```

```

DELPSI(0) = DELPSI(1)
DELTAT(0) = DELTAT(1)
PSI(0)    = PSI(1)

IF( ABS(PSI(1)-AZIDAT) .GE. 0.1*STEPMM) THEN
  GO TO 400
ELSE

```

```

&   TEETER = STEMP(1,NEW,0)*RAD2DG
      / (HUBRAD+BLTIP)

```

```

ANGLE = AZIDAT*RAD2DG

512 IF( ANGLE .LT. 360) GO TO 515
    ANGLE = ANGLE - 360.
    GO TO 512

```

C Set the blade angle BLDANG equal to the
 C azimuth angle of blade #2.

```

515 BLDANG = PSI(1)+PI

```

C First calculate blade loads for blade #2.
 C The response and load results for blade
 C #2 will be stored in the variable
 C RESULTS(2,I,J)

```

CALL LODCLC(BLDANG, STEMP, RESLTS)

```

C Set the blade angle BLDANG equal to the
 C azimuth angle for the first blade.

```

BLDANG = PSI(1)

```

C Calculate blade loads for blade #1.
 C The response and load results for blade
 C #1 will be stored in the variable
 C results(1,I,J).

CALL LODCLC(BLDANG, STEMP, RESLTS)

C Calculate the shaft loads once the
C loads for blades #1 and #2 are known.

CALL SHAFT2(RESLTS, HUBLOD, SHFLOD, PSI)

C This is the portion of the subroutine that writes
C out results to the user designated loads output
C file. ANGLE is the azimuth angle, TEETER is the
C rotor teeter angle (deg.), RESLTS(1,I,J) is the
C results matrix for blade #1 at the I'th station.
C J corresponds to the results item to printed out:
C J=1 to 7.
C RESLTS(2,I,J) is the results matrix for blade #2.
C I corresponds to the number of the blade radial
C station, from 1 to 21, with 1 at the root and
C 21 at the blade tip.

C J=1 BLADE DEFLECTION (FT.)
C J=2 BLADE SLOPE (FT./FT.)
C J=3 BLADE FLAP+TEETER VELOCITY (FT/SEC)
C J=4 BLADE TENSION (LB.)
C J=5 BLADE EDGEWISE SHEAR FORCE (LB.)
C J=6 BLADE FLAPWISE SHEAR FORCE (LB.)
C J=7 BLADE FLAPWISE MOMENT (FT-LB)
C J=8 BLADE EDGEWISE MOMENT (FT-LB)
C J=9 BLADE TORSIONAL MOMENT (FT-LB)

C SHFLOD(1) IS THE SHAFT FORCE IN THE Xr DIRECTION.
C SHFLOD(2) IS THE SHAFT FORCE IN THE Yr DIRECTION.
C SHFLOD(3) IS THE SHAFT FORCE IN THE Zr DIRECTION.
C SHFLOD(4) IS THE SHAFT MOMENT ABOUT THE Xr AXIS.
C SHFLOD(5) IS THE SHAFT MOMENT ABOUT THE Yr AXIS.
C SHFLOD(6) IS THE SHAFT MOMENT ABOUT THE Zr AXIS.

C Here we are printing out azimuth, teeter
C root flap-bending for blades 1 and 2,
C 65% span flap-bending for blade 1,
C and shaft moments about the Xr, Yr, and
C Zr axes. This statement will need to be
C changed by the user to write out items
C needed.

WRITE(16,*) ANGLE, TEETER, RESLTS(1,1,7),
& RESLTS(2,1,7),
& RESLTS(1,13,7), SHFLOD(4), SHFLOD(5),
& SHFLOD(6)

C Set the left-hand endpoint windspeeds to the right-
C hand values and read in a new line of windspeed
C data from the
C VEERS generated windspeed file, until NSCN is
C equal to NUMSCN.

DO 520 J = 1, MSTAT
520 VYSAV1(0,J) = VYSAV1(1,J)
VYSAV2(0,J) = VYSAV2(1,J)
VYSAVH(0) = VYSAVH(1)
NSCN = NSCN + 1
AZIOLD = AZIDAT
IF(NSCN .LE. NUMSCN) GO TO 1000
ENDIF
RETURN
END

SUBROUTINE NXTAZI (AZI)

C *****
C *
C * Subroutine NXTAZI computes the next value of the azi- *

```

C      * muth angle, Psi, based on the current value of the Euler error and the previous delta-Psi value. If the error is too large, a smaller delta-Psi will be used to compute the value of Psi, unless the delta-Psi is already at the STEPMM lower limit. Likewise, if the Euler error is below a certain limit, the delta-Psi value is increased for the next Psi computation.
C      *
C      *****

```

```

C      *****
C      *
C      *   External references in this routine:
C      *
C      *       none
C      *
C      *****

```

```

C      *****
C      *
C      *   Named COMMON blocks used in this routine:
C      *
C      *       CONST - Turbine and other constants used in load calculations.
C      *       LIMITC - Holds values used in the LIMITS routine.
C      *       POSITN - Holds parameters related to blade position such as PHI, PSI, etc.
C      *       TURBN - Holds turbine parameters such as number of blades, rotor speed, etc.
C      *
C      *****

```

```

C      *****
C      *
C      *   Local and dummy variables used in this routine:
C      *
C      *       AZI - The next STEPMX station .
C      *       NEW - Array index for new data.
C      *       OLD - Array index for old data.
C      *
C      *****

```

```

REAL      AZI

INTEGER   NEW
INTEGER   OLD

INCLUDE   'C:\INCLUDE\CONST2.INC'
INCLUDE   'C:\INCLUDE\LIMITC.INC'
INCLUDE   'C:\INCLUDE\POSITN.INC'
INCLUDE   'C:\INCLUDE\TURBN.INC'

SAVE     NEW
SAVE     OLD

DATA     NEW      / 1 /
DATA     OLD      / 0 /

```

```

C      Check the error. If it is too large, decrease delta-Psi by 50%. If it is very small, increase delta-Psi by 50%. The value of delta-Psi must always be greater than STEPMM.
C
C

```

```

IF ( ERROR .GT. EUERR ) THEN

    DELPSI(NEW) = AMAX1( 0.5*DELPSI(NEW) , STEPMM )

ELSE IF ( ERROR .LT. 0.1*EUERR ) THEN

    DELPSI(NEW) = 1.5*DELPSI(NEW)

END IF

```

```

C      This section of code checks the position of the new Psi against
C      the next STEPMX station. If the next Psi will go past the
C      STEPMX station, the delta-Psi is adjusted to bring the next Psi
C      directly onto the STEPMX point. If the next Psi will fall just
C      short of a STEPMX station, the delta-Psi is adjusted to assure
C      that it will not be necessary to use a delta-Psi smaller than
C      STEPMN to reach the next STEPMX station on the next iteration.

```

```

C      Programmer note:

```

```

C      The following logic can produce a delta-Psi that is less
C      than STEPMN for two steps. It would then move back within
C      tolerances. This case comes up when we are within slightly
C      less than two STEPMNs of the next STEPMX station. This
C      algorithm cannot produce a delta-Psi that is less than
C      STEPMN/2.                                MLB

```

```

IF ( PSI(OLD)+DELPSI(NEW) .GT. AZI ) THEN

```

```

    DELPSI(NEW) = AZI - PSI(OLD)

```

```

ELSE IF ( PSI(OLD)+DELPSI(NEW)+STEPMN .GT. AZI ) THEN

```

```

    IF ( (DELPSI(NEW) .NE. STEPMN) .OR. (ERROR .LE. EUERR) ) THEN

```

```

        DELPSI(NEW) = 0.5*( AZI - PSI(OLD) )

```

```

    END IF

```

```

END IF

```

```

C      Set new Psi and delta-Time.

```

```

PSI(NEW)    = PSI(OLD) + DELPSI(NEW)
DELTAT(NEW) = DELPSI(NEW)/OMEGA

```

```

C      Get time values if we're working on a yawing solution.

```

```

IF ( ITRIM .EQ. 0 ) TIME(NEW) = TIME(OLD) + DELTAT(NEW)

```

```

RETURN
END

```

```

SUBROUTINE LODCLC( BLDANG, STEMP, RESLTS)

```

```

C      *****
C      *
C      * Subroutine LODCLC is used to compute blade flap loads. *
C      *
C      *****

```

```

C      *****
C      *
C      * External references in this routine:
C      *
C      * TRPZOD - Composite trapezoidal integration.
C      *
C      *****

```

```

C      *****
C      *
C      * Named COMMON blocks used in this routine:
C      *
C      * AERO1 - Holds coefficients related to aerodynamic
C      *        loads calculations such as ClAlpha, CdZero,
C      *        etc.
C      * AIRFRC - Holds values used in aerodynamic calcula-
C      *        tions.
C      * BLADE - Holds blade property values such as stiff-
C      *        ness and mass distributions.
C      * CONST - Turbine and other constants used in load
C      *        calculations.
C      * LIMITC - Holds values used in the LIMITS routine.
C      *
C      *****

```

```

C      *      LODVAL - Holds values used to compute blade loads.      *
C      *      POSITN - Holds parameters related to blade position    *
C      *      such as PHI, PSI, etc.                                  *
C      *      SHAPE - Holds blade coordinate shape functions.        *
C      *      TENSIN - Holds tension component integrals.           *
C      *      TURBN - Holds turbine parameters such as number of    *
C      *      blades, rotor speed, etc.                              *
C      *      WIND - Holds wind shear and tower shadow parame-     *
C      *      ters.                                                  *
C      *
C      *****

```

```

C      *****
C      *
C      *      Local and dummy variables used in this routine:      *
C      *
C      *      BLDANG - Blade azimuth position.                       *
C      *      COEF1 - Complicated term.                              *
C      *      COEF2 - Complicated term.                              *
C      *      COEF4 - Complicated term.                              *
C      *      COEF7 - Complicated term.                              *
C      *      CPSI - Cosine of the blade angle.                      *
C      *      FACTR1 - Complicated term.                             *
C      *      FINTGR - Integral of FLAPFN.                           *
C      *      FLAPFN - Flapwise shear force distribution.            *
C      *      I - Generic index.                                     *
C      *      IPT - Data point index.                                *
C      *      J - Generic index.                                     *
C      *      LL - Generic index.                                    *
C      *      M - Generic index.                                     *
C      *      NPTS - Number of points along the blade used to      *
C      *      perform Simpson's integration for calculat-          *
C      *      ing the moments and forces at the blade               *
C      *      root. (passed from STRAP1)                             *
C      *      NSHP Counter on DO loops for the coordinate          *
C      *      shape function.                                        *
C      *      OMGASQ - Omega^2.                                     *
C      *      PRDCT0 - Temporary storage.                            *
C      *      PRDCT1 - Temporary storage.                            *
C      *      PRDCT2 - Temporary storage.                            *
C      *      PRODD - Temporary storage.                             *
C      *      PYAERO - Aerodynamic loads in the flapwise direc-    *
C      *      tion.                                                 *
C      *      RESLTS - Temporary storage.                            *
C      *      SPSI - Sine of the blade angle.                       *
C      *      TINTGR - Integral of TSNFCN.                           *
C      *      TSNFCN - Product of the tension force and the slope   *
C      *      at a blade station.                                    *
C      *
C      *****

```

```

REAL      BLDANG
REAL      COEF1
REAL      COEF2
REAL      COEF4
REAL      COEF7
REAL      CPSI
REAL      FACTR1
REAL      FINTGR
REAL      FLAPFN (21)
REAL      EDGEFN (21)
REAL      TSNFCN (21)
REAL      VXFCN (21)
REAL      OMGASQ
REAL      PRODD
REAL      PRDCT0
REAL      PRDCT1
REAL      PRDCT2
REAL      PRODD (21)
REAL      PYAERO (21)
REAL      PXAERO (21)
REAL      QZFCN (21)
REAL      RESLTS (2,21,9)
REAL      SPSI
REAL      STEMP (4, 0:1, 0:2)
REAL      TINTGR
REAL      TRPZOD

```

```

INTEGER      IDBLD
INTEGER      IPT
INTEGER      NPTS
INTEGER      NSHP

INCLUDE      'C:INCLUDE\AERO1.INC'
INCLUDE      'C:INCLUDE\AIRFRC.INC'
INCLUDE      'C:INCLUDE\BLADE2.INC'
INCLUDE      'C:INCLUDE\CONST2.INC'
INCLUDE      'C:INCLUDE\LIMITC.INC'
INCLUDE      'C:INCLUDE\LODVAL.INC'
INCLUDE      'C:INCLUDE\POSITN.INC'
INCLUDE      'C:INCLUDE\SHAPE.INC'
INCLUDE      'C:INCLUDE\TENSIN2.INC'
INCLUDE      'C:INCLUDE\TURBN.INC'
INCLUDE      'C:INCLUDE\WIND2.INC'

SAVE        NPTS
SAVE        NEW
SAVE        OLD

DATA        NPTS    / 21 /
DATA        NEW     / 1  /

```

C Initialize some constants

```

OMGASQ = OMEGA**2
COEF1  = 2.0*OMEGA*STHP
COEF4  = OMGASQ*STHP*CTHP
COEF6  = OMGASQ*CTHP*CTHP
COEF7  = OMGASQ*STHP*STHP

CPSI   = COS( BLDANG )
SPSI   = SIN( BLDANG )
FACTR1 = OMGASQ*BETA0 + 2*OMEGA*CPSI*PHI(1) + SPSI*PHI(2)
COEF2  = GRAV*( -CHI*CTHP + STHP*SPSI + BETA0*CTHP*CPSI )
COEF3  = GRAV*( CHI*STHP + CTHP*SPSI - BETA0*STHP*CPSI )
COEF5  = GRAV*CPSI
COEF8  = GRAV*SPSI*STHP

PRODD  = 0.5*RHOAIR*CSUBMA

IF( BLDANG .EQ. PSI(1)+PI) THEN
  IDBLD = 2
ELSE
  IDBLD = 1
ENDIF

DO 350 IPT = 1, NPTS

  RESLTS(IDBLD,IPT,1) = 0.
  RESLTS(IDBLD,IPT,2) = 0.
  RESLTS(IDBLD,IPT,3) = 0.

  DO 300 NSHP = 1, NSHAPS

    IF( BLDANG .EQ. PSI(1)+PI) THEN
      MARK = (-1.)**NSHP
    ELSE
      MARK = 1.
    ENDIF

    RESLTS(IDBLD,IPT,1) = RESLTS(IDBLD,IPT,1)+MARK

```

```

&      *SHAPE(NSHP,0,IPT)*STEMP(NSHP,NEW,0)
      RESLTS(IDBLD,IPT,2) = RESLTS(IDBLD,IPT,2)+MARK
&      *SHAPE(NSHP,1,IPT)*STEMP(NSHP,NEW,0)
      RESLTS(IDBLD,IPT,3) = RESLTS(IDBLD,IPT,3)+MARK
&      *SHAPE(NSHP,0,IPT)*STEMP(NSHP,NEW,1)
300  CONTINUE

      DMSBAC      = PRODD*CHORD(IPT)*WW(IPT)**2

      IF( BLDANG .EQ. PSI(1)+PI) THEN
        PYAERO(IPT) = TRPZOD( IPT, BLTIP, DAZET2, NPTS)
        PXAERO(IPT) = TRPZOD( IPT, BLTIP, DAET2, NPTS)
        QZFCN(IPT) = DMSBAC + ESUBAC(IPT)*DAZET2(IPT)
      ELSE
        PYAERO(IPT) = TRPZOD( IPT, BLTIP, DAZET1, NPTS)
        PXAERO(IPT) = TRPZOD( IPT, BLTIP, DAET1, NPTS)
        QZFCN(IPT) = DMSBAC + ESUBAC(IPT)*DAZET1(IPT)
      ENDIF

      PRDCT0 = 0.
      PRDCT1 = 0.
      PRDCT2 = 0.
      PRODC(IPT) = 0.

      DO 340 NSHP = 1, NSHAPS

        IF( BLDANG .EQ. PSI(1)+PI)THEN
          MARK = (-1.)**NSHP
        ELSE
          MARK = 1.
        ENDIF

        PRDCT0 = PRDCT0 +MARK*STEMP(NSHP,NEW,0)*TCORLS(NSHP,IPT)
        PRDCT1 = PRDCT1 +MARK*STEMP(NSHP,NEW,1)*TCORLS(NSHP,IPT)
        PRDCT2 = PRDCT2 +MARK*STEMP(NSHP,NEW,2)*TCORLS(NSHP,IPT)
        PRODC(IPT) = PRODC(IPT) + MARK*STEMP(NSHP,NEW,0)
          * CIFMOM(NSHP, IPT)
&
340  CONTINUE

      RESLTS(IDBLD,IPT,4) = OMGASQ * TOMGA(IPT) + COEF1 * PRDCT1
&      - COEF5 * TGRAV(IPT)

      RESLTS(IDBLD,IPT,5) = PXAERO(IPT) + FACTR1 * STHP * TOMGA(IPT)
&      + COEF4 * PRDCT0 + COEF3 * TGRAV(IPT)
&      + COEF6 * OFFMAS(IPT)

      RESLTS(IDBLD,IPT,6) = PYAERO(IPT) - FACTR1 * CTHP * TOMGA(IPT)
&      + COEF7 * PRDCT0 - PRDCT2 + COEF2 * TGRAV(IPT)
&      + COEF4 * OFFMAS(IPT)

      EDGEFN(IPT) = RESLTS(IDBLD,IPT,5)
      VXFCN (IPT) = EDGEFN(IPT)*RESLTS(IDBLD,IPT,2)
      TSNFCN(IPT) = RESLTS(IDBLD,IPT,4) * RESLTS(IDBLD,IPT,2)
      FLAPFN(IPT) = RESLTS(IDBLD,IPT,6)
350  CONTINUE

      DO 360 IPT = 1,NPTS

        EINTGR = TRPZOD(IPT, BLTIP, EDGEFN, NPTS)
        SINTGR = TRPZOD(IPT, BLTIP, VXFCN , NPTS)
        TINTGR = TRPZOD(IPT, BLTIP, TSNFCN, NPTS)
        QZAERO = TRPZOD(IPT, BLTIP, QZFCN , NPTS)
        FINTGR = TRPZOD(IPT, BLTIP, FLAPFN, NPTS)
        ECENGR = TRPZOD(IPT, BLTIP, ECNTFN, NPTS)

        RESLTS(IDBLD,IPT,7) = TINTGR - FINTGR + COEF7*PRODC(IPT)
        RESLTS(IDBLD,IPT,8) = EINTGR - OMGASQ*ECENGR
&      - COEF4*PRODC(IPT) + COEF5*OFFMAS(IPT)
&
        RESLTS(IDBLD,IPT,9) = QZAERO - SINTGR + COEF4*DELTIM(IPT)
&      + COEF8*OFFMAS(IPT)

360  CONTINUE

      RETURN
      END

```

SUBROUTINE SHAFT(IPS,IPR,NSTN)

```
INCLUDE      'C:INCLUDE\CONST2.INC'  
INCLUDE      'C:INCLUDE\TURBN.INC'  
INCLUDE      'C:INCLUDE\RESLTS.INC'  
INCLUDE      'C:INCLUDE\SHFLOD.INC'  
INCLUDE      'C:INCLUDE\BLADE2.INC'  
INCLUDE      'C:INCLUDE\HUBPRP2.INC'
```

```
CHARACTER * 50 SHFOUT  
CHARACTER * 15 LABEL1  
CHARACTER * 10 LABEL2(6)  
CHARACTER * 12 LABEL3(0:5)  
CHARACTER * 12 LABEL4(0:5)  
CHARACTER * 10 LABEL5(6)
```

```
INTEGER IPS(360)  
INTEGER IPR  
INTEGER NSTN  
INTEGER NSTN1  
INTEGER NST1  
INTEGER IST  
INTEGER IST2
```

```
REAL A(0:10,6)  
REAL B(0:10,6)  
REAL C(0:10,6)  
REAL D(0:10,6)  
REAL SBETA0  
REAL CBETA0  
REAL FACTOR  
REAL FI  
REAL PSI1  
REAL PSIN
```

```
SAVE LABEL1  
SAVE LABEL2  
SAVE LABEL3  
SAVE LABEL4  
SAVE LABEL5
```

```
DATA LABEL1  
& / 'Azimuth Angle'/
```

```
DATA LABEL2  
& / 'Fxr','Fyr',  
& 'Fzr','Mxr','Myr','Mzr'/
```

```
DATA LABEL3  
& / 'cos(0*PSI)', 'cos(PSI)',  
& 'cos(2.*PSI)', 'cos(3.*PSI)',  
& 'cos(4.*PSI)', 'cos(5.*PSI)'/
```

```
DATA LABEL4  
& / 'sin(0.*PSI)', 'sin(PSI)', 'sin(2.*PSI)',  
& 'sin(3.*PSI)', 'sin(4.*PSI)', 'sin(5.*PSI)'/
```

```
DATA LABEL5  
& / 'Fyh', 'Fyh', 'Fzh',  
& 'Mxh', 'Myh', 'Mzh'/
```

```
4000 FORMAT (/ , 1X, A, 6( 1X, A))
```

```
4200 FORMAT (/ , 2X, F10.3, 6(1X, F10.3))
```

```
4300 FORMAT (//)
```

```
4400 FORMAT (/ , 17X, 6(1X, A))
```

```
4500 FORMAT (/ , 1X, A , 6(1X, F10.3))
```

```
SBETA0 = SIN(BETA0)  
CBETA0 = COS(BETA0)
```

```
PSI1 = 0.  
PSIN = 360.*.017453293
```

```
FACTOR = FLOAT(IPR)/180.
```

```

PRINT *, 'Read in name of shaft loads output file < '
READ *, SHFOUT
PRINT *, SHFOUT
OPEN( 10, FILE = SHFOUT, STATUS = 'UNKNOWN')

```

```

NSTN1 = NSTN - 1
DO 360 IST = 1, NSTN1

```

```

F(1,1,IST) = RESLTS(IST,1,5)*CTHP + RESLTS(IST,1,6)*STHP

```

```

F(2,1,IST) = RESLTS(IST,1,6)*CBETA0*CTHP +
& RESLTS(IST,1,4)*SBETA0 - RESLTS(IST,1,5)*CBETA0*STHP

```

```

F(3,1,IST) = RESLTS(IST,1,5)*SBETA0*STHP -
& RESLTS(IST,1,6)*SBETA0*CTHP + RESLTS(IST,1,4)*CBETA0

```

```

F(4,1,IST) = RESLTS(IST,1,7)*CTHP + RESLTS(IST,1,8)*STHP

```

```

F(5,1,IST) = RESLTS(IST,1,8)*CBETA0*CTHP
& + RESLTS(IST,1,9)*SBETA0
& -RESLTS(IST,1,7)*CBETA0*STHP

```

```

F(6,1,IST) = RESLTS(IST,1,7)*SBETA0*STHP -
& RESLTS(IST,1,8)*SBETA0*CTHP + RESLTS(IST,1,9)*CBETA0

```

```

360 CONTINUE

```

```

DO 370 I = 1,6
F(I,1,NSTN) = F(I,1,1)

```

```

370 CONTINUE

```

```

NST1 = INT(NSTN1/2. + .0001)

```

```

DO 390 IST = 1, NSTN1

```

```

IST2 = IST + NST1

```

```

IF( IST2 .GE. NSTN) THEN
IST2 = IST - NST1
ENDIF

```

```

DO 385 I = 1,6

```

```

F(I,2,IST2) = F(I,1,IST)

```

```

385 CONTINUE

```

```

390 CONTINUE

```

```

DO 400 I = 1,6

```

```

F(I,2,NSTN) = F(I,2,1)

```

```

400 CONTINUE

```

```

DO 500 IST = 1, NSTN

```

```

PSI = IPS(IST) * .017453293

```

```

SHFLOD(1,IST) = F(1,1,IST) - F(1,2,IST)

```

```

SHFLOD(2,IST) = F(2,1,IST) + F(2,2,IST)

```

```

SHFLOD(3,IST) = F(3,1,IST) - F(3,2,IST)

```

```

SHFLOD(4,IST) = F(4,1,IST) - F(4,2,IST)
& - HUBRAD*( F(2,1,IST) - F(2,2,IST))
& - UNDSL*SHFLOD(3,IST)

```

```

SHFLOD(5,IST) = F(5,1,IST) + F(5,2,IST)
& + HUBRAD*( F(1,1,IST) + F(1,2,IST))

```



```

SHFL0D(6,IST) = F(6,1,IST) - F(6,2,IST)
&   + UNDSL0*SHFL0D(1,IST)

HUBL0D(1,IST) = SHFL0D(1,IST)*COS(P5I) + SHFL0D(3,IST)
&   * SIN(P5I)

HUBL0D(2,IST) = SHFL0D(2,IST)

HUBL0D(3,IST) = SHFL0D(3,IST)*COS(P5I) - SHFL0D(1,IST)
&   * SIN(P5I)

HUBL0D(4,IST) = SHFL0D(4,IST)*COS(P5I) + SHFL0D(6,IST)
&   * SIN(P5I)

HUBL0D(5,IST) = SHFL0D(5,IST)

HUBL0D(6,IST) = SHFL0D(6,IST)*COS(P5I) -
&   SHFL0D(4,IST)*SIN(P5I)

```

500 CONTINUE

```
WRITE(10,*) 'Low-Speed Shaft Loads in Rotor coordinates'
```

```
WRITE(10,4000) LABEL1, ( LABEL2(J), J=1,6)
```

```
DO 420 IST = 1, NSTN
```

```
WRITE(10,4200) IPS(IST), (SHFL0D(J,IST), J=1,6)
```

420 CONTINUE

```
WRITE(10,4300)
```

```
WRITE(10,*) 'Hub loads in fixed frame hub coordinates'
```

```
WRITE(10,4000) LABEL1, ( LABEL5(J), J=1,6)
```

```
DO 430 IST = 1, NSTN
```

```
WRITE(10,4200) IPS(IST), (HUBL0D(J,IST), J=1,6)
```

430 CONTINUE

- C Compute harmonics of rotor shaft loads: Fourier Series expansion.
C Also compute harmonics of fixed frame hub loads, in hub coordinates.

```
DO 800 II = 1,6
DO 700 I = 0,5
```

```
SUMA2 = 0.
SUMB2 = 0.
SUMC2 = 0.
SUMD2 = 0.
```

```
FI = FLOAT(I)
```

```
DO 650 IST = 2, NSTN1
```

```
PSI = IPS(IST) * .017453293
```

```
SUMA2 = SUMA2 + COS(FI*PSI)*SHFL0D(II,IST)
```

```
SUMB2 = SUMB2 + SIN(FI*PSI)*SHFL0D(II,IST)
```

```
SUMC2 = SUMC2 + COS(FI*PSI)*HUBL0D(II,IST)
```

```
SUMD2 = SUMD2 + SIN(FI*PSI)*HUBL0D(II,IST)
```

650 CONTINUE

```

SUMA = 2.*SUMA2 + COS(FI*PSI1)*SHFL0D(II,1) +
&   COS(FI*PSIN)*SHFL0D(II,NSTN)
SUMB = 2.*SUMB2 + SIN(FI*PSI1)*SHFL0D(II,1) +
&   SIN(FI*PSIN)*SHFL0D(II,NSTN)
SUMC = 2.*SUMC2 + COS(FI*PSI1)*HUBL0D(II,1) +
&   COS(FI*PSIN)*HUBL0D(II,NSTN)
SUMD = 2.*SUMD2 + SIN(FI*PSI1)*HUBL0D(II,1) +
&   SIN(FI*PSIN)*HUBL0D(II,NSTN)

```

```
IF( I .EQ. 0) THEN
```

```
A(I,II) = 0.25 * FACTOR*SUMA
```

```
B(I,II) = 0.25 * FACTOR*SUMB
```

```
C(I,II) = 0.25 * FACTOR*SUMC
```

```
D(I,II) = 0.25 * FACTOR*SUMD
```

```
ELSE
```

Appendix E

Modules 1 and 2 Flowcharts

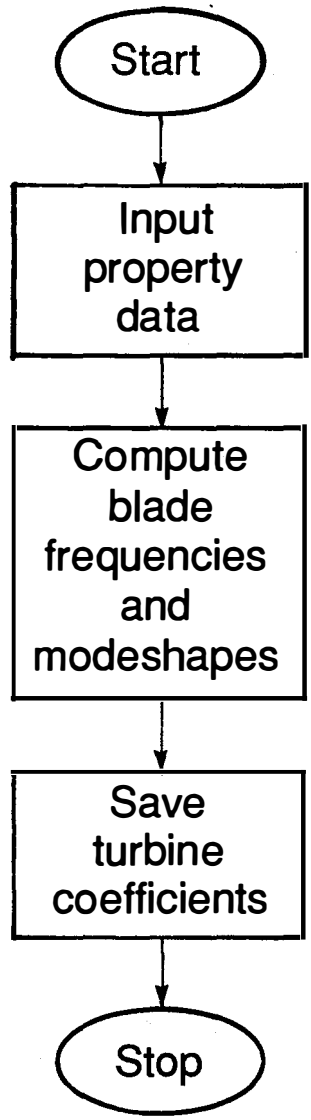


Figure E-1. Module 1 Flowchart

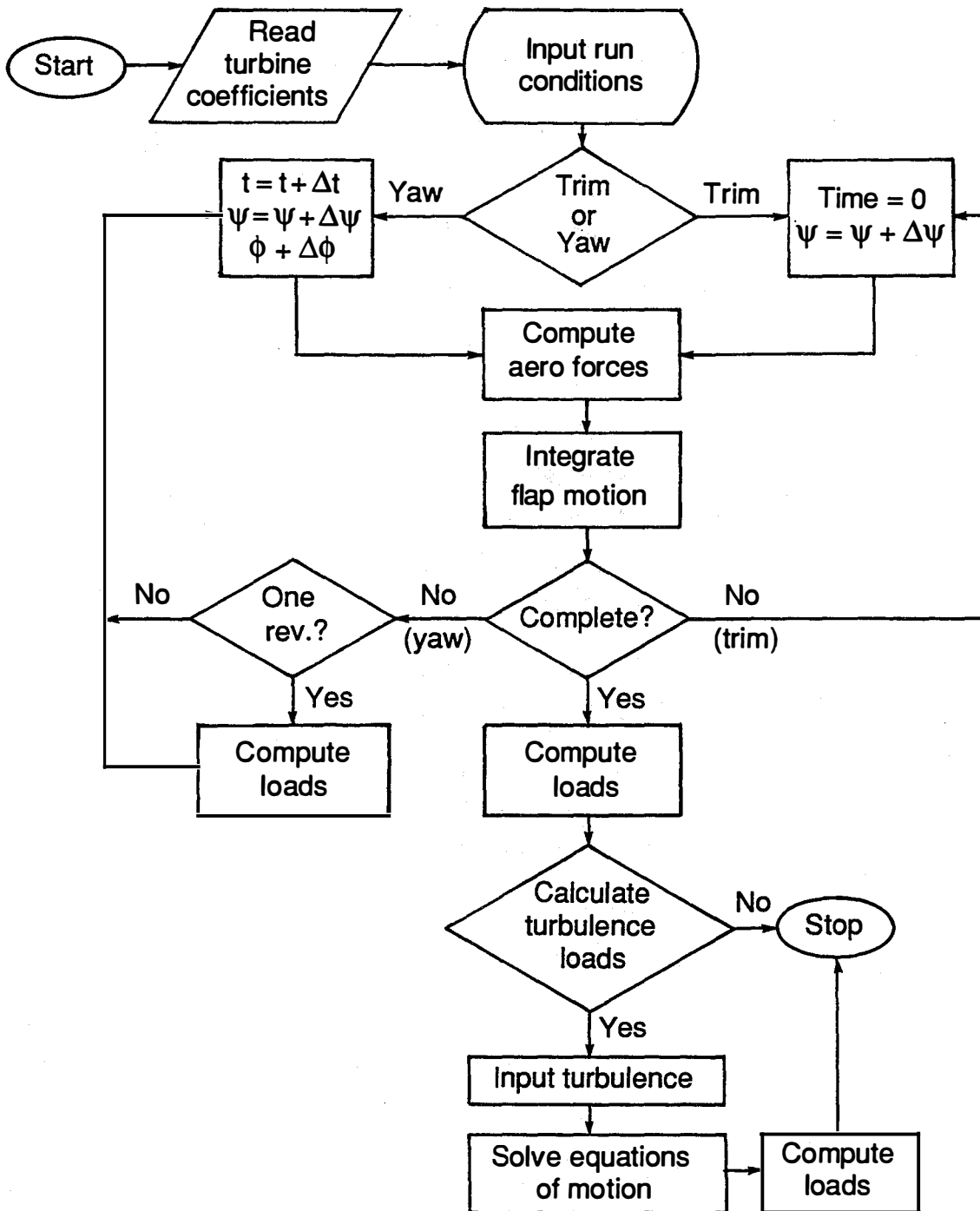


Figure E-2. Module 2 Flowchart

Document Control Page	1. NREL Report No. NREL/TP-257-4671	2. NTIS Accession No. DE92010580	3. Recipient's Accession No.
4. Title and Subtitle User's Guide for the NREL Teetering Rotor Analysis Program (STRAP)		5. Publication Date August 1992	
		6.	
7. Author(s) Alan D. Wright		8. Performing Organization Rept. No. NREL/TP-257-4671	
9. Performing Organization Name and Address National Renewable Energy Laboratory 1617 Cole Blvd. Golden, CO 80401		10. Project/Task/Work Unit No. WE21.8202	
		11. Contract (C) or Grant (G) No. (C) (G)	
12. Sponsoring Organization Name and Address National Renewable Energy Laboratory 1617 Cole Blvd. Golden, CO 80401		13. Type of Report & Period Covered Technical Report	
		14.	
15. Supplementary Notes			
16. Abstract (Limit: 200 words) This Report gives the reader an overview of and instructions on the proper use of the National Renewable Energy Laboratory (formerly the Solar Energy Research Institute, or SERI) Teetering Rotor Analysis Program (STRAP version 2.2). STRAP is a derivative of the Force and Loads Analysis Program (FLAP). It is intended as a tool for prediction of rotor and blade loads and response for only two-bladed teetering hub wind turbines. The effects of delta-3, undersling, hub mass, and wind turbulence are accounted for. The objectives of this report are to give an overview of the code and also show the methods of data input and correct code execution steps in order to model an example two-bladed teetering hub turbine. A large portion of the discussion (Sections 6.0, 7.0, and 8.0) is devoted to the subject of inputting and running the code for wind turbulence effects. The ability to include turbulent wind effects is perhaps the biggest change in the code since the release of FLAP version 2.01 in 1988. This report is intended as a user's guide. It does not contain a theoretical discussion on equations of motion, assumptions, underlying theory, etc.			
17. Document Analysis a. Descriptors wind turbines; loads; structures; computer codes b. Identifiers/Open-Ended Terms c. UC Categories 261			
18. Availability Statement National Technical Information Service U.S. Department of Commerce 5285 Port Royal Road Springfield, VA 22161		19. No. of Pages 214	
		20. Price A10	